JOURNAL OF COMPUTATIONAL PHYSICS **132,** 108–122 (1997)
ARTICLE NO. CP965626

Fast Adaptive 2D Vortex Methods

John Strain[1]

Department of Mathematics, University of California, Berkeley, California 94720
E-mail: strain@math.berkeley.edu

Received March 11, 1996; revised August 22, 1996

We present a new approach to vortex methods for the 2D Euler equations. We obtain long-time high-order accuracy at almost optimal cost by using three tools: fast adaptive quadrature rules, a free-Lagrangian formulation, and a useful new analysis of the consistency error. Our error analysis halves the order of differentiability required of the flow and suggests an efficient new balance of smoothing parameters which works well with fast summation schemes. Numerical experiments with our methods confirm our theoretical predictions and display excellent long-time accuracy.

1. INTRODUCTION

Vortex methods solve the 2D incompressible Euler equations in the vorticity formulation by discretizing the Biot–Savart law with the aid of the flow map. They have been extensively studied, widely generalized, and successfully applied to complex high-Reynolds-number flows. See [12] for a survey.

Vortex methods involve several subsidiary algorithms: velocity evaluation, vortex motion, diffusion, boundary conditions, regridding, and fast summation. In this paper, we improve the speed, accuracy, and robustness of the velocity evaluation. We eliminate the flow map, improve the quadrature used for the Biot–Savart law, and analyze the error in a straightforward but new way, requiring less differentiability of the flow and obtaining efficient new parameter balances. We employ standard ODE solvers for the vortex motion and consider inviscid free-space flow so diffusion and boundary conditions play no role. Our approach combines naturally with regridding and fast summation methods.

In Section 2, we review Lagrangian vortex methods. These move the nodes of a fixed quadrature rule with the computed fluid velocity, preserving the weights of the rule by incompressibility. This procedure loses accuracy when the flow becomes disorganized [5, 18], motivating many regridding techniques [17]. Even before the flow becomes disorganized, however, obtaining high-order accuracy with a single quadrature rule requires smoothing of the singular Biot–Savart kernel. Smoothing gives high-order accuracy for short times but slows down fast velocity evaluation techniques and halves the order of accuracy of the method relative to the order of differentiability of the flow.

In Section 3, we contrast two free-Lagrangian vortex methods, the triangulated vortex method of [20], and the quadrature-based method of [23]. The triangulated vortex method is robust, practical, and efficient but limited to second-order accuracy. The quadrature-based method computes adaptive quadratures tailored to the Biot–Savart kernel at each time step, yielding long-time high-order accuracy at asymptotically optimal cost.

The present paper develops a free-Lagrangian method which couples kernel smoothing with adaptive quadrature rules *not* tailored to the Biot–Savart kernel, producing long-time high-order accuracy. The asymptotic slowdown normally produced by kernel smoothing is almost eliminated by a careful choice of smoothing functions and parameters, based on a straightforward but new error analysis of the velocity evaluation. This analysis requires about half as many derivatives of the solution as the standard approach.

The structure of our method is standard: At each time step, the smoothed velocity is evaluated once and the vortices are moved with an explicit Runge–Kutta or multistep method. The velocity evaluation involves three steps: First, a data structure groups the N vortices into cells convenient for integration. Then a global order-q quadrature rule is built. Finally, the fast multipole method is used with this rule to evaluate the smoothed velocity field. The details are presented in Section 4.

Section 5 presents numerical experiments. The error is measured for standard test problems and our theoretical predictions are fully verified. Then more complex flows are computed.

2. LAGRANGIAN VORTEX METHODS

This section gives an overview of 2D vortex methods for incompressible inviscid flow. First, we describe how the

[1] Research supported by a NSF Young Investigator Award, Air Force Office of Scientific Research Grant FDF49620-93-1-0053, and the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

0021-9991/97 $25.00

2D Euler equations reduce to an infinite system of ordinary differential equations for the flow map. This formulation leads naturally to many different vortex methods. We contrast the Lagrangian and free-Lagrangian viewpoints, then review the well-developed convergence theory of Lagrangian methods.

Second, we explore avenues for improvement. We explain the conflict between smoothing for high-order accuracy and fast summation for speed, and demonstrate the Perlman effect in which the increasing derivatives of the evolving flow map interfere with the quadrature error bound.

2.1. *Equations of Motion*

The 2D incompressible Euler equations

$$\begin{aligned} \dot{u} + uu_x + vu_y + p_x/\rho &= 0 \\ \dot{v} + uv_x + vv_y + p_y/\rho &= 0 \\ u_x + v_y &= 0 \end{aligned}$$

involve the fluid velocity $u(z, t) = (u, v)$ where $z = (x, y)$, the pressure $p(z, t)$, and the constant density ρ. Taking the 2D curl eliminates the pressure, giving the vorticity equation

$$\dot{\omega} + u\omega_x + v\omega_y = 0$$

for the vorticity $\omega = v_x - u_y$. Let $z \mapsto \Phi(z, t)$ be the flow map, defined by

$$\dot{\Phi}(z, t) = u(\Phi(z, t), t), \qquad \Phi(z, 0) = z. \tag{1}$$

Then vorticity is conserved along particle paths,

$$\omega(\Phi(z, t), t) = \omega(z, 0). \tag{2}$$

Eqs. (1) and (2) for Φ and ω involve the unknown u as well. To close them, we solve the elliptic system

$$\begin{aligned} v_x - u_y &= \omega, \\ u_x + v_y &= 0 \end{aligned}$$

for the velocity (u, v). When ω has compact support, the solution u is given by the Biot–Savart law

$$u(z, t) = \int K(z - z')\omega(z')\, dx'\, dy', \tag{3}$$

where K is the Biot–Savart kernel

$$K(z) = \frac{z^\perp}{2\pi r^2}, \qquad z^\perp = (-y, x), \quad r^2 = x^2 + y^2. \tag{4}$$

Thus we have a closed system for Φ and ω alone, the "free-Lagrangian" equations of motion consisting of the vorticity transport law (2) coupled with

$$\dot{\Phi}(z, t) = \int K(\Phi(z, t) - z')\omega(z', t)\, dx'\, dy'. \tag{5}$$

The "Lagrangian" equation of motion is derived by changing variables $z' \leftarrow \Phi(z', t)$ in Eq. (5). The Jacobian of this change of variables is unity because the flow is incompressible, so this gives a closed equation for Φ alone:

$$\dot{\Phi}(z, t) = \int K(\Phi(z, t) - \Phi(z', t))\omega(z', 0)\, dx'\, dy'. \tag{6}$$

This requires values of ω only at time $t = 0$, and is the usual starting point for vortex methods.

2.2. *Lagrangian Vortex Methods*

Lagrangian vortex methods now discretize Eq. (6), moving N points $z_j(t) \approx \Phi(z_j, t)$ with the fluid velocity, starting at $t = 0$ from the nodes z_j of a quadrature formula with weights w_j. A typical order-q quadrature formula

$$\int g(z)\, dx\, dy = \sum_{j=1}^{N} w_j g(z_j) + E_N(g)$$

has an error bound

$$|E_N(g)| \le Ch^q \|g\|_q \tag{7}$$

for C^q integrands g. Here h is the mesh size of the rule and the C^q norm of g is given by

$$\|g\|_0 = \max_z |g(z)|, \qquad \|g\|_q = \|g\|_0 + \sum_{\alpha+\beta=q} \|\partial_x^\alpha \partial_y^\beta g\|_0.$$

Applying this quadrature formula to the Lagrangian equation of motion (6) gives a system of N ordinary differential equations:

$$\dot{z}_i(t) = \sum_{j \ne i} w_k K(z_i(t) - z_j(t))\omega(z_j, 0).$$

The quadrature error bound (7) is infinite since K is unbounded, so we replace K by the smoothed kernel

$$K_\delta(z) = \varphi_\delta * K(z),$$

where $*$ denotes convolution,

$$\varphi_\delta(z) = \delta^{-2}\varphi(r/\delta)$$

and φ is an appropriate radial "core function." Almost all modern vortex methods use such a smoothing [9], often with φ and the "core radius" δ chosen to give high-order convergence as the mesh size h vanishes [14]. High-order convergence can be guaranteed by the following conditions on φ and ω:

$$\int \varphi = 1,$$

$$\int x^\alpha y^\beta \varphi = 0, \qquad 1 \le \alpha + \beta \le m - 1, \tag{8}$$

$$\int |z|^m |\varphi| < \infty$$

$$\varphi \in C^L \qquad \text{and} \qquad \varphi(z) = 0 \quad \text{for} \quad |z| \ge 1, \tag{9}$$

$$\omega \in C^M \qquad \text{has compact support.} \tag{10}$$

High-order accuracy requires smooth solutions, so requiring ω to be C^M is natural. Compact support in condition (9) can be weakened, but it is important for efficiency. Given these conditions, a typical convergence theorem follows.

