Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

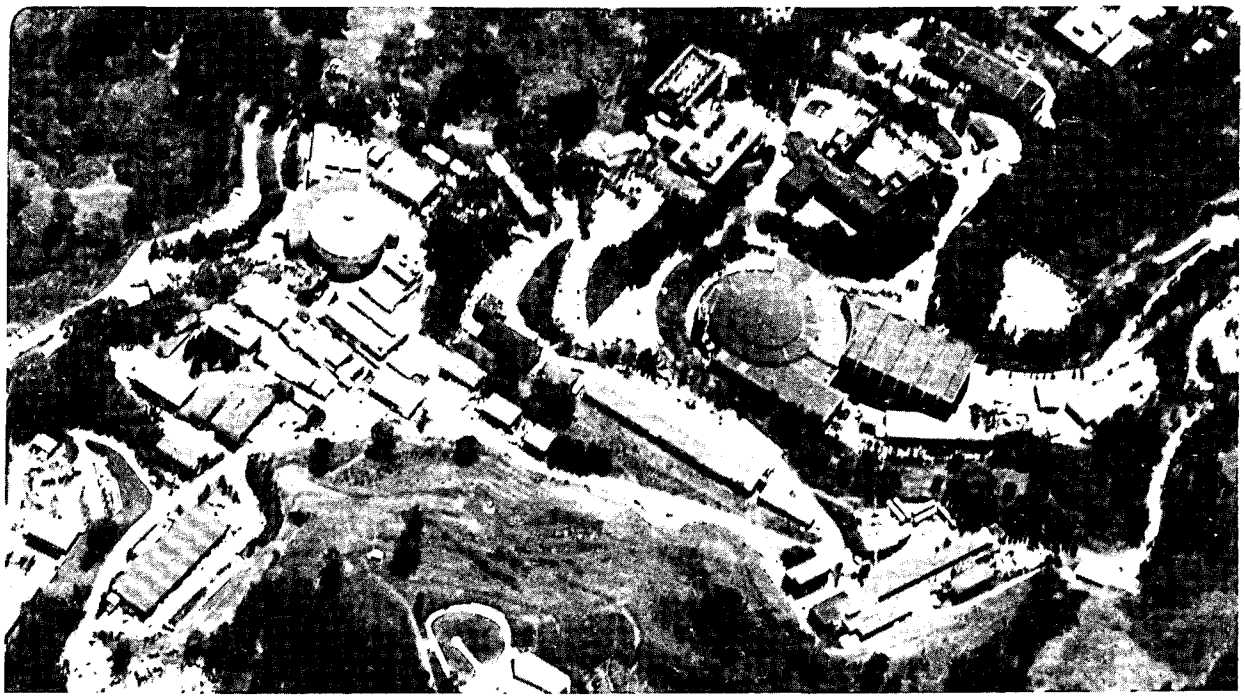
Physics Division

Mathematics Department

Higher Order Vortex Methods with Rezoning

H.O. Nordmark
(Ph.D. Thesis)

May 1988



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

DISCLAIMER

LBL-25259

HIGHER ORDER VORTEX METHODS WITH REZONING[1]

Henrik Olov Nordmark

Department of Mathematics
and
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720, USA

Ph.D. Thesis

May 1988

[1]Supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098.

Higher Order Vortex Methods with Rezoning

Henrik Olov Nordmark

Abstract

The vortex method is a numerical method for approximating the flow of an incompressible, inviscid fluid. We consider the two-dimensional case. The accuracy depends on the choice of the cutoff function which approximates the delta function, on the cutoff parameter δ and on the smoothness of the initial data. We derive a class of infinite-order cutoff functions with arbitrarily high rates of decay at infinity. We also derive an eighth order cutoff function with compact support. We test two versions of rezoning. Version 1 has been suggested and tested by Beale and Majda, while version 2 is new. Using rezoning, we test the eighth order cutoff function and one infinite-order cutoff function on three test problems for which the solution of Euler's equation is known analytically. The accuracies of the two methods are comparable. We also compute the evolution of two circular vorticity patches and the evolution of one square vorticity patch over long time intervals. Finally, we make a comparison between the direct method of velocity evaluation and the Rokhlin-Greengard algorithm. The numerical experiments indicate that for smooth flows, high-order cutoffs combined with rezoning give high accuracy for long time integrations.

Acknowledgements

The author wishes to thank Ole Hald for suggesting the topic for this thesis, for interesting discussions and helpful ideas, and for his patience throughout the period of research and writing of this thesis. Many thanks to James Sethian for having served on my committee and for reading and commenting on my thesis under great pressure of time. Many thanks also to Stanley Berger for serving on my committee and for reading my thesis with short notice.

A special word of thanks goes to my wife Teresa Díaz-González de Nordmark for her great moral support, help and patience throughout my years at Berkeley. I also thank Paul Concus, Scott Baden and Gerry Puckett for their advice on computer related matters.

The calculations presented in this thesis were carried out at the Lawrence Berkeley Laboratory.

Introduction

The vortex method is a numerical method for approximating the flow of an incompressible fluid without viscosity. Thus we assume that the flow is governed by the Euler equation rather than the full Navier-Stokes equation. The idea is to approximate a vorticity distribution by a finite set of "vortex blobs" which are multiples and translates of a certain function known as the cutoff function. The cutoff function is scaled by a parameter δ and approximates the delta function as δ approaches 0. The vortex blobs induce a velocity field, which in turn moves the vortex blobs. The evolution of the vortex blobs is computed by solving a system of ordinary differential equations by standard numerical methods. In this form, the vortex method was introduced by Chorin [10] in 1973, but its predecessor, the point vortex method, was introduced about 40 years earlier by Rosenhead [24] for calculating the behavior of vortex sheets. The point vortex method gives unreliable results however, see e.g. Beale and Majda [8]. There have been many applications of vortex methods, including the simulation of turbulent combustion in open and closed vessels, Sethian [25], the computation of unstable boundary layers, Chorin [11], aerodynamic computations, Cheer [9], Spalart [27], Leonard and Spalart [20], and flow of variable density, Anderson [2].

The vortex method can be extended to simulate viscous flow by letting each vortex take a step of a specified length in a random direction after each timestep. Recently, Sethian and Ghoniem [26] tested this procedure on viscous flow through a channel over a backwards-facing step. A variety of different Reynolds numbers were used, corresponding to flows ranging from laminar flow to completely turbulent flow and including the transition region between laminar and turbulent flow. Sethian and Ghoniem [26] tested many different combinations of numerical parameters and found that for laminar flow, the only parameter that significantly affected the numerical solution was the number of vortices. For non-laminar flow, they found that the size of the timestep was also critical. A larger number of vortices requires a smaller timestep. In both cases, the numerical results demonstrated the convergence of the method as the number of vortices goes to infinity. However, in sharp contrast to the inviscid case, the size of the cutoff parameter δ turned out to be of secondary importance. To avoid confusion we point out that for the remainder of this thesis, we consider the vortex method for inviscid flow.

In the late nineteen-seventies, del Prete and Hald [17] gave a convergence proof for the 2-D vortex method, for a short time interval, but under the mild assumption that the initial vorticity distribution is Hölder continuous. By requiring more smoothness of the vorticity, i.e. three continuous derivatives, Hald [15] subsequently proved convergence for arbitrarily long time intervals. Beale and Majda [6,7] gave convergence proofs for the three dimensional case as well as the two dimensional case. Cottet [14] provided a simpler proof of Beale and Majda's convergence theorem, which was simplified further by Anderson and Greengard [3]. Anderson and Greengard [3] also established the convergence of the time discretization for a large class of multistep methods and for second order Runge-Kutta methods. Hald [16] then proved fourth order convergence of the time discretization for the classical fourth order Runge-Kutta method, provided the flow is smooth enough.

The accuracy of the vortex method depends on how the delta function is approximated, which in turn depends both on the choice of cutoff function and on the choice of the parameter δ. Beale and Majda [8] introduced a family of smooth cutoff functions, with unbounded support, but decaying very rapidly at infinity. From this family, we can pick an n-th order cutoff function, where n is any desired positive integer, and by Beale and Majda's [7] convergence theorem, obtain a vortex method of order very close to n, if the flow is infinitely differentiable and if we pick δ close to h, where h is the original distance between adjacent vortices. However, numerical experiments by Perlman [22] have shown that for reasonable values of h, this is only true for short time intervals. In practice, we have to take δ proportional to h^q, with q less than 1 by a fair amount, and get a method of order nq if the flow is sufficiently smooth. Hald [16] presented several infinite-order cutoff functions. The rates of convergence for these cutoffs are only limited by the degree of smoothness of the flow. In this thesis, we test the practical accuracy of one of these cutoff functions for flows of different degrees of smoothness. Following Hald's recipe [16, p.567], we derive a large class of explicit infinite-order cutoff functions and velocity kernels with higher rates of decay at infinity. Our numerical results show that we *do* get orders of accuracy slightly exceeding the ones predicted by Hald's theory, but only for short time integrations. The deterioration in accuracy at later times observed by Perlman [22] is even more pronounced for infinite-order cutoff functions. A natural way to overcome this difficulty, is to use the rezoning technique. It was suggested and tested by Beale and Majda [8]. In this thesis, we present two versions of rezoning. The first

version is that of Beale and Majda [8], but with the added feature of a "built-in" criterion for determining at which times we introduce a new grid. The second version also has this feature. It is more accurate because it uses more vortices, but it costs more. A different method of improving the accuracy for large time integrations was recently introduced by Beale [5]. We made a small number of numerical tests on this method and found that it is significantly more accurate than the standard method but not as accurate as the method of rezoning. We do not present these tests here because we feel they are not sufficiently complete.

It follows from Hald's theory [16], that we should take δ proportional to $\sqrt{h}$ when using infinite order methods. However, it is not clear what the optimal proportionality constant is. That depends on a number of factors. First of all, in vortex methods without rezoning we always need to use a larger proportionality constant, for large integration times. Secondly, the choice depends somewhat on the form of the initial vorticity distribution. Perlman [22] observed that the choice of δ is essentially independent of the smoothness of the flow. The numerical results in this thesis show that although this seems to be true for radially symmetric vorticity distributions, it is not necessarily the case in general. Finally, the choice of proportionality constant depends strongly on the cutoff function and especially on the value of the cutoff function at the origin. For example, for the eighth order cutoff function derived in this thesis, we have to take a proportionality constant that is about 5.5 times larger than for Hald's infinite order cutoff function. This is due to the fact that the value of the first cutoff at the origin is about 30 times larger than for the second cutoff. A partial list of numerical experiments that test the accuracy of vortex methods includes del Prete and Hald [17], Beale and Majda [8], Beale [5], Perlman [22] and Nakamura, Leonard and Spalart [21].

Besides the accuracy of vortex methods, the computational speed is also important. The standard direct method of computation requires $O(N^2)$ flops, where N is the number of vortices. Anderson [1] has introduced a faster method, known as the method of local corrections. It requires $O(N \log N)$ flops *provided* δ *is proportional to* h, and uses a "fast Poisson solver" and interpolations. The practical speed of this method on a Cray computer has recently been tested by Baden [4]. This method may however introduce a significant amount of additional errors when the order of the cutoff function is high enough and the flow is sufficiently smooth. Another fast algorithm known as the method of multipole expansions, has

recently been introduced by Rokhlin and Greengard [23]. This method requires $O(N)$ flops for δ proportional to h, and when it is applicable, it is essentially as accurate as the direct method. Both of these fast methods are only strictly applicable when using cutoff functions with compact support. For this reason, we derive an eighth order cutoff function with compact support in this thesis. We test the Rokhlin-Greengard algorithm [23] using this cutoff function.

This thesis is divided into 6 chapters. In chapter 1 we present the derivation of the vortex method in two dimensions. In chapter 2 we derive a large class of infinite-order cutoff functions, present Hald's [16] convergence theorem for infinite-order methods, and give 3 examples of infinite-order cutoff functions from the large class. Chapter 3 deals with the economical numerical evaluation of infinite-order cutoff functions and velocity kernels. In chapter 4 we derive an eighth order cutoff function with compact support and compare it with Hald's infinite-order cutoff. In chapter 5, two versions of the method of rezoning are described, and finally, in chapter 6 we present our test problems and numerical results.

1. The Basic Equations

The vorticity-stream function form of Euler's equations in two dimensions is

$$\omega_t + (u\cdot\nabla)\omega = 0, \tag{1.1}$$

$$\Delta\psi = -\omega, \tag{1.2}$$

$$u = \psi_y, \quad v = -\psi_x, \tag{1.3}$$

where $\mathbf{u} = (u,v)$ is the velocity vector, $\mathbf{x} = (x,y)$ is the position vector, ω is the vorticity, and ψ is the stream function.

The solution of the Poisson equation (1.2) is given by

$$\psi(\mathbf{x}) = \int_{\Omega(t)} G(\mathbf{x}-\mathbf{x}')\,\omega(\mathbf{x}',t)\,d\mathbf{x}'. \tag{1.4}$$

where $G(\mathbf{x}) = -(2\pi)^{-1}\ln|\mathbf{x}|$, with $|\mathbf{x}|^2 = x^2+y^2$ is the fundamental solution of the 2-D Laplace equation , see [19, p.75], $d\mathbf{x}' = dx'dy'$, and $\Omega(t)$ denotes the support of ω in R^2 at time t. Using (1.3) and differentiating under the integral sign in (1.4) we get the velocity as

$$\mathbf{u}(\mathbf{x},t) = \int_{\Omega(t)} K(\mathbf{x}-\mathbf{x}')\,\omega(\mathbf{x}',t)\,d\mathbf{x}', \tag{1.5}$$

where

$$K(\mathbf{x}) = \frac{1}{2\pi|\mathbf{x}|^2}\begin{bmatrix} -y \\ x \end{bmatrix}.$$

In the Lagrangian description of the flow, we follow the motion of fluid particles. Let $\alpha = (\alpha_1,\alpha_2)$ be the Lagrangian coordinates of a particle starting at $\mathbf{x} = \alpha$ at time t=0. Then the path of that particle is determined by

$$\frac{d\mathbf{x}(\alpha,t)}{dt} = \mathbf{u}(\mathbf{x}(\alpha,t),t), \qquad \mathbf{x}(\alpha,0) = \alpha. \tag{1.6}$$

Equation (1.1) implies that the vorticity is preserved along particle paths, i.e. $\omega(\mathbf{x}(\alpha,t),t) = \omega(\alpha,0)$ for all t, see Chorin and Marsden [12, p.34]. Since the flow is incompressible, the Jacobian of the change of variables from $\mathbf{x}$ to α is 1, so we can rewrite (1.5) as

$$\mathbf{u}(\mathbf{x}(\alpha,t),t) = \int_{\Omega(0)} K(\mathbf{x}(\alpha,t)-\mathbf{x}(\beta,t))\,\omega(\beta,0)\,d\beta. \tag{1.7}$$

To discretize the system (1.6), (1.7) we cover the α plane by a square grid, with mesh length h.The coordinates of the grid points are then $\mathbf{j}h = (j_1, j_2)h$. Let $\mathbf{J}$ be the set of all double indices $\mathbf{j} = (j_1, j_2)$ such that $\mathbf{j}h \in \Omega(0)$, let $\mathbf{x}_\mathbf{j}(t)$ be the position of a particle starting at the point $\mathbf{j}h$ at time t=0, and let $\mathbf{u}_\mathbf{j}(t)$ be the velocity at $\mathbf{x}_\mathbf{j}(t)$ at time t.

One way to discretize the system (1.6), (1.7) is to replace the continuous indices α and β by the integer indices $\mathbf{i}$ and $\mathbf{j}$, and to replace the integral by a sum. This gives us the following system of ordinary differential equations

$$\frac{d\tilde{\mathbf{x}}_\mathbf{i}(t)}{dt} = \tilde{\mathbf{u}}_\mathbf{i}(t), \qquad \tilde{\mathbf{x}}_\mathbf{i}(0) = \mathbf{i}h, \tag{1.8}$$

where

$$\tilde{\mathbf{u}}_\mathbf{i}(t) = \sum_{\mathbf{j}\in\mathbf{J},\,\mathbf{j}\neq\mathbf{i}} K(\tilde{\mathbf{x}}_\mathbf{i}(t) - \tilde{\mathbf{x}}_\mathbf{j}(t))c_\mathbf{j}. \tag{1.9}$$

Here the "vorticity coefficients" $c_\mathbf{j}$ can be defined either by

$$c_\mathbf{j} = \omega(\mathbf{j}h)h^2$$

or by

$$c_\mathbf{j} = \int_{S_\mathbf{j}} \omega(\mathbf{x})d\mathbf{x},$$

where $S_\mathbf{j}$ denotes a square of length and width h centered at $\mathbf{j}h$. If we use the latter definition, the definition of $\mathbf{J}$ has to be changed. Cottet [14] has shown that the latter definition of $c_\mathbf{j}$ leads to an additional error of order $O(h^2)$. This has also been demonstrated numerically by Perlman [22]. Therefore we will always let $c_\mathbf{j} = \omega(\mathbf{j}h)h^2$.

The numerical solution of (1.8), (1.9) is known as the point vortex method, and was introduced in 1932 by Rosenhead [24] for the study of vortex sheets. It turns out that this method gives unreliable results, especially for calculating velocities off vortex paths. See for example Beale and Majda [8]. The reason for this is that $K(\mathbf{x})\to\infty$ as $\mathbf{x}\to\mathbf{0}$. Chorin [10] avoided this problem by replacing the kernel K by a kernel K_δ which is bounded at $\mathbf{x}$=$\mathbf{0}$. K_δ is the convolution of K and a smooth cutoff function Ψ_δ. i.e.

$$K_\delta(\mathbf{x}) = \int K(\mathbf{x}-\mathbf{x}')\,\Psi_\delta(\mathbf{x}')\,d\mathbf{x}'.$$

Here Ψ_δ is defined by $\Psi_\delta(\mathbf{x}) = \delta^{-2}\Psi(\mathbf{x}/\delta)$ where Ψ is a smooth radially symmetric function satisfying

$$\int_{R^2} \Psi(\mathbf{x}) d\mathbf{x} = 1$$

Hence, Ψ_δ approximates the Dirac delta function as $\delta \to 0$. Now the system (1.8), (1.9) is replaced by

$$\frac{d\bar{\mathbf{x}}_i(t)}{dt} = \tilde{\mathbf{u}}_i(t), \qquad \bar{\mathbf{x}}_i(0) = \mathbf{i}h, \tag{1.10}$$

where

$$\tilde{\mathbf{u}}_i(t) = \sum_{j \in J,\, j \neq i} K_\delta(\bar{\mathbf{x}}_i(t) - \bar{\mathbf{x}}_j(t)) c_j. \tag{1.11}$$

The numerical solution of this new system is known as the vortex blob method, or just vortex method. By imposing additional conditions on the cutoff function Ψ one can obtain high rates of convergence for this method. In this paper we derive a class of infinite order cutoff functions and an eighth order cutoff function with compact support.

2. Derivation of a Large Class of Infinite Order Cutoff Functions

Following Beale, Majda [6] and Hald [16], we define a general infinite order cutoff function Ψ via its Fourier transform $\hat{\Psi}$. Here

$$\Psi(\mathbf{x}) = \int e^{i\mathbf{x}\cdot\mathbf{k}}\,\hat{\Psi}(\mathbf{k})\,d\mathbf{k} \tag{2.1}$$

$$\hat{\Psi}(\mathbf{k}) = \frac{1}{(2\pi)^2}\int e^{-i\mathbf{x}\cdot\mathbf{k}}\,\Psi(\mathbf{x})\,d\mathbf{x} \tag{2.2}$$

where $\mathbf{x}\cdot\mathbf{k} = x_1k_1+x_2k_2$, $d\mathbf{x}=dx_1dx_2$ and $d\mathbf{k}=dk_1dk_2$. We assume that $\hat{\Psi}$ satisfies the following assumption,

(*i*) $\hat{\Psi}(t) = (2\pi)^{-2}$ for $0\le t\le 1$

(*ii*) $\hat{\Psi}(t) = 0$ for $t\ge b$.

(*iii*) *$\hat{\Psi}$ is real–valued and continuous for all t, continuously differentiable for $1\le t\le b$ and $\hat{\Psi}'$ is piecewise differentiable in the same interval.*

Hald [16] has shown that the previous assumption implies the following conditions:

(*i*) $|\psi^{(n)}(r)| \le L_0 r^{-(n+1)}$, $0<r\le 1$, $n=0,1$

(*ii*) $|\psi^{(n)}(r)| \le L_0 r^{-2.5}$, $1<r<\infty$, $n=0,1$

(*iii*) $\left|2\pi\int_0^r s\Psi(s)\,ds - 1\right| \le L_0 r^{-1.5}$ $0<r<\infty$.

Now, in order to simplify (2.1), we switch to polar coordinates. Let $(k_1,k_2) = t(\cos\phi,\sin\phi)$, $(x_1,x_2) = r(\cos\theta,\sin\theta)$. Then $dk_1dk_2=tdtd\phi$, and since $\hat{\Psi}(t) = 0$ for $t\ge b$, we get

$$\Psi(r,\theta) = \int_0^\infty\int_0^{2\pi} e^{i(rt\cos(\theta)\cos(\phi)+rt\sin(\theta)\sin(\phi))}\,\hat{\Psi}(t)t\;d\phi\,dt$$

$$= \int_0^\infty \hat{\Psi}(t)t\Big(\int_0^{2\pi} e^{i(rt\cos(\phi-\theta))}\,d\phi\Big)\,dt$$

$$= \int_0^b \hat{\Psi}(t)t\Big(\int_0^{2\pi} e^{i(rt\cos(\phi-\theta))}\,d\phi\Big)\,dt$$

The last integral is independent of θ because $\cos(\phi-\theta)$ is periodic with period 2π, so let $\theta = \frac{\pi}{2}$. Then by using the integral representation of Bessel functions we get

$$\int_0^{2\pi} e^{i(rt\cos(\phi-\theta))}\,d\phi = \int_0^{2\pi} e^{irt\sin(\phi)}\,d\phi = 2\pi J_0(rt).$$

Combining our results yields

$$\Psi(r) = 2\pi \int_0^b J_0(rt)\,t\,\hat{\Psi}(t)\,dt. \tag{2.3}$$

The trick is to pick $\hat{\Psi}(t)$, so that (2.3) can be evaluated explicitly. Here we need some properties of Bessel functions. The most fundamental one is

$$\frac{d(z^{-n}J_n(z))}{dz} = -z^{-n}J_{n+1}(z) \tag{2.4}$$

Replacing n by $-n$ in (2.4), and using that $J_{-n}(z) = (-1)^n J_n(z)$ we get

$$\frac{d(z^n J_n(z))}{dz} = z^n J_{n-1}(z). \tag{2.5}$$

It follows from (2.5) and the chain rule that

$$\frac{d((\sqrt{z})^n J_n(\sqrt{z}))}{dz} = \frac{(\sqrt{z})^n J_{n-1}(\sqrt{z})}{2\sqrt{z}} = \frac{1}{2}(\sqrt{z})^{n-1}J_{n-1}(\sqrt{z}) \tag{2.6}$$

We will use this result to integrate by parts in (2.3). First we need a change of variables.

Let $s=t^2r^2$. Then, $2tdt=r^{-2}ds$ and (2.3) becomes

$$\Psi(r) = \frac{\pi}{r^2}\int_0^{b^2r^2} J_0(\sqrt{s})\hat{\Psi}(\sqrt{s}/r)ds = \frac{1}{r^2}\int_0^{b^2r^2} J_0(\sqrt{s})g(s)ds, \tag{2.7}$$

where $g(s) = \pi\hat{\Psi}(\sqrt{s}/r)$.

Let g be a spline of order $n+1$. We can then integrate by parts in (2.7) repeatedly, and the final result will decay rapidly at infinity. More precisely, g should satisfy

(*i*) *g(s) is n times continuously differentiable for* $0<s<\infty$,

(*ii*) *The (n+1)-st derivative of g is piecewise constant,*

(*iii*) $g^{(k)}(s)\big|_{s=r^2} = g^{(k)}(s)\big|_{s=b^2r^2} = 0$ for $k = 1,\ldots,n$

Since $g\equiv(4\pi)^{-1}$ for $0<s<r^2$, we find after integrating by parts in (2.7)

$$\Psi(r)=\frac{(-2)^{n+1}}{r^2}\int_0^{b^2r^2}(\sqrt{s})^{n+1}J_{n+1}(\sqrt{s})g^{(n+1)}(s)\,ds=\frac{(-2)^{n+1}}{r^2}\int_{r^2}^{b^2r^2}(\sqrt{s})^{n+1}J_{n+1}(\sqrt{s})g^{(n+1)}(s)\,ds. \quad (2.8)$$

Note that the boundary terms vanish because of conditions (i)–(iii) above. The integral can be evaluated explicitly, since $g^{(n+1)}$ is piecewise constant. To define g more precisely, we let $b>1$ and pick n distinct points $x_1,\ldots,x_n$ in the open interval (r^2, b^2r^2). Then

$$g(s)=\frac{1}{4\pi} \qquad \text{for } 0\le s\le r^2$$

$$g(s)=\frac{1}{4\pi}+C_0(s/r^2-1)^{n+1} \qquad \text{for } r^2\le s\le x_1$$

$$\vdots$$

$$g(s)=\frac{1}{4\pi}+C_0(s/r^2-1)^{n+1}+\cdots+C_n(s/r^2-x_n/r^2)^{n+1} \qquad \text{for } x_n\le s\le b^2r^2$$

$$g(s)=0 \qquad \text{for } s>b^2r^2$$

where $C_0,\ldots,C_n$ are constants which we have to determine. We note that except for the point $s=b^2r^2$, g has n continuous derivatives regardless of the values of the constants $C_0,\ldots,C_n$. At $s=b^2r^2$ we must however satisfy the following $n+1$ conditions:

$$C_0(b^2-1)+\cdots+C_n(b^2-x_n/r^2)=0$$

$$\vdots$$

$$C_0(b^2-1)^n+\cdots+C_n(b^2-x_n/r^2)^n=0$$

$$C_0(b^2-1)^{n+1}+\cdots+C_n(b^2-x_n/r^2)^{n+1}=\frac{-1}{4\pi}$$

For clarity, we set $\Delta_0=b^2-1$, $\Delta_i=b^2-x_i/r^2$ for $i=1,\ldots,n$. Then, we can write the above continuity conditions in matrix form as follows:

$$\begin{bmatrix} \Delta_0 & \Delta_1 & \cdots & \Delta_n \\ \Delta_0^2 & \Delta_1^2 & \cdots & \Delta_n^2 \\ \vdots & \vdots & & \vdots \\ \Delta_0^{n+1} & \Delta_1^{n+1} & \cdots & \Delta_n^{n+1} \end{bmatrix}\begin{bmatrix} C_0 \\ \vdots \\ C_n \end{bmatrix}=\begin{bmatrix} 0 \\ \vdots \\ \frac{-1}{4\pi} \end{bmatrix} \quad (2.9)$$

This is a Vandermonde system, except for scaling of the columns. The matrix is non-singular, since the Δ's are distinct. The solution of the system is the last column of the inverse divided by -4π . In order to find this solution explicitly we need the following two lemmas.

LEMMA 1 *If B is the matrix defined by* $B_{ij} = \Delta_{i-1}^{j-1}$ for $i,j = 1,2,\ldots,n+1$ *,then*

$$B^{-1}_{(n+1)(k+1)} = \prod_{i=0,i\neq k}^{n} (\Delta_k - \Delta_i)^{-1} \text{ for } k = 0,1,\ldots,n.$$

Proof Consider the system $B\bar{a} = e_{k+1}$, where $\bar{a} = (a_0,\ldots,a_n)^T$ and e_{k+1} is the $(k+1)$-th column of the $(n+1)$x$(n+1)$ identity matrix. This is equivalent to $p(\Delta_k)=1$ and $p(\Delta_i)=0$ for $i=0,\ldots,k-1,k+1,\ldots,n$ where $p(x) = a_0 + a_1 x + \cdots + a_n x^n$. By using Lagrange's interpolation formula we find $p(x) = \prod_{i=0,i\neq k}^{n} (x-\Delta_i)/(\Delta_k - \Delta_i)$. Equating the coefficients of x^n in these two expressions we get $a_n = \prod_{i=0,i\neq k}^{n} (\Delta_k - \Delta_i)^{-1}$. But $a_n = B^{-1}_{(n+1)(k+1)}$.

This completes the proof of lemma 1.

LEMMA 2 *The solution of equation (2.9) is*

$$C_i = \left[-4\pi\Delta_i \prod_{j=0,j\neq i}^{n} (\Delta_i - \Delta_j) \right]^{-1}, \quad \text{for } i=0,\ldots,n.$$

Proof Let $\bar{b} = (B^{-1}_{(n+1)1},\ldots,B^{-1}_{(n+1)(n+1)})^T$ where B is defined as in lemma 1. Since $B^T B^{-T} = I$ and $\bar{b}$ is the last column of B^{-T} we have $B^T \bar{b} = e_{n+1}$, or componentwise

$$\sum_{j=0}^{n} \frac{\Delta_j^i}{\prod_{k=0,k\neq j}^{n} (\Delta_j - \Delta_k)} = \begin{cases} 0 & \text{for } i=0,\ldots,n-1 \\ 1 & \text{for } i=n \end{cases}$$

Hence,

$$\sum_{j=0}^{n} \frac{\Delta_j^{i+1}}{-4\pi\Delta_j \prod_{k=0,k\neq j}^{n} (\Delta_j - \Delta_k)} = \begin{cases} 0 & \text{for } i=0,\ldots,n-1 \\ \dfrac{-1}{4\pi} & \text{for } i=n \end{cases}$$

which can be written in matrix form as (2.9) with

$$C_i = \left[-4\pi\Delta_i \prod_{j=0, j\neq i}^{n} (\Delta_i - \Delta_j) \right]^{-1} \quad \text{for } i=0,\ldots,n.$$

This completes the proof.

