Vortex Method for computing high-Reynolds number flows: Increased accuracy with a fully mesh-less formulation

Thesis by

Lorena A. Barba

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2004
(Defended May 18, 2004)

Acknowledgements

It is of course self-evident that this doctoral work would not have been possible without the support of my supervisor, Prof. Anthony Leonard. But in this case, I can confidently say that it applies more than in any other. Prof. Leonard has been thanked by his students for many things: his open-door policy, "access to his wealth of experience and kindness" (I agree), his interest on the student's work as well as her well-being (indeed!), "great guidance, insight, and patience", "dedication to research and education" ... it is no wonder that he was recognized as Caltech Mentor of the Year 2000. My indebtedness extends in addition to having been offered a place in GALCIT to become his student, even though he was taking a sabbatical and wasn't really planning on getting new students that year. And above all, to his continued confidence, trust, and acceptance during all these years.

I am much obliged to the members of my examining committee: Prof. James L. Beck, Prof. Tim Colonius, Prof. Hans Hornung, and Prof. Dale Pullin. To Prof. Colonius in particular, thanks are due for calling to our attention the need to study the importance of convection error. To Prof. Pullin, my gratitude extends to his acceptance of chairing my Candidacy Committee, to the discussions we had of my work and his suggesting the application of the non-symmetric Burgers vortices, and also for showing interest in my well-being and success.

It is also my pleasure to acknowledge the early influence of Prof. Oscar Orellana, who nourished my thirst for learning with a myriad discussions on mathematics and philosophy. To him I owe finding the focus of my research interests in computational fluid dynamics, and defining my life perspectives by Aristotelian pleasure in learning and quest for happiness.

Abstract

For the applications of high Reynolds number flows, the vortex method presents the advantage of being free from numerically dissipative truncation error. In practice, however, many vortex methods introduce some numerical dissipation in mesh-based spatial adaption stages, or schemes such as vortex particle splitting. The need for spatial adaption in vortex methods arises from the Lagrangian framework, which results in an increase of discretization over time. Presently, a vortex method is devised that incorporates radial basis function (RBF) interpolation to provide spatial adaption in a fully mesh-less formulation. Numerical experiments show that there is a potential for higher accuracy in comparison with the standard remeshing techniques. The rate of convergence of the new spatial adaption method is exponential, however convection error limits the vortex method to second order convergence. Avenues for future research involve decreasing convection error, for example by means of deformable basis functions. Nevertheless, the RBF-based spatial adaption scheme has various advantages, in addition to a demonstrated higher accuracy and the obvious benefit of not requiring a regular arrangement of particles or mesh. One presently demonstrated advantage is automatic core size control for the core spreading viscous method, without the need for vortex particle splitting.

Three applications have been successfully treated with the presently developed vortex method. The relaxation of monopoles under non-linear perturbations has been computed, resulting in noticeable improvements compared to previously published results. The existence of a quasi-steady state consisting of a rotating tripole has been corroborated, for the case of large amplitude perturbations. The second application consists of the early adaptation of two co-rotating vortices at high Reynolds number, characterized by elliptical deformation of the cores, as well as small scale deformation in the weak areas of vorticity. This is considered to pose a severe test on the present method, or indeed any method. Comparison with results using spectral methods demonstrate in practice the potential for high accuracy computations using a mesh-less method, and in addition show that the naturally adaptive vortex method can result in considerably reduced problem sizes. Finally, for the calculation of non-symmetric Burgers vortices, a correction to the core spreading method for out-of-plane strain was developed. The results establish the capability of the vortex method for the computation of vortices under three-dimensional strain.

Contents

Chapter 1

Introduction and Overview

In this research, a new formulation of the vortex particle method for incompressible flows is implemented. The general objective has been to develop improvements in the vortex method to enhance its suitability for the computation of high-Reynolds number flows. To this purpose, the accuracy limitations of the different algorithmic elements of a vortex method have been investigated, with the aim of producing developments in the key components of a code implementation. An additional objective of this investigation is the development of a vortex method that is wholly grid-free, and the demonstration in practice of the capabilities of a grid-free method to accurately compute flows at moderate and high Reynolds numbers.

Vortex methods for the simulation of incompressible flows correspond to a numerical approach with three fundamental features. First, the Navier-Stokes or Euler equations are formulated in terms of vorticity and so the spatial discretization is carried out over the vorticity field instead of the velocity field. Second, making use of one of Helmholtz' theorems which states the correspondence of vorticity elements with material fluid elements, the computational vortex elements are Lagrangian and so convect with the fluid velocity. And third, to obtain the fluid velocity one makes use of the fact that the vorticity, defined as $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, can be inverted giving the velocity $\mathbf{u}$ as an integral over the vorticity field. This is the Biot-Savart law in vorticity kinematics, which allows to completely describe the flow field by tracking vorticity elements.

The fundamental features described above generate both the most desirable aspects of vortex methods, as well as their most serious difficulties. Describing the flow in terms of vorticity is desirable due to the intuitive power of visualizing the vorticity field, especially in complex and unsteady flows. Another advantage is the fact that the pressure drops out of the governing equation, and thus only needs to be solved for when and where force measurements are desired. In addition, as the vortex method literature profusely extols, the fact that the vorticity field is predominantly compact means that smaller sized computational domains can be used, in comparison with primitive variable formulations, and also boundary conditions at infinity can be automatically satisfied. In contrast, satisfying the free-space boundary condition of external flows can be a delicate matter in grid-based methods with truncated flow domains. Furthermore, the Lagrangian vortex particles convect without numerical dissipation, as the non-linear term of the Euler or Navier-Stokes equations is traded by a set of ordinary differential equations for the particle trajectories. This is, again, in contrast with grid-based schemes which inevitably suffer from numerical dissipation. Finally, the essentially grid-free nature of the vortex particle method is itself an advantage, as grid-generation is often one of the most expensive processes in computational fluid dynamics, CFD. The difficulties which arise with vortex methods, on the other hand, will be discussed after first giving a brief historical account.

Vortex methods have been around almost as long as finite differences and the earliest methods of computational mathematics. Indeed, the seminal work of Präger [160] with vortex distributions on surfaces is the origin of panel methods —widely used in the aeronautics industry to this day, whereas Rosenhead's work on the calculation of vortex sheets with the point vortex method [169] was of such great consequence that it is still very much cited today; this work is a true classic in the field. Interestingly, Rosenhead's point vortex method was partially discredited around 1960 by the observation that proof of convergence of the method was lacking [28], and by computer calculations which exhibited apparently chaotic motion of the particles [29]. This last problem was attributed to the singular character of the induced velocity close to a point vortex and different approaches were proposed to deal with it. One of these lead to the vortex blob method [42] which is used in this investigation, while others deal with the problem analytically by removing the singularity in the Biot-Savart expression (see [106, 108, 107]).

The modern vortex method was born in the 1970s and the prominent investigators involved in its early development are A. Chorin, A. Leonard, and C. Rehbach in France. Much interest in vortex methods during the early 1980s focused on mathematical aspects such as the convergence properties [83, 81, 13, 14, 15, 16]. In later years the development of the method was very rich, mainly in relation to the issues of accurate inclusion of viscous effects, the treatment of boundary conditions at solid surfaces, and the efficient reduction of the computational costs, so as to make them suitable for the high-resolution simulation of unsteady, high-Reynolds number flows. Comprehensive reviews of the development of vortex methods and their applications can be found in [113, 114, 197, 183], and [162]; see [5] for a collection of articles that reveal the state of the subject at the beginning of the 1990's. Recently, a book has been published that is dedicated to the subject including many practical considerations for the implementation of the methods [48].

In spite of their contemporaneous evolvement with finite differences, vortex methods have not become a member of the standard or mainstream tools of CFD. They have sometimes suffered the reputation of being mostly coarse attempts at modelling flows of high complexity and unsteadiness, those flows which are still all but intractable with the traditional CFD methods. The main difficulties for vortex methods to be accepted in the mainstream of computational fluid dynamics have been threefold: *(i)* the numerical complexity of calculating the velocity using the Biot-Savart law, which is in fact analogous to an "N-body problem" and hence requires order N^2 operations for N vortex elements; *(ii)* the inconvenience of adding viscous effects in a Lagrangian formulation, diffusion being viewed as much more readily computed using grid-based methods; and, *(iii)* the effect of the Lagrangian time evolution, which results on a loss of discretization accuracy due to the distortion of the particle distribution.

The first of the mentioned difficulties has been successfully addressed by the application of the Fast Multipole Method [80] for the calculation of the particle velocities, whereas some workers have bypassed the problem with mixed Eulerian-Lagrangian formulations such as the vortex-in-cell method [43, 55, 175, 45, 196, 30, 64, 119, 118, 54], although at the cost of adding interpolation errors. The inclusion of viscous effects, on the other hand, has benefited from profuse research, there being at least seven proven schemes, with varied degree of success. The loss of accuracy due to Lagrangian distortion of the particles, finally, has been generally dealt with by the application of "remeshing schemes", which utilize high-order interpolation kernels on a Cartesian tensor product formulation. The standard remeshing schemes have made long-time, accurate calculations of complex flows possible; they have, however, caused quite a bit of controversy as they add a mesh to an otherwise mesh-less method. In addition, they do introduce some interpolation error, generally accepted as tolerable. As one wishes to simulate flows at higher Reynolds numbers, however, increased resolution becomes crucial and the interpolation error may be a limitation. Also, a more accurate method may allow for reduced problem sizes at high Reynolds numbers (*i.e.*, smaller numbers of vortex particles for a given accuracy). But most importantly, there are problems in fluid dynamics where numerical diffusion, which is introduced inevitably by mesh-based interpolation schemes, can completely annihilate the physics. This is particularly true of vortical flows at high Reynolds numbers. One can cite the example of vortex-blade interaction in rotorblades, where capturing the blade tip vortices for one or more revolutions of the rotor is still impracticable using conventional mesh-based methods [2, 3]. For this reason, the present investigation endeavors to provide a fully mesh-less method of calculation. It is argued that the mesh-less approach can be even extended to spatial adaption processes, and this concept is demonstrated by using a technique of radial basis functions for scattered data interpolation. Ample numerical experimentation will demonstrate that increased accuracy is possible, in comparison with standard vortex methods, and applications in viscous vortex interaction at high Reynolds numbers will thus be practicable.

1.1 Basic Formulation of a Vortex Method

Let $\mathbf{u}(\mathbf{x}, t)$ be the velocity field and $\boldsymbol{\omega}(\mathbf{x}, t) = \nabla \times \mathbf{u}(\mathbf{x}, t)$ the vorticity field. Taking the curl of the momentum equation and considering an incompressible fluid for which $\nabla \cdot \mathbf{u}(\mathbf{x}, t) = 0$, the vorticity transport equation is obtained. This is the governing equation in vortex methods, which for three-dimensional flow corresponds to the following vector equation,

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathbf{u} \cdot \nabla \boldsymbol{\omega} = \boldsymbol{\omega} \cdot \nabla \mathbf{u} + \nu \Delta \boldsymbol{\omega}. \tag{1.1}$$

The assumptions in the above equation are: constant density flow, conservative body forces, an inertial frame of reference and unbounded domain. In the case of a two dimensional and inviscid flow the right-hand side of (1.1) is zero and the governing equation reduces to the simple form $\frac{D\omega}{Dt} = 0$, where $\frac{D}{Dt}$ stands for the material derivative. This corresponds to the basic formulation of vortex methods, for which clearly a Lagrangian method based on elements of vorticity is natural and ideal. Based on this simplest of formulations, the vortex method historically found its first successful applications in the simulation of phenomena governed by the 2D Euler equations. Subsequently, vortex methods have been extended to three-dimensional flow by including the vortex stretching/tilting term, and have incorporated the presence of internal boundaries by using vortex sheet formulations in the inviscid case and vorticity generation models with boundary elements for the viscous case. Viscous effects were added first by the random walk method [42], but a number of so-called deterministic viscous schemes have been proposed and tested during the last two decades. And recently, some researchers have ventured on the addition of compressibility effects [66, 67, 143].

In the vortex blob discretization, the elements are identified by a position vector, $\mathbf{x}_i$; a strength vector (vorticity$\times$volume) of circulation; and a core size, σ_i. The discretized vorticity field is expressed as the sum of the vorticities of the vortex elements in the following way:

$$\boldsymbol{\omega}(\mathbf{x}, t) \approx \boldsymbol{\omega}^h(\mathbf{x}, t) = \sum_{i=1}^{N} \boldsymbol{\Gamma}_i(t) \zeta_{\sigma_i} \left(\mathbf{x} - \mathbf{x}_i(t)\right), \tag{1.2}$$

where $\boldsymbol{\Gamma}_i$ corresponds to the vector circulation strength of particle i (scalar in 2D). In the blob version of the vortex method —in contrast to point vortices, the elements have a non-zero core size σ_i and a characteristic distribution of vorticity ζ_{σ_i} , commonly called the cutoff function. Frequently, the blob cutoff function is a Gaussian distribution and the core sizes are uniform ($\sigma_i = \sigma$), which means that in two dimensions one has

$$\zeta_\sigma(\mathbf{x}) = \frac{1}{k\pi\sigma^2} \exp\left(\frac{-|\mathbf{x}|^2}{k\sigma^2}\right), \tag{1.3}$$

where the constant k determines the width of the cutoff and is chosen by different authors as either 1, 2 or 4. For example, in the review paper of Leonard [113], the Gaussian blob is presented with $k = 1$; this form is also used in [16, 162] and many others [70, 44, 147, 126]. In the accuracy studies of Perlman [152], $k = 2$ is chosen, as in [105] and [193]. In Rossi's studies of core spreading diffusion [170], finally, $k = 4$ is used. The three versions of the 2D Gaussian blob are plotted in Figure 1.1.

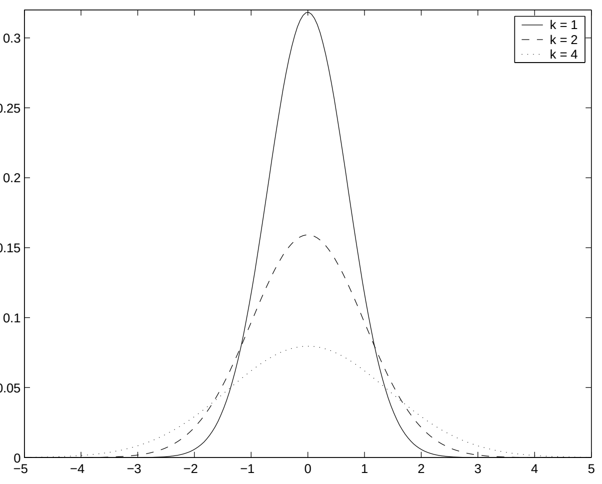


Figure 1.1: Gaussian blob distributions, three different versions (1D slice of 2D functions).

In this investigation, we have used $k = 2$.

In the majority of vortex methods (almost all), the Lagrangian formulation is expressed by assuming that the vortex elements convect without deformation with the local velocity. The velocity is obtained from the vorticity using the Biot-Savart law:

$$\begin{aligned}\mathbf{u}(\mathbf{x}, t) &= \int (\nabla \times \mathbf{G})(\mathbf{x} - \mathbf{x}')\omega(\mathbf{x}', t)d\mathbf{x}' \\ &= \int \mathbf{K}(\mathbf{x} - \mathbf{x}')\omega(\mathbf{x}', t)d\mathbf{x}' = (\mathbf{K} * \omega)(\mathbf{x}, t) \end{aligned} \tag{1.4}$$

where $\mathbf{K} = \nabla \times \mathbf{G}$ is known as the Biot-Savart kernel, $\mathbf{G}$ is the Green's function for the Poisson equation, and $*$ represents convolution. For example, in two dimensions the Biot-Savart law is written explicitly as

$$\mathbf{u}(\mathbf{x}, t) = \frac{-1}{2\pi} \int \frac{(\mathbf{x} - \mathbf{x}') \times \omega(\mathbf{x}', t)\hat{\mathbf{k}}}{|\mathbf{x} - \mathbf{x}'|^2} d\mathbf{x}'. \tag{1.5}$$

For the customary case of an axisymmetric cutoff function $\zeta = \zeta(r)$, $r = |\mathbf{x}|$, the velocity kernel can be obtained analytically. The velocity regularization function is defined as the integral

$$q(r) = \int_0^r \zeta(r)\, r\, dr. \tag{1.6}$$

The regularized Biot-Savart kernel is expressed as follows, where $\times$ represents cross product (with the vorticity vector, or $\omega\hat{e}_{\mathbf{z}}$ in the 2D case) and d is the dimension:

$$\mathbf{K}_\sigma(\mathbf{x})\times = -\frac{q(|\mathbf{x}|/\sigma)}{|\mathbf{x}|^d} \cdot \mathbf{x}\times \tag{1.7}$$

Therefore, for the 2D Gaussian blob with $k = 2$ one has

$$\mathbf{K}_\sigma(\mathbf{x}) = \frac{1}{2\pi r^2}(-y, x)\left(1 - \exp(-\frac{r^2}{2\sigma^2})\right). \tag{1.8}$$

The formula for the discrete Biot-Savart law in two dimensions gives the velocity as follows,

$$\mathbf{u}(\mathbf{x}, t) = -\sum_{j=1}^{N} \Gamma_j\ \mathbf{K}_\sigma(\mathbf{x} - \mathbf{x}_j). \tag{1.9}$$

Finally, the Lagrangian formulation of the (viscous) vortex method in two-dimensions is expressed in the following system of equations:

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{u}(\mathbf{x}_i, t) \tag{1.10}$$

$$\frac{d\omega}{dt} = \nu\nabla^2\omega + \text{B.C.} \tag{1.11}$$

The complete numerical method is defined by Equations (1.10) and (1.11) which express that the method is to be implemented by integrating the particle trajectories due to the local fluid velocity, while the velocity is obtained from the vorticity using the Biot-Savart law. The vorticity field evolves due to the effects of viscosity, both in the free-stream and on the boundaries (no-slip condition, denoted by B.C.). The viscous effects in the free-stream are enforced by one of a variety of viscosity schemes available for vortex methods (described in §2.1), while the effects due to solid boundaries are traditionally accounted for by generation of vorticity implemented in a version of the boundary element method. This is based on the physical mechanism by which the solid wall is a source of vorticity that enters the flow, so a vorticity flux $\frac{\partial\omega}{\partial n}$ may be determined at the wall to satisfy the boundary condition of no-slip at the surface [104].

1.2 Overview

As mentioned, the objective of this research is the development of a vortex method which is capable of computing high-Reynolds number flows accurately. The main difficulty in computing viscous flow at high Reynolds numbers is the fact that numerical diffusion can dominate over the physical viscosity, which many times destroys the physics that one wishes to observe. This is particularly true of viscous flows with concentrated regions of vorticity.

To improve on the accuracy that can be obtained with the current vortex methods, one needs to address one-by-one the different components of a realistic implementation of the method, and find where the accuracy issues lie. To start, it is helpful to get a broad picture by considering the "building-blocks" of any viscous vortex method implementation. Based on the description of the algorithm given above, a basic vortex method implementation can be represented graphically by the flowchart in Figure 1.2. This diagram represents the most basic building blocks of a vortex method,

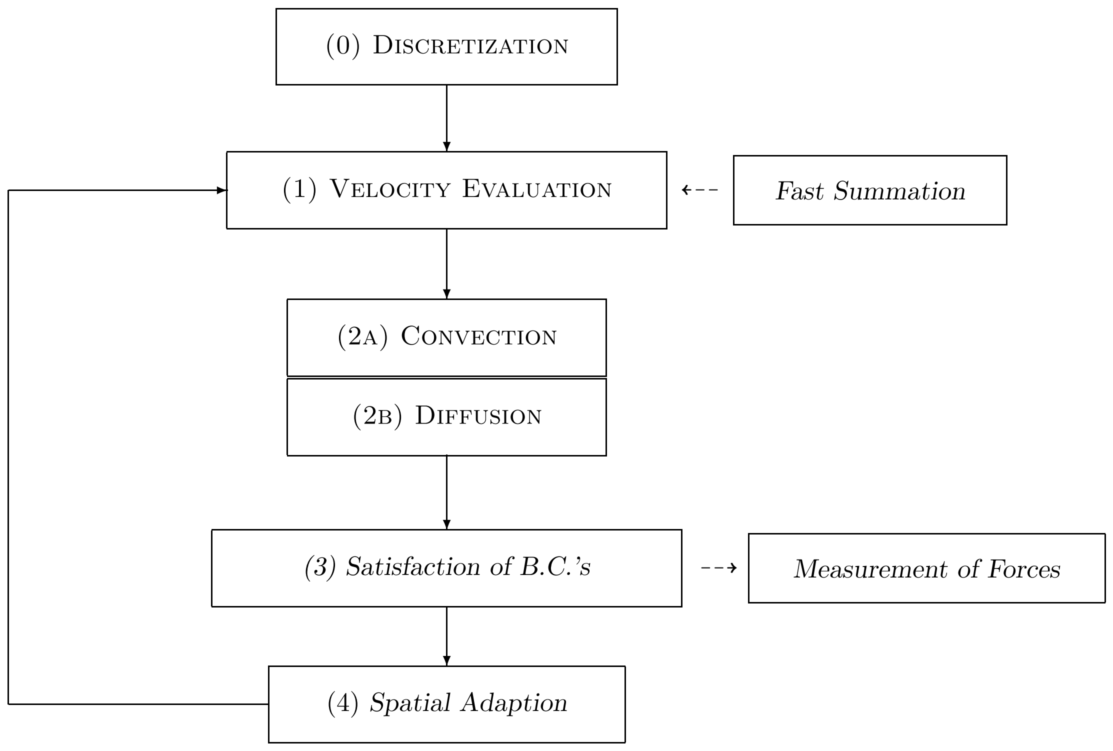


Figure 1.2: Basic building blocks of a viscous vortex method implementation.

and their algorithmic relationships. In small capital letters are indicated the compulsory, minimal components of a viscous vortex method. In slanted font are indicated the parts which, although not obligatory, are generally necessary in a modern application of vortex methods to bounded flows. The arrows designate the program flow in a time-marching algorithm.

