Journal of Computational Physics **172,** 879–907 (2001)
doi:10.1006/jcph.2001.6862, available online at http://www.idealibrary.com on IDEAL®

# A Particle Method and Adaptive Treecode for Vortex Sheet Motion in Three-Dimensional Flow

Keith Lindsay* and Robert Krasny†

**National Center for Atmospheric Research, P. O. Box 3000, Boulder, Colorado 80307-3000; and †Department of Mathematics, University of Michigan, Ann Arbor, Michigan 48109-1109*
E-mail: *klindsay@cgd.ucar.edu, †krasny@math.lsa.umich.edu

Received September 18, 2000; revised June 19, 2001

A particle method is presented for computing vortex sheet motion in three-dimensional flow. The particles representing the sheet are advected by a regularized Biot–Savart integral in which the exact singular kernel is replaced by the Rosenhead–Moore kernel. New particles are inserted to maintain resolution as the sheet rolls up. The particle velocities are evaluated by an adaptive treecode algorithm based on Taylor approximation in Cartesian coordinates, and the necessary Taylor coefficients are computed by a recurrence relation. The adaptive features include a divide-and-conquer evaluation strategy, nonuniform rectangular clusters, variable-order approximation, and a run-time choice between Taylor approximation and direct summation. Tests are performed to document the treecode's accuracy and efficiency. The method is applied to simulate the roll-up of a circular-disk vortex sheet into a vortex ring. Two examples are presented, azimuthal waves on a vortex ring and the merger of two vortex rings. 

*Key Words:* particle method; adaptive treecode; vortex sheet; vortex ring; three-dimensional flow.

## 1. INTRODUCTION

Vortex sheets are widely used in fluid dynamics to model thin shear layers in slightly viscous flow. In this case, the shear layer is replaced by a jump discontinuity across the sheet surface and the evolution of the layer is reduced to tracking the self-induced motion of the sheet. The present work develops a Lagrangian particle method for computing vortex sheet motion in three-dimensional flow. There is a large body of work dealing with the application of such methods to three-dimensional vortex dynamics, and several survey articles and texts can be consulted for an overview [1–5].



0021-9991/01 $35.00

We start by describing a number of special difficulties that arise in vortex sheet computations. In two-dimensional flow, the point vortex method replaces a continuous vortex sheet by a set of discrete point vortices [6, 7], but the method fails to converge past a finite critical time when a curvature singularity forms in the underlying sheet [8–11]. One way to proceed is to regularize the problem by applying the vortex-blob method, and this approach captures the spiral roll-up of the vortex sheet past the critical time [12–14]. Aside from the issue of singularity formation, another difficulty arises because perturbations introduced by machine roundoff error are amplified by Kelvin–Helmholtz instability, leading to the rapid loss of computational accuracy [7, 10]. This can be prevented by using higher precision arithmetic or filtering [10], and more generally the severity of the problem can be reduced by applying the vortex-blob method [14]. There is also a difficulty due to the fact that the sheet is a material surface and its shape can become quite convoluted in time. Unless some precaution is taken, the numerical resolution of the surface will deteriorate. Here too the vortex-blob method can be applied, to limit the deformation of the surface, but there is still a need for an efficient method of resolving the surface, especially in the case of three-dimensional flow. One remedy represents the sheet as a collection of vortex filaments and uses cubic spline interpolation to insert new particles as the filaments deform [15]. Another approach represents the sheet as a triangulated surface and uses panel methods to perform adaptive mesh refinement [16–18]. Here we will represent the sheet as a set of material lines, and we maintain resolution by using local cubic polynomial interpolation to insert new particles and material lines as the sheet rolls up [19].

Another important issue, and the main topic of this work, is the large amount of CPU time required in vortex sheet computations. To explain how this arises we need to briefly describe our numerical method. The sheet is represented at the discrete level by a set of particles $\mathbf{x}_i(t)$ with vector-valued weights $\mathbf{w}_i$, for $i = 1, \ldots, N$, and the particles are advected by the equations

$$\frac{d\mathbf{x}_i}{dt} = \sum_{j=1}^{N} \mathbf{K}_\delta(\mathbf{x}_i, \mathbf{x}_j) \times \mathbf{w}_j, \tag{1}$$

where

$$\mathbf{K}_\delta(\mathbf{x}, \mathbf{y}) = -\frac{1}{4\pi} \frac{\mathbf{x} - \mathbf{y}}{(|\mathbf{x} - \mathbf{y}|^2 + \delta^2)^{3/2}} \tag{2}$$

is the Rosenhead–Moore kernel, a regularized form of the Biot–Savart kernel [20, 21]. This is an example of a vortex-blob method wherein $\delta$ is an artificial smoothing parameter. The exact kernel is recovered by setting $\delta$ to zero, but the simulations use a nonzero value in order to overcome the problems mentioned above due to singularity formation, Kelvin–Helmholtz instability, and surface resolution. More details about the discretization will be given below. The Rosenhead–Moore kernel is popular in three-dimensional vortex sheet computations [15–18], but other expressions are sometimes also used [22, 23].

Evaluating the sum in (1) for $i = 1, \ldots, N$ is an example of an $N$-body problem, analogous to problems involving point charges or point masses in classical physics [24]. The simplest evaluation procedure is direct summation, but this requires computing $O(N^2)$

particle–particle interactions and hence is prohibitively expensive when $N$ is large. Several approaches have been developed to reduce the computational cost, and in the context of three-dimensional vortex sheet motion these include a vortex-in-cell method [25] and a level-set method [26]. The present work takes a different approach using the concept of a treecode algorithm.

In a treecode algorithm, the particles are divided into a nested set of clusters and the $O(N^2)$ particle–particle interactions are replaced by a smaller number of particle–cluster interactions which can be efficiently evaluated using a multipole approximation. The earliest treecode algorithms used a monopole approximation and a divide-and-conquer evaluation strategy [27, 28]. The fast multipole method (FMM) uses higher order approximations, either a Laurent series in two dimensions or a classical multipole expansion involving spherical harmonics in three dimensions [29, 30]. The FMM also employs a more elaborate evaluation procedure in which the far-field multipole approximation is converted to a local approximation. Treecode algorithms reduce the operation count to $O(N \log N)$ or $O(N)$. They have had great impact in particle simulations and there is ongoing interest in optimizing their performance [31–39].

Treecodes are a natural choice to consider for evaluating the sums in (1), but there is an obstacle; the Rosenhead–Moore kernel (2) is nonharmonic when $\delta > 0$ and so it cannot be expanded in a classical multipole series. One way of overcoming this problem is to replace (2) by a different form of regularized kernel in which the smoothing effect decays more rapidly with distance. Then direct summation can be applied to compute regularized particle–particle interactions inside a specified cutoff radius, while a treecode using a classical multipole expansion can be applied outside the cutoff radius where the regularization effectively vanishes [23, 40].

Here we want to retain the Rosenhead–Moore kernel and in order to do so we develop a treecode algorithm based on Taylor approximation in Cartesian coordinates rather than a classical multipole expansion involving spherical harmonics. The Taylor approximation converges because the kernel (2) is real analytic, and moreover, the necessary Taylor coefficients can be efficiently computed by a recurrence relation. This approach was originally developed for vortex sheet computations in two-dimensional flow [41, 42] and was later extended to the case of three-dimensional flow [43a, 43b]. The algorithm uses a divide-and-conquer strategy to evaluate the particle velocities [27, 28] and it employs several adaptive techniques to gain efficiency. The tree consists of nonuniform rectangular clusters adapted to the particle distribution. For each particle–cluster interaction, the order of approximation is chosen adaptively, and a run-time choice is made between Taylor approximation and direct summation based on empirical estimates of the required CPU time. Tests are performed to document the treecode's accuracy and efficiency, and the results show that the algorithm is significantly faster than direct summation for systems having a large number of particles. The particle method and adaptive treecode algorithm are then applied to simulate the roll-up of a circular-disk vortex sheet into a vortex ring. Two examples are presented, the growth of azimuthal waves on a vortex ring and the merger of two vortex rings moving side by side.

The paper is organized as follows. Section 2 describes the particle method for computing vortex sheet motion in three-dimensional flow. Section 3 presents the treecode algorithm for evaluating the particle velocities and Section 4 documents the algorithm's accuracy and efficiency on a test case. Section 5 presents the vortex sheet simulations. The work is summarized in Section 6.

## 2. PARTICLE METHOD

### 2.1. Lagrangian Formulation

The simulations are based on the Lagrangian formulation of vortex sheet motion in three-dimensional flow [44–46]. For the intended application to vortex rings we view the sheet as a parametrized surface $\mathbf{x}(\Gamma, \theta, t)$ composed of closed material lines, where $\Gamma$ measures circulation across the lines and $\theta$ is a $2\pi$-periodic parameter along the lines. Figure 1 depicts the initial parametrization of a circular-disk vortex sheet defined in terms of Cartesian coordinates, $\mathbf{x}(\Gamma, \theta, 0) = (x_1, x_2, x_3)$, where

$$x_1 = (1 - \Gamma^2)^{1/2} \cos\theta, \quad x_2 = (1 - \Gamma^2)^{1/2} \sin\theta, \quad x_3 = 0, \tag{3}$$

and $0 \le \Gamma \le 1$, $0 \le \theta \le 2\pi$. Equation (3) describes the bound vortex sheet associated with potential flow past a circular disk. The lines of constant circulation $\Gamma$ correspond to vortex lines. Note that $\Gamma = (1 - r^2)^{1/2}$, where $r = (x_1^2 + x_2^2)^{1/2}$ is the radial coordinate of a point on the sheet measured from the center $r = 0$. The square-root singularity in $\Gamma(r)$ at $r = 1$ causes the edge of the sheet to roll up into a spiral for $t > 0$.

With this parametrization, the equation governing the motion of the sheet is

$$\frac{\partial \mathbf{x}}{\partial t} = \int_0^{2\pi} \int_0^1 \mathbf{K}_\delta(\mathbf{x}, \tilde{\mathbf{x}}) \times \frac{\partial \tilde{\mathbf{x}}}{\partial \theta} d\tilde{\Gamma} d\tilde{\theta}, \tag{4}$$

where $\mathbf{x} = \mathbf{x}(\Gamma, \theta, t)$, $\tilde{\mathbf{x}} = \mathbf{x}(\tilde{\Gamma}, \tilde{\theta}, t)$, and $\mathbf{K}_\delta(\mathbf{x}, \tilde{\mathbf{x}})$ is the Rosenhead–Moore kernel. The right side of (4) is a regularized form of the Biot–Savart integral specialized to the case of a vortex sheet, and the equation states that the sheet is a material surface moving in its self-induced velocity field. The term $\partial\tilde{\mathbf{x}}/\partial\theta$ accounts for vortex stretching. With the circular-disk initial condition given in (3), the sheet rolls up into an axisymmetric vortex ring [47, 48]. In Section 5 we perturb this initial condition to obtain examples of fully three-dimensional motion.

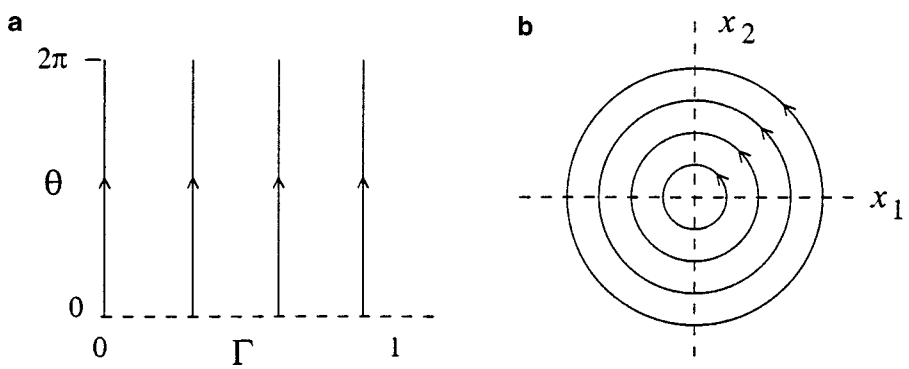


**FIG. 1.** Parametrization of a circular-disk vortex sheet (3). (a) parameter space $(\Gamma, \theta)$; (b) physical space $(x_1, x_2, x_3)$. $\Gamma$ measures circulation across the vortex lines and $\theta$ is a $2\pi$-periodic parameter along the lines. The sheet lies in the plane $x_3 = 0$.

