Lawrence Berkeley National Laboratory
Recent Work

Title
Fast Adaptive 2D Vortex Methods

Permalink
https://escholarship.org/uc/item/9dx476qc

Author
Strain, John A.

Publication Date
1996-03-01

eScholarship.org Powered by the California Digital Library
University of California

LBL-38457
UC-405
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Physics Division

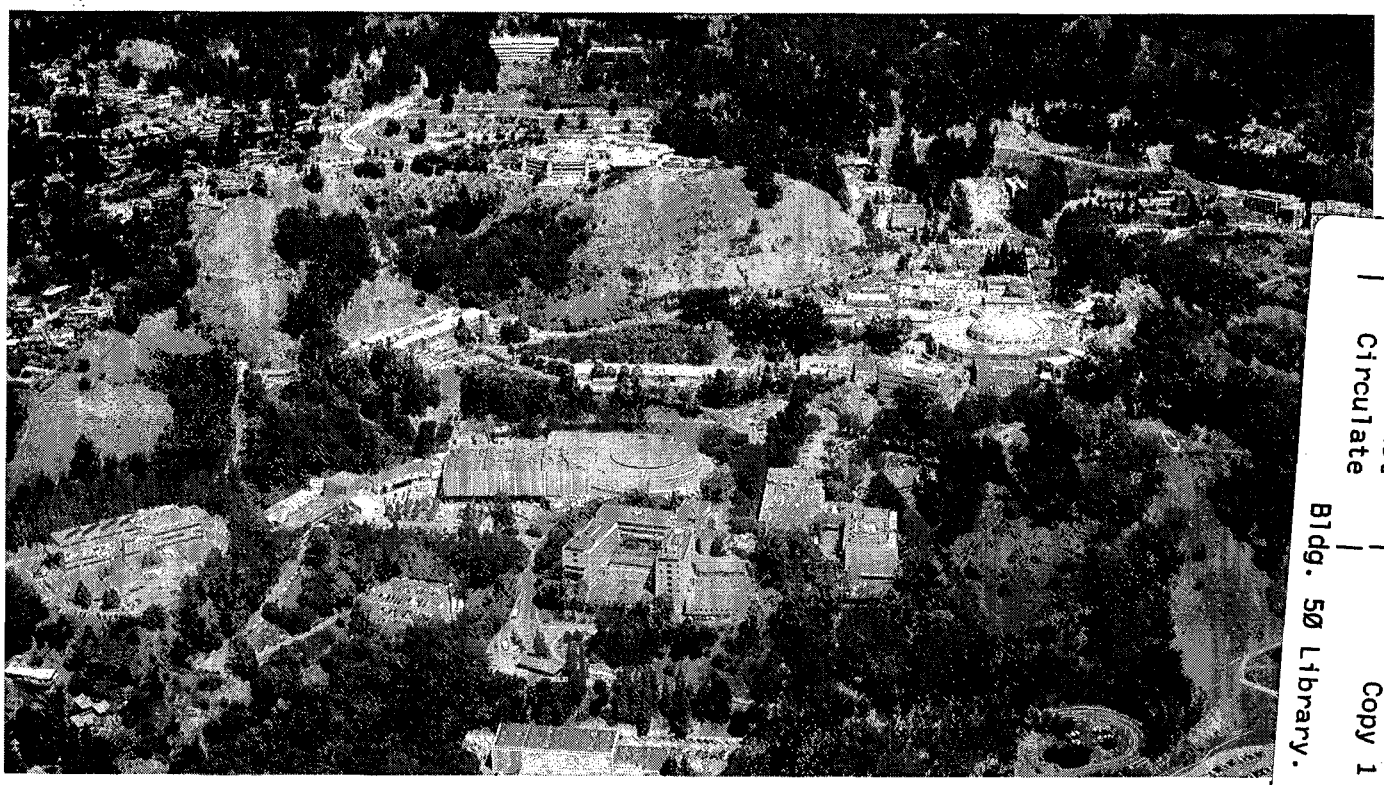
Mathematics Department

To be submitted for publication

Fast Adaptive 2D Vortex Methods

J. Strain

March 1996



REFERENCE COPY
Does Not
Circulate
LBL-38457
Copy 1
Bldg. 50 Library.

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

LBL-38457

FAST ADAPTIVE 2D VORTEX METHODS

John Strain[1]
Department of Mathematics
and
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

March 1996

[1]This research was 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.

Abstract

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 nonstandard error analysis. Our error analysis halves the differentiability required of the flow, suggests an efficient new balance of smoothing parameters, and combines naturally 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 applied to complex high-Reynolds-number flows: See [11] for a survey.

Vortex methods involve several components: velocity evaluation, vortex motion, diffusion, boundary conditions and regridding. 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 nonstandard way, requiring less differentiability of the flow and obtaining efficient new parameter balances. We employ standard techniques for the vortex motion and consider inviscid free-space flow to eliminate diffusion and boundary conditions. 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, 17], a difficulty which motivates many regridding techniques [16]. Even before the flow becomes disorganized, however, obtaining high-order accuracy with a fixed 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 relative to the differentiability of the flow.

In Section 3, we review two free-Lagrangian vortex methods, the triangulated vortex method of [19] and the quadrature-based method of [22]. Triangulated vortex methods are robust, practical and efficient but limited to second-order accuracy. Quadrature-based methods compute 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 produced by kernel smoothing is almost eliminated by a careful choice of smoothing functions and parameters, based on a 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 multistep method. The velocity evaluation is nonstandard: 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 is an overview of 2D vortex methods. First, we describe how the 2D Euler equations reduce to an infinite system of ordinary differential equations for the flow map. This formulation leads naturally to vortex methods. We contrast the Lagrangian and free-Lagrangian viewpoints, then review the convergence theory of Lagrangian methods.

Second, we explore avenues for improvement. We explain the conflict between smoothing for accuracy and fast summation for speed, and demonstrate the Perlman effect in which the derivatives of the 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:

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

We close Eqns. (1) and (2) for Φ and ω by solving 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 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, 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 $z' \leftarrow \Phi(z',t)$. The Jacobian is unity because the flow is incompressible, so this gives a closed system 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 Eqn. (6), tracking N points $z_j(t) \approx \Phi(z_j, t)$ moving with the fluid velocity, starting at $t = 0$ from the nodes z_j of a quadrature formula with weights w_j. Suppose we use a quadrature formula

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

with a qth-order error bound

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

for $g \in C^q$. Here h is the mesh size of the rule and the C^q norm is defined 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 to the Lagrangian equation of motion (6) gives a system of N ordinary differential equations:

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

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 smoothing [8], often with φ and the "core radius" δ chosen to give high-order convergence as the mesh size h vanishes [13]. This can be guaranteed by the following conditions on φ and ω:

$$\begin{aligned} \int \varphi &= 1, \\ \int x^\alpha y^\beta \varphi &= 0, \qquad 1 \le \alpha + \beta \le m-1, \qquad (8) \\ \int |z|^m |\varphi| &< \infty \\ \varphi &\in C^L \quad \text{and} \quad \varphi(z) = 0 \quad \text{for} \quad |z| \ge 1, \qquad (9) \\ \omega &\in C^M \qquad \text{has compact support.} \qquad (10) \end{aligned}$$

High-order accuracy requires smooth solutions, so condition (10) on ω 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

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

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 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 [12] and Nordmark [16] show that $\delta = O(\sqrt{h})$ is a good choice. Then $2m$ derivatives of ω guarantee only $O(h^m)$ accuracy. Later, we reduce this to $m+1$ derivatives at the cost of redefining convergence.