The first basic component of a vortex method, the Discretization, consists of representing accurately a given, initial, vorticity field using vortex particles or blobs. This stage includes the choice of a cutoff function and optimal discretization parameters. In essence, as will be discussed amply later on, the problem of accurately discretizing a vorticity field in the vortex method is one of function approximation using nodal functions with global influence. This problem, fortunately, has recently benefitted from considerable research efforts in the function approximation community, and the present work will use their results to remarkable benefit.

The next component in the diagram is the Velocity Evaluation on the location of each vortex element, by use of the discrete Biot-Savart law. Due to the global influence of the vortex particles, calculating the velocity on a single point in space requires $O(N)$ operations, and so the complexity of the direct evaluation of velocity is $O(N^2)$. This situation is analogous to the calculation of gravity forces due to the interaction of N masses, which has become known as the "N-body problem". Thanks to this analogy, a very important development in vortex methods was due to the cross-over application of the "fast-methods" developed for astrophysics, in particular the Fast Multipole Method (FMM) [80]. These methods have led to the approximate evaluation of velocity with a complexity of $O(N \log N)$ and even $O(N)$. Although a production vortex code would almost certainly require a fast summation method for velocity evaluation, particularly in three dimensions, this component has not been incorporated in the present investigation. On the one hand, the solution is well known, and we cannot hope to contribute much in this subject; the application of the FMM in vortex methods is well researched and efficient implementations have been developed [182]. It is, in addition, a component of the implementation that involves great programming effort. On the other hand, the use of a fast Biot-Savart method involves an approximation, and it introduces a controllable error. In this investigation, it is desirable to study the accuracy of all the other components of the vortex method with respect to the direct Biot-Savart evaluation. Subsequently, in a production code with fast velocity evaluation, the error introduced by the fast velocity evaluation is controlled by the multipole acceptance criterion [181], which can be provided as an input parameter to the calculation.

Perhaps at the heart of the vortex method are the components labelled (2a) and (2b), *i.e.*, Convection and Diffusion of the vortex blob elements. The Lagrangian convection of the vortex particles involves using an adequate time-stepping scheme and choosing an appropriate step size according to the characteristics of the flow and the desired accuracy. These issues will be briefly discussed later, and this investigation will for the most part utilize Runge-Kutta integration schemes of fourth order. Providing viscous diffusion effects, on the other hand, can be quite difficult in the context of a Lagrangian method. Over the past three decades, a vast amount of research in this subject has produced at least seven different schemes for adding viscosity in a vortex method calculation. In more or less chronological order, these are the random vortex method (RVM), core spreading, particle strength exchange (PSE), the vortex redistribution method, diffusion velocity, Fishelov's method, and the triangulated vortex element method. Due to this profusion of methods, this investigation includes a review and assessment of the advantages and disadvantages of viscous schemes, after which a case is made for the core spreading method. Also, the problems associated with using core spreading will be tackled in a novel (yet simple) way, avoiding numerically diffusive splitting and merging schemes [170, 171].

The splitting/merging schemes mentioned above are one example of *Spatial Adaption* in vortex methods. In the case of core spreading, convergence of the method necessitates some form of core size control, tackling the concerns of [79]. In general, however, the Lagrangian deformation of the particle field is tied to a loss of accuracy of discretization, no matter the viscous scheme used. This issue will be thoroughly investigated in the present work, bringing into play new research in the area of function approximation with nodal functions. Traditionally, vortex methods have incorporated spatial adaption in the form of "remeshing" and "rezoning" algorithms, which consist in re-starting the particle field on a regular lattice every few time steps, and obtaining the circulation strength at the new particle locations by interpolation or other means. The present investigation will include a considerable amount of numerical experiments and an analysis of the standard methods. In particular, it will be demonstrated that remeshing can introduce visible interpolation errors and can put a limitation on the accuracy of the vortex method. Furthermore, an approach for providing spatial adaption in a mesh-less formulation will be developed and demonstrated, which is based on the technique of radial basis function (RBF) interpolation [31, 32]. This will be the principal contribution of the present research to increasing the accuracy of vortex methods for high-Reynolds number flow computations.

Finally, a vortex method application to bounded flows necessitates the *Satisfaction of boundary conditions* at the wall. The standard way to provide this, since the introduction of the vortex blob method by Chorin [42], is by application of the concept of vorticity generation at the surface. Implementations for this concept vary, but there is a widespread preference for Neumann-type formulation of the vorticity boundary condition and many workers use a model of vorticity creation at the wall. The vorticity creation algorithm is inspired by the model of Lighthill [116], who invoked the existence of vorticity sources and vorticity sinks in regions of falling or rising pressure (respectively) along a boundary. The prominent approach of formulating a viscous splitting algorithm [46, 104] to satisfy the boundary conditions has led to the popular use of boundary element methods (BEMs) coupled with the vortex method [101, 103, 105, 115, 193, 155, 194, 154, 156]. Although the present investigation has not included a study regarding the accuracy of the standard BEM-vortex method coupling, some ideas have sprouted from the survey of research in the subject of radial basis functions. It is well-known that panel methods "leak", that is, due to the satisfaction of the boundary conditions at a control point and the use of flat panels to approximate a curved surface, there is a non-zero velocity at the edges of the boundary elements. Since radial basis functions are now also being utilized for accurate three-dimensional geometric modelling [35], it is possible to conceive an application of nodal functions on surfaces to formulate boundary conditions in a vortex method *without panels*. This, however, is only speculative at the moment and it is proposed as one of the future roads for research.

In the next chapter, the current standard and experimental techniques that have been identified as topics where this investigation makes a contribution will be presented. This includes the subject of viscous vortex methods and also the vortex blob discretization and spatial adaption. In the first case, one is confronted by a vast assortment of methods that have been developed over the years, neither of which seems to be fully satisfactory. The goal of the discussion of this topic will be to identify where there is the most potential for an accurate method to compute flows at high Reynolds numbers, and this potential will be associated with the facility to produce a mesh-less formulation. Although, as will be seen, hundreds, maybe thousands, of research hours have been spent developing some elaborate algorithms to account for viscous effects, this investigation will be in support of the most utterly simple approach, namely, the core spreading method. It will be argued that the sole problem that rests is the provision of an accurate means for adaptive spatial refinement. Which brings to the next topic in the chapter: the spatial discretization in vortex methods. The way in which an existing or initial vorticity field is discretized using vortex blobs will be discussed first. Then, the different schemes that have been used for providing spatial adaption will be reviewed. It will be seen that all the existing techniques have the shortcoming of introducing a mesh, or rather of being dependent on a regular, Cartesian arrangement of particles. That is, all except for the vortex splitting concept, which nevertheless suffers from numerical dissipation. This discussion will aid in later making the connection between the vortex blob representation and the approximation of functions using nodal bases of global influence, which are briefly introduced at the end of Chapter 2. This, in turn, will lead to formulating applications of the technique of radial basis function interpolation to the initialization of a vortex method calculation and to spatial adaption. These applications will not only allow for a fully grid-free numerical method, but will provide for considerable increase in the accuracy that can be expected from a vortex method computation.

Subsequently, Chapter 3 will present a numerical investigation into the accuracy of vortex methods, including the effects of standard remeshing schemes. To investigate the accuracy of the vortex blob discretization and of the Lagrangian evolution of the vortex particles, two classic test problems are used, both problems of the simplest possible nature: an axisymmetric, inviscid vortex patch (1.12), and a Lamb-Oseen vortex (1.13), given by

$$\omega(r) = \begin{cases} (1-r^2)^k & r \leqslant 1 \\ 0 & r > 1 \end{cases} \tag{1.12}$$

$$\omega(r,t) = \frac{\Gamma_0}{4\pi\nu t} \exp\left(-\frac{r^2}{4\nu t}\right), \tag{1.13}$$

where $r = x^2 + y^2$. The first problem is particularly suited to observe the effects of features in the inviscid vortex method, as the exact solution consists of circular trajectories of different velocity and the initial particle distribution gets rapidly distorted due to the large shear (hence, this flow belongs to the class of problems known as circular shear layers). This class of problems has been used by many authors to test their methods. The case $k = 3$ was used for numerical experiments in [16] and it was also used in an example to observe the effect of different particle initializations in [48] (p. 28). Nordmark [144] used $k = 3, 7, 14$ in his numerical experiments and the case $k = 7$ was also used for accuracy tests by Perlman [152]. In the present work, we have used mostly $k = 3$, but $k = 7$ will be used briefly as well. The second test problem, the Lamb-Oseen vortex, is especially useful to consider different viscosity schemes, being the simplest viscous vortex flow and having an analytical solution; in addition, the vorticity transport equation reduces to the diffusion equation for this problem, so that the viscous effects are decoupled from the nonlinear effects. This classic test problem has been utilized by so many authors, it would be hard to list all of them. Presently, it will be useful in examining the effect of Reynolds number in the loss of regularity of the particle field. It will be seen how for large Reynolds number (small viscosity), the discretization errors quickly grow in a time-marching calculation. This responds to the fact that the vortex particles grow apart in certain areas of the domain, causing gaps to appear and thus becoming unable to reconstruct the smooth vorticity field. This is a well-known problem with vortex methods, which calls for the need to use spatial adaption algorithms.

Using the test problems described above, a study is performed of the errors produced by discretizing the vorticity using vortex blobs, and the importance of the overlap ratio, defined as the inter-particle spacing divided by the core size, h/σ, is clearly demonstrated. The effects of the Lagrangian deformation of the particle field will then be discussed and exposed by numerical experiments. Finally, numerical calculations using the standard remeshing schemes will show how these techniques serve to maintain accuracy for long-time simulations, but at the same time introduce noticeable errors themselves.

In addition, Chapter 3 will present numerical experiments exploring the comparative accuracy of different time stepping schemes. This will help to support the use of Runge-Kutta methods, which are sometimes discouraged due to the need for multiple velocity evaluations. In comparison to the one-evaluation Adams-Bashford schemes, it will be shown that the need for a much smaller time step when using Runge-Kutta results in their application being not only more accurate but also more efficient. Finally, this chapter will discuss and present numerical experiments using classic rezoning schemes [16], and demonstrate that they are useful but should only be applied when using high-order blob kernels.

Chapter 4 will develop the formulation of the vortex method with mesh-less spatial adaption based on radial basis function (RBF) interpolation. Preliminary numerical experiments are presented where, once again using the classic test problems discussed above, the comparative accuracy of the new method with respect to standard remeshing schemes can be ascertained. These experiments will show the potential of the new method for considerably increased accuracy. Also, the spatial adaption scheme will be used to provide the needed core size control for the application of the core spreading viscous scheme, without vortex particle splitting. The efficient implementation of the RBF interpolation problem by means of iterative methods of solution, in particular the generalized minimum residual or GMRES method, is here developed and tested. In addition, a first approach at studying the convergence properties of the new method will be shown to suggest spectral-like convergence; this is consistent with the error bounds for interpolation with Gaussian kernels, as established in the literature on radial basis function interpolation. Some additional discussions will be included regarding the numerical complexity of the evaluation of the RBF interpolants, which can be expensive unless fast methods are used, and regarding the possibilities for truly adaptive spatial refinement that is local, and based on error measurements or estimates.

To follow the presentation and the numerical experiments of the new, mesh-less vortex method formulation, a discussion of some numerical analysis topics in Chapter 5 will clarify some of the possible limitations. A discussion of the effect of a different initial lattice of particle locations will be presented, and an analysis of the convection error. Finally, this chapter includes some topics of numerical analysis of radial basis function interpolation that are likely to be relevant for their application in vortex methods.

In Chapter 6 are presented three applications of significant fluid dynamical interest where the capabilities of the vortex method with mesh-less spatial adaption are tested. The first application consists of the relaxation of monopoles under non-linear perturbations of quadrupolar structure. This flow has previously been shown to possess two possible quasi-steady states, one consisting of a rotating tripole and the other being the axisymmetric state. Which of these "attractors" is approached as the flow relaxes depends on the amplitude of the perturbation. The results obtained here corroborate the existence of this "tripole attractor", first observed in [174], and also provide smoother and better quality visualization than the previous work. Discrepancy is observed for the low Reynolds number case, which is attributed to the numerical dissipation present in the vortex particle splitting scheme used in [174].

The second application will explore the early interaction of two co-rotating vortices at high-Reynolds number, a problem recently studied by means of spectral methods in [111]. Our calculations are able to reproduce very well the previous results, which constitutes a severe test for the present method. Here, small-scale deformations of the vorticity field are observed in very weak areas, down to a level of 10^{-6}, and we are able to capture these very well. It is also shown that coarser resolutions smooth out these small scale deformations and lose some steepness of the vorticity gradient. Significantly, the vortex method is able to reproduce the spectral method results with a problem size reduced by two orders of magnitude. This is attributed to the fact that the vortex method can concentrate computational effort where it is needed, whereas the spectral method requires a very large computational domain to minimize the effects of image vorticity.

In the third and final application, that of non-symmetric Burgers vortices, the core spreading vortex method has needed a correction to account for the out-of-plane strain. This correction has allowed the use of the two-dimensional method for computing flows with uni-directional vorticity and three-dimensional strain, broadening the possible applications for the present method. Results are compared with previously published computations using pseudo-spectral methods [161].

Finally, Chapter 6 includes a grid-refinement study, performed with a parallel implementation of the program to compute the perturbed monopoles. This flow does not have radial symmetry and as a result this study obtains an observed second order of convergence. This has been attributed to the convection error, shown in §5.1 to be bounded by second order terms with respect to the core size of the elemental vortices. In consequence, and comparing with the first convergence study using a radially symmetric test problem, the next area of improvement for the vortex method lies in providing higher order convergence for the convection. This could be possible for example by allowing deformable basis functions, which is a subject of some current research efforts.

Chapter 2

Contributions to the Vortex Method

2.1 Viscous Schemes for Vortex Methods

Vortex methods have long proved to be an effective tool for the approximation of solutions to the Euler equations, and have been used for decades in the simulation of both unbounded and bounded flows. In most applications of more than academic interest, however, the limitations of the inviscid approximation cannot be accepted. For example, in flows around solid bodies viscous effects are needed to account for the mechanisms of vorticity generation at the surface and to accurately describe the vorticity dynamics in the wake. Unfortunately, it is not easy to implement a numerical solution of the diffusion term in the vorticity transport equation that is compatible with the Lagrangian formulation. This section examines the different schemes that have been introduced over the years to include viscous effects in vortex methods. First, the viscous splitting algorithm, or fractional step method, is described on which the majority of viscous vortex methods are based. The first scheme that was put forward to account for viscous effects, namely, the random vortex method, is described briefly, since it has become standard practice in engineering applications. The diverse deterministic (as opposed to random or stochastic) viscous schemes that have been introduced during the past twenty years or so as alternatives to the random vortex method include the particle strength exchange method, the redistribution method, Fishelov method, diffusion velocity method, and core spreading method, among others. This diversity of approaches deserves an in-depth analysis, not least because viscosity is a crucial fluid property that can dictate the physics of high-Reynolds number flows.

The subject is far from a consensus in regards to the "best" viscous scheme for Lagrangian methods based on vorticity, and ample room for improvement still exists. Many methods are based on the concept of viscous splitting which, although convergent and convenient in its implementation, means that no better than first order temporal accuracy can be achieved, irrespective of the time marching scheme (unless an elaborate multi-step splitting is used, rather than the conventional two-step method). Other methods are not based on viscous splitting but critically depend on the preservation of a certain "order" in the arrangement of vortex elements (*e.g.*, particle strength exchange, Fishelov method) which of course can be a serious problem in a Lagrangian method where elements are bound to become disorganized through convection. Hence, these susceptible methods depend on the application of remeshing processes consisting of tensor product interpolation with high-order kernels. The remeshing processes introduce numerical dissipation due to the interpolations and also do away with one of the favourable features of vortex methods, namely, their grid-free nature. Some methods are intrinsically computationally expensive and make use of many *ad hoc* numerical parameters, *e.g.*, the redistribution method, or require further development to be accurate and efficient in the application problems of interest, or are difficult to implement in three dimensions, which could be the case of the redistribution method as well as triangulated vortex elements.

After discussing the features of the different viscous schemes for vortex methods, it will be argued that the simplest of these approaches, core spreading, has the most potential for the applications of high-Reynolds number flows. This method has not been the choice of many workers in vortex methods. First of all, the mathematical objections of Greengard [79] effectively stalled the research efforts using this method, until a solution based on vortex blob splitting was proposed about a decade later by Rossi [170]. The consistency error of core spreading is caused by the advection without deformation of larger and larger vortex blobs as they spread. On the other hand, Rossi's vortex splitting idea is convergent and effective, but it introduces non-negligible errors. These will be argued here to be caused by numerical diffusion during splitting and lack of any overlap control. In fact, core spreading with vortex splitting may have an accuracy that is no better than the random vortex method. In the present investigation, the core size control needed for convergence of the core spreading method will be implemented without vortex splitting. This contribution will prove to be effectual in making core spreading a suitable method for computing high-Reynolds number flows.

2.1.1 Viscous Splitting

Some authors consider it a natural approach in a time-stepping scheme to take into account the viscous and inviscid parts of the governing equations as successive sub-steps. Indeed, in [48] the concept is associated to the division made in 1904 by Prandtl between viscous and inviscid effects. Viscous splitting —sometimes called "fractional step" method— is in fact a particular case of the general technique of "operator splitting", and it is introduced to viscous vortex methods together with the random walk diffusion scheme by Chorin [42]. Convergence of the viscous splitting algorithm for the Navier-Stokes equations in an unbounded flow is proved in [13] and [207]; bounded flow was considered in [24, 1] and [206]. The algorithm consists of sub-time-steps where the effects of convection and diffusion are considered successively. More sub-steps are involved in higher order schemes, but the basic two-step viscous splitting algorithm is second-order accurate at each time step and first-order overall (irrespective of the time-stepping scheme used). Following [48], this can be realized more easily by looking at the linear convection-diffusion equation for a scalar W,

$$\frac{\partial W}{\partial t} + \mathbf{c} \cdot \nabla W = \nu \Delta W$$

with initial condition $W_o(\mathbf{x})$. Using operator notation, where $\mathbf{c} \cdot \nabla \rightarrow A$ and $\nu\Delta \rightarrow B$, the above equation can be written as $\frac{dW}{dt} = AW + BW$, which is integrated to

$$W(t) = W_o \, e^{(A+B)t}.$$

Considering discrete time steps of length δt the solution at time $n\delta t$ is used as initial condition to obtain the solution at the subsequent time step, $W^{n+1} = e^{(A+B)\delta t} W^n$. The fractional step method is then expressed in the following way:

- *Sub-step 1: Convection.*

$$\frac{dW}{dt} = AW \quad \Rightarrow \quad W^{n+\frac{1}{2}} = e^{A\delta t} W^n$$

- *Sub-step 2: Diffusion.*

$$\frac{dW}{dt} = BW \quad \Rightarrow \quad W^{n+1} = e^{B\delta t} W^{n+\frac{1}{2}}$$

- *Approximated solution:*

$$W^{n+1} = e^{B\delta t} e^{A\delta t} W^n$$

For δt small, the operator in the expression above can be Taylor expanded into

$$\left(I + \delta t B + \frac{\delta t^2}{2} B^2 + \cdots \right) \left(I + \delta t A + \frac{\delta t^2}{2} A^2 + \cdots \right)$$

A comparison with the Taylor expansion of the exact operator $e^{(A+B)\delta t}$ reveals that the two expansions are equivalent only in the case of commutivity of the operators A and B. In general, however A and B don't commute, due to the vector field $\mathbf{c}$ being space-dependent, and so the error introduced is $O(\delta t)^2$ at each time step. Hence, this fractional step method is always first-order accurate in time.

Temporal accuracy of second order can be achieved with a three-step procedure called "Strang splitting" after [198]. This method is expressed in the following algorithm:

$$W^{n+1} = e^{B\frac{\delta t}{2}} e^{A\delta t} e^{B\frac{\delta t}{2}} W^n$$

Going back to the nonlinear Navier-Stokes equations in vorticity formulation, the (two-step) fractional step method is expressed in the sequence $\omega^{n+1} = H^{\nu}(\delta t)E(\delta t)\omega^{n}$ —where $E(t)\omega_o$ and $H^{\nu}(t)\omega_o$ are respectively the solutions at time t to the Euler and diffusion equations with initial condition ω_o—. Algorithmically, the method consists in advancing the vortices with the convective velocity in the first sub-step, and taking account of diffusion in the second sub-step. For two-dimensional, viscous flows

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \Delta \omega. \tag{2.1}$$

- *Sub-step 1: Convection.*

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \frac{D\omega}{Dt} = 0 \begin{cases} \dfrac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p) \\ \\ \dfrac{d\omega_p}{dt} = 0 \end{cases} \tag{2.2}$$

- *Sub-step 2: Diffusion.*

$$\frac{\partial \omega}{\partial t} = \nu \Delta \omega \begin{cases} \dfrac{d\mathbf{x}_p}{dt} = 0 \\ \\ \dfrac{d\omega_p}{dt} = \nu \Delta \omega(\mathbf{x}_p) \end{cases} \tag{2.3}$$

The convective sub-step in a vortex method is ideally expressed in the Lagrangian frame of reference, the same as for an inviscid flow. Using the discrete representation of the vorticity field, the velocity is obtained by the Biot-Savart law and the particles are advanced in the inviscid sub-step. But the discretization of the vorticity field in vortex methods, conceived for the simulation of the inviscid vorticity equation, is not well suited for the evaluation of the Laplacian in the diffusive term, because of the unstructured nature of the data. Hence the wide variety of approaches that have been put forward to include diffusive effects in a vortex method, as will be discussed in detail.

A straightforward procedure may be to use a grid-dependent scheme such as finite differences to solve the viscous sub-step, and apply mappings of the data from a structured grid overlaid on the irregular particle distribution, as was done in [37], where this method is used to study flow around a circular cylinder at Reynolds numbers of 300–10^6. This approach partially defeats the advantages of a Lagrangian method, since it brings about the need to generate grids in the fluid domain that conform to boundaries and it introduces numerical diffusion due to the interpolations between meshes. Finding other ways of accounting for diffusion in a Lagrangian method has been an active topic of research for many years. The oldest viscous scheme is the random vortex method, as already mentioned, and to make a distinction with this method of stochastic nature, alternative schemes proposed later have been termed "deterministic"; they include the core spreading method, particle strength exchange, redistribution method and diffusion velocity method, among others.