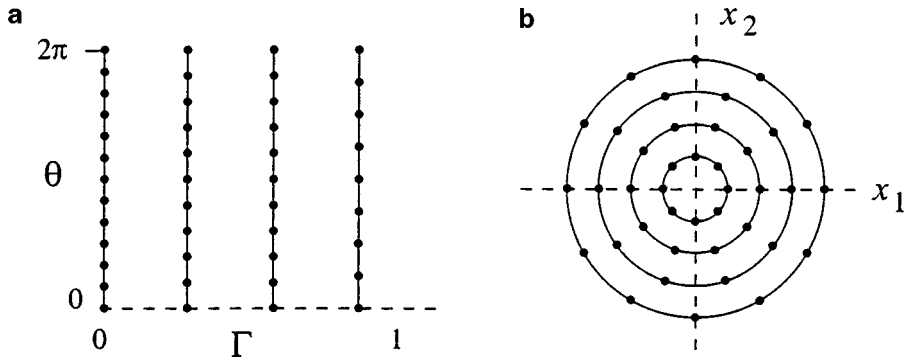


**FIG. 2.** Discretization of a circular-disk vortex sheet into particles (●). (a) parameter space ($\Gamma, \theta$); (b) physical space ($x_1, x_2, x_3$).

### 2.2. Discretization

The sheet is represented by a set of Lagrangian particles $\mathbf{x}_i(t), i = 1, \ldots, N$, corresponding to a discretization of the $\Gamma - \theta$ parameter space. The initial discretization for a circular-disk vortex sheet is shown schematically in Fig. 2. First the sheet is discretized in circulation $\Gamma$ to obtain a finite set of material lines and then each line is discretized in $\theta$ to obtain the particles $\mathbf{x}_i(t)$. In practice, the $\Gamma$-mesh is dense near the edge of the disk in order to resolve the spiral roll-up. The $\theta$-mesh is chosen so that the particle spacing along each line is roughly uniform.

Given a set of particles, the integral in (4) is discretized using the trapezoid rule with respect to $\Gamma$ and $\theta$, in that order. This yields a system of ordinary differential equations for the motion of the particles, shown in (1). The particle weights are defined by $\mathbf{w}_j = D_\theta(\mathbf{x}_j)\Delta\Gamma_j\Delta\theta_j$, where $D_\theta(\mathbf{x})$ is a second-order finite-difference approximation to $\partial\mathbf{x}/\partial\theta$, and $\Delta\Gamma_j, \Delta\theta_j$ are the quadrature weights. The system (1) is solved by the fourth-order Runge–Kutta method using the treecode to evaluate the particle velocities.

### 2.3. Particle Insertion Scheme

New particles are inserted during the computation to maintain resolution as the sheet rolls up. This can occur in two ways (a) inserting a particle on a material line (refining the $\theta$-mesh for a given value of $\Gamma$) and (b) inserting particles to create a new material line (refining the $\Gamma$-mesh). The insertion scheme relies on two user-specified parameters, $\epsilon_\theta$ and $\epsilon_\Gamma$.

Figure 3a depicts the scheme for inserting a new particle on a material line [19]. A new particle is inserted between two adjacent particles if the distance between the particles is greater than $\epsilon_\theta$. The $\theta$-value of the new particle is the average $\theta$-value of the two adjacent particles. The location of the new particle is computed using a cubic interpolating polynomial in $\theta$ based on the four adjacent particles, two on each side.

Figure 3b depicts the scheme for inserting particles to create a new material line. A new line is created between two adjacent lines if the maximum distance between the lines is greater than $\epsilon_\Gamma$. The $\Gamma$-value of the new line is the average $\Gamma$-value of the two adjacent lines. The $\theta$-values of the particles on the new line are transferred from an adjacent line. To compute a new particle location for a given $\theta$-value, we use a cubic interpolating polynomial

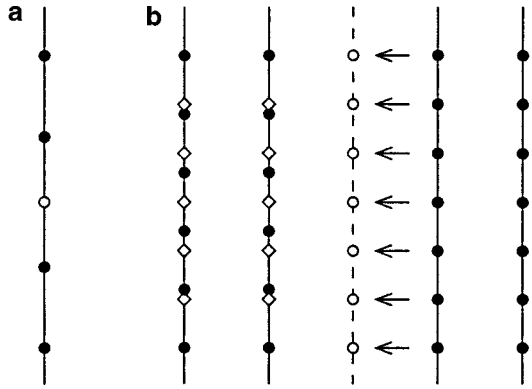


**FIG. 3.** Particle insertion scheme. (a) inserting a new particle on a material line (refining the $\theta$-mesh for a given value of $\Gamma$); (b) inserting particles to create a new material line (refining the $\Gamma$-mesh). Particle (●); new particle (○); auxiliary particle (◇); material line (——); new material line (- - -).

in $\Gamma$ based on particle locations on the four adjacent lines, two on each side. If there is no particle at a given $\theta$-value on these lines, auxiliary particles are created using a cubic interpolating polynomial in $\theta$.

The simulations start from simple initial conditions that are well-resolved by a small number of particles. As the sheet rolls up, many new particles are inserted and direct summation becomes impractical. In the next section we present a treecode algorithm to deal with this problem.

## 3. TREECODE ALGORITHM

### 3.1. Overview

The treecode algorithm is used by the time-integration routine to compute the particle velocities on the right side of (1). There are two main steps (a) constructing the tree for a given set of particles and (b) computing the particle velocities with the aid of the tree. The tree is constructed by subdividing the particles into a nested set of clusters. Once the tree is constructed, each particle velocity is expressed as a sum of particle–cluster interactions

$$\frac{d\mathbf{x}_i}{dt} = \sum_c \sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y}_j) \times \mathbf{w}_j, \tag{5}$$

where $c = \{\mathbf{y}_j,\ j = 1, \ldots, N_c\}$ denotes a cluster of particles and $\mathbf{w}_j$ is the weight associated with $\mathbf{y}_j$. The clusters $c$ appearing in (5) are determined adaptively for each particle $\mathbf{x}_i$ using a divide-and-conquer strategy [27, 28, 36].

Figure 4 depicts a particle $\mathbf{x}_i$ and a disjoint cluster $c$ (there and sometimes below, a two-dimensional schematic is shown). The associated cell is defined to be the smallest rectangular box containing the particles in $c$, and the cell center $\mathbf{y}_c$ is the geometric center of the box. Strictly speaking a *cell* is a region of space containing a *cluster* of particles, but we use the terms interchangeably. The particle–cluster interaction gives the velocity of the particle $\mathbf{x}_i$ induced by the cluster $c$. As explained below, the interactions are evaluated using either Taylor approximation or direct summation.

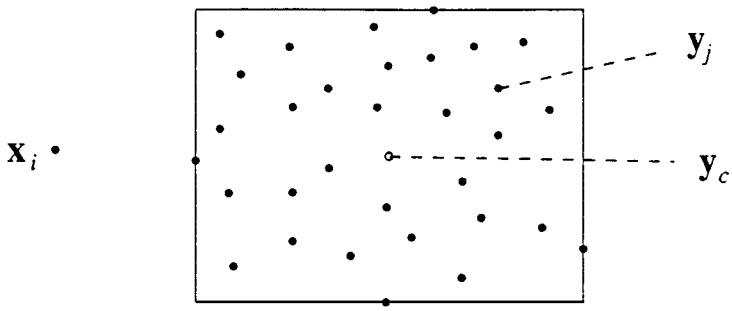


**FIG. 4.** A particle $\mathbf{x}_i$ and a disjoint cluster $c = \{\mathbf{y}_j,\ j = 1, \ldots, N_c\}$. The associated cell is defined to be the smallest rectangular box containing the particles in $c$, and the cell center $\mathbf{y}_c$ is the geometric center of the box.

The remainder of this section describes the components of the treecode algorithm: the Taylor approximation for a particle–cluster interaction (Section 3.2), the recurrence relation for the Taylor coefficients of the regularized kernel (Section 3.3), the criterion for choosing the order of approximation (Section 3.4), the tree construction procedure (Section 3.5), and the divide-and-conquer strategy for evaluating the particle velocities (Section 3.6).

### 3.2. Taylor Approximation

Our aim is to derive a Taylor approximation for a particle–cluster interaction. Referring to the right side of (5), we expand $\mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y}_j)$ in a Taylor series with respect to $\mathbf{y}$ about the cell center $\mathbf{y}_c$. Using Cartesian coordinates and standard multi-index notation, this yields

$$\begin{aligned}\sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y}_j) \times \mathbf{w}_j &= \sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y}_c + (\mathbf{y}_j - \mathbf{y}_c)) \times \mathbf{w}_j \\ &= \sum_{j=1}^{N_c} \sum_{\mathbf{k}} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y}_c)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \times \mathbf{w}_j \\ &= \sum_{\mathbf{k}} \mathbf{a}_{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c) \times \mathbf{m}_{\mathbf{k}}(c), \end{aligned} \tag{6}$$

where $\mathbf{k} = (k_1, k_2, k_3)$ is an integer multi-index with all $k_i \geq 0$,

$$\mathbf{a}_{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c) = \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y}_c) \tag{7}$$

is the $\mathbf{k}$th Taylor coefficient of $\mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y})$ at $\mathbf{y} = \mathbf{y}_c$, and

$$\mathbf{m}_{\mathbf{k}}(c) = \sum_{j=1}^{N_c} (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \mathbf{w}_j \tag{8}$$

is the $\mathbf{k}$th moment of cluster $c$ about its center $\mathbf{y}_c$. The approximation is obtained by truncating the infinite series in (6),

$$\sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x}_i, \mathbf{y}_j) \times \mathbf{w}_j \approx \sum_{\|\mathbf{k}\| < p} \mathbf{a}_{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c) \times \mathbf{m}_{\mathbf{k}}(c), \tag{9}$$

where $\|\mathbf{k}\| = k_1 + k_2 + k_3$ and the order $p$ is chosen to ensure that the error is small. We refer to the right side of (9) as a $p$th-order particle–cluster approximation; it will be shown in Section 3.4 that the error is $O(h^p)$, where $h = r/R$ is the ratio of the cluster radius

$$r = \max\{|\mathbf{y}_j - \mathbf{y}_c|, \quad j = 1, \ldots, N_c\} \tag{10}$$

and the regularized distance between the particle and the cell center

$$R = (|\mathbf{x}_i - \mathbf{y}_c|^2 + \delta^2)^{1/2}. \tag{11}$$

Note that the Taylor coefficients $\mathbf{a}_\mathbf{k}(\mathbf{x}_i, \mathbf{y}_c)$ are independent of the particles $\mathbf{y}_j$ in cluster $c$, and the cluster moments $\mathbf{m}_\mathbf{k}(c)$ are independent of the particle $\mathbf{x}_i$. These features permit an efficient computation of a $p$th-order particle–cluster approximation (9) as follows.

Step 1: Compute the Taylor coefficients $\mathbf{a}_\mathbf{k}(\mathbf{x}_i, \mathbf{y}_c)$ for $\|\mathbf{k}\| < p$. There are $O(p^3)$ coefficients and each of them can be computed in $O(1)$ operations using the recurrence relation derived below.

Step 2: Compute the cluster moments $\mathbf{m}_\mathbf{k}(c)$ for $\|\mathbf{k}\| < p$, unless they are already available from a previous interaction. There are $O(p^3)$ moments and altogether they can be computed in $O(p^3 N_c)$ operations. However, when the moments of a particular cluster $c$ are first computed, they are stored and used again in subsequent interactions between $c$ and other particles $\mathbf{x}_i$. In practice, the cost of computing the moments is a small fraction of the total CPU time required by the treecode.

Step 3: Sum the truncated series on the right side of (9). The sum has $O(p^3)$ terms and each of them can be computed in $O(1)$ operations.

The key point is that the operation counts for Steps 1 and 3 are independent of the number of particles in the cluster $N_c$. Thus assuming the cluster moments are available, a $p$th-order particle–cluster approximation can be computed in $O(p^3)$ operations.

### 3.3. Recurrence Relation

To evaluate the $p$th-order particle–cluster approximation in (9), we require the Taylor coefficients of the regularized kernel $\mathbf{a}_\mathbf{k}(\mathbf{x}, \mathbf{y})$ for $\|\mathbf{k}\| < p$ (for clarity we write $(\mathbf{x}, \mathbf{y})$ instead of $(\mathbf{x}_i, \mathbf{y}_c)$). Explicit formulas for the coefficients can be derived, but the expressions become unwieldy as $\|\mathbf{k}\|$ increases. Instead we derive a recurrence relation permitting rapid computation of the coefficients.

Consider the following regularized Newtonian potential,

$$\phi_\delta(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi} \frac{1}{(|\mathbf{x} - \mathbf{y}|^2 + \delta^2)^{1/2}}, \tag{12}$$

also known as the Plummer potential in the context of gravitational attraction [36]. Note that the Rosenhead–Moore kernel (2) is the gradient of the Plummer potential,

$$\mathbf{K}_\delta(\mathbf{x}, \mathbf{y}) = \nabla_\mathbf{x} \phi_\delta(\mathbf{x}, \mathbf{y}) = -\nabla_\mathbf{y} \phi_\delta(\mathbf{x}, \mathbf{y}). \tag{13}$$

Let

$$b_\mathbf{k}(\mathbf{x}, \mathbf{y}) = \frac{1}{\mathbf{k}!} D_\mathbf{y}^\mathbf{k} \phi_\delta(\mathbf{x}, \mathbf{y}) \tag{14}$$

be the **k**th Taylor coefficient of $\phi_\delta(\mathbf{x}, \mathbf{y})$ with respect to **y**. It follows from (13) that the Taylor coefficients of the Rosenhead–Moore kernel and the Plummer potential satisfy the relation

$$\mathbf{a}_\mathbf{k} = -\sum_{i=1}^{3}(k_i + 1)b_{\mathbf{k}+\mathbf{e}_i}\mathbf{e}_i, \tag{15}$$

where $\mathbf{e}_i$ is the $i$th Cartesian-basis vector (here and sometimes below we omit (**x**, **y**) for clarity). Hence to compute the coefficients $\mathbf{a}_\mathbf{k}$, it is sufficient to derive a recurrence relation for $b_\mathbf{k}$.

