

Application of Fast Parallel and Sequential Tree Codes to Computing Three-Dimensional Flows with the Vortex Element and Boundary Element Methods*

G.S. Winckelmans[†]
Mechanical Engineering Dept.
Université Catholique de Louvain
Louvain-la-Neuve, B-1348, Belgium

J.K. Salmon[‡]
Center for Advanced Computing Research
California Institute of Technology
Pasadena CA 91125, USA

M.S. Warren[§]
Theoretical Astrophysics
Los Alamos National Laboratories
Los Alamos, NM 87545, USA

A. Leonard[¶]
Graduate Aeronautical Laboratories
California Institute of Technology
Pasadena CA 91125, USA

B. Jodoin
Chemical Eng. Dept.
Université de Sherbrooke
Sherbrooke (Qc), Canada

Abstract

A fast parallel oct-tree code originally developed for three-dimensional N-body gravitational simulations was modified into (1) a fast N-vortex code for viscous and inviscid vortex flow computations using the regularized vortex particle method (VEM), and (2) a fast N-panel code for solving boundary integral equations in potential flow aerodynamics using the boundary element method (BEM). The core of the fast tree code remains essentially unchanged between the different application codes: gravitation, VEM, BEM, etc. Only the modules that actually encode the physical model are changed. Particular attention is given to controlling the error introduced by the use of multipole expansions to represent the field produced by groups of elements, i.e., the tree code error. In particular, the acceptable error bound for use of any multipole expansion approximation is a run-time parameter. Program outputs include statistics on the errors for the field evaluation at all element locations. Problems in VEM and BEM involving N in the range 10^4 to over 10^6 are computed on parallel supercomputers. Problems with N in the range 10^3 to 10^5 are computed on workstations. Performance results are presented, together

*Computing performed, in part, using the Intel Touchstone Delta and the Intel Paragon operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access provided by Caltech and LANL. Computing also performed on the NAS IBM-SP2 at NASA Ames, with access provided by LANL.

[†]Part of the work done while Assistant Professor, Mech. Eng. Dept., University of Sherbrooke, Sherbrooke, QC, J1K-2R1 Canada, with support of University Grant, Individual Research Grant from National Sciences and Engineering Research Council, Establishment of New Researcher Grant from “Fonds pour la Formation de Chercheurs et l’Aide à la Recherche”. Part of the work also done while Postdoctoral Research Fellow, Graduate Aeronautical Laboratories, California Institute of Technology, Pasadena, CA 91125, with support of NSF Center for Research in Parallel Computation and Office of Naval Research, Navy Grant N00014-92-J-1072.

[‡]Supported by NSF CRPC and by NASA Grant under the High Performance Concurrent Computing program.

[§]Supported by a NASA grant under the HPCC program.

[¶]Supported by ONR, Navy Grant N00014-94-1-0793.

with sample computational results. For the VEM method, a high order particle redistribution scheme has been incorporated, in an efficient way, into the parallel tree code. It is applied, if necessary, to ensure that the core overlapping condition remains satisfied in long time computations. In addition, two different relaxation schemes have also been incorporated and partially tested. Such schemes are applied, if necessary, to ensure that the particle representation of the vorticity field remains a good representation of the true divergence free vorticity field in long time computations.

1 Introduction

A fast oct-tree code, originally developed for three-dimensional N-body gravitational problems [26–28, 32–34] has been modified into (1) a fast N-vortex code for viscous and inviscid vortex flow computations [29] using the regularized vortex particle method (= vortex element method, VEM) [19, 20, 25, 36–40] combined with the particle strength exchange scheme for viscous diffusion [4, 21], and (2) a fast N-panel code for solving boundary integral equations in potential flow aerodynamics [41] using the boundary element method (BEM, = panel method) [12, 13, 15]. The core of the fast tree code remains essentially unchanged between the different application codes: gravitation, VEM, BEM, etc. Only the modules that actually encode the physical model are changed. In gravitation, the acceleration of one mass element is the gradient of the potential induced by all mass elements, according to Newton’s law of gravitation. Mass elements are accelerated by the local acceleration. In vortex flows computed with the VEM, the velocity of one vortex element is the curl of the vector potential induced by all vortex elements according to the Biot-Savart law. Vortex elements are convected by the local velocity and their vorticity vector is subjected to stretching by the local velocity gradient. The vortex particle code is thus immediately more costly than the gravitational code: (1) particle strengths and potential fields are vectors rather than scalars, and (2) both the first and second derivatives of the vector potential must be evaluated in order to obtain both the velocity and the velocity gradient. Moreover, one needs to maintain the condition that particle cores continue to overlap in long time computations. This requires that a particle redistribution scheme be incorporated into the method. Finally, the particle representation of the vorticity field does not constitute a generally divergence free basis [25, 36]. Thus, although the initial particle discretization of a vorticity field can be made very near divergence free, this condition does not necessarily remain satisfied in long time computations. A relaxation scheme can be applied, if and when necessary, which ensures that the particle field remains a good representation of the true divergence free vorticity field. Different approaches have been proposed [22, 38, 40]. Two approaches have been incorporated into the tree code and have been partially tested.

Computing the time evolution of a set of vortex elements requires, at each time step, that one computes the velocity and velocity gradient, i.e., first and second derivatives of the vector potential field induced by all N elements. This operation is analogous to a matrix-vector multiply for a full $N \times N$ matrix applied to a vector of N elements. This is, by far, the most expensive part of the computation. The remaining aspects of the computation are fairly local and are not as expensive: particle exchange scheme for viscous diffusion, particle redistribution scheme, relaxation schemes. In the diffusion scheme, only particles that are in the neighborhood of a given particle contribute to the change in that particle’s strength. Once computed, the velocity gradient tensor contains all necessary components to evaluate the true vorticity field, $\nabla \times \mathbf{u}$, at the particle’s location. Then, in the relaxation scheme by Pedrizzetti (P-scheme), the particle’s strength is modified using the true vorticity at the particle’s location. In the relaxation scheme by Winckelmans et al. (W-scheme), the particle’s strength is modified using the true vorticity at the particle’s location and the strength of the neighboring particles.

When solving potential flow problems with scalar BEM [12, 13] (or vector BEM), the situation is essentially the same as for VEM: the normal (tangential) velocity at the control point of each boundary element is the gradient (curl) of the (vector) potential induced by all boundary elements. Solving for the unknown element strengths using an iterative technique requires, at each iteration, an operation analogous to a matrix-vector multiply: for each of the N elements, find the velocity (= first derivatives of the field) induced by all N elements. This constitutes the expensive part of the computation.

2 Fast tree-codes and our approach

Thus, the above problems (gravitation, VEM, BEM, etc.) are all $O(N^2)$ in complexity (per time step for gravitation and VEM, per iteration for BEM): for each of the N elements, find the derivatives of the field induced by all N elements. The use of fast tree codes, in 2D and 3D [2, 3, 5–10, 14, 16–18, 23, 24, 26–35, 41, 43], reduces the computing cost associated with all evaluations from $O(N^2)$ to something much more tractable: $O(N \log N)$, or $O(N^{1+\epsilon})$ with $\epsilon \ll 1$, or even $O(N)$ depending on the complexity of the implementation. The “big- O ” notation can however be misleading for practical values of N and desired levels of accuracy.