2.1.2 Random Vortex Method

The random vortex method (RVM) was introduced by Chorin [42] and is formulated essentially as a fractional step method. It takes into account the viscous effects in the mean, simulating diffusion by a random walk, *i.e.*, a Brownian-like motion of the vortex particles. Consider the discretized vorticity as expressed by (1.2); in the inviscid case, the particle strengths Γ_i remain constant, and ζ is not a function of time, *i.e.*, the vortices are elements of fixed geometry, and the particle positions are updated only as a result of convection. Various viscous vortex methods require that either the particle strengths or the base functions be modified in such a way that the diffusion equation is approximately solved. In contrast, the random vortex method acts by modifying the *positions* of the particles at each (diffusive sub-) step by adding a random walk, that is, the particle locations are transformed using: $x_i^{n+1} = x_i^n + \xi_i^n$, where the ξ_i^n are Gaussian independent random variables of zero mean and variance equal to $2\nu\Delta t$. This formula is based on the probabilistic interpretation of the diffusion equation, which says that the probability of finding a particle that moves at random in Brownian motion is given by (2.3). Consider the fundamental solution of a one-dimensional diffusion equation $\omega_t = \mathrm{Re}^{-1}\omega_{xx}$, for $-\infty < x < \infty, t > 0$:

$$G(x,t) = \sqrt{\frac{\mathrm{Re}}{4\pi t}} \exp\left(-\frac{\mathrm{Re}}{4t}x^2\right). \tag{2.4}$$

The function above is the same as the probability density function of a Gaussian random variable with zero mean and standard deviation $\sigma = \sqrt{2t/\mathrm{Re}}$. Hence, in two dimensions (2.3) is simulated stochastically by a displacement of the particles in two orthogonal directions, using two independent Gaussian random variables with $\sigma = \sqrt{2\Delta t/\mathrm{Re}}$.

The accuracy of the random vortex method was studied in [133], for an initially finite region of vorticity in an unbounded domain. They estimate the error to be of order $\sqrt{\nu/N}$, for N vortex particles, which in two dimensions corresponds to first order in the particle spacing, h, for a regular grid. Similarly, the rate of convergence corresponds to order $h^{3/2}$ in a three-dimensional rectangular discretization. Convergence proofs for the random vortex method were provided by [120] and [77]. A more detailed description of this method, presentation of convergence analysis and aspects of numerical analysis are provided in [48] (pp. 130–141).

It is worth mentioning that the RVM has been used extensively for modeling unsteady flows around bodies, where it has proved to reproduce global quantities and main features of the flow, while preserving the grid-free nature of vortex methods. The main disadvantage is that it requires a large number of particles to obtain reasonable accuracy, due to its slow convergence rate. On the other hand, only slightly viscous flows can be modeled, and at low Reynolds numbers the solution can be quite rough. Generally, the (lower) limit of the method is taken as Re=100. Additionally, forces and pressures can only be obtained by averaging or smoothing over several time steps.

Some benchmark numerical investigations using the RVM can be cited; for example, the circular cylinder flow studies of [175] at Re=250, the simulation of flow around general cylinder shapes at Re=1000 in [195], and circular cylinder flows at Re=3000, 9500 and 10^4 in [39], and at Re=300, 550, 3000, 9500 in [196]. The RVM has been implemented in parallel [190] taking advantage of the fact that the local nature of the scheme lends itself easily for parallelization. A more recent three-dimensional parallel RVM application to wall bounded flows was presented in [75], where the number of elements used was in the order of 10^5. Also, many hybrid methods have been introduced that, based on domain decomposition ideas, utilize the RVM in certain portions of the fluid and either inviscid or other viscous schemes in other zones. For example, in [44] the random walk and diffusion velocity methods were combined for the study of impulsively started flow around a circular cylinder. Diffusion velocity is used near the body surface, while random walk is applied at a certain distance from the body where it is assumed that the noise introduced by the method will not disrupt the boundary layer and affect force calculations. More recently, workers have experimented with a hybrid RVM/Finite volume method as well [157]. Research using the RVM is still active, and further applications include internal flows [76], turbulent flows with flames [36], and flows with free-surface effects [168, 189].

It is safe to say that the RVM has been the most widely utilized of viscous vortex methods, especially in engineering applications. But we wish to quote Prof. Sarpkaya [183] when he says that “the simulation of viscous diffusion by random walk is based on numerical convenience and it has nothing whatever to do with the physical process being simulated.”

2.1.3 Deterministic Vortex Methods

The limited accuracy of the random vortex method motivated the development of deterministic schemes, from the 1980’s onward. Many of these are also based on the viscous splitting concept, while some of the most prevalent are formulated by an integral representation of the solution to the heat equation, and the discretization of the integral operators using the particle positions as quadrature points, as described below.

The solution of the diffusion equation is expressed in integral form as follows,

$$\omega(\mathbf{x},t) = \int G(\mathbf{x}-\mathbf{y},\nu t)\omega_0(\mathbf{y})d\mathbf{y} \tag{2.5}$$

where G represents the heat kernel, *i.e.*, the Green’s function solution of the diffusion equation, which in two dimensions is

$$G(\mathbf{x}-\mathbf{y},\tau) = \frac{1}{4\pi\tau}e^{-|\mathbf{x}-\mathbf{y}.|^2/4\tau} \tag{2.6}$$

Viscous schemes which have been termed “resampling methods” [50, 163, 51] use this representation and, in contrast to the random-walk schemes which modify the particle locations, introduce a change in the strength of the vortex particles Γ_i at the diffusion sub-step. The method consists of resampling the vorticity field induced by each particle on its neighbors, in the following manner. If in the discretized vorticity (1.2), the circulation is replaced by vorticity times volume and the delta function is used as cutoff, and in (2.5) the vorticity at time step t_n is used as initial condition to obtain the vorticity at the next time step due to diffusion, after evaluating at the particle locations the following formula is obtained:

$$\omega_p^{n+1} = \sum_{q=1}^{N} v_q \omega_q^n G[\mathbf{x}_p^n - \mathbf{x}_q^n, \nu\delta t]. \tag{2.7}$$

This last equation is the induction rule used to update the vorticity of each particle at the diffusion sub-step. However, if used as presented above, the method does not conserve total circulation. To correct it, the fact is used that the heat kernel integrates to 1 over the whole domain, and the vorticity at t_{n+1} is re-written as

$$\omega(\mathbf{x}, t_{n+1}) = \omega(\mathbf{x}, t_n) + \int G(\mathbf{x} - \mathbf{y}, \nu\delta t)[\omega(\mathbf{y}, t_n) - \omega(\mathbf{x}, t_n)]d\mathbf{y}. \tag{2.8}$$

Again, substituting the discretized form of the vorticity and evaluating at the particle locations, the following conservative scheme is obtained to update the particle vorticities,

$$\omega_p^{n+1} = \omega_p^n + \sum_{q=1}^{N} v_q (\omega_q^n - \omega_p^n) G[\mathbf{x}_p^n - \mathbf{x}_q^n, \nu\delta t]. \tag{2.9}$$

For more details, see [48]. As explained therein, other resampling methods have been proposed, for example by utilizing another cutoff function instead of G which has the same moment properties. The method of Particle Strength Exchange (PSE) is a generalization of the idea of redistributing the particle circulations, similarly based on the integral approximation of the diffusion equation, but not conceptually linked to viscous splitting. Resampling methods turn out to be a particular case of PSE where a low-order discretization in time is used; hence, PSE, which has been amply tested in high-resolution simulation of moderate to high-Reynolds number flows, is discussed next.

2.1.4 Particle Strength Exchange

The principal features of the PSE method are, *(i)* it is based on the exchange of circulation among particles to approximate diffusion; *(ii)* it involves approximating the Laplacian at a particle's location based on nearby particles, and *(iii)* it is formulated grid-free but requires frequent remeshing of the particle field onto a well-ordered field. PSE is based on the general particle methods proposed by Degond and Mas-Gallic [57, 58]. The basis of the algorithm is that the Laplacian can be replaced by an integral operator —given that the smoothing function ζ satisfies certain moment conditions, see [48] p. 145— in this way,

$$\nabla^2\omega(\mathbf{x}) \approx \frac{2}{\sigma^2}\int \zeta_\sigma\left(\mathbf{x}-\mathbf{x}'\right)\left[\omega(\mathbf{x}')-\omega(\mathbf{x})\right]d\mathbf{x}' \tag{2.10}$$

or,

$$\nabla^2\omega(\mathbf{x}) \approx \int G_\sigma\left(\left|\mathbf{x}-\mathbf{x}'\right|\right)\left[\omega(\mathbf{x}')-\omega(\mathbf{x})\right]d\mathbf{x}'. \tag{2.11}$$

The order of accuracy of the approximation above depends on the smoothing function. It is accurate to $O(\sigma^2)$ for the Gaussian, in which case one has

$$G_\sigma\left(\mathbf{x}\right) = \frac{2}{\pi\sigma^2}\exp\left(\frac{-\left|\mathbf{x}\right|^2}{2\sigma^2}\right) \tag{2.12}$$

so that when $\sigma^2 = 2\nu\delta t$, G_σ is equivalent to the heat kernel. The integral operator in (2.11) is discretized by quadrature, using as quadrature points the locations of the particles. Hence, for the case of bounded flow, one can formulate the PSE method in the following way. The Lagrangian viscous vortex method expressed by Equations (1.10) and (1.11) is translated into the integral formulation

$$\begin{aligned}\frac{d\mathbf{x}_p}{dt} &= -\frac{1}{2\pi}\int \mathbf{K}(\mathbf{x}_p-\mathbf{x}')\times\omega d\mathbf{x}' + \mathbf{U}_o(\mathbf{x}_p,t)\\ \frac{d\omega}{dt} &= \nu\int G_\sigma\left(\left|\mathbf{x}-\mathbf{x}'\right|\right)\left[\omega(\mathbf{x}')-\omega(\mathbf{x})\right]d\mathbf{x}'\\ &+\nu\int H(\mathbf{x}_p,\mathbf{x}')\frac{\partial\omega}{\partial n}(\mathbf{x}')d\mathbf{x}'\end{aligned}$$

where the second term on the right-hand side of the second equation is the contribution from vorticity generation at the surface; U_o contains the contribution from the free stream velocity and any solid body rotation, and $\mathbf{K}\left(\mathbf{z}\right) = \mathbf{z}/\left|\mathbf{z}\right|^2$. Note that the mechanism of vorticity generation at the solid surface is expressed by an integral operator also, but this topic will not be discussed further here.

The full discretized equations can now be written. Using $\omega^h\left(\mathbf{x},t\right) = \sum_i \Gamma_i(t)\,\zeta_\sigma\left(\mathbf{x}-\mathbf{x}_i(t)\right)$, the following system is obtained,

$$\frac{d\mathbf{x}_p}{dt} = -\frac{1}{2\pi}\sum_i \Gamma_i\,\mathbf{K}_\sigma(\mathbf{x}_p-\mathbf{x}_i) + \mathbf{U}_o(\mathbf{x}_p,t) \tag{2.13}$$

$$\frac{d\Gamma_p}{dt} = \nu\sum_{i=1}^{N}\left[\Gamma_i-\Gamma_p\right]G_\sigma\left(\left|\mathbf{x}-\mathbf{x}'\right|\right) + \nu\sum_{m=1}^{M} H(\mathbf{x}_p,\mathbf{x}_m)\frac{\partial\omega}{\partial n}(\mathbf{x}_m) \tag{2.14}$$

Note that it is assumed that the solid surface was discretized using M panel elements. Both singular integral operators were convolved with the smoothing function and are replaced by smooth operators; $\mathbf{K}_\sigma$ is the regularized Biot-Savart kernel, $\mathbf{K}_\sigma = \mathbf{K} * \zeta_\sigma$.

It should be noted that PSE is not formulated in terms of viscous splitting; however, it does implement a fractional-step scheme for the enforcement of the boundary conditions. Generally, the splitting is as follows,

1. Particles are advanced using the velocity computed by the Biot-Savart law and integrated using an appropriate time-stepping scheme; their strength is modified using PSE, and the no-through flow boundary condition is enforced. At the end of this sub-step there is a slip velocity at the solid boundary.

2. The no-slip boundary condition is enforced by a vorticity generation algorithm. The vorticity flux from the surface within the time step is calculated so as to cancel the slip velocity generated at the previous sub-step.

In summary, the main feature of the PSE method is the replacement of differential operators by integral operators, more suited to the particle representation of the data. The integral operators are discretized by quadrature on the locations of the particles, then the discrete integral operator for diffusion reduces to a contribution from nearby particles to a change of circulation on a given vortex blob. Note that in Equation (2.14) the strength exchange is written as involving all particles; in practice, though, each vortex particle is allowed contributions to the change in its circulation strength from particles within a radius of, say, 5σ (as in [155]). This is allowed by the rapidly decaying smoothing function; it does mean that the discrete PSE operator does not approximate exactly the continuous integral, and so sometimes a re-normalization of the PSE kernel becomes necessary.

In principle, the formulation of PSE is grid-free, but the fact that the accuracy relies strongly on the quadrature rules used for the discretized integral means that in practice the method hinges on having nearly uniformly spaced particle locations. For this reason, the extensive utilization of the PSE method has promoted the development and widespread use of remeshing schemes in vortex methods [101, 103, 102, 115, 52] (see §2.2). This has caused a bit of debate, as some workers maintain that the grid-free nature of the vortex method is undermined when remeshing schemes relying on regular particle grids are applied, sometimes as often as *every* time step. Indeed, if this were the case, as pointed out by G. Winckelmans (private communication, 2002), then there may not be much difference between PSE with remeshing and vortex-in-cell methods (which nowadays utilize the same interpolation kernels as used in particle remeshing). In both cases, each interpolation step introduces some numerical diffusion, although these errors are generally considered acceptable and indeed a number of remarkable results have been obtained on unsteady wake flows, *e.g.*, [155, 156] using PSE with remeshing, and [158, 53] using the vortex-in-cell approach. In spite of this, it is possible that the interpolation errors in both of these approaches impose a limitation on the accuracy of the vortex method, especially at high Reynolds numbers where the small physical viscosity might be overwhelmed by the numerical effects. This thought will be one of the important motivations of the present investigation.

Let us conclude by saying that the PSE method has been extensively developed and applied in many benchmark problems of bluff-body flow. For example, high-resolution simulation of the impulsively started cylinder at Reynolds number from 40 to 9500 in [101, 103], and impulsively started and uniformly accelerated flat plate in [105]. The method has also been implemented in parallel [182] and applied to flows around arbitrary geometries [155]. For this reason, whereas the RVM can be considered the workhorse vortex method used in engineering, PSE can be considered the prevalent vortex method used by the academic community.

2.1.5 Redistribution Method

The motivation for the development of the vortex redistribution method (VRM) was precisely that of providing an alternative to PSE that does away with the need to perform remeshing of the particle field; in other words, the aim was retaining the grid-free formulation of the vortex methods [192]. Similarly to PSE, this method is based on the exchange of circulation between particles to simulate diffusion, but the formulation is different from PSE as it does not use integral operators. The algorithm hinges on finding fractions of each particle's circulation that will be redistributed among its neighbors, these being defined as those particles within a maximum distance in the order of the typical diffusion distance, $h_\nu = \sqrt{\nu \Delta t}$. The 'redistribution fractions' f_{ij}^n are solved for by assembling a system of equations which is formulated so that locally there is conservation of circulation, linear and angular momentum. The following transformation is sought:

$$\omega^n = \sum_i \Gamma_i^n \zeta_\sigma(\mathbf{x} - \mathbf{x}_i) \longrightarrow \omega^{n+1} = \sum_i \sum_j f_{ij}^n \Gamma_i^n \zeta_\sigma(\mathbf{x} - \mathbf{x}_j). \tag{2.15}$$

The approach used to determine the redistribution fractions is to demand that all finite wave numbers of the Fourier transform be correctly damped. The Fourier transform of the new vorticity field is

$$\widehat{\omega}^{n+1} = \widehat{\zeta}(k\sigma) \sum_i \Gamma_i^n \exp(-i\mathbf{k} \cdot \mathbf{x}_i) \sum_j f_{ij}^n \exp(-i\mathbf{k} \cdot (\mathbf{x}_j - \mathbf{x}_i)), \tag{2.16}$$

while the Fourier transform of the exactly diffused vorticity is

$$\widehat{\omega_e}^{n+1} = \widehat{\zeta}(k\sigma) \sum_i \Gamma_i^n \exp(-i\mathbf{k} \cdot \mathbf{x}_i) \exp(-k^2 \nu \Delta t). \tag{2.17}$$

These two Fourier transforms cannot be equal for all values of k using a finite number of vortices. Using the fact that within a "neighbourhood" of vortex i the distance $|\mathbf{x}_j - \mathbf{x}_i| \sim \sqrt{\Delta t}$ is small, the trailing exponentials in the two Fourier transforms can be approximated by a truncated Taylor series. The resulting equations are the redistribution equations, below. The neighbourhood of a particle is taken as a distance in the order of the typical diffusion distance, $h_\nu = \sqrt{\nu \Delta t}$; so, a vortex j is inside the neighbourhood of vortex i if: $|\mathbf{x}_j - \mathbf{x}_i| \leq R h_\nu$, the parameter R being chosen empirically (in [192], $R = \sqrt{12}$). Defining the scaled relative vortex position as: $\xi_{ij} \equiv \frac{\mathbf{x}_j - \mathbf{x}_i}{\sqrt{\nu \Delta t}}$ $(\leq R)$. The redistribution equations are

$$O(1): \qquad \sum_j f_{ij}^n = 1 \tag{2.18}$$

$$O(\sqrt{\Delta t}): \qquad \sum_j f_{ij}^n \xi_{1ij} = 0; \qquad \sum_j f_{ij}^n \xi_{2ij} = 0 \tag{2.19}$$

$$O(\Delta t): \qquad \sum_j f_{ij}^n \xi_{1ij}^2 = 2; \qquad \sum_j f_{ij}^n \xi_{2ij}^2 = 2; \qquad \sum_j f_{ij}^n \xi_{1ij} \xi_{2ij} = 0 \tag{2.20}$$

$$\begin{aligned} O(\sqrt{\Delta t^3}): \qquad & \sum_j f_{ij}^n \xi_{1ij}^3 = 0; \qquad \sum_j f_{ij}^n \xi_{1ij}^2 \xi_{2ij} = 0; \\ & \sum_j f_{ij}^n \xi_{2ij}^3 = 0; \qquad \sum_j f_{ij}^n \xi_{1ij} \xi_{2ij}^2 = 0. \end{aligned} \tag{2.21}$$

$$O(\sqrt{\Delta t^m}): \qquad \text{higher-order moment equations}, m = 4 \cdots M + 1.$$

The first three redistribution equations are necessary to ensure consistency in the sense that the numerical solution approximate the $O(\Delta t)$ diffusive changes in the exact solution, with a truncation error of order h_ν. The physical meaning of the equations is the following: (2.18) implies conservation of circulation for each vortex; (2.19) locally conserves the center of vorticity; and, (2.20) implies the correct expansion of the mean diameter of the vortex system. The subsequent equations can be used to obtain a higher order of accuracy. However, viscous splitting error is present.

Additional components of the scheme are, first, that positivity of the solution is enforced as a stability condition, *i.e.*, it is demanded that all fractions be positive: $f_{ij}^n \geq 0$. The positivity condition expresses the physical fact that reverse vorticity cannot form spontaneously in the flow field. Second, whenever it is encountered that there is no acceptable solution to the system of equations, an *ad hoc* algorithm is used that inserts new vortex particles within the neighbourhood in question. The number of vortices can increase without apparent bound when the Reynolds number is high, so vortex particle merging is sometimes used to alleviate this problem. Finally, preconditioning of the system of equations is needed in practice.

The authors of the VRM claim that the advantage of the method is that it retains the grid-free nature of vortex methods, because it is not needed that the particles be in an ordered distribution (in contrast to PSE). This is achieved, however, at the high cost of solving N underdetermined systems for N particles, at each time step. The size of these systems is determined by the redistribution influence neighbourhood, $|\mathbf{x}_j - \mathbf{x}_i| \leq R\sqrt{\nu \Delta t}$, involving an empirically chosen parameter R. Then again, as will be discussed in §§2.2.1, 3.2 and 3.3, no matter the viscous scheme, the accuracy of the vortex method in general does depend on preserving overlap throughout a calculation. The vortex insertion algorithm in the VRM does provide some form of spatial adaption, but with no overlap control. Finally, there may also be numerical diffusion involved in the vortex merging processes.

The method also has the disadvantages of using various parameters that need to be "tuned" in a practical implementation, and involving a considerable computational cost, which can be of the same order as the velocity evaluation, and even larger. In comparison, as reported in [48] (p. 163), the computation of diffusion using PSE including remeshing has a small ($\sim 2\%$) computational cost compared to the convective step.

2.1.6 Fishelov Method

A deterministic scheme for diffusion which is not based on the exchange of circulation was proposed by Fishelov [70], whose suggestion was to convolve the vorticity with a cutoff function and then approximate the second-order derivatives in the Laplacian by explicit differentiation of the cutoff function. The concept is based on Anderson and Greengard [4], who explicitly differentiate the smooth kernel to obtain the stretching term in a three-dimensional inviscid vortex method algorithm.