PROPOSITION 3.1. *The Taylor coefficients $b_\mathbf{k}$ of the Plummer potential satisfy the recurrence relation*

$$\|\mathbf{k}\|R^2 b_\mathbf{k} - (2\|\mathbf{k}\| - 1)\sum_{i=1}^{3}(x_i - y_i)b_{\mathbf{k}-\mathbf{e}_i} + (\|\mathbf{k}\| - 1)\sum_{i=1}^{3} b_{\mathbf{k}-2\mathbf{e}_i} = 0 \tag{16}$$

*for $\|\mathbf{k}\| \geq 1$, where $b_0 = \phi_\delta(\mathbf{x}, \mathbf{y})$, $b_\mathbf{k} = 0$ if any $k_i < 0$, and $R^2 = |\mathbf{x} - \mathbf{y}|^2 + \delta^2$.*

*Proof.* The Plummer potential $\phi_\delta(\mathbf{x}, \mathbf{y})$ satisfies the differential equation

$$(|\mathbf{x} - \mathbf{y}|^2 + \delta^2)D_{y_1}\phi_\delta - (x_1 - y_1)\phi_\delta = 0. \tag{17}$$

Applying the operator $D_{y_1}^{k_1-1}$ and using Leibniz's rule for differentiating a product we obtain

$$(|\mathbf{x} - \mathbf{y}|^2 + \delta^2)D_{y_1}^{k_1}\phi_\delta - (2k_1 - 1)(x_1 - y_1)D_{y_1}^{k_1-1}\phi_\delta + (k_1 - 1)^2 D_{y_1}^{k_1-2}\phi_\delta = 0. \tag{18}$$

Next we apply $D_{y_2}^{k_2}D_{y_3}^{k_3}$ and substitute the definitions of $b_\mathbf{k}$ and $R^2$ to obtain

$$k_1 R^2 b_\mathbf{k} - 2k_1\sum_{i=1}^{3}(x_i - y_i)b_{\mathbf{k}-\mathbf{e}_i} + k_1\sum_{i=1}^{3} b_{\mathbf{k}-2\mathbf{e}_i} - (x_1 - y_1)b_{\mathbf{k}-\mathbf{e}_1} - b_{\mathbf{k}-2\mathbf{e}_1} = 0. \tag{19}$$

Equation (19) is a recurrence relation for $b_\mathbf{k}$ in which the index 1 plays a special role. Similar equations can be obtained for indices 2 and 3. Summing these equations, we obtain the symmetric form in (16). ■

Equation (16) is the recurrence relation used to compute the Taylor coefficients $b_\mathbf{k}$ of the Plummer potential. Figure 5 shows the order in which the coefficients are computed in practice. Since there are $O(p^3)$ indices satisfying $\|\mathbf{k}\| \leq p$ and each application of the recurrence relation requires $O(1)$ operations, the necessary coefficients $b_\mathbf{k}$ can be computed in $O(p^3)$ operations. Then using (15), the Taylor coefficients $\mathbf{a}_\mathbf{k}$ of the Rosenhead–Moore kernel can be computed with an additional $O(p^3)$ operations.

Note that there is a similarity between the three-dimensional recurrence relation for $b_\mathbf{k}$ in (16) and the one-dimensional recurrence relation for the Legendre polynomials $P_n(x)$,

$$nP_n(x) - (2n - 1)xP_{n-1}(x) + (n - 1)P_{n-2}(x) = 0 \tag{20}$$

for $n \geq 2$, where $P_0(x) = 1$ and $P_1(x) = x$ [49]. This is not surprising since the Taylor coefficients $b_\mathbf{k}$ arise by expanding the Plummer potential with respect to Cartesian coordinates while the Legendre polynomials $P_n(x)$ arise by expanding the Newtonian potential with respect to spherical coordinates. The error analysis below makes use of this observation.

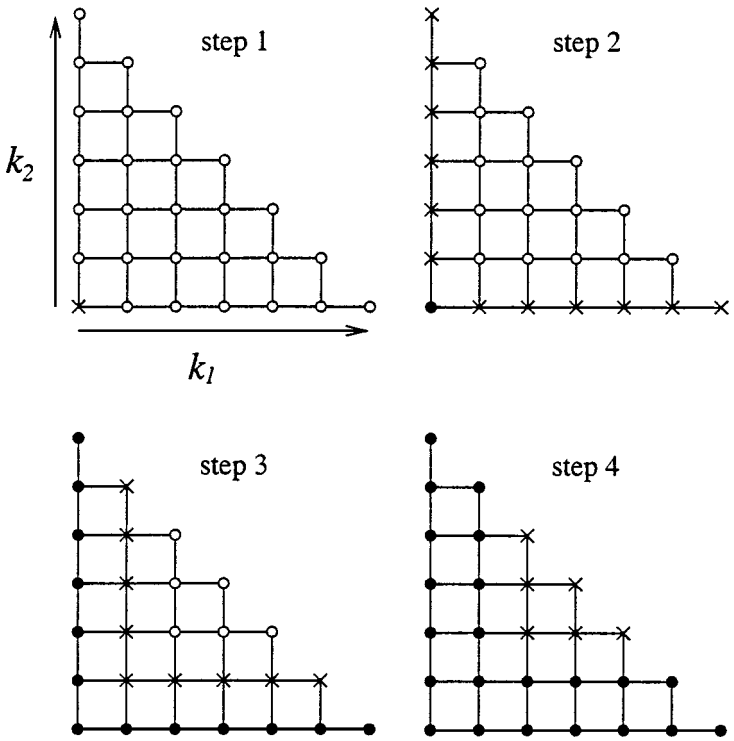


**FIG. 5.** Steps showing the order in which the Taylor coefficients $b_{\mathbf{k}}$ are computed for $||\,\mathbf{k}\,|| \le p$ using the recurrence relation (16); current step (×); later step (○); previous step (●).

### 3.4. Error Analysis

Next we analyze the error in a particle–cluster approximation (9) to obtain a criterion for choosing the order $p$. First define a vector potential associated with a particle–cluster interaction

$$\sum_{j=1}^{N_c} \phi_\delta(\mathbf{x}, \mathbf{y}_j)\mathbf{w}_j, \tag{21}$$

and note that the curl of the vector potential is the velocity,

$$\nabla_{\mathbf{x}} \times \sum_{j=1}^{N_c} \phi_\delta(\mathbf{x}, \mathbf{y}_j)\mathbf{w}_j = \sum_{j=1}^{N_c} \mathbf{K}_\delta(\mathbf{x}, \mathbf{y}_j) \times \mathbf{w}_j. \tag{22}$$

Following the steps leading to the approximation of the velocity (9), the vector potential has the approximation

$$\sum_{j=1}^{N_c} \phi_\delta(\mathbf{x}, \mathbf{y}_j)\mathbf{w}_j \approx \sum_{\|\mathbf{k}\|<p} b_{\mathbf{k}}(\mathbf{x}, \mathbf{y}_c)\mathbf{m}_{\mathbf{k}}(c). \tag{23}$$

We will analyze the error in the approximation of the vector potential (23); the velocity error can be treated similarly and we comment on this below. Using the definition of a cluster

moment (8), the sum of the neglected terms in (23) is

$$\sum_{\|\mathbf{k}\|\geq p} b_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)\mathbf{m}_{\mathbf{k}}(c) = \sum_{n\geq p}\sum_{j=1}^{N_c} B_n(\mathbf{x},\mathbf{y}_c,\mathbf{y}_j)\mathbf{w}_j, \tag{24}$$

where

$$B_n(\mathbf{x},\mathbf{y}_c,\mathbf{y}_j) = \sum_{\|\mathbf{k}\|=n} b_{\mathbf{k}}(\mathbf{x},\mathbf{y}_c)(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}}. \tag{25}$$

We will obtain an alternative expression for $B_n(\mathbf{x},\mathbf{y}_c,\mathbf{y}_j)$.

PROPOSITION 3.2. *The quantity $B_n(\mathbf{x},\mathbf{y}_c,\mathbf{y}_j)$ satisfies the recurrence relation*

$$nR^2B_n - (2n-1)\alpha B_{n-1} + (n-1)\beta^2 B_{n-2} = 0 \tag{26}$$

*for $n\geq 1$, where $B_0=\phi_\delta(\mathbf{x},\mathbf{y}_c)$, $B_{-1}=0$, $\alpha=(\mathbf{x}-\mathbf{y}_c)\cdot(\mathbf{y}_j-\mathbf{y}_c)$, $\beta=|\mathbf{y}_j-\mathbf{y}_c|$, and $R^2=|\mathbf{x}-\mathbf{y}_c|^2+\delta^2$.*

*Proof.* Multiplying the recurrence relation for $b_{\mathbf{k}}$ in (16) by $(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}}$ and summing over all indices $\mathbf{k}$ with $||\mathbf{k}||=n$, we obtain

$$nR^2\sum_{||\mathbf{k}||=n} b_{\mathbf{k}}(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}} - (2n-1)\sum_{||\mathbf{k}||=n}\sum_{i=1}^{3}(x_i-y_{ci})b_{\mathbf{k}-\mathbf{e}_i}(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}} + (n-1)$$
$$\times\sum_{||\mathbf{k}||=n}\sum_{i=1}^{3} b_{\mathbf{k}-2\mathbf{e}_i}(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}} = 0. \tag{27}$$

Recalling that $b_{\mathbf{k}}=0$ if any index $k_i$ is negative, it can be shown that

$$\sum_{||\mathbf{k}||=n} b_{\mathbf{k}-l\mathbf{e}_i}(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}-l\mathbf{e}_i} = B_{n-l} \tag{28}$$

for $l=0,1,2$. The result follows by substituting (28) into (27). ∎

Comparing (20) and (26), it follows that

$$B_n(\mathbf{x},\mathbf{y}_c,\mathbf{y}_j) = \frac{1}{4\pi R}\left(\frac{\beta}{R}\right)^n P_n\left(\frac{\alpha}{\beta R}\right), \tag{29}$$

where $P_n(x)$ is the $n$th Legendre polynomial. Using the fact that $|P_n(x)|\leq 1$ for $|x|\leq 1$ and the bound

$$\left|\frac{\alpha}{\beta R}\right| = \left|\frac{(\mathbf{x}-\mathbf{y}_c)\cdot(\mathbf{y}_j-\mathbf{y}_c)}{|\mathbf{y}_j-\mathbf{y}_c|(|\mathbf{x}-\mathbf{y}_c|^2+\delta^2)^{1/2}}\right| \leq 1, \tag{30}$$

we obtain

$$|B_n(\mathbf{x},\mathbf{y}_c,\mathbf{y}_j)| \leq \frac{1}{4\pi R}\left(\frac{|\mathbf{y}_j-\mathbf{y}_c|}{R}\right)^n. \tag{31}$$

It follows that the approximation error for the vector potential, given in (24), is majorized by a geometric series. A strict error bound can be derived, but it overestimates the actual error. Instead, we take the first term of the series in (24) as a heuristic estimate of the error; with (31) this yields the criterion

$$\frac{M_p(c)}{4\pi R^{p+1}} \le \epsilon, \tag{32}$$

where

$$M_p(c) = \sum_{j=1}^{N_c} |\mathbf{y}_j - \mathbf{y}_c|^p |\mathbf{w}_j| \tag{33}$$

is the $p$th absolute moment of the cluster and $\epsilon$ is a user-specified parameter for controlling the accuracy. In practice, when a particle–cluster approximation is to be evaluated, the order $p$ is set to the minimum value satisfying (32). This leads to a variable-order treecode algorithm. The velocity error can be treated similarly [43] and the resulting criterion is

$$\frac{(p+1)^2 M_p(c)}{4\pi R^{p+2}} \le \epsilon. \tag{34}$$

We call (32) the potential criterion and (34) the velocity criterion; they will be tested and compared in Section 4. In either case it follows that the error in a $p$th-order particle–cluster approximation is $O(h^p)$, where $h = r/R$ is the ratio of the cluster radius $r$ and the regularized distance $R$ between the particle and cluster.

Before concluding this section, we note that error estimates have previously been implemented in variable-order treecode algorithms [35, 36]. In particular, [36] analyzed the error in the Cartesian Taylor approximation for the Newtonian potential and the Plummer potential, although that work recommended $p_{max} = 2$ for the maximum order of approximation. In contrast, we find that higher order approximations are cost-effective when evaluated using the recurrence relation (16) and we set $p_{max} = 8$ in the simulations presented below.