THEOREM 1 [1]. *Assume conditions* (8) *through* (10) *are satisfied with* $L \ge 3$, $M \ge \max(L + 1, m + 2)$ *and* $m \ge 4$. *Let* $\delta = ch^a$ *where* $0 < a < 1$. *Suppose* L *is large enough to satisfy*

$$L > \frac{(m-1)a}{1-a}.$$

Then the computed flow map $\Phi_{h,\delta}$ *satisfies*

$$\|\Phi - \Phi_{h,\delta}\|_h \le O(h^{ma})$$

as h *and* δ *go to zero.*

Here the discrete 2-norm is given by

$$\|g\|_h = \left(h^2 \sum_i |g(z_i)|^2 \right)^{1/2},$$

where the z_i are the initial vortex positions, and similar bounds hold for the computed velocity and vorticity.

This theorem allows a close to 1 and δ close to $O(h)$ only for very smooth flows, where L and M are large. For general flows, Hald [13] and Nordmark [17] have shown that $\delta = O(\sqrt{h})$ is a good choice. Then $2m$ derivatives of ω guarantee only $O(h^m)$ accuracy. In Section 4, we derive methods requiring only $m + 1$ derivatives at the cost of redefining convergence.

2.3. *Cost and Accuracy*

Convergence theory must be augmented by practical considerations: How much does a method cost to achieve a specified accuracy?

Since there are N vortices and each velocity value is a sum

$$u_{h,\delta}(z_i) = \sum_{j=1}^{N} K_\delta(z_i - z_j) w_j \omega(z_j, 0),$$

a directly velocity evaluation costs $O(N^2)$ work. This is extremely expensive if the flow is complex, since many vortices are required. The expense has been reduced by fast summation schemes such as the method of local corrections [2], the fast multipole method [6], and Ewald summation [21]. These schemes evaluate unsmoothed ($\delta = 0$) sums like

$$u(z_i) = \sum_{j=1}^{N} K(z_i - z_j) w_j$$

to accuracy ε in $O(N \log \varepsilon)$ work, by separating local from global interactions and applying separation of variables globally. They run much faster than direct evaluation when N is large.

Consider the fast multipole method, for example. The essential observation is that the sum of fields $K(z - z_j)w_j$ due to *all* vortices outside a disk D with center c can be formed into a single Taylor expansion about c,

$$\sum_{z_j \notin D} K(z - z_j) w_j = \sum_{\alpha=0}^{p} a_\alpha (z - c)^\alpha + O(\varepsilon),$$

where the truncation error ε is uniformly small for z in any smaller disk $D' \subset D$. These Taylor expansions can be formed and evaluated at all the vortices z_i in $O(N)$ CPU time, leaving only the nearby "local" interactions between z_i and $z_j \in D$ requiring direct evaluation. However, the expansion technique works only for the exact Biot–Savart kernel, so the disk D can never be smaller than the smoothing radius δ inside which K is replaced by $K_\delta \ne K$.

Thus fast summation methods cannot speed up the smoothed interaction $K_\delta(z_i - z_j)$ between vortices z_i and z_j closer than δ. Asymptotically as $N \to \infty$, there are $O(N\delta^2)$ vortices in a circle of radius δ, so if $\delta = O(\sqrt{h})$ there are a total of

$$O(N^2\delta^2) = O(N^2 h) = O(N^{3/2})$$

local interactions to be evaluated directly. Thus fast summation schemes slow down from $O(N)$ to $O(N^{3/2})$ when K is smoothed with $\delta = O(\sqrt{h})$.

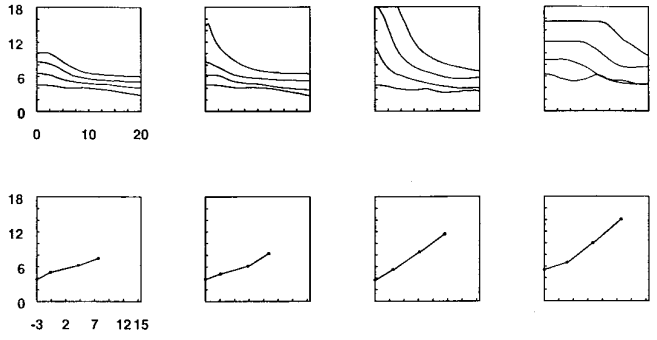

FIG. 1. Correct bits in velocity in L^1 norm for a fourth-order vortex method, plotted vs. time (top) and bits in CPU time (bottom), for smoothing radii $\delta = C\sqrt{h}$ where $C = 1/4$, 1/2, 1, and 2 (left to right).

Hence there is a conflict between smoothing and fast summation. If we take δ close to $O(h)$ to speed up fast summation, we need many derivatives of the flow for a modest order of convergence. Larger δ is more accurate for rougher flows, but slows down fast summation schemes. In Section 4 we resolve this conflict by deriving an error bound which allows us to use much smaller δ.

2.4. *The Perlman Effect*

A completely different obstacle to accurate calculations with vortex methods is the "Perlman effect." Since the error bound for numerical quadrature in Eq. (7) depends on order-q derivatives of the integrand, here

$$g(z') = K_\delta(\Phi(z,t) - \Phi(z',t))\omega(z',0),$$

the higher derivatives of the flow map will affect the error bound. For almost any interesting flow, the flow map moves fluid particles far apart and therefore develops large derivatives when the flow becomes disorganized. Thus vortex methods lose high-order accuracy during long-time calculations [5, 18].

This loss of accuracy is exhibited in Fig. 1, where correct bits in the velocity u, measured in the discrete L^1 norm, are plotted vs. time t for $0 \le t \le 20$ (top row) and vs. the number of bits in total CPU time (bottom row). The vorticity is a standard test case $\omega_7(r) = \max(0, 1-r^2)^7$, discussed fully in Section 5.

Four runs were made with $N = 44, 164, 640, 2512$ vortices in an equidistant grid on the support ($|z| \le 1$) of the vorticity. The ODEs were solved with fourth-order Adams, as described in subsection 4.7, with 20, 40, 80, 160 time steps from $t = 0$ to $t = 20$. Nordmark's fourth-order kernel (see Subsection 4.6) was used, with smoothing radius $\delta = C\sqrt{h}$, where the constant C increases from 1/4 to 2 from left to right in the figure. For each fixed C, the method is fourth-order accurate in h and the time step, and costs $O(N^{5/2})$ to run from $t = 0$ to $t = 20$.

The top row, which plots correct bits vs. time, clearly shows the rapid degradation of initially excellent high-order accuracy. This is striking because the velocity and vorticity of this test case are independent of time. The plots also show how the error varies as the smoothing constant C is increased. Clearly the value $C = 1$ produces initially high accuracy which degrades rapidly in time, while other values give larger but more uniform errors. The plots of correct bits vs. bits of CPU time in the second row of the figure show that the cost of the computation increases by an order of magnitude as C increases from 1/4 to 2 with $N = 2512$. This is entirely due to the expense of evaluating the smoothed kernel since fast summation was not employed. The plots all share the same vertical and horizontal scales, and the tick marks are placed so that in theory we should advance one tick mark along each axis each time N is doubled. In practice, the cost is larger and the error decreases about half as fast as theory predicts.

The Perlman effect has motivated much research on regridding, the idea being to avoid large derivatives of the flow map by restarting before the flow becomes disorganized [17]. Similarly, Beale has developed an iterative reweighting scheme to overcome the Perlman effect [4]. The free-Lagrangian vortex methods we discuss next constitute a general approach to avoiding the Perlman effect.

3. FREE-LAGRANGIAN METHODS

Free-Lagrangian methods overcome the Perlman effect by removing the flow map from the Biot–Savart integral. Thus

$$\dot\Phi = \int K(\Phi - z')\omega(z',t)\,dx'\,dy',$$

replaces the Lagrangian equation of motion (6). Since ω values are known only at the moving points $z_j(t)$, each velocity evaluation requires a fresh set of quadrature weights adapted to the current vortex positions. Two such methods are discussed below.

3.1. *Triangulated Methods*

Triangulated vortex methods evolve points $z_j(t)$ by

$$\dot z_j(t) = u_h(z_j,t) = \int K(z_j(t) - z')\omega_h(z',t)\,dx'\,dy', \quad (11)$$

where ω_h is a piecewise linear interpolant to the vorticity values

$$\omega_h(z_j(t),t) = \omega_h(z_j,0) = \omega(z_j,0)$$

and the nodes $z_j(t)$ form the vertices of a triangulation of $\mathbf{R}^2$.