We have now found the function g. To evaluate $\Psi(r)$ explicitly, we set $x_0 = r^2, x_{n+1} = b^2 r^2$ and rewrite (2.8) in the following form

$$\Psi(r) = \frac{(-2)^{n+1}}{r^2} \sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} (\sqrt{s})^{n+1} J_{n+1}(\sqrt{s})\, g^{(n+1)}(s)\, ds. \tag{2.10}$$

Here,

$$g^{(n+1)}(s) = \begin{cases} \dfrac{(n+1)! C_0}{r^{2n+2}} & \text{for } x_0 \le s \le x_1 \\ \cdot \\ \cdot \\ \cdot \\ \dfrac{(n+1)!}{r^{2n+2}} \displaystyle\sum_{i=0}^{j} C_i & \text{for } x_j \le s \le x_{j+1} \\ \cdot \\ \cdot \\ \cdot \\ \dfrac{(n+1)!}{r^{2n+2}} \displaystyle\sum_{i=0}^{n} C_i & \text{for } x_n \le s \le x_{n+1} \end{cases}$$

Using these known values of $g^{(n+1)}(s)$ and equation (2.6) we get

$$\int_{x_i}^{x_{i+1}} (\sqrt{s})^{n+1} J_{n+1}(\sqrt{s})\, g^{(n+1)}(s)\, ds = \frac{(n+1)!}{r^{2n+2}} \left[\sum_{k=0}^{i} C_k \right] \int_{x_i}^{x_{i+1}} (\sqrt{s})^{n+1} J_{n+1}(\sqrt{s})\, ds =$$

$$= \frac{2(n+1)!}{r^{2n+2}} \left[\sum_{k=0}^{i} C_k \right] \left[(\sqrt{x_{i+1}})^{n+2} J_{n+2}(\sqrt{x_{i+1}}) - (\sqrt{x_i})^{n+2} J_{n+2}(\sqrt{x_i}) \right], \quad \text{for } i = 0,1,\ldots,n.$$

Then,

$$\int_{0}^{x_{n+1}} (\sqrt{s})^{n+1} J_{n+1}(\sqrt{s})\, g^{(n+1)}(s)\, ds = \frac{2(n+1)!}{r^{2n+2}} \sum_{i=0}^{n} \left[(\sqrt{x_{i+1}})^{n+2} J_{n+2}(\sqrt{x_{i+1}}) \sum_{k=0}^{i} C_k - (\sqrt{x_i})^{n+2} J_{n+2}(\sqrt{x_i}) \sum_{k=0}^{i} C_k \right]$$

$$= \frac{2(n+1)!}{r^{2n+2}} \left[(\sqrt{x_{n+1}})^{n+2} J_{n+2}(\sqrt{x_{n+1}}) \sum_{k=0}^{n} C_k - \sum_{i=0}^{n} C_i (\sqrt{x_i})^{n+2} J_{n+2}(\sqrt{x_i}) \right]$$

And finally,

$$\Psi(r)=\frac{(-2)^{n+2}(n+1)!}{r^{2n+4}}\left[-(\sqrt{x_{n+1}})^{n+2}J_{n+2}(\sqrt{x_{n+1}})\sum_{k=0}^{n}C_k+\sum_{i=0}^{n}C_i(\sqrt{x_i})^{n+2}J_{n+2}(\sqrt{x_i})\right] \tag{2.11}$$

In order to get a convenient cutoff function, we choose $x_i=k_i^2r^2$ for $i=1,\ldots,n$ where $k_1,\ldots,k_n$ are positive integers. Set $k_0=1$ and $k_{n+1}=b$. Then,

$$\Psi(r)=\frac{(-2)^{n+2}(n+1)!}{r^{n+2}}\left[-b^{n+2}J_{n+2}(br)\sum_{k=0}^{n}C_k+\sum_{i=0}^{n}C_ik_i^{n+2}J_{n+2}(k_ir)\right] \tag{2.12}$$

To find $K_\delta(x,y)$ we let $\Psi_\delta(s)=\frac{1}{\delta^2}\Psi(\frac{s}{\delta})$ and get

$$\begin{aligned}K_\delta(x,y)&=\frac{(-y,x)^T}{r^2}\int_0^r s\,\Psi_\delta(s)\,ds\\&=\frac{(-y,x)^T}{r^2}\int_0^{r/\delta}u\,\Psi(u)\,du.\end{aligned} \tag{2.13}$$

Fortunately (2.13) can be evaluated explicitly when Ψ has the form (2.11). The only sticky part is to evaluate integrals of the form $\int_0^{r/\delta}u^{-(n+1)}J_{n+2}(k_iu)\,du$. Using (2.4) and the chain rule we get

$$\frac{d(u^{-(n+1)}J_{n+1}(k_iu))}{du}=\frac{-k_iJ_{n+2}(k_iu)}{u^{n+1}}. \tag{2.14}$$

Letting r tend to infinity in condition (iii) we see that $\int_0^\infty u\,\Psi(u)\,du=(2\pi)^{-1}$. We can therefore write $\int_0^{r/\delta}u\,\Psi(u)\,du=(2\pi)^{-1}-\int_{r/\delta}^\infty u\,\Psi(u)\,du$. Thus it follows from (2.14) that

$$\int_{r/\delta}^\infty u^{-(n+1)}J_{n+2}(k_iu)\,du=-\frac{1}{k_i}\left[\lim_{t\to\infty}\frac{J_{n+1}(k_it)}{t^{n+1}}-\frac{J_{n+1}(k_ir/\delta)}{(r/\delta)^{n+1}}\right]=\frac{J_{n+1}(k_ir/\delta)}{k_i(r/\delta)^{n+1}},$$

since the limit as t goes to infinity is 0. Combining (2.13) and (2.11) we now get after some calculations

$$K_\delta(x,y)=\frac{(-y,x)^T}{r^2}\left[\frac{1}{2\pi}+\frac{(-2)^{n+2}(n+1)!}{(r/\delta)^{n+1}}\left[b^{n+1}J_{n+1}(br/\delta)\sum_{k=0}^{n}C_k-\sum_{i=0}^{n}C_ik_i^{n+1}J_{n+1}(k_ir/\delta)\right]\right] \tag{2.15}$$

To simplify the notation let

$$\gamma_i = -2\pi(-2)^{n+2}(n+1)!C_i k_i^{n+1} = -\frac{(-2k_i)^{n+1}(n+1)!}{(b^2-k_i^2)\prod_{j=0,j\neq i}^{n}(k_j^2-k_i^2)} \quad \text{for } i=0,\ldots,n$$

and

$$\gamma_{n+1} = 2\pi(-2)^{n+2}(n+1)!b^{n+1}\sum_{k=0}^{n}C_k = (-2b)^{n+1}(n+1)!\sum_{i=0}^{n}\frac{1}{(b^2-k_i^2)\prod_{j=0,j\neq i}^{n}(k_j^2-k_i^2)}$$

We can then rewrite (2.15) in a more compact form:

$$K_\delta(x,y) = \frac{(-y,x)^T}{2\pi r^2}\left[1+\left[\frac{r}{\delta}\right]^{-(n+1)}\left[\sum_{j=0}^{n+1}\gamma_j J_{n+1}(k_j r/\delta)\right]\right] \tag{2.16}$$

We will now present Hald's convergence theorem for infinite order cutoffs. First, we need to introduce the norms and seminorms

$$||\omega||_{C^{m+\lambda}(D)} = \sum_{\nu=0}^{m} D^\nu \max_{|\gamma|=\nu}||\partial^\gamma\omega|| + D^{m+\lambda}\max_{|\gamma|=m}H_\lambda(\partial^\gamma\omega)$$

$$|x|_{C^{m+1+\lambda}(D)} = \sum_{\nu=0}^{m+1} D^\nu \max_{|\gamma|=\nu}||\partial^\gamma x|| + D^{m+1+\lambda}\max_{|\gamma|=m+1}H_\lambda(\partial^\gamma x)$$

Here $\partial^\gamma = \partial_1^{\gamma_1}\partial_2^{\gamma_2}$ and $H_\lambda(f) = \sup_{x\neq y}|f(x)-f(y)|/|x-y|^\lambda$.

THEOREM (Hald [16]). *Let D be larger than the diameter of the support of ω and assume that* $||\omega||_{C^{m+\lambda}(D)}\leq C$ *and* $|\partial_t^\nu x(t)|_{C^{m+1+\lambda-\nu}(D)}\leq\frac{1}{2}C$ *for* $0<\lambda<1, \nu=0,1,\ldots,m+1, m\geq 6$ *and* $0\leq t\leq T$. *Let our assumptions on $\hat{\Psi}$ hold and set $\delta = \text{constant}\cdot h^q$ with $q=\frac{1}{2}(m+\lambda)/(m+\lambda-1)$ and $c_j = \omega(jh)h^2$. Solve the differential equation (1.10),(1.11) by the classical Runge-Kutta method with $\Delta t\leq h^{(1+\gamma)/4}, \gamma>0$. Let $1\leq p<\infty$. Then there exist two constants C_0 and h_0 such that*

$$||\tilde{x}(t)-x(t)||_p \leq C_0(h^{(m+\lambda)/2}+(\Delta t)^4)$$

for all $h\leq h_0$ and $t\leq T$.

We shall conclude this chapter by presenting three examples of cutoff functions out of the general class given by (2.12) and (2.16).

Example 1 Let $n=1$, $k_1=2$ and $b=4$. Then

$\Delta_0=15$, $\Delta_1=12$,

$$\gamma_0=-\frac{4\cdot 2}{(16-1)(4-1)}=-\frac{8}{45},\ \gamma_1=-\frac{16\cdot 2}{(16-4)(1-4)}=\frac{32}{36},\ \gamma_2=64\cdot 2\left[\frac{1}{45}-\frac{1}{36}\right]=-\frac{32}{45},$$

$$C_0=\frac{-1}{4\pi\Delta_0(\Delta_0-\Delta_1)}=\frac{-1}{180\pi}\text{ and }C_1=\frac{-1}{4\pi\Delta_1(\Delta_1-\Delta_0)}=\frac{1}{144\pi}.$$

Plugging in these constants into (2.12) and (2.16) we get

$$\Psi(r)=\frac{4}{45\pi r^3}\left[16J_3(4r)-10J_3(2r)+J_3(r)\right]$$

and

$$K_\delta(x,y)=\frac{(-y,x)^T}{2\pi r^2}\left[1-\frac{8}{45(r/\delta)^2}\left[4J_2(4r/\delta)-5J_2(2r/\delta)+J_2(r/\delta)\right]\right]$$

This cutoff function was introduced by Hald [16], and has been tried out on all the test problems in this paper. The optimal value of δ for this cutoff function seems to vary somewhat with the initial vorticity distribution, but typically it lies between $0.3\sqrt{h}$ and $0.4\sqrt{h}$. However, in a case with long time integrations we needed to take a larger value of δ to retain the high rate of convergence.

Example 2 Let $n=2$, $k_1=2$, $k_2=3$ and $b=4$. Then

$\Delta_0=15$, $\Delta_1=12$, $\Delta_2=7$,

$$\gamma_0=-\frac{(-8)\cdot 3!}{15\cdot 3\cdot 8}=\frac{48}{360},\ \gamma_1=-\frac{(-64)\cdot 3!}{12\cdot(-3)\cdot 5}=-\frac{384}{180},$$

$$\gamma_2=-\frac{(-216)\cdot 3!}{7\cdot(-8)\cdot(-5)}=\frac{1296}{280},\ \gamma_3=(-512)\cdot 3!\left[\frac{1}{360}-\frac{1}{180}+\frac{1}{280}\right]=-\frac{256}{105},$$

$$C_0=\frac{-1}{4\pi\Delta_0(\Delta_0-\Delta_1)(\Delta_0-\Delta_2)}=-\frac{1}{1440\pi},\ C_1=\frac{-1}{4\pi\Delta_1(\Delta_1-\Delta_0)(\Delta_1-\Delta_2)}=\frac{1}{720\pi},$$

$$C_2=\frac{-1}{4\pi\Delta_2(\Delta_2-\Delta_0)(\Delta_2-\Delta_1)}=-\frac{1}{1120\pi}.$$

Plugging in these constants into (2.12) and (2.16) we get

$$\Psi(r)=\frac{1}{105\pi r^4}\left[512J_4(4r)-729J_4(3r)+224J_4(2r)-7J_4(r)\right]$$

and

$$K_\delta(x,y)=\frac{(-y,x)^T}{2\pi r^2}\left[1-\frac{2}{105(r/\delta)^3}\left[128J_3(4r/\delta)-243J_3(3r/\delta)+112J_3(2r/\delta)-7J_3(r/\delta)\right]\right]$$

Example 3 Let $n=3$, $k_1=2$, $k_2=3$, $k_3=4$ and $b=5$. Then

$\Delta_0=24$, $\Delta_1=21$, $\Delta_2=16$, $\Delta_3=9$,

$$\gamma_0=-\frac{16\cdot 4!}{24\cdot 15\cdot 8\cdot 3}=-\frac{42}{945},\ \gamma_1=-\frac{256\cdot 4!}{21\cdot 12\cdot 5\cdot(-3)}=\frac{1536}{945},$$

$$\gamma_2=-\frac{1296\cdot 4!}{16\cdot 7\cdot(-5)\cdot(-8)}=-\frac{6561}{945},\ \gamma_3=-\frac{4096\cdot 4!}{9\cdot(-7)\cdot(-12)\cdot(-15)}=\frac{8192}{945},$$

$$\gamma_4=10000\cdot 4!\left[\frac{1}{8640}-\frac{1}{3780}+\frac{1}{4480}-\frac{1}{11340}\right]=-\frac{3125}{945},$$

$$C_0=\frac{-1}{4\pi\Delta_0(\Delta_0-\Delta_1)(\Delta_0-\Delta_2)(\Delta_0-\Delta_3)}=-\frac{1}{34560\pi},\ C_1=\frac{-1}{4\pi\Delta_1(\Delta_1-\Delta_0)(\Delta_1-\Delta_2)(\Delta_1-\Delta_3)}=\frac{1}{15120\pi},$$

$$C_2=\frac{-1}{4\pi\Delta_2(\Delta_2-\Delta_0)(\Delta_2-\Delta_1)(\Delta_2-\Delta_3)}=-\frac{1}{17920\pi},\ C_3=\frac{-1}{4\pi\Delta_3(\Delta_3-\Delta_0)(\Delta_3-\Delta_1)(\Delta_3-\Delta_2)}=\frac{1}{45360\pi}.$$

Again, plugging in these constants into (2.12) and (2.16) we get

$$\Psi(r)=\frac{1}{1890\pi r^5}\left[15625J_5(5r)-32768J_5(4r)+19683J_5(3r)-3072J_5(2r)+42J_5(r)\right]$$

$$K_\delta(x,y)=\frac{(-y,x)^T}{2\pi r^2}\left[1-\frac{1}{945(r/\delta)^4}\left[3125J_4(5r/\delta)-8192J_4(4r/\delta)+6561J_4(3r/\delta)-1536J_4(2r/\delta)+42J_4(r/\delta)\right]\right]$$

A limited number of numerical tests, were carried out using this cutoff function, and the results indicated the same rate of convergence as that obtained with Hald's infinite order cutoff function.

Figures (2.1)–(2.3) show the graphs of the cutoff functions in these three examples.

Example 1

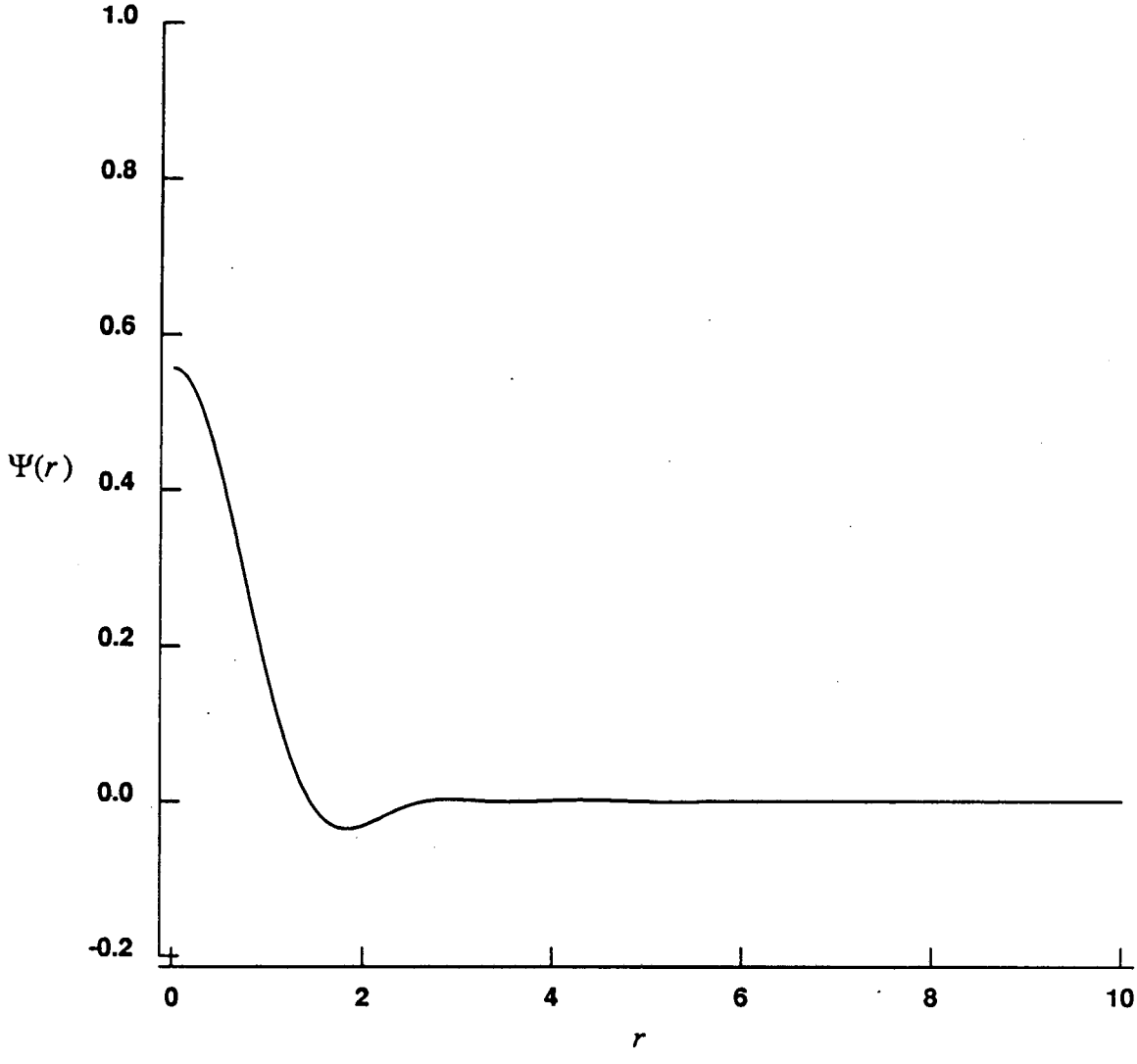


Fig. 2.1. $\Psi(r) = \frac{4}{45\pi r^3}\left[16J_3(4r) - 10J_3(2r) + J_3(r)\right]$

Example 2

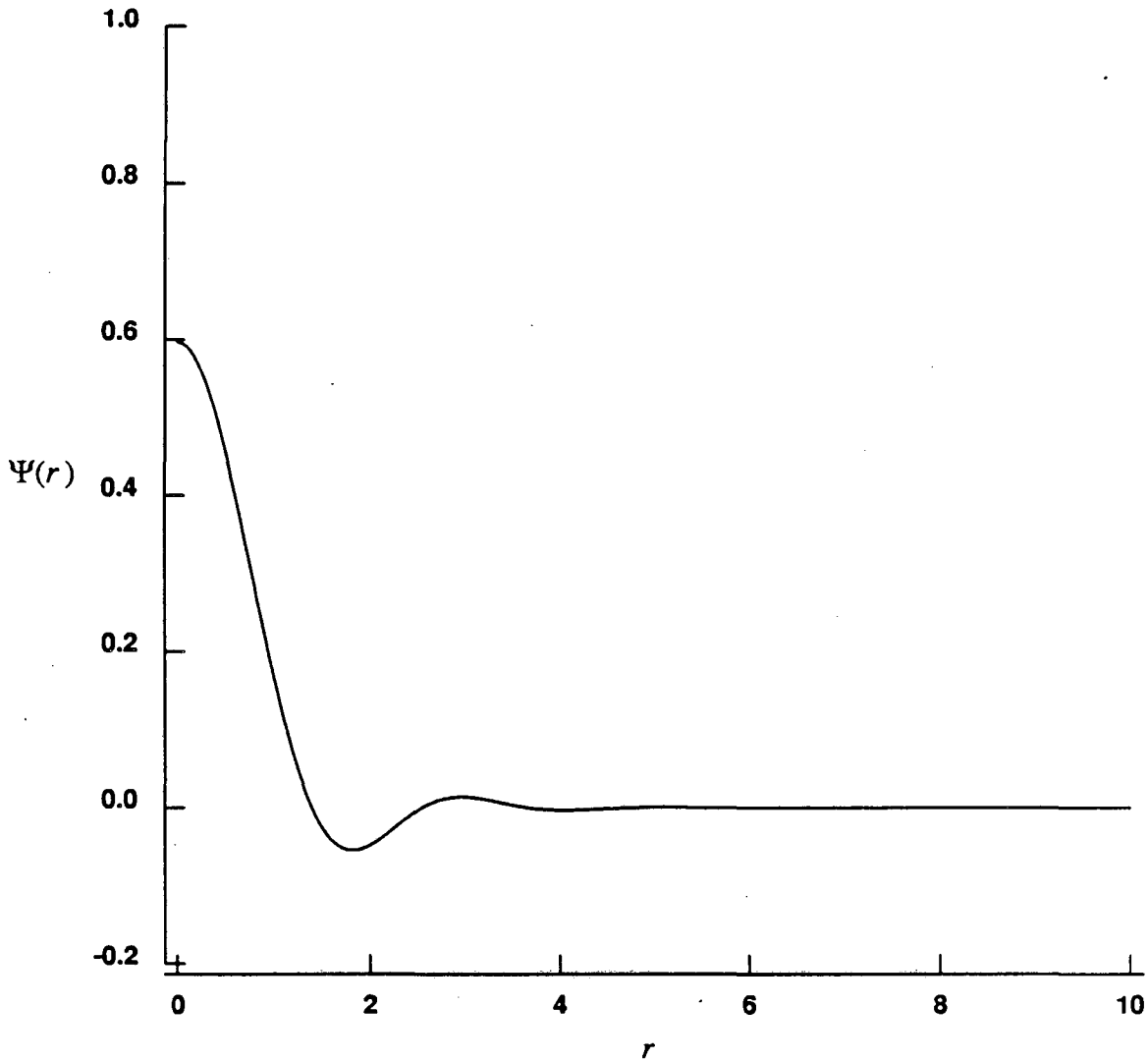


Fig. 2.2. $\Psi(r) = \frac{1}{105\pi r^4}\left[512J_4(4r) - 729J_4(3r) + 224J_4(2r) - 7J_4(r)\right]$

Example 3

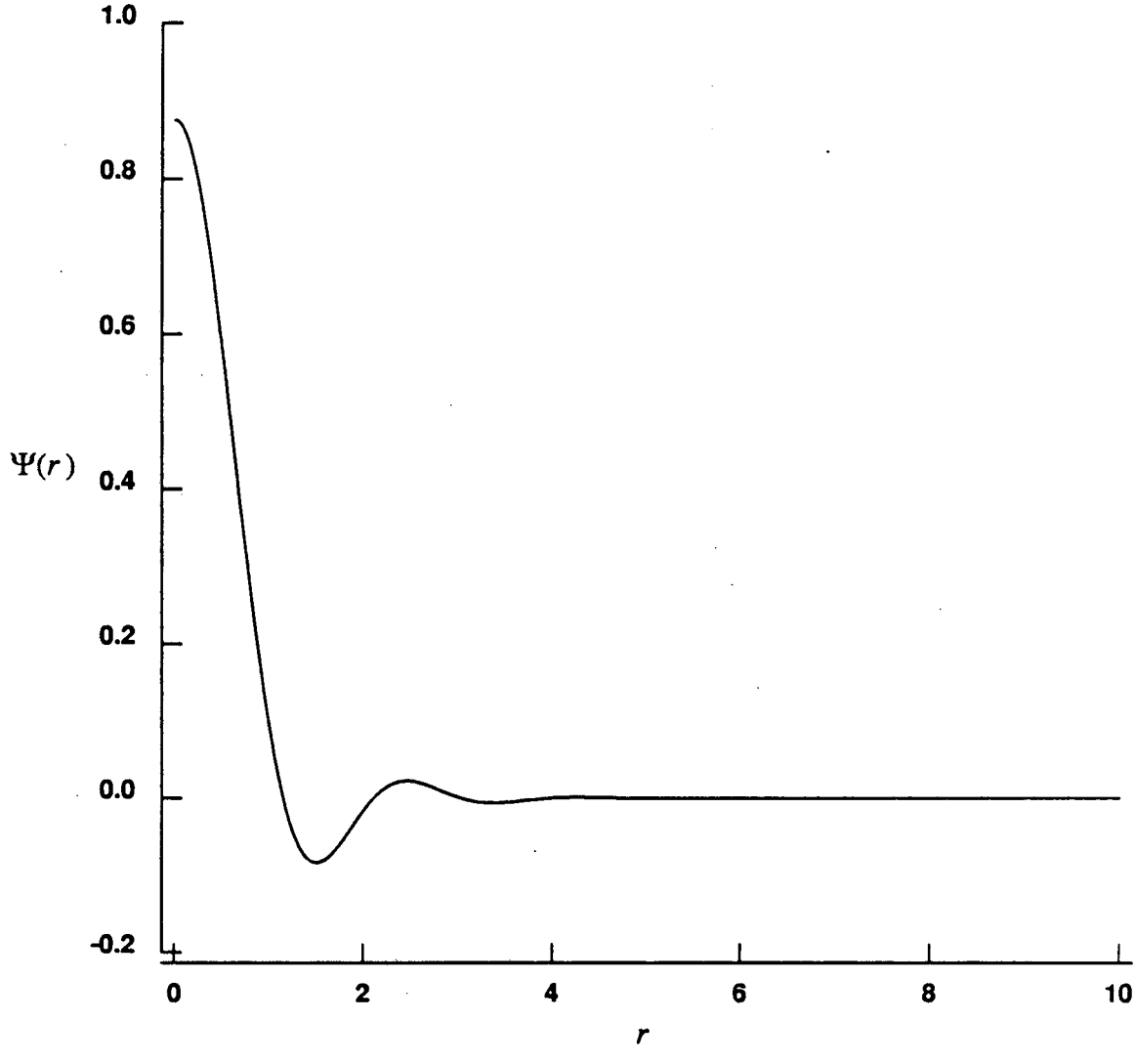


Fig. 2.3. $\Psi(r) = \frac{1}{1890\pi r^5}\left[15625J_5(5r) - 32768J_5(4r) + 19683J_5(3r) - 3072J_5(2r) + 42J_5(r)\right]$

3. Numerical Evaluation of Infinite Order Cutoff Functions.

Since the cutoff functions derived in the previous chapter contain several terms involving Bessel functions, it is computationally expensive to evaluate these terms "individually", using subroutines for Bessel functions. Instead, we approximate the whole cutoff function by local polynomials. These polynomials can be computed at the beginning of the program using for example the IMSL subroutine "IRATCU" [18]. The computational labor involved in finding the appropriate polynomials is usually negligible compared to the overall computations in a typical vortex computation, even when the integration time is short. Nevertheless, if many runs are to be made, it is better to store the coefficents of the polynomials in a data file. In the numerical experiments of this paper we used polynomials of degree ≤ 9 to evaluate $K_\delta(x,y)$ with a maximum error of 10^{-10} and polynomials of degree ≤ 12 to evaluate $\Psi(r)$ with a maximum error of 10^{-12}. The details of the procedure are as follows.

STEP 1 The infinite-order K_δ's derived in the previous chapter have the general form $K_\delta(x,y) = (-y,x)^T F(r^2/\delta^2)/\delta^2$.

Estimate the maximum value M of r/δ that is likely to be encountered during the course of computations. In the numerical experiments presented in this paper we have used $M=120$.

STEP 2 For each positive integer $j \leq M$ find the best polynomial approximation $P_j(r^2/\delta^2 - j^2)$ of the function $F(r^2/\delta^2)$ in the interval $j-0.5 \leq r/\delta \leq j+0.5$.

STEP 3 Every time $K_\delta(x,y)$ has to be evaluated, we first compute r^2/δ^2. We then compute the square-root of this value, rounded to the nearest integer k. Finally, we evaluate $K_\delta(x,y) \approx (-y,x)^T P_k(r^2/\delta^2 - k^2)/\delta^2$. If $k>M$, we can use a short asymptotic expansion to approximate F rather than a polynomial, and when $k=0$, i.e. $r^2/\delta^2<0.25$ we use a truncated MacLaurin expansion of F.