### 3.5. Tree Construction

The tree construction procedure divides the particles into nested cells that are used in evaluating the velocity. Treecode algorithms typically use an oct-tree structure in which the cells on each level are uniform cubes; the cells on one level are obtained by bisecting the cells on the previous level in the three coordinate directions. The tree is often adapted to the particle distribution by leaving undivided any cell containing fewer than a user-specified number of particles $N_0$.

The present algorithm follows this general approach but enhances the adaptivity by shrinking the cells at each step in the construction. The procedure starts with the root cell containing all the particles. Figure 6 shows how a cell is subdivided into subcells. First the cell is bisected in its long directions, i.e., any direction in which its length is greater than $L/\sqrt{2}$, where $L$ is the length of the longest edge. This yields two, four, or eight subcells, depending on the aspect ratio of the original cell. Before further subdivision, each subcell is shrunk to the smallest rectangular box containing its particles. The shrunken subcells form the next level in the tree. The bisect-and-shrink procedure continues until a cell has fewer than $N_0$ particles.

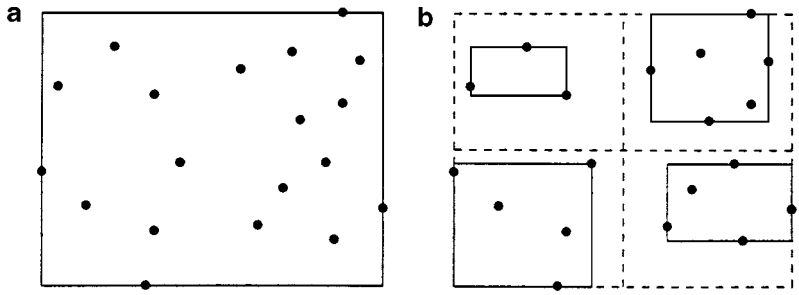


**FIG. 6.** Subdividing a cell into subcells. (a) cell and (b) subcells before shrinking (- - -), subcells after shrinking (——).

This procedure is especially effective for nonhomogeneous particle distributions. Figure 7 shows the cells constructed for a set of particles lying on a vortex sheet spiral, by the standard scheme and the present scheme. The nonuniform rectangular cells constructed by the present scheme are well adapted to the particle distribution; the cell radii on a given level are smaller than for the standard scheme and this leads to a reduction in CPU time since the accuracy criterion ((32) or (34)) is satisfied by a lower order approximation. This explains why we bisect a cell only in the long directions; bisecting in a short direction does not significantly reduce the cell radius. There is little extra work involved in shrinking the cells, but it yields a substantial benefit for vortex sheet computations.

The choice of $N_0$ affects the treecode's performance. If $N_0$ is too small, the tree has many levels and memory usage is high, while if $N_0$ is too large, the cell radii are large and high-order approximation is required, leading to increased CPU time. Tests were performed to determine a suitable value [43] and the present simulations use $N_0 = 500$.

We note that nonuniform rectangular cells have previously been used in a two-dimensional treecode for viscous flow [50]. To achieve load balancing on a parallel processor, in [50] all cells on a given level of the tree were required to have the same number of particles. The present scheme does not enforce this condition. In general, the optimal choice of particle clusters is an interesting problem for future investigation.

### 3.6. Evaluation of Particle Velocities

The treecode is applied to evaluate the particle velocities in each stage of the Runge–Kutta time integration scheme. This is accomplished using two functions. Figure 8a describes the first function, **stage**$(c_0, \epsilon)$, which takes the root cell $c_0$ and accuracy parameter $\epsilon$ as input and returns the particle velocities. The function starts by constructing the tree associated with $c_0$ and then computes the velocity of each particle by calling the second function. Note that **stage**$(c_0, \epsilon)$ is called with a different $c_0$ in each of the four stages comprising one timestep of the Runge–Kutta scheme.

Figure 8b describes the second function, **compute_velocity**$(\mathbf{x}, c, \epsilon)$, which returns the velocity of particle $\mathbf{x}$ induced by cluster $c$ using a divide-and-conquer strategy [27, 28]. It starts by determining the minimum order $p$ satisfying the accuracy criterion (either (32) or (34)) for the input parameters $\mathbf{x}, c, \epsilon$. The function then determines the CPU time required for $p$th-order Taylor approximation ($t_{app}$) and direct summation ($t_{dir}$). The times are estimated empirically as follows. A stand-alone program was written to evaluate the particle–cluster velocity by Taylor approximation and by direct summation for various values of $p$ and $N_c$.

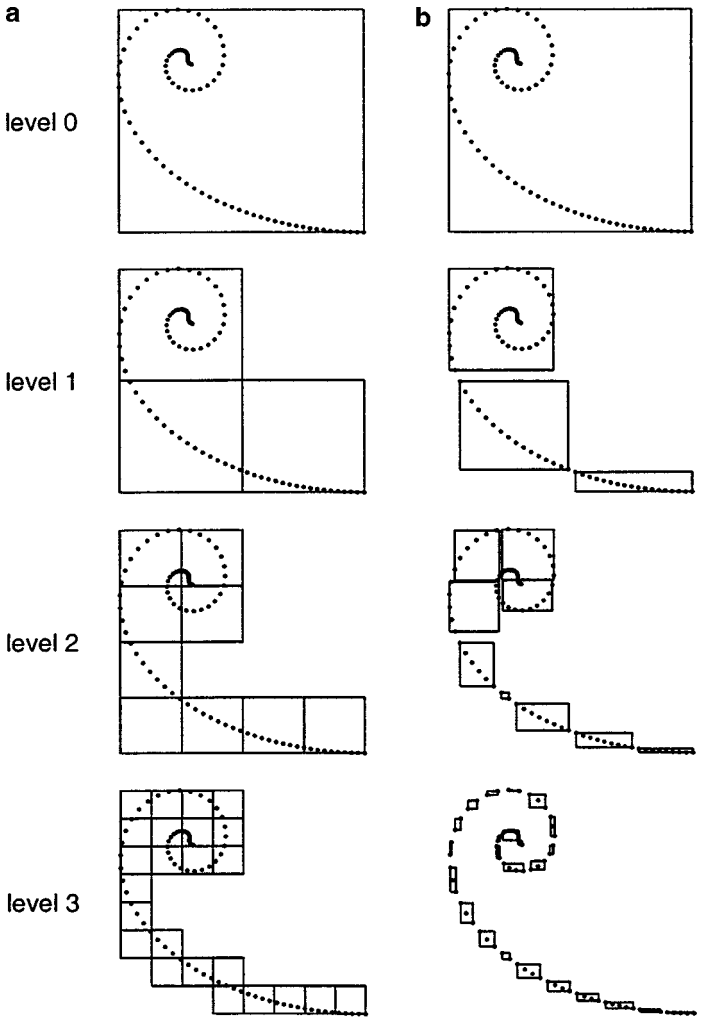


**FIG. 7.** Example of tree construction for a set of particles lying on a vortex sheet spiral. (a) standard scheme (bisect) and (b) present scheme (bisect-and-shrink).

Assuming the cell moments are available, the approximation time $t_{app}$ depends only on the order $p$, and the measured values of $t_{app}$ are stored in a lookup table. The direct summation time $t_{dir}$ depends linearly on the number of particles in the cluster $N_c$, and the stand-alone program determines the parameters in a linear least squares fit. When **compute_velocity** is called for a given particle–cluster interaction, it accesses the lookup table and least-squares parameters to determine $t_{app}$ and $t_{dir}$. Note that these estimated CPU times depend on the coding of the algorithm and on the hardware; if these change, then the lookup table and least-squares parameters should be recomputed.

Next **compute_velocity** compares the estimated CPU times for Taylor approximation and direct summation. If approximation is faster ($t_{app} < t_{dir}$) and the required order is small ($p \le p_{max}$, where $p_{max}$ is a user-specified parameter), then the approximation is performed by evaluating the right side of (9). If direct summation is faster or high-order approximation is required, then direct summation is performed if $c$ is a leaf ($N_c \le N_0$). Otherwise, the code descends to the next level of the tree and recursively calls **compute_velocity** for each child $\hat{c}$ of cluster $c$. The rationale for descending the tree is that the children have smaller

(a) **stage**($c_0$, $\epsilon$)
construct tree associated with root cell $c_0$
**for** $i = 1 : N$
**compute_velocity**($\mathbf{x}_i$, $c_0$, $\epsilon$)
**end for**
return particle velocities

(b) **compute_velocity**($\mathbf{x}$, $c$, $\epsilon$)
$p$ = minimum order satisfying accuracy criterion
$t_{app}$ = CPU time required for $p$th order approximation
$t_{dir}$ = CPU time required for direct summation
**if** $t_{app} < t_{dir}$ and $p \le p_{max}$
compute Taylor coefficients $\mathbf{a_k}(\mathbf{x}, \mathbf{y}_c)$ for $||\mathbf{k}|| < p$
compute cell moments $\mathbf{m_k}(c)$ for $||\mathbf{k}|| < p$ (if necessary)
compute particle-cluster velocity using $p$th order approximation
return velocity
**else**
**if** $N_c \le N_0$
compute particle-cluster velocity by direct summation
return velocity
**else**
**for** each child $\hat{c}$ of cluster $c$
compute $\hat{\epsilon}$
**compute_velocity**( $\mathbf{x}$, $\hat{c}$, $\hat{\epsilon}$ )
return sum of returned velocities
**end for**
**end if**
**end if**

**FIG. 8.** Functions used in the treecode. (a) **stage**($c_0$, $\epsilon$) takes the root cell $c_0$ and accuracy parameter $\epsilon$ as input and returns the particle velocities; (b) **compute_velocity**($\mathbf{x}$, $c$, $\epsilon$) returns the velocity of particle $\mathbf{x}$ induced by cluster $c$ using a divide-and-conquer strategy [27, 28].

radii and fewer particles, so it is more likely that the accuracy criterion will be satisfied. The recursive call to **compute_velocity** for a child $\hat{c}$ of cluster $c$ requires a value $\hat{\epsilon}$ of the accuracy parameter; in the present work this is taken to be

$$\hat{\epsilon} = \frac{M_0(\hat{c})}{M_0(c)} \cdot \epsilon, \tag{35}$$

where

$$M_0(c) = \sum_{j=1}^{N_c} |\mathbf{w}_j| \tag{36}$$

is the total weight of the particles in cluster $c$. In words, the accuracy parameter $\epsilon$ is distributed to the children of $c$ in proportion to their weight.

This completes the description of the treecode algorithm. The code was implemented in the C programming language using double precision arithmetic and dynamic memory allocation. The computations were performed on Sun and SGI workstations. Several parameters

are specified by the user: $N_0$, the maximum number of particles in a leaf of the tree; $p_{max}$, the maximum order of Taylor approximation; and $\epsilon$, the treecode accuracy parameter. Tests were performed to study the effect of these parameters [43]. The simulations below use the values $N_0 = 500$, $p_{max} = 8$. The effect of $\epsilon$ is discussed in the next section.

## 4. TREECODE PERFORMANCE

In this section we examine the accuracy, CPU time, and memory usage of the treecode algorithm in comparison with direct summation. The test case is a fixed surface representing a rolled-up vortex sheet; no time evolution is involved. The sheet was discretized in the manner previously described and the number of particles $N$ was made to vary by changing the refinement. The value $\delta = 0.1$ was chosen for the smoothing parameter. The exact velocity of each particle was computed using direct summation, and an approximation was computed using the treecode for three values of the accuracy parameter, $\epsilon = 10^{-2}, 10^{-3}, 10^{-4}$. Results were obtained using both the potential criterion (32) and the velocity criterion (34). The recorded error is the maximum norm over all particles of the difference between the exact velocity and the treecode approximation.

Figure 9 plots the error as a function of the accuracy parameter $\epsilon$ for several values of $N$. The solid lines are based on the potential criterion and the dashed lines are based on the velocity criterion. In both cases, the error decreases as $\epsilon$ is reduced. Using the potential criterion the error is between one and two orders of magnitude smaller than $\epsilon$, while using the velocity criterion the error is between three and four orders of magnitude smaller than $\epsilon$. The error is fairly insensitive to the number of particles $N$.