In our implementations of the VEM (with smoothing of compact support) [35, 42] and of the BEM [29, 41, 42], multipole expansions of order $p = 2$ are used (i.e., monopole + dipole + quadrupole). Particular attention is given to ensuring that the error introduced by the use of multipole expansion approximations remains below a desired level for all evaluations. A runtime parameter, e_{tol} , determines the maximum allowed error bound for any particular multipole evaluation. It was shown [28] that the error on the field first derivatives which is introduced by using an order p multipole representation of a group of elements (i.e., a *cell*) is bounded as follows:

$$e_p(\mathbf{x}) \leq \frac{1}{(d-b)^2} \left[(p+2) \frac{B_{p+1}}{d^{p+1}} - (p+1) \frac{B_{p+2}}{d^{p+2}} \right] \quad (1)$$

where d is the distance between the evaluation point \mathbf{x} and the multipole expansion center \mathbf{x}_c , b is the radius of the smallest sphere centered at \mathbf{x}_c and containing the distributed strengths and

$$B_l = \int_{\text{cell}} \|\mathbf{x}' - \mathbf{x}_c\|^l \|\boldsymbol{\gamma}'\| d\mathbf{x}' . \quad (2)$$

In VEM and vector BEM, $\boldsymbol{\gamma}'$ is a vector and e_p is the L_2 -norm error bound on $\|\nabla \times \boldsymbol{\psi}(\mathbf{x})\|$. In gravitation and scalar BEM, $\boldsymbol{\gamma}'$ is a scalar and the error bound is on $\|\nabla \phi(\mathbf{x})\|$. For particle methods, integrals such as Eq. 2 reduce to sums over the discrete particles contained in the cell; for BEM, one must first integrate the distributed strength over each panel before summing the contributions of the panels contained in a cell.

For ease of implementation and numerical efficiency, error estimates involving only B_0 and B_2 are used: no square root is needed in evaluating the B_2 property of a cell, and the translation of B_2 between children and parents during tree construction is much easier. Using Hölder’s inequality for sums, it can be shown that a generalization of a result presented in [28] is $B_l^{m+n} \leq B_{l-m}^n B_{l+n}^m$ which, together with the more obvious result $B_{l+k} \leq b^k B_l$, leads to optimum (B_0, B_2) -based error bounds for order p multipole expansions:

$$e_2(\mathbf{x}) \leq \frac{B_0}{(d-b)^2} \left[4 \left(\frac{B_2}{B_0 b^2} \right) \left(\frac{b}{d} \right)^3 - 3 \left(\frac{B_2}{B_0 b^2} \right)^2 \left(\frac{b}{d} \right)^4 \right] , \quad (3)$$

$$e_1(\mathbf{x}) \leq \frac{B_0}{(d-b)^2} \left[3 \left(\frac{B_2}{B_0 b^2} \right) \left(\frac{b}{d} \right)^2 - 2 \left(\frac{B_2}{B_0 b^2} \right)^{3/2} \left(\frac{b}{d} \right)^3 \right] , \quad (4)$$

$$e_0(\mathbf{x}) \leq \frac{B_0}{(d-b)^2} \left[2 \left(\frac{B_2}{B_0 b^2} \right)^{1/2} \left(\frac{b}{d} \right) - \left(\frac{B_2}{B_0 b^2} \right) \left(\frac{b}{d} \right)^2 \right]. \quad (5)$$

When building the tree and the multipoles for all active cells, this equation is solved (very efficiently, using Newton-Raphson) for d_{crit} such that the rhs is equal to e_{tol} . This becomes a cell property. Only for elements further away than d_{crit} will that multipole be used: if $d \geq d_{\text{crit}}$, the multipole expansion is used and work is saved, together with the actual error bound, e_{bound} , for that particular interaction at that particular distance d ; if $d < d_{\text{crit}}$, the multipole expansion is not used and the test is applied recursively to all active children cells. A valuable program output is \tilde{e}_{bound} , the square root of the sum of the error bounds squared for all multipole expansion evaluations used in computing the field derivatives at one element. This bound is a good estimate of the total tree code error for that element if one assumes that the errors due to the different multipole expansions used act in random directions. Numerical experiments and comparison with exact $O(N^2)$ computations indicate that this is indeed the case. Typically, one obtains values of \tilde{e}_{bound} that are fairly uniform (with a maximum value roughly twice that of the mean over all particles). Typically, one also obtains that the mean of \tilde{e}_{bound} is three to seven times e_{tol} .

Notice that, for smoothing methods that don't have a compact support, one must develop the multipole expansions corresponding to the smooth kernel, together with the appropriate error bound estimates. This was done for two cases: the low and the high order algebraic smoothings of [40]. This work will be reported in another paper.

All arithmetic is done in single precision except for situations in which one computes a summation of hundreds or thousands of independent quantities. Such summations are done in double precision to prevent loss of accuracy due to roundoff. Finally, all $1/\sqrt{x}$ functions are evaluated using the fastest possible inlined hardware instruction. Depending on the details of the chip architecture, the result may only be accurate to within about 1%. This is sufficient for error estimate evaluations. For field evaluations, the result is iterated twice using a Newton-Raphson procedure to provide accuracy approaching that of single-precision arithmetic. The same goes for the rapid evaluation of quotients and the reciprocal function. In both cases the result may differ from the IEEE-754 specification for the equivalent function. The small errors introduced by roundoff in this way are much smaller than those inherent in the multipole approximation.

3 Back to the VEM method

In the regularized vortex particle method, the particle vorticity field is

$$\tilde{\omega}_\sigma(\mathbf{x}, t) = \sum_s \frac{1}{\sigma^3} \zeta \left(\frac{\|\mathbf{x} - \mathbf{x}^s(t)\|}{\sigma} \right) \gamma^s(t) \quad (6)$$

with $\gamma^s = \boldsymbol{\omega}^s \text{vol}^s$ the particle strength (where vol^s is the fluid volume associated with that particle, and $\boldsymbol{\omega}^s$ is to be understood as the 'averaged' vorticity within that volume) and σ the regularization parameter (the particle 'core' size). Since the flow is incompressible, the fluid volume associated with each particle remains constant in time. Moreover, in the particular case of particles generated on a $h \times h \times h$ lattice (e.g., at initial condition, or after particle redistribution, see below), all particles have the same fluid volume, $\text{vol}^s = h^3$. Finally, the regularization parameter σ is taken as uniform. It is also kept constant in time (since viscous diffusion is taken into account using the particle exchange scheme, see below). Convergence of the regularized method requires that particle cores overlap slightly. Thus, the spacing between neighbor particles should not be allowed to grow much larger than σ (hence the need for a particle redistribution scheme, see below).

Particles are convected by the local velocity,

$$\frac{d}{dt} \mathbf{x}^q(t) = \mathbf{u}_\sigma(\mathbf{x}^q(t), t), \quad (7)$$

and their strength is subjected to the 3D stretching of vortex lines. The general mixed scheme is obtained as [36–40],

$$\frac{d}{dt} \boldsymbol{\gamma}^q(t) = \left(\alpha \nabla \mathbf{u}_\sigma(\mathbf{x}^q(t), t) + (1 - \alpha) (\nabla \mathbf{u}_\sigma(\mathbf{x}^q(t), t))^T \right) \cdot \boldsymbol{\gamma}^q(t) \quad (8)$$

for, in principle, any α (but, usually, $0 \leq \alpha \leq 1$, with three typical cases: $\alpha = 1$ (classical scheme), $\alpha = 0$ (transpose), $\alpha = 1/2$ (symmetric)).