In the same manner, we find the collection of polynomials $Q_j(r)$ to approximate $\Psi(r)$. Then $\Psi_\delta(r)$ is approximated by $Q_k(r^2/\delta^2 - k^2)/\delta^2$ where k is the integer closest to r/δ.

The error bounds in the polynomial approximations are provided by the subroutine "IRATCU". In this case it turns out that if we use polynomials of constant degree, the error gets smaller as r/δ gets larger, and conversely, if we specify an error bound of say 10^{-10}, we may use polynomials of lower degree for larger arguments. Finally we should point out that if we make the intervals shorter we may be able to use polynomials of lower degree, but it has been our experience that in order to reduce the degree of the polynomials significantly, say by a factor of two, without increasing the error we must make the intervals *much* shorter which does not seem practical.

To give some indication of typical cases, we will list the coefficients of the polynomials which approximate F and Ψ in three different intervals in the case when Ψ is Hald's infinite order cutoff, i.e. example 1 of the previous chapter. We should point out that P_0 and Q_0 are the truncated MacLaurin series of Ψ and F, *not* the best polynomial approximations of these functions for $r/\delta \le 0.5$. However, for $k \ge 1$, P_k and Q_k are the best polynomial approximations of Ψ and F.

For $r/\delta \le 0.5$ we use

$$P_0(x) = \sum_{k=0}^{9} c_{(0,k+1)} x^k, \qquad Q_0(x) = \sum_{k=0}^{11} d_{(0,k+1)} x^k,$$

$c_{(0,1)} = 0.278521150410817$
$c_{(0,2)} = -0.147964361155746$
$c_{(0,3)} = 0.400443231381822 \cdot 10^{-1}$
$c_{(0,4)} = -0.669859081907091 \cdot 10^{-2}$
$c_{(0,5)} = 0.766254806513328 \cdot 10^{-3}$
$c_{(0,6)} = -0.638691859290432 \cdot 10^{-4}$
$c_{(0,7)} = 0.405541846198420 \cdot 10^{-5}$
$c_{(0,8)} = -0.202773823804794 \cdot 10^{-6}$
$c_{(0,9)} = 0.819291107003087 \cdot 10^{-8}$
$c_{(0,10)} = -0.273097279835706 \cdot 10^{-9}$

$d_{(0,1)} = 0.557042300821634$
$d_{(0,2)} = -0.591857444622986$
$d_{(0,3)} = 0.240265938829093$
$d_{(0,4)} = -0.535887265525673 \cdot 10^{-1}$
$d_{(0,5)} = 0.766254806513328 \cdot 10^{-2}$
$d_{(0,6)} = -0.766430231148519 \cdot 10^{-3}$
$d_{(0,7)} = 0.567758584677788 \cdot 10^{-4}$
$d_{(0,8)} = -0.324438118087670 \cdot 10^{-5}$
$d_{(0,9)} = 0.147472399260556 \cdot 10^{-6}$
$d_{(0,10)} = -0.546194559671411 \cdot 10^{-8}$
$d_{(0,11)} = 0.168059902078596 \cdot 10^{-9}$
$d_{(0,12)} = -0.436519250570586 \cdot 10^{-11}$

For $49.5 \le r/\delta \le 50.5$ we use

$$P_{50}(x) = \sum_{k=0}^{4} c_{(50,k+1)} x^k, \qquad Q_{50}(x) = \sum_{k=0}^{9} d_{(50,k+1)} x^k,$$

$$c_{(50,1)} = 0.636614909788761 \cdot 10^{-4}$$
$$c_{(50,2)} = -0.254567912072669 \cdot 10^{-7}$$
$$c_{(50,3)} = 0.104739981041642 \cdot 10^{-10}$$
$$c_{(50,4)} = -0.103221399644189 \cdot 10^{-13}$$
$$c_{(50,5)} = -0.181116361842747 \cdot 10^{-16}$$

$$d_{(50,1)} = 0.460700238644189 \cdot 10^{-7}$$
$$d_{(50,2)} = 0.294187407199615 \cdot 10^{-8}$$
$$d_{(50,3)} = -0.127067586043915 \cdot 10^{-9}$$
$$d_{(50,4)} = -0.503513664434355 \cdot 10^{-12}$$
$$d_{(50,5)} = 0.204816631431159 \cdot 10^{-13}$$
$$d_{(50,6)} = 0.229931884309735 \cdot 10^{-16}$$
$$d_{(50,7)} = -0.113942361053185 \cdot 10^{-17}$$
$$d_{(50,8)} = -0.220120185455031 \cdot 10^{-21}$$
$$d_{(50,9)} = 0.296391416039998 \cdot 10^{-22}$$
$$d_{(50,10)} = -0.886890490271324 \cdot 10^{-26}$$

For $99.5 \le r/\delta \le 100.5$ we use

$$P_{100}(x) = \sum_{k=0}^{2} c_{(100,k+1)} x^k, \qquad Q_{100}(x) = \sum_{k=0}^{7} d_{(100,k+1)} x^k,$$

$$c_{(100,1)} = 0.159154820701304 \cdot 10^{-4}$$
$$c_{(100,2)} = -0.159211079192825 \cdot 10^{-8}$$
$$c_{(100,3)} = 0.163998371754908 \cdot 10^{-12}$$

$$d_{(100,1)} = -0.894075393611259 \cdot 10^{-8}$$
$$d_{(100,2)} = 0.308592662541504 \cdot 10^{-9}$$
$$d_{(100,3)} = -0.185399763454803 \cdot 10^{-12}$$
$$d_{(100,4)} = -0.226344722920123 \cdot 10^{-13}$$
$$d_{(100,5)} = 0.278116058868214 \cdot 10^{-16}$$
$$d_{(100,6)} = 0.452272542132804 \cdot 10^{-18}$$
$$d_{(100,7)} = -0.424452772521097 \cdot 10^{-21}$$
$$d_{(100,8)} = -0.374858233204224 \cdot 10^{-23}$$

In the appendix we give a fortran program which generates the coefficients of the polynomials which approximate F and Ψ in every interval up to $119.5 \le r/\delta \le 120.5$.

4. Derivation of an eighth order cutoff function with compact support.

Although the infinite order cutoff functions derived in chapter 2 give the best accuracy for smooth flows, they suffer from the disadvantage of not being compatible with any of the known "fast", i.e. $O(N)$, vortex methods such as the Rokhlin-Greengard algorithm [23] or Anderson's method of local corrections [1]. The infinite order cutoff functions may still be preferable in such cases where a small enough error can be achieved with a relatively small number of vortices. This is the case in the test problems presented at the end of this paper.

For the cases in which a large number of vortices is necessary, but in which the flow is still quite smooth, e.g. the support of the vorticity may be very large, we propose an eighth order cutoff function which is derived in this chapter. Since it has compact support, it can be implemented in combination with "fast" vortex methods. We must however bear in mind that the speedup in using a fast algorithm is limited by the size of the cutoff parameter δ , which for high order cutoff functions must be proportional to $\sqrt{h}$ in order to maintain high accuracy for long time integrations. In this case, the amount of computational labor due to "local" interactions is $O(N^{1.5})$.

We shall look for a cutoff function $\Psi(r)$, where $r=\sqrt{x^2+y^2}$, satisfying the following conditions:

(i) $\displaystyle\int_{R^2}\Psi(r)\,dxdy = 1$

(ii) $\displaystyle\int x^n y^m \Psi(r)\,dxdy = 0,$ for $1 \le n+m \le 7$, where n and m are non-negative integers.

(iii) $\Psi(r) = 0$ for $r \ge 1$ and $\Psi^{(k)}(1) = 0$ for $k=1,\ldots,8.$

Switching to polar coordinates, (ii) becomes:

$$\int_0^{2\pi}\int_0^{\infty}\cos^n\theta\sin^m\theta\, r^{n+m}\Psi(r)r\,drd\theta = \int_0^{2\pi}\cos^n\theta\sin^m\theta\left[\int_0^{\infty} r^{n+m}\Psi(r)r\,dr\right]d\theta$$

$$= \int_0^{2\pi}\cos^n\theta\sin^m\theta\,d\theta\int_0^{\infty} r^{n+m+1}\Psi(r)\,dr.$$

But,

$$\int_0^{2\pi} \cos^n\theta \sin^m\theta \, d\theta = \int_0^{\pi} \cos^n\theta \sin^m\theta \, d\theta + \int_{\pi}^{2\pi} \cos^n\theta \sin^m\theta \, d\theta$$

$$= (1 + (-1)^{n+m}) \int_0^{\pi} \cos^n\theta \sin^m\theta \, d\theta$$

$$= 0, \text{ when } n+m \text{ is odd.}$$

Therefore, condition (*ii*) reduces to $\int_0^{\infty} r^k \Psi(r)\, dr = 0$ for k=3,5 and 7.

Now we let $\Psi(r) = \begin{cases} a(1-r^2)^9(1+br^2+cr^4+dr^6) & \text{for } 0 \le r \le 1 \\ 0 & \text{for } r \ge 1 \end{cases}$

and solve the following linear system for a, b, c and d :

$$\int_0^1 r^3(1-r^2)^9(1+br^2+cr^4+dr^6)\, dr = 0$$

$$\int_0^1 r^5(1-r^2)^9(1+br^2+cr^4+dr^6)\, dr = 0$$

$$\int_0^1 r^7(1-r^2)^9(1+br^2+cr^4+dr^6)\, dr = 0$$

$$\int_0^1 ar(1-r^2)^9(1+br^2+cr^4+dr^6)\, dr = (2\pi)^{-1}$$

The solution is $a = 52/\pi, b = -21, c = 105$ and $d = -140$.

Hence,

$$\Psi(r) = \begin{cases} -52(1-r^2)^9(140r^6 - 105r^4 + 21r^2 - 1)/\pi & \text{for } 0 \le r \le 1 \\ 0 & \text{for } r \ge 1 \end{cases}$$

The corresponding K_δ is:

$$K_\delta(x,y) = \begin{cases} \dfrac{(-y,x)^T}{2\pi r^2}\left[1 + \left[1 - \dfrac{r^2}{\delta^2}\right]^{10}\left[286 - 1092\left[1 - \dfrac{r^2}{\delta^2}\right] + 1365\left[1 - \dfrac{r^2}{\delta^2}\right]^2 - 560\left[1 - \dfrac{r^2}{\delta^2}\right]^3\right]\right] & \text{for } r \le \delta \\ \dfrac{(-y,x)^T}{2\pi r^2} & \text{for } r > \delta \end{cases}$$

This cutoff function was tried out on test problems 1-3, and the results are compared to those obtained using Hald's infinite order cutoff function, at the end of this paper. We found that the optimal value of δ for this cutoff function is about $1.7\sqrt{h}$ for test problems 1-3.

The Fourier transform of Ψ is given by

$$\hat{\Psi}(t) = \frac{-6656 \cdot 11!}{\pi^2}\left[\frac{J_{10}(t)}{t^{10}} - \frac{84J_{11}(t)}{t^{11}} + \frac{2520J_{12}(t)}{t^{12}} - \frac{26880J_{13}(t)}{t^{13}}\right]$$

$\hat{\Psi}(t)$ is bounded for all t and $\hat{\Psi}(t)$ is of order $O(t^{-10.5})$ as $t \to \infty$. Hence $\hat{\Psi}$ satisfies the following condition with $L=10.5$.

(*iv*) *For some* $L>0$, *and for any double index* α

$$\sup_{\mathbf{k} \in R^2} |D_{\mathbf{k}}^{\alpha} \hat{\Psi}(\mathbf{k})| \leq C_{\alpha}(1+|\mathbf{k}|)^{-L-|\alpha|}$$

We shall now present a special case of a convergence theorem for vortex methods due to Beale and Majda [7], which is applicable to the eighth order cutoff function derived here.

THEOREM (Beale and Majda [7]). *Assume that the cutoff function* Ψ *satisfies* $\Psi \in C^2(R^2)$ *and conditions* (*i*),(*ii*) *and* (*iv*) *for some* $2 \leq L < \infty$. *Choose* $\delta = constant \cdot h^q$, *with* $q < (L-1)/(L+8)$. *If the velocity field* $\mathbf{u}(\mathbf{x},t)$ *is sufficiently smooth for* $\mathbf{x} \in R^2$ *and* $0 \leq t \leq T$ *and the initial vorticity distribution has compact support, then for any* $1 < \mu < \infty$ *and* $T>0$ *there exists a constant* $h_0>0$ *such that for all* $h<h_0$

$$\max_{0 \leq t \leq T} || \mathbf{x}_j(t) - \tilde{\mathbf{x}}_j(t) ||_{L^{\mu}} \leq Ch^{8q},$$

$$\max_{0 \leq t \leq T} || \mathbf{u}_j(t) - \tilde{\mathbf{u}}_j(t) ||_{L^{\mu}} \leq Ch^{8q}.$$

Since $L=10.5$ for our cutoff function, we can take $q=0.5 < 9.5/18.5$, which would give us fourth order convergence if the flow is smooth enough. Fig. (4.1) shows the graph of the eighth order cutoff function Ψ. We note that the shape of the graph is similar to the shape of the graph of Hald's infinite order cutoff function $\tilde{\Psi}$ (Fig. 2.1), but the scaling is entirely different. In particular, $\Psi(0)=52/\pi$ while $\tilde{\Psi}(0)=1.75/\pi$. Therefore, rather than comparing Ψ and $\tilde{\Psi}$, we compare Ψ_{α} and $\tilde{\Psi}$, where $\alpha=\sqrt{52/1.75}$ and $\Psi_{\alpha}(r) = \alpha^{-2}\Psi(r/\alpha)$. Then, $\Psi_{\alpha}(0)=\tilde{\Psi}(0)$, and interestingly enough we see by plotting $\Psi_{\alpha}(r)$ and $\tilde{\Psi}(r)$ on the same graph, that $\Psi_{\alpha}(r)=\tilde{\Psi}(r)$ for *any* r. See fig. 4.2. It is also interesting to

compare the Fourier transforms of these two cutoff functions. Since $\hat{\Psi}_\alpha(t)=\hat{\Psi}(\alpha t)$, we plot $\hat{\Psi}(\alpha t)$ and $\hat{\hat{\Psi}}(t)$ on the same graph. See fig. 4.3. Once again, we get close agreement. We conclude that if we use $\delta=Ch^q$ with the eighth order cutoff function Ψ, and $\delta'=C'h^q$ with Hald's infinite order cutoff, we should have $C/C'=\sqrt{(52/1.75)}\approx 5.45$. Indeed, in test problems 1-3 we found by experiments that $\delta'=0.3\sqrt{h}$ was the best choice for Hald's infinite order cutoff function while the best value of δ for the eighth order cutoff function was about $1.7\sqrt{h}$. Note that $1.7/0.3\approx 5.67$! This analysis suggests that if we have found the best value of δ as a function of h experimentally for a particular cutoff function Ψ_1, then we can determine the best value of δ as a function of h for any other cutoff function Ψ, provided both cutoff functions are bounded and positive at 0. Take

$$\delta_{optimal} = (\delta_1)_{optimal}\left[\frac{\Psi(0)}{\Psi_1(0)}\right]^{1/2}$$

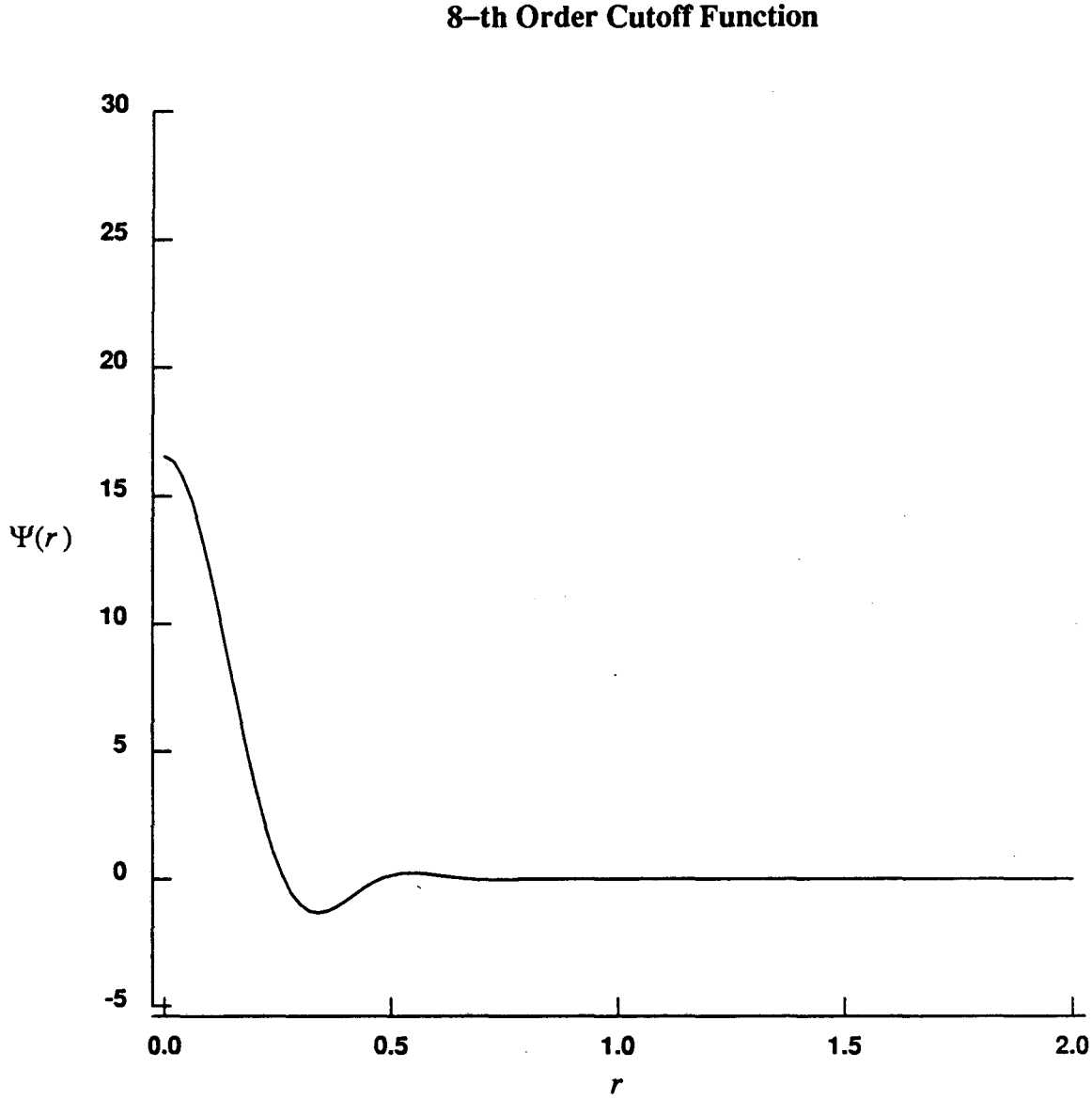


Fig. 4.1. $\Psi(r) = \begin{cases} -52(1-r^2)^9(140r^6 - 105r^4 + 21r^2 - 1)/\pi & \text{for } 0 \le r \le 1 \\ 0 & \text{for } r \ge 1 \end{cases}$

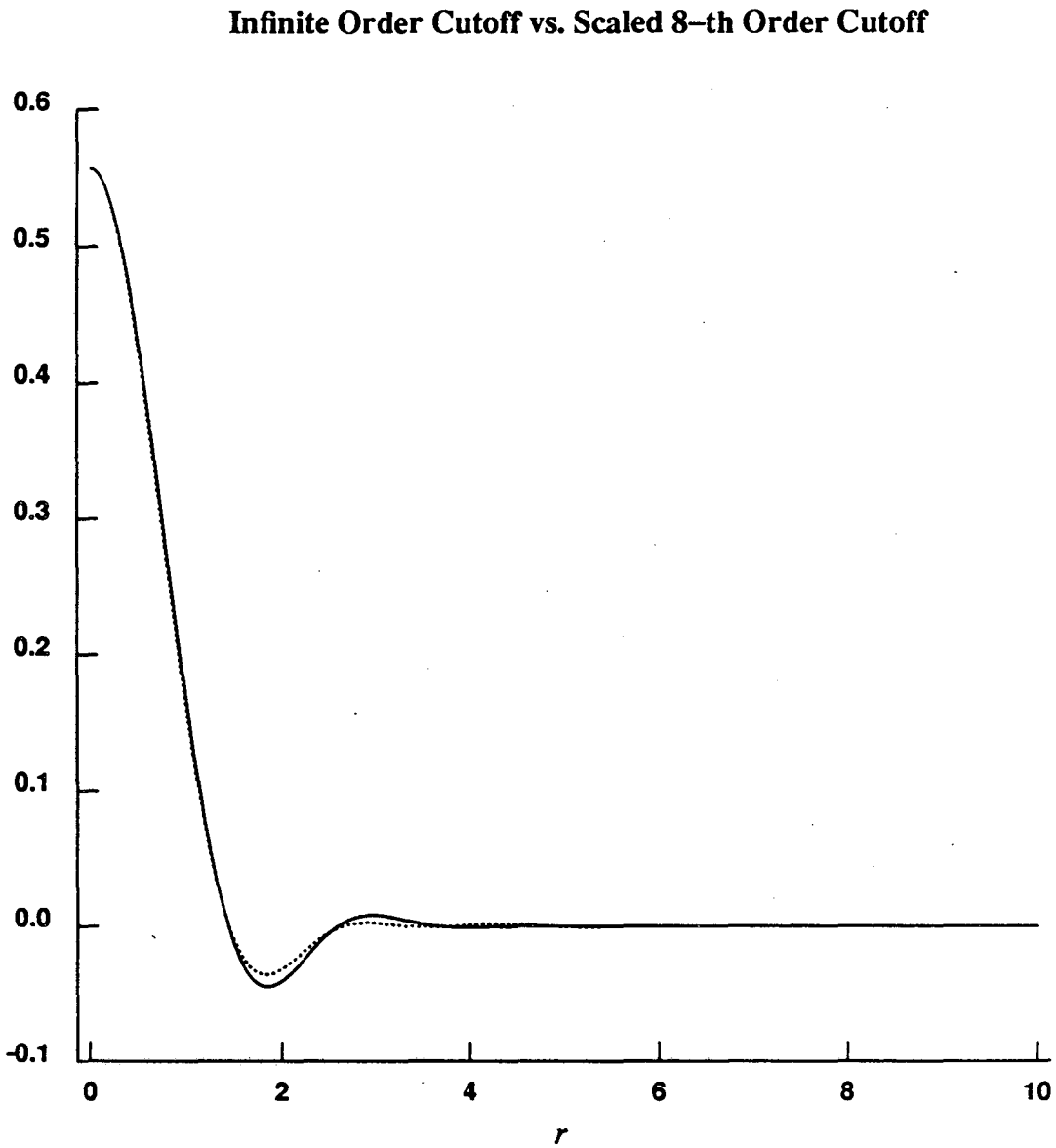


Fig. 4.2. Solid curve = $\Psi_\alpha(r)$, dotted curve = $\tilde{\Psi}(r)$, $\alpha=\sqrt{(52/1.75)}$.

$$\tilde{\Psi}(r) = \frac{4}{45\pi r^3}\Big[16J_3(4r) - 10J_3(2r) + J_3(r)\Big]$$

Fourier Transforms of Hald's Infinite Order Cutoff

and the Scaled 8–th Order Cutoff

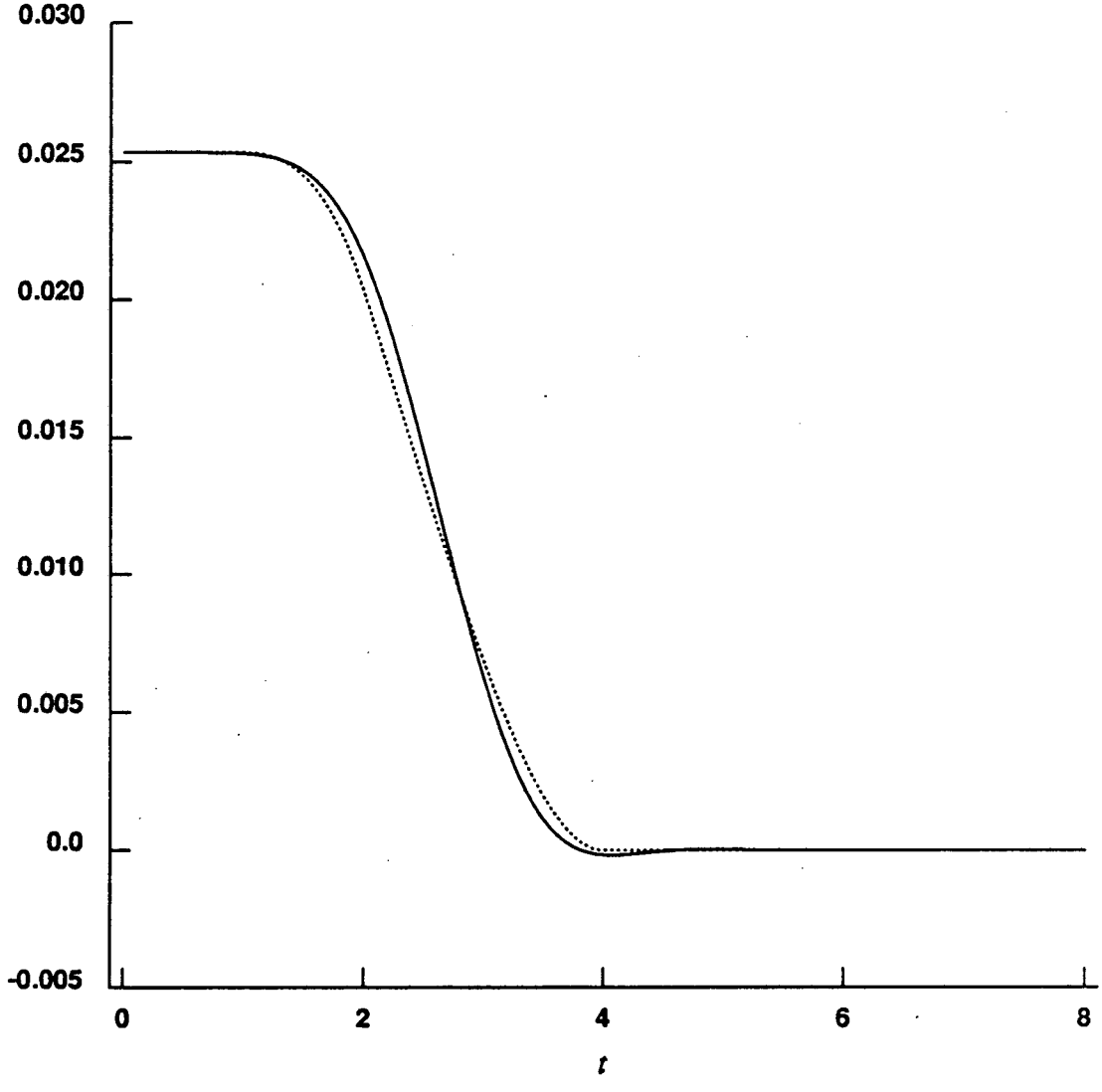


Fig. 4.3. Solid curve = $\hat{\Psi}(\alpha t)$, dotted curve = $\hat{\Psi}(t)$, $\alpha=\sqrt{(52/1.75)}$.

$$\hat{\Psi}(t)=\begin{cases}(2\pi)^{-2} & \text{for } 0\le t\le 1\\ (2\pi)^{-2}(44+2t^2-t^4)/45 & \text{for } 1\le t\le 2\\ (2\pi)^{-2}(256-32t^2+t^4)/180 & \text{for } 2\le t\le 4\\ 0 & \text{for } t\ge 4\end{cases}$$

5. Rezoning

Numerical experiments with infinite order cutoff functions showed that for smooth flows these cutoffs give extremely accurate values of velocity and vorticity for short integration times. Unfortunately, this high accuracy is lost as time increases, so that for long integration times, these cutoffs are not significantly more accurate than lower order cutoffs. Unfortunately, there has been no satisfactory explanation of this phenomenon. We shall show that one way to overcome this problem is the rezoning strategy suggested by Beale and Majda [8]. We will present a version of rezoning similar to theirs, which we call version 1, and a new method, which we call version 2.

First we note that since Ψ_δ approximates the delta function as $\delta \to 0$ we have at time $t=0$ that

$$\omega(\mathbf{z},0) \approx \sum_{\mathbf{j}\in\mathbf{J}} \Psi_\delta(\mathbf{z}-\mathbf{x}_\mathbf{j}(0))c_\mathbf{j}$$

This holds for all $\mathbf{z}$. Here $\mathbf{J}$ is the set of all double indices $\mathbf{j} = (j_1, j_2)$ such that $\mathbf{j}h \in \Omega(0)$, the support of the initial vorticity distribution. Since vorticity is preserved along particle paths, we also expect that at later times t

$$\omega(\mathbf{z}, t) \approx \sum_{\mathbf{j}\in\mathbf{J}} \Psi_\delta(\mathbf{z}-\mathbf{x}_\mathbf{j}(t))c_\mathbf{j} \tag{5.1}$$

In particular, letting $\mathbf{z}=\mathbf{x}_\mathbf{i}(t)$ in (5.1), gives

$$\omega(\mathbf{x}_\mathbf{i}(t), t) \approx \sum_{\mathbf{j}\in\mathbf{J}} \Psi_\delta(\mathbf{x}_\mathbf{i}(t)-\mathbf{x}_\mathbf{j}(t))c_\mathbf{j} \tag{5.2}$$

Multiplying both sides of (5.2) by h^2, and recalling that $c_\mathbf{i}=h^2\omega(\mathbf{x}_\mathbf{i}(t), t)$ for any t gives

$$c_\mathbf{i} \approx h^2 \sum_{\mathbf{j}\in\mathbf{J}} \Psi_\delta(\mathbf{x}_\mathbf{i}(t)-\mathbf{x}_\mathbf{j}(t))c_\mathbf{j} \tag{5.3}$$

Therefore we define

$$c_\mathbf{i}^h(t) = h^2 \sum_{\mathbf{j}\in\mathbf{J}} \Psi_\delta(\mathbf{x}_\mathbf{i}(t)-\mathbf{x}_\mathbf{j}(t))c_\mathbf{j} \tag{5.4}$$

and

$$E_\omega(t) = \left[h^2 \sum_{\mathbf{j}\in\mathbf{J}} (c_\mathbf{j}^h(t) - c_\mathbf{j})^2 \right]^{1/2} \tag{5.5}$$

Here $E_\omega(t)$ stands for "the average vorticity error along vortex paths". With these preliminaries out of the way, we can now present the first version of rezoning.