2.3 Cost and accuracy

Convergence theory must be augmented by practical considerations of cost and 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 direct velocity evaluation costs $O(N^2)$ work. This is prohibitively 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 [20]. These schemes evaluate unsmoothed sums like

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

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

However, this does not completely resolve the difficulty. Fast methods cannot evaluate the smoothed interaction $K_\delta(z_i - z_j)$ between vortices z_i and z_j closer than δ, because $K_\delta \neq K$. Asymptotically, 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 $\delta = O(\sqrt{h})$.

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 hampers fast summation schemes. In Section 4 we resolve this conflict by allowing another $O(\epsilon)$ in the error bound.

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 Eqn. (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. 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 in long-time calculations [5, 17]. For example, Figure 1 plots the

number of correct bits in the computed velocity and vorticity of a standard test case for a fourth-order vortex method. Fourth-order accuracy—evidenced by the gain of one tick mark per line in the figure—is attained only during a very short initial time period. Figure 1 also plots the errors when the C^6 vorticity is replaced by a C^2 vorticity. The order of accuracy of the velocity is halved, indicating that the differentiability requirement is genuine. The vorticity errors, however, continue to converge with fourth-order accuracy during the usual short initial time period. The numerical parameters used are summarized in Table 3 in Section 5.2.

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 [16]. Similarly, Beale has developed an iterative reweighting scheme to overcome the Perlman effect [4]. The Perlman effect also motivated the free-Lagrangian vortex methods we discuss next.

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 adaptive quadratures with new 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

$$\dot{z}_j(t) = u_h(z_j, t) = \int K(z_j(t) - z')\omega_h(z', t)dx'dy', \tag{11}$$

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 $\mathbf{R}^2$.

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 convergent, the resulting scheme costs $O(N^2)$ work per time step with a large constant and loses accuracy quickly because the triangulation degenerates.

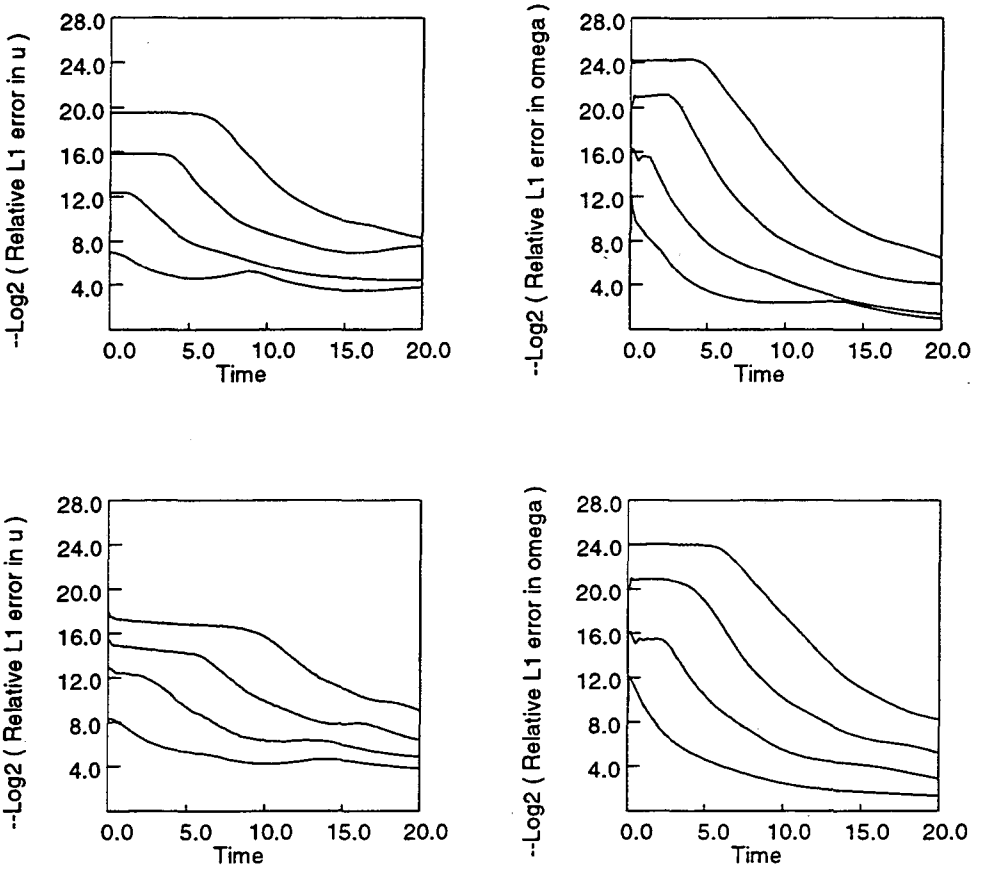


Figure 1: Correct bits for the vortex method with parameters from Table 3 in Section 5.2. The top row plots errors when ω is C^6, the second when ω is C^2.

We developed practical triangulated vortex methods in [19]: a fast summation scheme brought the cost down to $O(N^{4/3})$ 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 Figure 1, with numerical parameters given in Table 4 in Section 4.2. The error displays no Perlman effect; second-order accuracy (one tick per line) is maintained uniformly in time. The triangulated approach is now being applied to flows in three dimensions with viscosity and boundaries [10, 15]. However, it seems difficult to make a triangulated vortex method with higher than second-order accuracy. This motivated the next approach we discuss.