The problem is how to approximate the spatial derivatives appearing in the vorticity transport equation. Assume a uniform grid at t=0, with grid spacing h, and write $\mathbf{x}_p^h(t)$, $\omega_p^h(t)$ for the approximate particle locations and approximate vorticity respectively, then using the discretized form of (1.4) one can write

$$\frac{d\mathbf{x}_p^h(t)}{dt} = \sum_{j=1}^{N} \mathbf{K}_\sigma(\mathbf{x}_p^h(t) - \mathbf{x}_j^h(t))\omega_j^h(t)h^2, \tag{2.22}$$

where $\mathbf{K}_\sigma$ is the regularized Biot-Savart kernel, $\mathbf{K}_\sigma = \mathbf{K} * \zeta_\sigma$. Then, by explicit differentiation of the smoothed kernel in Eulerian coordinates, the stretching term can be written as

$$\omega_p^h(t) \sum_{j=1}^{N} \nabla\mathbf{K}_\sigma(\mathbf{x}_p^h(t) - \mathbf{x}_j^h(t))\omega_j^h(t)h^2. \tag{2.23}$$

Similarly for the viscous term, when the vorticity is convolved with the smoothing function, $\omega \approx \zeta_\sigma * \omega$, then the Laplacian can be approximated in this way: $\Delta\omega \approx \Delta(\zeta_\sigma * \omega) = \Delta\zeta_\sigma * \omega$. Approximating the integrals by the trapezoid rule, the following ODE's are obtained, which together with (2.22) constitute Fishelov's vortex method, expressed by the following system of equations:

$$\begin{aligned}\frac{d\omega_p(t)}{dt} &= \omega_p^h(t) \sum_{j=1}^{N} \nabla\mathbf{K}_\sigma(\mathbf{x}_p^h(t) - \mathbf{x}_j^h(t))\omega_j^h(t)h^2 \\ &\quad +\nu \sum_{j=1}^{N} \Delta\zeta_\sigma(\mathbf{x}_p^h(t) - \mathbf{x}_j^h(t))\omega_j^h(t)h^2. \end{aligned}\tag{2.24}$$

Note that in the above formulation time-splitting has not been applied. Fishelov proves stability under the condition that the Fourier transform of the cutoff function is nonnegative, and gives five examples that satisfy the condition. Using one of these —a fourth-order cutoff function based on exponentials—, she carries out some simple numerical experiments in two dimensions, using two test problems: a step function initial vorticity, which she compares to RVM results; and Chorin's periodic test case (from [41]), which has an exact solution.

In [71], Fishelov utilized her method for the simulation of flow around a sphere at Re=3000, where the vortex method was used only in the far-away region and a boundary layer formulation was used near the body with random walks to approximate viscosity. The results presented, however, seem quite coarse and the simulation short-lived. The method was also used by Bernard for two-dimensional turbulent boundary layer flow in [25]. He used vortex sheet elements of rectangular geometry or "tiles" of vorticity, which are convected as solid bodies. The application problems are start-up channel flow at Re=1000 and zero-pressure gradient (Blasius) and stagnation (Faulkner-Skan) boundary layers at Re=1000. These simulations utilize small numbers of elements ($N \sim 10^3$) and boundary layer approximations to evaluate the velocity field. Also, an approximate method is used for evaluating the Laplacian of the cutoff function taking advantage of the regular geometry of the vortex elements, and many *ad hoc* considerations are included for ensuring that no voids in the element population arise as a result of convection, as these would distort the diffusive process.

Applications of the Fishelov method seem to be scarce and have not provided much confidence in it. The method encounters difficulties when the particle field gets distorted, and seems to be even more sensitive to this problem than the PSE scheme. There do not appear to be any ongoing research efforts to use the method with frequent remeshing processes, but in this case PSE may be preferable. However, in the context of analyzing the use of higher-order cutoff functions, the Fishelov method is used in [145] to calculate unbounded two-dimensional vortex flows, using what is called an "automatic rezoning strategy"; this consists in resetting the particle locations on a uniform grid when the error in vorticity has grown past a certain limit. The rezoning scheme was previously introduced for inviscid flows in [144]. This work has not been extended to flows with boundaries; indeed, in [145] it is argued that the use of vorticity creation at the boundary to approximate boundary conditions introduces errors that make the use of high order vortex methods ineffectual.

Finally, as noted in [48] (p. 147), the Fishelov method is not conservative, as it does not include the correction term $-\omega_p \sum_q v_q \Delta\zeta_\sigma(\mathbf{x}_p - \mathbf{x}_q)$ that the integral approach requires, just as in the resampling methods, *cf.* (2.9). This could be rectified, but there does not seem to be further research in this line.

2.1.7 Diffusion Velocity

The diffusion velocity approach [146] can be deduced from considering the general case of an arbitrary scalar function in two space dimensions $F(x, y, t)$ that moves with velocity $\mathbf{u}(x, y, t) = (u, v)$, whose evolution equation can be written as

$$\frac{\partial F}{\partial t} + \frac{\partial uF}{\partial x} + \frac{\partial vF}{\partial y} = 0. \tag{2.25}$$

The two-dimensional vorticity transport equation (2.1) can be written in this form, using the incompressibility condition,

$$\frac{\partial \omega}{\partial t} + \frac{\partial}{\partial x}\left[\left(u - \frac{\nu}{\omega}\frac{\partial \omega}{\partial x}\right)\omega\right] + \frac{\partial}{\partial y}\left[\left(v - \frac{\nu}{\omega}\frac{\partial \omega}{\partial y}\right)\omega\right] = 0. \tag{2.26}$$

Comparing (2.25) with (2.26), where (u, v) is the convective velocity, the diffusion velocity is defined by the extra contribution to the total velocity, as follows,

$$\mathbf{u}_d = -\frac{\nu}{\omega}\left(\frac{\partial \omega}{\partial x}, \frac{\partial \omega}{\partial y}\right). \tag{2.27}$$

So that now the vorticity equation can be written in a conservation form as

$$\frac{\partial \omega}{\partial t} + \nabla(\omega \mathbf{u} + \omega \mathbf{u}_d) = 0. \tag{2.28}$$

The concept of a diffusion velocity implies that the net flow of vorticity is proportional to the vorticity gradient, which is analogous to Fick's First Law of diffusion where ν/ω is taken as a non-constant diffusion coefficient. It translates into saying that ω has a preferred direction of transport, namely that of $\nabla\omega$. The way the vorticity gradient is obtained in the diffusion velocity method is by directly taking the derivatives of the cutoff function in the discretized vorticity representation.

The suitability of the diffusion velocity concept was demonstrated in [146] for a one-dimensional diffusion test problem and for the case of a circular cylinder at Re=1200 and Re=40 (below the limit of applicability of the RVM). However, the results presented for cylinder flow are of rather low resolution when compared with finite-difference calculations (The authors themselves admit: "the streamlines by our method are somewhat clumsy..."). One notes, as well, that the one-dimensional proof-of-concept calculation was started with core size $\sigma = 0.4h$, which means that overlap of the vortex particles was not enforced.

Later work [44] demonstrated a higher-resolution application of diffusion velocity, and pointed out that the scheme models diffusion correctly only where particle overlap is maintained. That investigation used a hybrid method, with diffusion velocity near the surface of the body, and random walks farther away where overlap has been lost due to convection but the low resolution of the RVM is not damaging to force calculations and boundary layer resolution. The application code is written in parallel and benchmarked using the problem of the translating circular cylinder at Reynolds numbers of 300, 550, 3000, and a rotating circular cylinder at Re=1000. A vortex panel method is used for satisfaction of the boundary conditions at the body by a vorticity creation algorithm, and forces were calculated as well as streamline patterns finding good agreement with experiment and results produced using finite differences. One should rightly be apprehensive, however, of the authors' decision to use the *strength* of the particular vortex particle whose diffusion velocity is being calculated, rather than the local vorticity, in the denominator of Equation (2.27). The authors simply say that the quality of results was "unaffected" by this decision. In addition, to avoid problems were particles are very weak, resulting in very large diffusion velocity, the vortices that are weaker than 0.1% of the maximum circulation are simply deleted.

Additional results for the circular cylinder using the diffusion velocity scheme, but for a wider range of Reynolds numbers in the range of $10^{-1} - -10^{7}$, were presented in [147]. Although these calculations are performed over relatively long times, the unsteady forces are rather noisy and the streamline patterns quite rough. Still, enough simulations were performed to present a plot of the change of drag coefficient with Reynolds number, which shows a "good resemblance to that of the experimental data", according to the authors. One must question, however, results of two-dimensional calculations that claim good agreement with experimental data at Re up to 10^{7}, as it is well-known that there exist considerable three-dimensional effects from around $Re = 300$ up. If one attentively examines the plots presented in [147], one sees qualitative agreement at best.

The diffusion velocity scheme was demonstrated for calculation of thin boundary layers in [199], where a fix was provided for the problem that can present itself where the vorticity is zero while the gradient of vorticity is not (which risks a division by zero). Also, a "bi-zonal" approach is used to solve the topological problem which exists when using vortex blobs near a wall, consisting of using stream-wise elongated elements in the wall region and blobs in the outer region. The method was benchmarked using the flow of an infinite plate moving at constant velocity, an impulsively started plate (Stokes' 1st problem), a sinusoidally moving plate (Stokes' 2nd problem), and a Blasius boundary layer. Reynolds numbers are in the order of 10^{5} and good agreement with analytical solutions is reported. Later, the diffusion of an initially square patch of vorticity and flow around a wedge were added to the test cases [200]. These authors formulate the diffusion velocity concept a little differently, starting from the definition of circulation around a curve which encloses surface S

$$\Gamma = \int_S \boldsymbol{\omega} \cdot \hat{\mathbf{n}} dS. \tag{2.29}$$

Defining $\tilde{\mathbf{u}}$ to be the velocity of the area occupied by vorticity, they take the time derivative of the above equation to obtain

$$\frac{d\Gamma}{dt} = \int_S \left[\frac{\partial \boldsymbol{\omega}}{\partial t} + (\tilde{\mathbf{u}} \cdot \nabla)\boldsymbol{\omega} + \boldsymbol{\omega}(\nabla \cdot \tilde{\mathbf{u}}) - (\boldsymbol{\omega} \cdot \nabla)\tilde{\mathbf{u}} \right] \cdot \hat{\mathbf{n}} dS = 0, \tag{2.30}$$

which is true when the integrand is zero:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\tilde{\mathbf{u}} \cdot \nabla)\boldsymbol{\omega} + \boldsymbol{\omega}(\nabla \cdot \tilde{\mathbf{u}}) - (\boldsymbol{\omega} \cdot \nabla)\tilde{\mathbf{u}} = 0. \tag{2.31}$$

Let $\tilde{\mathbf{u}} = \mathbf{u} + \mathbf{u}_d$, where $\mathbf{u}$ is the fluid velocity and $\mathbf{u}_d$ is the diffusion velocity, and subtract the three-dimensional vorticity transport equation (1.1) from (2.31), to obtain the following governing equation for the diffusion velocity:

$$(\mathbf{u}_d \cdot \nabla)\boldsymbol{\omega} + \boldsymbol{\omega}(\nabla \cdot \mathbf{u}_d) - (\boldsymbol{\omega} \cdot \nabla)\mathbf{u}_d = -\nu\Delta\boldsymbol{\omega}. \tag{2.32}$$

Using the solenoidal property of the vorticity and some vector identities, the above equation is written in the following form, where the diffusion velocity appears once,

$$-\nabla \times (\mathbf{u}_d \times \boldsymbol{\omega}) = \nu\nabla \times (\nabla \times \boldsymbol{\omega}). \tag{2.33}$$

The above equation implies

$$\mathbf{u}_d \times \boldsymbol{\omega} = -\nu\nabla \times \boldsymbol{\omega} + \nabla\phi. \tag{2.34}$$

The gradient of the arbitrary scalar field, $\nabla\phi$, can be shown to be equal to zero. At this point the authors of [199] restrict themselves to two dimensions and obtain the definition of diffusion velocity as expressed by (2.27). They add a further simplification by assuming a thin layer where the diffusion in the direction tangential to the wall can be neglected. But the presentation of the diffusion velocity in this way aids in the extension of the scheme to three dimensions.

The extension of the diffusion velocity concept to three-dimensional flows is introduced in [165], and applied to the evolution of an axisymmetric vortex ring with a small periodic perturbation. Since the vorticity transport equation in 3D is a vector equation, the concept of viscous rotation needs to be added to the viscous velocity. These authors decompose the vorticity field in two parts, denoted by $\boldsymbol{\omega}_t$ and $\boldsymbol{\omega}_n$, which are defined locally at point $\mathbf{x}_\circ$ by

$$\boldsymbol{\omega}_t(\mathbf{x}) = \frac{\boldsymbol{\omega}(\mathbf{x}) \cdot \boldsymbol{\omega}(\mathbf{x}_\circ)}{|\boldsymbol{\omega}(\mathbf{x}_\circ)|^2}\boldsymbol{\omega}(\mathbf{x}_\circ), \tag{2.35}$$

$$\boldsymbol{\omega}_n(\mathbf{x}) = \boldsymbol{\omega}(\mathbf{x}) - \boldsymbol{\omega}_t(\mathbf{x}). \tag{2.36}$$

The decomposition above is used to express the diffusion term $\nu\Delta\boldsymbol{\omega}$ in two parts:

$$\nu\Delta\boldsymbol{\omega} = \nu(\Delta\boldsymbol{\omega}_t + \Delta\boldsymbol{\omega}_n). \tag{2.37}$$

The first component of the diffusion term represents a longitudinal diffusion, preserving the structure of the vortex elements. The second part represents a rotation of the vorticity vectors. Taking into account this decomposition, the vorticity transport equation may be written in the following form:

$$\begin{aligned}\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \boldsymbol{\omega}) - (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} &= \nu(\Delta \boldsymbol{\omega}_t + \Delta \boldsymbol{\omega}_n) \\ &= -\nabla \cdot (\mathbf{u}_d \otimes \boldsymbol{\omega}) + \boldsymbol{\omega}_d \times \boldsymbol{\omega}\end{aligned} \tag{2.38}$$

$$\text{where:} \qquad \mathbf{u}_d = -\nu \frac{(\nabla \otimes \boldsymbol{\omega}) \cdot \boldsymbol{\omega}}{|\boldsymbol{\omega}|^2} \qquad \text{and,} \qquad \boldsymbol{\omega}_d = -\nu \frac{\Delta \boldsymbol{\omega}_n \times \boldsymbol{\omega}}{|\boldsymbol{\omega}|^2}. \tag{2.39}$$

Now the vorticity transport equation can be written in the following form:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla((\mathbf{u} + \mathbf{u}_d) \otimes \boldsymbol{\omega}) - (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} - \boldsymbol{\omega}_d \times \boldsymbol{\omega} = 0. \tag{2.40}$$

This form of the Navier-Stokes equation is now a nonlinear advection equation, *i.e.*, the diffusion effect was written as a convection term just like in the two-dimensional diffusion velocity method. The first two terms are interpreted as a convective derivative, the third term is still vortex stretching/tilting, and the last term expresses a rotation applied to the vorticity field. At this point, the Lagrangian particle discretization can be applied and the problem is translated into a set of ordinary differential equations. The Lagrangian system is

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}_p + \mathbf{u}_d \tag{2.41}$$

$$\frac{d\boldsymbol{\omega}_p}{dt} = (\boldsymbol{\omega}_p \cdot \nabla)\mathbf{u}_p - \boldsymbol{\omega}_d \times \boldsymbol{\omega}_p. \tag{2.42}$$

The diffusion velocity in the above system is obtained by a discrete form of equations (2.39), using the discrete vorticity field (1.2) and approximating the term $\nabla \times \boldsymbol{\omega}$ by

$$\nabla \times \boldsymbol{\omega}(\mathbf{x}) \approx \sum_i \boldsymbol{\Gamma}_i \times \nabla \zeta_\sigma(\mathbf{x}_i - \mathbf{x}). \tag{2.43}$$

Finally, the calculation of the *diffusion vorticity* or viscous rotation $\boldsymbol{\omega}_d$ requires the approximation of the Laplacian $\Delta \boldsymbol{\omega}_n$, for which the authors of [165] propose the direct discretization of the integral approximation, as used in the PSE method for the complete diffusion operator.

Investigations into the use and extension of the diffusion velocity concept continue and are plentiful, including a specific method for axisymmetric flows [164], application to Fokker-Planck equations [117], and an extension of the method to the case of anisotropic diffusion [22].

2.1.8 Core Spreading

At the turn of the 1990's, only two viscous vortex methods were well known: the random vortex method of Chorin [42] and core spreading, which apparently was used for the first time in [109]. The latter method was, however, under discredit due to the mathematical objections of Greengard

[79]. Into this stage, many new performers were introduced during the decade: particle strength exchange and general particle methods, vortex redistribution method, Fishelov method, diffusion velocity method. They all boasted their ‘deterministic’ nature, and many workers compared results with RVM and were satisfied. Then, a correction was suggested for the core spreading method by Rossi [170], and interest in this scheme was revived, but still only a very few workers have performed further investigations with it.

The core spreading method, presented by Leonard in [113], is a purely Lagrangian scheme that accounts for viscous effects by changing the *core size* of the particles to exactly solve the diffusion equation. It is easily understood using as analogy the classical exact solution of the two-dimensional Navier-Stokes equations termed “spreading line vortex” (see Batchelor [9], p. 204). In this problem the vorticity is given by

$$\omega(\mathbf{x},t) = \frac{\Gamma}{4\pi\nu t}\exp -\frac{|\mathbf{x}|^2}{4\nu t}. \tag{2.44}$$

Consider again the approximate vorticity given by the discretized form (1.2), but write it slightly differently to express the fact that the core function will now carry the dependence on time,

$$\omega(\mathbf{x},t) \approx \omega^h(\mathbf{x},t) = \sum_{i=1}^{N} \Gamma_i\, \zeta_t(\mathbf{x}-\mathbf{x}_i(t))\,. \tag{2.45}$$

The core function is now chosen to be the solution of the heat equation with initial data ζ_o

$$\zeta_t(\mathbf{x}) = \frac{1}{4\pi\nu t}\int e^{-(\mathbf{x}-\mathbf{y})^2/4\nu t}\zeta_o(\mathbf{y})d\mathbf{y} = (G * \zeta_o)(\mathbf{x}), \tag{2.46}$$

where G is the heat kernel. If the initial distribution function ζ_o is a Dirac delta, then

$$\zeta_t(\mathbf{x}) = \frac{1}{4\pi\nu t}e^{-(\mathbf{x})^2/4\nu t}. \tag{2.47}$$

By comparing with (2.44), the discretized vorticity field in two dimensions can be seen as a superposition of “spreading line vortices” of different circulation strengths. The core spreading vortex method is then formulated so as to satisfy identically the viscous part of the vorticity equation by expanding σ^2 linearly according to

$$\frac{d\sigma^2}{dt} = 4\nu, \tag{2.48}$$

which means that the core of each particle must spread out at a rate proportional to $\sqrt{\nu t}$, or $\sqrt{\nu\Delta t}$ at each time step. As we use a Gaussian blob function with $k=2$, the method is expressed in the following simple algorithmic rule:

$$\sigma_i^2\,(t+\Delta t) = \sigma_i^2(t) + 2\nu\Delta t, \qquad i = 1\cdots N \tag{2.49}$$

The attraction of this formulation —apart from its utter simplicity of implementation— is that the method is fully localized and grid-free in nature (hence more easily expressed in a parallel application code), and that it is fully deterministic (so allows faster convergence and better error control compared with the RVM). Additionally, a core spreading scheme does not necessarily rely on the fractional step method. However, as mentioned, research in its regard was largely stalled when the method was declared inconsistent in 1985 [79], and it was proved that the scheme converges to an equation different from the Navier-Stokes equations.

The inconsistency of the core spreading method is related to the treatment of the particles as solid bodies, from which the "convection error" arises. With the core spreading method the diffusion of vorticity is approximated accurately, but the vorticity is advected with an average velocity and not with the actual local velocity. The vorticity is incorrectly convected even in the limit of infinitely many particles. Greengard derives in his note [79] the actual equation which is solved by the vorticity obtained by core spreading, which differs from the Navier-Stokes equation in the convection term only. But it is easy to understand the problem with a simple argument, noting that with the uncorrected method particle cores will grow to a size of at least $\sqrt{\nu T}$ in a simulation ran to a final time T. As the convergence of a vortex method depends on core size remaining small, the method will clearly eventually break down. This simple argument illuminates how a correction is implemented, based on adding spatial refinement, *e.g.*, splitting of the blobs which have grown beyond a specified maximum into smaller elements. Rossi [170] proved the convergence of the corrected core spreading method with vortex splitting and provided details of implementation. He went on to propose vortex merging as a means to control problem size [171], as the splitting can rapidly increase the blob population. The vortex splitting scheme of Rossi will be described in more detail in §2.2.

The core spreading method of Rossi has been used in unbounded flow only. Shiels [193] incorporated boundaries and tested his method on the flow around a circular cylinder, as his goal was the simulation of bluff body flows. He also constructed rules to provide for variable spatial resolution and hence increase efficiency by having coarser blob population far from the body, but many *ad hoc* numerical parameters were introduced. Shiels' experiments with a circular cylinder at Re=100 and 3000 were promising, as an accuracy comparable to the PSE method was reported, while at a much lower computational cost. His preliminary simulations at high Reynolds number (15,000) show general qualitative agreement with benchmark PSE computations but the resolution is not comparable. Still, the core spreading method evidences considerable potential for improvement and the work described above has broken new ground. The higher Reynolds number simulations were published recently [194], in what is probably one isolated application of the core spreading method with boundaries.

There are no extensions to three dimensions of either the method of Rossi or of Shiels. Rossi is proposing an enhancement of his core spreading method by using deformable elliptic blobs [172, 173], which provide higher order of spatial accuracy. It does seem that his elliptic blobs are farther from an extension to three dimensions, as complicated expressions are used for the calculation of the Biot-Savart velocity using the anisotropic elements. Rossi could not express the Biot-Savart integral in terms of elementary functions with his elliptical Gaussian blobs, so he used asymptotic approximations. Although it would be possible to carry out the same sort of asymptotics for the three dimensional case, there are many mathematical challenges, as Rossi acknowledges in [172].