Version 1

STEP 1 First compute $E_\omega(0)$. Then, as in the standard vortex method, solve the following system of ordinary differential equations

$$\frac{d\tilde{\mathbf{x}}_\mathbf{i}(t)}{dt} = \tilde{\mathbf{u}}_\mathbf{i}(t), \qquad \tilde{\mathbf{x}}_\mathbf{i}(0) = \mathbf{i}h, \tag{5.6}$$

where

$$\tilde{\mathbf{u}}_\mathbf{i}(t) = \sum_{\mathbf{j}\in\mathbf{J}} K_\delta(\tilde{\mathbf{x}}_\mathbf{i}(t) - \tilde{\mathbf{x}}_\mathbf{j}(t))c_\mathbf{j}. \tag{5.7}$$

After each time-step Δt, calculate $E_\omega(t)$ and $E_\omega(t)/E_\omega(0)$. Continue to solve (5.6), (5.7) until

$$\frac{E_\omega(t)}{E_\omega(0)} > \eta$$

where η is a parameter we have to specify. In our numerical experiments we have used η=1.1, η=1.25, or η=1.5. When $E_\omega(t)/E_\omega(0) > \eta$ we no longer solve (5.6), (5.7) but go on to the next step.

STEP 2 Suppose $t=T_1$ when we quit step 1. Now we set $\tilde{\mathbf{z}}_\mathbf{j}(T_1) = \tilde{\mathbf{x}}_\mathbf{j}(T_1)$ for every $\mathbf{j}\in\mathbf{J}$. Then we introduce a new square grid, occupying a region $A \subset R^2$, which is somewhat larger than what is needed to cover all point vortices at time $t=T_1$. Let $\mathbf{J}_A$ denote the set of double indices $\mathbf{j}$ such that $\mathbf{j}h \in A$. For every $\mathbf{j}\in\mathbf{J}_A$ introduce a new vortex at every grid-point $\mathbf{j}h$. We use the old vortices one last time to compute a new "initial" vorticity distribution. To be more precise, we let

$$(c_\mathbf{i})_{new} \equiv h^2 \sum_{\mathbf{j}\in\mathbf{J}} \Psi_\delta(\mathbf{i}h - \tilde{\mathbf{x}}_\mathbf{j}(T_1))(c_\mathbf{j})_{old}, \qquad \text{for every } \mathbf{i}\in\mathbf{J}_A \tag{5.8}$$

Now we "throw away" all the old vortices $\tilde{\mathbf{x}}_\mathbf{j}$ and denote the *new* vortices as $\tilde{\mathbf{x}}_\mathbf{j}$ with $\tilde{\mathbf{x}}_\mathbf{j}(T_1) = \mathbf{j}h$. We then delete all the new vortices $\tilde{\mathbf{x}}_\mathbf{i}$ for which $|\,(c_\mathbf{i})_{new}\,| < \varepsilon$, where ε is a certain tolerance. Let $\mathbf{J}_1$ be the subset of $\mathbf{J}_A$ such that $|\,(c_\mathbf{i})_{new}\,| \geq \varepsilon$ for every $\mathbf{i}\in\mathbf{J}_1$. Now for $t \geq T_1$ we solve the following larger system of ordinary differential equations.

$$\tilde{\mathbf{u}}(\mathbf{x}, t) = \sum_{\mathbf{j}\in \mathbf{J}_1} K_\delta(\mathbf{x} - \tilde{\mathbf{x}}_\mathbf{j}(t))(c_\mathbf{j})_{new}, \tag{5.9}$$

$$\frac{d\tilde{\mathbf{x}}_\mathbf{i}(t)}{dt} = \tilde{\mathbf{u}}(\tilde{\mathbf{x}}_\mathbf{i}, t) \qquad \tilde{\mathbf{x}}_\mathbf{i}(T_1) = \mathbf{i}h, \qquad \text{for every } \mathbf{i}\in \mathbf{J}_1,$$

$$\frac{d\tilde{\mathbf{z}}_\mathbf{i}(t)}{dt} = \tilde{\mathbf{u}}(\tilde{\mathbf{z}}_\mathbf{i}, t) \tag{5.10}$$

Remark: Here the $\tilde{\mathbf{z}}_\mathbf{i}$'s denote the original vortices. If we are not interested in the paths of the original vortices, but the paths of some other particles, we should let the $\tilde{\mathbf{z}}_\mathbf{i}$'s denote these particles. In that case, we do not set $\tilde{\mathbf{z}}_\mathbf{j}(T_1) = \tilde{\mathbf{x}}_\mathbf{j}(T_1)$ at the begining of step 2.

Now again we compute $E_\omega(t)$ and $E_\omega(t)/E_\omega(T_1)$ after every time-step Δt, but now using the new vorticity coefficients in (5.4) and (5.5). Continue solving (5.9), (5.10) until $E_\omega(t)/E_\omega(T_1) > \eta$.

STEP 3 Suppose $t=T_2$ when we quit step 2. Now repeat step 2 but replacing T_1 by T_2 in all the equations. Also in this step, *do not* set $\tilde{\mathbf{z}}_\mathbf{j}(T_2) = \tilde{\mathbf{x}}_\mathbf{j}(T_2)$, for $\mathbf{j}\in \mathbf{J}_1$. Continue this process until we reach $t=T_{\max}$.

Numerical experiments using this technique have indicated a great reduction in velocity errors for long integration times compared to the corresponding errors without rezoning, but with the same grid-spacing h. In vortex methods without rezoning, we are forced to pick a considerably larger value of δ for long integration times. With rezoning however, we can usually use values of δ close to the optimal value of δ at time $t=0$. Nevertheless, we have also noted that when we use this form of rezoning, the velocity error takes a jump every time a new vorticity distribution is computed using (5.8). These jumps are small compared to the sharp increase in error experienced when no rezoning is applied, but still significant compared to the marginal increase in velocity error at intermediate times. To reduce this effect, and the effect of "numerical viscosity" we propose the following scheme.

Version 2

In version 2 we introduce a finer grid than in version 1, for the purpose of calculating the new vorticity distributions on the new grids. The velocity evaluations are however done on the coarser grid. This method has some similarities with multi-grid methods, but it does not fall into the framework of such methods. The details of version 2 are as follows.

STEP 1 Let Q_0 be the set of double indices $\mathbf{q}=(q_1, q_2)$ such that $\mathbf{q}h \in \Omega(0)$, where q_1 and q_2 are *integers or half-integers.* For every $\mathbf{q} \in Q_0$ introduce a vortex $\mathbf{x}_\mathbf{q}$ with strength $c_\mathbf{q}=\omega(\mathbf{q}h)h^2$. Let $\mathbf{J}$ be the integer-pair subset of Q_0. Then solve the following system of ordinary differential equations, for every $\mathbf{q} \in Q_0$.

$$\frac{d\tilde{\mathbf{x}}_\mathbf{q}(t)}{dt} = \tilde{\mathbf{u}}_\mathbf{q}(t), \qquad \tilde{\mathbf{x}}_\mathbf{q}(0) = \mathbf{q}h, \tag{5.11}$$

where

$$\tilde{\mathbf{u}}_\mathbf{q}(t) = \sum_{\mathbf{j} \in \mathbf{J}} K_\delta(\tilde{\mathbf{x}}_\mathbf{q}(t) - \tilde{\mathbf{x}}_\mathbf{j}(t))c_\mathbf{j}. \tag{5.12}$$

After each time-step Δt, calculate $E_\omega(t)$ and $E_\omega(t)/E_\omega(0)$ using (5.4),(5.5). Note that in calculating $E_\omega(t)$ and $E_\omega(t)/E_\omega(0)$ we use *only* the vortices and vorticity coefficients with integer indices. Continue to solve (5.11), (5.12) until

$$\frac{E_\omega(t)}{E_\omega(0)} > \eta$$

STEP 2 Suppose $t=T_1$ when we quit step 1. Set $\tilde{\mathbf{z}}_\mathbf{j}(T_1) = \tilde{\mathbf{x}}_\mathbf{j}(T_1)$ for every $\mathbf{j} \in \mathbf{J}$. As in version 1, we introduce a new square grid, occupying a region $A \subset R^2$. Let Q_A be the set of double indices $\mathbf{q}=(q_1, q_2)$ such that $\mathbf{q}h \in A$, with q_1 and q_2 assuming both integer and half-integer values. Let $\mathbf{J}_A$ be the integer-pair subset of Q_A. For every $\mathbf{q} \in Q_A$ intoduce a new vortex at every grid-point $\mathbf{q}h$. Define the new vorticity distribution by

$$(c_\mathbf{q})_{new} \equiv h^2/4 \sum_{\mathbf{r} \in Q_0} \Psi_{\delta'}(\mathbf{q}h - \tilde{\mathbf{x}}_\mathbf{r}(T_1))(c_\mathbf{r})_{old}, \qquad \text{for every } \mathbf{q} \in Q_A \tag{5.13}$$

Note that the effective grid-spacing in (5.13) is $h/2$ rather than h. That is why we have a factor of $h^2/4$ in front of the summation sign instead of h^2. We must also replace Ψ_δ by $\Psi_{\delta'}$, where δ' is the cutoff parameter corresponding to a grid-spacing of $h/2$. If for any value of h we pick $\delta = constant \cdot h^q$, then $\delta' = 2^{-q}\delta$. The purpose of using (5.13) instead of (5.8) is to make the error in the computed new vorticity distribution small compared to the error in the velocity evaluations, thereby reducing "numerical viscosity". As in version 1, we now "throw away" the old vortices $\tilde{\mathbf{x}}_\mathbf{q}$ and use this notation for the new vortices such that $\tilde{\mathbf{x}}_\mathbf{q}(T_1) = \mathbf{q}h$. Then let Q_1 be the subset of Q_A such that $|(c_\mathbf{q})_{new}| \geq \varepsilon$ for every $\mathbf{q} \in Q_1$, where ε is a certain tolerance. Let $\mathbf{J}_1$ be the integer-pair subset of Q_1. For $t \geq T_1$ solve the following system of ordinary

differential equations.

$$\tilde{\mathbf{u}}(\mathbf{x}, t) = \sum_{\mathbf{j}\in \mathbf{J}_1} K_\delta(\mathbf{x} - \tilde{\mathbf{x}}_\mathbf{j}(t))(c_\mathbf{j})_{new}, \tag{5.14}$$

$$\frac{d\tilde{\mathbf{x}}_\mathbf{q}(t)}{dt} = \tilde{\mathbf{u}}(\tilde{\mathbf{x}}_\mathbf{q}, t) \qquad \tilde{\mathbf{x}}_\mathbf{q}(T_1) = \mathbf{q}h, \qquad \text{for every } \mathbf{q}\in Q_1,$$

$$\frac{d\tilde{\mathbf{z}}_\mathbf{l}(t)}{dt} = \tilde{\mathbf{u}}(\tilde{\mathbf{z}}_\mathbf{l}, t) \tag{5.15}$$

After every timestep Δt compute $E_\omega(t)$ and $E_\omega(t)/E_\omega(T_1)$ using the new vorticity coefficients with integer indices, and continue solving (5.14), (5.15) until $E_\omega(t)/E_\omega(T_1) > \eta$.

STEP 3 Suppose $t=T_2$ when we quit step 2. Repeat step 2 replacing T_1 by T_2 where applicable. As in version 1, do not set $\tilde{\mathbf{z}}_\mathbf{j}(T_2) = \tilde{\mathbf{x}}_\mathbf{j}(T_2)$, for $\mathbf{j}\in \mathbf{J}_1$ at this point. Continue in the same manner until $t=T_{\max}$.

6. Numerical Results

We present five test problems. In the first three test problems, the solution is known analytically since the vorticity distribution is radially symmetric: The flow is circular, and the velocity at any point and at any time is given by

$$\mathbf{u}(x,y) = (u,v) = \mu(r)(-y,x)^T \tag{6.1}$$

where $\mu(r) = \frac{1}{r^2}\int_0^r s\,\omega(s)\,ds$ is the angular velocity of the flow.

The other two test problems do not have a known analytical solution. Here we show the numerical solutions graphically, and the rate of convergence is estimated by using Richardsons's extrapolation. In the first three test problems, two different cutoff functions are used, namely Hald's infinite order cutoff and our 8-th order cutoff. Unless specified otherwise, we have used version 2 rezoning. In all cases we used the classical fourth order Runge-Kutta method for time integration.

We now look at the first three test problems in detail. In each of these, the vorticity distribution has the form

$$\omega(r) = \begin{cases} (1-r^2)^k & \text{for } r \le 1 \\ 0 & \text{for } r > 1 \end{cases} \tag{6.2}$$

where k=3 in test problem #1, k=7 in test problem #2 and k=14 in test problem #3. The case k=3 has been tested numerically by Beale and Majda [8] and Beale [5], and the case k=7 by Perlman [22].

It can be shown that the Fourier transform of a vorticity distribution of this form is $C_k J_{k+1}(t)/t^{k+1}$ where C_k is a constant. Thus, $\hat{\omega}(t)$ is of order $O(t^{-(k+1.5)})$ as $t \to \infty$, although $\omega(r)$ has only k bounded derivatives. In general, a vorticity distribution ω with compact support and k bounded derivatives only guarantees that $\hat{\omega}(t)$ is of order $O(t^{-k})$ as $t \to \infty$. This means that for test problems #1-3, Hald's estimate, [16, p. 568], of the moment error for infinite-order cutoffs can be improved to order $O(\delta^{k+0.5})$ rather than $O(\delta^{k-1})$.

The solutions of the Euler equation for these vorticity distributions are given by (6.1) with

$$\mu(r) = \begin{cases} \dfrac{1-(1-r^2)^{k+1}}{2(k+1)r^2} & \text{for } r \le 1 \\ \dfrac{1}{2(k+1)r^2} & \text{for } r > 1 \end{cases} \tag{6.3}$$

We measure the velocity error in the discrete L^2 norm.

$$E_u = \left[h^2 \sum_{j \in J} | u_j(t) - \tilde{u}_j(t) |^2 \right]^{1/2} \tag{6.4}$$

The rate of convergence is estimated by using two successive values of h.

$$\text{rate of convergence} = \frac{\ln(E_u(h_1)/E_u(h_2))}{\ln(h_1/h_2)} \tag{6.5}$$

Tables 6.1a and 6.1b give the velocity errors in test problem #1 at different times up to time t=50, for different values of h, and for the two different cutoff functions. We choose $\delta = constant \cdot \sqrt{h}$ so that the moment error and the discretization error will be of approximately the same order in h. The proportionality constant has been chosen so as to minimize the velocity error at time t=0 when h=0.100. Comparing these two tables, we see that the errors are only between 5% and 29% larger when the eighth order cutoff is used. The rates of convergence for the two cutoffs are also approximately the same for corresponding times t, as seen in tables (6.4a) and (6.4b). Since we take δ proportional to $\sqrt{h}$, we can expect the moment error to be of order $O(h^{1.75})$ and the discretization of order $O(h^2)$, with the infinite-order cutoff function. We could therefore only expect a rate of convergence of 1.75. However, the observed rate of convergence is greater than 2. In particular, at time t=0, the computed rate of convergence is 2.3, both using infinite-order and eighth order cutoff. Since Perlman [22] has shown numerically that at time t=0 the moment error is much larger than the discretization error, this seems to indicate that the moment error is actually of order $O(\delta^{4.6})$ rather than the best theoretical estimate of $O(\delta^{3.5})$. Beale and Majda [8] observed a rate of convergence of 3.6 at time t=0 for this problem, using an eighth order Gaussian cutoff function, but with δ proportional to $h^{0.75}$, which would correspond to a moment error of order $O(\delta^{4.8})$ which is quite close to what we observed. Beale and Majda [8] also applied rezoning to this problem, with h=0.125 and δ=0.25. In comparing our results with theirs, we have to take into account that they reported a relative velocity error, obtained by dividing the absolute error by an average velocity U, where

$$U^2 = \frac{1}{\pi}\int\int_{r\le 1} |\mathbf{u}|^2\, dx\, dy = 2\int_0^1 |\mathbf{u}|^2\, r\, dr.$$

For test problem #1, $U \approx 0.1505$, so if we divide our values of E_u in tables 6.1a and 6.1b by 0.1505, we obtain relative velocity errors ranging from 0.011% at time t=0 to 0.16% at time t=35, using infinite-order cutoff with h=0.125 and $\delta = 0.3\sqrt{h} \approx 0.1061$. With the eighth order cutoff, the corresponding relative errors are 0.012% at time t=0 and 0.23% at time t=35. Beale and Majda [8] reported relative errors of 0.055% at time t=0 and 0.30% at time t=36.

The results of test problem #2 are summarized in tables 6.2a-c. As in test problem #1, we pick $\delta = 1.7\sqrt{h}$, for the eighth order cutoff, and $\delta = 0.3\sqrt{h}$ for the infinite order cutoff. Numerical tests have shown that these values of δ are close to the optimal ones at time t=0, with h=0.100, even for this vorticity distribution. We also repeated the tests using the infinite-order cutoff but with $\delta = 0.355\sqrt{h}$. Comparing table 6.2a to table 6.2b we see that the errors using the infinite-order cutoff are smaller than the corresponding errors for the eighth order cutoff by a factor ranging from about 3 to 6. Nevertheless, the rate of convergence is around 4 for both methods at all times. Theoretically, the moment error for infinite-order cutoffs is of order $O(\delta^{7.5})$ for this vorticity distribution, so since δ is proportional to $\sqrt{h}$ we would expect a rate of convergence of 3.75 in this case. Hence, the observed rate of convergence is slightly higher than the theoretical rate as in test problem #1. Now comparing table 6.2a to table 6.2c we notice that by choosing a larger proportionality factor between δ and $\sqrt{h}$ we get larger velocity errors at time t=0 as expected since the moment error increases. At later times however, the errors seem to become almost equal for the two δ's but always with the smaller error for the smaller δ. This is very different from what we get in vortex methods without rezoning, where the errors at later times are smaller for larger values of δ. Perlman [22] tested Gaussian cutoff functions of different orders on this vorticity distribution, but without rezoning. Using an eighth order Gaussian cutoff, she had to take $\delta = h^{0.7}$ to minimize the velocity error at time t=10 and $\delta = h^{0.6}$ to minimize the error at time t=20. With these parameters and h=0.05, she obtained a minimum error of $7.42 \cdot 10^{-5}$ at time t=10 and $5.77 \cdot 10^{-4}$ at time t=20. Our smallest errors at time t=10 and t=20, with h=0.05 are $3.59 \cdot 10^{-7}$ and $7.36 \cdot 10^{-7}$ respectively, so the rezoning procedure seems to pay off, at least when the flow is this smooth.

TABLE 6.1a

	E_u			
t	h=0.125	h=0.100	h=0.0625	h=0.05
0.0	$0.1665 \cdot 10^{-4}$	$0.9000 \cdot 10^{-5}$	$0.3142 \cdot 10^{-5}$	$0.1886 \cdot 10^{-5}$
5.0	$0.2962 \cdot 10^{-4}$		$0.4347 \cdot 10^{-5}$	$0.2552 \cdot 10^{-5}$
10.0	$0.5340 \cdot 10^{-4}$	$0.2562 \cdot 10^{-4}$	$0.6775 \cdot 10^{-5}$	$0.3829 \cdot 10^{-5}$
15.0	$0.8373 \cdot 10^{-4}$		$0.1006 \cdot 10^{-4}$	$0.5443 \cdot 10^{-5}$
20.0	$0.1201 \cdot 10^{-3}$	$0.5473 \cdot 10^{-4}$	$0.1377 \cdot 10^{-4}$	$0.7372 \cdot 10^{-5}$
25.0	$0.1590 \cdot 10^{-3}$		$0.1810 \cdot 10^{-4}$	$0.9633 \cdot 10^{-5}$
30.0	$0.2011 \cdot 10^{-3}$	$0.9110 \cdot 10^{-4}$	$0.2268 \cdot 10^{-4}$	$0.1200 \cdot 10^{-4}$
35.0	$0.2479 \cdot 10^{-3}$		$0.2776 \cdot 10^{-4}$	$0.1456 \cdot 10^{-4}$
40.0	$0.2982 \cdot 10^{-3}$	$0.1358 \cdot 10^{-3}$	$0.3321 \cdot 10^{-4}$	$0.1727 \cdot 10^{-4}$
45.0	$0.3539 \cdot 10^{-3}$		$0.3890 \cdot 10^{-4}$	$0.2021 \cdot 10^{-4}$
50.0	$0.4141 \cdot 10^{-3}$	$0.1880 \cdot 10^{-3}$	$0.4559 \cdot 10^{-4}$	$0.2327 \cdot 10^{-4}$

$\omega(r)=(\max(0,1-r^2))^3$

Cutoff function: *Hald's infinite–order*

$\eta = 1.1$

$\delta = 0.3\sqrt{h}$

$\Delta t = 4.0h$

$T_{\max} = 50.0$

TABLE 6.1b

	E_u			
t	h=0.125	h=0.100	h=0.0625	h=0.05
0.0	$0.1751 \cdot 10^{-4}$	$0.9473 \cdot 10^{-5}$	$0.3329 \cdot 10^{-5}$	$0.1998 \cdot 10^{-5}$
5.0	$0.3398 \cdot 10^{-4}$		$0.4969 \cdot 10^{-5}$	$0.2689 \cdot 10^{-5}$
10.0	$0.6122 \cdot 10^{-4}$	$0.2945 \cdot 10^{-4}$	$0.7661 \cdot 10^{-5}$	$0.4233 \cdot 10^{-5}$
15.0	$0.9967 \cdot 10^{-4}$		$0.1103 \cdot 10^{-4}$	$0.6031 \cdot 10^{-5}$
20.0	$0.1519 \cdot 10^{-3}$	$0.6172 \cdot 10^{-4}$	$0.1497 \cdot 10^{-4}$	$0.8168 \cdot 10^{-5}$
25.0	$0.2114 \cdot 10^{-3}$		$0.1956 \cdot 10^{-4}$	$0.1060 \cdot 10^{-4}$
30.0	$0.2757 \cdot 10^{-3}$	$0.1062 \cdot 10^{-3}$	$0.2429 \cdot 10^{-4}$	$0.1339 \cdot 10^{-4}$
35.0	$0.3467 \cdot 10^{-3}$		$0.2999 \cdot 10^{-4}$	$0.1642 \cdot 10^{-4}$
40.0	$0.4207 \cdot 10^{-3}$	$0.1594 \cdot 10^{-3}$	$0.3609 \cdot 10^{-4}$	$0.1974 \cdot 10^{-4}$
45.0	$0.4972 \cdot 10^{-3}$		$0.4284 \cdot 10^{-4}$	$0.2311 \cdot 10^{-4}$
50.0	$0.5799 \cdot 10^{-3}$	$0.2225 \cdot 10^{-3}$	$0.5000 \cdot 10^{-4}$	$0.2681 \cdot 10^{-4}$

$\omega(r)=(\max(0,1-r^2))^3$

Cutoff function: 8–*th order with compact support*

$\eta = 1.1$

$\delta = 1.7\sqrt{h}$

$\Delta t = 4.0h$

$T_{max} = 50.0$

TABLE 6.2a

	E_u			
t	h=0.125	h=0.100	h=0.0625	h=0.05
0.0	$0.3242 \cdot 10^{-5}$	$0.1293 \cdot 10^{-5}$	$0.1587 \cdot 10^{-6}$	$0.6384 \cdot 10^{-7}$
5.0	$0.9354 \cdot 10^{-5}$		$0.4465 \cdot 10^{-6}$	$0.1859 \cdot 10^{-6}$
10.0	$0.1634 \cdot 10^{-4}$	$0.5244 \cdot 10^{-5}$	$0.8657 \cdot 10^{-6}$	$0.3588 \cdot 10^{-6}$
15.0	$0.2375 \cdot 10^{-4}$		$0.1316 \cdot 10^{-5}$	$0.5432 \cdot 10^{-6}$
20.0	$0.3291 \cdot 10^{-4}$	$0.1108 \cdot 10^{-4}$	$0.1791 \cdot 10^{-5}$	$0.7356 \cdot 10^{-6}$
25.0	$0.4298 \cdot 10^{-4}$		$0.2287 \cdot 10^{-5}$	$0.9409 \cdot 10^{-6}$
30.0	$0.5286 \cdot 10^{-4}$	$0.1815 \cdot 10^{-4}$	$0.2810 \cdot 10^{-5}$	$0.1143 \cdot 10^{-5}$
35.0	$0.6505 \cdot 10^{-4}$		$0.3347 \cdot 10^{-5}$	$0.1358 \cdot 10^{-5}$
40.0	$0.7567 \cdot 10^{-4}$	$0.2639 \cdot 10^{-4}$	$0.3910 \cdot 10^{-5}$	$0.1580 \cdot 10^{-5}$
45.0	$0.8856 \cdot 10^{-4}$		$0.4494 \cdot 10^{-5}$	$0.1809 \cdot 10^{-5}$
50.0	$0.1027 \cdot 10^{-3}$	$0.3600 \cdot 10^{-4}$	$0.5098 \cdot 10^{-5}$	$0.2046 \cdot 10^{-5}$
55.0			$0.5724 \cdot 10^{-5}$	
60.0			$0.6373 \cdot 10^{-5}$	
65.0			$0.7045 \cdot 10^{-5}$	
70.0			$0.7742 \cdot 10^{-5}$	
75.0			$0.8460 \cdot 10^{-5}$	
80.0			$0.9203 \cdot 10^{-5}$	
85.0			$0.9967 \cdot 10^{-5}$	
90.0			$0.1076 \cdot 10^{-4}$	
95.0			$0.1157 \cdot 10^{-4}$	
100.0			$0.1240 \cdot 10^{-4}$	

$\omega(r)=(\max(0,1-r^2))^7$
Cutoff function: *Hald's infinite–order*
$\eta = 1.25$
$\delta = 0.3\sqrt{h}$
$\Delta t = 4.0h$

TABLE 6.2b

	E_u			
t	h=0.125	h=0.100	h=0.0625	h=0.05
0.0	$0.1352 \cdot 10^{-4}$	$0.4651 \cdot 10^{-5}$	$0.8460 \cdot 10^{-6}$	$0.3576 \cdot 10^{-6}$
5.0	$0.3361 \cdot 10^{-4}$		$0.1989 \cdot 10^{-5}$	$0.7915 \cdot 10^{-6}$
10.0	$0.6466 \cdot 10^{-4}$	$0.2441 \cdot 10^{-4}$	$0.3822 \cdot 10^{-5}$	$0.1518 \cdot 10^{-5}$
15.0	$0.1103 \cdot 10^{-3}$		$0.6061 \cdot 10^{-5}$	$0.2336 \cdot 10^{-5}$
20.0	$0.1521 \cdot 10^{-3}$	$0.5617 \cdot 10^{-4}$	$0.8522 \cdot 10^{-5}$	$0.3209 \cdot 10^{-5}$
25.0	$0.2077 \cdot 10^{-3}$		$0.1117 \cdot 10^{-4}$	$0.4182 \cdot 10^{-5}$
30.0	$0.2642 \cdot 10^{-3}$	$0.9726 \cdot 10^{-4}$	$0.1414 \cdot 10^{-4}$	$0.5246 \cdot 10^{-5}$
35.0	$0.3282 \cdot 10^{-3}$		$0.1723 \cdot 10^{-4}$	$0.6370 \cdot 10^{-5}$
40.0	$0.3964 \cdot 10^{-3}$	$0.1460 \cdot 10^{-3}$	$0.2057 \cdot 10^{-4}$	$0.7588 \cdot 10^{-5}$
45.0	$0.4676 \cdot 10^{-3}$		$0.2415 \cdot 10^{-4}$	$0.8886 \cdot 10^{-5}$
50.0	$0.5477 \cdot 10^{-3}$	$0.2034 \cdot 10^{-3}$	$0.2794 \cdot 10^{-4}$	$0.1023 \cdot 10^{-4}$
55.0			$0.3186 \cdot 10^{-4}$	
60.0			$0.3600 \cdot 10^{-4}$	
65.0			$0.4030 \cdot 10^{-4}$	
70.0			$0.4481 \cdot 10^{-4}$	
75.0			$0.4947 \cdot 10^{-4}$	
80.0			$0.5440 \cdot 10^{-4}$	
85.0			$0.5948 \cdot 10^{-4}$	
90.0			$0.6476 \cdot 10^{-4}$	
95.0			$0.7012 \cdot 10^{-4}$	
100.0			$0.7578 \cdot 10^{-4}$	