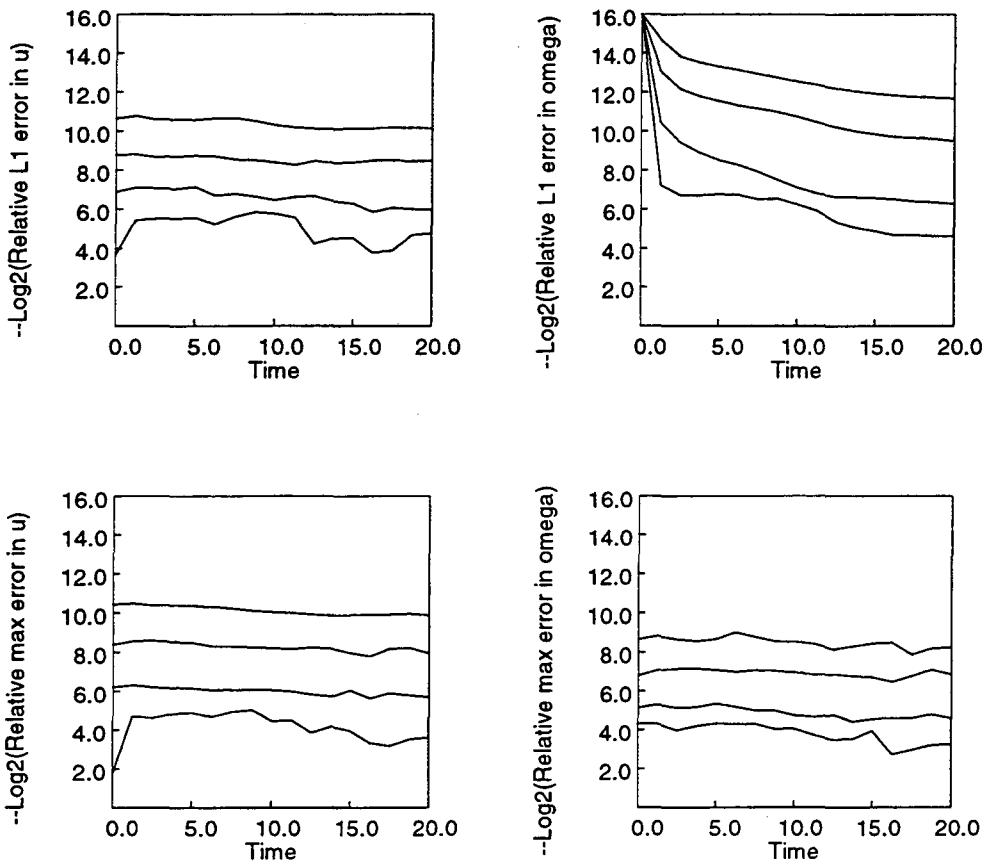


Figure 2: Correct bits for the triangulated vortex method with parameters from Table 4 in Section 5.2.

3.2 Quadrature-based methods

We developed higher-order free-Lagrangian methods in [22]. The basic idea is to construct time-dependent quadrature weights $w_{ij}(t)$ which give high-order accuracy in the Biot-Savart law:

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

For example, high-order product integration weights [9] make smoothing unnecessary, but the i-dependence of $w_{ij}(t)$ precludes fast summation methods. Thus 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. The price for efficiency, however, is a redefinition of convergence. The error bound for these quadratures is $O(\epsilon + h^q)$, 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 $O(\epsilon)$. This is 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 $O(\epsilon)$ error as well. High-order accuracy can be maintained for long times, though these rules are somewhat expensive to implement.

4 A new approach

We now present a new high-order 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 Biot-Savart kernel.
- New 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. The method is flowcharted in Figure 6.

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, a guarantee of order-q accuracy is unavailable. Thus we construct rules with an error bound composed of two factors. The first depends only on the point locations and is easily computable a posteriori, as a monitor for bad point 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 Section 4.2, we construct order-q rules on each cell in Section 4.3. The union of these rules is globally accurate of order q: We quote an error bound from [21] in Section 4.4.

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

Section 4.7 presents our multistep time stepping procedure and the starting value problem. Finally, Section 4.8 presents a nonstandard error analysis of velocity evaluation which requires fewer derivatives of the vorticity and leads to an efficient new balance between quadrature and smoothing.

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.

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:

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

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

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

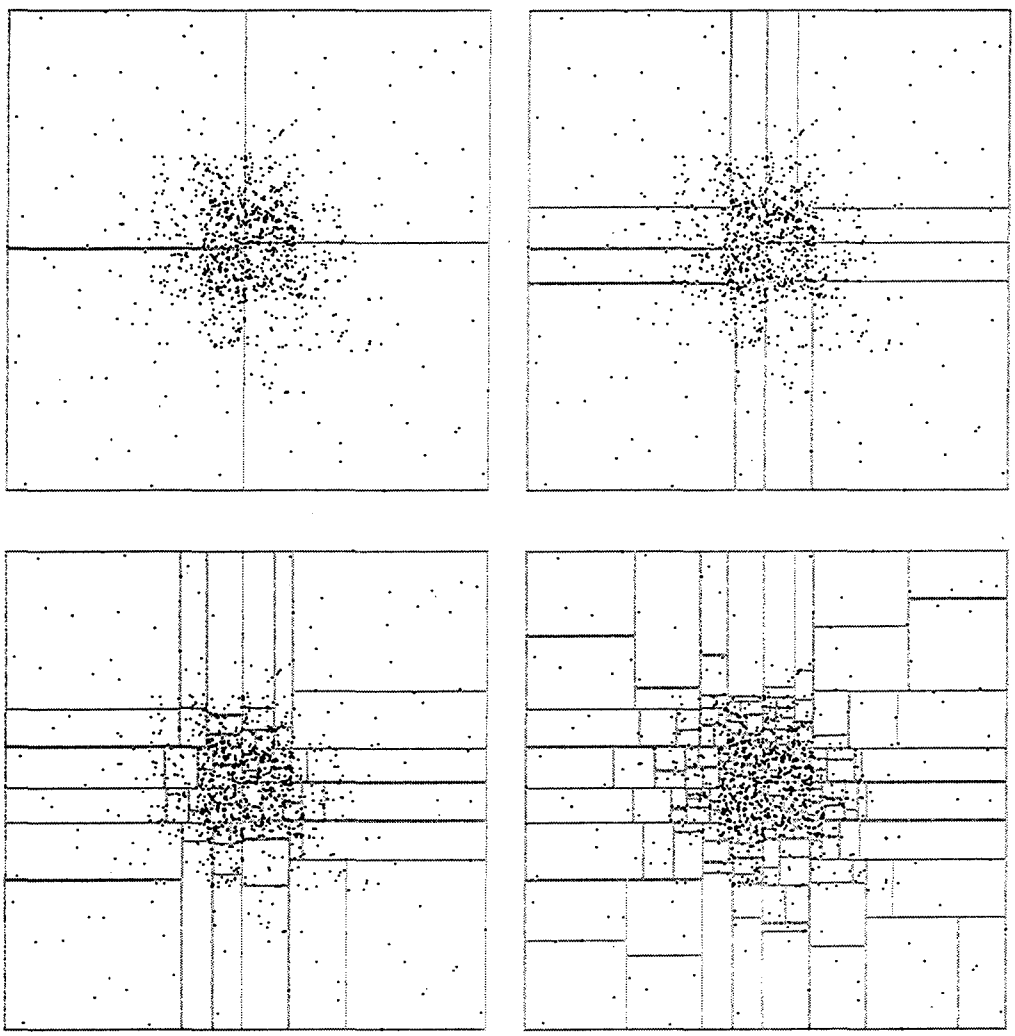


Figure 3: Levels 1, 3, 5 and 7 in the cell data structure with $N = 1000$ nonuniformly distributed points. Here each level-7 cell contains either 4 or 5 points, suitable for a quadrature rule of order $q = 2$ since $q(q+1)/2 = 3$.

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 $p \geq m$, 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 Figure 4.