Perhaps it is less known that Japanese scientists have investigated with the core spreading scheme for quite some time. Before Rossi's celebrated correction on the method, there were remarks in the Japanese literature regarding the validity of core spreading within a finite time [138], which also leads to correction concepts. There are early attempts at constructing a three-dimensional application of the core spreading method in [139], and improved treatment of boundaries for simulation of bluff body flow in [148]. In this three-dimensional core spreading the stretching and diffusion effects are separately considered to effect a change in the core radius of particles:

$$\sigma^{n+1} = \sigma^n + \left(\left.\frac{d\sigma}{dt}\right|_{\text{STRETCH}} + \left.\frac{d\sigma}{dt}\right|_{\text{DIFFUSE}} \right) \Delta t. \tag{2.50}$$

For the account of the different contributions it is now necessary to replace the generic "core size" by two dimensions, the core radius σ and the blob length l which is in the direction of the vorticity vector. At the beginning of a time step the blobs have spherical symmetry so that $l = 2\sigma$. The change in core radius due to the stretching term is obtained from

$$\frac{d\boldsymbol{\omega}}{dt} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} \tag{2.51}$$

$$\frac{dl}{dt} = \frac{l_t}{|\boldsymbol{\omega}_t|} \left| \frac{d\boldsymbol{\omega}}{dt} \right| \tag{2.52}$$

$$\left.\frac{d\sigma}{dt}\right|_{\text{STRETCH}} = -\frac{\sigma_t}{2l_t}\frac{dl}{dt}. \tag{2.53}$$

The authors of [139] use a somewhat bizarre version of the core spreading due to diffusion, given by

$$\left.\frac{d\sigma}{dt}\right|_{\text{DIFFUSE}} = \frac{c^2\nu}{2\sigma_t}, \tag{2.54}$$

where the constant c has the value 2.242, which they claim is obtained from the exact solution of a diffusing infinitely long vortex filament. Now, updated values for blob length and vorticity are obtained using the expression (2.50) and

$$l_{t+\Delta t} = l_t + \frac{dl}{dt}\Delta t \tag{2.55}$$

$$|\boldsymbol{\omega}_{t+\Delta t}| = |\boldsymbol{\omega}_t| \left(\frac{\sigma_{t+\Delta t}}{\sigma_t} \right)^2. \tag{2.56}$$

Finally, the new element is replaced by a spherical blob of equivalent volume. The particulars of the derivation of some expressions in this method are not very clear in the papers consulted, while many of the references therein were not available to us, so we are not able to express confidence in the method as yet. However, the results presented in [148] do seem to simulate well the flow features, such as boundary layer separation and development of the vortical wake, for the cases studied. These correspond to the flow around a sphere at Reynolds numbers of 300 and 1000, and the flow around a prolate spheroid after impulsive start at Re=1000 for angles of attack of 0, 10 and 30 degrees. The main contribution of [148] over [139] seems to be a different scheme for vorticity creation at the boundary, while the three-dimensional core spreading method used in both cases is the same.

The two-dimensional core spreading method was given an alternative mathematical treatment which is interesting [99]. It consists of deriving a Fredholm integral equation for a function which represents a sort of “vorticity flux” supplied due to the difference in convection velocity. After this mathematical derivation, the Japanese researchers propose the existence of higher-order core spreading schemes. The mathematical derivation starts from the two-dimensional vorticity equation (2.1) and introduces the Lagrangian variable

$$\mathbf{X} = \mathbf{x} - \Phi_t(\mathbf{a}), \tag{2.57}$$

where $\Phi_t(a)$ represents the approximate trajectory of a fluid particle which is initially at point $\mathbf{a}$, assumed to be approximately convected by a velocity $\hat{\mathbf{u}}(\mathbf{x}, t)$. Using the new coordinate system, the vorticity equation is expressed as

$$\frac{\partial \omega}{\partial t} = \nu \frac{\partial^2 \omega}{\partial X_i \partial X_i} + f(\mathbf{X}, \mathbf{a}, t), \tag{2.58}$$

where $\omega = \omega(\mathbf{X}, \mathbf{a}, t)$, summation convention is assumed on the index i, and $f(\mathbf{X}, \mathbf{a}, t)$ is the correction term due to the difference between the approximate velocity $\hat{\mathbf{u}}$ and the exact velocity $\mathbf{u}$. The solution of (2.58) in term of the Green’s function is written as

$$\omega(\mathbf{x}, t) = \omega_o(\mathbf{a}) G\left(t, |\mathbf{x} - \Phi_t(\mathbf{a})|\right) + \int_0^t d\tau \int_D f(\mathbf{X}', \mathbf{a}, \tau) G\left(t - \tau, |\mathbf{x} - \Phi_t(\mathbf{a}) - \mathbf{X}'|\right) d\mathbf{X}', \tag{2.59}$$

where $D = \mathbb{R}^2$ and

$$\begin{aligned} G(t, |\mathbf{x}|) =& \frac{1}{\pi} \frac{1}{\sigma_o^2 t} \exp\left(-\frac{|\mathbf{x}|^2}{\sigma_o^2 t}\right) \\ \sigma_o =& \sqrt{4\nu}. \end{aligned} \tag{2.60}$$

This solution corresponds to the vorticity of the one fluid particle considered. If the total initial vorticity has compact support S_o, it is assumed that the total vorticity field is obtained by superimposing the solution (2.59) over S_o with respect to $\mathbf{a}$, so that it can be written as

$$\begin{aligned}\omega(\mathbf{x},t) = &\int_{S_o} \omega_o(\mathbf{a}) G\left(t, |\mathbf{x}-\Phi_t(\mathbf{a})|\right) d\mathbf{a} \\ &+ \int_{S_o} d\mathbf{a} \int_0^t d\tau \int_D f(\mathbf{X}',\mathbf{a},\tau) G\left(t-\tau, |\mathbf{x}-\Phi_t(\mathbf{a})-\mathbf{X}'|\right) d\mathbf{X}'. \qquad (2.61)\end{aligned}$$

The above is only an approximate solution to the vorticity transport equation, but it suggests to the authors of [99] that the exact solution can be expressed in the same integral form, only substituting the function $f(\mathbf{X},\mathbf{a},t)$ by an unknown function $\hat{f}(\mathbf{X},\mathbf{a},t)$ which must be determined from the Navier-Stokes equations. The problem is now to find the governing equation for $\hat{f}$, which —after a significant amount of algebra work— turns out to be the following integral equation:

$$\begin{aligned}\hat{f}(\mathbf{X},\mathbf{a},t) = &\frac{\partial}{\partial \mathbf{X}_i}\Big(\omega_o(\mathbf{a}) G\left(t, |\mathbf{X}|\right) \left[\hat{u}_i(\Phi_t(\mathbf{a}),t) - u_i(\mathbf{X}+\Phi_t(\mathbf{a}),t)\right]\Big) \\ &+ \frac{\partial}{\partial \mathbf{X}_i}\Big(\left[\hat{u}_i(\Phi_t(\mathbf{a}),t) - u_i(\mathbf{X}+\Phi_t(\mathbf{a}),t)\right] \qquad (2.62) \\ &\times \int_0^t d\tau \int_D \hat{f}(\mathbf{X}',\mathbf{a},t) G\left(t-\tau, |\mathbf{X}-\mathbf{X}'|\right) d\mathbf{X}'\Big)\end{aligned}$$

A new scalar function Ω is defined by

$$\hat{f}(\mathbf{X},\mathbf{a},t) = \nabla \cdot \big(\left[\hat{\mathbf{u}}(\Phi_t(\mathbf{a}),t) - \mathbf{u}(\mathbf{X}+\Phi_t(\mathbf{a}),t)\right] \Omega(\mathbf{X},\mathbf{a},t)\big), \qquad (2.63)$$

so that the governing equation for Ω is the following:

$$\begin{aligned}\Omega(\mathbf{X},\mathbf{a},t) = \omega_o(\mathbf{a}) G(t,|\mathbf{X}|) + \int_0^t d\tau \int_D \frac{\partial}{\partial \mathbf{X}'_i}\Big(&\left[\hat{u}_i(\Phi_t(\mathbf{a}),t) - u_i(\mathbf{X}+\Phi_t(\mathbf{a}),t)\right] \\ &\times \Omega(\mathbf{X}',\mathbf{a},t)\Big) G(t-\tau, |\mathbf{X}-\mathbf{X}'|) \, d\mathbf{X}'. \qquad (2.64)\end{aligned}$$

Looking at this last equation and comparing with (2.61) it is seen that

$$\omega(\mathbf{x},t) = \int_{S_o} \Omega(\mathbf{x}-\Phi_t(\mathbf{a}),\mathbf{a},t) \, d\mathbf{a} \qquad (2.65)$$

and, as remarked in [99], the first approximation of (2.64) would be equivalent to a classic core spreading method. An alternative approach is suggested by this analysis which is termed "higher-order core spreading" by the authors and is based on the "reformation of the discretized vorticity field at each time step", *i.e.*, the discretized vorticity field is transformed according to

$$\hat{\omega}(\mathbf{x}, t_{n+1}) = \int_D \hat{\omega}(\mathbf{a}, t_n) G(t, |\mathbf{x} - \Phi_{t_{n+1}}(\mathbf{a})|)\, d\mathbf{a}. \tag{2.66}$$

This looks very much like a resampling method, as described in the beginning of §5. Not many details of the algorithm are given in [99], but it is stated that the circulation of each particle "is not conserved and is set at every time step". This seems to be saying that there is an exchange of circulation between the vortices incorporated into the algorithm. Convergence and stability properties of both the classic core spreading method, called "Algorithm A", and the higher order method, called "Algorithm B", are analyzed in [99], where the latter is shown to converge to the Navier-Stokes equations (unlike the first one which as noted by Greengard converges to a different equation). Error bounds are given and also a simple one-dimensional numerical example is performed, where both methods are seen to give reasonably good results for short times. In a separate publication, the research group in Osaka reports two-dimensional numerical experiments of a simple Oseen-type initial vorticity, and again concludes that both their Algorithms A and B are suitably accurate within a small lapse of time [100]; however, the numerical experiment is of quite low resolution. In later work [98], additional numerical experiments were reported, this time adding the presence of a solid boundary by coupling with a panel method and including boundary layer considerations in the line of [39]. The treatment of the boundary in this approach incorporates vortex sheet elements within a stipulated boundary layer thickness, which transfer the vorticity to the rest of the flow with the creation of new vortex blobs. A separate solution for the flow inside and outside of the boundary layer is obtained, and the two solutions are matched at their edges. In any case, the presentation in the above mentioned publications is not easy to follow and we suggest studying it in more detail at a later time. As well, it would be interesting to complement by looking at other publications by the same authors and their most recent work, since the mathematical approach they present is quite interesting even if their numerical examples are not striking.

2.1.9 Least-Squares and Triangulated Vortex Methods

There are at least two more deterministic approaches for inclusion of diffusion in vortex methods. As we remark below, they have been later combined together. The first approach is the so-called "free Lagrangian" method [176], which is based on a discrete approximation of the Laplacian on an irregular grid. A Voronoi triangulation is built using as nodes the particle positions, and discrete differential operators are approximated on this mesh. The construction of a Voronoi diagram can be computationally expensive, and it has to be updated at every time step as the particle positions are advected. In subsequent work [177] a fast method is used to construct a Delaunay triangulation that is of $O(N \log N)$, incorporating also a fast summation technique for the velocity evaluation, but restricting this time to inviscid flows. (A Delaunay triangulation is not the same as the Voronoi diagram, but one can be obtained from the other). This last work includes commentaries on how to generalize the fast triangulated vortex method to viscous flows, flows with boundaries and three-dimensional flows, but neither of these generalizations was carried out. Actual application of the Delaunay triangulated vortex method to two-dimensional viscous flows with solid boundaries is presented in [89, 90], where the concept of diffusion velocity is incorporated as well as a coupled panel method. This work also integrates a least-squares approach to calculate derivatives, described below, hence the combined use of the two methods that we remarked upon.

The other concept used in vortex methods for inclusion of viscous effects is the "moving least-squares method" of [126]. The authors were motivated by the loss of accuracy of both the Fishelov method and PSE when the particles become irregularly spaced, and wanted to provide an alternative means of obtaining the derivatives of vorticity. The least-squares method consists of fitting a polynomial of order two to the vorticity field in a neighbourhood of the point where the derivative is desired. When the points are regularly spaced and only close neighbours are considered, the approach is equivalent to a centered difference. The drawback is having to solve a two-by-two linear system per particle, at each time step. Consider this one-dimensional example, define $\zeta_m(x)$ as the approximation to the vorticity $\omega(x)$ in the neighbourhood of a control point x_m, denote $\omega_m \equiv \omega(x_m)$, and write

$$\zeta_m(x) = \omega_m + B_m(x - x_m) + C_m(x - x_m)^2, \tag{2.67}$$

where the constants B_m and C_m correspond to approximations of the first and second derivatives of the vorticity at the control point. Define the error at the control point by

$$J_m \equiv \sum_{n=1}^{N} L_{nm} \left[\omega_n - \zeta_m(x_n)\right]^2 . \tag{2.68}$$

The coefficients L_{nm} are used to establish the width of the neighbourhood around the control point. Defining a length scale δ_m to set up the locality of L_{nm}, the authors of [126] use

$$L_{nm} = \exp\left[\frac{-(x_n - x_m)^2}{\delta_m^2}\right] . \tag{2.69}$$

Now, minimizing the error, the following equations are obtained for the coefficients B_m and C_m

$$\sum_{n=1}^{N} L_{nm}(x_n - x_m)^i \left[\omega_n - \zeta_m(x_n)\right] = 0, \qquad i = 1, 2. \tag{2.70}$$

The least-square method is basically a way for calculating derivatives. Once a polynomial function has been fit onto the control points in the neighbourhood of the point of interest, then the derivatives of vorticity at this point are approximated by differentiating the polynomial fit. The method was tested in [126] by measuring the *RMS* error in the second derivative of a one-dimensional vorticity field (a simple Gaussian) and plotting *versus* the overlap ratio. This error was compared with the Fishelov method and PSE for different irregular grids. The results show how the error increases severely with both Fishelov and PSE methods when the grid becomes irregular. In contrast, the least-squares calculation remains accurate with very irregular grids. For tests using an error-function initial vorticity field, it was necessary to add image vortices near the discontinuity (which is analogous to a "wall") at $x = 0$. The results showed that the least-squares calculation exhibits errors comparable to a central difference formula when sufficient images are included.

The application of the least-squares approach to two- or three-dimensional vortex methods was implemented by combining it with a diffusion velocity concept which is used to spread the vorticity support. The details of the full-fledged vortex method are not given in [126], which basically reports the preliminary one-dimensional numerical experiments to obtain error measurements. The implementation of the vortex method is only sketched and described to be underway. But in [127] it is shown how the diffusion velocity method is combined with the moving least-squares scheme to calculate derivatives, although this work is restricted by the simplifications of axial symmetry.

Recently, the authors of the moving least-squares method have combined it with the triangulation method of [177]. In [128] a three-dimensional viscous vortex method is described which relies on the moving least-squares approach to approximate the derivatives for calculation of the stretching and diffusion terms, but instead of using vortex blobs the vorticity field is interpolated on a tetrahedral mesh that is fitted to the Lagrangian points. We stress that the triangulated methods are *not* vortex blob methods, and therefore do not require cutoff functions. Hence, the advantage is attained that the vorticity field does not penetrate the surface of the body. Additionally, it is possible to have node points in a distribution with high anisotropy in one direction, which is advantageous when boundary layer flows are computed. The method —given the name "tetrahedral vortex element" or TVE method— was benchmarked on the flow past a sphere at Reynolds number of 100, and has recently been applied to a more complicated three-dimensional flow in [78] with notable results. But perhaps one could consider this a method closer to vortex-in-cell than to vortex blob methods, where the mesh is now unstructured. It is clearly not a grid-less method, as a tetrahedral mesh is constructed and fit to the Lagrangian points on each time step.

In conclusion, with so many different approaches to construct a viscous vortex method, it is clear that the field is still maturing and is yet far from a consensus in regards to the "best" viscous scheme. Each method reviewed above has some desirable characteristics as well as some disadvantages. Particle strength exchange is very sensitive to having an ordered distribution of particles, and so there has been a great amount of work on remeshing schemes. The method of superposing derivatives of the cutoff function, as used by Anderson and Greengard for vortex stretching and Fishelov for diffusion, is also dependent on a regular particle arrangement to maintain accuracy; diffusion velocity, as well, requires constant overlap of blobs. It would seem that the problem for an accurate viscous vortex method is not the viscous scheme, but spatial adaption. Indeed, to liberate themselves from the problem generated by irregular particle fields, the authors of the vortex redistribution method resort to elaborate and computationally expensive algorithms, while the tetrahedral vortex element method does away with the particle representation altogether (at the cost of constructing a mesh at every time step). Even the core spreading method suffers from the problem of how to limit the size that vortex blobs can grow, so needs a form of spatial adaption. But in contrast to the other methods, core spreading is utterly simple in its approach to satisfying diffusion effects. If the spatial accuracy can be maintained with some form of adaptive refinement, the core spreading method seems to offer the opportunity of a truly grid-less viscous vortex method.

2.2 Vortex Blob Discretization and Spatial Adaption

2.2.1 Discretization and Vortex Method Initialization

Upon initializing a vortex method calculation, one needs to obtain the identifying quantities of the vortex particles, *i.e.*, their location and circulation strength. Their core size is chosen initially as a discretization parameter, dictating the resolution of the calculation. As has been discussed, the discretized vorticity field is expressed in two dimensions as the aggregation of the vortex particles in the following way:

$$\omega^h(\mathbf{x},t) = \sum_{i=1}^{N} \Gamma_i \, \zeta_\sigma \left(\mathbf{x} - \mathbf{x}_i\right), \tag{2.71}$$

where Γ_i is the scalar circulation strength of the two dimensional particle i. (In three dimensions the particle strengths become vector quantities, and the discretized vorticity is the vector field obtained by superposition of the vector vortex particles.)

The initial particle locations are most commonly chosen to be on a Cartesian mesh. An alternative could be to divide the support of the initial vorticity in cells of uniform size and initialize a particle in a random location inside the cell; this has been called "quasi-random" initialization [48], while a random initialization will not construct any form of ordered lattice. On a square lattice of initial particle locations, one can assign the circulation values by simply evaluating the vorticity at the particle location and multiplying by the cell area (or volume in 3D). That is,

$$\Gamma_i^o = \omega_i^o h^d = \omega(\mathbf{x}_i, t=0)\, h^d. \tag{2.72}$$

For the random or quasi-random initializations one could use again the local value of vorticity, and multiply by the average cell area (obtained by dividing the support of the vorticity by the number of particles, N). This approach is less accurate but can be preferable if the vorticity is not smooth; but the initialization on a square lattice and the use of (2.72) is the most commonly used initialization method. Note, however, that the use of (2.72) to initialize the particle strengths incurs an error related to the midpoint rule; one can try to decrease this error by iterating on the Γ_i's, for example (as will be discussed in detail later).

Indeed, the particle representation of the initial vorticity field incorporates a sort of numerical diffusion when (2.72) is used; this 'diffusive effect' of the discretization can be quantified for the particular case of Gaussian blobs used on flows which are *exact solutions* of the diffusion equation. This can be seen by writing the general solution of the heat equation as (in one dimension, for simplicity)

$$g(x,t) = \frac{1}{\sqrt{4\pi\nu t}} \int_{-\infty}^{\infty} g(x',t_o) \exp\left(-\frac{|x'-x|^2}{4\nu t}\right) dx'. \tag{2.73}$$

Writing the discrete representation of the integral above using single interval extrapolative rule, it can be seen that if one spatially discretizes the function $g(x,t)$ using Gaussian cores (with $k=2$) the summations obtained on the left and right hand sides of the equation can be made equivalent by making $\sigma^2 = 2\nu t$. Hence, the discretization reconstructs not the initial vorticity but this vorticity field as if it had diffused for a time interval of $\sigma^2/2\nu$.

As a result, a scheme for improving the accuracy of initialization for test problems based on exact solutions of the diffusion equation —as is the case of the Lamb-Oseen vortex, the classic test problem which will also be used in this investigation— is the application of a numerical "anti-diffusion" process, equivalent to shifting the initial time backwards by an amount $\sigma^2/2\nu$. Hence, the initial circulation strengths for a Lamb-Oseen vortex at time-zero, t_o, would be obtained by

$$\Gamma_i^o = \omega_i^o h^2 = \frac{\Gamma_0 h^2}{4\pi\nu(t_o - \sigma^2/2\nu)} \exp\left(-\frac{x_i^2 + y_i^2}{4\nu(t_o - \sigma^2/2\nu)}\right). \tag{2.74}$$

This "time-shift correction" can be applied in the very particular situation of discretizing an exact solution of the diffusion equation using Gaussian blobs. As such, it will not be usable in a practical application; it allows, however, the production of accurate initialization of the Lamb vortex for the purposes of studying overlap dependence and the effects of the particle distribution. This will be useful in the numerical investigation of Chapter 3 regarding the accuracy of the blob discretization.

As some example calculations given in Chapter 3 will show, the error of initialization with Equation (2.72) can be considerable, at least when using simple Gaussians or other cutoff functions of low order. For this reason, one finds that in the calculations of [102] the vortex method is initialized by solving, for unknown Γ_i's, the following set of equations:

$$\omega(\mathbf{x}_j, 0) = \sum_{i=1}^{N} \Gamma_i \, \zeta_\sigma(\mathbf{x}_j - \mathbf{x}_i), \quad \text{for } j = 1, \ldots, N \tag{2.75}$$

Koumoutsakos [102] reports that the system can be solved with the method of successive over-relaxation (SOR), using a coefficient in the range of 0.3 to 0.4 (*i.e*, under-relaxed). In another example of formulating a linear system for the accurate discretization of an initial condition, a procedure is developed in [115] which is conditional of having a regular, square lattice of particles. This condition allows the authors to construct an exact inverse, which is unfortunately not possible to implement in an actual calculation due to the global influence. Therefore, a preconditioner is built based on a "best compact approximation" to the inverse, and an iterative scheme similar to the Jacobi method is used to solve the system of equations.

The subject of this linear system will come up again in §2.2.3, where Beale's method of circulation processing is presented, and again later in the discussion of radial basis functions. It will be seen that there is a close relationship between the initialization of the vortex method and the solution of a radial basis function interpolation problem. Hence, many recent developments in the latter subject will aid in conquering what was qualified as "still a current research topic" in [48] (p. 211), namely, the efficient inversion of the system (2.75).

2.2.2 Rezoning

Most vortex method application programs incorporate spatial adaptation in the form of "remeshing" or "rezoning" algorithms, which consist in re-starting the particle field on a regular grid every few time steps, and re-calculating the particle circulation strengths by interpolation or other means. This subject constitutes an important active area of research in vortex methods.