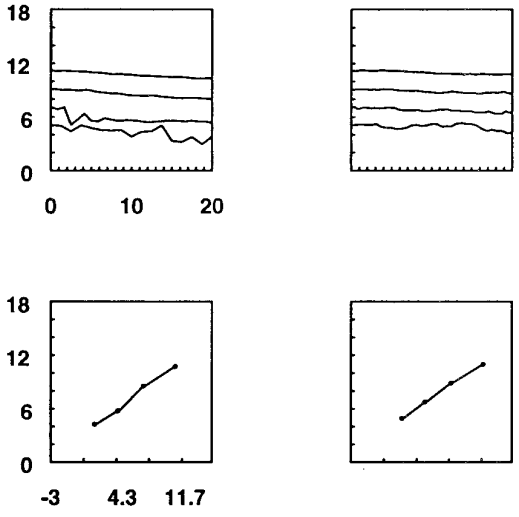

FIG. 2. Correct bits in velocity in L^1 norm for the triangulated vortex method, plotted vs. time (top) and CPU time (bottom). The right set of plots used twice as many time steps.

Given any piecewise linear function ω_h on a triangulation of $\mathbf{R}^2$, one can evaluate u_h exactly, with results depending strongly on the triangulation. In [7], this observation was combined with a fixed triangulation carried by the flow. While theoretically convergent, the resulting scheme costs $O(N^2)$ work per time step with a large constant and loses accuracy quickly and severely because the triangulation degenerates.

We developed practical triangulated vortex methods in [20]: a simple fast summation scheme brought the cost down to $O(N^{4/3})$ with a reasonable constant, and a fast Delaunay triangulation scheme gave excellent long-time accuracy. An adaptive initial triangulation technique made the method robust enough to compute even discontinuous patches of vorticity, a difficult task for a method of this generality. Figure 2 shows results for the standard test case used in Fig. 1, with $N = 70$, 225, 745, and 2729 vertices on the support of ω. Two runs were made, the first one (left) with 16, 24, 32, 48 steps of fourth-order Adams from $t = 0$ to $t = 20$, and the second one (right) with twice as many time steps. The error displays no Perlman effect; second-order accuracy (one horizontal and vertical tick each time N is increased) is maintained uniformly in time. The triangulated approach is now being applied to flows in three dimensions with viscosity and boundaries [11, 16]. However, it seems difficult to make a triangulated vortex method with higher than second-order accuracy. This motivated the next approach we discuss.

3.2. *Quadrature-based Methods*

We developed higher-order free-Lagrangian methods in [23]. The basic idea is to construct time-dependent quadrature weights $w_{ij}(t)$ which give high-order accuracy without smoothing the Biot–Savart kernel:

$$u(z_i, t) = \int K(z_i - z')\omega(z', t)\, dx'\, dy'$$
$$\approx \sum_{j=1}^{N} w_{ij}(t)\, K(z_i - z_j)\omega(z_j, t).$$

For example, high-order product integration weights [10] make smoothing unnecessary, but the i-dependence of $w_{ij}(t)$ precludes fast summation methods. To take advantage of fast summation, we construct w_{ij} with the "locally-corrected property" that

$$w_{ij} = w_j \qquad \text{for almost all } j$$

for each point of evaluation z_i and some "smooth" quadrature rule with points z_j and weights w_j. Such rules can be built and the velocity evaluated in $O(N \log^2 N)$ work per time step. The price for efficiency, however, is a redefinition of convergence. The error bound for these quadratures is $O(\varepsilon + h^q)$, where ε is an arbitrary user-specified error tolerance and the constant in the $O(N)$ cost depends weakly on ε. Thus one gets order-q convergence only down to $O(\varepsilon)$. This is sufficient for three reasons: computer arithmetic operates with finite precision, practical computations can afford rather low accuracy for the most part, and fast summation methods introduce an $O(\varepsilon)$ error as well. High-order accuracy can be maintained for long times, though these rules are computationally expensive. The constant in the $O(N \log^2 N)$ cost per step is less favorable, making these methods less practically useful than the triangulated method so far.

4. A NEW APPROACH

We now present a new fast adaptive vortex method which aims to avoid obstacles both to speed and to accuracy. The key ingredients are

- A free-Lagrangian formulation to avoid the Perlman effect.
- Adaptive quadrature rules with high-order accuracy on smooth functions, but *not* tailored to the singularity of the Biot–Savart kernel.
- Simple and useful new consistency error bounds requiring fewer derivatives of the vorticity and leading to an efficient new smoothing strategy.

These ingredients combine to give a method with almost optimal efficiency and long-time high-order accuracy without excessive smoothness requirements on the solution.

4.1. *Overview*

We begin with quadrature. Given N arbitrary nodes $z_j \in \mathbf{R}^2$, we construct the weights of a quadrature rule having order-q accuracy on C^q functions if the nodes are well distributed. Note that without some restriction on the asymptotic distribution of nodes, no guarantee of order-q accuracy is possible. Thus we construct rules with an error bound composed of two factors. The first depends only on the node locations and is easily computable a posteriori, as a monitor for bad node distributions. The second depends only on the mesh size and the C^q norm of the integrand.

Our quadratures are composite. After partitioning the nodes into rectangular cells in Subsection 4.2, we construct order-q rules on each cell in Subsection 4.3. The union of these rules is globally accurate of order q. We quote a useful error bound from [22] in Subsection 4.4.

After quadrature, we analyze smoothing. Subsection 4.5 presents a standard smoothing error bound. In Subsection 4.6, we construct a family of arbitrary-order core functions and shape factors.

Subsection 4.7 presents a multistep time stepping procedure, and discusses the starting value problem. Finally, Subsection 4.8 combines the results of Subsections 4.4 and 4.5 to obtain an extremely useful error analysis of velocity evaluation which requires fewer derivatives of the vorticity than the usual approach and leads to an efficient new balance between quadrature and smoothing. This new balance permits smaller smoothing radii δ and therefore speeds up fast summation by reducing the number of local interactions.

4.2. *Data Structures*

Let $B = [a, b] \times [c, d]$ be a rectangle containing the nodes z_j. Composite quadrature partitions B into a union of rectangular cells B_i, each containing enough nodes to construct an order-q quadrature. There are $m := q(q+1)/2$ monomials $x^\alpha y^\beta$ of degree $\alpha + \beta \le q - 1$, so we will need at least $p \ge m$ nodes per cell. Thus we partition B into cells, each containing p or $p+1$ nodes. (Some cells must have $p+1$ if p does not divide N exactly.) This is conveniently done via the following tree structure.

Let $B = B_1$ be the level-0 root of the tree. Divide B_1 in half along its longest edge, with the dividing plane located so that each half of B_1 contains either $\lfloor N/2 \rfloor$ or $\lfloor N/2 \rfloor + 1$ nodes. This gives the level-1 cells B_2 and B_3. Recursively, split B_2 and B_3 along their longest edges to get B_4 through B_7, each containing $\lfloor N/4 \rfloor$ or $\lfloor N/4 \rfloor + 1$ nodes z_j. Repeat this procedure L times to get $M = 2^L$ cells B_i on the finest level L, numbered from $i = M$ to $i = 2M - 1$, each containing $p = \lfloor N/M \rfloor$ or $p+1$ nodes z_j. The union of all the cells on any given level is B. The tree structure is stored by listing the boundaries of each cell $B_i = [a_i, b_i] \times [c_i, d_i]$ from $i = 1$ to $i = 2M - 1$, a total of $4 \cdot 2M$ numbers, and indexing the nodes into a list so that the nodes $z_j \in B_i$ are given by $j = j(s)$ for $s = b(i), \ldots, e(i)$ and three integer functions j, b, and e. This can be done in $O(N \log N)$, but the simplest method requires sorting each cell before each subdivision, giving a total cost $O(N \log^2 N)$ for the tree construction with an $O(N \log N)$ sorting method such as Heapsort. Figure 3 shows an example of this tree structure.

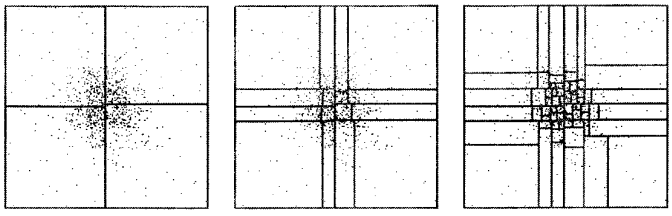

FIG. 3. Levels 1, 3, and 5 in the cell data structure with $N = 1000$ nonuniformly distributed points. Here each level-5 cell contains either 15 or 16 points, suitable for a quadrature rule of order $q = 4$ since $q(q+1)/2 = 10$.