Figure 10 plots the treecode CPU time as a function of the error for the same values of $\epsilon$ and $N$ as in Fig. 9. Each connected line denotes a specific number of particles $N$. As above, the solid lines are based on the potential criterion and the dashed lines are based

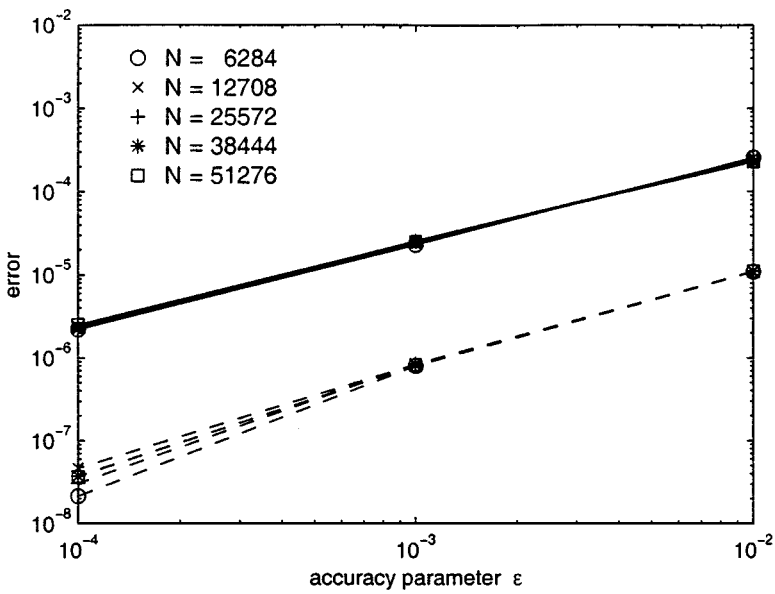


**FIG. 9.** Test case. The treecode error is plotted as a function of the accuracy parameter $\epsilon = 10^{-2}, 10^{-3}, 10^{-4}$ for several values of $N$. (—) potential criterion (32) and (– –) velocity criterion (34).

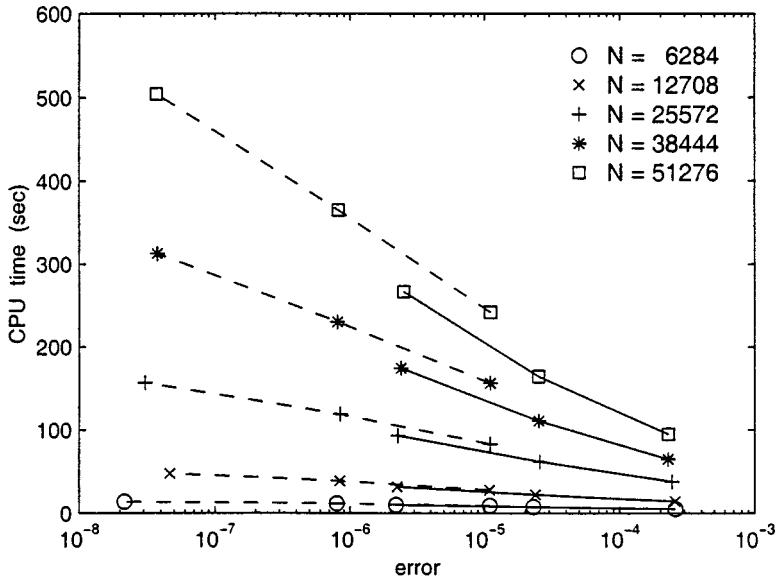


**FIG. 10.** Test case. The treecode CPU time is plotted as a function of the error for the same values of $\epsilon$ and $N$ as in Fig. 9. Each connected line denotes a specific value of $N$; (—) potential criterion (32) and (– –) velocity criterion (34). Each symbol on a connected line denotes a value of the accuracy parameter $\epsilon$, decreasing to the left on each line.

on the velocity criterion. Each symbol on a connected line denotes a value of the accuracy parameter $\epsilon$, decreasing to the left on each line. For example, using the potential criterion with $N = 51{,}276$ and $\epsilon = 10^{-3}$, the error is less than $3 \times 10^{-5}$ and the treecode CPU time is about 175 s. The results show that the CPU time increases as the error is reduced. Note that for each value of $N$, the solid line falls below the corresponding dashed line; this means that the potential criterion requires less CPU time than the velocity criterion to achieve a specified error. This effect becomes more pronounced as $N$ increases. In this sense, the potential criterion is more efficient and so we use it instead of the velocity criterion in the remainder of this work.

Figure 11a plots the CPU time for direct summation and the treecode as a function of the number of particles $N$. The same three values of the accuracy parameter $\epsilon$ are used. The treecode requires more CPU time as $\epsilon$ is reduced, but it is still faster than direct summation in each case displayed. Figure 11b plots the speedup, defined as the ratio of the direct summation time and treecode time. For example, with $N = 102{,}684$ and $\epsilon = 10^{-3}$, the treecode is 10 times faster than direct summation. It is difficult to analyze the operation count for the present adaptive algorithm, but the data in Fig. 11 are consistent with $O(N \log N)$, the expected rate for a particle–cluster treecode [27, 28, 36].

Figure 12 plots the memory usage as a function of $N$. The memory usage for direct summation is $O(N)$. The memory usage for the treecode is higher and increases at a rate slightly faster than $O(N)$ due to the storage required for the cell moments, but even so it remains less than twice the value required for direct summation up to $N = 154{,}108$.

The results presented in Figs. 9–12 show that the treecode algorithm is significantly faster than direct summation for systems having a large number of particles and that the error can be controlled by varying the accuracy parameter. In the next section we apply the particle method and adaptive treecode algorithm to simulate time-dependent vortex sheet motion in three-dimensional flow.

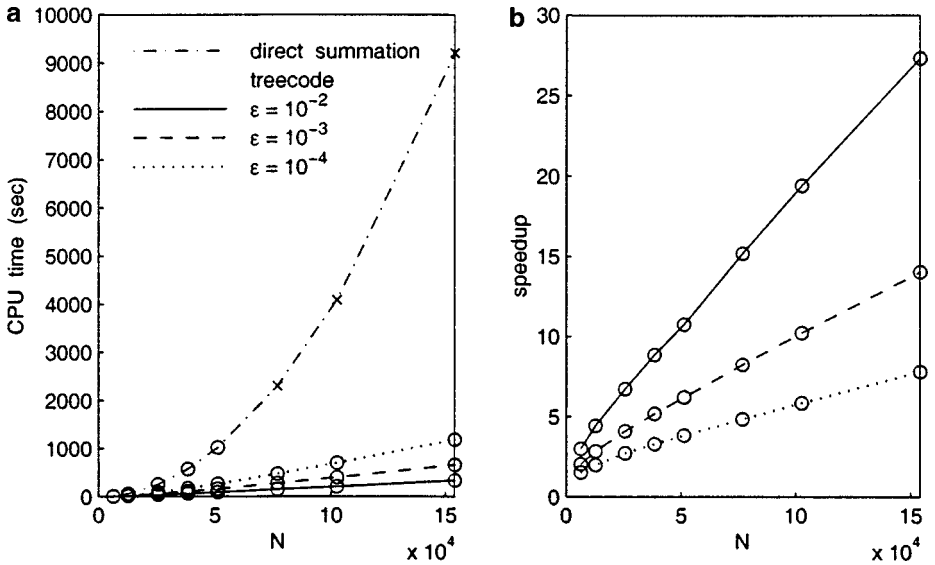


**FIG. 11.** Test case. (a) CPU time for direct summation and the treecode as a function of the number of particles $N$; (○) measured data; (×) projected data. (b) speedup = direct summation time / treecode time.

## 5. VORTEX SHEET SIMULATIONS

There is widespread interest in vortex ring dynamics from theoretical, experimental, and numerical points of view [51, 52]. Here we present two examples in which a circular-disk vortex sheet rolls up into a vortex ring. Our aim is mostly to demonstrate the capability of the numerical method, and although we present some preliminary analysis of the fluid flow, more detailed study is reserved for future work.

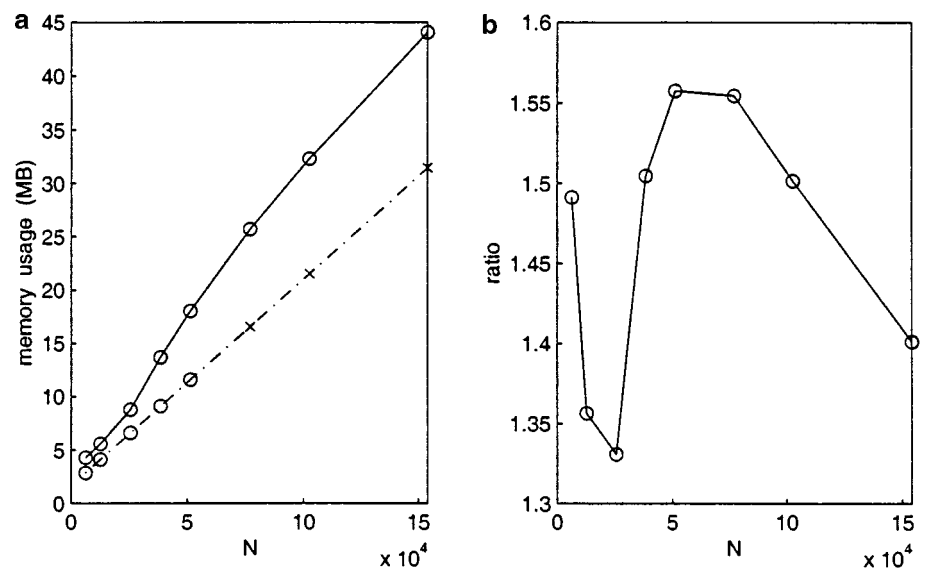


**FIG. 12.** Test case. (a) memory usage as a function of the number of particles $N$; (– · –) direct summation; (——) treecode; (○) measured data; (×) projected data. (b) ratio = treecode usage / direct summation usage.

The simulations below use the value $\delta = 0.1$ for the smoothing parameter. This is meant as a representative value; the effect of varying $\delta$ has been studied in two-dimensional [14, 19], axisymmetric [47, 53], and three-dimensional vortex sheet simulations [17], and although the sheet rolls up more tightly as $\delta$ is reduced, the large-scale structure of the sheet surface does not depend strongly on the precise value of $\delta$. Once a value of $\delta$ is chosen, the remaining numerical parameters must be determined to provide adequate resolution. In the present work, the timestep lies in the range $0.07 \leq \Delta t \leq 0.1$, the particle insertion parameters are $\epsilon_\Gamma = 0.075$ and $\epsilon_\theta = 0.05$, the treecode accuracy parameter is $\epsilon = 10^{-3}$, the maximum order of Taylor approximation is $p_{max} = 8$, and the maximum number of particles in a leaf of the tree is $N_0 = 500$.

In this work, the vortex sheet was visualized using the surface rendering tool in the AVS graphics package. On input, the tool requires a tensor product grid of particles, and on output, it produces a surface that interpolates the particles. Since the discretization in the particle method is not a tensor product in the $\Gamma$-$\theta$ parameter space, it was necessary to create such a grid for plotting purposes. Therefore, using the piecewise cubic interpolation scheme from Section 2.3, a set of particles was obtained having uniform increment $\Delta\theta$ along each material line. Due to memory constraints $\Delta\theta$ could not be chosen too small, and hence some material lines in the tensor product grid had fewer particles than in the original discretization. As a result, the rendered surface may not be sufficiently well resolved in some cases even though the underlying particle discretization was accurate. We shall see an instance of this at the final time in the second example below.

### 5.1. Azimuthal Waves on a Vortex Ring

When fluid is ejected from a circular tube, the separating shear layer rolls up into a vortex ring that propagates away from the tube exit plane [53]. At early times the ring is axisymmetric, but experiments show that azimuthal waves develop later in time, and numerical studies have documented the relation between the azimuthal wavenumber and the perturbation growth rate [54, 55]. To simulate these waves, we introduce a transverse perturbation in the circular-disk vortex sheet defined in (3). The $x_1$ and $x_2$ coordinates are unchanged, but the third coordinate changes from $x_3 = 0$ to $x_3 = 0.1\, r^2 \cos k\theta$, where $(r, \theta)$ are polar coordinates in the $x_1$-$x_2$ plane and $k$ is the perturbation wavenumber. The factor $r^2$ is included to smooth the perturbation at the origin $r = 0$.

Figure 13 plots the resulting motion for two values of the perturbation wavenumber, $k = 5$ and $k = 9$, at times $t = 0, 2, 4,$ and 6. The edge of the sheet rolls up into a spiral, forming a vortex ring with $k$ azimuthal waves around the circumferential ring axis. Figure 14 plots a section of the sheet surface to reveal the spiral core. The azimuthal instability appears to be stronger with $k = 9$ than with $k = 5$. This is supported by Fig. 15, showing a subset of material lines in the core of the ring. With $k = 5$, the lines undergo small amplitude oscillation about a circular shape. With $k = 9$, the lines evolve in a more complex manner leading to the formation of hairpins that wrap around the core as observed in previous simulations [54, 56, 57].