$\omega(r)=(\max(0,1-r^2))^7$
Cutoff function: 8–*th order with compact support*
$\eta = 1.25$
$\delta = 1.7\sqrt{h}$
$\Delta t = 4.0h$

TABLE 6.2c

	E_u			
t	h=0.125	h=0.100	h=0.0625	h=0.05
0.0	$0.1740 \cdot 10^{-4}$	$0.5371 \cdot 10^{-5}$	$0.7276 \cdot 10^{-6}$	$0.2645 \cdot 10^{-6}$
5.0	$0.3007 \cdot 10^{-4}$		$0.9202 \cdot 10^{-6}$	$0.3368 \cdot 10^{-6}$
10.0	$0.5173 \cdot 10^{-4}$	$0.1276 \cdot 10^{-4}$	$0.1397 \cdot 10^{-5}$	$0.4897 \cdot 10^{-6}$
15.0	$0.7608 \cdot 10^{-4}$		$0.1820 \cdot 10^{-5}$	$0.6774 \cdot 10^{-6}$
20.0	$0.9869 \cdot 10^{-4}$	$0.2411 \cdot 10^{-4}$	$0.2347 \cdot 10^{-5}$	$0.8770 \cdot 10^{-6}$
25.0	$0.1225 \cdot 10^{-3}$		$0.2890 \cdot 10^{-5}$	$0.1087 \cdot 10^{-5}$
30.0	$0.1486 \cdot 10^{-3}$	$0.3916 \cdot 10^{-4}$	$0.3456 \cdot 10^{-5}$	$0.1305 \cdot 10^{-5}$
35.0	$0.1727 \cdot 10^{-3}$		$0.4035 \cdot 10^{-5}$	$0.1532 \cdot 10^{-5}$
40.0	$0.2002 \cdot 10^{-3}$	$0.5612 \cdot 10^{-4}$	$0.4633 \cdot 10^{-5}$	$0.1763 \cdot 10^{-5}$
45.0	$0.2290 \cdot 10^{-3}$		$0.5259 \cdot 10^{-5}$	$0.2002 \cdot 10^{-5}$
50.0	$0.2565 \cdot 10^{-3}$	$0.7570 \cdot 10^{-4}$	$0.5899 \cdot 10^{-5}$	$0.2248 \cdot 10^{-5}$
55.0			$0.6553 \cdot 10^{-5}$	
60.0			$0.7226 \cdot 10^{-5}$	
65.0			$0.7912 \cdot 10^{-5}$	
70.0			$0.8620 \cdot 10^{-5}$	
75.0			$0.9344 \cdot 10^{-5}$	
80.0			$0.1009 \cdot 10^{-4}$	
85.0			$0.1084 \cdot 10^{-4}$	
90.0			$0.1162 \cdot 10^{-4}$	
95.0			$0.1242 \cdot 10^{-4}$	
100.0			$0.1323 \cdot 10^{-4}$	

$\omega(r)=(\max(0,1-r^2))^7$
Cutoff function: *Hald's infinite–order*
$\eta = 1.25$
$\delta = 0.355\sqrt{h}$
$\Delta t = 4.0h$

TABLE 6.3a

	E_u			
t	h=0.125	h=0.100	h=0.0625	h=0.05
0.0	$0.4859 \cdot 10^{-4}$	$0.1782 \cdot 10^{-4}$	$0.1304 \cdot 10^{-5}$	$0.2337 \cdot 10^{-6}$
5.0	$0.1002 \cdot 10^{-3}$		$0.2428 \cdot 10^{-5}$	$0.4164 \cdot 10^{-6}$
10.0	$0.2033 \cdot 10^{-3}$	$0.8081 \cdot 10^{-4}$	$0.4243 \cdot 10^{-5}$	$0.7356 \cdot 10^{-6}$
15.0	$0.3154 \cdot 10^{-3}$		$0.6225 \cdot 10^{-5}$	$0.1085 \cdot 10^{-5}$
20.0	$0.4347 \cdot 10^{-3}$	$0.1502 \cdot 10^{-3}$	$0.8455 \cdot 10^{-5}$	$0.1445 \cdot 10^{-5}$
25.0	$0.5720 \cdot 10^{-3}$		$0.1081 \cdot 10^{-4}$	$0.1813 \cdot 10^{-5}$
30.0	$0.7052 \cdot 10^{-3}$	$0.2375 \cdot 10^{-3}$	$0.1321 \cdot 10^{-4}$	$0.2184 \cdot 10^{-5}$
35.0	$0.8539 \cdot 10^{-3}$		$0.1561 \cdot 10^{-4}$	$0.2559 \cdot 10^{-5}$
40.0	$0.1000 \cdot 10^{-2}$	$0.3369 \cdot 10^{-3}$	$0.1804 \cdot 10^{-4}$	$0.2940 \cdot 10^{-5}$
45.0	$0.1160 \cdot 10^{-2}$		$0.2070 \cdot 10^{-4}$	$0.3325 \cdot 10^{-5}$
50.0	$0.1339 \cdot 10^{-2}$	$0.4475 \cdot 10^{-3}$	$0.2357 \cdot 10^{-4}$	$0.3711 \cdot 10^{-5}$

$\omega(r)=(\max(0,1-r^2))^{14}$

Cutoff function: *Hald's infinite–order*

$\eta = 1.1$

$\delta = 0.3\sqrt{h}$

$\Delta t = 4.0h$

$T_{max} = 50.0$

TABLE 6.3b

	E_u			
t	h=0.125	h=0.100	h=0.0625	h=0.05
0.0	$0.5584 \cdot 10^{-4}$	$0.2759 \cdot 10^{-3}$	$0.5139 \cdot 10^{-5}$	$0.2264 \cdot 10^{-5}$
5.0	$0.1416 \cdot 10^{-3}$		$0.1146 \cdot 10^{-4}$	$0.9267 \cdot 10^{-5}$
10.0	$0.2795 \cdot 10^{-3}$	$0.1289 \cdot 10^{-3}$	$0.2214 \cdot 10^{-4}$	$0.9603 \cdot 10^{-5}$
15.0	$0.4881 \cdot 10^{-3}$		$0.3403 \cdot 10^{-4}$	$0.1485 \cdot 10^{-4}$
20.0	$0.7324 \cdot 10^{-3}$	$0.2955 \cdot 10^{-3}$	$0.4790 \cdot 10^{-4}$	$0.2050 \cdot 10^{-4}$
25.0	$0.9890 \cdot 10^{-3}$		$0.6326 \cdot 10^{-4}$	$0.2662 \cdot 10^{-4}$
30.0	$0.1238 \cdot 10^{-2}$	$0.7755 \cdot 10^{-3}$	$0.7981 \cdot 10^{-4}$	$0.3306 \cdot 10^{-4}$
35.0	$0.1554 \cdot 10^{-2}$		$0.9687 \cdot 10^{-4}$	$0.3997 \cdot 10^{-4}$
40.0	$0.1807 \cdot 10^{-2}$	$0.1819 \cdot 10^{-2}$	$0.1149 \cdot 10^{-3}$	$0.4732 \cdot 10^{-4}$
45.0	$0.2160 \cdot 10^{-2}$		$0.1340 \cdot 10^{-3}$	$0.5512 \cdot 10^{-4}$
50.0	$0.2522 \cdot 10^{-2}$	$0.3071 \cdot 10^{-2}$	$0.1554 \cdot 10^{-3}$	$0.6337 \cdot 10^{-4}$

$\omega(r)=(\max(0,1-r^2))^{14}$

Cutoff function: 8–*th order with compact support*

$\eta = 1.1$

$\delta = 1.7\sqrt{h}$

$\Delta t = 4.0h$

$T_{max} = 50.0$

TABLE 6.4a

Rate of convergence of the velocity approximations in test problems 1-3, using Hald's infinite order cutoff. $\delta=0.3\sqrt{h}$ except as indicated otherwise.

	Rate of Convergence			
t	$\omega(r)=(1.0-r^2)^3$	$\omega(r)=(1.0-r^2)^7$	$\omega(r)=(1.0-r^2)^7$, $\delta=0.355\sqrt{h}$	$\omega(r)=(1.0-r^2)^{14}$
0.0	2.3	4.1	4.5	7.7
10.0	2.6	4.0	4.7	7.8
20.0	2.8	4.0	4.4	7.9
30.0	2.8	4.0	4.4	8.1
40.0	2.9	4.1	4.3	8.1
50.0	3.0	4.1	4.3	8.3

TABLE 6.4b

Rate of convergence of the velocity approximations in test problems 1-3, using the 8-th order cutoff. $\delta=1.7\sqrt{h}$.

	Rate of Convergence		
t	$\omega(r)=(1.0-r^2)^3$	$\omega(r)=(1.0-r^2)^7$	$\omega(r)=(1.0-r^2)^{14}$
0.0	2.3	3.9	3.7
10.0	2.7	4.1	3.7
20.0	2.7	4.4	3.8
30.0	2.7	4.4	4.0
40.0	2.7	4.5	4.0
50.0	2.8	4.5	4.0

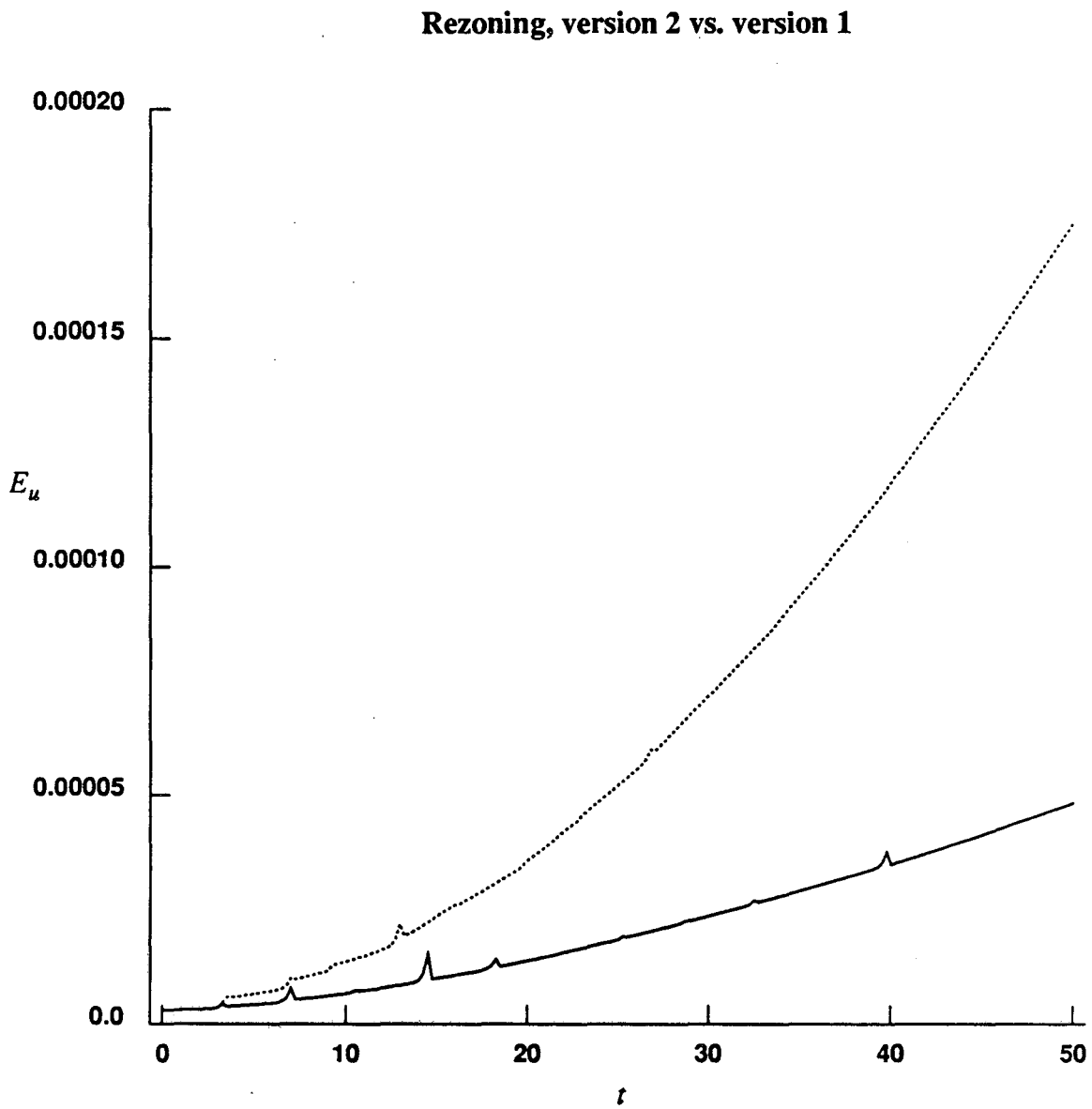


Fig. 6.1 $\omega(r)=(\max(0,1-r^2))^3$, solid curve = version 2, dotted curve = version 1, $h=0.0625$, $\delta=0.3\sqrt{h}$, $\eta=1.25$, $\Delta t=4.0h$.

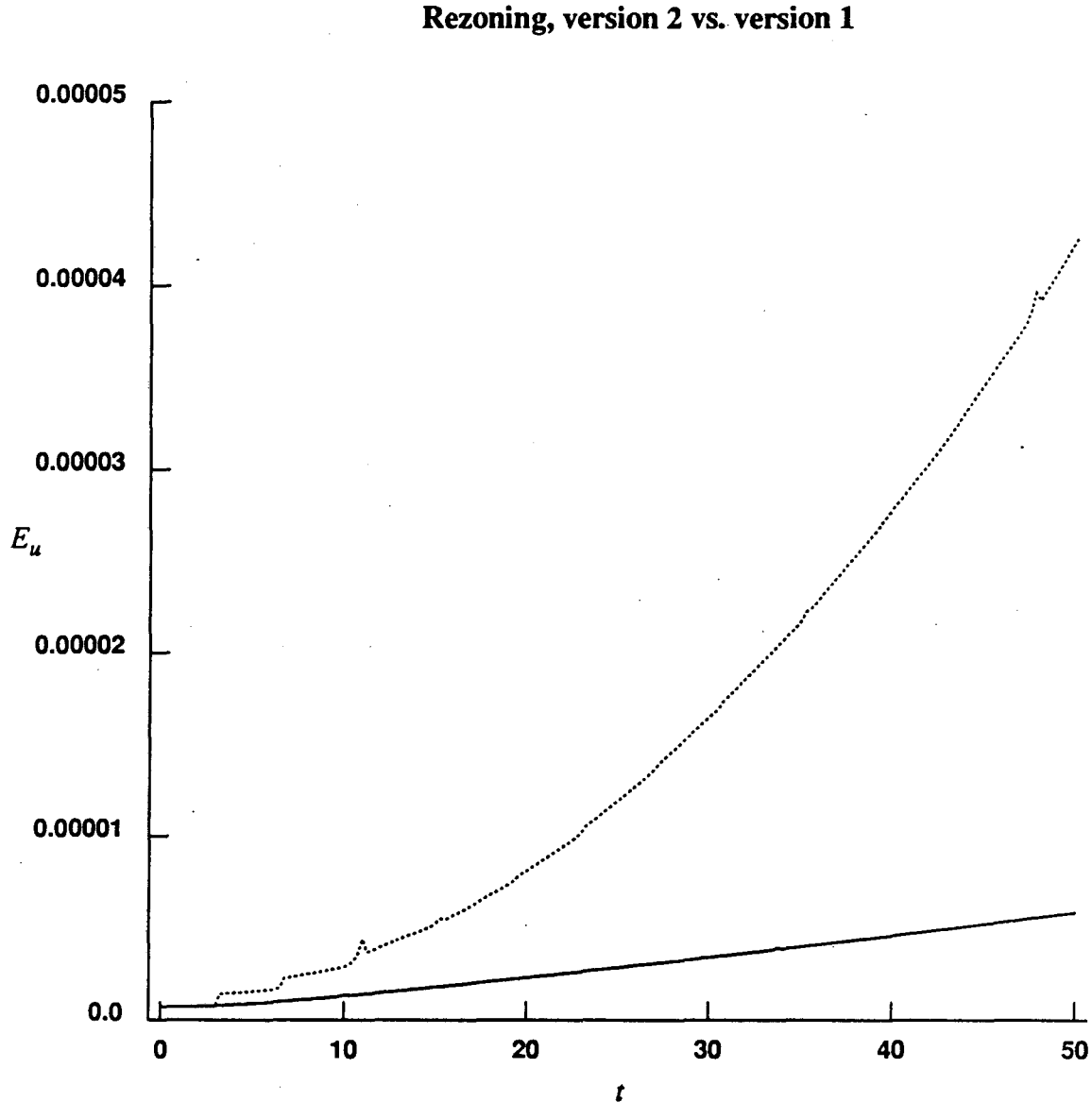


Fig. 6.2 $\omega(r)=(\max(0,1-r^2))^7$, solid curve = version 2, dotted curve = version 1, $h=0.0625$, $\delta=0.355\sqrt{h}$, $\eta=1.25$, $\Delta t=4.0h$.

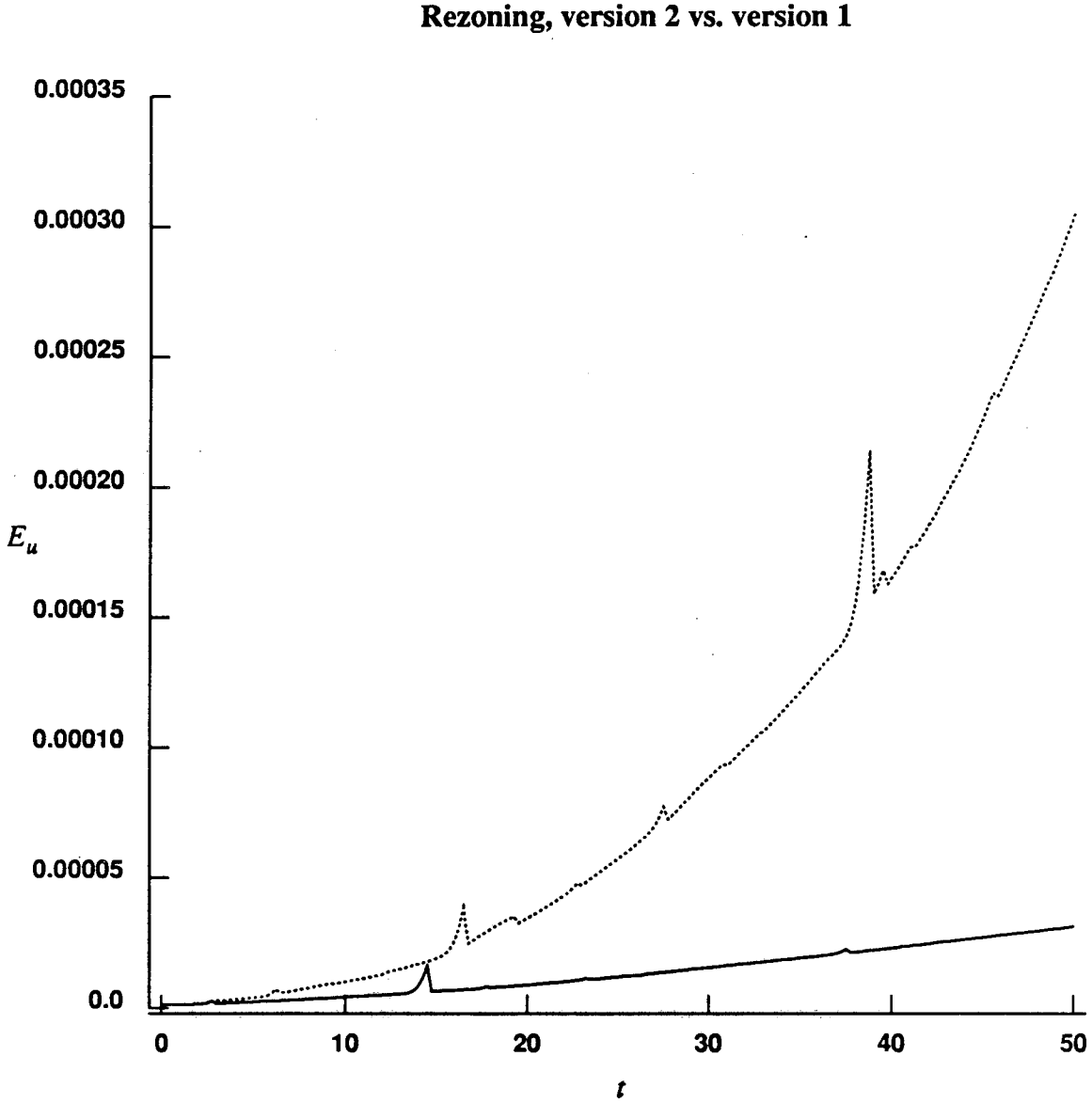


Fig. 6.3 $\omega(r)=(\max(0,1-r^2))^{14}$, solid curve = version 2, dotted curve = version 1, $h=0.0625$, $\delta=0.3\sqrt{h}$, $\eta=1.25$, $\Delta t=4.0h$.

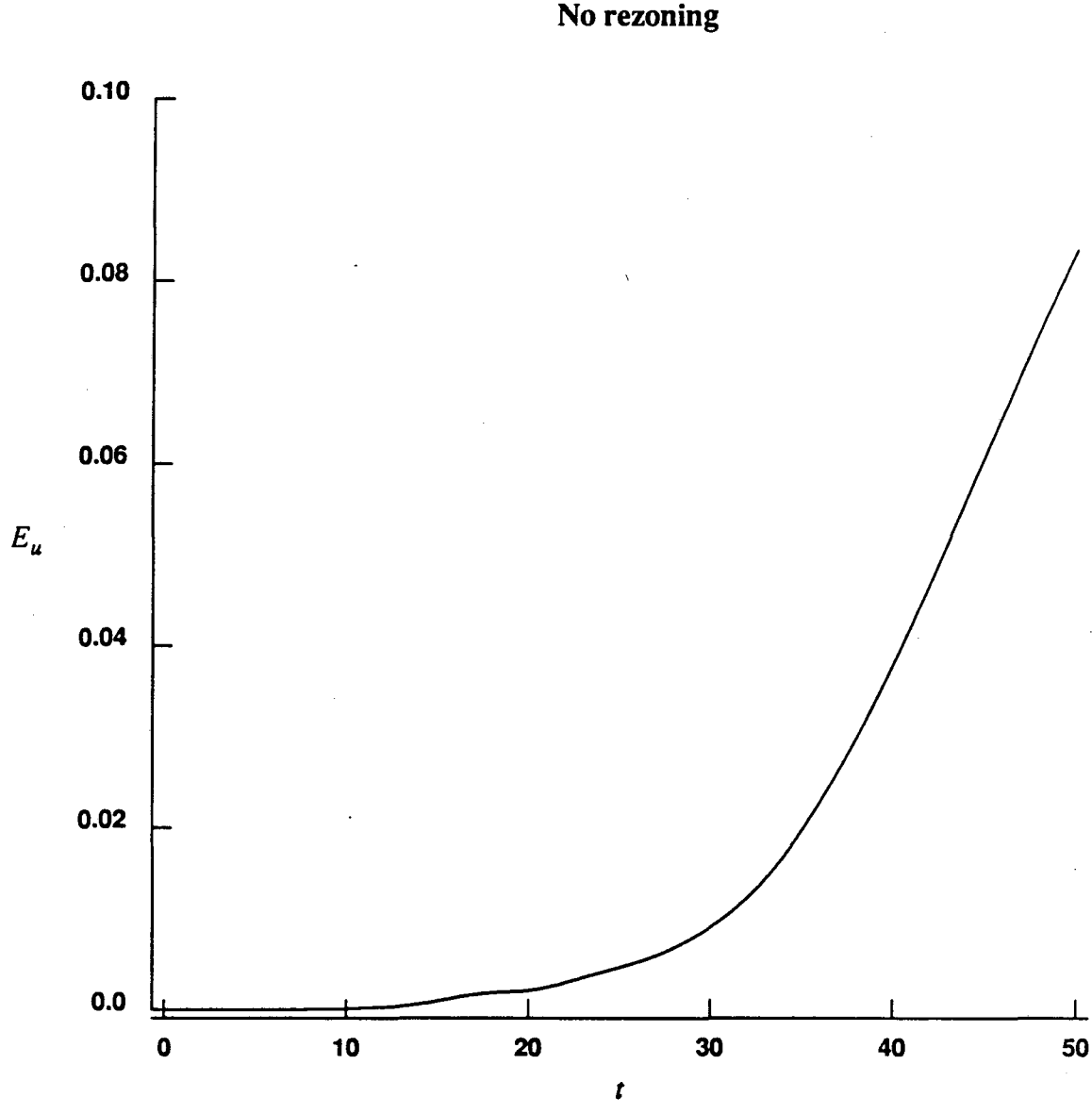


Fig. 6.4 $\omega(r)=(\max(0,1-r^2))^7, h=0.0625, \delta=0.5\sqrt{h}$, $\Delta t=4.0h$.

In test problem #3, the difference in velocity errors between the two cutoff functions is small for h=0.125 , but it increases as h gets smaller. For h=0.05 the error is 9.7 times smaller at time t=0 and 17.1 times smaller at time t=50 for the infinite-order cutoff compared to the eight order cutoff. The rate of convergence is close to 8 for the infinite-order cutoff, but as expected around 4 for the eighth order cutoff. The theoretical rate of convergence for the infinite-order cutoff is 7.25 in this case, since the moment error is of order $O(\delta^{14.5})$, so once again the observed rate of convergence is higher than the theoretical rate. We also made a comparison of rezoning version 1 vs. rezoning version 2 using test problems #1-3 with the infinite order cutoff. The results are shown graphically in figures 6.1-6.3. We see that version 2 gives a significantly lower error, and that the gap between the two versions increases with increasing smoothness of the flow. The sharp peaks in the graphs are due to the fact that sometimes the velocity error increases faster than the vorticity error. Then, after rezoning, the velocity error decreases again. In practice, these peaks do not matter, since the error at any time is much smaller than what is obtained without rezoning as we see in Fig. 6.4.

In the fourth test problem, we distribute the vorticity on two circles according to

$$\omega(x,y) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7 \tag{6.6}$$

Thus, we have two vorticity patches with the vorticity distributed as in test problem #2. Note that this test problem differs from the famous test problems considered by Christiansen [13]. Christiansen [13] used uniform vorticity distribution within the two circles. However, in our test problem the vorticity is concentrated at the centers of the circles and decays to 0 in a smooth fashion as we approach the boundary. The numerical solution using Hald's infinite-order cutoff with rezoning is shown in figures 6.5-6.15. The graphs represent vorticity level sets at different times from time t=0 to time t=100. To estimate the rate of convergence, we have used Richardson's extrapolation with three different gridsizes h, 2/3 h and h/2. Assuming the rate of convergence is q, we can write $\tilde{u}_l = u_l + h^q e(x,y,t) +$ (higher order terms). Then

$$\frac{||\tilde{u}_l^h - \tilde{u}_l^{2h/3}||}{||\tilde{u}_l^{2h/3} - \tilde{u}_l^{h/2}||} \sim \frac{h^q - (2h/3)^q}{(2h/3)^q - (h/2)^q} = \frac{1-(2/3)^q}{(2/3)^q - (1/2)^q} \tag{6.7}$$

The norm is taken to be the discrete L_2 norm of the differences in the computed velocities for vortices with the same initial positions.

TABLE 6.5

Rate of convergence of the velocity approximations in test problem 4 using Hald's infinite order cutoff.

t	*Rate of Convergence*
0.0	3.7
10.0	4.1
20.0	4.3
30.0	4.1
40.0	4.4
50.0	4.4
60.0	4.2
70.0	4.6
80.0	4.5
90.0	4.5
100.0	4.9

$\omega(x,y) = (\max(0 , (0.25-(| x | -0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

Once we have computed the first quotient in (6.7), we set the third quotient equal to this value, and solve for q numerically. Using the three grid-sizes $h=1/10$, $h=1/15$ and $h=1/20$ we obtain the rates of convergence in Table 6.5. We see that the rates of convergence for this problem are similar to the rates observed in problem 2.