4.3. *Quadrature Rules*

We now construct order-q quadrature rules on B with N given quadrature nodes z_j. Assume $N \ge m := q(q+1)/2$, and choose an integer $L \ge 0$ with $p := \lfloor N/2^L \rfloor \ge m$. The data structure just constructed divides B into $M = 2^L$ rectangular cells B_i, each containing either p or $p+1$ nodes z_j. We construct local weights W_j^i for nodes $z_j \in B_i$ by solving the following system of m linear equations in at least p unknowns:

$$\sum_{z_j \in B_i} P_\alpha(x_j) P_\beta(y_j) W_j^i = \int_{B_i} P_\alpha(x) P_\beta(y)\, dx\, dy = \delta_{\alpha 0}\delta_{\beta 0}|B_i|, \qquad 0 \le \alpha + \beta \le q - 1.$$

Here $|B_i| = (b_i - a_i)(d_i - c_i)$ is the area of B_i and

$$P_\alpha(x) = p_\alpha(t), \qquad x = x_m + t x_h,$$

where $p_\alpha(t)$ are the usual Legendre polynomials on $[-1, 1]$ and $x_m = (b_i + a_i)/2$, $x_h = (b_i - a_i)/2$, with similar expressions for the y variable. Since $p \ge m$, this system of m equations in at least p unknowns generically has solutions. We compute the solution W_j^i of least 2-norm, using a complete orthogonal factorization routine from LAPACK [3]. The weights of the rule W are then defined to be $W_j = W_j^i$ where $z_j \in B_i$. The algorithm is summarized in Fig. 4.

Remark. In most vortex methods, the vorticity ω is known only at the vortices, so interpolation is needed to

Quadrature Algorithm

Input:
  Nodes z_i, $1 \le i \le N$.
  Quadrature order q and safety factor S.
  Maximum permitted cell condition number: Ω_m.
Set parameters:
  Minimum degrees of freedom required per cell: $m = q(q+1)/2$.
  Top level in cell structure: $L = \lfloor \log_2(N/Sm) \rfloor$.
  Points per level-L cell: $p = \lfloor N/2^L \rfloor$.
Construct cell data structure:
  $B_1 = B$.
  do $l = 1, L-1$
    Divide level-l cells along longest edge with approximately
    half the points in each subcell, yielding level-$l+1$ cells.
  end do
  Result: 2^L cells B_i on level L with p or $p+1$ points each.
Compute weights W_i one cell at a time.
  do $i = 1, 2^L$
    Compute least-2-norm solution W of
    $\sum_{z_j \in B_i} W_j P_\alpha(x_j) P_\beta(y_j) = \delta_{\alpha 0}\delta_{\beta 0}|B_i|$ for $0 \le \alpha + \beta \le q-1$
    Compute cell condition number
    $\Omega^i = 1 + \frac{1}{|B_i|} \sum_{z_j \in B_i} |W_j|$.
    if $\Omega^i \ge \Omega_m$ then
      Merge cell B_i with its sibling, flag cell and sibling
      done, and recompute weights on level-$(L-1)$ cell B^I.
    end if
  end do

FIG. 4. Adaptive order-q quadrature algorithm with N points z_j in a rectangle B.

evaluate the vorticity elsewhere. The tree structure provides a natural interpolation technique. Suppose vortices z_j lie in a cell C and we want $\omega(z)$ for $z \in C$. We approximate $\omega(z)$ by

$$\omega(z) \approx \sum_{z_j \in C} \Omega_j(z)\omega(z_j),$$

where the interpolation weights $\Omega_j(z)$ form the least 2-norm solution of the underdetermined linear system

$$\sum_{z_j \in C} \Omega_j(z) P_\alpha(x_j) P_\beta(y_j) = P_\alpha(x) P_\beta(y), \qquad 0 \le \alpha + \beta \le q-1.$$

This gives a qth order interpolation formula on each cell. The weights are bounded if there are enough nodes z_j in C. To contour the computed vorticity in Subsection 5.3, we interpolated ω to a fine equidistant grid, then contoured on the grid.

4.4. *Quadrature Error Bounds*

The weights W_j now integrate all polynomials of degree less than q exactly over all level-L cells B_i. This property implies order-q accuracy:

THEOREM 2 [22]. *Let $B = \cup_{i=1}^M B_i$ where $B_i = [a_i, b_i] \times [c_i, d_i]$. Suppose that W integrates $x^\alpha y^\beta$ exactly over each B_i for $0 \le \alpha + \beta \le q-1$. Then for any C^q function g on B, the quadrature error*

$$E = \int_B g(z)\, dx\, dy - \sum_{j=1}^N W_j g(z_j) \tag{12}$$

satisfies the bound

$$|E| \le \Omega\, |B| \frac{h^q}{q!} \|g\|_q,$$

where $h = \max_i \max(b_i - a_i, d_i - c_i)$ is the longest cell edge,

$$\Omega = 1 + \frac{1}{|B|}\sum_{j=1}^{N} |W_j|$$

and $|B| = (b - a)(b - c)$ *is the area of* B.

In general, the condition number Ω cannot be bounded a priori for arbitrary points, but we can easily compute it a posteriori, yielding an excellent diagnostic for the quality of the rule.

Remark. By reducing each cell condition number $\Omega^i = 1 + (1/|B_i|)\ \Sigma_{z_j \in B_i} |W_j|$, we can reduce the global condition number $\Omega = \Sigma_i \Omega_i$. Increasing p reduces Ω, since the additional degrees of freedom can be applied to reducing the 2-norm of W_j^i, but it is too expensive to increase p globally. Thus we reduce Ω adaptively: when Ω^i exceeds a tolerance Ω_m, we merge B_i with its sibling in the tree structure, obtaining a cell B_I containing twice as many points z_j. We then recompute all weights W_j for which $z_j \in B_I$, reducing Ω^I at the cost of a larger linear system and a larger cell size h.

This adaptive technique also treats the degenerate cases when no solution exists on a cell B_i, because the points z_j are not in the sufficiently general position. A solution is more likely to exist after such a cell is merged with its sibling,

Remark. In practice, the choice of L may be difficult. L too small increases h, while L too large precludes order-q accuracy. Thus our code accepts a user-specified safety parameter $S \geq 1$ and chooses L so that each level-L cell contains at least $\lfloor Sq(q + 1)/2 \rfloor$ points. We always take $S = 2$.

Remark. The error estimates are given in terms of the maximum cell size only for theoretical convenience; the proof in [22] shows that the local derivatives of the integrand times the local cell size gives an equally valid error bound. Thus local increases in h are permitted if the integrand is either smooth or small where h is large.

4.5. *Smoothing Error Bounds*

Since convolution is associative, replacing K by K_δ is equivalent to smoothing the velocity field u with the core function φ. The following is a standard error bound for such smoothing.

THEOREM 3. [19]. *Assume the compactly supported core function* φ *satisfies the moment conditions*

$$\int \varphi = 1,$$

$$\int x^\alpha y^\beta \varphi = 0, \qquad 1 \leq \alpha + \beta \leq m - 1, \quad (13)$$

$$M = \frac{1}{m!}\int |z|^m |\varphi| < \infty.$$

Suppose u *belongs to the Sobolev space* $W^{m,p}$ *of functions with* m *distributional derivatives in* L^p, *where* $1 \leq p \leq \infty$ *and* $m \geq 2$. *Then*

$$\|\varphi_\delta * u - u\|_{L^p} \leq M\delta^m \sum_{\alpha+\beta=m} \|\partial_x^\alpha \partial_y^\beta u\|_{L^p}.$$

Proof. Suppose by density that u is smooth and Taylor expand:

$$u(z' - z) = u(z') + \sum_{l=1}^{m-1} \frac{(-1)^l}{l!} \sum_{\alpha+\beta=l} \partial_x^\alpha \partial_y^\beta u(z') x^\alpha y^\beta - \int_0^1 \frac{(t-1)^{m-1}}{(m-1)!} \sum_{\alpha+\beta=m} \partial_x^\alpha \partial_y^\beta u(z' - tz) x^\alpha y^\beta \, dt.$$

Multiply by $\varphi_\delta(z)$, integrate, and use the moment conditions (13):

$$\varphi_\delta * u(z') - u(z') = -\int_0^1 \frac{(t-1)^{m-1}}{(m-1)!} \sum_{\alpha+\beta=m} \int \partial_x^\alpha \partial_y^\beta u(z' - tz) x^\alpha y^\beta \, \varphi_\delta(z) \, dx \, dy \, dt.$$

Take L^p norms and use the fact that the norm of an integral is less than or equal to the integral of the norm:

$$\|\varphi_\delta * u - u\|_{L^p} \leq \int_0^1 \frac{|t-1|^{m-1}}{(m-1)!} \sum_{\alpha+\beta=m} \int \|\partial_x^\alpha \partial_y^\beta u(\cdot - tz)\|_{L^p} |x^\alpha y^\beta| \, |\varphi_\delta(z)| \, dx \, dy \, dt.$$