The first attempt at controlling Lagrangian distortion errors was termed "rezoning" [16], and it was suggested in the context of high-order vortex methods. As will be shown with experiments later, the high-order blob kernels are more vulnerable to the disorder in the particle field than their lower-order counterparts, and this was recognized upon their introduction. Rezoning consists in re-starting the particles on a regular mesh, and obtaining the vorticity at the new particle locations as the value induced by the sum of the old blobs, to be discarded. In other words, one defines a continuous vorticity field using the current particle distribution, and then evaluates this function on any new blob location $\tilde{\mathbf{x}}$, *i.e.*, summing over all j current blobs:

$$\omega_{\text{comp}}(\tilde{\mathbf{x}}, t) = \sum_j \zeta_j(\tilde{\mathbf{x}} - \mathbf{x}_j)\, \omega_j h^2. \tag{2.76}$$

After this, the vortex method is re-initialized by obtaining the new circulation strengths using the standard initialization formula (2.72), on the new particle locations.

Using a naïve approach to evaluate the function (2.76) at a new blob location on a square mesh, the vorticity contribution of each old particle would be added, requiring $O(N \cdot M)$ operations for N old particles and M new ones. This approach is expensive computationally, and although in theory one could implement some form of fast summation, the literature does not to our knowledge show any such implementation. A criterion for determining how often to execute a rezoning process was introduced in [144], but it is based on calculating at every time step the "vorticity error along vortex paths", which is just as expensive. On the other hand, if one is to obtain the new circulation strengths using the calculated vorticity values at the new particle locations and equation (2.72), then an error equivalent to the initialization error is incurred. For this reason, it seems that rezoning as described by Beale should be applied only when using high-order blobs. This will be demonstrated later with numerical experiments.

2.2.3 Beale's Method of Circulation Processing

An alternative approach to control the effects of Lagrangian deformation, not based on restarting the particle field, is the processing of the particle circulations to improve the way the discretized vorticity field approximates the exact one at the particle locations. The concept of recalculating the circulation values can be understood as a way of adjusting the particle volumes to account for the changes in overlap. To improve on the approximation given by (1.2) at a given time, one may search for new values of the circulation strengths γ_i's so that

$$\sum_{i=1}^{N} \gamma_i \, \zeta_\sigma \left(\mathbf{x}_j(t) - \mathbf{x}_i(t)\right) = \omega_j, \tag{2.77}$$

where $\omega_j = \omega(\mathbf{x}_j, t)$ is the local vorticity value at the particle location, *i.e.*, the value one wishes to approximate. Multiplying this equation by h^2, representing the volume of a blob in a regular, square mesh, one obtains

$$h^2 \sum_{i=1}^{N} \gamma_i \, \zeta_\sigma \left(\mathbf{x}_j(t) - \mathbf{x}_i(t)\right) = \Gamma_j, \tag{2.78}$$

where the Γ_j's are the known, current circulation values of the blobs. Equation (2.78) represents a linear system for the coefficients γ_i, which can be written in matrix form as

$$A\boldsymbol{\gamma} = \boldsymbol{\Gamma}, \tag{2.79}$$

where, $A_{ij} = h^2 \zeta_\sigma(\mathbf{x}_j(t) - \mathbf{x}_i(t))$. An iterative method to solve this system was proposed by Beale [12], who observed that the previous circulation values are a natural first guess for the γ_i's. First rewriting the system (2.79) as

$$(A - I)\boldsymbol{\gamma} + \boldsymbol{\gamma} = \boldsymbol{\Gamma}, \tag{2.80}$$

then Beale's iterative method is expressed by the following rule:

$$\gamma_i^{n+1} = \Gamma_i + \gamma_i^n - \sum_j h^2 \gamma_j^n \zeta_\sigma \left(\mathbf{x}_j(t) - \mathbf{x}_i(t) \right). \tag{2.81}$$

This iterative method was used at every time step in [12], as well as in [40] in combination with the PSE viscous scheme. However, Beale's method is not guaranteed to converge, and in the example given in p. 209 of [48] one can see that even though an improvement of two orders of magnitude in the velocity accuracy is obtained at time-zero, the errors do seem to grow persistently so that the final accuracy is just slightly better than the unprocessed case. The problem that arises here is that, as recognized in [12], the matrix $A_{ij} = h^2\zeta_\sigma(\mathbf{x}_j(t) - \mathbf{x}_i(t))$ is severely ill-conditioned. This matter will be discussed amply later on, in the context of radial basis function interpolation theory.

2.2.4 Remeshing Schemes

The now prevalent approach of "remeshing" (also called "redistribution" by some workers) the particle field consists of constructing a Cartesian lattice of new particle locations, and obtaining the new circulation values from the old particles by interpolation. The 2D or 3D interpolation rules are built by Cartesian tensor product of 1D kernels, and these have been constructed of increasing order in terms of the interparticle separation, h. The commonly used interpolation kernels are of two families, the "Λ" and the "M" family, and are given by the following formulas:

$$\Lambda_0(u) = \begin{cases} 1 & \text{if } 0 \le u \le \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.82}$$

$$\Lambda_1(u) = \begin{cases} 1-u & \text{if } 0 \le u \le 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.83}$$

$$\Lambda_2(u) = \begin{cases} 1-u^2 & \text{if } 0 \le u \le \frac{1}{2}, \\ \frac{1}{2}(1-u)(2-u) & \text{if } \frac{1}{2} \le u \le \frac{3}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.84}$$

$$\Lambda_3(u) = \begin{cases} \frac{1}{2}(1-u^2)(2-u) & \text{if } 0 \le u \le 1, \\ \frac{1}{6}(1-u)(2-u)(3-u) & \text{if } 1 \le u \le 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.85}$$

$$M_3(u) = \begin{cases} \frac{1}{2}\left(\frac{3}{2}+u\right)^2 - \frac{3}{2}\left(u+\frac{1}{2}\right)^2 & \text{if } 0 \le u \le \frac{1}{2}, \\ \frac{1}{2}\left(\frac{3}{2}-u\right)^2 & \text{if } \frac{1}{2} \le u \le \frac{3}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.86}$$

$$M_4(u) = \begin{cases} \frac{1}{6}(2-u)^3 - \frac{4}{6}(1-u)^3 & \text{if } 0 \le u \le 1, \\ \frac{1}{6}(2-u)^3 & \text{if } 1 \le u \le 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.87}$$

$$M_4'(u) = \begin{cases} 1 - \frac{5}{2}u^2 + \frac{3}{2}u^3 & \text{if } 0 \le u \le 1, \\ \frac{1}{2}(1-u)(2-u)^2 & \text{if } 1 \le u \le 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.88}$$

Using one of the interpolation kernels above, the remeshing schemes obtain the contribution of circulation $\Delta\Gamma_{j,i}$ from the i-th old vortex with Γ_i to the new mesh point $(\widetilde{x}_j, \widetilde{y}_j)$ according to (in the two-dimensional case)

$$\Delta\Gamma_{j,i} = \Gamma_i \, \Lambda\left(\frac{\widetilde{x}_j - x_i}{h}\right) \Lambda\left(\frac{\widetilde{y}_j - y_i}{h}\right), \tag{2.89}$$

where Λ represents the 1D interpolation kernel.

The first interpolation formula listed above, the Λ_0 kernel, gives a first order scheme, equivalent to nearest grid point (NGP) interpolation. It is the same as the corresponding M_1 kernel (omitted) and can approximate exactly only constant functions. When used with vortex methods, it will ensure conservation of total circulation only (zero-th vorticity moment). The so-called "tent-function", Λ_1, is a second order interpolant and is equivalent to the M_2 kernel (also omitted). It can approximate exactly only linear functions, and when used in vortex methods it will conserve total circulation and linear impulse (first moment of vorticity), using a 2^d stencil (with d the dimension). Following is the third-order Λ_2 kernel, corresponding to quadratic interpolation; it will additionally conserve second moment of vorticity, with a 3^d stencil. This scheme was used in [101, 105]; its main disadvantage is its lack of smoothness, as it is not even continuous. The piecewise-cubic and continuous kernel Λ_3 is known as Everett's fourth order formula, and requires a 4^d stencil, conserving up to third moment of vorticity. (Indeed, the Λ family of kernels is constructed precisely by specifying that increasing moments of vorticity be conserved.)

The interpolation kernels of the "M" family are derived from splines; they are characterized by being more regular than the Λ family. The M_3 kernel is the first of the central B-splines that has continuous first derivatives, and is dubbed "triangular-shaped cloud" (TSC), while the M_4 kernel

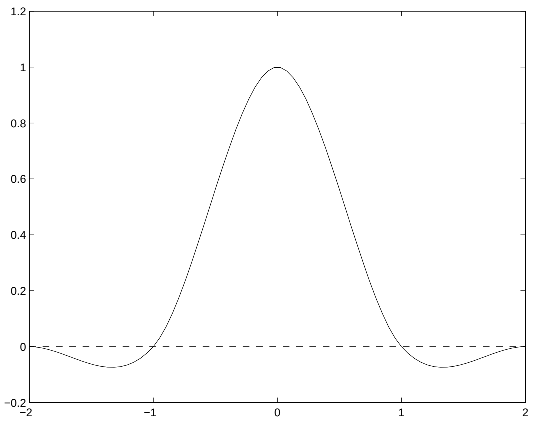
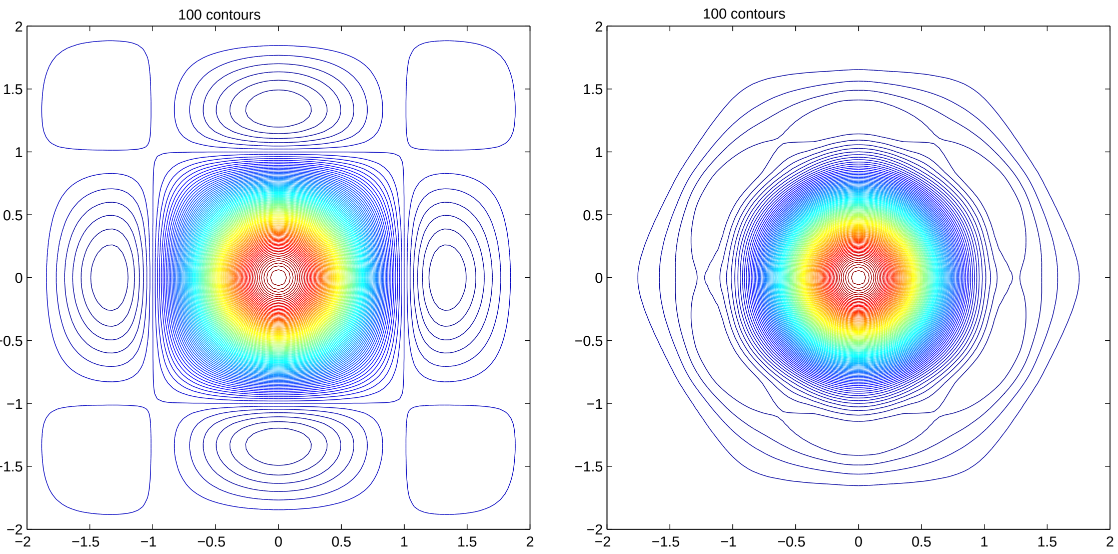


Figure 2.1: Plot of the M_4' interpolation kernel.

is of class C^2; they are both second-order schemes, however, as central B-splines are capable of interpolating exactly only linear functions. The improved M_4' kernel introduced in [135] is of higher accuracy (third order) and when used in vortex methods will conserve the first three invariants (total circulation, linear impulse and angular impulse). This kernel is shown in Figure 2.1, where one can see that a small amount of negative circulation is introduced. M_4' remeshing has been used by researchers who want to ensure highly accurate results (*e.g.* [102]); it has the advantage of being quite smooth (class C^1) and in [52] it was shown to be considerably more accurate than the Λ_2 scheme which is of the same formal order. One can safely say that it is the preferred remeshing scheme used today by many of the leading experts in vortex blob methods, as well as vortex-in-cell methods. It is also still used in smooth particle hydrodynamics (SPH) methods, the subject where it was originally introduced.

Remeshing schemes are still an active area of research in vortex methods. A new scheme has recently been developed (P. Chatelain, private communication) which has the advantage of a narrower stencil than M_4', and a more isotropic distribution of the circulation in comparison to the standard approach of tensor products for higher dimensions. This new scheme operates on a triangular lattice of particles, and is based on the face-centered schemes introduced in [38]. This "hexagonal redistribution" scheme is third order accurate, like M_4', but if one observes the result of applying the latter in 2D tensor product (see Figure 2.2) it is clear that the circulation is anisotropically distributed; the hexagonal redistribution corrects this very effectively. Also, as mentioned, it has a narrower stencil, which has the result that the problem size grows more slowly (due to the need for a stencil on the edge of the domain) than when using M_4'. In the present investigation, some numerical experiments will also be performed using the hexagonal redistribution scheme.



(a) (b)

Figure 2.2: Contour plots of interpolating functions in 2D. (a) M_4' scheme; (b) hexagonal redistribution scheme.

2.2.5 Vortex Blob Splitting

Finally, let us discuss the splitting method of Rossi for core spreading. In [170], a blob splitting algorithm is proposed to control the consistency error of the core spreading method, which maintains the maximum blob radius below a stipulated value, l. The core spreading method is thus corrected by "adaptive spatial refinement". The refinement consists in replacing a single vortex blob of width $\sigma > l$ with a number (chosen as 4) of children blobs of width $\alpha\sigma$. The algorithm is controlled by the numerical parameter α, while other parameters are determined by imposing the conservation of vorticity moments. Since the splitting generates exponential growth of the problem size, a fusion algorithm is proposed to alleviate this problem. A convergence analysis is developed which proves linear convergence (*i.e.*, assuming known velocity) to the Navier-Stokes equations in the L^∞ norm. (As it is pointed out in the paper, the classic convergence theory of Beale and Majda [14] utilizes the L^p norm in the full nonlinear case. It is also noted that the standard convergence theory is applied to the velocity, while Rossi utilizes the vorticity instead.) The error of the refinement is proved to converge to zero as $\alpha \to 1$, and for the particular choice of 1:4 refinement used by Rossi, it is proved to be bounded by $O(1-\alpha^2)$ in the linear case (L^∞ norm vorticity error).

The basic resolution parameter in the splitting algorithm is the fixed maximum blob width l. The accuracy parameter of the splitting is α, determining the width $\alpha\sigma$ of the children blobs. The children blobs' circulation is of course $\frac{1}{4}$ the strength of the parent, for conservation of zeroth moment of vorticity (total circulation). They are placed with their center at a distance r from the parent blob's center, where r is determined from conservation of second moment of vorticity, giving $r = 2\sigma(1-\alpha^2)^{\frac{1}{2}}$. In [170], the values of α are chosen between 0.7 and 0.9 and numerical experiments are presented using a Lamb vortex and a co-rotating vortex pair, where this latter case was compared with an RVM calculation using much larger N. In the case of the Lamb vortex, plotted results of tangential velocity at a given location show quite visible errors, while the plots showing blob locations reveal that there is loss of overlap in some areas. In the case of the co-rotating vortices, the figures also indicate an overlap loss, in particular near the edges of each vortex. The calculation is said to compare well with RVM, but it must be said that at the Reynolds number of the experiment, equal to 100, random walk can be quite crude. It would seem, then, that the chosen splitting scheme is of rather low accuracy.

One of the problems of the splitting technique of Rossi is the lack of any overlap control. In addition, the scheme is numerically diffusive. Furthermore, it is not clear that the criterion for calculating r based on conservation of second moment of vorticity is the best choice. An alternative scheme was studied in [193], where r is chosen to conserve the value of vorticity at the parent blob's center. In one-dimensional experiments, it was found that the center vorticity constraint is more accurate than conservation of second moment of vorticity for one-blob splitting, but it was less accurate in an experiment with eleven blobs superimposed. Since neither constraint proved to be 'ideal', a set of two-dimensional tests using a Lamb vortex were carried out with the aim of finding an empirical relationship $r = f(\alpha)$. A linear fit was performed to (α, r) pairs obtained on the basis of minimizing L^1 norm errors of vorticity, comparing with the analytical solution, which resulted in a relationship somewhere in between the center vorticity and second moment constraints. This empirical law was built into the code used for the core spreading experiments in [194], one of the few recent implementations of the core spreading vortex method.

In the case of using the second moment criterion for r, one can demonstrate how the splitting is diffusive, for in this case the vorticity of a blob placed at the origin is (note that Rossi uses $k = 4$ for the Gaussian blob)

$$\omega_p = \frac{\gamma}{4\pi\sigma^2}\exp\left(-\frac{|\mathbf{x}_p|^2}{4\sigma^2}\right) = \frac{\gamma}{4\pi\sigma^2}\,, \tag{2.90}$$

whereas the vorticity of the four children placed at a distance $r = 2\sigma\sqrt{1-\alpha^2}$ from the parent blob is

$$\omega_c = \frac{\gamma}{16\pi\alpha^2\sigma^2}\sum_{i=1}^{4}\exp\left(-\frac{|\mathbf{x}_p-\mathbf{x}_i|^2}{4\alpha^2\sigma^2}\right) = \frac{\gamma}{4\pi\alpha^2\sigma^2}\exp\left(-\frac{(1-\alpha^2)}{\alpha^2}\right). \tag{2.91}$$

Performing the ratio between the added vorticity of the children blobs and the original vorticity of the parent blob, one sees that the result is always less than 1 (see Figure 2.3), indicating that the maximum vorticity has decreased and hence there has been diffusion.

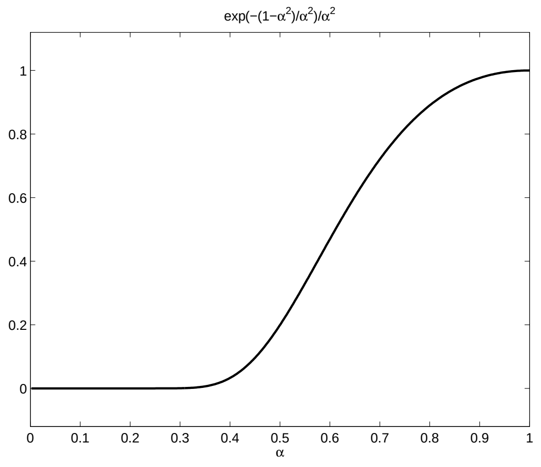


Figure 2.3: Ratio of the maximum vorticity of the four children blobs to the maximum vorticity of the parent blob placed at the origin, for Rossi's 1:4 splitting.

2.3 Introduction and Applications of Radial Basis Functions

Both the static discretization (initialization) and the problem of spatial adaption for vortex methods —*i.e.*, determining the identifying quantities (location and circulation) of a new set of well-overlapped particles to best approximate the current vorticity field— can be approached as a problem of function approximation. In this perspective, one can see a similarity between the vortex blob discretization given by (1.2) and the technique of radial basis function (RBF) interpolation, a tool for solving multivariate scattered data interpolation problems. In the context of Beale's method of circulation processing, the problem of finding the new circulation values so that the particle representation best approximates the vorticity field has been recognized as a problem of "scattered data interpolation" in [12] and [40]. They both refer to Franke's review paper [73], where a study is made of different algorithms for the problem of scattered data interpolation on a set of known surfaces. Franke assessed about 30 algorithms and ranked them based on six criteria —accuracy, visual aspect, sensitivity to parameters, execution time, storage requirements and ease of implementation—. The two methods that ranked best, the so-called multi-quadrics (MQ) and thin-plate splines (TPS) are examples of radial basis functions. Since Franke's review work, however, a significant volume of literature has been published on this subject.

The problem of scattered data interpolation can be formulated as how to best approximate an unknown function $f \in C(\Omega)$ whose values are known on a set of points $X = \{x_1, \ldots, x_N\} \subset \Omega \subset \mathbb{R}^d$, which are scattered on the domain. The RBF approach, following the notation of [188], is to choose the function that approximates f to be of the form

$$s_{f,X}(x) = \sum_{j=1}^{N} \alpha_j \; \Phi_j(x, x_j) + p(x), \tag{2.92}$$

where $p(x)$ is a low-degree polynomial, and $\Phi : \Omega \times \Omega \to \mathbb{R}$ is a fixed function that is translation invariant and in particular satisfies

$$\Phi(x, y) = \phi(\|x - y\|_2) \quad \text{with } \phi : [0, \infty) \to \mathbb{R} \text{ (radiality)} \tag{2.93}$$

Clearly, the blob discretization of the vorticity is analogous to the interpolant (2.92), where the polynomial part is chosen as null and the basis function is the cutoff function, a Gaussian for example. Indeed, Gaussians are used in RBF interpolation (and usually with null polynomial part), as are basis functions of the following types:

$$(i) \quad \phi(r) = r^{\beta},\ \beta > 0,\ \beta \notin 2\mathbb{N} : \text{ pseudo-cubics} \tag{2.94}$$

$$(ii) \quad \phi(r) = r^{2k}\log(r),\ k \in \mathbb{N} : \text{ thin-plate splines} \tag{2.95}$$

$$(iii) \quad \phi(r) = (c^2 + r^2)^{\beta},\ \beta > 0,\ \beta \notin \mathbb{N} : \text{ multi-quadrics} \tag{2.96}$$

The construction of the interpolant (2.92) requires the satisfaction of the interpolation conditions by collocation, leading to a linear system of equations for the coefficients $\alpha = (\alpha_1, \ldots, \alpha_N)$ and the polynomial coefficients. For our purposes, we can assume that the polynomial part is null, and write the system as

$$\mathbf{\Phi}\alpha = \vec{f}, \tag{2.97}$$

where $\vec{f}$ represents the vector of function values at the centres, $\vec{f} = \{f(x_1), \ldots, f(x_N)\}$, and $\mathbf{\Phi}_{ij} = \phi(\|x_i - x_j\|)$. The matrix $\mathbf{\Phi}$ being full and ill-conditioned, Franke [73] concluded that global basis function methods are not feasible for large N. But since then, a great deal of work has contributed to effectively resolve this and several other difficulties. Preconditioning operators were first introduced in [63] for the cases of the MQ and TPS, based on triangulation of the data points and construction of discrete approximations to the iterated Laplacian operators, Δ^k. At the same time, progress was made in regards to theoretical aspects of the problem, with the result of [132] that the interpolation system (2.97) is guaranteed a solution whenever the function $\Phi(x, y)$ is strictly conditionally positive definite and the data distinct. As described in [188] (where proofs are given), the theory has been greatly extended and many basis functions have been characterized, for example, the MQ interpolant is conditionally positive definite and can be made positive definite by appending a linear polynomial, and the Gaussian is positive definite hence not requiring a polynomial part.