For the present version of the VEM code, the Gaussian smoothing is used [20, 29, 38–40]:

$$\zeta(\rho) = \frac{1}{4\pi} \left(\frac{2}{\pi} \right)^{1/2} e^{-\rho^2/2}, \quad (9)$$

$$G(\rho) = \frac{1}{4\pi} \frac{1}{\rho} \operatorname{erf} \left(\frac{\rho}{\sqrt{2}} \right), \quad (10)$$

$$K(\rho) = \frac{1}{\rho^2} (G(\rho) - \zeta(\rho)), \quad (11)$$

$$F(\rho) = \frac{1}{\rho^2} (3K(\rho) - \zeta(\rho)), \quad (12)$$

with ζ the vorticity smoothing function, G the Green's function for the vector potential, K the Biot-Savart function for the velocity evaluation, F a function used in evaluating the velocity gradient, and $\rho = r/\sigma$ the dimensionless distance.

Since the Gaussian smoothing decays exponentially, there is negligible error from treating it as though it has compact support, and vanishes beyond some prescribed cutoff. The cutoff is a controllable program input. It is usually set to $\rho_{\text{cut}} = 5$, hence a relative error of 3.7×10^{-6} . Thus, for distances less than 5σ , direct interaction is used and no multipole expansion is allowed, no matter what the error criterium e_p (not valid within the core anyway) gives. For very accurate computations, the cutoff might be set to higher values, e.g., $\rho_{\text{cut}} = 6$ with a relative error of 1.5×10^{-8} . Higher values of ρ_{cut} imply, of course, an increase in the CPU time per field evaluation. In fact, the choice of ρ_{cut} is not independent from the choice of e_{tol} : for a given e_{tol} , the choice of ρ_{cut} is taken so as to ensure that the multipole error (as measured by the obtained \tilde{e}_{bound}) dominates the cutoff error.

The error function $\operatorname{erf}(x)$ is computed using e^{-x^2} and Eq. 7.1.26 in [1]. For small ρ , Taylor series expansions are used to evaluate G , K and F .

With the particle strength exchange scheme for viscous diffusion [4, 21], we have:

$$\frac{d}{dt} \boldsymbol{\gamma}^q(t) = \dots + \frac{2\nu}{\sigma^2} \sum_s \left(\frac{vol^q}{\sigma^3} \boldsymbol{\gamma}^s(t) - \frac{vol^s}{\sigma^3} \boldsymbol{\gamma}^q(t) \right) \eta \left(\frac{\|\mathbf{x}^s(t) - \mathbf{x}^q(t)\|}{\sigma} \right), \quad (13)$$

where $\eta(\rho) = -\frac{1}{\rho} \frac{d}{d\rho} \zeta(\rho)$. For compact support smoothings (or smoothings that decay fast enough), only the s particles that are in the neighborhood of the q particle are needed. With the Gaussian smoothing (which is also such that $\eta(\rho) = \zeta(\rho)$), we use the same compact support cutoff as for the rest of the code.

3.1 Particle redistribution schemes

One needs to maintain the condition that particle cores overlap. This calls for a particle redistribution scheme if and when necessary. The Λ_2 scheme [16–18] is adopted. It consists in replacing the distorted set of vortex particles by a new set where the new particles are again located on a $h \times h \times h$ lattice. Consider first the normalized 1D problem with unit spacing. Then, in the $\Lambda_2(x)$ scheme, an old particle located at $-\frac{1}{2} \leq x \leq \frac{1}{2}$ gives $-\frac{1}{2}x(1-x)$ of its strength to the new particle located at -1 , $(1-x)(1+x)$ to the new particle located at 0 , and $\frac{1}{2}x(1+x)$ to the new particle located at 1 . This scheme is such that:

$$x^n = (-1)^n \left(-\frac{1}{2}x(1-x) \right) + (0)^n \left((1-x)(1+x) \right) + (1)^n \left(\frac{1}{2}x(1+x) \right) \quad (14)$$

for $n = 0, 1, 2$. In 3D, one applies the scheme as $\Lambda_2(x) \Lambda_2(y) \Lambda_2(z)$. This scheme then conserves exactly total vorticity, linear impulse and angular impulse. It usually does a very good job at energy conservation, and a good job at enstrophy conservation.

Notice that a simpler scheme is the Λ_1 scheme: in that case, an old particle located at $-\frac{1}{2} \leq x \leq \frac{1}{2}$ gives $\frac{1}{2} - x$ of its strength to the new particle located at $-\frac{1}{2}$, and $\frac{1}{2} + x$ to the new particle located at $\frac{1}{2}$. This scheme is such that:

$$x^n = \left(-\frac{1}{2} \right)^n \left(\frac{1}{2} - x \right) + \left(\frac{1}{2} \right)^n \left(\frac{1}{2} + x \right) \quad (15)$$

for $n = 0, 1$. Again, in 3D, one applies the scheme as $\Lambda_1(x) \Lambda_1(y) \Lambda_1(z)$. This scheme then conserves exactly total vorticity and linear impulse. It does not conserve angular impulse. It usually does a poor job at energy conservation, and a very poor job at enstrophy conservation. We recommend that it never be used.

We consider two approaches to the problem of redistributing vorticity from an old set of particles to a new set. The simplest approach consists in creating an empty $N_1 \times N_2 \times N_3$ matrix and filling it by looping over the list of the N particles. Then, the list of new particles is generated from the non-zero matrix elements. This approach is not viable for arbitrary distributions of vorticity since the zero elements in the matrix lead to excessive memory requirements.

Thus, it is desirable that the redistribution scheme exploit the sparseness of the vorticity distribution, i.e. that it avoids creating a mostly empty $N_1 \times N_2 \times N_3$ matrix explicitly in memory. We accomplish this by constructing a oct-tree with the property that every terminal cell in the oct-tree contains either zero or one particle. This is precisely the same data structure used in the fast evaluation of the field quantities, so all the necessary algorithmic machinery was already in place. New particles are simply placed at the centers of terminal nodes of the tree, and their strengths are determined from nearby old particles, which can be easily found by traversing the tree. Notice that the memory required by this construction is modest, no matter how irregular the distribution of particles in space.

The performance of the two approaches is illustrated in Fig. 1 for two sample 2D configurations. It is seen that, for sparse vorticity configurations, the tree-approach is far superior to the matrix-approach. It runs faster and uses less memory (here, saturation at $N = 10^4$ for the matrix-approach). As expected, for dense vorticity configurations, the tree-approach is slightly more expensive than the matrix-approach.

The Λ_2 scheme has been incorporated in the fast 3D parallel tree code as well. Particle redistribution is also done using a tree, without the need for large memory or large message passing requirements. It runs very efficiently. Its cost is negligible compared to the cost associated with the field evaluation.

Finally, it should be noted that newly created particles that are too small in strength are not kept (yet another run-time parameter for the code). This provides control to prevent the number of particles from growing too fast.