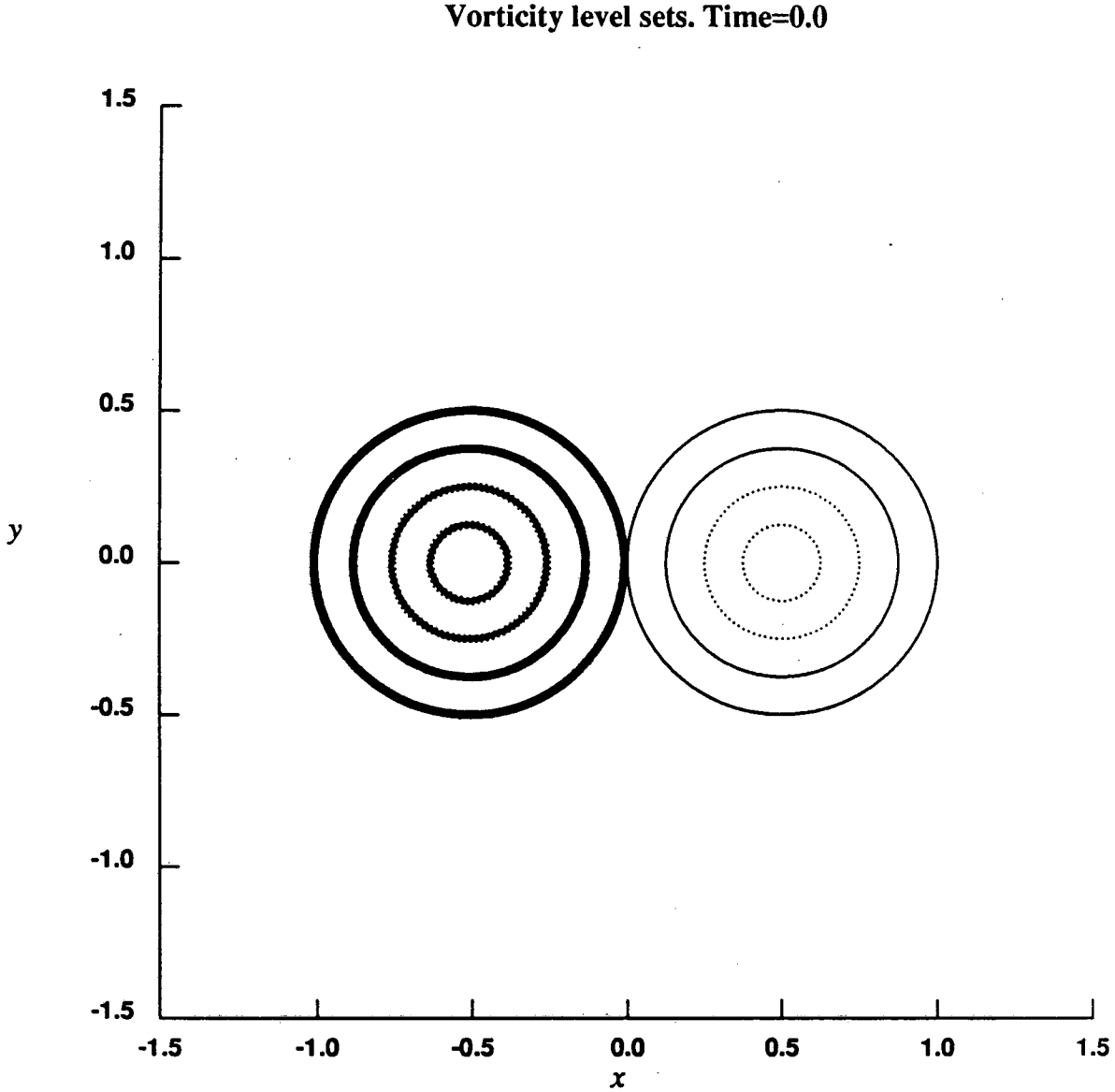


Fig. 6.5. $\omega(x,y,0) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

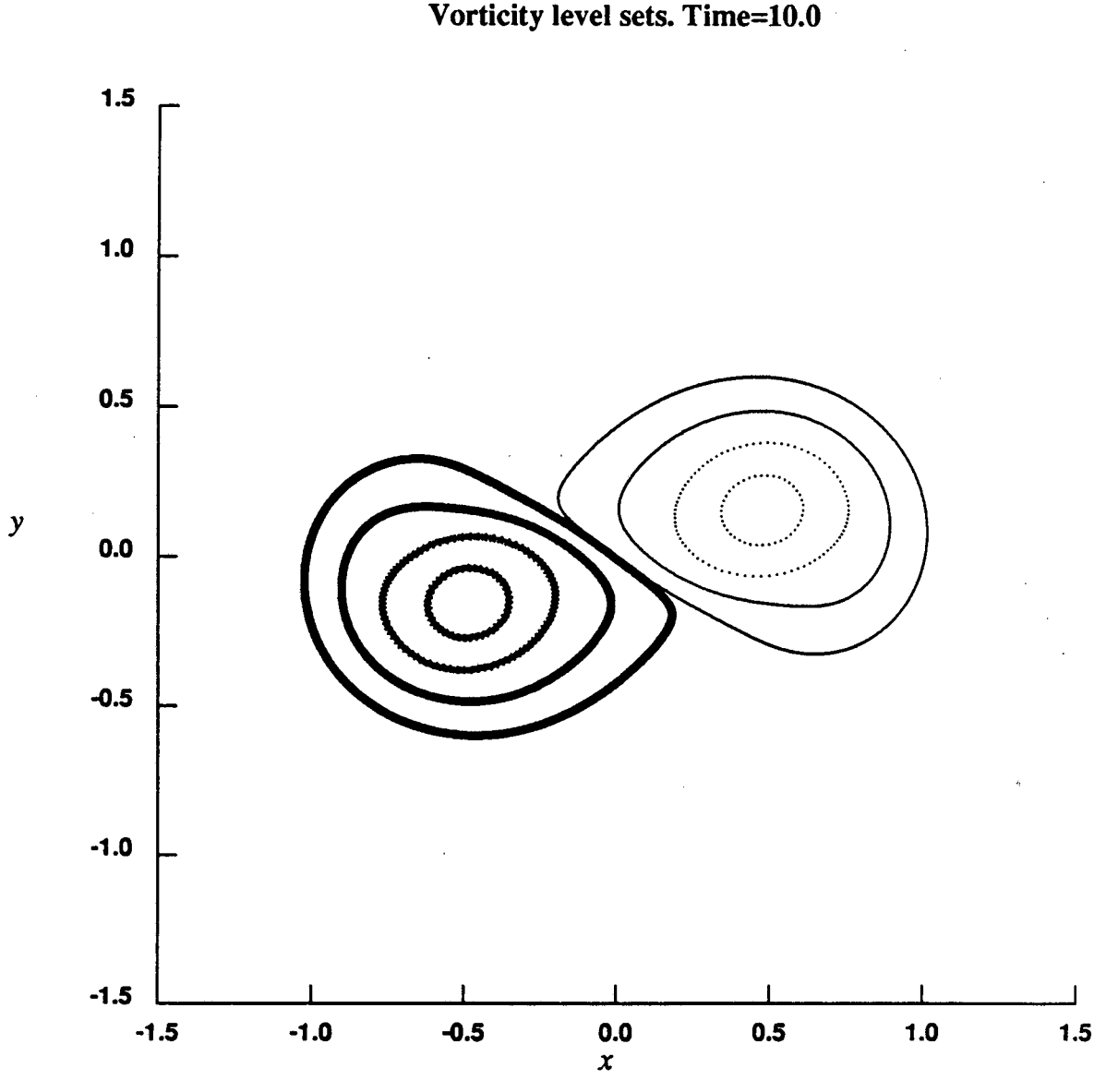


Fig. 6.6. $\omega(x,y,0) = (\max(0 , (0.25-(| x | -0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

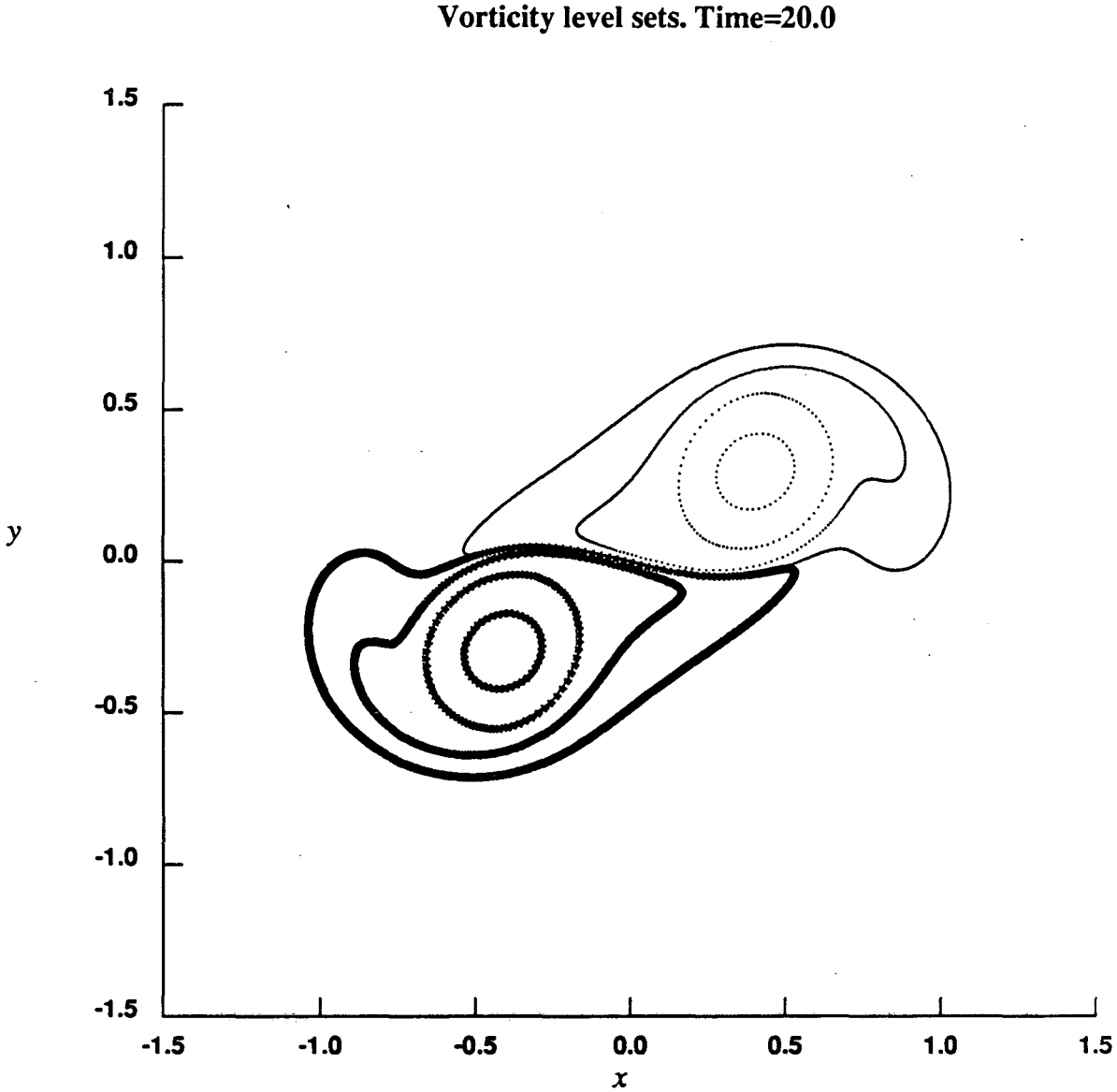


Fig. 6.7. $\omega(x,y,0) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

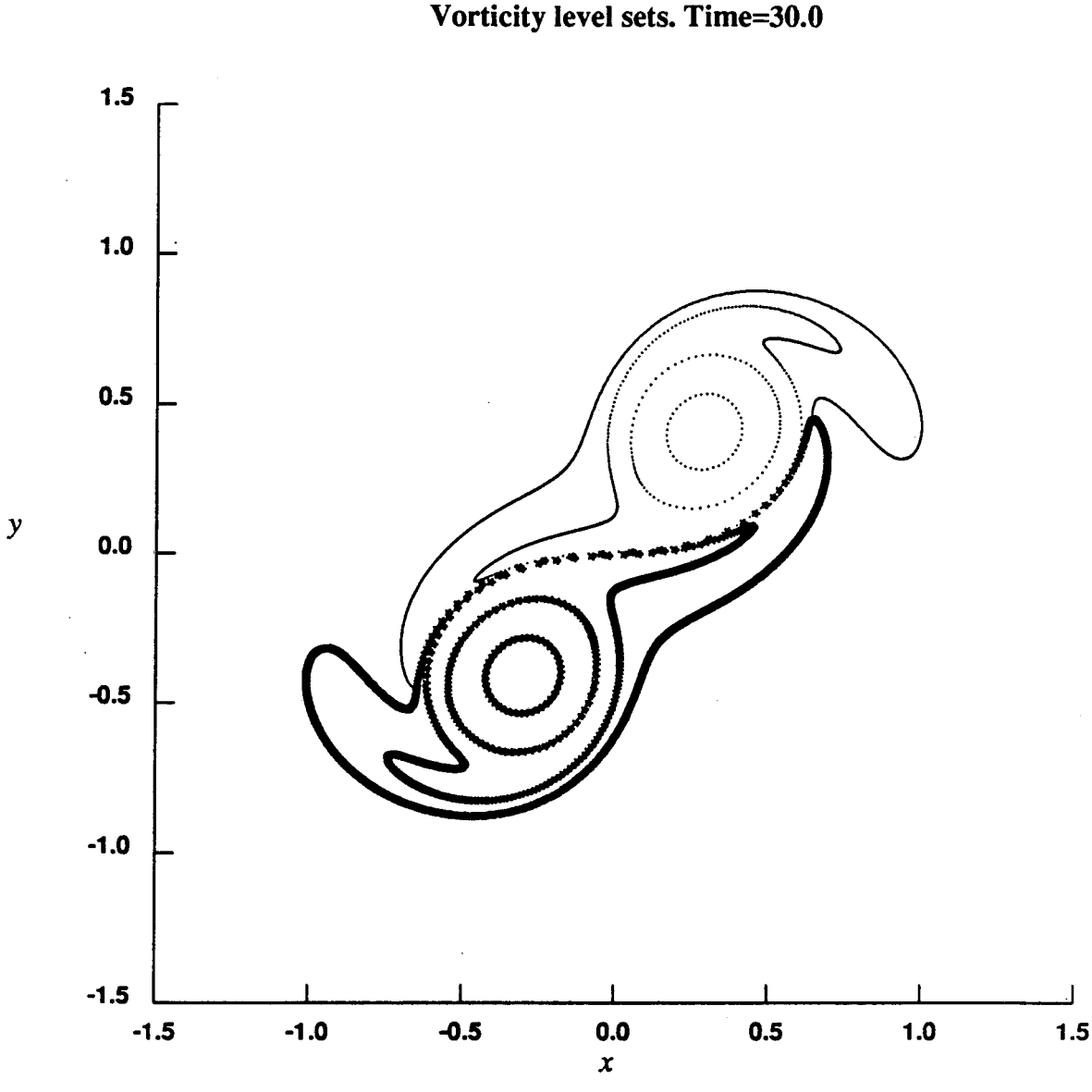


Fig. 6.8. $\omega(x,y,0) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7$,

h=0.0625, δ=0.28$\sqrt{h}$, η=1.25, ε=0.00004h^2, Δt=5.0h.

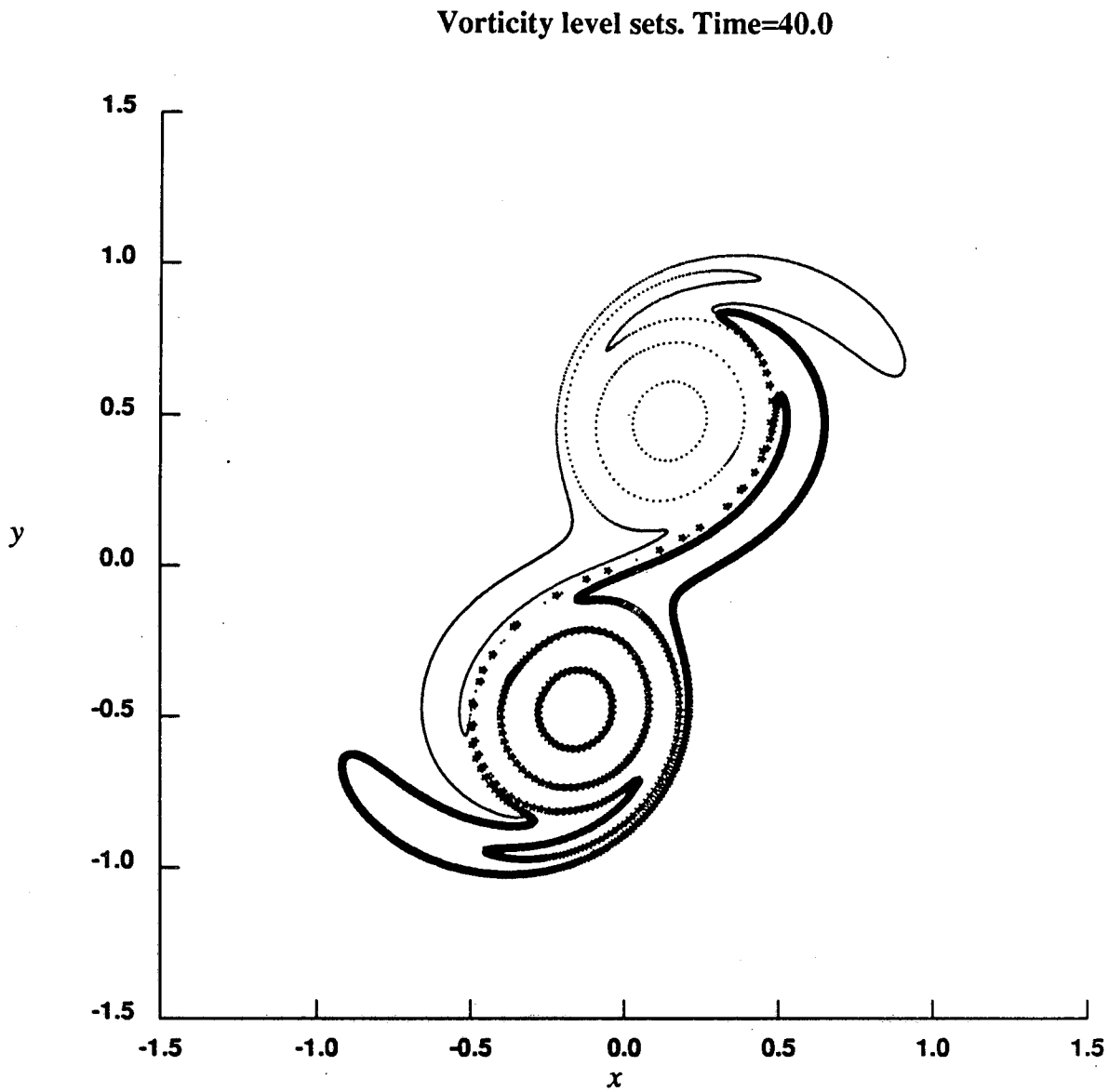


Fig. 6.9. $\omega(x,y,0) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

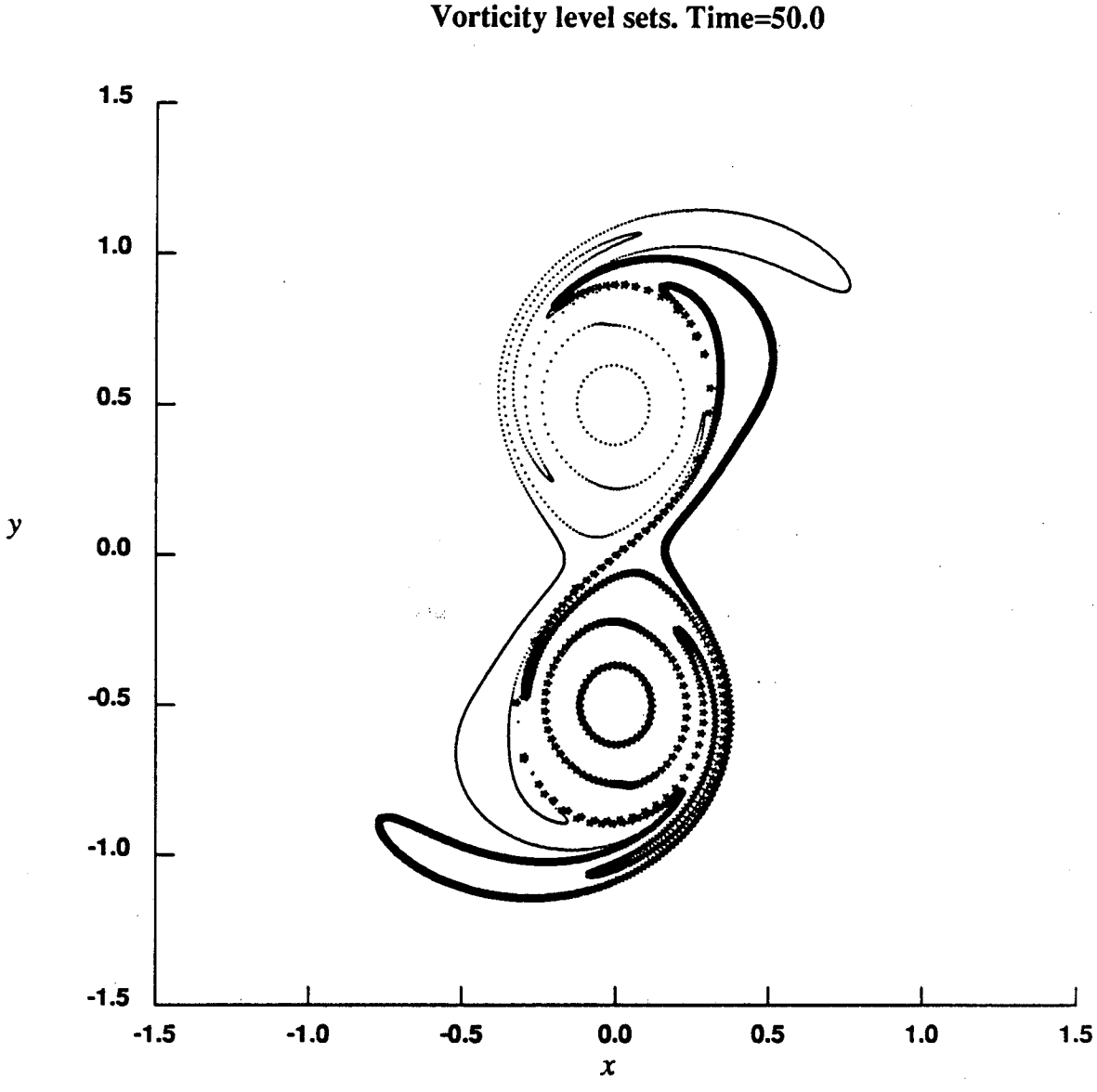


Fig. 6.10. $\omega(x,y,0) = (\max(0 , (0.25-(| x | -0.5)^2-y^2)))^7$,

h=0.0625, δ=0.28$\sqrt{h}$, η=1.25, ε=0.00004h^2, Δt=5.0h.

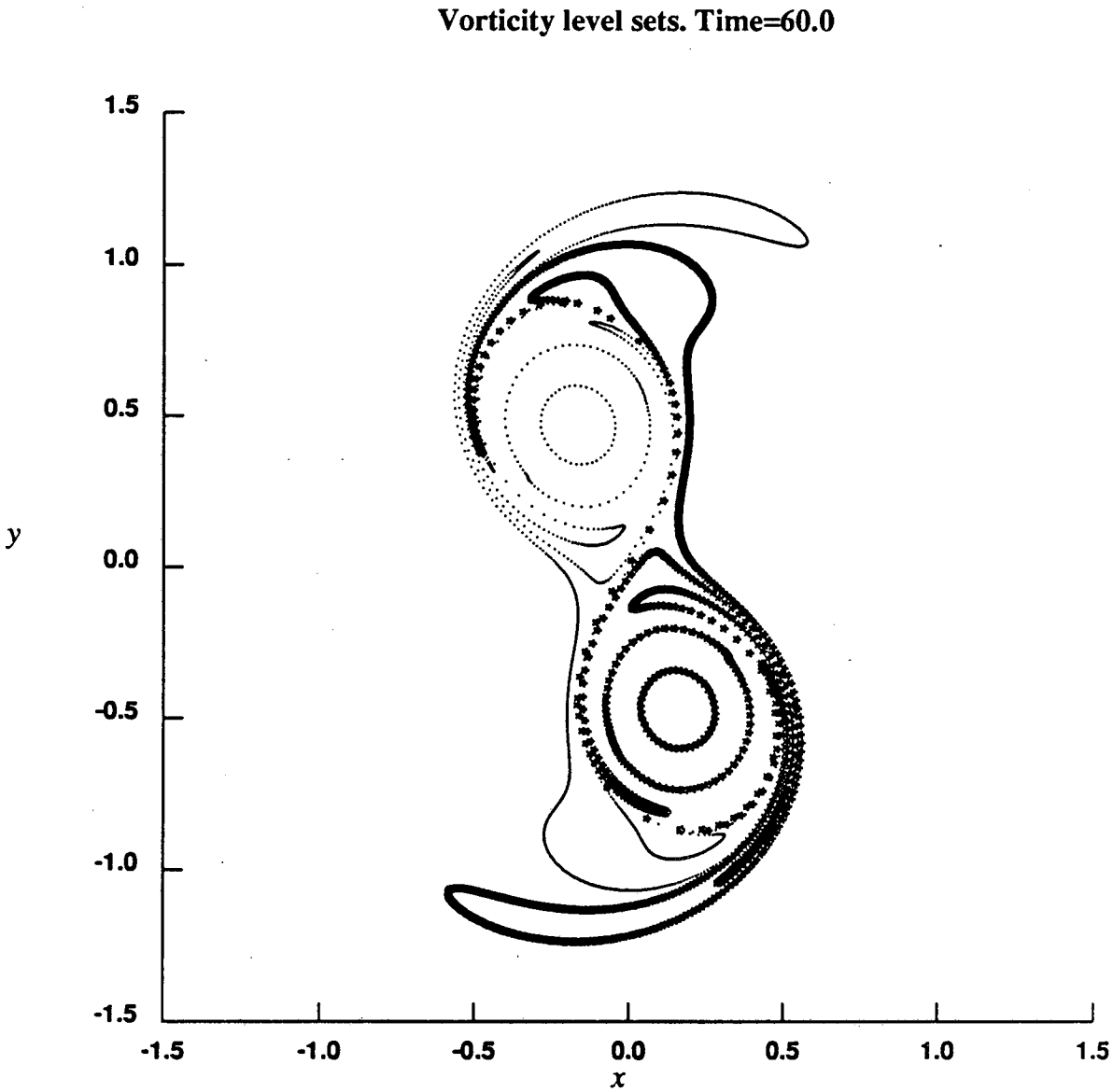


Fig. 6.11. $\omega(x,y,0) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

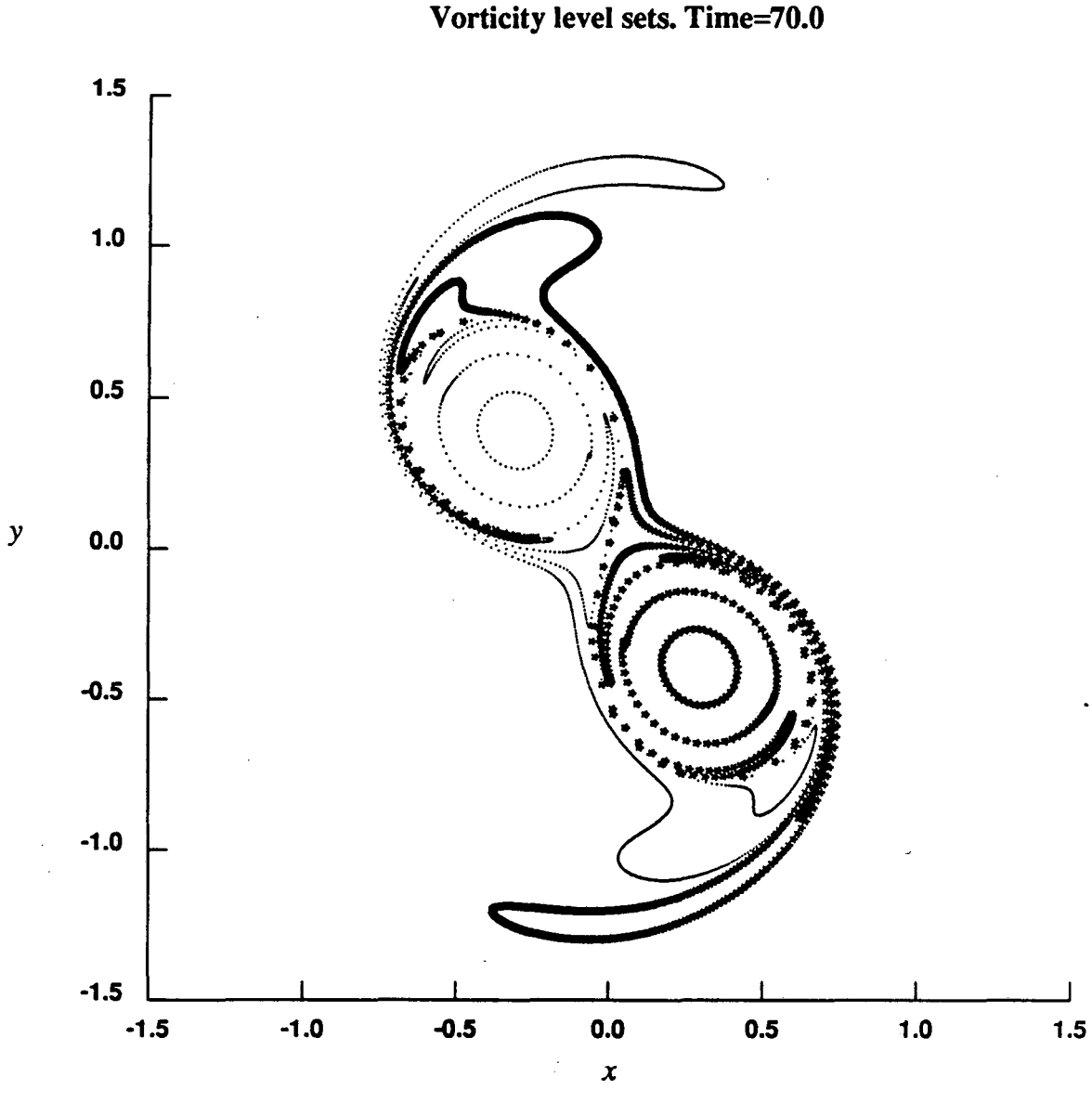


Fig. 6.12. $\omega(x,y,0) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

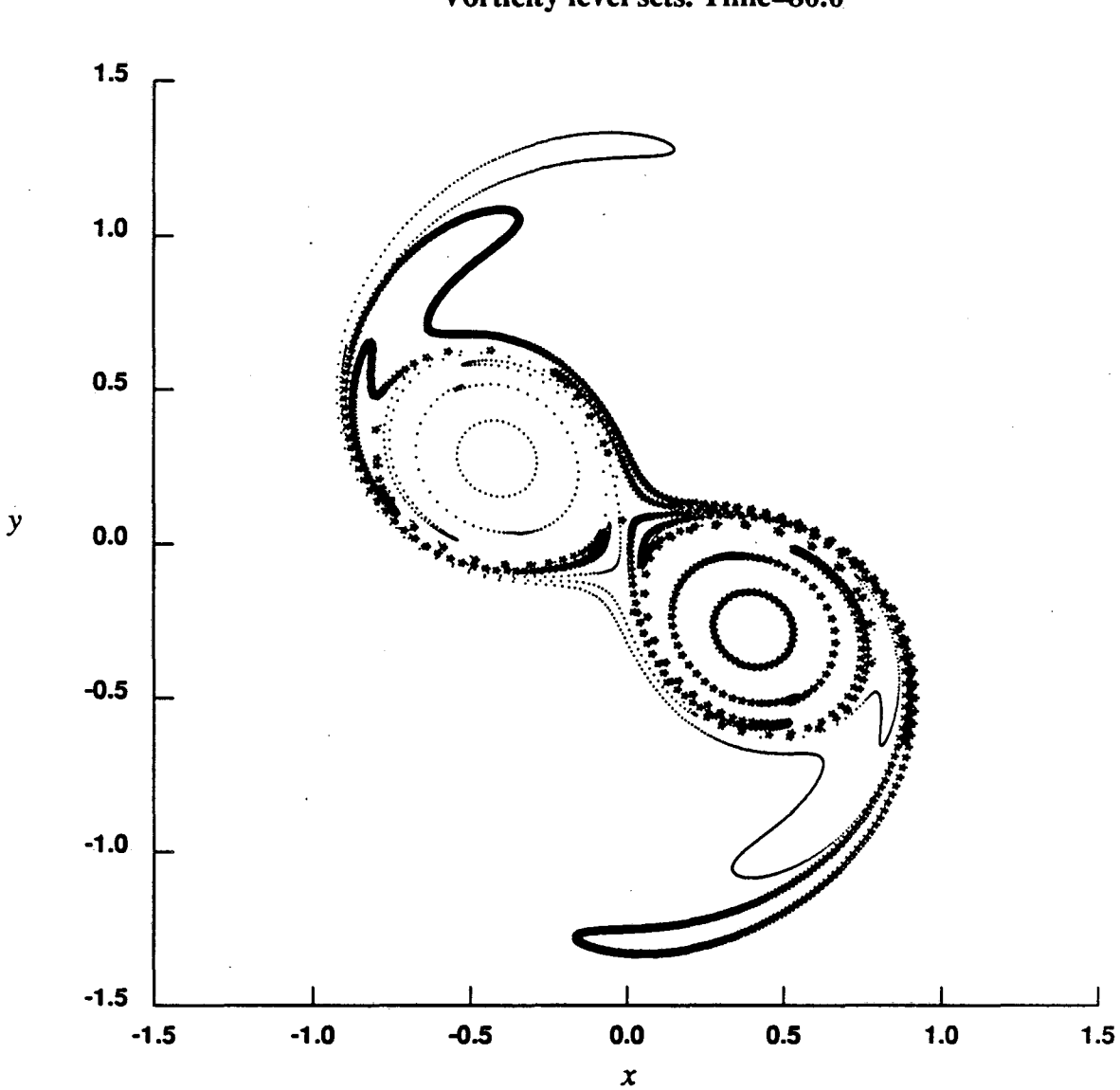


Fig. 6.13. $\omega(x,y,0) = (\max(0, (0.25-(|x|-0.5)^2-y^2)))^7$,

$h=0.0625$, $\delta=0.28\sqrt{h}$, $\eta=1.25$, $\varepsilon=0.00004h^2$, $\Delta t=5.0h$.