The problem of ill-conditioning of the distance matrix $\mathbf{\Phi}$ has been subject of extensive analysis aiming to establish bounds on the norm of the inverse, $\|\mathbf{\Phi}^{-1}\|$, and on the spectral condition number of $\mathbf{\Phi}$ for different basis functions. In [141, 142] upper bounds on the inverse, for a given basis function, are found to depend only on the dimension of the domain space $\mathbb{R}^d$ and the *separation radius* of the data locations, which is defined by

$$q_X := \frac{1}{2} \min_{1\leq j\neq k\leq N} \|x_j - x_k\|. \tag{2.98}$$

This density measure of a point set is the minimal (half) distance that separates one site to its nearest neighbour in the data set. If it is very small, it means that two data locations are very close together, which in turn makes the distance matrix close to singular. The estimates on the condition number, $\kappa(\mathbf{\Phi}) = \|\mathbf{\Phi}\|\ \|\mathbf{\Phi}\|^{-1}$, depend additionally on N, through the dependence of $\|\mathbf{\Phi}\|$ on N. For strictly positive definite basis functions (including the Gaussian), however, estimates independent of N were given in [140]. These estimates reveal that the condition number becomes very large as the minimal overlap ratio, which we could define as q_X/σ for the $k=1$ Gaussian, becomes small. This article also demonstrates how, for the Gaussian, the requirement of good conditioning is at odds with the accuracy of the interpolation, what is called the problem of "good conditioning *vs.* good fit". Additional results in regards to upper bounds on the inverse $\|\mathbf{\Phi}^{-1}\|$ are found in [7], whereas [8] provides lower bounds for special cases of regular arrangements of the data, and [185] provides general lower bounds for scattered data which are not expressed as a function (explicitly) of separation radius. In this last work, it is shown how the lower bounds for $\|\mathbf{\Phi}^{-1}\|$ depend on the smoothness of Φ, becoming larger as the smoothness increases; also, support is provided to the idea that regular placement of the data is most favorable to the conditioning.

Much theoretical progress has been made, in addition, in the estimation of the error of interpolation with RBF's. Error bounds of arbitrarily high order were proved for the multi-quadrics in [122, 123], and for Gaussians in [205], where a simpler theoretical approach is used. Both of these works require a certain restrictive condition on the Fourier transform of the function f being approximated. The error estimates refer to the pointwise difference between the function f and the interpolant $s_{f,X}$, and are found to be of $O(h_X^k)$ where k depends on the RBF $\mathbf{\Phi}$ and the density measure h_X is called the *fill distance* and defined by

$$h_{X,\Omega} := \sup_{x\in\Omega} \min_{1\leq k\leq N} \|x - x_k\|. \tag{2.99}$$

The fill distance measures the maximal distance from any point $x \in \Omega$ (not necessarily a data location) to its nearest point in the data set. In the terminology of computational geometry, it is the radius of the largest empty circle in the data. Hence, it measures how the data "fills" the support Ω, and the quality of the approximation using RBF's (for all different choices of RBF) will deteriorate as $h_{X,\Omega}$ gets larger. For some basis functions, including the multi-quadrics and the Gaussian, improved error estimates based on $h \leq h_{X,\Omega}$ were found in [124]; these local estimates are $O(\lambda^{1/h})$ as $h \to 0$, with $0 < \lambda < 1$. Subsequently, a representation of the norm of the error functional for different RBF's that is workable numerically has been developed [184, 186]. This representation is bounded by the so-called *power function*, $P(x)$, in the following manner:

$$|f(x) - s_f(x)| \leq |f|_{F_\Phi} \cdot P(x), \tag{2.100}$$

where F_Φ is an inner-product space of functions defined via Φ and $|\cdot|_{F_\Phi}$ is the seminorm defined in the function space. For the Gaussian, F_Φ is the space of C^∞ functions. The upper bound of $P(x)$ is written in the form $P^2(x) \leq F(h(x))$ where $h(x)$ is a local measure of fill distance. For example, the Gaussian $\Phi(x) = e^{-\beta\|x\|^2}$ has $F(x) = e^{-\frac{\delta}{h^2}}$, with $\delta > 0$. Note that this most favorable case of exponential convergence for Gaussians comes at the cost of the worst situation in terms of conditioning, due to it being very smooth. This was developed into a general notion of an *uncertainty principle* for RBF interpolation by Schaback in [186], which says that good reproduction quality is only obtained at the cost of poor numerical stability and *vice versa*. By introducing increased smoothness constraints on the function f, approximation orders are doubled in [187], but for the Gaussian with exponential error bounds, this does not bear a major improvement.

Finally, the extensive research in RBF interpolation has also made significant progress in regards to computational efficiency. First, consider that evaluating a function whose approximation has been expressed as an expansion in RBF's can be quite expensive, involving $O(N)$ operations for each evaluation point. Multipole expansions for the fast evaluation of the interpolant (2.92) were introduced for the TPS in [20], where in addition the fast algorithms are also applied to the matrix-vector product required at each step of an iterative solution method, in particular the pre-conditioned conjugate-gradient method. A new method for the fast evaluation of RBF expansions, based on generalizing the multipole method so that changes of basis are easily performed, was presented in [21]. Second, the actual solution of the RBF interpolation problem can be prohibitive for large N, unless fast methods are implemented. This was successfully addressed in [17], where preconditioning strategies were used in conjunction with fast matrix-vector multiplication and a GMRES iterative solution method. The preconditioning method is based on changing the basis in which the RBF's are represented, using approximate cardinal functions. Numerical experiments with TPS and MQ demonstrated significant clustering of the eigenvalues, improving the condition number by several orders of magnitude. This led to the GMRES solution converging in only a few iterations. The overall strategy entails $O(N \log N)$ operations and $O(N)$ storage. Alternatively, an approach for the fast solution of RBF interpolation analogous to forward substitution is developed in [159, 69], based on generalization of the iterative method constructed in [18]; this was applied to TPS while the extension to the MQ and other conditionally positive definite functions was shown to be accessible. Considerable improvements to this method have been performed, in particular the inclusion of a Krylov subspace algorithm which is guaranteed to converge [68]. A third approach for the efficient solution of the RBF interpolation system is based on domain decomposition [19]; such a method was applied to data sets of up to 5 million two-dimensional points.

The RBF technique for scattered data interpolation originates on the work of Hardy [85, 86], who derived the multi-quadric scheme in two dimensions to approximate geographical surfaces and gravitational and magnetic anomalies. The thin-plate spline, on the other hand, is generally attributed to Duchon [60, 61, 62], but it was previously discovered in [84]. The TPS is a two-dimensional analogue of the cubic spline in one dimension, constructed so that it passes through the interpolation points minimizing the "bending energy".

The use of RBF's has been proposed for the numerical solution of partial differential equations by Kansa [92, 93], who used the multi-quadrics as the spatial discretization scheme for parabolic, hyperbolic and the elliptic Poisson equation. Kansa's approach is the following. Consider the linear advection-diffusion problem: $f_t + uf_x - Df_{xx} = 0$ (+ B.C.'s). Now, expand f in terms of MQ basis functions and add a low-order polynomial

$$\begin{aligned} f(x) &= a_1 + a_2 x + \sum_{j=3}^{N} a_j \, \tilde{g}(x - x_j) && (2.101) \\ \tilde{g}(x - x_j) &= g(x - x_j) - \frac{(x_2 - x_j) g(x - x_1) + (x_j - x_1) g(x - x_2)}{(x_2 - x_1)} && (2.102) \\ g(x - x_j) &= \sqrt{(x - x_j)^2 + r^2} && (2.103) \end{aligned}$$

where r^2 is a non-zero parameter, the "shape parameter" of the MQ basis. The set of discretized values of the unknown function f, $\{f_i\}$, $i = 1, \ldots, N$, are transformed from the set of expansion coefficients $\{a_i\}$ according to the linear system of equations

$$f_i = \sum_{j=1}^{N} A_{ij} \, a_j, \tag{2.104}$$

where the matrix A_{ij} has the following components in the row i:

$$\begin{aligned} A_{i,1} &= 1, \\ A_{i,2} &= x_i, \\ A_{i,j} &= \tilde{g}(x_i - x_j), \quad \text{for } 3 \leq j \leq N. \end{aligned}$$

Now one can construct the solution of the advection-diffusion equation by obtaining the partial derivatives of f using the MQ expansion as follows:

$$\begin{aligned} \left(\frac{\partial f}{\partial x}\right)_i &= a_2 + \sum_{j=3}^{N} \frac{\partial \tilde{g}_{ij}}{\partial x} \, a_j, && (2.105) \\ \left(\frac{\partial^2 f}{\partial x^2}\right)_i &= \sum_{j=3}^{N} \frac{\partial^2 \tilde{g}_{ij}}{\partial x^2} \, a_j && (2.106) \end{aligned}$$

where the partial derivatives of $\tilde{g}_{ij}$ and g_{ij} are obtained explicitly. Subsequently, the advection-diffusion equations are solved in the Eulerian frame with nodes whose location in space is fixed. The spatial basis functions, then, are fixed in time, whereas the expansion coefficients vary in time, $a_j = a_j(t)$, which is analogous to the finite element approach.

Kansa compared the method sketched above with finite differences, using a weighted average of upwind and central difference; in this case, upwinding is necessary to stabilize the advection term. He points out that, in contrast, the MQ approach does not require upwinding. The results presented by Kansa demonstrate that coarser data can be used with the MQ spatial discretization to obtain the same accuracy as with finite differences, showing that RBF's can be very efficient for the solution of PDE's. He does find that the MQ method behaves badly as the diffusion coefficient tends toward zero (when a front is formed in the solution tending toward a step-function). In this situation, the flat regions fore and aft of the shock are very noisy. The problem is that these flat regions need very flat basis functions (large r^2), which give rise to very ill-conditioned coefficient matrices.

Kansa's second test problem was a 1D von Neumann blast wave, and his third was a 2D elliptic Poisson equation with either Dirichlet or Neumann boundary conditions. In general, his conclusions are that the MQ scheme is very high order, hence excellent results can be obtained with a coarser distribution of data points than with a finite difference approximation. Being a global scheme, however, he concludes that it is impractical for large fluid dynamics problems.

Maybe one should pause for a moment, and consider the similarity between the approach of Kansa for the advection-diffusion equation, and the method of obtaining derivatives used by some workers of vortex methods. It was already mentioned that Fishelov [70] obtains the Laplacian of the vorticity by directly differentiating the cutoff function, thus obtaining an expression for the diffusion term in the vorticity transport equation. This same approach was used by Greengard and Anderson [4], who explicitly differentiate the cutoff function to obtain the stretching of vorticity term. This can be seen to be the same basic idea as that used in Equation (2.105). Perhaps one could say that in the vortex blob method the RBF approach to solving PDE's was being applied long before the work of Kansa. There is one fundamental difference, however, and that is that in the vortex method the nodes are allowed to vary in time, whereas Kansa's method is Eulerian.

Chapter 3

Numerical Experiments on the Accuracy of Vortex Methods

3.1 Error Definitions and Measurements

When one wishes to study the accuracy of a vortex method, there are three fields one can look at and measure the errors on: the vorticity field —whose governing equation is actually being solved by the method—, the velocity field, and the flow map. The classical accuracy analyses provide estimates or bounds for the *velocity* errors. But there are different sources of error which are relevant in a vortex method computation. For example, Perlman [152] distinguishes between these sources of error in the following way. The *consistency error* is $\|\mathbf{u}-\mathbf{u}^h\|$, the distance between the exact velocity $\mathbf{u}$ and the discrete velocity $\mathbf{u}^h$, the latter obtained by a Biot-Savart calculation over the collection of vortex blobs used to discretize the vorticity field. This error has two components, the *smoothing error*, and the *discretization error*, so that using the triangle inequality one can write

$$\|\mathbf{u}-\mathbf{u}^h\| \leqslant \|\mathbf{u}-\mathbf{u}^\sigma\| + \|\mathbf{u}^\sigma-\mathbf{u}^h\|. \tag{3.1}$$

The first term on the right hand side of the equation above is produced by the regularization of the singular Biot-Savart kernel, and the introduction of the cutoff function to replace the Delta function in the discretization. It depends on the cutoff parameter σ and it will also depend on time. The second term is due to the numerical integration of the regularized Biot-Savart velocity by the trapezoid rule, and therefore depends on the mesh width h, in addition to σ and t. There is as well the error introduced by the fact that the velocity is calculated using an approximate particle path, and not the exact one, and this is called the *stability error* and represented by $\|\mathbf{u}^h-\tilde{\mathbf{u}}^h\|$.

The convergence theory of vortex methods establishes convergence in two dimensions for the particle paths, for the discrete velocity and for the continuous velocity [81, 14, 15]. In three dimensions there is additionally proof of convergence of the particle vorticities and of the vorticity field

[11]. Finally, the time discretized method is proved convergent for the particle paths using specific time-stepping schemes in [4, 82].

In the present experiments, the error of the discretization is measured in terms of the maximum relative error in vorticity and velocity —*i.e.* the maximum field value of the point-by-point difference between the exact and discretized vorticity/velocity, divided by the maximum value of vorticity/velocity—, and a discrete L^2-norm error, calculated using the following formula:

$$\|\omega - \tilde{\omega}^h\| = \left(\sum_i |\omega(\mathbf{x}_i) - \omega^h(\mathbf{x}_i)|^2 h^2 \right)^{\frac{1}{2}} \tag{3.2}$$

where, most of the time, the value of h in the formula above will be in practice replaced by $h/2$, to measure errors in a sub-grid scale as well.

3.2 Errors of the Blob Discretization

The first fundamental question one may pose in regards to the practical use of the vortex (blob) method refers to the accuracy that one can expect from the spatial discretization of the vorticity field, by means of equation (1.2). Note that we do not deal with the point vortex method which presents certain difficulties due to its singular nature. The vortex blob method deals with this problem by using elements of vorticity which have a distribution that is a smooth approximation to the delta function (the "cutoff function"). Both the vortex blob method and the point vortex method have an extensive convergence analysis [83, 81, 14, 15, 82, 47, 88], as already mentioned, but here we examine the accuracy of the blob discretization in practice, based on numerical experiments.

Since the study of Perlman [152], one of the early efforts to analyze and quantify the errors in vortex methods, it has become well-known that the accuracy of discretization with the vortex method fundamentally depends on three factors: the choice of cutoff function used in the discretization, the value of the cutoff parameter, σ, and the way an initially existing vorticity field is discretized and the numerical method initialized. Two choices of cutoff function were used in the present calculations: the most popular choice with most workers is the Gaussian (1.3), formally a second order cutoff, *i.e.* $O(\sigma^2)$; in addition, we used the algebraic eighth-order cutoff of Nordmark [144], given by

$$\zeta^{(8)}(r) = \begin{cases} \frac{52}{\pi}(1 - 21r^2 + 105r^4 - 140r^6)(1 - r^2)^9 & r \leqslant 1 \\ 0 & r > 1 \end{cases} \tag{3.3}$$

where $r = |\mathbf{x}|$, and $\zeta_\sigma(\mathbf{x}) = 1/\sigma^d \, \zeta(|\mathbf{x}|/\sigma)$, with d the dimension of the problem.

As mentioned in the Introduction, two classic test problems will be used: the Lamb-Oseen vortex given in (1.13), and the inviscid vortex patch, or circular shear layer, given in (1.12). In the

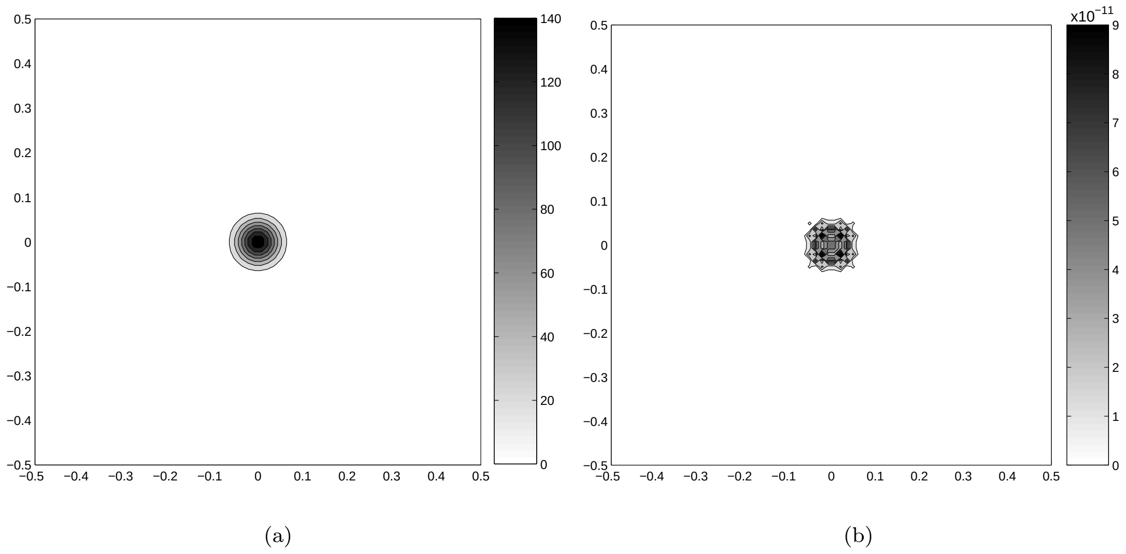


Figure 3.1: Lamb vortex, $\Gamma_o = 1$, $\nu = 0.01$, $t_o = 0.05$. (a) Vorticity at $t = t_o$; (b) point-wise error in vorticity when discretized with Gaussian blobs with $\sigma = 0.02$ and $h/\sigma = 0.7$, using the time shift correction. The maximum relative error (*i.e.*, normalized by $\omega_{\max}$) is 9.2514e-11; $N = 5184$.

numerical experiments one chooses the following parameters: for the Lamb vortex, the initial time t_o and maximum circulation Γ_o (usually taken as 1); for both test problems, the cutoff parameter σ and the inter-particle spacing in the initially regular lattice, h.

When using the Gaussian cutoff, the relevant spatial discretization parameters that are varied in the experiments are the blob core size, σ, and the overlap ratio, defined by h/σ. On the other hand, it is standard for higher-order cutoff functions to require a relationship between core size and inter-particle separation to be of the type $\sigma \sim h^q$, with $q < 1$. This is to conform to the convergence theory developed, in particular, in [81, 82]. For this reason, the discretization parameters to be chosen in the case of the Nordmark blobs are the inter-particle spacing h and the proportionality constant c used in the relationship $\sigma = c\sqrt{h}$.

The first set of tests was performed by discretizing the vorticity field of a Lamb vortex, with a fixed value of h, but varying overlap ratio. The vortex blobs are placed on a square lattice and their circulation strengths are obtained using the 'time shift correction' given in Equation (2.74). To illustrate the efficacy of the correction for initialization, suppose the Lamb vortex with $t_o = 0.05$ and $\nu = 0.01$ is discretized with blobs of size $\sigma = 0.02$ and overlap ratio $h/\sigma = 0.7$. Note that this is a very "spiky" initial vorticity, with $\omega_{\max} = 151.45$, which is plotted as contours in Figure 3.1(a). Vortex blobs are placed on a square lattice in the (x, y)-domain $[-0.5, 0.5]^2$. The vorticity error that is obtained, measured in the maximum norm and normalized by the maximum vorticity, is plotted in the contours of Figure 3.1(b), and is very small (of order 10^{-11}). If one did not use the time shift correction, but instead used Equation (2.72) directly, the maximum vorticity error is

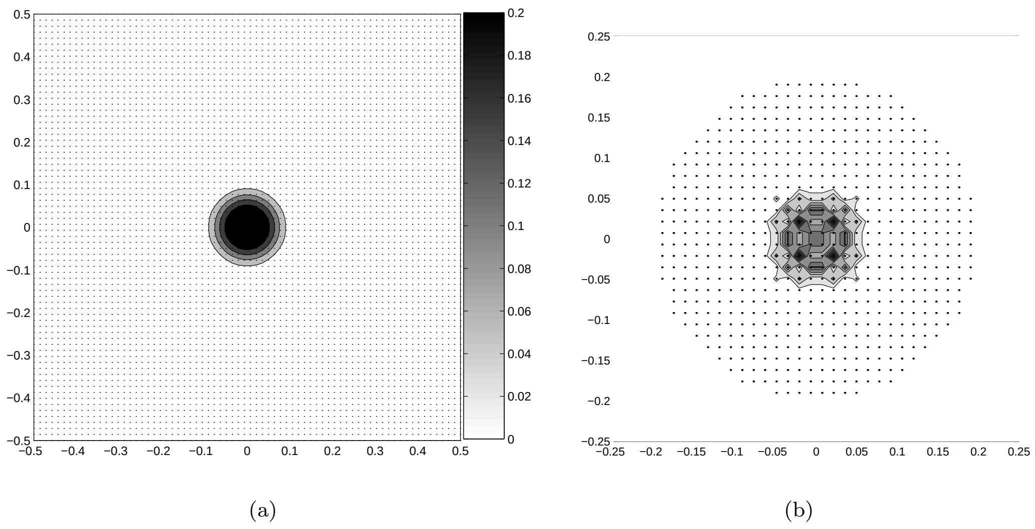
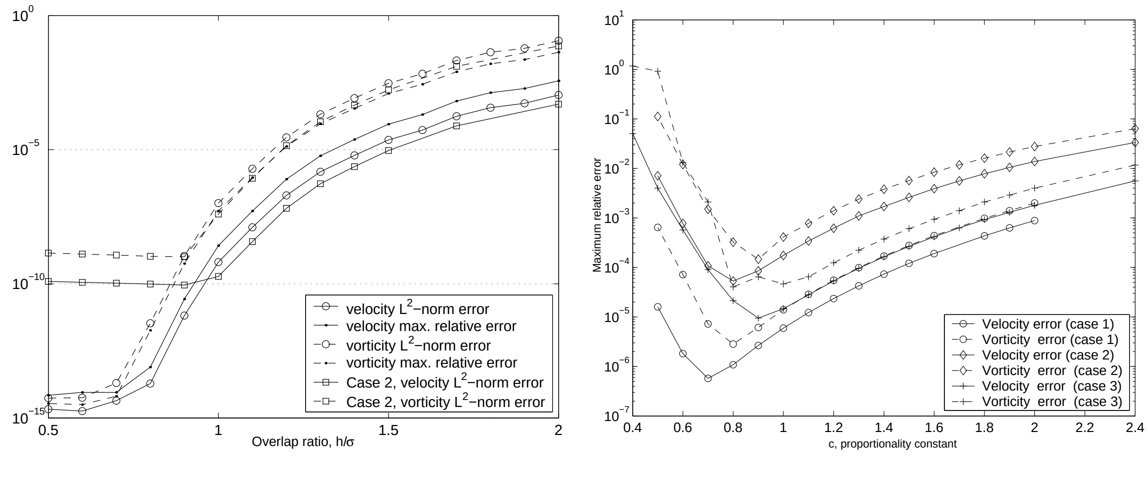


Figure 3.2: Lamb vortex, as in Figure 3.1(a). (a) Error of discretizing the vorticity without the time shift correction; (b) error using the time shift correction, plus population control, $N = 624$.

almost 24% of the maximum vorticity, which clearly illustrates that the standard initialization can lead to very large errors (in this case, they are dramatically large due to the high gradients of the initial vorticity; for a very diffused Lamb vortex, say $t_o = 1.0$, the error obtained is about 2%, which is still considerable). Figure 3.2(a) shows the error field when the vorticity is discretized without the time shift correction, plus the vortex blob locations are indicated with a dot; the number of particles is 5184. The error when using the time-shift is shown again in Figure 3.2(b), in a closer view (the color-bar of Fig. 3.1(b) applies), and also a technique is illustrated which will be used often: population control. In this case, upon initializing, all particles with a circulation strength smaller than the machine round-off error (about 2.2e-16) were eliminated. As one can see, this had no effect on the error (which is obviously calculated *after* population control), but the problem size was reduced drastically to $N = 624$. This example is quite extreme, due to the high gradients of the initial vorticity, but it serves well the purpose of illustration.