3.2 Relaxation schemes for the particle vorticity field

Pedrizzetti's relaxation scheme [22] (P-scheme) was developed in the framework of singular vortex particles. It is modified to be used in the context of regularized vortex particles. At every time step, the particle strength vector is modified using the filtering:

$$\boldsymbol{\gamma}_{\text{new}}^q = (1 - f \Delta t) \boldsymbol{\gamma}^q + f \Delta t \frac{\boldsymbol{\omega}_\sigma(\mathbf{x}^q)}{\|\boldsymbol{\omega}_\sigma(\mathbf{x}^q)\|} \|\boldsymbol{\gamma}^q\| \quad (16)$$

where $\boldsymbol{\omega}_\sigma(\mathbf{x}^q)$ is the true local vorticity field (i.e., the curl of the velocity field) and where f is a frequency factor. The time scale $1/f$ is tuned with respect to the time scale(s) of the physical phenomena under study to give satisfactory results. This relaxation scheme basically acts as a 'spring' that tries to maintain the particle strength vector aligned with the true vorticity vector. This scheme is a simple local operation on the particle strength vector. No system of linear equations involving neighbor particles needs to be solved.

Winckelmans's relaxation scheme [38, 40] (W-scheme) is based on the smooth-function representation of the vorticity field: one requires that, at particle locations, the regularized particle field, $\tilde{\boldsymbol{\omega}}_\sigma(\mathbf{x}^q)$, represented by the new particle strengths be equal to the divergence free vorticity field, $\boldsymbol{\omega}_\sigma(\mathbf{x}^q) = \nabla \times \mathbf{u}_\sigma(\mathbf{x}^q)$, computed using the old particle strengths:

$$\sum_s \frac{1}{\sigma^3} \zeta\left(\frac{\|\mathbf{x}^q - \mathbf{x}^s\|}{\sigma}\right) \boldsymbol{\gamma}_{\text{new}}^s = \boldsymbol{\omega}_\sigma(\mathbf{x}^q). \quad (17)$$

This scheme is best applied just after the particle redistribution scheme. The fact that the particles are then on a regular lattice greatly favors the good-quality reconstruction of a smooth function from particle strengths. It is also best to use the Gaussian smoothing as it too permits the good-quality reconstruction of a smooth function from particle strengths in quite a range of values for $0 < h/\sigma \leq 1$.

The W-scheme with Gaussian smoothing (and cutoff) amounts to solving a system of linear equations involving only neighbor particles. This is done using an iterative method such as relaxed-Jacobi (in the parallel code) or relaxed Gauss-Seidel. Notice that the matrix is not diagonally dominant. In fact, the smaller h/σ , the worse the non-diagonal dominance. At this point, the efficient iterative solution of this system is still a subject of active research within our group, e.g., through the development of an efficient preconditioner. This will be reported in another paper when the work is completed.

The above P- and W-schemes don't have general conservation properties (total vorticity, linear and angular impulse, energy, enstrophy). Nevertheless, our experience to date with the W-scheme is that it performs quite well when properly applied.

3.3 Time integration

For time integration, the $O((\Delta t)^2)$ Adams-Bashforth scheme (AB2) is used. Since this scheme is not self-starting, an $O((\Delta t)^2)$ Runge-Kutta scheme (RK2) is used for the first time step (after the initial condition or after each particle redistribution). This scheme allows one to maintain second order accuracy throughout. Numerical experiments have indeed shown that an $O(\Delta t)$ Euler scheme is simply not acceptable. The RK2 scheme is efficiently programed as follows: Euler predictor, trapezoidal rule corrector.

4 The BEM code so far

For the BEM method so far, we use the formulation with triangular panels of uniform strength [12, 13]. No optimization of the iterative method has been carried out so far. A relaxed Jacobi scheme is used in the parallel code. Two types of panels were used: scalar ‘source/sink’ strengths and vector ‘vortex sheet’ strengths. The BEM code was tested on the sphere and on ellipsoids at angle of attack, with very good convergence when the relaxation factor is set to 0.75. Even for flat ellipsoids ($a/b = 2$ and $b/c=2$), the convergence rate was very good. It is planned to further investigate and possibly improve the BEM code by possibly: (1) incorporating panels with linear variation of the strength, and (2) using more sophisticated iterative solvers.

5 Performance results for the VEM and BEM codes

For performance analysis of the VEM code, we consider vortex particles initially on the surface of the unit sphere ($R = 1$), and of strength corresponding to potential flow past the sphere with unit free stream velocity ($U_\infty = 1$) [29]. This initial condition then evolves dynamically, see Fig. 4. The sphere is discretized by recursively splitting the faces of an icosahedron into equilateral triangles, and then projecting them onto the sphere. This produces a uniform discretization of the sphere surface as all panels are close to equilateral triangles. For the VEM test, the triangles obtained at the lowest splitting level are replaced by vortex particles using the Gaussian smoothing with σ equal to the linear size of the unprojected triangles. For the BEM test, the triangles are the panels [41]. Ellipsoids can be obtained by linear stretching and squeezing of the sphere.

Performance results for computations performed on the Intel Touchstone Delta, an MIMD with up to 512 processors (intel i860 processors with 16 MB of RAM each), are presented in Fig. 2. A rough scaling for the CPU cost is obtained as $T \propto N^{1.1}/P^{0.9}$. The performance of the tree code is thus very good ($N^{1.1}$), and so is its parallel implementation ($P^{0.9}$), see [26, 28, 29, 34].

The tree code is written entirely in ANSI C and has been ported to several other parallel and sequential platforms [29, 35]. In particular, problems with $N = O(10^3 - 10^5)$ are now easily solved on the degenerate parallel case of single processor workstations, e.g., see Fig. 3.

Our basic VEM and BEM codes are $p = 2$, but we have also conducted tests to find what multipole expansion order p is near optimal. Three different VEM codes were produced and tested [42], each using the optimum error bound. For comparable levels of obtained accuracy (i.e., comparable \tilde{e}_{bound}), it was found that (a) the $p = 2$ and $p = 1$ codes perform almost equally and (b) they both outperform the $p = 0$ code. It thus appears that multipole expansions of order higher than $p = 2$ should not be used in 3D: the program complexity is increased considerably (e.g., multipole translations from children cells to parent cells, multipole evaluations, memory requirements) while the performance, at equal level of obtained accuracy, is not improved.

6 Sample computational results

One interesting problem was already described above: the time evolution of a spherical vortex sheet with, as initial condition, the vorticity distribution corresponding to potential flow past the unit sphere with unit free stream velocity. Initially, the vortex intensity is proportional to $\sin\theta$ where θ is the angle relative to the free stream. The run was done with 81,920 particles ($h \approx 0.016$), $\Delta t = 0.025$, $\sigma = 0.05$, $\alpha = 0.5$ and $e_{\text{tol}} = 0.001$. No diffusion, no particle redistribution, and no relaxation schemes were employed. Snapshots of the solution are shown in Fig. 4. Although of azimuthal symmetry initially, this problem goes unstable (here in mode 4) and the axisymmetry

is lost quite rapidly. Eventually, regions of very intense vorticity are formed and the flow becomes ‘turbulent’. There is a direct analogy between this problem and another problem in gas dynamics. The interaction of a weak shock wave with a spherical gas inhomogeneity (e.g., a spherical helium bubble in air) [11] will deposit, on the interface between the two gases, a vorticity layer of strength roughly proportional to $\sin \theta$. (Indeed, baroclinic generation of vorticity is proportional to $\nabla p \times \nabla \rho$ which, in first approximation, goes like $\sin \theta$.) The time evolution of this vorticity field is then very similar to our incompressible flow computation, see Fig. 8 in [11].