### 5.2. Vortex Ring Merger

A number of laboratory experiments have been performed to study the interaction of two vortex rings moving side by side in the same direction [58–63]. As time proceeds, the rings are drawn together and they merge into a single ring that later splits apart again into two rings. The change in ring topology is due to vortex reconnection and this is a topic of

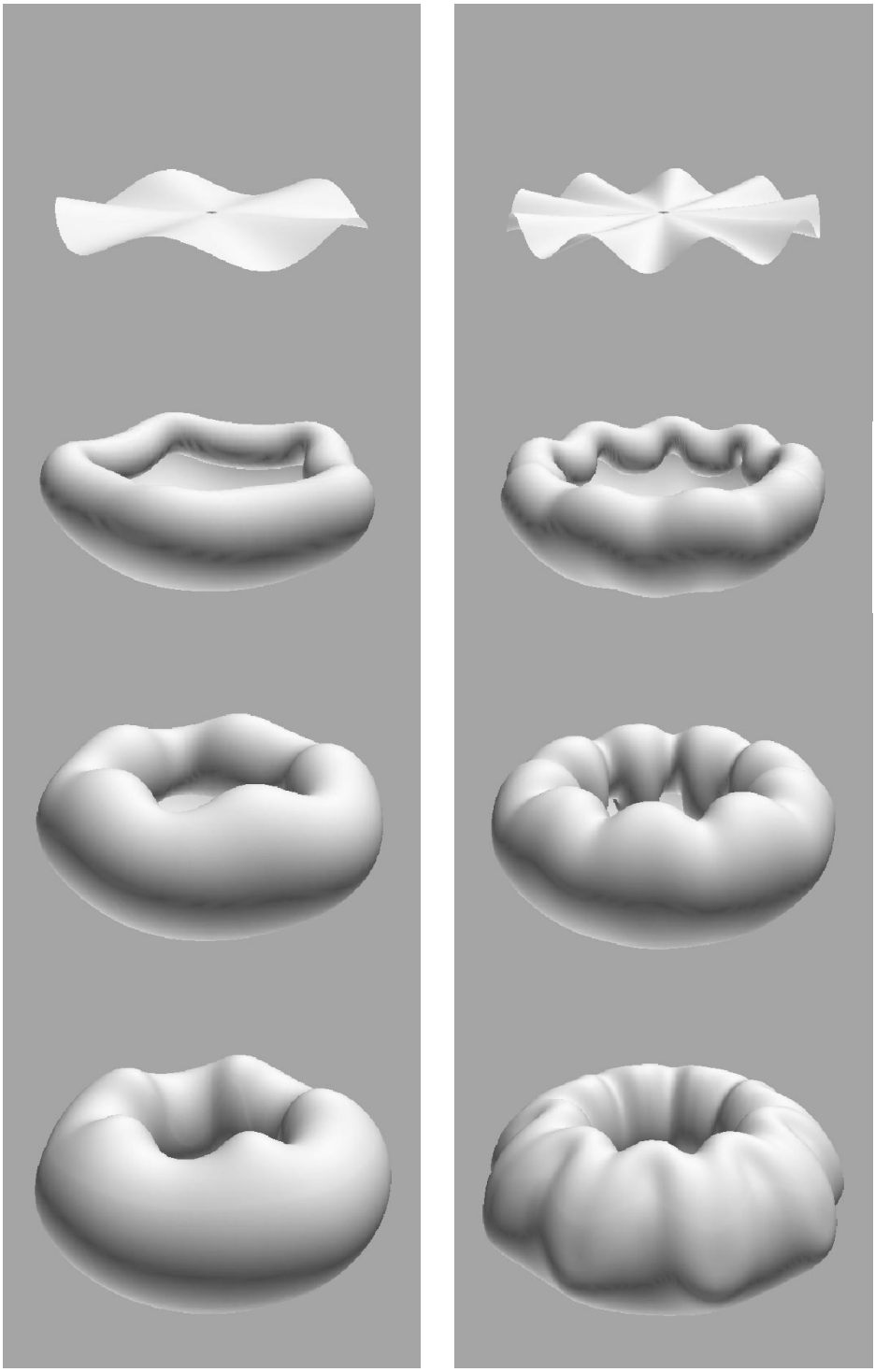

**FIG. 13.** A circular-disk vortex sheet with transverse perturbation of wavenumber $k$. The sheet rolls up into a vortex ring with $k$ azimuthal waves around the circumferential ring axis. $k = 5$ (left); $k = 9$ (right). $t = 0$, 2, 4, and 6.

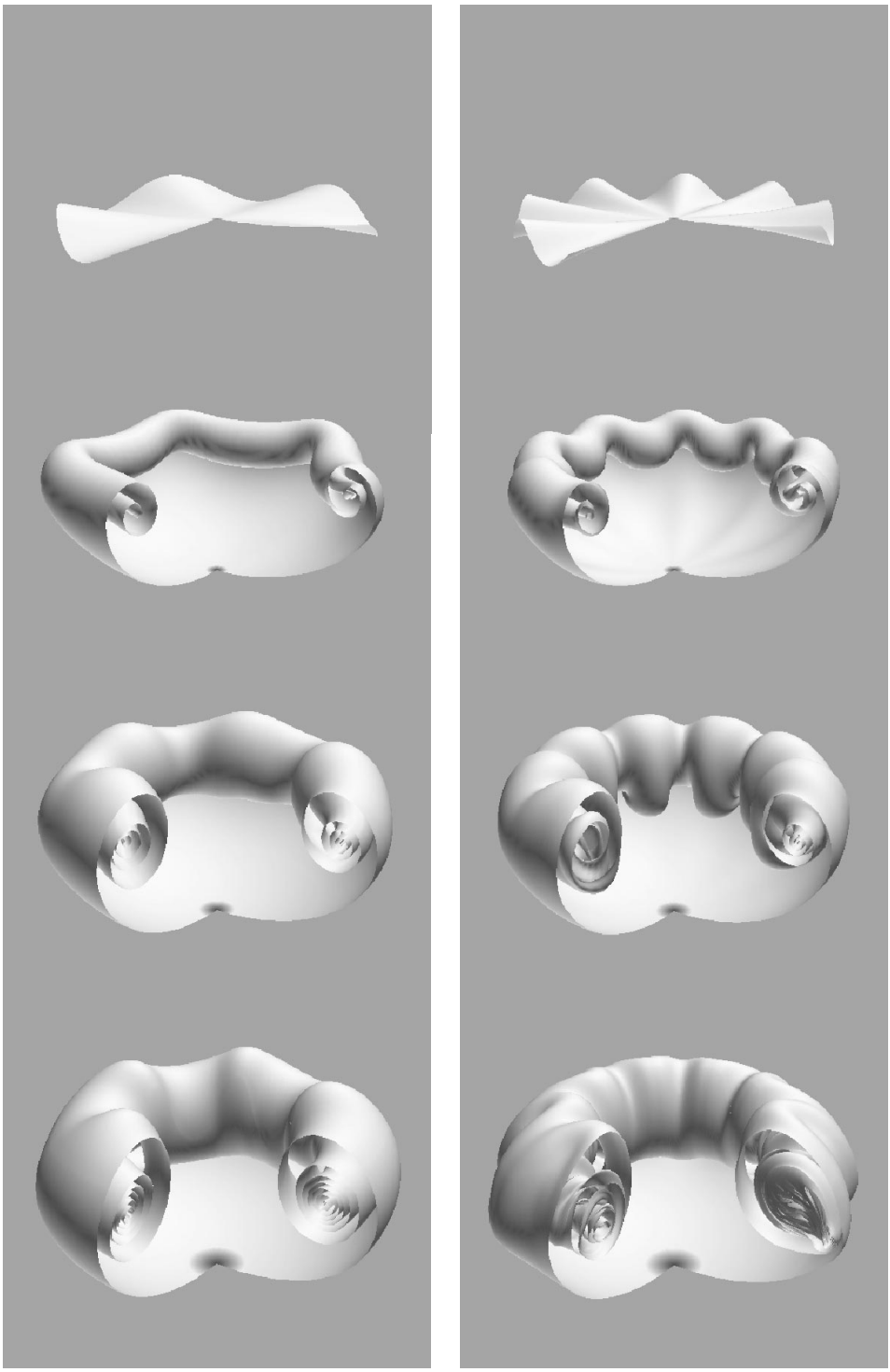

**FIG. 14.** Section of the sheet surface from Fig. 13. $k = 5$ (left); $k = 9$ (right). $t = 0$, 2, 4, and 6.

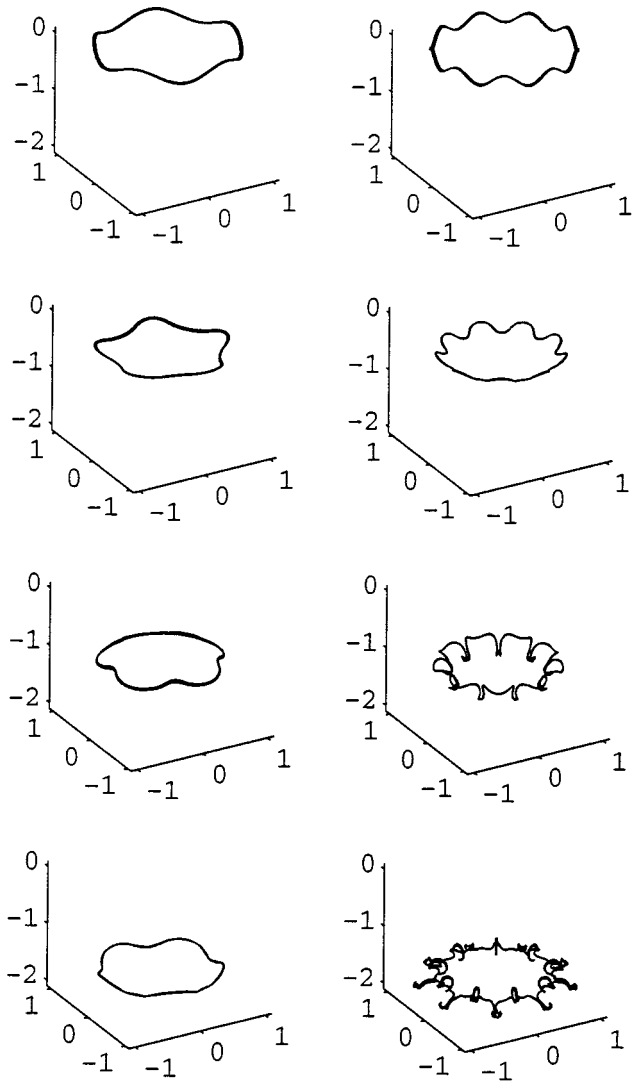


**FIG. 15.** A subset of material lines in the core of the ring from Figs. 13, 14. $k = 5$ (left); $k = 9$ (right). $t = 0$, 2, 4, and 6.

fundamental interest [64]. The experimental results have inspired many numerical studies [65–72]. Here we simulate the ring merger problem using two circular-disk vortex sheets initially inclined at 30° from the horizontal. Figure 16 plots the resulting motion at times $t = 0, 1, 2, 3,$ and 4. The sheets roll up into a pair of vortex rings that interact with each other. At early times ($t = 1$) the rings are nearly axisymmetric and the core radius is almost uniform around the circumferential ring axis. At later times ($t \geq 2$) the core radius becomes increasingly nonuniform.

Figure 17 shows a closeup of the sheet surface at time $t = 4$. One can distinguish two regions, an inner region where the rings are close together, and an outer region where they are further apart. The core radius is nonuniform; it is small in the inner region and large in the outer region. In the inner region, the rings form a pair of antiparallel vortex tubes