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

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

This gives an 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 Section 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 ([21]) *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 $0 \leq \alpha + \beta \leq q - 1$. Then for any C^q function g on B, the quadrature error*

$$E = \int_B g(z)dxdy - \sum_{j=1}^N W_j g(z_j)$$

satisfies the bound

$$|E| \leq \Omega\, |B| \frac{h^q}{q!} \|g\|_{C^q(B)}$$

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 $|B| = (b-a)(d-c)$ 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 $\Omega^i = 1+\frac{1}{|B_i|}\sum_{z_j\in B_i}|W_j|$, we can reduce the global condition number $\Omega = \sum_i \Omega_i$. 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 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. Values of S typically range from 1 to 2.

4.5 Smoothing error bounds

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

Theorem 3 ([18]) *Assume the compactly supported core function φ satisfies the moment conditions*

$$\begin{aligned} \int \varphi &= 1, \\ \int x^\alpha y^\beta \varphi &= 0, \quad 1 \leq \alpha+\beta \leq m-1, \\ M = \frac{1}{m!}\int |z|^m |\varphi| &< \infty. \end{aligned} \tag{12}$$

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

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

Quadrature Algorithm

Set parameters:
  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 cell: $p = N/2^L$.
  Maximum cell condition number: Ω_m.
Construct cell data structure:
  $B_1 = B$, a rectangle enclosing all the points z_i.
  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 double cell B^I.
    end if
  end do

Figure 4: Order-q quadrature with N points z_j in a rectangle B.

Multiply by $\varphi_\delta(z)$, integrate and use the moment conditions (12):

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

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

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

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 moment conditions (12) become

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

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 function" 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.

We now construct a family of shape functions f. A convenient ansatz suggested by [16] is

$$f(r) = \varrho^p \left[a_d \varrho^d + \cdots + a_0\right] + 1 \tag{13}$$

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

$$\begin{aligned}\varphi(r) &= \frac{1}{2\pi r} f'(r) \\ &= \frac{-1}{\pi}\left[(p+d)a_d\varrho^{p+d-1} + \cdots + pa_0\varrho^{p-1}\right] \end{aligned} \tag{14}$$

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 $d \geq n$. If $d > n$, the linear system of moment conditions is underdetermined, and we use a complete orthogonal factorization routine to find the solution with smallest 2-norm.

A brief calculation shows that the moment conditions are equivalent to a linear system

$$Aa = b$$

where $b_0 = -1$, $b_i = 0$ for $i > 0$, $a = (a_0, a_1, \ldots, a_d)$ 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.

Given the coefficients a_i, we have

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

where $z^\perp = (-y, x)$. Thus we expect roundoff problems when $r \ll \delta$. They 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

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

The coefficients b_j are given by

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

and

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

Several well-known core functions are included in this scheme. For example, Nordmark's eighth-order core function from [16] has $p = 10$, $d = n = 3$ and $m = 8$: the corresponding shape factor is

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

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

The polynomial degree d makes little difference to the values of high-order kernels, but Table 1 shows that increasing d can noticeably reduce the sizes of the coefficients and thus the smoothing error bound. Indeed,

$$\begin{aligned} M &= \frac{1}{m!}\int |z|^m |\varphi(z)| dxdy \\ &\le \frac{1}{\pi m!}\int_0^1 r^{m+1}\left[(p+d)|a_d|\varrho^{p+d-1} + \cdots + p|a_0|\varrho^{p-1}\right] dr \\ &\le \frac{1}{4\cdot m!}\left[|a_d| + \cdots + |a_0|\right]. \end{aligned}$$

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	$5.0-15$

Table 1: Error constants M as a function of moment order m, smoothness p and polynomial degree d for the piecewise polynomial shape factors (13) shown in Figure 5.

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 methods. These methods require an accurate procedure for computing the s starting values. Suppose we use an explicit s-step Adams method with a fixed time step Δ_f. We begin with a tiny time step $\Delta_i << \Delta_f$ 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 $t = \Delta_f$.

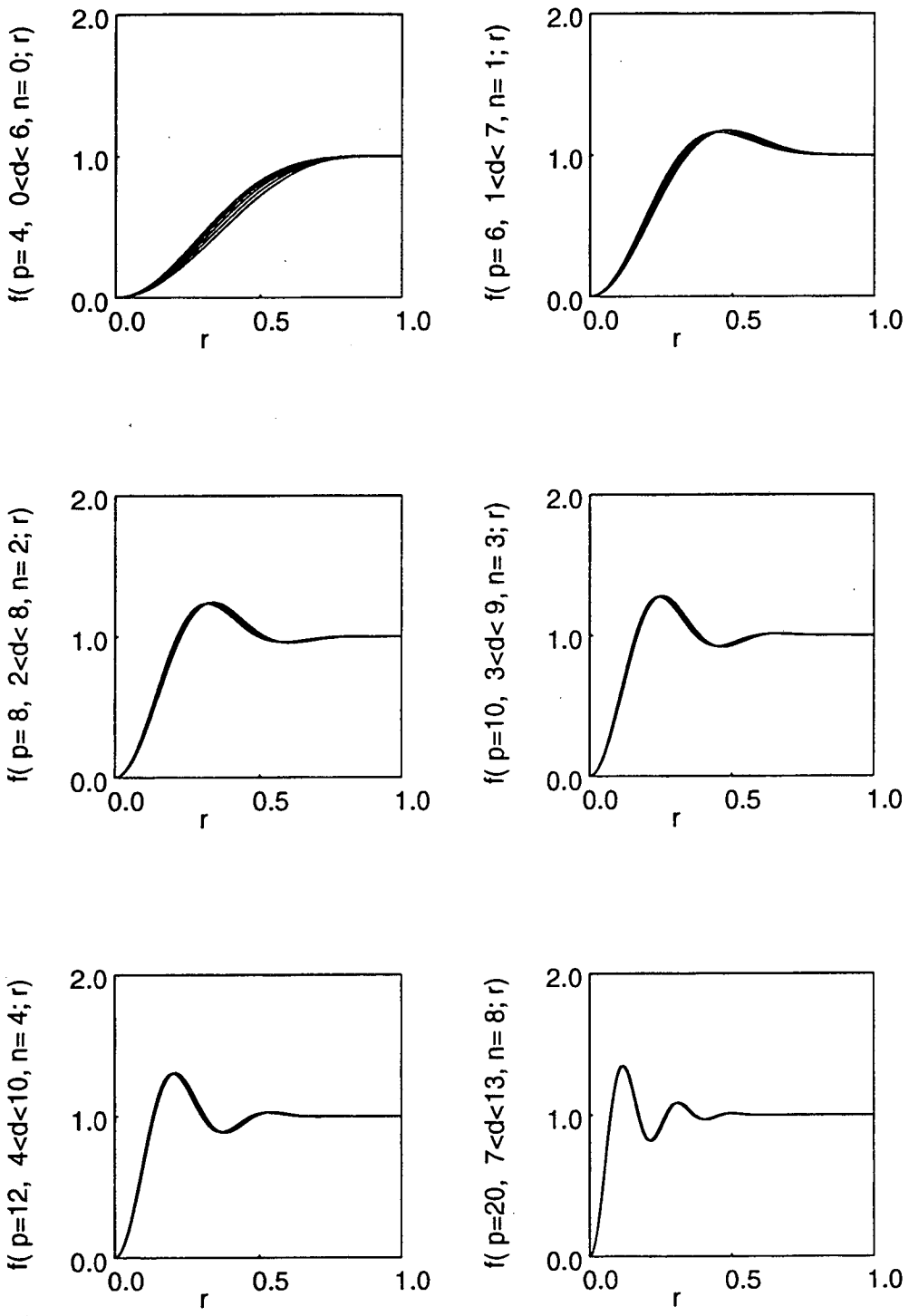


Figure 5: Piecewise polynomial shape factors f with various parameters.