To illustrate the latest version of the VEM code, we consider a high resolution computation of the fusion of two vortex rings: radius $R = 1$, circulation $\Gamma = 1$, initial Gaussian vorticity distribution with $\sigma_R = 0.10$ (hence $\omega_{\max}^q(t = 0) = 15.92$), spacing of the two rings center to center $S = 2.70$, angle of each ring w.r.t. vertical of 20 degrees. Each ring is discretized with 126 sections and 225 particles per section (i.e., using 7 layers, see [40]) with $\omega_{\min}^q(t = 0) \approx 0.0065$. The inter-particle spacing is then $h \approx 0.05$. The computations were run with $\Delta t = 0.05$, $\sigma = 0.0625$, $\alpha = 0$, $\nu = 0.0025$ (i.e., $Re = \Gamma/\nu = 400$) and $e_{\text{tol}} = 0.0001$ on both 32 nodes of an IBM-SP2 and 64 nodes of an Intel Paragon. Initially, there were 56,700 particles (19 CPU seconds per step on SP2-32 and 68 on Paragon-64). The Λ_2 particle redistribution scheme with $h = 0.05$ was used every 10 time steps (with the smallest particle strength kept set to $\omega_{\min}^q = 0.001$). At the end of the run, there were 218,696 particles (87 CPU seconds per step on SP2-32 and 236 on Paragon-64). We obtained, for the mean over all elements, $\bar{e}_{\text{bound}} \approx 0.0006$. This computation was a ‘capability demonstration’ and a bit of an overkill: $e_{\text{tol}} = 0.001$ would likely have sufficed (leading to a factor of roughly two improvement in the CPU cost at equal number of particles), together with $\omega_{\min}^q = 0.01$ for the redistribution scheme (leading to a slower growth rate in the number of particles). It is also believed that the computation was over-resolved for the Reynolds number considered. It is seen in the histograms of Fig. 5 that the diffusion scheme, when combined with the high order particle redistribution scheme, correctly captures the fusion process: First, the energy and enstrophy losses associated with the Λ_2 scheme are so small that they cannot be seen in the histograms. (They can only slightly be seen when the histograms are differentiated numerically.) Second, the histogram of normalized energy decay rate follows the histogram of enstrophy, as it should. (A better overlap of the two histograms would likely be obtained by optimizing a bit the ratio h/σ .) For comparison, a run without particle redistribution was also done. In that case, the energy decay rate does not follow the enstrophy and is thus clearly incorrect. Finally, the conservation of linear impulse is also much improved by the use of the redistribution scheme. Yet, even with particle redistribution, linear impulse starts decreasing at $t \approx 4$. It is believed that the particle vorticity field is then beginning to deviate significantly from the divergence free vorticity field.

At this point, we are experimenting with the two relaxation schemes, P-scheme and W-scheme, when used in conjunction with the redistribution scheme. Results obtained so far are encouraging, yet too preliminary to be reported. We are also setting up for computations at higher Reynolds numbers and at higher resolution (e.g., with N in the range 10^6). Also of interest is the influence of the parameter α in the 3D stretching.

7 Conclusions

The VEM method has come a long way since its early stages: accurate viscous diffusion, fast and accurate field evaluation on both sequential and parallel platforms, particle redistribution schemes, relaxation schemes for the particle vorticity field. This work is still in progress. It is however believed that the progress made so far, combined with more recent developments in vortex techniques for wall-bounded flows [16–18, 23, 39, 40], will soon permit the simulation of 3D unsteady

problems of engineering interest: flow past bluff bodies or past streamlined bodies at high angle of attack, including vortex wake. These body/wake computations will require the merging of the VEM code with the BEM code: need to determine, at each time step, the vorticity flux necessary at solid boundaries in order to satisfy the no-slip boundary condition.

References

- [1] Abramowitz, M. and Stegun, I.E., Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables, National Bureau of Standards, Applied Mathematics Series 55, Tenth Printing, 1972.
- [2] Appel, A.W., “An Efficient Program for Many-Body Simulation,” *SIAM J. Sci. Stat. Comp.*, **6**, pp. 85, 1985.
- [3] Barnes, J.E. and Hut, P., “A Hierarchical $O(N \log N)$ Force-Calculation Algorithm,” *Nature*, **324**, pp. 446–449, 1986.
- [4] Degond, P. and Mas-Gallic, S., “The Weighted Particle Method for Convection-Diffusion Equations, Part I: The Case of an Isotropic Viscosity, Part II: The Anisotropic Case,” *Math. Comp.*, **53**, pp. 485–526, 1989.
- [5] Ding, H.-Q., Karasawa, N. and Goddard, W., “Atomic Level Simulations of a Million Particles: The Cell Multipole Method for Coulomb and London Interactions,” *J. of Chemical Physics*, **97**, pp. 4309–4315, 1992.
- [6] Dubinski, J. and Carlberg, R.G., “The Structure of Cold Dark Matter Halos,” *Ap. J.*, **378**, pp. 496, 1991.
- [7] Greengard, L., The Rapid Evaluation of Potential Fields in Particle Systems, Ph.D. thesis, Yale University, 1987. Published by MIT Press, Cambridge, 1988.
- [8] Greengard, L., “The Numerical Solution of the N-Body Problem,” *Comp. Phys.*, **4**, pp. 142–152, 1990.
- [9] Greengard, L. and Rokhlin, V., “A Fast Algorithm for Particle Simulations,” *J. Comp. Phys.*, **73**, pp. 325, 1987.
- [10] Greengard, L. and Rokhlin, V., “On the Evaluation of Electrostatic Interactions in Molecular Modeling,” *Chemica Scripta*, **29 A**, pp. 139–144, 1989.
- [11] Hass, J.-F. and Sturtevant, B., “Interaction of Weak Shock Waves with Cylindrical and Spherical Gas Inhomogeneities,” *J. Fluid Mech.*, **181**, pp. 41–76, 1987.
- [12] Hess, J.L. and Smith, A.M.O., “Calculation of Potential Flow about Arbitrary Bodies,” *Prog. Aeronaut. Sci.*, **8**, pp. 1–138, 1966.
- [13] Hess, J.L., “Panel Methods in Computational Fluid Dynamics,” *Ann. Rev. Fluid Mech.*, **22**, pp. 255–274, 1990.
- [14] Katz, N., Hernquist, L. and Weinberg, D.H., “Galaxies and Gas in a Cold Dark Matter Universe,” *Ap. J.*, **399**, pp. L109, 1992.