Since the L^p norm is translation invariant and $|x^\alpha y^\beta| \leq |z|^m$ for $\alpha + \beta = m$, we have

$$\|\varphi_\delta * u - u\|_{L^p} \leq \frac{1}{m!} \sum_{\alpha+\beta=m} \|\partial_x^\alpha \partial_y^\beta u\|_{L^p} \int |z|^m |\delta^{-2} \varphi(z/\delta)| \, dx \, dy \leq M\delta^m \sum_{\alpha+\beta=m} \|\partial_x^\alpha \partial_y^\beta u\|_{L^p}.$$

4.6. *Explicit Core Functions*

Suppose φ is a continuous radial function and write $\varphi(z) = \varphi(r)$ where $r^2 = |z|^2 = x^2 + y^2$. Then $\int x^\alpha y^\beta \varphi(z)\, dx\, dy = 0$ if α or β is odd, so the order-m moment conditions (13) become

$$\int_0^1 \varphi(r) r dr = 1/2\pi, \qquad \int_0^1 \varphi(r) r^{2j+1} \, dr = 0, \quad j = 1, ..., n,$$

where $m = 2n + 2$ is even.

Using scaling, the explicit formula (4) for K, polar coordinates, and the standard integral,

$$\int_0^{2\pi} \frac{1 - a\cos\theta}{1 - 2a\cos\theta + a^2}\, d\theta = \begin{cases} 2\pi & \text{if } a^2 < 1 \\ 0 & \text{if } a^2 > 1 \end{cases}$$

gives the useful result

$$K_\delta(z) = \varphi_\delta * K(z) = f\left(\frac{r}{\delta}\right) K(z),$$

where the "shape factor" f is given by

$$f(r) = 2\pi \int_0^r \varphi(s) s\, ds.$$

Since $\varphi(r) = 0$ for $r > 1$, we have $f(r) = 1$ for $r > 1$. This facilitates fast summation methods since $K_\delta = K$ for $r \geq \delta$.

We now construct a family of shape factors f. A convenient ansatz suggested by [17] is

$$f(r) = \varrho^p \left[a_d \varrho\, d + \cdots + a_0\right] + 1, \tag{14}$$

where $\varrho = (1 - r^2)_+ = \max(0, 1 - r^2)$ and

$$\begin{aligned} \varphi(r) &= \frac{1}{2\pi r} f'(r) \\ &= \frac{-1}{\pi}\left[(p + d)a_d \varrho^{p+d-1} + \cdots + p a_0 \varrho^{p-1}\right] \end{aligned} \tag{15}$$

for $r^2 < 1$. For $r^2 > 1$, $\varphi(r)$ vanishes. Such a core function φ has $p - 2$ continuous and $p - 1$ bounded derivatives.

The $d + 1$ coefficients a_i must be chosen so that φ satisfies $n + 1$ moment conditions, so we cannot expect a solution unless $d \geq n$. A brief calculation shows that the moment conditions are equivalent to a linear system

$$Aa = b,$$

where $b_0 = -1$, $b_i = 0$ for $i > 0$, $a = (a_0, a_1, ..., a_d)$ and the $n + 1$ by $d + 1$ matrix A is determined by the recurrence

$$A_{ij} = \frac{1}{p + i + j} A_{i-1,j}, \qquad 0 < i \leq n, 0 \leq j \leq d,$$

with initial values $A_{0j} = 1$ for $0 \leq j \leq d$. When p is large, each row is almost proportional to the previous one, so A is highly ill-conditioned. If $d > n$, the linear system of moment conditions is underdetermined, and we use complete orthogonal factorization to find the solution with smallest 2-norm.

Given the coefficients a_i, we have

$$K_\delta(z) = \frac{z^\perp}{2\pi r^2}\left[(1 - r^2/\delta^2)_+^p (a_d(1 - r^2/\delta^2)_+^d + \cdots + a_0) + 1\right],$$

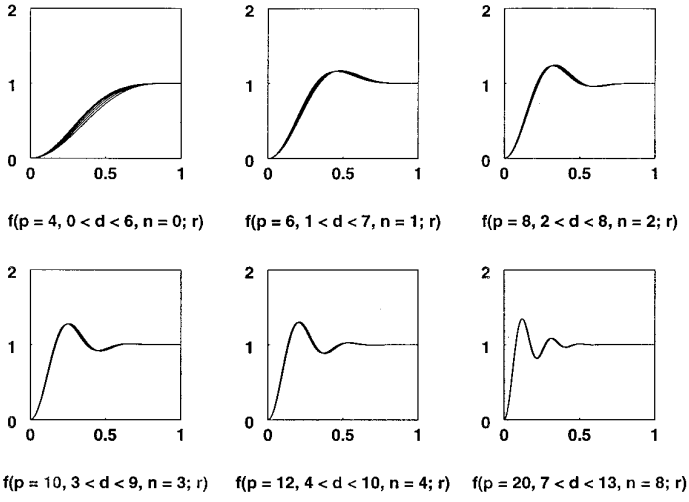

FIG. 5. Piecewise polynomial shape factors f with various parameters.

where $z^\perp = (-y, x)$. Since subtraction of the $O(1)$ quantities inside the square brackets must yield $O(r)$ as $r \to 0$, we expect roundoff problems in evaluating K_δ when $r \ll \delta$. Roundoff can be reduced by observing that since $f(0) = 0$, there exists a polynomial g such that

$$f(r) = r^2 g(\varrho) = r^2\left[b_{p+d-1}\varrho^{p+d-1} + \cdots + b_0\right].$$

In terms of g, we have a convenient formula

$$K_\delta(z) = \frac{z^\perp}{2\pi \max(r^2, \delta^2)} g((1 - r^2/\delta^2)_+),$$

where g need never become smaller than $O(\delta^2)$ as $r \to 0$. The coefficients b_j are given by

$$b_{p-1} = b_{p-2} = \cdots = b_1 = b_0 = 1$$

and

$$b_p = b_{p-1} + a_0, ..., b_{p+d-1} = b_{p+d-2} + a_{d-1}.$$

Several well-known core functions are included in this scheme. For example, Nordmark's fourth-order core function from [17] has $p = 10$, $d = n = 3$, and $m = 8$: the corresponding shape factor is

$$\begin{aligned} f(r) &= \varrho^{10}\left[-560\varrho^3 + 1365\varrho^2 - 1092\varrho + 286\right] + 1 \\ &= r^2\left[560\varrho^{12} - 805\varrho^{11} + 287\varrho^{10} + \varrho^9 + \varrho^8 + \cdots + 1\right]. \end{aligned}$$

Figure 5 shows several shape factors of this type, for various choices of parameters. The increasing oscillation as n increases follows naturally from the vanishing of more moments.

TABLE I

n	m	p	d	M	d	M
0	2	4	1	2.1–2	5	1.6–2
1	4	6	2	3.6–3	6	1.5–3
2	6	8	3	3.8–4	7	8.9–5
3	8	10	4	2.0–5	8	3.2–6
4	10	12	5	6.9–7	9	8.5–8
8	18	20	8	1.8–13	12	5.0–15

Note. Error constants M as a function of moment order m, smoothness p, and polynomial degree d for the piecewise polynomial shape factors (14) shown in Fig. 5.

The polynomial degree d makes little difference to the values of high-order kernels, but Table I shows that increasing d can noticeably reduce the sizes of the coefficients and thus the smoothing error bound. Indeed,

$$\begin{aligned} M &= \frac{1}{m!}\int |z|^m|\varphi(z)|\,dx\,dy \\ &\le \frac{1}{\pi m!}\int_0^1 r^{m+1}\left[(p+d)|a_d|\varrho^{p+d-1} + \cdots p|a_0|\varrho^{p-1}\right] dr \\ &\le \frac{1}{2\pi m!}\left[|a_d| + \cdots + |a_0|\right]. \end{aligned}$$

4.7. *Time Stepping Techniques*

Since the Euler equations are not stiff and we are constructing high-order vortex methods, we discretize time with explicit s-step Adams or s-stage Runge–Kutta methods. Adams methods are attractive because they require only one velocity evaluation per time step, while Runge–Kutta methods require s velocity evaluations. However, the smaller error constants of Runge–Kutta methods sometimes balance out this advantage. Also, Adams methods require an accurate procedure for computing the s starting values. A natural thought is to start with Runge–Kutta steps, but this turns out to almost double the memory requirements of the ODE solver, because during the Runge–Kutta steps we need storage for the Runge–Kutta stages while we simultaneously build up memory for the Adams methods. Thus we use a variable-step variable-order self-starting Adams method constructed as follows.