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

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

Here E_δ is the smoothing error, which satisfies

$$E_\delta \leq M\delta^m \|u\|_m$$

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

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

for each fixed z. By a standard inequality for the C^q norm of a product of two functions [14], we have

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

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

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

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

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

so the quadrature error satisfies

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

Hence the total error in one velocity evaluation satisfies

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

where q is the order of quadrature and $\varphi \in C^q$ satisfies moment conditions (12) of order m.

We now take advantage of the separation between $\|\omega\|_0$ and $\|\omega\|_q$ to derive a nonstandard error bound. We choose δ as a function of h to make

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

where ϵ is a user–specified error tolerance, fixed as h vanishes. This implies

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

and our error bound becomes

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

The choice $m = q$ balances the two remaining terms, since an elementary calculation with the Biot-Savart integral shows that $\|u\|_m \le C\|\omega\|_m$ if ω has compact support, and we find

$$E \le C\left[\epsilon\|\omega\|_0 + h^k(\|\omega\|_q + \|u\|_q)\right] \quad = O(\epsilon + h^k)$$

where $k = q^2/(q+1) = q - 1 + \frac{1}{q+1} > q - 1$. 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 $b = 1 + \frac{1}{q+1} = 1.33, 1.20, 1.14, 1.11, 1.09$, very close to 1.

This error bound is nonstandard but extremely useful. It gives almost optimal accuracy and efficiency at the price of a nonstandard definition of convergence. 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 (1). An order $O(\epsilon + h^k)$ error in the velocity u at each time step fortunately does not accumulate in the multistep solution of

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

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

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

as h and Δ_f vanish. This would imply similar estimates for the velocity and vorticity by standard arguments [13].

5 Implementation and numerical results

We implemented a version of the fast adaptive vortex method in Fortran 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 Velocity evaluation

We studied the accuracy of the velocity evaluation of orders $k = 1.33, 3.20, 5.14$ and 7.11 corresponding to $m = q = 2, 4, 6$, and 8, using the well-known Perlman test case [17]

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

where $P = 10$. 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

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

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.

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

Free-Lagrangian Algorithm

Read parameters from input file:

Plotting, output, housekeeping.
Time stepping: t_i, t_f, Δ_i, Δ_f, R, k.
Quadrature order and safety factor: q, S.
Smoothing: p, d, n, $\delta = Ch^a$.
Fast summation tolerance: ϵ.
Initial vorticity ω_0, grid points z_i and domain.

Construct initial grid, vorticity values, shape factor coefficients.

Set $t = t_i$ and time step $\Delta = \Delta_i/R$.
Do while $t < t_f$:
Compute new time step $\Delta = \min(R\Delta, \Delta_f, t_f - t)$.
Compute new order $k = \min(k, j)$.
Evaluate quadrature weights w_j by method of Figure 4.
Apply fast multipole method with smoothing to get
$u_i = \sum_{j=1}^{N} K_\delta(z_i - z_j) w_j \omega_j$ for $1 \le i \le N$
Estimate error, write output, store data and plot results.
Calculate VSVO Adams coefficients.
Update velocity differences.
Advance vortices by one order-k Adams step of size Δ
$z_i = z_i + \Delta(a_1 u_i + \text{differences})$ for $1 \le i \le N$
End while

Figure 6: Outline of a free-Lagrangian vortex method with quadrature, fast summation and Adams time-stepping.

vortices, using core functions and quadratures of orders $m = q = 2, 4, 6, 8$. The number of correct bits

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

in the computed velocity $u_{h,\delta}$ in L^1 and L^∞ norms, the CPU times T (in seconds on a Sparc-2 workstation) and other statistics are reported in Table 2. The velocity evaluation produces error $O(\epsilon + N^{-k/2})$ with $k/2 = 0.67$, 1.60, 2.57 and 3.55 in $O(N^b \log \epsilon)$ 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 when N doubles, the average cell size h decreases by a factor $\sqrt{2}$, so we expect to gain $k/2$ bits per line in each table until $O(\epsilon)$ is reached.

For first-order methods, the $O(h^{1.33})$ errors dominate so the $O(\epsilon)$ 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 $O(\epsilon)$ limit has not appeared. After the limit is reached, convergence continues slowly.

$m = q = 2$, $p = 4$, $d = 1$, $k = 1.33$					
N	h	δ	B_1	B_∞	T
500	0.497	0.631	1.95	1.42	4.83
1000	0.328	0.479	2.48	2.03	13.8
2000	0.205	0.351	3.28	2.79	43.6
4000	0.142	0.275	3.91	3.41	142.7
8000	0.089	0.203	4.74	4.26	336.2
16000	0.064	0.163	5.38	4.88	1155
32000	0.039	0.118	6.29	5.79	2051
64000	0.028	0.095	6.93	6.41	6493

$m = q = 4$, $p = 6$, $d = 2$, $k = 3.20$					
N	h	δ	B_1	B_∞	T
500	1.300	1.481	1.02	0.52	6.87
1000	0.807	1.011	2.26	1.79	22.2
2000	0.443	0.625	4.49	3.71	75
4000	0.300	0.457	6.00	5.21	209
8000	0.180	0.305	8.16	7.22	632
16000	0.128	0.232	9.71	9.00	1485
32000	0.078	0.156	11.9	10.3	4498
64000	0.057	0.121	13.3	12.0	8111

$m = q = 6$, $p = 8$, $d = 3$, $k = 5.14$					
N	h	δ	B_1	B_∞	T
500	1.760	1.948	0.0	0.0	8.66
1000	1.170	1.374	0.31	0.0	26.9
2000	0.721	0.905	4.49	2.67	90.6
4000	0.386	0.529	8.21	5.51	281
8000	0.263	0.381	10.2	7.18	774
16000	0.158	0.245	12.0	6.74	1574
32000	0.114	0.185	14.0	9.30	4988
64000	0.068	0.118	15.0	10.1	7634

$m = q = 8$, $p = 10$, $d = 4$, $k = 7.11$					
N	h	δ	B_1	B_∞	T
500	1.810	2.033	1.34	0.48	9.22
1000	1.690	1.912	0.0	0.0	34.9
2000	1.100	1.304	3.34	2.23	111
4000	0.677	0.848	7.79	6.29	358
8000	0.362	0.486	10.4	6.89	960
16000	0.247	0.346	11.9	8.00	2923
32000	0.146	0.217	12.9	8.15	5828
64000	0.106	0.162	15.1	9.99	14650

Table 2: Velocity evaluation errors in ω_8 with N adaptive random points. Correct bits B_1 and B_∞ in L_1 and L^∞, CPU times T, cell size h and core radius δ. Here q is the quadrature order and m is the moment order.