- [15] Kellogg, O.D., Foundations of Potential Theory, Dover Press, New York, 1953.
- [16] Koumoutsakos, P. and Leonard, A., “Direct Numerical Simulations using Vortex Methods,” *Proc. NATO Adv. Res. Workshop: Vortex Flows and Related Numerical Methods*, Grenoble, France, edited by Goule, Cottet and Huberson, Kluwer, pp. 179–190, 1992.
- [17] Koumoutsakos, P., Direct Numerical Simulations of Unsteady Separated Flows using Vortex Methods, Ph.D. thesis, California Institute of Technology, 1993.
- [18] Koumoutsakos, P. and Leonard, A., “High-Resolution Simulations of the Flow around an Impulsively Started Cylinder using Vortex Methods,” *J. Fluid Mech.*, **296**, pp. 1–38, 1995.
- [19] Leonard, A., “Review: Vortex Methods for Flow Simulation,” *J. Comp. Phys.*, **37 (3)**, pp. 289–335, 1980.
- [20] Leonard, A., “Computing Three-Dimensional Incompressible Flows with Vortex Elements,” *Ann. Rev. Fluid Mech.*, **17**, pp. 523–559, 1985.
- [21] Mas-Gallic, S., Contribution à l’analyse numérique des méthodes particulières, Thèse d’État, Université Paris VI, 1987.
- [22] Pedrizzetti, G., “Insight into Singular Vortex Flows,” *Fluid Dyn. Res.*, **10**, pp. 101–115, 1992.
- [23] Pépin, F., Simulation of Flow past an Impulsively Started Cylinder using a Discrete Vortex Method, Ph.D. thesis, California Institute of Technology, 1990.
- [24] Rokhlin, V., “Rapid Solution of Integral Equations of Classical Potential Theory,” *J. Comp. Phys.*, **60**, pp. 187–207, 1985.
- [25] Saffman, P.G. and Meiron, D.I., “Difficulties with Three-Dimensional Weak Solutions for Inviscid Incompressible Flow,” *Phys. Fluids*, **29 (8)**, pp. 2373–2375, 1986.
- [26] Salmon, J.K., Parallel Hierarchical N-body Methods, Ph.D. thesis, California Institute of Technology, 1990.
- [27] Salmon, J.K., Quinn, P.J. and Warren, M.S., “Using Parallel Computers for Very Large N-Body Simulations: Shell Formation using 180k Particles,” *Heidelberg Conference on Dynamics and Interactions of Galaxies*, New York, edited by R. Wielen, Springer-Verlag, pp. 216–218, 1990.
- [28] Salmon, J.K. and Warren, M.S., “Skeletons from the Treecode Closet,” *J. Comp. Phys.*, **111 (1)**, pp. 136–155, 1994.
- [29] Salmon, J.K., Warren, M.S. and Winckelmans, G.S., “Fast Parallel Tree Codes for Gravitational and Fluid Dynamical N-Body Problems,” *Int. J. Supercomputer Appl.*, **8 (2)**, pp. 129–142, 1994.
- [30] Schmidt, K. and Lee, M.A., “Implementing the Fast Multipole Method in Three Dimensions,” *J. Stat. Phys.*, **63 (5/6)**, pp. 1223–1235, 1991.
- [31] Sugihara, T., Suto, Y., Bouchet, F.R. and Hernquist, L., “Cosmological N-Body Simulations with a Tree Code: Fluctuations in the Linear and Nonlinear Regimes,” *Ap. J. Suppl.*, **75**, pp. 631, 1991.

- [32] Warren, M.S., Quinn, P.J., Salmon, J.K. and Zurek, W.H., “Dark Halos Formed via Dissipationless Collapse: I. Shapes and Alignment of Angular Momentum,” *Ap. J.*, **399**, pp. 405–425, 1992.
- [33] Warren, M.S. and Salmon, J.K., “Astrophysical N-Body Simulations using Hierarchical Tree Data Structures,” *Supercomputing '92*, Los Alamitos, IEEE Comp. Soc., 1992.
- [34] Warren, M.S. and Salmon, J.K., “A Parallel Hashed Oct-Tree N-Body Algorithm,” *Supercomputing '93*, Los Alamitos, IEEE Comp. Soc., 1993.
- [35] Warren, M.S. and Salmon, J.K., “A Parallel, Portable and Versatile Treecode,” *Proc. Seventh SIAM Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, pp. 319–324, 1995.
- [36] Winckelmans, G.S. and Leonard, A., “Weak Solutions of the Three-Dimensional Vorticity Equation with Vortex Singularities,” *Phys. Fluids, Letters*, **31 (7)**, pp. 1838–1839, 1988.
- [37] Winckelmans, G.S. and Leonard, A., “Improved Vortex Methods for Three-Dimensional Flows,” *SIAM Workshop on Mathematical Aspects of Vortex Dynamics*, Leesburg, VA, April, 1988, *SIAM Proc. Series*, edited by R.E. Caflisch, pp. 25–35, 1989.
- [38] Winckelmans, G.S., Topics in Vortex Methods for the Computation of Three- and Two-Dimensional Incompressible Unsteady Flows, Ph.D. thesis, California Institute of Technology, 1989.
- [39] Winckelmans, G.S., “Comments on a Paper by Kiya et al. on the Numerical Simulation of Pseudo-Elliptical Vortex Rings using the Vortex Particle Method,” *Fluid Dyn. Res., Brief Comm.*, **12**, pp. 57–60, 1993.
- [40] Winckelmans, G.S. and Leonard, A., “Contributions to Vortex Particle Methods for the Computation of Three-Dimensional Incompressible Unsteady Flows,” *J. Comp. Phys.*, **109 (2)**, pp. 247–273, 1993.
- [41] Winckelmans, G.S., Salmon, J.K., Warren, M.S. and Leonard, A., “The Fast Solution of Three-Dimensional Potential Flow Boundary Integral Equations using Parallel and Sequential Tree Codes,” Caltech Internal Report, to be submitted.
- [42] Winckelmans, G.S., Salmon, J.K., Warren, M.S. and Leonard, A., “The Fast Solution of Three-Dimensional Fluid Dynamical N-Body Problems using Parallel Tree Codes: Vortex Element Method and Boundary Element Method,” *Proc. Seventh SIAM Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, pp. 301–306, 1995.
- [43] Zhao, F., An $O(N)$ Algorithm for Three-Dimensional N-Body Simulations, Master’s thesis, Massachusetts Institute of Technology, 1987.