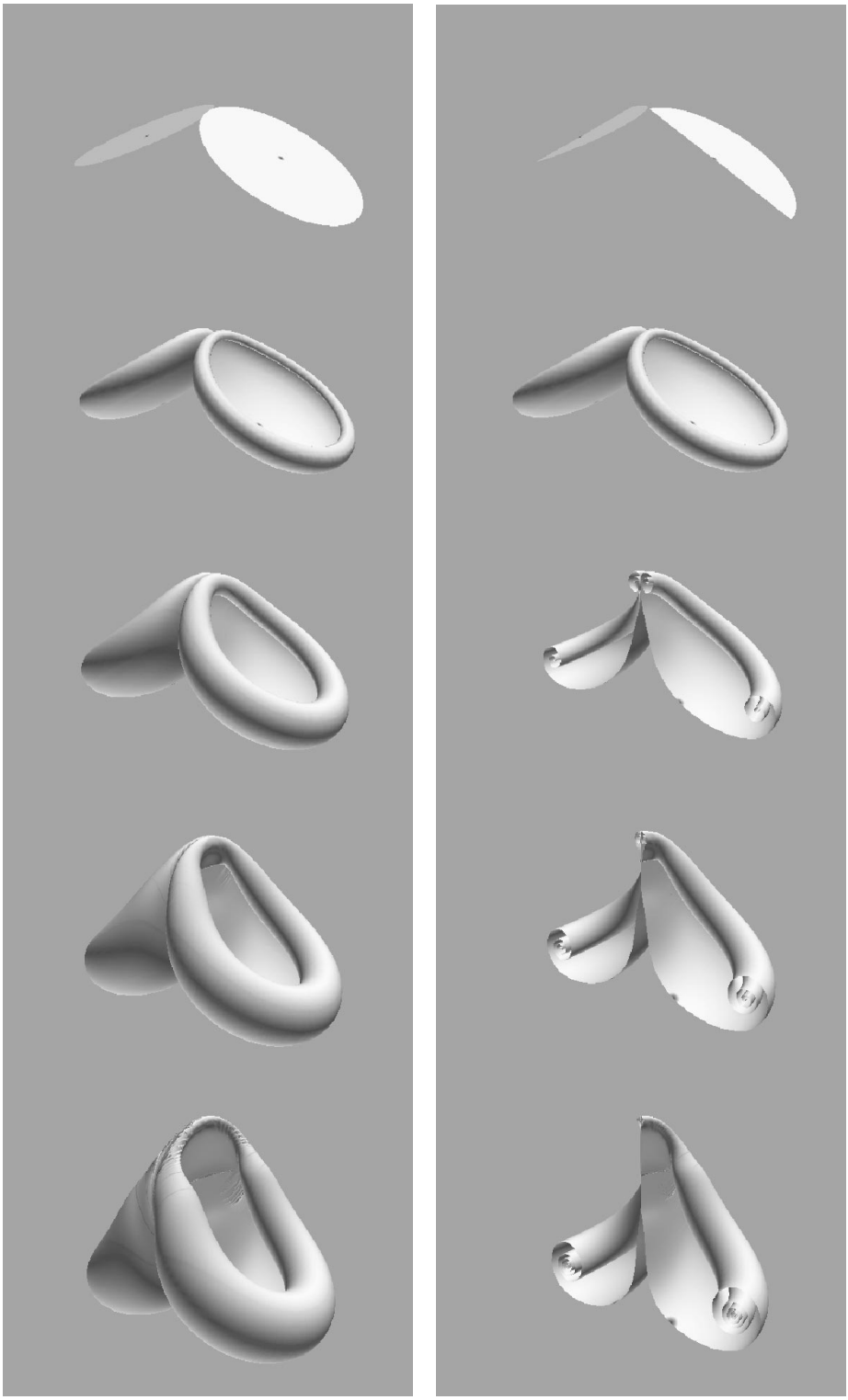

**FIG. 16.** Simulation of vortex ring merger. Two circular-disk vortex sheets are initially inclined at 30° from the horizontal. Sheet surface (left); section (right). $t = 0$, 1, 2, 3, and 4.

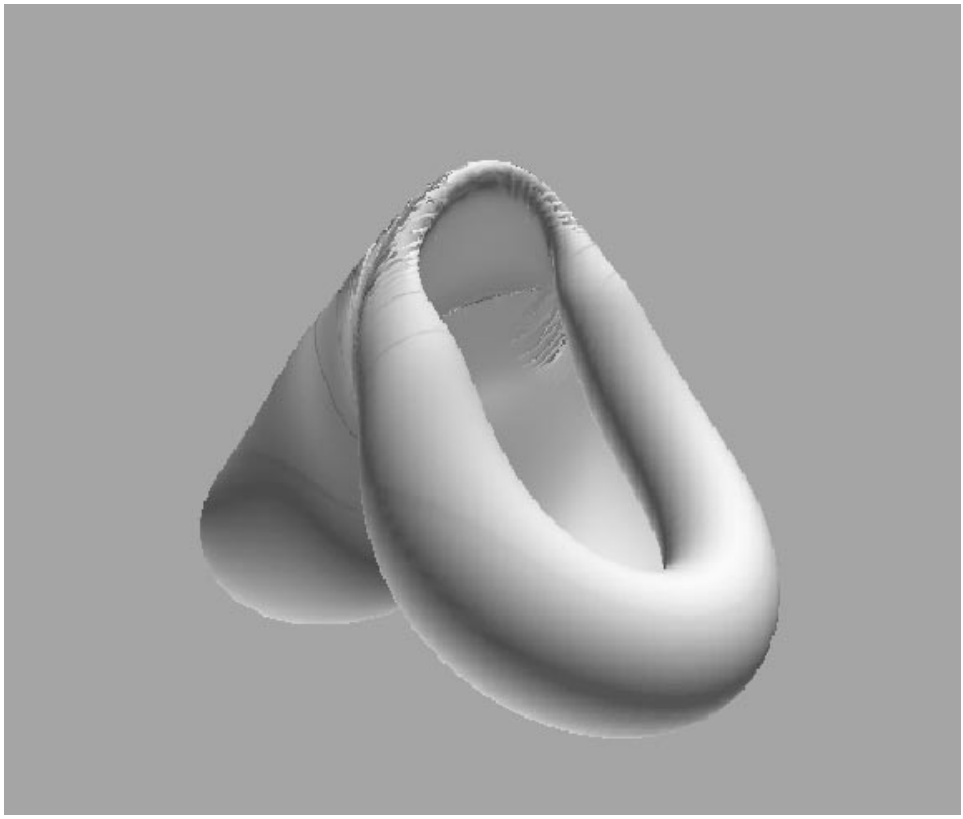

**FIG. 17.** Closeup of sheet surface from Fig. 16 at time $t = 4$.

that propagate vertically as a dipole. Note that the outer region of each ring is shaped like a horseshoe, and the inner region is shaped like an arch connecting the legs of the horseshoe, while the transition zone between the two regions is shaped like a funnel. The texture of the sheet surface in the outer region is smooth, but there are small-scale features on the sheet surface in the inner region; these are not due to errors in the particle method, but rather to inadequate resolution in the rendered surface as explained earlier in this section.

Examining Fig. 16, one can see that the core radius in the outer region increases monotonically in time, while the core radius in the inner region increases until time $t = 2$ and then decreases at later times. The decrease in core radius at later times is associated with material stretching in the inner region along the circumferential ring axis. There are several factors that contribute to this stretching (a) the self-induced dipole velocity of the arch causes it to move away from the legs of the horseshoe and (b) the velocity induced by one horseshoe causes the legs of the other horseshoe to separate in time. Presumably, there is also axial flow in the core of the ring at later times, but this is left for future study.

Figure 18 plots a time sequence of vorticity isosurfaces in the ring merger simulation (these are surfaces on which the vorticity field has constant magnitude). To obtain this plot, the vorticity was evaluated on a uniform grid by taking the curl of the regularized velocity field. Two isosurface levels are shown: $\frac{1}{3}$ (light gray) and $\frac{2}{3}$ (dark gray) of the initial peak value. Four views are displayed: perspective, front, side, and top. Initially, the isosurfaces of one ring are disjoint from those of the other ring, but the isosurfaces merge as the rings

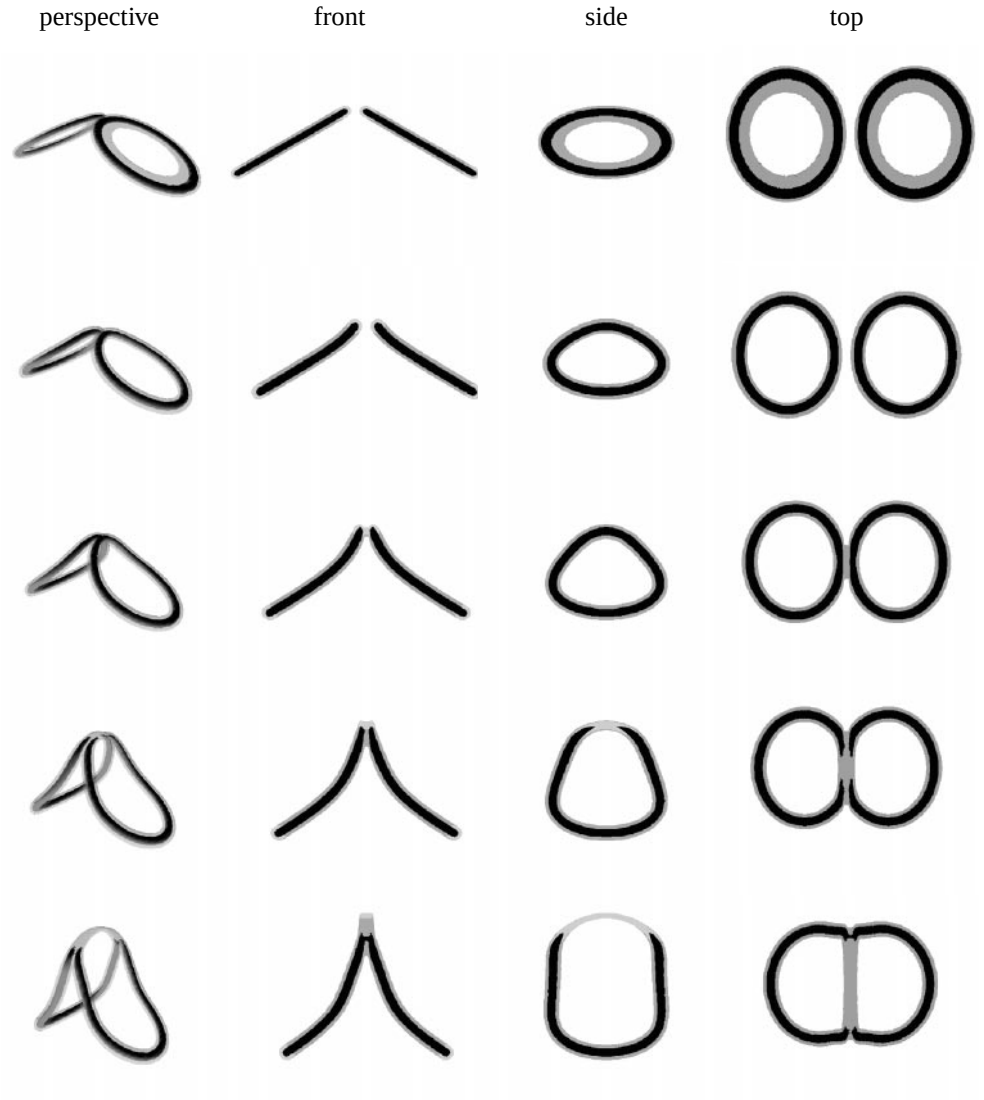


**FIG. 18.** Vorticity isosurfaces in the ring merger simulation. Two isosurface levels are shown: $\frac{1}{3}$ (light gray) and $\frac{2}{3}$ (dark gray) of the initial peak value. Four views are displayed: perspective, front, side, and top. $t = 0$, 1, 2, 3, and 4.

approach each other in time; the $\frac{1}{3}$-isosurfaces merge at $t = 2$ and the $\frac{2}{3}$-isosurfaces merge at $t = 3$. At the final time, the $\frac{2}{3}$-isosurface is effectively a single ring. The $\frac{1}{3}$-isosurface surrounds this ring and also forms an arch spanning the middle of the ring. These results are in qualitative agreement with previous studies [64].

It is important to emphasize the difference between the vortex sheet (Figs. 16, 17) and the vorticity isosurfaces (Fig. 18). The vortex sheet is a material surface and it remains topologically equivalent to a disk as it rolls up. In contrast, the vorticity isosurfaces are not material surfaces and their topology is not preserved in time. In a real flow, topological

changes in the vorticity field are attributed to viscous reconnection [64], but since the present model is nominally inviscid, there must be another explanation for the topological changes seen in Fig. 18. We suggest that when regions of opposite-signed vorticity are in close proximity, for example, the antiparallel vortex tubes in the arch, the contributions that these regions make to the regularized Biot–Savart integral cancel each other; in other words, integration provides a mechanism whereby opposite-signed vorticity is cancelled and this permits topological changes to occur in the regularized vorticity field. While this may explain what is happening in Fig. 18, the physical validity of these results is still uncertain and can only be determined by comparison with genuine viscous simulations and experiments.

To conclude this section, we note that the ring merger simulation started with about 15,000 particles and ended with about 350,000 particles. At the final time, each timestep required approximately 2.5 h of CPU time on an SGI Power Challenge workstation (75 MHz, 128 Mbytes).

## 6. SUMMARY

We presented a Lagrangian particle method for computing vortex sheet motion in three-dimensional flow. The particles are advected by a regularized Biot–Savart integral in which the exact singular kernel is replaced by the Rosenhead–Moore kernel. A particle insertion scheme is applied to maintain resolution as the sheet rolls up. The particle velocities are evaluated by a treecode algorithm which replaces the particle–particle interactions by suitable particle–cluster interactions using a divide-and-conquer strategy [27, 28, 36]. Since the Rosenhead–Moore kernel is nonharmonic, the particle–cluster interactions are approximated by Taylor expansion in Cartesian coordinates rather than a classical multipole expansion involving spherical harmonics. The necessary Taylor coefficients are efficiently computed by a recurrence relation [41, 43]. Several adaptive techniques are employed to gain efficiency. The tree consists of nonuniform rectangular cells adapted to the particle distribution. For each particle–cluster interaction, the order of approximation is chosen adaptively, and a run-time choice is made between Taylor approximation and direct summation based on empirical estimates of the required CPU time. Tests were performed to document the algorithm's accuracy and efficiency, and simulations using up to 350,000 particles were performed on a workstation, which would have been impractical by direct summation.