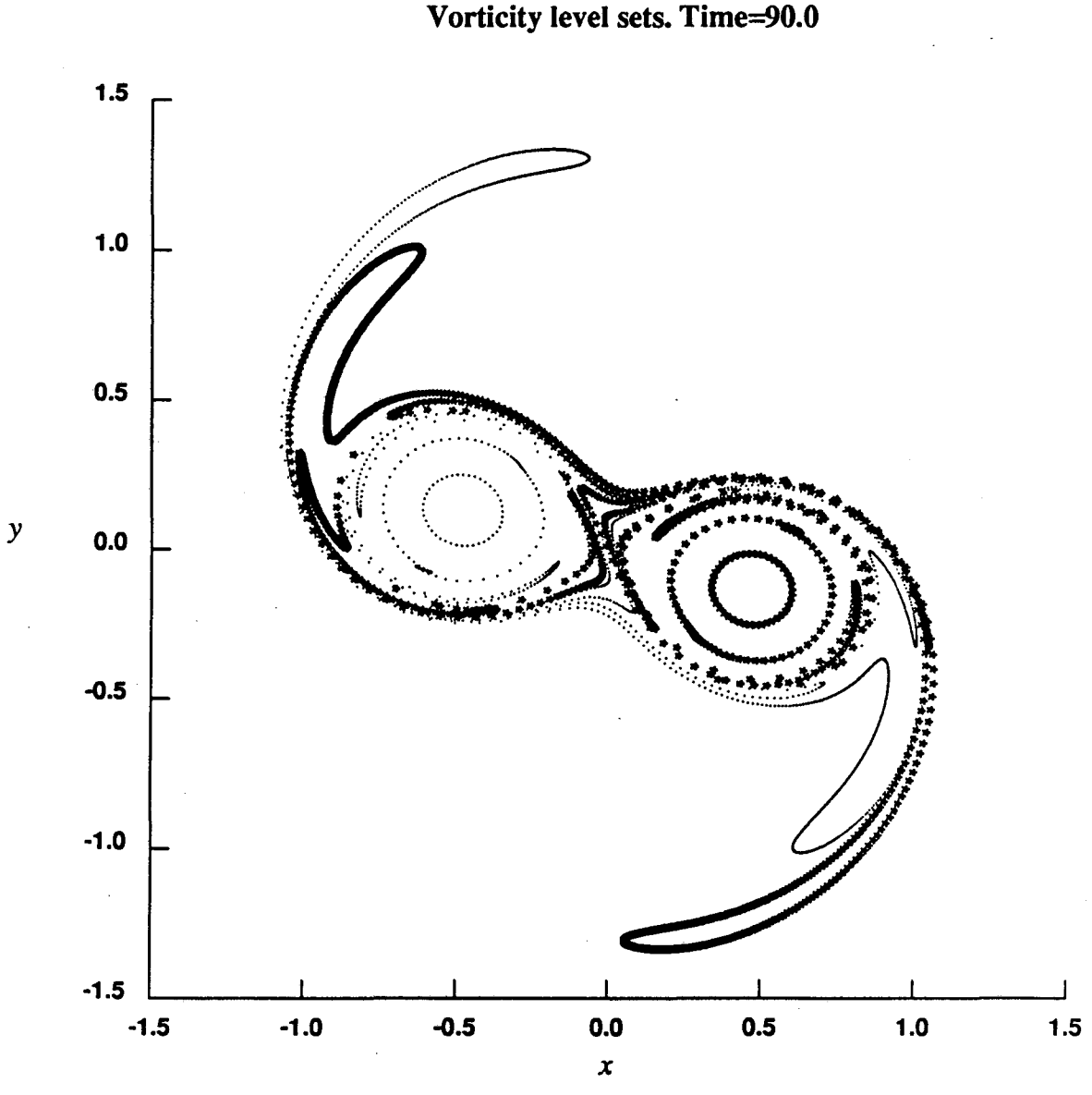


Fig. 6.14. $\omega(x,y,0) = (\max(0 , (0.25-(| x | -0.5)^2-y^2)))^7$,

h=0.0625, δ=0.28$\sqrt{h}$, η=1.25, ε=0.00004h^2, Δt=5.0h.

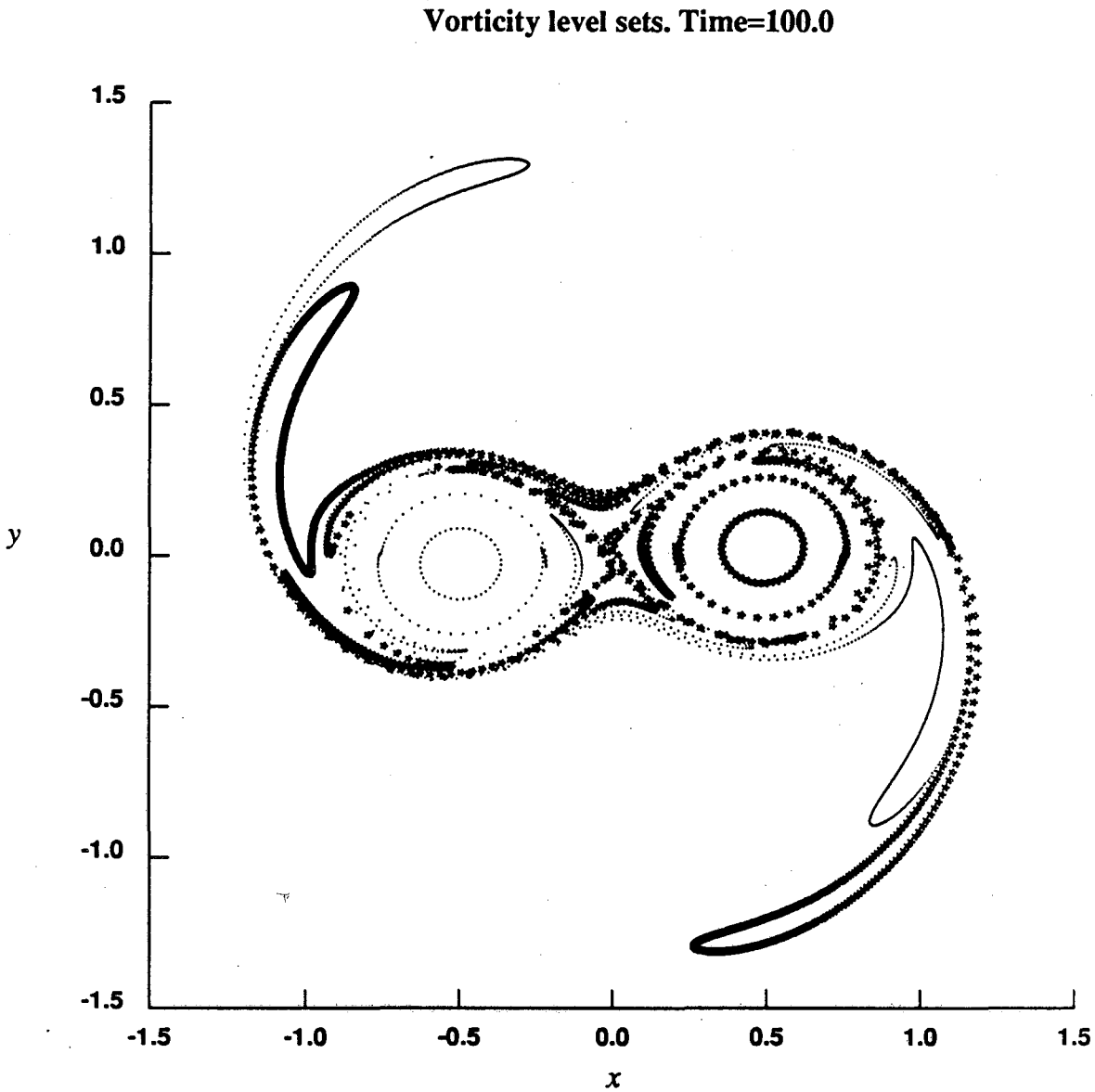


Fig. 6.15. $\omega(x,y,0) = (\max(0 , (0.25-(| x | -0.5)^2-y^2)))^7$,

h=0.0625, δ=0.28$\sqrt{h}$, η=1.25, ε=0.00004h^2, Δt=5.0h.

TABLE 6.6

Rate of convergence of the velocity approximations in test problem 5 using Hald's infinite order cutoff.

t	*Rate of Convergence*
0.0	3.2
10.0	3.2
20.0	3.2
30.0	3.3
40.0	3.3
50.0	3.3

$\omega(x,y) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$,

$h=0.0625$, $\delta=0.6\sqrt{h}$, $\eta=1.5$, $\Delta t=4.0h$.

In the fifth test problem, the initial vorticity is distributed on a square according to

$$\omega(x,y) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7 \tag{6.8}$$

The rates of convergence at different times up to time $t=50$ are estimated in the same way as in test problem #4, using the same three grid-sizes. Here, we had to take a larger value of δ in order to maintain a high rate of convergence up to time $t=50$. The observed rates of convergence are lower than in problems 2 and 4, although the initial vorticity distribution has the same smoothness in this case. It is possible that the Fourier transform of the vorticity distribution has a lower rate of decay in this case, causing a lower rate of convergence. Figures 6.16–6.22 show the computed vorticity level sets at times $t=0,10,20,30,40,50$ and 100.

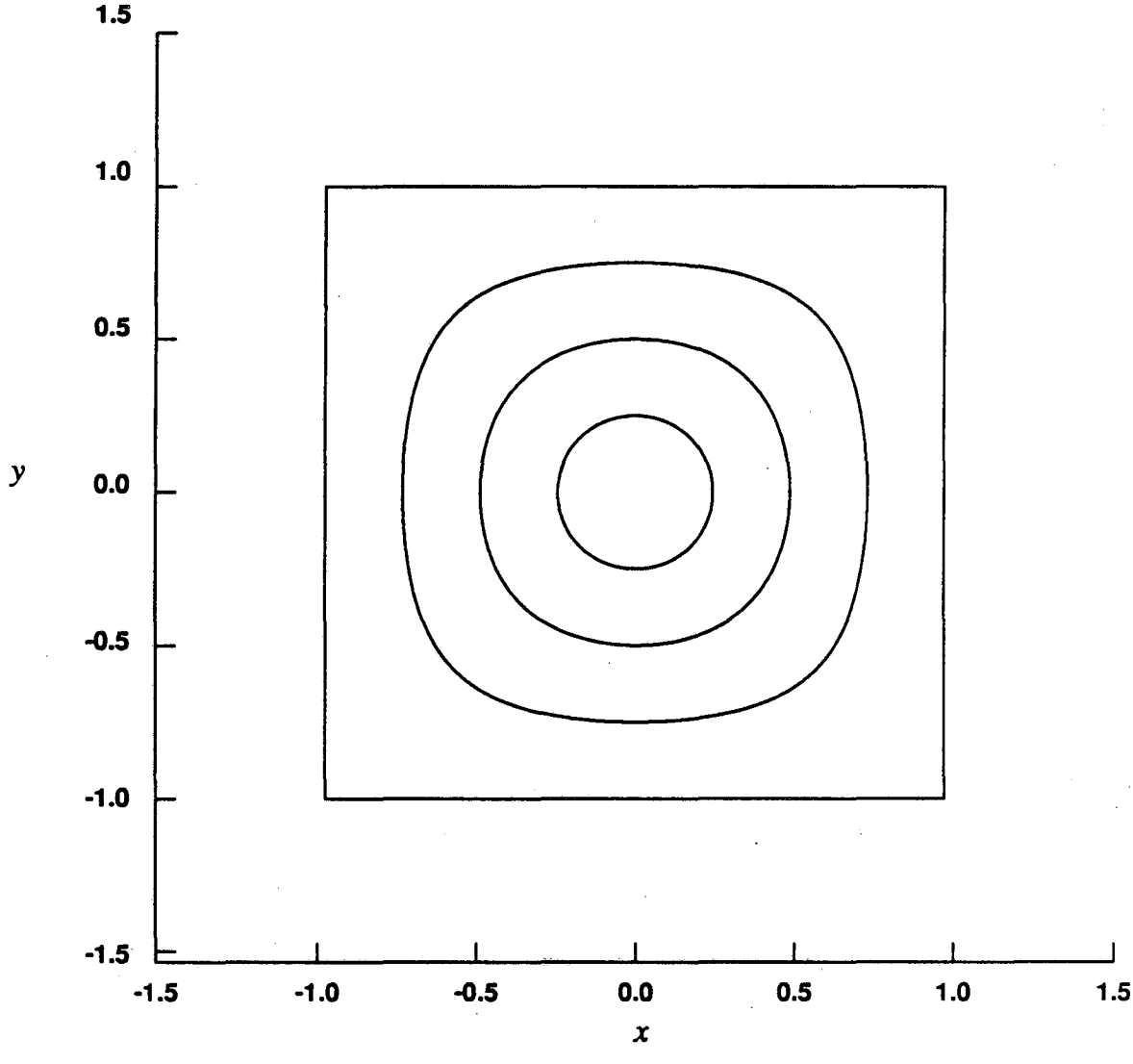


Fig. 6.16. $\omega(x,y,0) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$

h=0.0625, δ=0.355$\sqrt{h}$, η=1.5, Δt=4.0h.

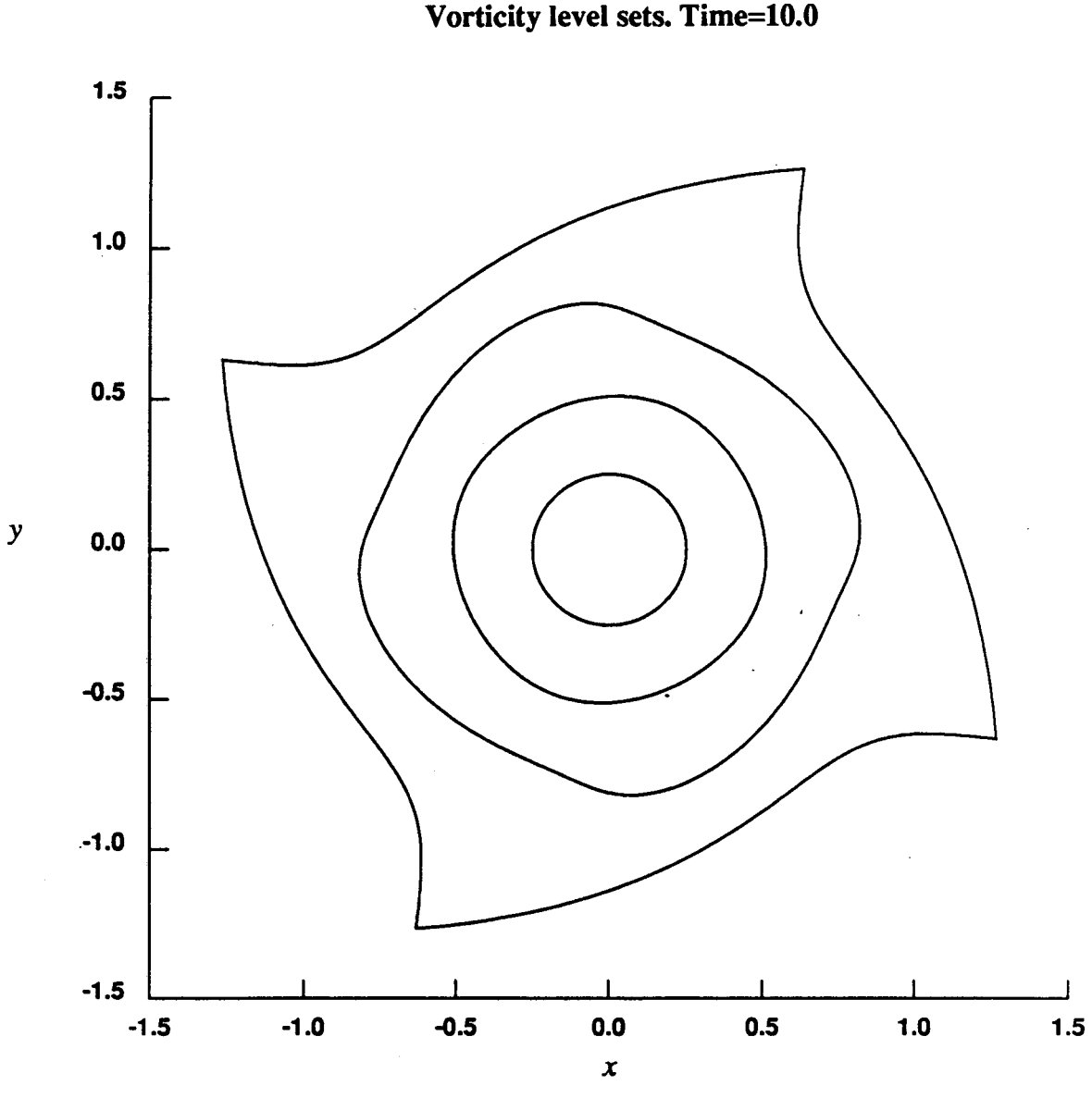


Fig. 6.17. $\omega(x,y,0) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$

h=0.0625, δ=0.355$\sqrt{h}$, η=1.5, Δt=4.0h.

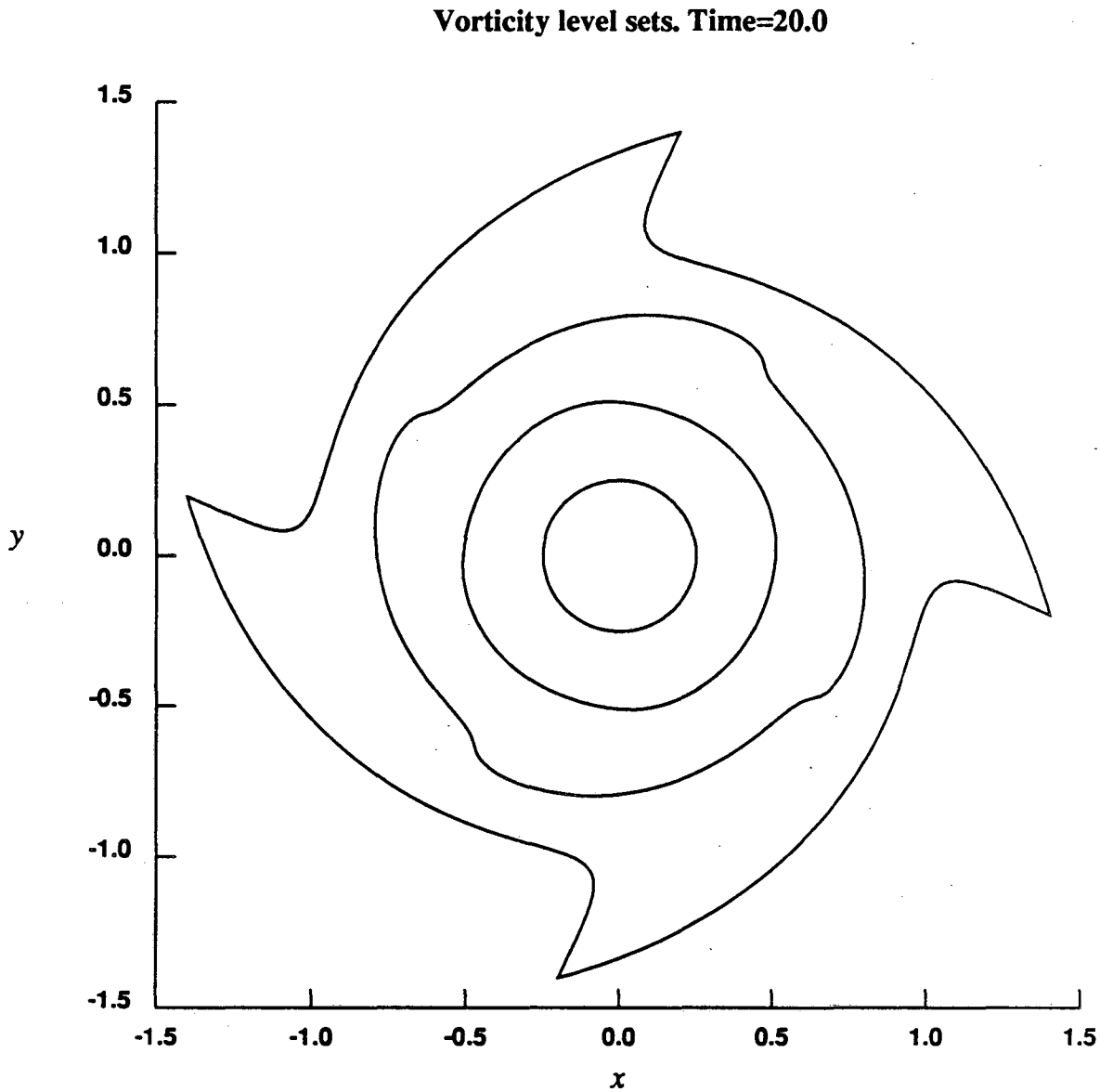


Fig. 6.18. $\omega(x,y,0) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$

h=0.0625, δ=0.355$\sqrt{h}$, η=1.5, Δt=4.0h.

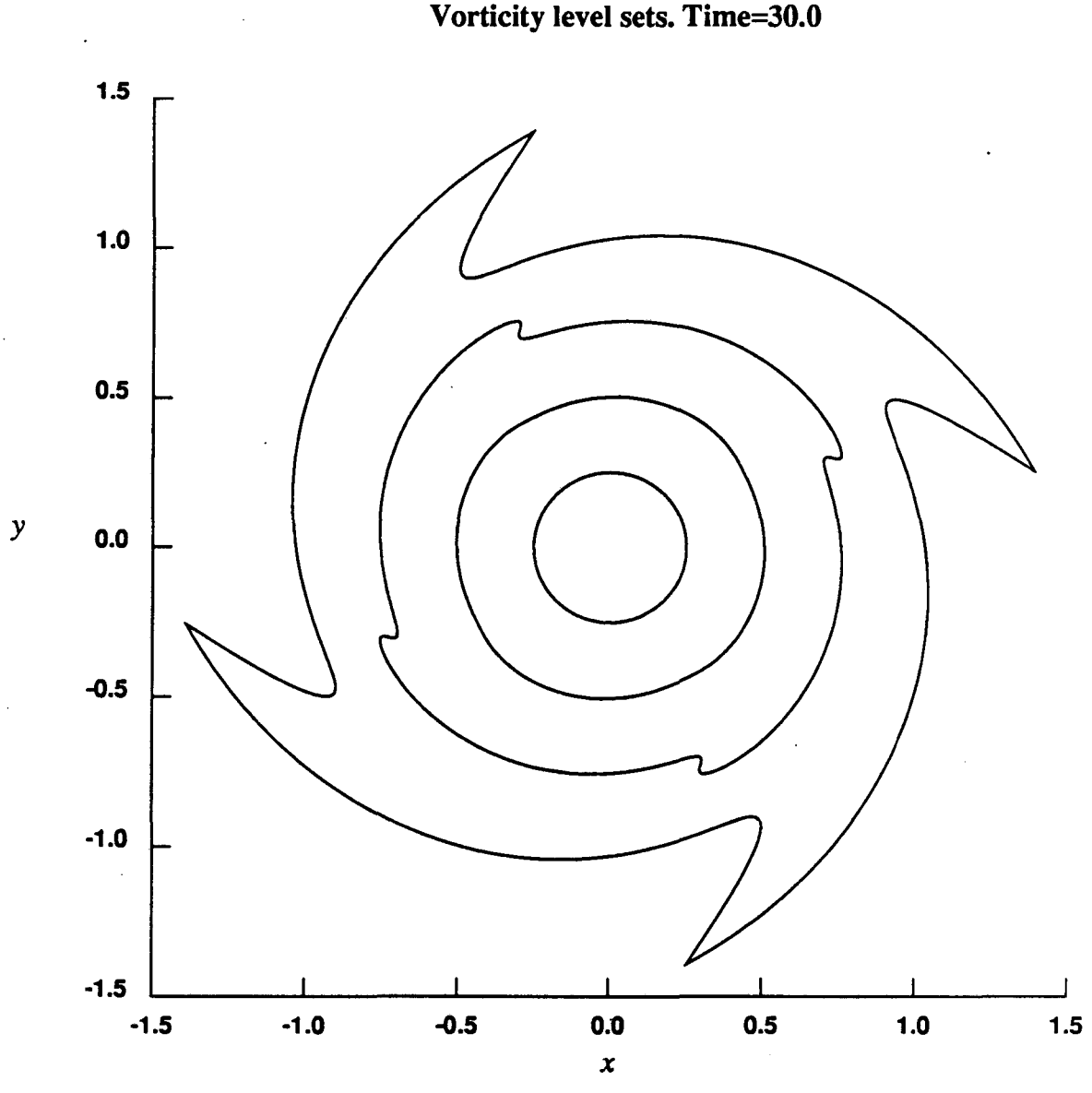


Fig. 6.19. $\omega(x,y,0) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$

h=0.0625, δ=0.355$\sqrt{h}$, η=1.5, Δt=4.0h.