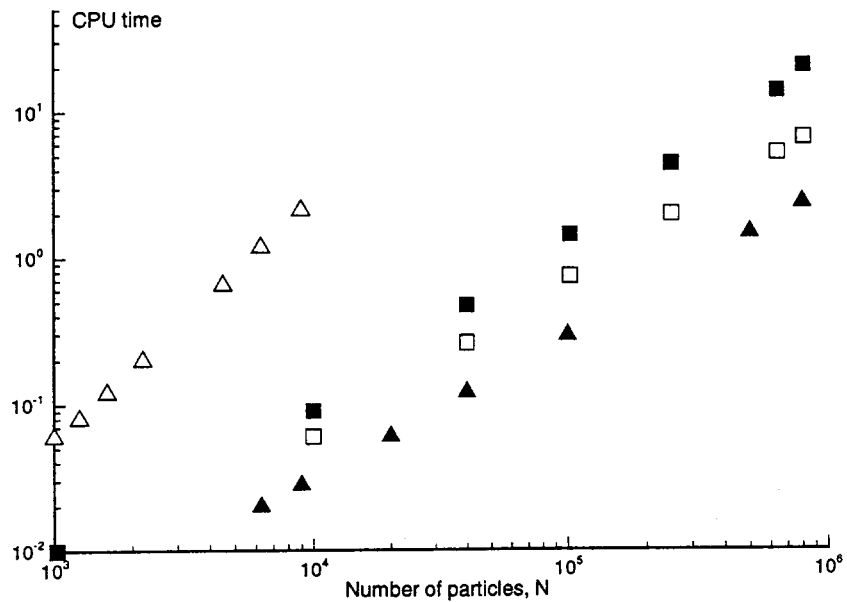


Figure 1: Performance, on an IBM POWERserver 590 (SPECfp92: 260) with 128 MB of RAM, of the two approaches for the particle redistribution scheme on two sample 2D configurations: full configuration with particles filling up a whole square box (squares) and sparse configuration with particles on the perimeter of a circle (triangles); matrix-approach (hollow) and tree-approach (solid).

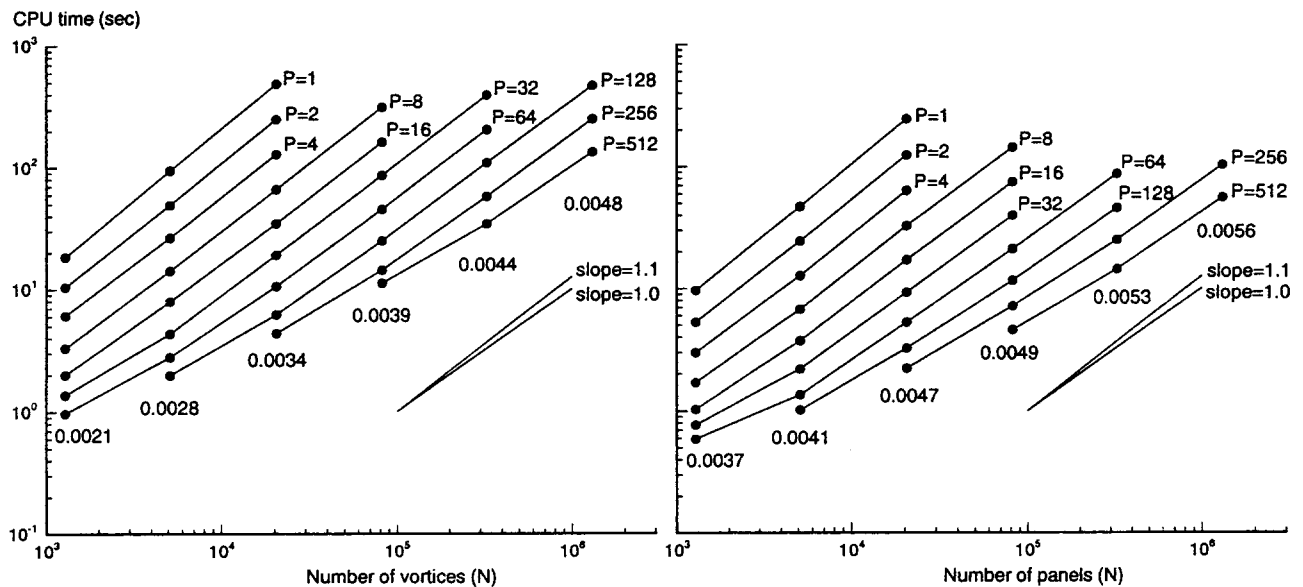


Figure 2: Performance study of VEM and scalar BEM $p = 2$ codes on the Intel Touchstone Delta. Ranges are from $P = 1$ to $P = 512$ processors, and from $N = 1,280$ to $N = 1,310,720$ elements; $e_{tol} = 0.001$; also reported is the mean, over all elements, of \bar{e}_{bound} .

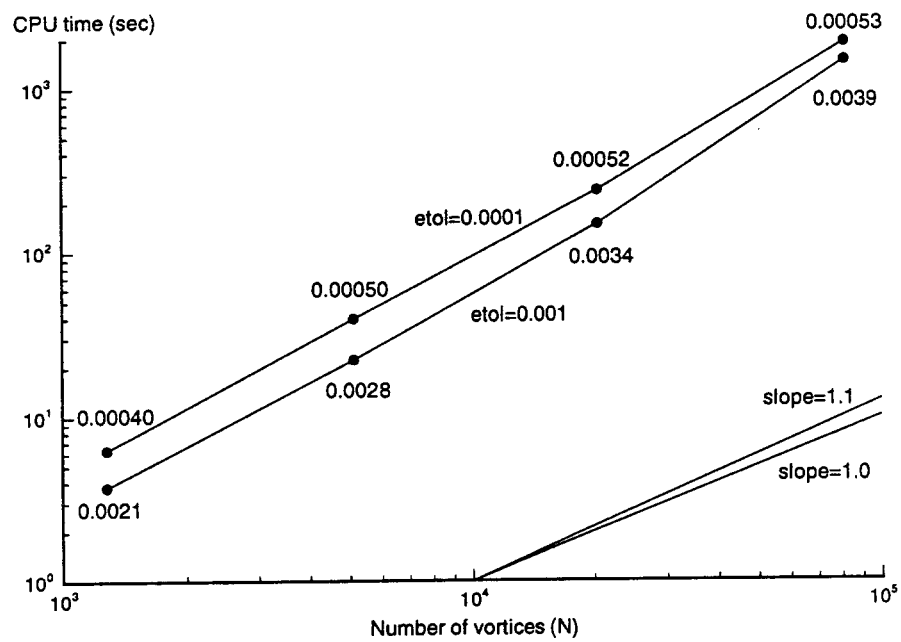


Figure 3: Performance study of the VEM $p = 2$ code on an IBM POWERstation 375 (SPECfp92: 120) with 64 MB of RAM. Range is from $N = 1,280$ to $N = 81,920$ elements; $e_{tol} = 0.001$ and $e_{tol} = 0.0001$; also reported is the mean, over all elements, of \tilde{e}_{bound} .