5.2 Long-time accuracy

We also tested the long-time accuracy of the method on several Perlman test cases, 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 $n^2 \approx 0.8N$, and used core functions, quadratures and Adams methods of orders $m = q = s = 2$, 4 and 6, yielding adaptive vortex methods of orders $k = 1.33$, 3.20 and 5.14. We tested each method on a Perlman patch of minimal smoothness, with $P = q + 1 = 3$, 5 and 7. In particular, the errors at different orders are unrelated. Table 5 shows the other numerical parameters. For comparison, Tables 3 and 4 give the parameters used in the standard and triangulated vortex methods, for the test cases plotted in Figures 1 and 2.

The correct bits in L^1 in the velocity and vorticity are plotted in Figure 7. The plots are individually scaled and ticked in such a way that the number of correct bits should increase by half a tick mark at each line. 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 Figure 1). The errors are highly oscillatory on a small scale, because a new quadrature rule is built from scratch at each step.

N	h	δ	Δ_f	Δ_i	T	$B_1(u_7)$	$B_1(\omega_7)$	$B_1(u_3)$	$B_1(\omega_3)$
64	0.27	1.05	0.512	0.0512	0.5	3.77	1.00	3.85	1.37
256	0.14	0.74	0.256	0.0128	1.4	4.42	1.40	4.86	2.89
1024	0.07	0.52	0.128	0.0032	12.0	7.52	4.10	6.35	5.19
4096	0.03	0.37	0.064	0.0008	85.6	8.29	6.49	9.02	8.17

Table 3: Number of vortices N, mesh size h at $t = 0$, core radius δ, time steps Δ_f and Δ_i and CPU time T per step in seconds for the standard vortex method. Here $B_1(u_P)$ and $B_1(\omega_P)$ are measured at $t = 20$.

N	Δ_f	T	$B_1(u)$	$B_1(\omega)$
70	0.625	0.86	4.76	4.61
225	0.41666	5.1	5.97	6.27
745	0.3125	23.9	8.48	9.48
2729	0.20833	120.2	10.15	11.70

Table 4: Number of vertices N, time step Δ_f and CPU time T per step for the triangulated vortex method. Here $B_1(u)$ and $B_1(\omega)$ are measured at $t = 20$.

$m = q = 2$, $p = 4$, $d = 1$, $k = 1.33$								
N	n	Δ_f	Δ_i	h	δ	T	$B_1(u)$	$B_1(\omega)$
250	14	1.12	0.056	0.840	0.807	0.58	2.94	2.85
500	20	0.80	0.040	0.516	0.599	3.29	3.75	4.21
1000	28	0.56	0.028	0.345	0.441	6.63	4.78	6.02
2000	40	0.40	0.020	0.198	0.321	20.9	5.73	7.14
4000	56	0.28	0.014	0.136	0.231	35.4	6.64	8.41
8000	80	0.20	0.007	0.076	0.173	112	7.26	9.84

$m = q = 4$, $p = 6$, $d = 2$, $k = 3.20$								
N	n	Δ_f	Δ_i	h	δ	T	$B_1(u)$	$B_1(\omega)$
250	14	0.28	0.01024	1.97	1.33	0.66	0.75	0.68
500	20	0.20	0.00512	1.02	0.93	4.18	2.93	4.20
1000	28	0.14	0.00256	0.67	0.64	12.9	6.16	6.00
2000	40	0.10	0.00128	0.40	0.44	27.1	7.97	7.66
4000	56	0.07	0.00064	0.27	0.29	84.1	8.56	9.01
8000	80	0.05	0.00032	0.15	0.21	139	10.1	10.2

$m = q = 6$, $p = 8$, $d = 3$, $k = 5.14$								
N	n	Δ_f	Δ_i	h	δ	T	$B_1(u)$	$B_1(\omega)$
250	14	0.14	0.01024	2.00	3.07	1.08	0.0	0.0
500	20	0.10	0.00512	1.97	1.82	6.96	0.46	0.21
1000	28	0.07	0.00128	1.06	1.24	19.9	2.94	3.87
2000	40	0.05	0.00032	0.68	0.83	59.9	6.64	8.39
4000	56	0.035	0.00008	0.41	0.55	197	9.06	9.27
8000	80	0.025	0.00002	0.26	0.38	365	10.6	12.0

Table 5: Number of vortices N (with n^2 in regular grid), time steps Δ_f and Δ_i, mesh size h and core radius δ at $t = 0$ and CPU time T per step for adaptive methods of orders 1.33 (top), 3.20 (center) and 5.14 (bottom). Here $B_1(u)$ and $B_1(\omega)$ are measured at $t = 20$.