Suppose we use an explicit s-step Adams method with a fixed time step Δ_f. We begin with a tiny time step $\Delta_i \ll \Delta_f$ and 1-step Adams, giving error $O(\Delta_i^2)$. Since our final method is order-s accurate, we should choose $\Delta_i = O(\Delta_f^{s/2})$. We now increase the order of the Adams method by 1 at each step until order s is reached, simultaneously increasing Δ_i by a factor $R \le 2$ until Δ_f is reached. The final non-equidistant step is adjusted to land precisely at a multiple of $t = \Delta_f$. For more demanding calculations a truly variable step and order would be necessary, but in the present research-oriented code this has not been felt necessary.

4.8. *Balance of Error*

We now balance the errors due to smoothing and quadrature. The error in velocity evaluation splits naturally into two parts

$$\begin{aligned} E &= \Big|u(z) - \sum_{j=1}^N w_j K_\delta(z - z_j)\omega(z_j)\Big| \\ &\le |K * \omega(z) - K_\delta * \omega(z)| \\ &\quad + \Big|K_\delta * \omega(z) - \sum_{j=1}^N \omega_j K_\delta(z - z_j)\omega(z_j)\Big| \\ &= E_\delta + E_{N,\delta}. \end{aligned}$$

Here E_δ is the smoothing error, which satisfies

$$E_\delta \le M\,\delta^m \|u\|_m$$

if φ satisfies moment calculations (Eq. (13)) of order m and $u \in C^m$. The second term $E_{N,\delta}$ is the quadrature error, which satisfies (Eq. (12))

$$E_{N,\delta} \le \Omega\,|B|\,\frac{h^q}{q!}\,\|g\|_q, \qquad g(z') = K_\delta(z - z')\omega(z')$$

for each fixed z. By a standard inequality for the C^q norm of a product of two functions [15], we have

$$\|g\|_q \le C(\|K_\delta\|_q\|\omega\|_0 + \|K_\delta\|_0\|\omega\|_q)$$

for some constant C. We know that

$$\begin{aligned} K_\delta(z) &= \int \delta^{-2}\varphi\left(\frac{z - z'}{\delta}\right)\frac{z'^\perp}{2\pi|z'|^2}\,dx'\,dy' \\ &= \delta^{-1}\int \varphi\left(\frac{z}{\delta} - z'\right)\frac{z'^\perp}{2\pi|z'|^2}\,dx'\,dy', \end{aligned}$$

so there is some constant C, depending only on φ, such that

$$\|\partial_x^\alpha\partial_y^\beta K_\delta\|_0 \le C\delta^{\alpha+\beta-1}$$

if $\varphi \in C^{\alpha+\beta}$. Thus if $\varphi \in C^q$, we have

$$\|g\|_q \le \frac{1}{\delta}\,C(\delta^{-q}\|\omega\|_0 + \|\omega\|_q)$$

so the quadrature error satisfies

$$E_{N,\delta} \le C\left(\delta^{-1} \left(\frac{h}{\delta}\right)^q \|\omega\|_0 + \delta^{-1} h^q \|\omega\|_q \right).$$

Hence the total error in one velocity evaluation statisfies

$$E_\delta + E_{N,\delta} \le C\left(\delta^m \|u\|_m + \delta^{-1} \left(\frac{h}{\delta}\right)^q \|\omega\|_0 + \delta^{-1} h^q \|\omega\|_q \right),$$

where q is the order of quadrature and $\varphi \in C^q$ satisfies moment conditions (13) of order m.

We now take advantage of the separation between $\|\omega\|_0$ and $\|\omega\|_q$ to derive a consistency error bound. (A similar separation has been used in [8].) We choose δ as a function of h to make

$$\delta^{-1} \left(\frac{h}{\delta}\right)^q \le \varepsilon,$$

where ε is a user-specified error tolerance, presumably small but *fixed* as $h \to 0$. This implies

$$\delta = O(\varepsilon^{-1/(q+1)} h^{q/(q+1)}) = O(h^a), \qquad a = 1 - \frac{1}{q+1},$$

and our error bound becomes

$$E \le C(\varepsilon \|\omega\|_0 + h^{mq/(q+1)} \|u\|_m + h^{q^2/(q+1)} \|\omega\|_q).$$

The choice $m = q$ balances the two remaining h-dependent terms (since $\|u\|_m \le C\|\omega\|_m$ if ω has compact support), and we find

$$E \le C\,[\varepsilon \|\omega\|_0 + h^k (\|\omega\|_q + \|u\|_q)] = O(\varepsilon + h^k),$$

where $k = q^2/(q+1) = q - 1 + 1/(q+1) > q - 1$.

For quadrature of orders $q = 2, 4, 6, 8, 10$, the exponent a in $\delta = O(h^a)$ is 0.66, 0.80, 0.86, 0.89, 0.91, respectively, with order of accuracy k equal to 1.33, 3.20, 5.14, 7.11, 9.09 rapidly approaching $q - 1$ from above as q increases. Thus δ is very close to $O(h)$ for methods of high order k, with only q derivatives of ω required.

This allows us to use fast summation methods with excellent efficiency: the fast multipole method with this δ costs $O(N^b)$ with $b = 1 + 1/(q+1) = 1.33, 1.20, 1.14, 1.11, 1.09$, very close to 1. Even for the slowest method with $k = 4/3$, the $O(N^{4/3})$ cost of this method is comparable to the second-order triangulated vortex method and better than the classical vortex method with $\delta = O(\sqrt{h})$.

This error bound is not standard—due to the $O(\varepsilon)$ term—but extremely useful. It suggests parameter balances which give almost optimal accuracy and efficiency at the price of a definition of convergence slightly different from usual. Such a definition costs us very little in this context, because the fast multipole method already involves error ε.

We combine this order-k velocity evaluation with an Adams method of order $s = q > k$, because the (first-order) Euler equations imply that the velocity should have roughly the same order of smoothness in time as in space, with the particle positions one order smoother by the flow map equation (Eq. (1)). An order $O(\varepsilon + h^k)$ error in the velocity u at each time step fortunately does not accumulate in the numerical solution of the ODEs

$$\dot{\Phi}(z,t) = u(\Phi(z,t),t)$$

so we expect to obtain a maximum norm error in Φ of order

$$O(\varepsilon + \Delta_f^s + h^k)\|\omega\|_q$$

as h and Δ_f vanish. This would imply similar estimates for the velocity and vorticity by standard stability arguments [14], though these arguments have not yet been extended to our method. However, the numerical results display the expected order of accuracy, indicating that the method is stable.

5. IMPLEMENTATION AND NUMERICAL RESULTS

We implemented the fast adaptive vortex method of Section 4 and studied several numerical examples. First, we measured the accuracy and efficiency of the velocity evaluation scheme in isolation. Then we measured the error in long-time calculations with the full method. Finally, we studied the interaction of several smooth patches of vorticity.

5.1. *Implementation*

We constructed a single portable Fortran code to carry out all the computations for this work. In order efficiently to compare the present method to the classical method and the triangulated method, we developed a highly modular reverse communication driver program for a general vortex method. Reverse communication is a useful paradigm for designing routines which must obtain information from other routines with unpredictable calling sequences. Many standard packages for ODE solving, for example, require the user of the package to force the right-hand side of his ODE into a standard calling sequence. Reverse communication, by contrast, simply sets a flag and returns control to the user whenever an external function evaluation (or—for efficiency—a vector of function evaluations) is required. Thus the user calls a reverse communication based ODE

```
Reverse Communication Vortex Method:  Driver Program

Input parameters:

   Output and housekeeping parameters for driver only:
      t_i, t_f, t_p, output files and controls.
   Initial vorticity parameters.
   Initial grid parameters:  number of vortices N, type of grid.
   ODE solver parameters:
      Initial time step Δ_i, final time step Δ_f, ratio R, order k.
   Velocity evaluation parameters:
      Quadrature:  order q, safety factor S.
      Smoothing:  smoothness p, degree d, moments n, radius δ = Ch^a.
      Fast summation:  tolerance ε.


Initialize output, grid, vorticity, ODE solver, Biot-Savart
solver.

Do while t < t_f:
   Evaluate velocity from Biot-Savart law.
   Estimate error, write output, store data (postprocess separately).
   Do while (ODE solver not done) :
      Call ODE routine:  move points and store in z_i.
      If (ODE solver not done)
         Evaluate velocity from Biot-Savart law at points z_i.
      End if
   End while
End while
```

FIG. 6. Outline of driver routine for a vortex method based on reverse communication with an arbitrary ODE solver.

solver repeatedly within each time step until the demands of the solver for information about the ODE are satisfied. The structure of our driver is outlined in Fig. 6.