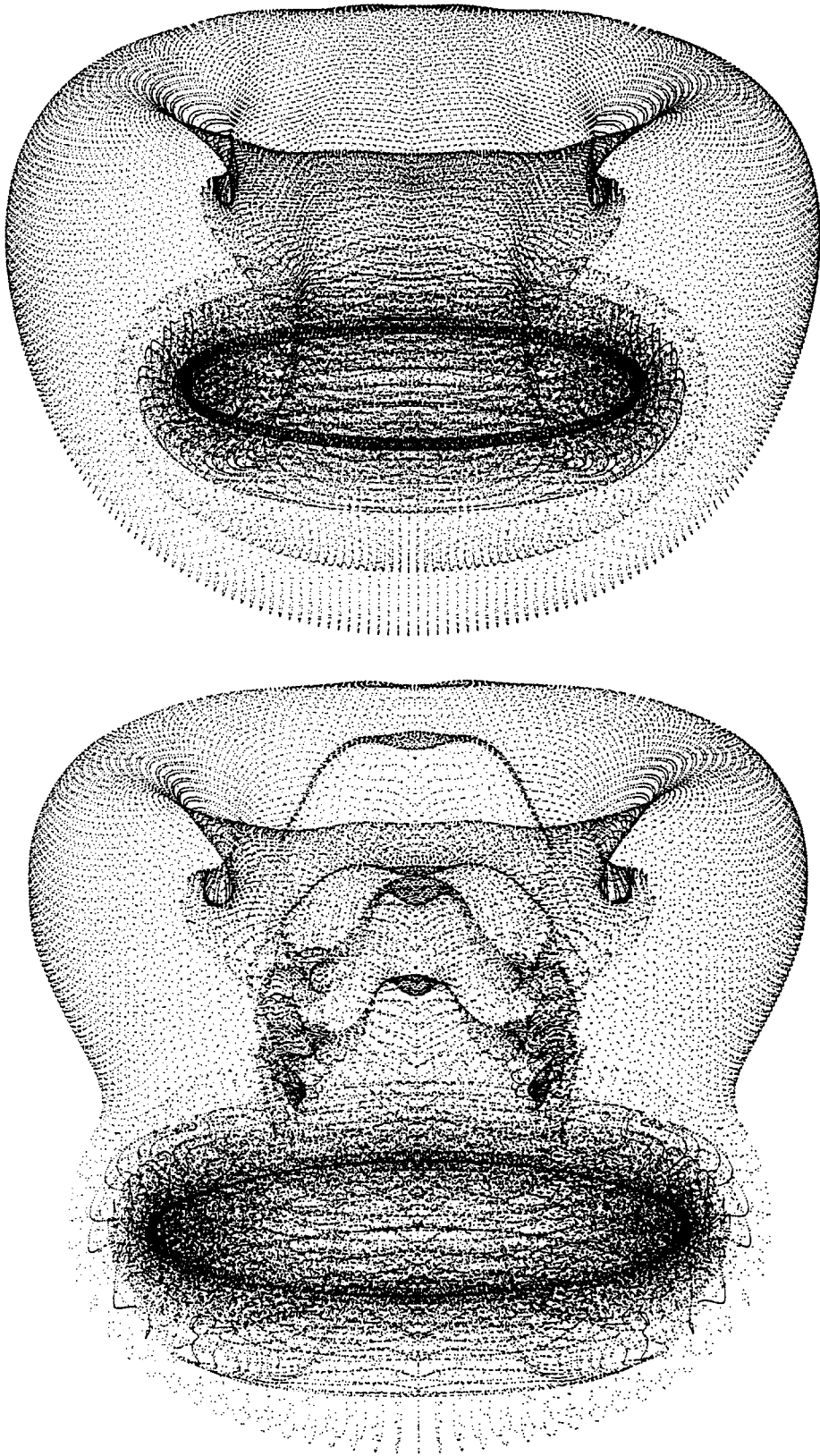


Figure 4: Time evolution of a spherical vortex sheet with, as initial condition, the vorticity distribution corresponding to potential flow past the unit sphere with unit free stream velocity. 3D view of the particle positions at $t = 2.5$ and $t = 5.0$.

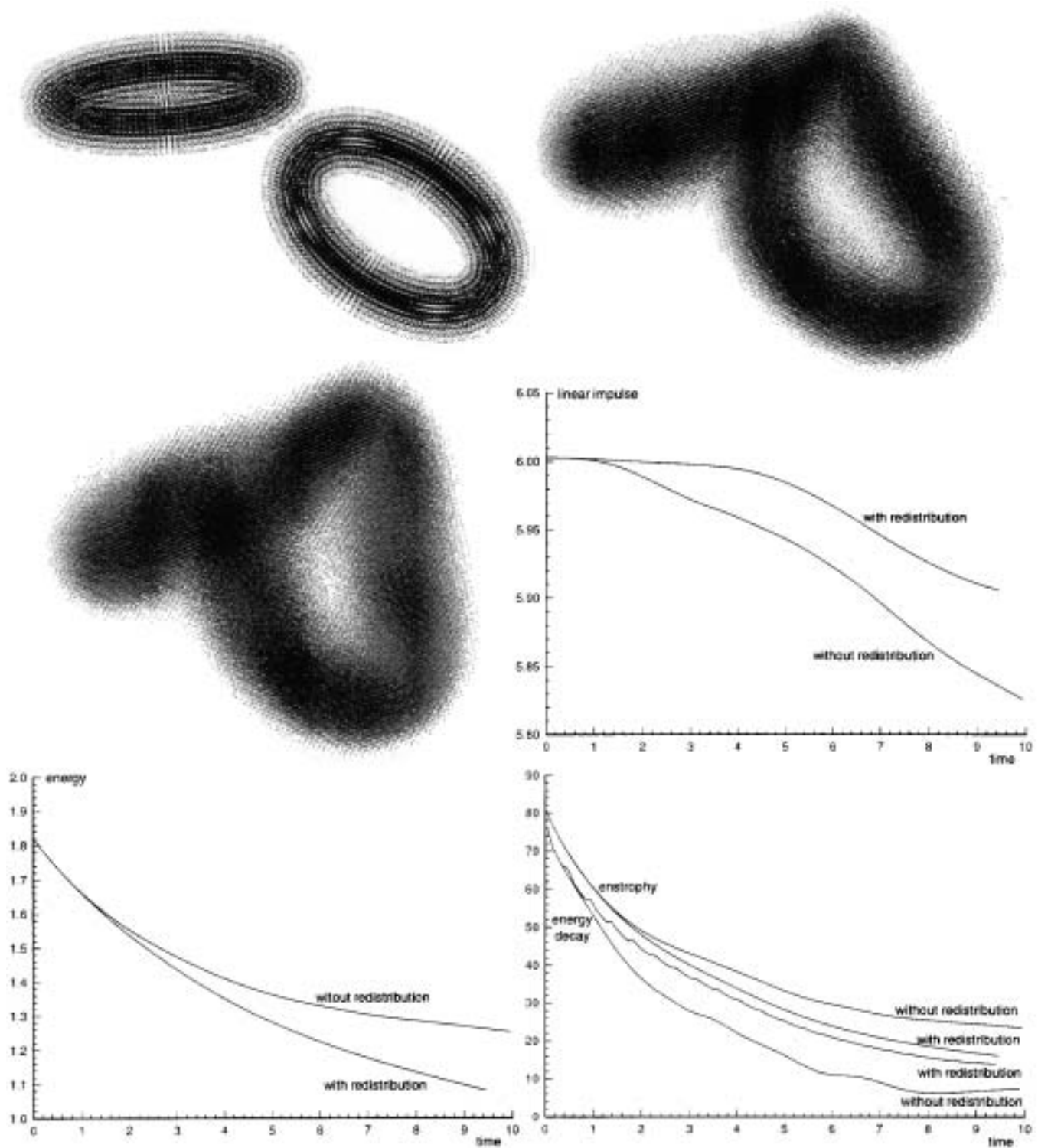


Figure 5: Fusion of two vortex rings. 3D view of the particle strength vectors at $t = 0.0, 5.4, 9.0$; Histogram of linear impulse, I_z ; Histogram of energy, E ; Histogram of enstrophy, \mathcal{E} , and of normalized energy decay rate, $-\frac{1}{\nu} \frac{dE}{dt}$.