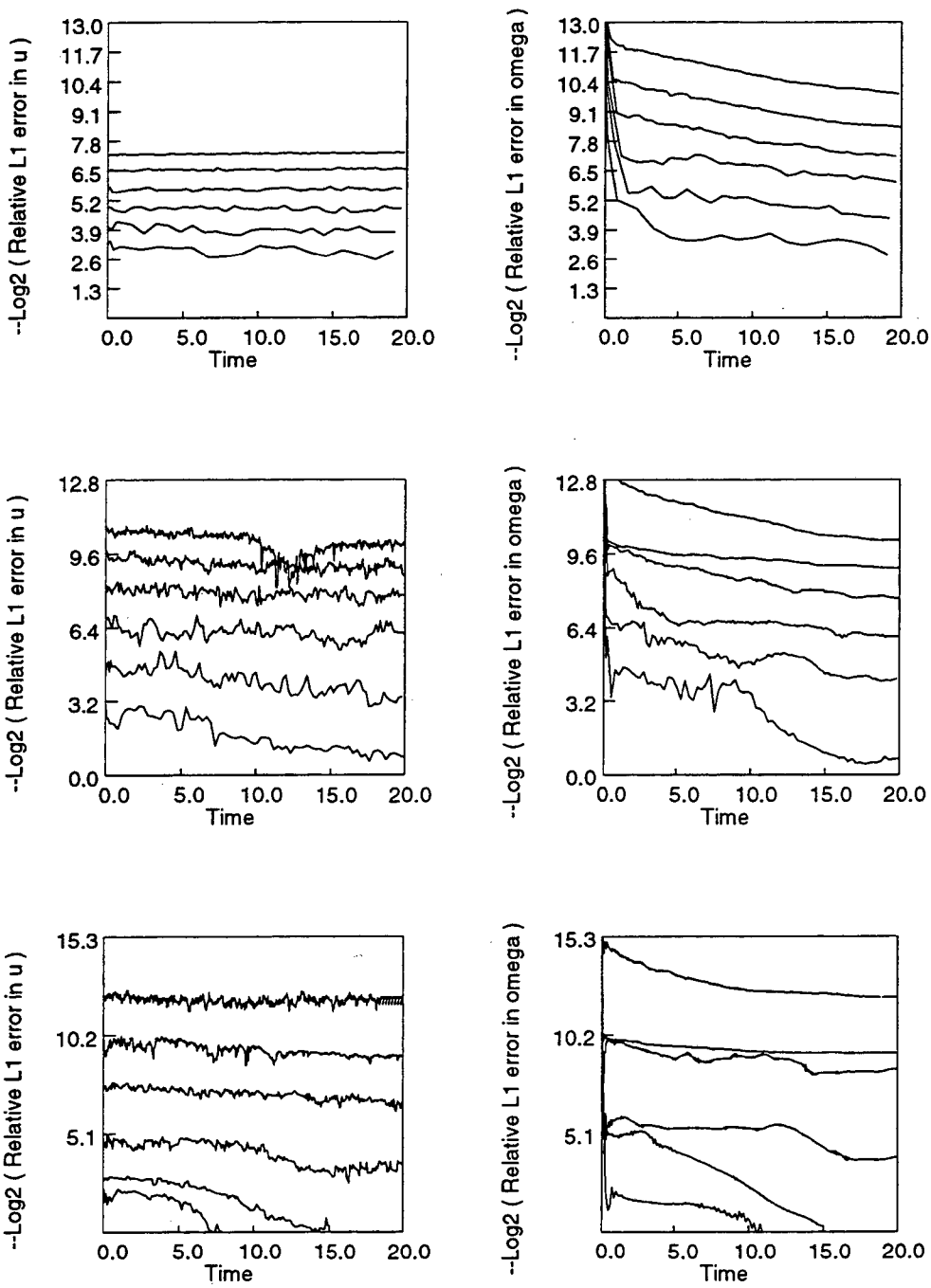


Figure 7: Correct bits $B_1(u)$ and $B_1(\omega)$ in velocity u (left column) and vorticity ω (right column), for adaptive vortex methods of orders 1.33 (top row), 3.20 (second row) and 5.14 (last row). The numerical parameters are given in Table 5.

5.3 Interacting vortex patches

As a more complex example, we used the order-1.33 method with parameters given in Table 6 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 8 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 Figure 9. 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.

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

Table 6: Number of vortices N, mesh size h at $t = 0$, core radius δ, time steps Δ_f and Δ_i and CPU time T per step 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 $t = 30$.

References

[1] C. Anderson and C. Greengard. On vortex methods. *SIAM J. Math. Anal.*, 22:413–440, 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.*, 62:111–127, 1986.

[3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

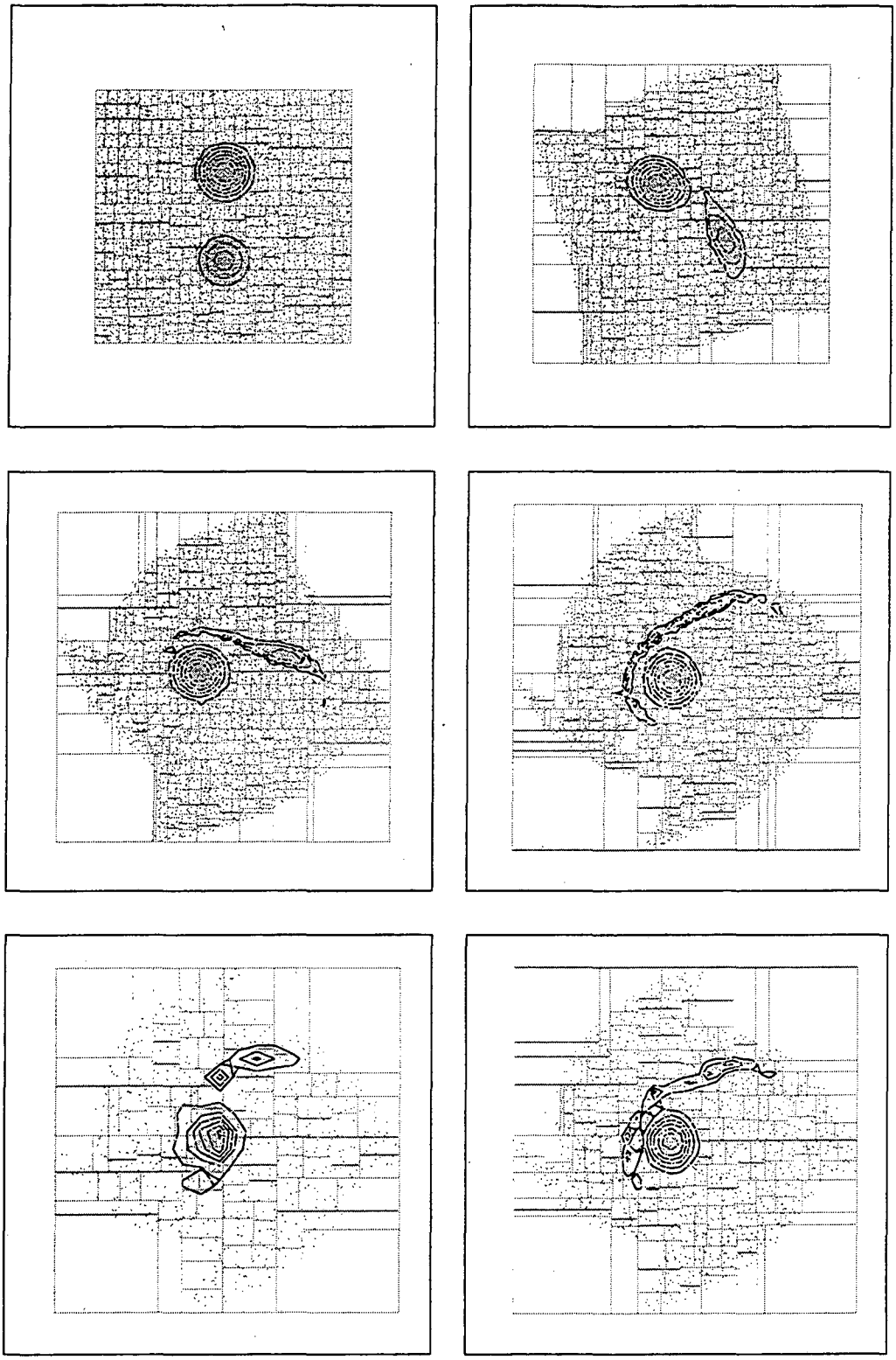


Figure 8: Vorticity contours for two interacting Perlman patches with $P = 3$, computed with the adaptive method of order 1.33 and numerical parameters given in Table 6. The first four plots show the evolution at $t = 0$, 10, 20 and 30 with $N = 4000$ vortices, the last row shows the final frame $t = 30$, computed with $N = 1000$ (left) and $N = 2000$(right).