5.2. *Velocity Evaluation*

We studied the accuracy of the velocity evaluation of orders $k = 1.33$, 3.20, 5.14, and 7.11 corresponding to $m = q = 2$, 4, 6, 8, using the well-known Perlman test case [18]

$$\omega_P(z) = (\max(0, 1 - r^2))^P,$$

where $P = 7$. The vorticity ω_P is a C^{P-1} function on $\mathbf{R}^2$, while the corresponding velocity fields are C^P:

$$u(z) = (1 - \omega_{P+1}(z)) \frac{z^\perp}{(2P+2)r^2}.$$

This is a stationary radial solution of the Euler equations with shear and a popular test case for vortex methods.

We tested our method with the following random initial grid. Given N and n with $n^2 \le N$, first distribute n^2 vortices uniformly over a rectangle R enclosing the support of the vorticity: Divide R into a $n \times n$ grid and choose a point z_i randomly in the ith grid cell. Of the remaining $M = N - n^2$ vortices, put

$$m_i = \left\lfloor \frac{M|\omega(z_i, 0)|}{\Sigma_i |\omega(z_i, 0)|} \right\rfloor$$

or $m_i + 1$ random vortices located in the ith cell of the $n \times n$ grid. Thus the remaining $N - n^2$ vortices are distributed in regions where the vorticity is large, providing some degree of adaptivity despite their randomness. Note that the vorticity is conserved along particle paths, so the particles tend to stay where ω is large.

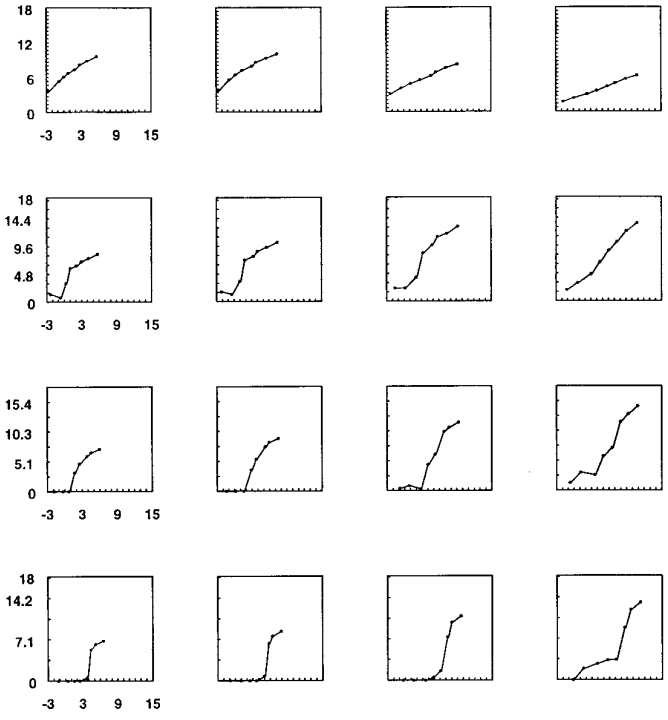

FIG. 7. Correct bits in velocity in L^1 norm plotted vs. bits in CPU time, for orders $m = q = 2, 4, 6, 8$ (from top) and smoothing radii $\delta = Ch^a$ where $C = 1/4, 1/2, 1$, and 2 (from left).

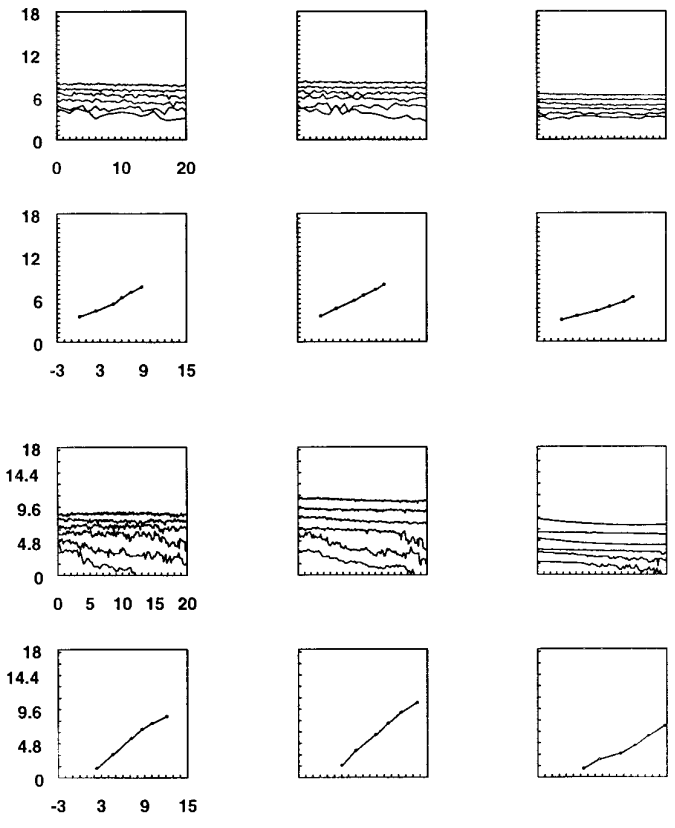

FIG. 8. Correct bits $B_1(u)$ in velocity u, plotted vs. time (top of each pair) and CPU time (bottom of each pair) for adaptive vortex methods of orders 1.33 (top pair) and 3.20 (bottom pair). The smoothing radius $\delta = Ch^a$ where C increases from left to right in each row.

We generated $N = 500, 1000, 2000, ..., 64000$ vortices in such an adaptive random grid with $n^2 \approx N/10$ and evaluated the velocity at each of the vortices. Figure 7 plots the number of correct bits

$$B_1 = \max\left(0, -\log_2\left[\frac{\|u - u_{h,\delta}\|_1}{\|u\|_1}\right]\right)$$

in the computed velocity $u_{h,\delta}$ in the discrete L^1 norm

$$\|u\|_1 = \frac{1}{N}\sum_{i=1}^{N} |u_i|$$

vs. the CPU time T (in seconds on a Ultra-1 workstation). We used core functions and quadratures of orders $m = q = 2, 4, 6, 8$ (from top to bottom in each column of the figure), four values $C = 1/4, 1/2, 1$, and 2 (from left to right in each row of the figure) of the constant in the formula

$$\delta = Ch^a, \qquad a = 1 - \frac{1}{q+1}.$$

As one would expect, lower-order methods did well with small C and higher-order methods did better with larger C, though the cost of the computation increases rapidly with C. More precisely, the velocity evaluation produces error $O(\varepsilon + N^{-k/2})$ with $k/2 = 0.67, 1.60, 2.57$, and 3.55 in $O(N^b \log \varepsilon)$ CPU time with $b = 1.33, 1.20, 1.14$, and 1.11 and a constant of proportionality depending very weakly on the order q. Note that each time N is doubled, the average cell size h decreases by a factor $\sqrt{2}$, so we expect to gain $k/2$ bits until $O(\varepsilon)$ is reached. For this reason, the

TABLE II

N	h	δ	Δ_f	Δ_i	T	$\|u\|_1$	$\|\omega\|_1$
250	1.30	1.19	0.20	0.020	0.59	0.1527	0.2181
500	0.86	0.90	0.14	0.014	2.92	0.1835	0.2048
1000	0.64	0.74	0.10	0.010	9.89	0.1724	0.2302
2000	0.43	0.57	0.07	0.007	36.25	0.1834	0.2108
4000	0.32	0.47	0.05	0.005	68.72	0.1808	0.2315

Note. Number of vortices N, mesh size h at $t = 0$, smoothing radius δ, time steps Δ_f and Δ_i, and CPU time T per step (on a Sparc-2 workstation) for the adaptive method of order 1.33. Here $\|u\|_1$ and $\|\omega\|_1$ are the L^1 norms of the velocity and vorticity, measured at time $t = 30$.

FIG. 9. Vorticity contours for two interacting Perlman patches with $P = 3$, computed with the adaptive method of order 1.33 and numerical parameters given in Table II. The first four plots show the evolution at $t = 0$, 10, 20, and 30 with $N = 4000$ vortices, the last row shows the final frame $t = 30$, computed with $N = 1000$ (left) and $N = 2000$ (right).

plots share a common scale for ease of comparison, but the tick marks are placed in such a way that the number of correct bits and the bits in the CPU time should increase by one tick mark each time N is doubled.

For first-order methods, the $O(h^{1.33})$ errors decrease slowly enough that the $O(\varepsilon)$ limit on accuracy never appears. For higher-order methods, we get higher-order convergence in the region where the smoothed kernel is resolved but the $O(\varepsilon)$ limit has not appeared. After the limit is reached, convergence continues slowly.