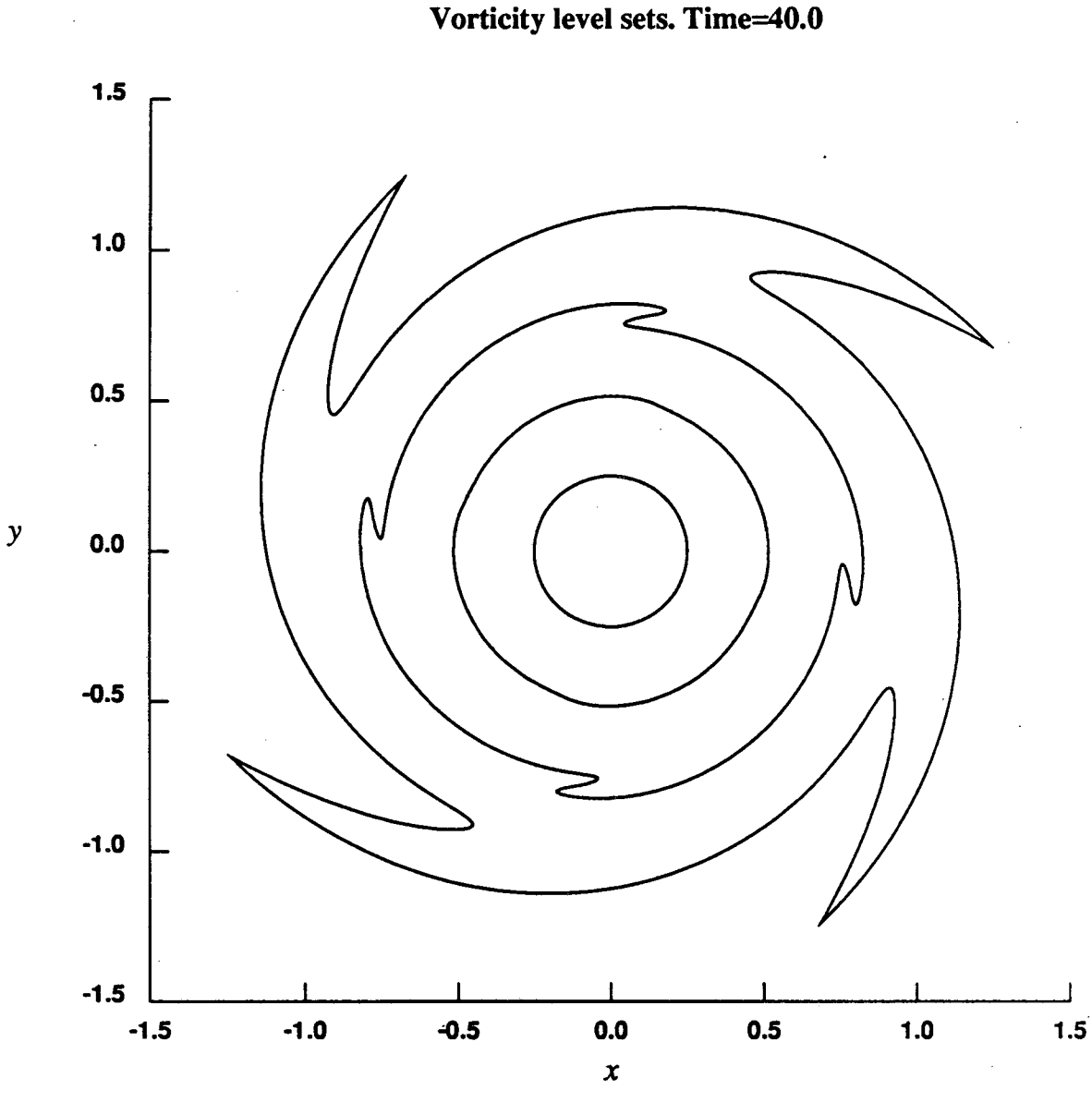


Fig. 6.20. $\omega(x,y,0) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$

$h=0.0625$, $\delta=0.355\sqrt{h}$, $\eta=1.5$, $\Delta t=4.0h$.

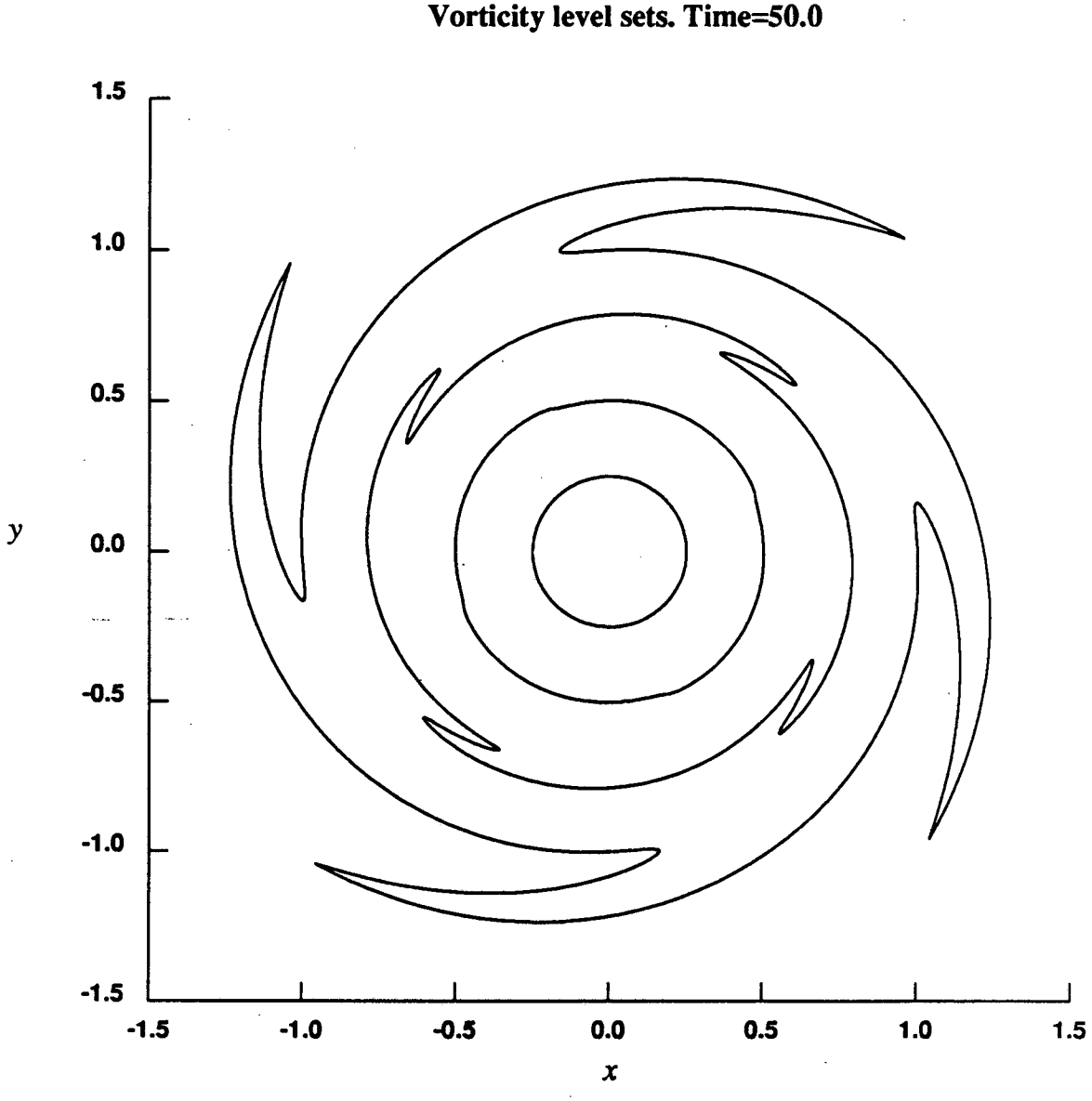


Fig. 6.21. $\omega(x,y,0) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$

h=0.0625, δ=0.355$\sqrt{h}$, η=1.5, Δt=4.0h.

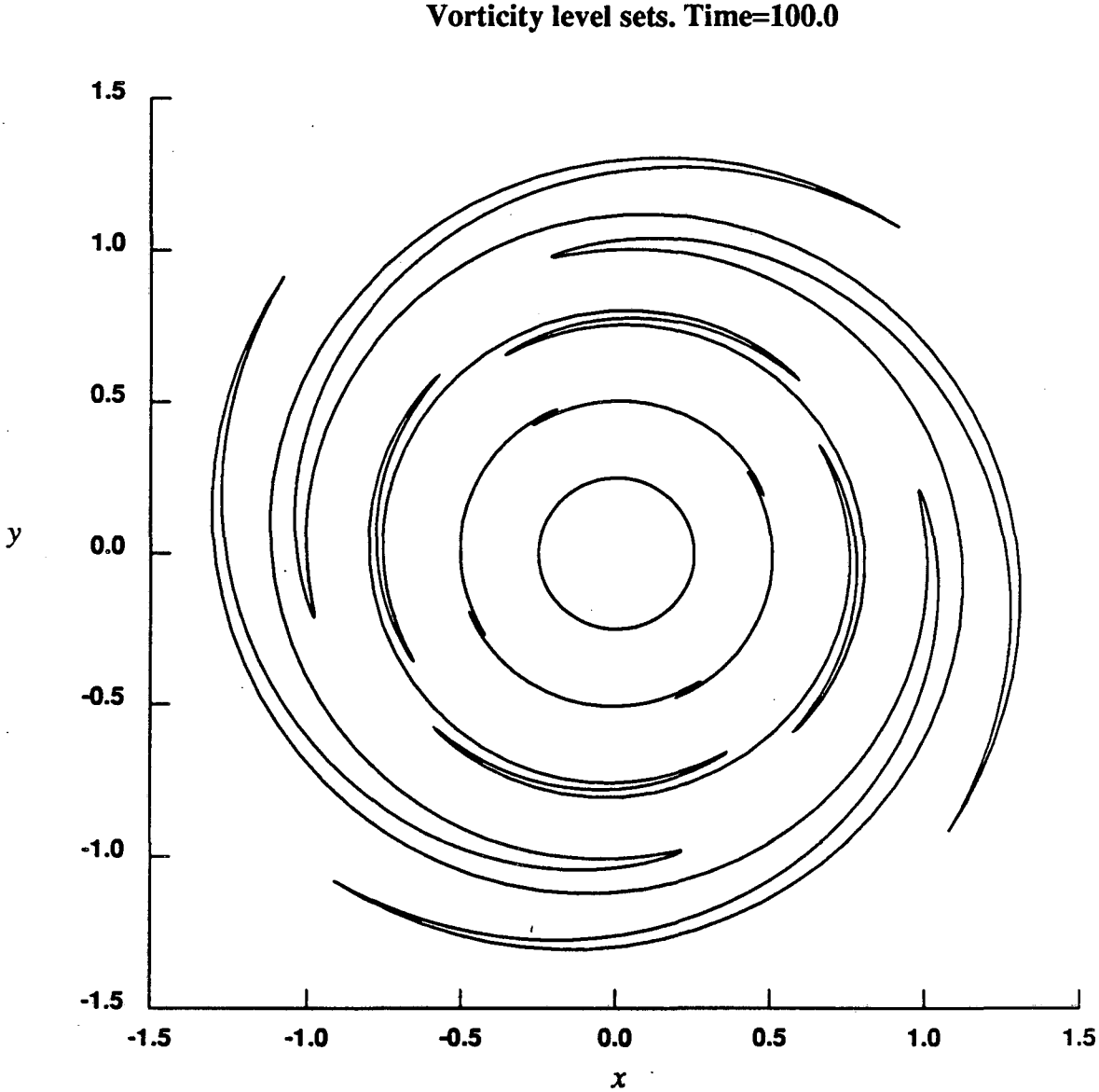


Fig. 6.22. $\omega(x,y,0) = \left[(\max(0, 1-x^2))\,(\max(0, 1-y^2)) \right]^7$

h=0.0625, δ=0.355$\sqrt{h}$, η=1.5, Δt=4.0h.

TABLE 6.7

A comparison between the direct method and the Rokhlin-Greengard method.

		Direct method		*Rokhlin–Greengard method*	
h	N	E_u	CPU time	E_u	CPU time
1/20	1257	$0.3576 \cdot 10^{-6}$	3.7	$0.3576 \cdot 10^{-6}$	4.3
1/40	5025	$0.2507 \cdot 10^{-7}$	63.5	$0.2507 \cdot 10^{-7}$	45.8
1/64	12853	$0.4021 \cdot 10^{-8}$	394.6	$0.4021 \cdot 10^{-8}$	106.9

Finally, we made a comparison between the direct method of evaluating the sum in (1.11) and the Rokhlin-Greengard algorithm [23]. For this, we used the vorticity distribution of test problem #2, the eighth order cutoff function, and $\delta = 1.7\sqrt{h}$. The number of terms in the multipole expansion, see [23], was set to 20. The results are summarized in table 6.7. Here, N stands for the number of vortices, E_u for the velocity error at time t=0, and the CPU time is given in minutes for one velocity evaluation at time t=0 on a VAX computer. We have to emphasize that the speed of the Rokhlin-Greengard algorithm applied to vortex methods is limited by the size of the cutoff parameter δ. In fact, the maximum number of levels of refinement, see [23], must not exceed $1 - \log_2(\delta/l)$, where l is the length of one side of the computational box. Here we have used l=2, since this is exactly what we need to cover the support of the initial vorticity distribution in test problem #2, but we have to admit that the speed of this algorithm will increase somewhat if we pick l so that $1 - \log_2(\delta/l)$ is exactly an integer, and set the maximum number of levels of refinement equal to this value. The Rokhlin-Greengard algorithm would also run faster if we choose δ smaller, but that would force us to use a lower order cutoff function, which we do not believe is such a good idea for smooth flows. However, if the flow is not very smooth, we may very well use a lower order cutoff function, a smaller δ , and use the Rokhlin-Greengard algorithm with maximum efficiency.

Appendix

To make it easier for the reader who wishes to use infinite-order vortex methods, we supply two subroutines. The first subroutine finds the coefficients of the polynomials which approximate Hald's infinite-order cutoff function and velocity kernel as described in chapter 3. The second subroutine evaluates the sum in equation (1.11) or equation (5.9) using these polynomials. If the second subroutine is used on a Cray computer, we recommend not changing the structure of the subroutine since that would probably make it more expensive. For example, if we express Horner's rule as a loop, the CPU time requirement increases by a factor of about 4 on a Cray computer and most of the overall computational work of the vortex method is done in this subroutine. We tested many different versions of this subroutine on a Cray computer and found that this version was by far the fastest one. This is the main reason for presenting it here. However, if a VAX computer is used, the code may be simplified without decreasing the computational speed. We do not present our rezoning subroutine here, but it has the same structure as the second subroutine.

```
c       Program for finding polynomials approximating
c       Hald's infinite order cutoff function and
c       velocity kernel. (Example 1, chap.2 )

        parameter(m=120)
        common/prmtr/k
        double precision c(0:m,1:10)
        double precision d(0:m,1:13)

        do 2 k=1,m
        call coeffs(c,d,m)
2       continue
        end

        subroutine coeffs(c,d,m)

c |-------------------------------------------------------------------|
c |                                                                   |
c |  This subroutine finds the best polynomial approximations         |
c |  in different intervals                                           |
c |  to  Hald's infinite-order cutoff function (example 1)            |
c |  and to the corresponding scaling function F that is related to   |
c |  the velocity kernel.                                             |
c |                                                                   |
c |  List of variables:                                               |
c |                                                                   |
c |  Input:                                                           |
c |                                                                   |
c |  m            = the number of intervals in which we wish to       |
c |                   approximate F and PSI.                          |
c |                                                                   |
c |  k            = the center of the interval under consideration    |
c |                                                                   |
c |                                                                   |
c |  Output:                                                          |
c |                                                                   |
c |  c(0:m,1:10)  = array of coefficients of the polynomials          |
c |                   approximating F in all intervals.               |
c |                                                                   |
c |  d(0:m,1:10)  = array of coefficients of the polynomials          |
c |                   approximating the cutoff function PSI           |
c |                   in all intervals.                               |
c |                                                                   |
c |  Library functions:                                               |
c |                                                                   |
c |  mmbsj0       = Bessel function of order 0                        |
c |                                                                   |
c |  mmbsj1       = Bessel function of order 1                        |
c |                                                                   |
c |  Local functions:                                                 |
c |                                                                   |
c |  f2           = Bessel function of order 2                        |
c |                                                                   |
c |  f3           = Bessel function of order 3                        |
c |                                                                   |
c |-------------------------------------------------------------------|
```

```
c |-------------------------------------------------------------------|
c |                                                                   |
c |  Input to library subroutine iratcu:                              |
c |                                                                   |
c |  l             = degree of the polynomial in the numerator of     |
c |                  the approximating rational function.             |
c |                                                                   |
c |  m1            = degree of the polynomial in the denominator of   |
c |                  the approximating rational function.             |
c |                  Here we always let m1=0.                         |
c |                                                                   |
c |  a,b           = endpoints of the interval under consideration.   |
c |                                                                   |
c |  f             = F                                                |
c |                                                                   |
c |  psi           = PSI                                              |
c |                                                                   |
c |  g             = weight function, which is identically 1 here.    |
c |                                                                   |
c |  phi(x)        = x**2-k**2                                        |
c |                                                                   |
c |  We express the best polynomial approximation as a polynomial     |
c |  in phi(x).                                                       |
c |                                                                   |
c |                                                                   |
c |  Output from library subroutine iratcu:                           |
c |                                                                   |
c |  p(13)         = vector of coefficients of the polynomial in the  |
c |                  numerator of the approximating rational function.|
c |                                                                   |
c |  q(1)          = vector of coefficients of the polynomial in the  |
c |                  denominator of the approximating rational        |
c |                  function. In this case q(1) is identically 1. .  |
c |                                                                   |
c |  wk(315)       = "work vector" needed by the library subroutine   |
c |                  iratcu. wk(1) gives the maximum error in the     |
c |                  approximation in the interval [a,b]              |
c |                                                                   |
c | ier            = error parameter required by subroutine           |
c |                  iratcu. See IMSL manual vol.2                    |
c |                                                                   |
c |-------------------------------------------------------------------|

      common/prmtr/k
      integer l,m1,ier
      double precision p(13),q(1),wk(315),a,b
      double precision f3,psi,f,f2,phi,g,mmbsj1,mmbsj0
      double precision pi,c(0:m,1:10),d(0:m,1:13)
      external f,phi,g,f2,f3,psi

      pi=dacos(-1.0d0)
      a=dble(k)-0.5
      b=dble(k)+0.5
      l=9
```

```
      if(k.eq.1)  l=7
      if(k.gt.10) l=7
      if(k.gt.20) l=6
      if(k.gt.30) l=5
      if(k.gt.40) l=4
      if(k.gt.80) l=2
      call iratcu(f,phi,g,a,b,l,m1,p,q,wk,ier)
      print *,wk(1)

      do 4 i=1,l+1
      c(k,i)=p(i)
      write(15,32) k,i,c(k,i)
32    format(1H ,'       c(',I3,',',I2,')=',D22.15)
4     continue

      l=12
      if(k.eq.1)  l=9
      if(k.eq.2)  l=10
      if(k.gt.10) l=11
      if(k.gt.20) l=10
      if(k.gt.30) l=9
      if(k.gt.50) l=8
      if(k.gt.90) l=7
      call iratcu(psi,phi,g,a,b,l,m1,p,q,wk,ier)
      print *,wk(1)

      do 41 i=1,l+1
      d(k,i)=p(i)
      write(15,33) k,i,d(k,i)
33    format(1H ,'       d(',I3,',',I2,')=',D22.15)
41    continue
      end


      double precision  function f2(x)
      double precision x,mmbsj1,mmbsj0
      integer ier2

c     The Bessel function of order 2 is expressed in terms of the
c     Bessel functions of orders 0 and 1

      f2=2.0*mmbsj1(x,ier2)/x-mmbsj0(x,ier2)
      return
      end


      double precision function f3(x)
      double precision x,mmbsj1,mmbsj0
      integer i

c     The Bessel function of order 3 is expressed in terms of the
c     Bessel functions of orders 0 and 1

      f3=(8.0/x**2-1.0)*mmbsj1(x,i)-4.0*mmbsj0(x,i)/x
      return
      end
```

```
      double precision function f(x)
      double precision x,pi,f2

      pi=dacos(-1.0d0)
      f=(.5-4.*(4.*f2(4.*x)-5.*f2(2.*x)+f2(x))/(45.*x**2))/(pi*x**2)
      return
      end


      double precision function psi(x)
      double precision x,pi,f3

      pi=dacos(-1.0d0)
      psi=(6.4*f3(4.0*x)-4.0*f3(2.0*x)+0.4*f3(x))/(4.5*pi*x**3)
      return
      end


      double precision function phi(x)
      double precision x
      common/prmtr/k

c  the approximating polynomial is given as a polynomial in (x**2-k**2)

      phi=x**2-dble(k**2)
      return
      end


      double precision function g(x)
      double precision x

c  "g" is a weight function which must be specified.

      g=1.0d0
      return
      end
```

```
c |-------------------------------------------------------------------|
c |                                                                   |
c |  This subroutine evaluates the velocity at a point (xi,yi)        |
c |  according to equation (1.11) or equation (5.9)                   |
c |  using infinite order velocity kernel approximated by             |
c |  a collection of polynomials, as described in chapter 3.          |
c |                                                                   |
c |  List of variables:                                               |
c |                                                                   |
c |  Input:                                                           |
c |                                                                   |
c |  xi,yi    = point at which we wish to compute velocity            |
c |                                                                   |
c |  nn       = max. number of linear sub-divisions between 0 and 1   |
c |                                                                   |
c |  x(1:4*nn**2), y(1:4*nn**2) =                                     |
c |             array of vortices inducing the velocity field         |
c |                                                                   |
c |  tot      = total number of vortices                              |
c |                                                                   |
c |  cc(1:4*nn**2) =                                                  |
c |             vorticity coefficients                                |
c |                                                                   |
c |  x4       = the cutoff parameter delta                            |
c |                                                                   |
c |  c(0:m,1:10) =                                                    |
c |             the set of coefficients of the polynomials            |
c |             approximating an infinite order velocity kernel       |
c |                                                                   |
c |  Output:                                                          |
c |                                                                   |
c |  u,v      = the sum in (1.11) or (5.9)                            |
c |                                                                   |
c |  Local:                                                           |
c |                                                                   |
c |  m        = number of approximating polynomials -1                |
c |                                                                   |
c |  NMAX     = same as nn, but for local use                         |
c |                                                                   |
c |  x5       = delta squared                                         |
c |                                                                   |
c |  r2       = the square of the distance from the point (xi,yi)      |
c |             to a particular vortex                                |
c |                                                                   |
c |  arg(1:4*NMAX**2) =                                               |
c |             the set of distances from (xi,yi) to all vortices     |
c |             divided by delta squared                              |
c |                                                                   |
c |  i1(1:4*NMAX**2)  =                                               |
c |             the set of indices for the polynomials                |
c |             used to approximate the velocity kernel               |
c |                                                                   |
c |  c2(0:4*NMAX**2,1:10) =                                           |
c |             a renaming of the polynomial coefficients             |
c |                                                                   |
c |  x3,k3,term1,term2   =  temporary variables                       |
c |                                                                   |
c |-------------------------------------------------------------------|
```

```
      subroutine sum(c,xi,yi,x,y,tot,cc,nn,u,v,x4)

      integer m,NMAX
      parameter(m=120)
      parameter (NMAX = 40)
      double precision u,v,term1,term2,xi,yi,r2,x3,x4,x5
      double precision x(1:4*nn**2),y(1:4*nn**2),cc(1:4*nn**2)
      double precision arg(1:4*NMAX**2),c(0:m,1:10)
      double precision c2(0:4*NMAX**2,1:10)
      integer i1(1:4*NMAX**2), k3, k, tot

      x5=x4**2
      u=0.0d0
      v=0.0d0

      do 1 k= 1,tot
c     Find the distances from (xi,yi) to all vortices

       r2=(xi-x(k))**2+(yi-y(k))**2
          arg(k)=r2/x5
          i1(k)=nint(sqrt(r2/x5))
          arg(k)=arg(k)-dble(i1(k)**2)
1     continue
      do 110 k=1,tot
         k3 = i1(k)
c     Rename the polynomial coefficients
            c2(k,10)=c(k3,10)
            c2(k,9)=c(k3,9)
            c2(k,8)=c(k3,8)
            c2(k,7)=c(k3,7)
            c2(k,6)=c(k3,6)
            c2(k,5)=c(k3,5)
            c2(k,4)=c(k3,4)
            c2(k,3)=c(k3,3)
            c2(k,2)=c(k3,2)
            c2(k,1)=c(k3,1)
110   continue
      do 4 k=1,tot
c     Horner's rule:
             x3=c2(k,10)
             x3=c2(k,9)+x3*arg(k)
             x3=c2(k,8)+x3*arg(k)
             x3=c2(k,7)+x3*arg(k)
             x3=c2(k,6)+x3*arg(k)
             x3=c2(k,5)+x3*arg(k)
             x3=c2(k,4)+x3*arg(k)
             x3=c2(k,3)+x3*arg(k)
             x3=c2(k,2)+x3*arg(k)
             x3=c2(k,1)+x3*arg(k)

             term1=x3*cc(k)/x5
             term2=term1*(xi-x(k))
             term1=term1*(y(k)-yi)
             u=u+term1
             v=v+term2
4         continue
      return
      end
```

References

[1] C.R. Anderson, "A Method of Local Corrections for Computing the Velocity Field Due to a Distribution of Vortex Blobs", J. Comp. Phys., **62** (1986), pp. 111-123.

[2] C.R. Anderson, "A Vortex Method for Flows with Slight Density Variations", J. Comp. Phys., **61** (1985), pp. 417-432.

[3] C.R. Anderson and C. Greengard, "On Vortex Methods", SIAM J. Numer. Anal., **22** (1985), pp. 413-440.

[4] S.B. Baden, "Run-Time Partitioning of Scientific Continuum Calculations Running on Multiprocessors", Ph.D. Thesis, Univ. of California, Berkeley, 1987.

[5] J.T. Beale, "On the Accuracy of Vortex Methods at Large Times", Proceedings of the Workshop on Computational Fluid Dynamics and Reacting Gas Flows, I.M.A., Univ. of Minnesota, 1986.

[6] J.T. Beale and A. Majda, "Vortex Methods. I: Convergence in Three Dimensions", Math. Comp. **39** (1982), pp. 1-27.

[7] J.T. Beale and A. Majda, "Vortex Methods. II: Higher Order Accuracy in Two and Three Dimensions", Math. Comp. **39** (1982), pp. 29-52.

[8] J.T. Beale and A. Majda, "Higher Order Accurate Vortex Methods with Explicit Velocity Kernels", J. Comp. Phys. **58** (1985), pp. 188-208.

[9] A.Y. Cheer, "Numerical Study of Incompressible Slightly Viscous Flow Past Blunt Bodies and Airfoils", SIAM J. Sci. Stat. Comp. **4** (1983), pp. 685-705.

[10] A.J. Chorin, "Numerical Study of Slightly Viscous Flow", J. Fluid. Mech., **57** (1973), pp. 785-796.

[11] A.J. Chorin, Vortex Models and Boundary Layer Instability, SIAM J. Sci. Stat. Comp. 1 (1980), pp. 1-21.

[12] A.J. Chorin and J. Marsden, *"A Mathematical Introduction to Fluid Mechanics"*, Springer-Verlag, New York, 1979.

[13] J.P. Christiansen, "Numerical Simulation of Hydrodynamics by the Method of Point Vortices", J. Comp. Phys., 13 (1973), pp. 363-379.

[14] G.H. Cottet, Méthodes Particulaires pour l'équation d'Euler dans le plan, Thèse de 3ème cycle, l'Université Pierre et Marie Curie, Paris, p.6, 1982.

[15] O.H. Hald, "Convergence of Vortex Methods for Euler's Equations, II", SIAM J. Numer. Anal., 16 (1979), pp. 726-755.

[16] O.H. Hald, "Convergence of Vortex Methods for Euler's Equations, III", SIAM J. Numer. Anal., 24 (1987), pp. 538-582.

[17] O.H. Hald and V.M. Del Prete, "Convergence of Vortex Methods for Euler's Equations", Math. Comp., 32 (1978), pp. 791-809.

[18] IMSL Library, 2,4, Houston, Texas, 1982.

[19] F. John, *"Partial Differential Equations"*, Springer-Verlag, New York.

[20] A. Leonard and P.R. Spalart, "Computation of Separated Flows by a Vortex Tracing Algorithm", in AIAA 14th Fluid and Plasma Dynamics Conference, AIAA-81-1246, 1981.

[21] Y. Nakamura, A. Leonard and P.R. Spalart, "Vortex Simulation of an Inviscid Shear Layer", in AIAA/ASME 3rd Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference, AIAA-82-0948, 1982.

[22] M. Perlman, "On the Accuracy of Vortex Methods", J. Comp. Phys., **59** (1985), pp. 200-223.

[23] V. Rokhlin and L. Greengard, "A Fast Algorithm for Particle Simulations", Research Report YALEU/DCS/RR-459, 1986.

[24] L. Rosenhead, "The Point Vortex Approximation of a Vortex Sheet", Proc. R. Soc. London Ser. A **134** (1932), pp. 170-192.

[25] J.A. Sethian, "Turbulent Combustion in Open and Closed Vessels", J. Comp. Phys., **54** (1984), pp. 425-456.

[26] J.A. Sethian and A.F. Ghoniem, "Validation Study of Vortex Methods", J. Comp. Phys., **74** (1988), pp. 283-316.

[27] P.R. Spalart, "Numerical Simulation of Separated Flows," Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University, 1982.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720