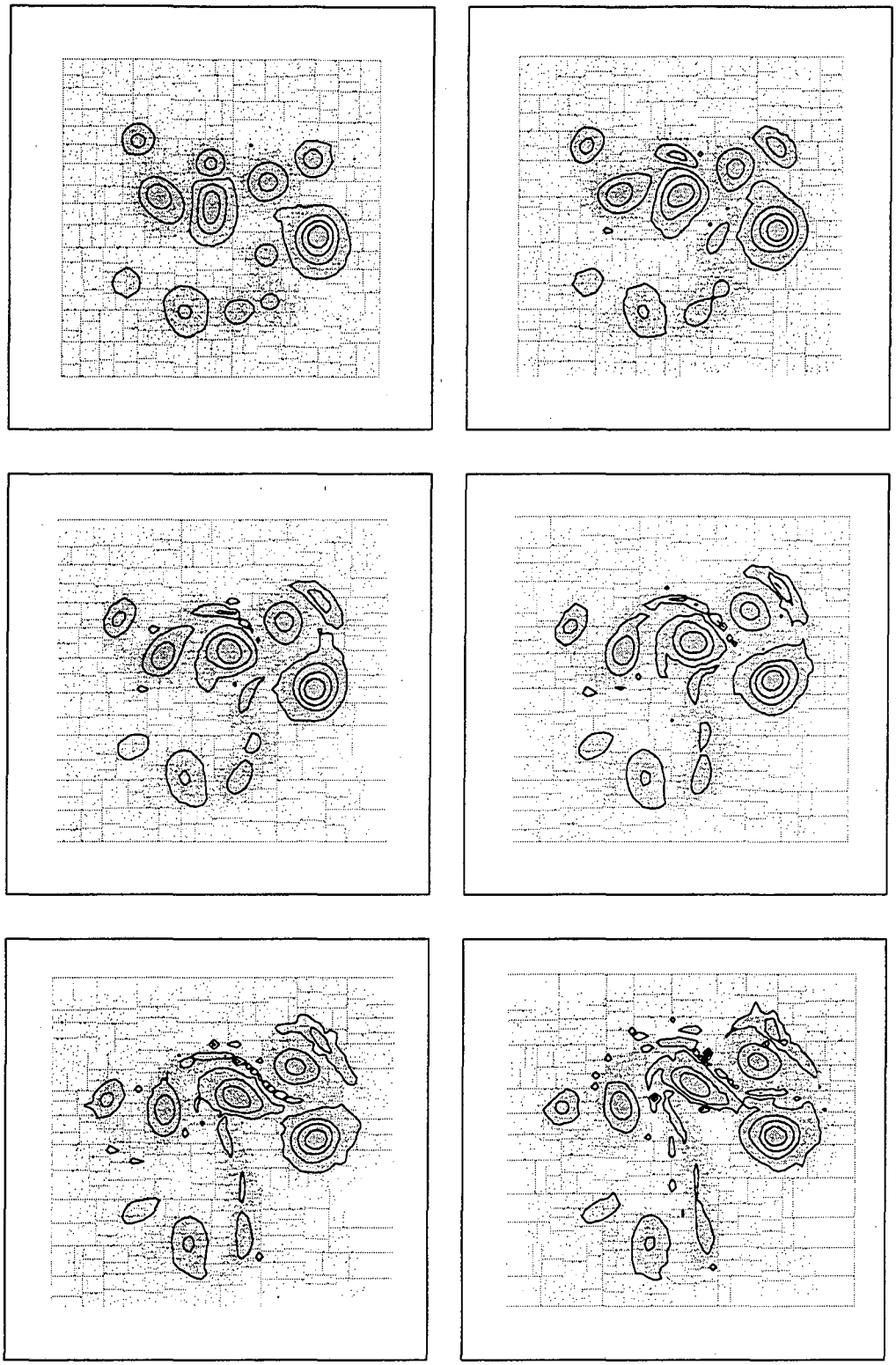


Figure 9: Vorticity contours for 20 interacting Perlman patches with $P = 5$, computed with the adaptive method of order 3.20. Results are shown at $t = 0$, 4, 8, 12, 16 and 20 with $N = 10000$ vortices.

[4] J. T. Beale. On the accuracy of vortex methods at large times. In B. Engquist, M. Luskin, and A. Majda, editors, *Computational fluid dynamics and reacting gas flow*, volume 12 of *IMA volumes in mathematics and applications*. Springer-Verlag, 1988.

[5] J. T. Beale and A. Majda. High order accurate vortex methods with explicit velocity kernels. *J. Comput. Phys.*, 58:188–208, 1985.

[6] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole method for particle simulations. *SIAM J. Sci. Stat. Comput.*, 9:669–686, 1988.

[7] T. Chacon Rebollo and T. Y. Hou. A Lagrangian finite element method for the 2-D Euler equations. *Comm. Pure Appl. Math.*, XLIII:735–767, 1990.

[8] A. J. Chorin. *Computational Fluid Mechanics: Selected Papers*. Academic Press, 1989.

[9] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Computer science and applied mathematics. Academic Press, second edition, 1984.

[10] 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-Savrt integral in three dimensions. Technical report, NUWC, 1994.

[11] K. E. Gustafson and J. A. Sethian, editors. *Vortex methods and vortex motion*. SIAM, Philadelphia, 1991.

[12] O. H. Hald. Convergence of vortex methods for Euler's equations III. *SIAM J. Numer. Anal.*, 24:538–582, 1987.

[13] O. H. Hald. Convergence of vortex methods. In Gustafson and Sethian [11], pages 33–58.

[14] L. Hörmander. The boundary problems of physical geodesy. *Arch. Rational Mech. Analysis*, 62:1–52, 1976.

[15] S. A. Huyer and J. R. Grant. Incorporation of boundaries for 2D triangular vorticity element methods. Technical report, NUWC, 1994.

[16] H. O. Nordmark. Rezoning for high-order vortex methods. *J. Comput. Phys.*, 97:366, 1991.

[17] M. Perlman. On the accuracy of vortex methods. *J. Comput. Phys.*, 59:200–223, 1985.

[18] P. A. Raviart. An analysis of particle methods. In F. Brezzi, editor, *Numerical methods in fluid dynamics (Lecture notes in mathematics; 1127) Fondazione C.I.M.E., Firenze.* Springer-Verlag, 1985.

[19] G. Russo and J. Strain. Fast triangulated vortex methods for the 2-D Euler equations. *J. Comput. Phys.*, 111:291–323, 1994.

[20] J. Strain. Fast potential theory II: Layer potentials and discrete sums. *J. Comput. Phys.*, 99:251–270, 1992.

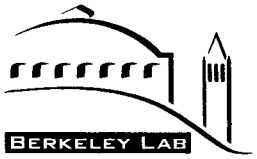
[21] J. Strain. Locally-corrected multidimensional quadrature rules for singular functions. *SIAM J. Sci. Comput.*, 16:1–26, 1995.

[22] J. Strain. 2-D vortex methods and singular quadrature rules. *J. Comput. Phys.*, 124:1–23, 1996.

AMS Subject Classifications: 76M10, 76M25, 65M50, 65M60, 65Y25, 65D32, 65D05, 65D30, 65R20

Key words and phrases: quadrature, vortex methods, Euler equations, Legendre polynomials, least-squares problems, quadtrees, data structures, free-Lagrangian methods, adaptive methods, interpolation, product integration.

E-mail address: strain@math.berkeley.edu.



ERNEST ORLANDO LAWRENCE BERKELEY NATIONAL LABORATORY
TECHNICAL AND ELECTRONIC INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA | BERKELEY, CALIFORNIA 94720