5.3. *Long-time Accuracy*

We also tested the long-time accuracy of the method on the Perlman test case with $P = 7$, running for $0 \le t \le 20$, a final time at which the fastest-moving particles of fluid (near the origin) have completed 1.6 revolutions while the slowest have completed only 0.2. This strong shear is usually considered a severe test for a vortex method. We started with an almost uniformly distributed adaptive random grid with $n^2 \approx 0.8N$, for $N = 250$, 500, 1000, 2000, 4000, and 8000, and used core functions, quadratures, and Adams methods of orders $m = q = s = 2$ and 4, yielding adaptive vortex methods of orders $k = 1.33$ and 3.20. Accuracy requires a smaller time step and a larger smoothing radius as the order increases; we used 20, 30, 40, 60, 80, 120 time steps for $s = 2$ and 80, 120, 160, 240, 320, 480 time steps for $s = 4$. The smoothing radius $\delta = Ch^a$ where $C = 1/4$, 1/2, 1 for $s = 2$ and $C = 1/2$, 1, 2 for $s = 4$.

The correct bits in L^1 norm in the velocity are plotted in Fig. 8. The plots share a common scale, but the tick marks are placed in such a way that the number of correct bits should increase by one tick mark each time N is dou-

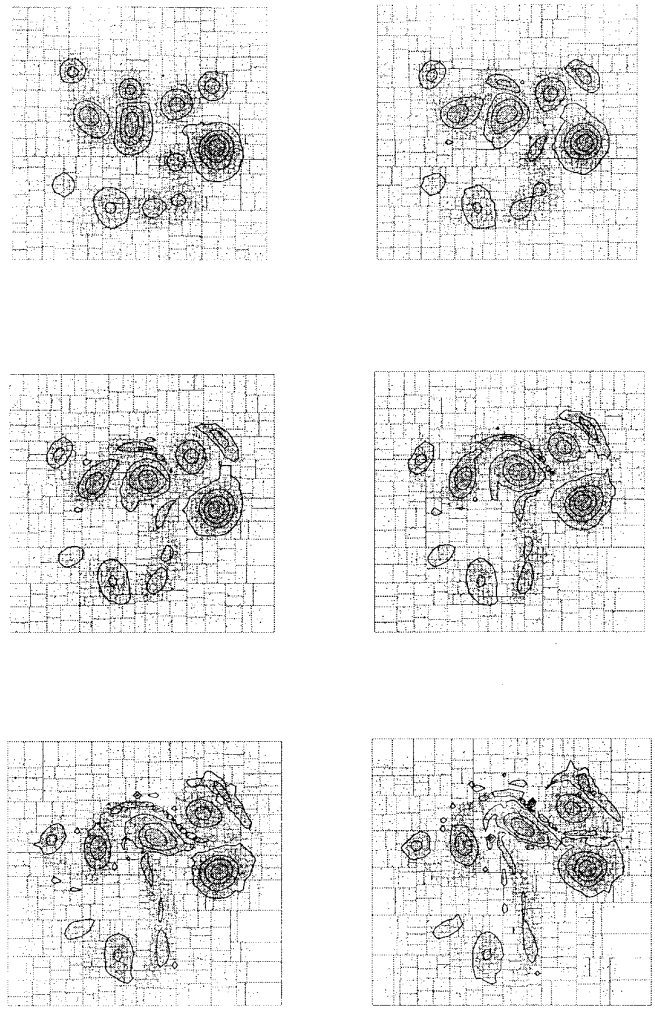

FIG. 10. Vorticity contours for 20 interacting Perlman patches with $P = 5$, computed with the adaptive method of order 3.20. Results are shown at $t = 0$, 4, 8, 12, 16, and 20 with $N = 10000$ vortices.

bled. These results clearly confirm the long-time high-order accuracy of the method; they do not show the loss of accuracy observed in Lagrangian vortex methods (for example, in Fig. 1). The errors are slightly oscillatory on a small time scale, because a new quadrature rule is built from scratch at each step.

5.4. *Interacting Vortex Patches*

As a more complex example, we used the order-1.33 method with parameters given in Table II to compute two interacting smooth patches of vorticity. Thus the initial vorticity is given by

$$\omega(z, 0) = \sum_{j=1}^{Q} \Omega_j (1 - |z - z_j|^2)^P,$$

where $Q = 2$, $P = 3$, and z_j and Ω_j are given by $z_1 = (0, 1.05)$, $z_2 = (0, -1.05)$, $\Omega_1 = 2$, and $\Omega_2 = 1$. Figure 9 shows the final result at $t = 30$ with $N = 1000$, 2000, and 4000; the large-scale features of the results are clearly converged.

We also carried out a similar computation with 20 randomly located and scaled patches ($Q = 20$, $P = 5$) with random strengths Ω_j, using the order-3.20 method with $\Delta_1 = 0.10$ and $\delta = 1.2h^{4/5}$. Some sample vorticity contours are shown in Fig. 10. The L^1 norm of ω is conserved exactly by our method, even for this fairly complicated flow. The L^∞ norm is trivially conserved since the vorticity values are carried by the flow.

REFERENCES

1. C. Anderson and C. Greengard, On vortex methods, *SIAM J. Math. Anal.* **22,** 413 (1985).
2. C. R. Anderson, A method of local corrections for computing the velocity field due to a collection of vortex blobs, *J. Comput. Phys.* **62,** 111 (1986).
3. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongara, J. du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide* (SIAM, Philadelphia, 1992).
4. J. T. Beale, On the accuracy of vortex methods at large times, in *Computational Fluid Dynamics and Reacting Gas Flow,* edited by B. Engquist, M. Luskin, and A. Majda, IMA Volumes in Mathematics and Applications, Vol. 12 (Springer-Verlag, New York/Berlin, 1988).
5. J. T. Beale and A. Majda, High order accurate vortex methods with explicit velocity kernels, *J. Comput. Phys.* **58,** 188 (1985).
6. J. Carrier, L. Greengard, and V. Rokhlin, A fast adaptive multiple method for particle simulations, *SIAM J. Sci. Statist. Comput.* **9,** 669 (1988).
7. T. Chacon Rebollo and T. Y. Hou, A Lagrangian finite element method for the 2-D Euler equations, *Comm. Pure Appl. Math.* **43,** 735 (1990).
8. J.-P. Choquin and G.-H. Cottet, Sur l'analyse d'une classe de méthodes de vortex tridimensionelles, *C. R. Acad. Sci.* **306,** 739 (1988).
9. A. J. Chorin, *Computational Fluid Mechanics: Selected Papers* (Academic Press, San Diego, 1989).
10. P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration,* Computer Science and Applied Mathematics (Academic Press, San Diego, 1984), 2nd ed.
11. J. R. Grant, S. A. Huyer, and J. S. Uhlman, Solution of the vorticity equation on a Lagrangian mesh using triangularization: Computation of the Biot-Savart integral in three dimensions, Technical Report, NUWC, 1994 (unpublished).
12. K. E. Gustafson and J. A. Sethian (Eds.), *Vortex Methods and Vortex Motion* (SIAM, Philadelphia, 1991).
13. O. H. Hald, Convergence of vortex methods for Euler's equations, III, *SIAM J. Numer. Anal.* **24,** 538 (1987).
14. O. H. Hald, Convergence of vortex methods, in *Vortex Methods and Vortex Motion,* edited by K. E. Gustafson and J. A. Sethian (SIAM, Philadelphia, 1991), pp. 33–58.
15. L. Hörmander, The boundary problems of physical geodesy, *Arch. Rational Mech. Anal.* **62,** (1976).
16. S. A. Huyer and J. R. Grant, Incorporation of boundaries for 2D triangular vorticity element methods, Technical Report, NUWC, 1994 (unpublished).
17. H. O. Nordmark, Rezoning for high-order vortex methods, *J. Comput. Phys.* **97,** 366 (1991).
18. M. Perlman, On the accuracy of vortex methods, *J. Comput. Phys.* **59,** 200 (1985).
19. P. A. Raviart, An analysis of particle methods, in *Numerical Methods in Fluid Dynamics,* edited by F. Brezzi, Lecture Notes in Mathematics, Vol. 1127 (Springer-Verlag, New York/Berlin, 1985).
20. G. Russo and J. Strain, Fast triangulated vortex methods for the 2-D Euler equations, *J. Comput. Phys.* **111,** 291 (1994).
21. J. Strain, Fast potential theory. II. Layer potentials and discrete sums, *J. Comput. Phys.* **99,** 251 (1992).
22. J. Strain, Locally-corrected multidimensional quadrature rules for singular functions, *SIAM J. Sci. Comput.* **16,** 1 (1995).
23. J. Strain, 2-D vortex methods and singular quadrature rules, *J. Comput. Phys.* **124,** 1 (1996).