The method was applied to simulate the roll-up of a circular-disk vortex sheet into a vortex ring. Two examples were presented, the growth of azimuthal waves on a vortex ring and the merger of two vortex rings moving side by side. The particle method is well suited for tracking the deformation of the material sheet surface. This type of regularized vortex sheet model has previously been validated in several cases by comparison with viscous simulations [25] and experiments [53]. An important goal in future work will be to extend such comparisons to three-dimensional vortex ring dynamics.

Finally we note that the treecode algorithm developed here for the Rosenhead–Moore kernel can be applied to a variety of other kernels; the main prerequisite is that the Taylor coefficients of the kernel should satisfy a simple recurrence relation. This approach has recently been applied to problems involving screened electrostatics and general power-law interactions in molecular dynamics [73–75].

## ACKNOWLEDGMENT

This work was supported by the National Science Foundation through Grants DMS-9506452 and DMS-9973293.

## REFERENCES

1. A. Leonard, Computing three-dimensional incompressible flows with vortex elements, *Annu. Rev. Fluid Mech.* **17**, 523 (1985).
2. E. G. Puckett, Vortex methods: An introduction and survey of selected research topics, in *Incompressible Computational Fluid Dynamics-Trends and Advances*, edited by M. D. Gunzburger and R. A. Nicolaides (Cambridge Univ. Press, Cambridge, UK, 1993), p. 335.
3. E. Meiburg, Three-dimensional vortex dynamics simulations, in *Fluid Vortices*, edited by S. I. Green (Kluwer Academic, Dordrecht/Norwell, MA, 1995), p. 651.
4. G.-H. Cottet and P. D. Koumoutsakos, *Vortex Methods: Theory and Practice* (Cambridge Univ. Press, Cambridge, UK, 2000).
5. A. J. Majda and A. Bertozzi, *Vorticity and Incompressible Flow* (Cambridge Univ. Press, Cambridge, UK, 2001).
6. L. Rosenhead, The formation of vortices from a surface of discontinuity, *Proc. Roy. Soc. London Ser. A* **134**, 170 (1931).
7. D. W. Moore, On the point vortex method, *SIAM J. Sci. Stat. Comput.* **2**, 65 (1981).
8. D. W. Moore, The spontaneous appearance of a singularity in the shape of an evolving vortex sheet, *Proc. Roy. Soc. London Ser. A* **365**, 105 (1979).
9. D. I. Meiron, G. R. Baker, and S. A. Orszag, Analytic structure of vortex sheet dynamics. 1. Kelvin–Helmholtz instability, *J. Fluid Mech.* **114**, 283 (1982).
10. R. Krasny, A study of singularity formation in a vortex sheet by the point-vortex approximation, *J. Fluid Mech.* **167**, 65 (1986).
11. M. J. Shelley, A study of singularity formation in vortex-sheet motion by a spectrally accurate vortex method, *J. Fluid Mech.* **244**, 493 (1992).
12. A. J. Chorin and P. S. Bernard, Discretization of a vortex sheet, with an example of roll-up, *J. Comput. Phys.* **13**, 423 (1973).
13. C. R. Anderson, A vortex method for flows with slight density variations, *J. Comput. Phys.* **61**, 417 (1985).
14. R. Krasny, Desingularization of periodic vortex sheet roll-up, *J. Comput. Phys.* **65**, 292 (1986).
15. W. T. Ashurst and E. Meiburg, Three-dimensional shear layers via vortex dynamics. *J. Fluid Mech.* **189**, 87 (1988).
16. M. E. Agishtein and A. A. Migdal, Dynamics of vortex surfaces in three dimensions: Theory and simulations, *Physica D* **40**, 91 (1989).
17. M. Brady, A. Leonard, and D. I. Pullin, Regularized vortex sheet evolution in three dimensions, *J. Comput. Phys.* **146**, 520 (1998).
18. C. Pozrikidis, Theoretical and computational aspects of the self-induced motion of three-dimensional vortex sheets. *J. Fluid Mech.* **425**, 335 (2000).
19. R. Krasny, Computation of vortex sheet roll-up in the Trefftz plane, *J. Fluid Mech.* **184**, 123 (1987).
20. L. Rosenhead, The spread of vorticity in the wake behind a cylinder, *Proc. Roy. Soc. London Ser. A* **127**, 590 (1930).
21. D. W. Moore, Finite amplitude waves on aircraft trailing vortices, *Aero. Quart.* **23**, 307 (1972).
22. O. M. Knio and A. F. Ghoniem, Three-dimensional vortex simulation of rollup and entrainment in a shear layer, *J. Comput. Phys.* **97**, 172 (1991).
23. G. S. Winckelmans, J. K. Salmon, M. S. Warren, A. Leonard, and B. Jodoin, Application of fast parallel and sequential tree codes to computing three-dimensional flows with the vortex element method and boundary element methods, in *Vortex Flows and Related Numerical Methods II*, edited by Y. Gagnon, G.-H. Cottet,

D. G. Dritschel, A. F. Ghoniem, and E. Meiburg, ESAIM: Proceedings (1996) Vol. 1, p. 225, available at http://www.emath.fr/Maths/Proc/Vol.1/index.htm.

24. L. Greengard, Fast algorithms for classical physics, *Science* **265**, 909 (1994).
25. G. Tryggvason, W. J. A. Dahm, and K. Sbeih, Fine structure of vortex sheet rollup by viscous and inviscid simulation, *J. Fluids Eng.* **113**, 31 (1991).
26. E. Harabetian, S. Osher, and C.-W. Shu, An Eulerian approach for vortex motion using a level set regularization procedure, *J. Comput. Phys.* **127**, 15 (1996).
27. A. Appel, An efficient program for many-body simulation, *SIAM J. Sci. Stat. Comput.* **6**, 85 (1985).
28. J. Barnes and P. Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, *Nature* **324**, 446 (1986).
29. L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* **73**, 325 (1987).
30. L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems* (MIT Press, Cambridge, MA, 1988).
31. F. Zhao, *An O(N) Algorithm for Three-Dimensional N-Body Simulations*, AI-TR-995 (Massachusetts Institute of Technology, Cambridge, MA, 1987).
32. J. Carrier, L. Greengard, and V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, *SIAM J. Sci. Stat. Comput.* **9**, 669 (1988).
33. L. van Dommelen and E. A. Rundensteiner, Fast, adaptive summation of point forces in the two-dimensional Poisson equation, *J. Comput. Phys.* **83**, 126 (1989).
34. C. R. Anderson, An implementation of the fast multipole method without multipoles, *SIAM J. Sci. Stat. Comput.* **13**, 923 (1992).
35. H. G. Petersen, D. Soelvason, J. W. Perram, and E. R. Smith, The very fast multipole method, *J. Chem. Phys.* **101**, 8870 (1994).
36. J. K. Salmon and M. S. Warren, Skeletons from the treecode closet, *J. Comput. Phys.* **111**, 136 (1994).
37. D. W. Elliott and J. A. Board, Jr., Fast Fourier transform accelerated multipole algorithm, *SIAM J. Sci. Comput.* **17**, 398 (1996).
38. J. H. Strickland and R. S. Baty, A pragmatic overview of fast multipole methods. *Lect. Appl. Math.* **32**, 807 (1996).
39. H. Cheng, L. Greengard, and V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, *J. Comput. Phys.* **155**, 468 (1999).
40. J. T. Hamilton and G. Majda, On the Rokhlin-Greengard method with vortex blobs for problems posed in all space or periodic in one direction, *J. Comput. Phys.* **121**, 29 (1995).
41. C. Draghicescu and M. Draghicescu, A fast algorithm for vortex blob interactions, *J. Comput. Phys.* **116**, 69 (1995), doi:10.1006/jcph.1995.1006.
42. T. Sakajo and H. Okamoto, An application of Draghicescu's fast summation method to vortex sheet motion, *J. Phys. Soc. Japan* **67**, 462 (1998).

43a. K. Lindsay, *A Three-Dimensional Cartesian Tree-Code and Applications to Vortex Sheet Roll-Up*, Ph.D. thesis (University of Michigan, Ann Arbor, MI, 1997).

43b. T. Sakajo, Numerical Computation of a Three-Dimensional Vortex Sheet in a Swirl Flow, *Fluid Dyn. Res.* **28**, 423 (2001).

44. R. E. Caflisch, Mathematical analysis of vortex dynamics, in *Mathematical Aspects of Vortex Dynamics*, edited by R. E. Caflisch (SIAM, Philadelphia, PA, 1988), p. 1.
45. Y. Kaneda, A representation of the motion of a vortex sheet in a three-dimensional flow, *Phys. Fluids A* **2**, 458 (1990).
46. R. E. Caflisch and X. Li, Lagrangian theory for 3D vortex sheets with axial or helical symmetry, *Trans. Thy. Stat. Phys.* **21**, 559 (1992).
47. M. Nitsche, *Axisymmetric Vortex Sheet Roll-Up*, Ph.D. thesis (University of Michigan, Ann Arbor, MI, 1992).
48. R. Krasny and M. Nitsche, The onset of chaos in vortex sheet flow, *J. Fluid. Mech.* submitted for publication.
49. G. E. Andrews, R. Askey, and R. Roy, *Special Functions* (Cambridge Univ. Press, Cambridge, UK, 1999).
50. N. R. Clarke and O. R. Tutty, Construction and validation of a discrete vortex method for the two-dimensional incompressible Navier–Stokes equations, *Comput. Fluids* **23**, 751 (1994).

51. K. Shariff and A. Leonard, Vortex rings, *Annu. Rev. Fluid. Mech.* **24**, 235 (1992).
52. T. T. Lim and T. B. Nickels, Vortex rings, in *Fluid Vortices*, edited by S. I. Green (Kluwer Academic, Dordrecht/Norwell, MA, 1995), p. 95.
53. M. Nitsche and R. Krasny, A numerical study of vortex ring formation at the edge of a circular tube, *J. Fluid Mech.* **276**, 139 (1994).
54. O. M. Knio and A. F. Ghoniem, Numerical study of a three-dimensional vortex method, *J. Comput. Phys.* **86**, 75 (1990).
55. K. Shariff, R. Verzicco, and P. Orlandi, A numerical study of three-dimensional vortex ring instabilities; viscous corrections and early nonlinear stage. *J. Fluid Mech.* **279**, 351 (1994).
56. A. J. Chorin, Hairpin removal in vortex interactions, *J. Comput. Phys.* **91**, 1 (1990).
57. A. J. Chorin, Hairpin removal in vortex interactions II, *J. Comput. Phys.* **107**, 1 (1993).
58. T. Kambe and T. Takao, Motion of distorted vortex rings, *J. Phys. Soc. Japan* **31**, 591 (1971).
59. T. Fohl and J. S. Turner, Colliding vortex rings, *Phys. Fluids* **18**, 433 (1975).
60. Y. Oshima and S. Asaka, Interaction of two vortex rings along parallel axes in air, *J. Phys. Soc. Japan* **42**, 708 (1977).
61. P. R. Schatzle, *An Experimental Study of Fusion of Vortex Rings*, Ph.D. thesis (California Institute of Technology, Pasadena, CA, 1987).
62. Y. Oshima and N. Izutsu, Cross-linking of two vortex rings, *Phys. Fluids* **31**, 2401 (1988).
63. T. T. Lim, An experimental study of a vortex ring interacting with an inclined wall. *Exp. Fluids* **7**, 453 (1989).
64. S. Kida and M. Takaoka, Vortex reconnection, *Annu. Rev. Fluid Mech.* **26**, 169 (1994).
65. W. T. Ashurst and D. I. Meiron, Numerical study of vortex reconnection, *Phys. Rev. Lett.* **58**, 1632 (1987).
66. C. R. Anderson and C. Greengard, The vortex ring merger problem at infinite Reynolds number, *Comm. Pure Appl. Math.* **42**, 1123 (1989).
67. A. Leonard and K. Chua, Three-dimensional interactions of vortex tubes, *Physica D* **37**, 490 (1989).
68. H. Aref and I. Zawadzki, Linking of vortex rings, *Nature* **354**, 50 (1991).
69. S. Kida, M. Takaoka, and F. Hussain, Collision of two vortex rings, *J. Fluid Mech.* **230**, 583 (1991).
70. G. S. Winckelmans and A. Leonard, Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows, *J. Comput. Phys.* **109**, 247 (1993).
71. A. S. Almgren, T. Buttke, and P. Colella, A fast adaptive vortex method in three dimensions, *J. Comput. Phys.* **113**, 177 (1994).
72. J. Steinhoff and D. Underhill, Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings, *Phys. Fluids* **6**, 2738 (1994).
73. Z.-H. Duan and R. Krasny, An Ewald summation based multipole method, *J. Chem. Phys.* **113**, 3492 (2000).
74. Z.-H. Duan and R. Krasny, An adaptive treecode for computing nonbonded potential energy in classical molecular systems, *J. Comput. Chem.* **22**, 184 (2001).
75. R. Krasny and Z.-H. Duan, Treecode algorithms for computing nonbonded particle interactions, in *Methods for Macromolecular Modeling*, edited by T. Schlick and H. H. Gan, Lecture Notes in Computational Science and Engineering (Springer-Verlag, Berlin/New York, to appear).