To perform the tests of discretization with varying overlap ratio, more diffused states of the Lamb vortex are chosen, one case with $t_o = 0.25$ and $t_o = 0.5$ in the second case; ν is still 0.01 and $\Gamma_o = 1$. These initializations are once again performed placing the particles on a square lattice, now in the domain $[-0.6, 0.6]^2$, and using the time shift correction. Figure 3.3(a) shows the plotted results of maximum relative error in vorticity, discrete L^2-norm error in vorticity and the same two measures of error for the velocity, in the first case of $t_o = 0.25$. The overlap ratio is varied from a minimum of 0.2 to a maximum of 2.0 (only 0.5 to 2.0 shown on the plot), and the number of vortex blobs varies from $31^2 = 961$ with an overlap of 2, to $301^2 = 90601$ with an overlap of 0.2. In Case 2, $t_o = 0.5$, a larger domain was used to start with but population control was enforced by deleting



(a) (b)

Figure 3.3: Lamb vortex, $\Gamma_o = 1$, $\nu = 0.01$. (a) Vorticity and velocity errors *vs.* overlap ratio, using Gaussian blobs with $\sigma = 0.02$; (b) Vorticity and velocity errors *vs.* c, with $\sigma = c\sqrt{h}$ for Nordmark blobs. Case 1: $t_o = 0.25$, $h = 0.01$, $N = 5177$; Case 2: $t_o = 0.25$, $h = 0.025$, $N = 925$; Case 3: $t_o = 0.5$, $h = 0.025$, $N = 925$.

blobs with strength less than `eps`=2.2e-16 (as before); only the L^2-norm measures of error are shown (for clarity of the plot). Due to population control, the errors are bounded below by about $\sim 10^{-10}$, but the problem size is drastically reduced. These larger errors now occur on the edge of the particle field (where blobs were deleted), not on the region of maximum vorticity.

The results shown in Figure 3.3(a) demonstrate how, with the appropriate choice of the overlap ratio, it is possible in practice to obtain negligibly small errors with the vortex blob representation. In this case of the Lamb vortex discretized with Gaussian blobs ($k = 2$), an optimum overlap ratio in the range $(0.7, 1.0)$ will produce velocity errors smaller than about $O(10^{-8})$. It is clear as well how strongly the initialization depends on overlap ratio, there being a loss of several orders of magnitude in the accuracy as h/σ increases passing through the value of 1. Another observation that one can make from the figure is the fact that the vorticity errors are always larger than the velocity errors, which is consistent with the general results of the convergence theory of vortex methods. Note, as well, that the errors are slightly smaller for Case 2, with $t_o = 0.5$, before the effects of population control are felt at the smaller values of overlap ratio; this is accounted for by the fact that this more diffused state of the Lamb vortex is more accurately discretized with the same value of σ, as it has a larger "characteristic length" than the less diffused state.

Using the Nordmark cutoff function, the Lamb vortex is now discretized with different values of the proportionality constant c used in the relation $\sigma = c\sqrt{h}$. The particles are placed on a square lattice, and the standard initialization formula (2.74) is used. The results shown in Figure 3.3(b) demonstrate how it is possible to obtain tolerably small errors when using standard initialization

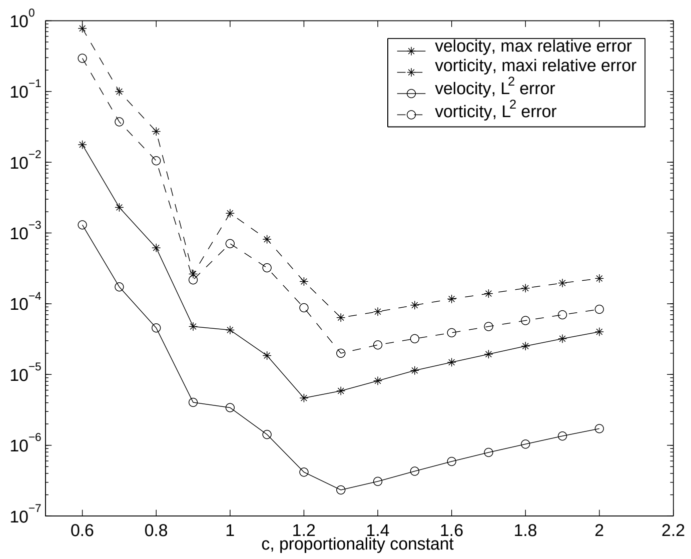


Figure 3.4: Errors of discretization of a vortex patch of vorticity given by Equation (1.12) with $k = 3$, using Nordmark blobs with varying proportionality constant c, $\sigma = c\sqrt{h}$.

with high-order cutoff functions (which is one of the reasons why they were introduced in [16]). This is of course quite useful, due to the fact that the time shift correction used with the Gaussian cutoff is of purely academic use, since it can only be applied in the very particular case of the Lamb-Oseen vortex which is an exact solution of the diffusion equation. For this reason, one could argue that the accuracy demonstrated in Figure 3.3(a) is the best possible accuracy that one can expect of the discretization using Gaussian vortex blobs. It will be later discussed at some length how one can obtain comparable accuracy as when using the time shift correction, but with any initial vorticity field. This will be an application of radial basis function interpolation and will be discussed in §5.3.

The results of discretizing the Lamb vortex using (2.72) with Nordmark blobs and varying proportionality constant c (Figure 3.3(b)) indicate that minimal errors can be obtained with $c \in (0.7, 0.9)$. Nordmark reported in [144] finding an optimal value of $c = 1.7$ for his cutoff, using the test problem of the inviscid axisymmetric vortex patch (1.12). This optimum was chosen by him to minimize the velocity error at time-zero with $h = 0.1$. The optimum proportionality constant, one can conclude, is dependent on the flow field being approximated. Next, using the test problem of the vortex patch, the errors of discretization with Nordmark blobs were obtained for $h = 0.05$; results are presented in Figure 3.4. It can be seen that this time the smallest errors are obtained for $c \in (1.3, 1.6)$, which is closer to what was reported by Nordmark.

The discretizations used for the plot in Figure 3.3(a) were carried out with particles placed on a square lattice. To observe the effect of a different particle arrangement, discretizations were carried out on the Lamb vortex with initial conditions as in case 1 of Figure 3.3(a), but using both a square lattice and a triangular lattice of equilateral triangles. Until now, the errors had been measured

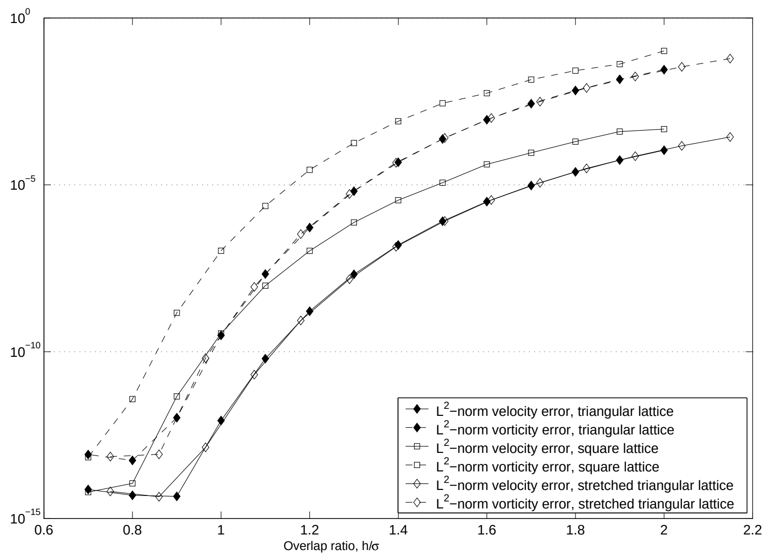


Figure 3.5: Lamb vortex $\Gamma_o = 1$, $t_o = 0.25$ and $\nu = 0.01$. Vorticity and velocity errors *vs.* overlap ratio for Gaussian blobs with $\sigma = 0.02$, on square *vs.* triangular lattice.

on the particle locations, that is, on a mesh of size h. This time, the discrete L^2-norm errors were measured on a finer mesh of spacing $h/2$ for h the inter-particle spacing. Discretizations on the triangular lattice were carried out twice: once with the same value of h as the ones on the square lattice, which produces in the denser triangular lattice a slightly increased resolution, and once with a "stretched" value of h to obtain an equivalent resolution to the square lattice (that is, for equal cell area in both lattices; see Figure 3.6). This last option results in an effective larger overlap ratio for the same nominal value of h, for comparison with the same resolution case on a square lattice. For the same value of h, the triangular lattice "fills space" 15% more in 2D, *i.e.*, there are 15% more particles on average in the same area. With the stretched value of h, of course, the number of particles for each value of overlap ratio is almost the same for both lattices. The domain of initialization is obtained by first filling with blobs a square area larger than the support of vorticity, then calculating the particle strengths using the time-shift correction and finally eliminating all particles whose calculated strength is less than the machine roundoff, 2e-16. The errors are calculated after this form of population control is enforced, which has the effect of limiting the accuracy that is obtained at the smallest values of overlap ratio. For clarity, the plot in Figure 3.5 shows only the errors calculated with a discrete L^2-norm on an $\frac{h}{2}$-mesh (*i.e.*, the errors on the maximum norm are omitted); also note the lowest overlap ratio used in this Figure is 0.7 rather than 0.5 as was in Figure 3.3(a).

Figure 3.5 demonstrates how the triangular lattice provides an increased accuracy in comparison with a square lattice. For example, with an overlap ratio of 1.0 at the same value of h, the triangular lattice proves to be almost three orders of magnitude more accurate, although with a larger N (by

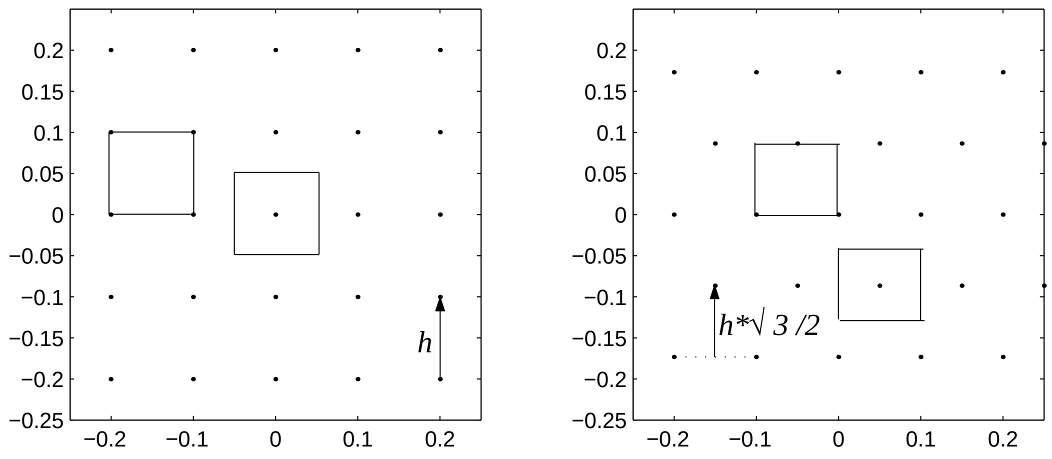


Figure 3.6: The area associated to a particle in the square lattice, for inter-particle spacing h, is h^2, whereas for the triangular lattice it is $\sqrt{3}h^2/2$. For this reason, to get equivalent resolutions on both lattices, the value of h is 'stretched' for the triangular lattice.

about 15%, as already mentioned). If one compares the equivalent resolution cases for a nominal overlap ratio of 1.0, then the triangular lattice gives about one order of magnitude improvement (with an effective overlap of 1.075). We will come back to this when discussing radial basis function interpolation, in §5.3.

The value of the overlap ratio at which one obtains a desired accuracy is associated to the choice of cutoff function; in particular, the numerical value for a given accuracy at the same value of σ will be different for Gaussians of different spread. Consider the plots in Figure 3.7, where the errors are once again obtained on a mesh of width $h/2$. The initializations were repeated for the three usual choices of Gaussian cutoff, with varying overlap ratio, and one can see that the errors obtained with the $k = 2$ Gaussian at $h/\sigma = 1$ are reproduced at a smaller value of h/σ with the $k = 1$ Gaussian, and at a larger value of h/σ with the $k = 4$ Gaussian. In other words, the scale that the parameter σ measures in each case is different. For this reason one needs to be careful when comparing the work of different authors, in particular when a given 'optimum' value of overlap is given. Considering the resulting requirement that the overlap ratio be less than 1 as a very convenient one, we opt for the $k = 2$ Gaussian.

Finally, the relationship between the overlap ratio and the conditioning of the distance matrix formed using the cutoff function has been examined. For both the square and the triangular lattices, the condition number is plotted against the overlap ratio in Figure 3.8, where it can be seen how the matrix becomes increasingly ill-conditioned as the overlap decreases below the value of 1 (once again, this is for the $k = 2$ Gaussian). At an overlap ratio of 0.7, the condition number for the square lattice is of order $\sim 10^8$, but note that for the equivalent resolution triangular lattice (stretched h), the condition number is smaller by two orders of magnitude.

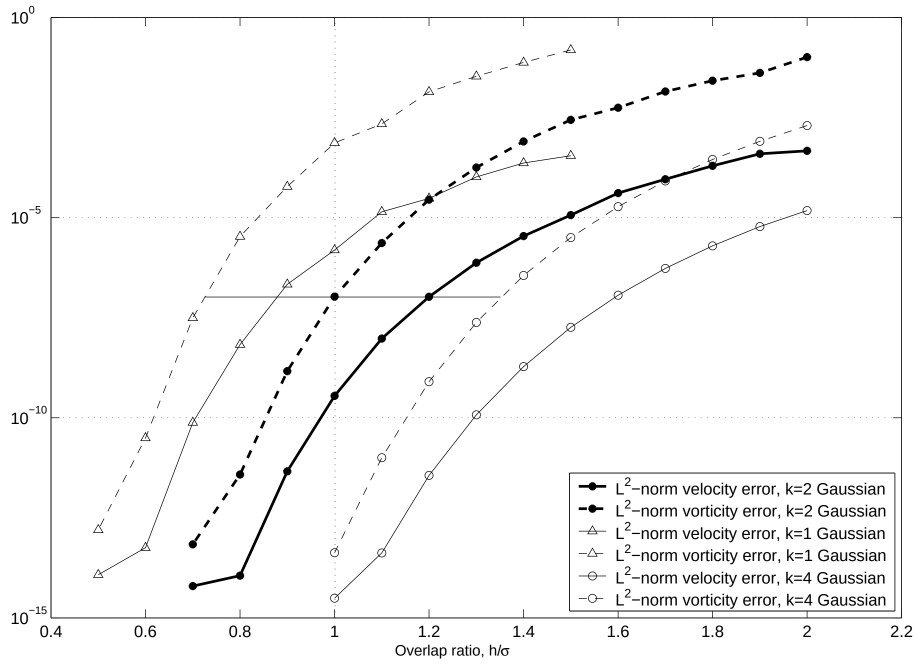


Figure 3.7: Errors of discretization with Gaussian blobs *vs.* overlap ratio, for different widths of the cutoff.

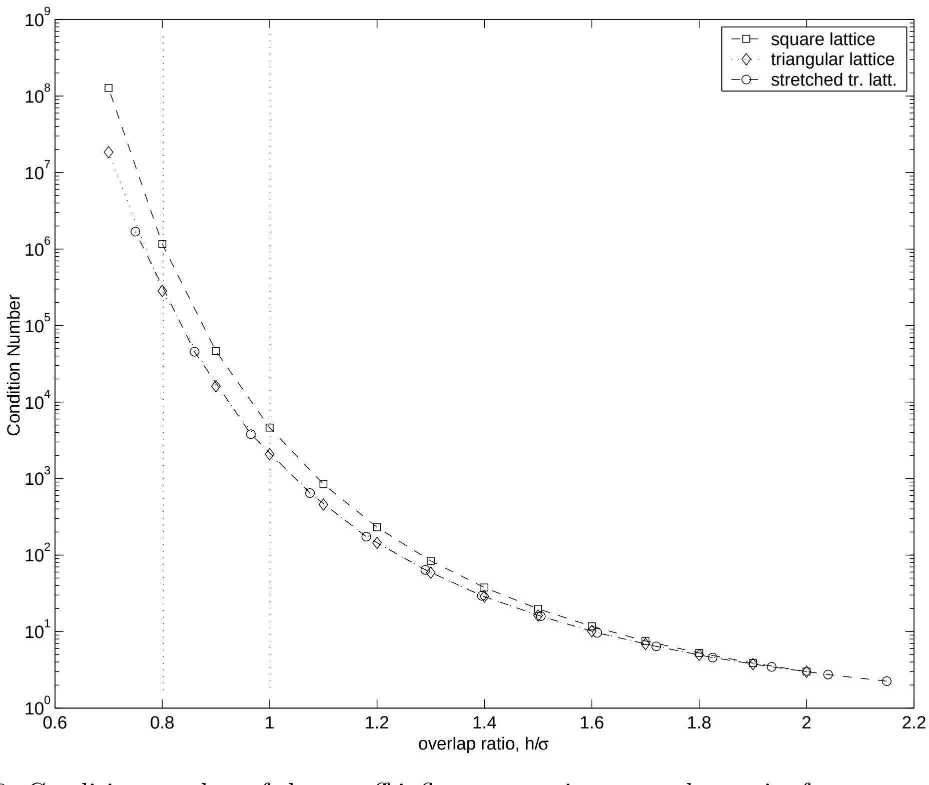


Figure 3.8: Condition number of the cutoff-influence matrix *vs.* overlap ratio, for square and triangular lattices.

3.3 Effects of the Lagrangian Time Evolution

It was seen in the previous section how the accuracy of discretization with vortex blobs depends crucially on the overlap ratio. This demonstrates the practical situation when the vortex blobs cease to overlap fully (overlap ratio larger than 1, for $k = 2$ Gaussian blobs). The theoretical situation, on the other hand, is that convergence proofs of the vortex method require overlap of the blobs, in the sense that it is an assumption in the proof [83, 81, 14, 4, 82]. One expects, therefore, that as the vortex particles are allowed to evolve in a Lagrangian manner, the strain of the flow field will cause the overlap to be lost in some areas thereby increasing the errors. The problem becomes of crucial importance as the viscosity value is reduced, and it is the determining factor in the results that can be obtained from an inviscid simulation. Early on, numerical experiments [152] showed that for $\sigma \approx h$ the initial accuracy is lost in a relatively short time. Hence, it was proposed that the optimal choice of σ be made based on the final time desired for the computation, in other words, it was suggested that providing an initially denser particle field was necessary for longer time-marching calculations. This of course refers to calculations with no added spatial adaption algorithms.

In the present work, a large number of numerical calculations have been performed, using both test problems described previously and both the Gaussian and Nordmark blobs. Different time stepping methods were used as well, including Adams-Bashford of second and third order with different start-up schemes, and Runge-Kutta of fourth order. The experiments show clearly that good initialization and time-stepping do not guarantee that the accuracy can be maintained at later times of a computation. In what follows, some exemplary cases will be presented.

A typical calculation of a Lamb vortex with small viscosity, giving a moderately high Reynolds number of 2000, is presented in Figure 3.9. The initial vorticity is discretized using Gaussian blobs on a square lattice with the time-shift correction, resulting in an initial velocity error of $O(10^{-9})$ with $h/\sigma = 0.8$. As shown in Figure 3.9(a), the errors quickly grow, and oscillate around a deteriorated accuracy level about five orders of magnitude larger than initially. Note that the errors do not grow without bound, which has been pointed out before [152, 16, 12]. Figure 3.9(b) shows the contours of vorticity error as well as particle locations at the final time of the calculation. The error contours show how the Lagrangian distortion of the particles introduces spurious (although weak) vortical structures. The vorticity error is measured in a maximum norm, and the contour of largest absolute value is 1.23% of the maximum vorticity, while the outermost contour has a value of 0.2% the maximum vorticity. In this calculation, vorticity diffusion was provided by core spreading and the final value of σ is 0.0490 (note that in a problem with radial symmetry, Greengard's inconsistency observations [79] do not apply).

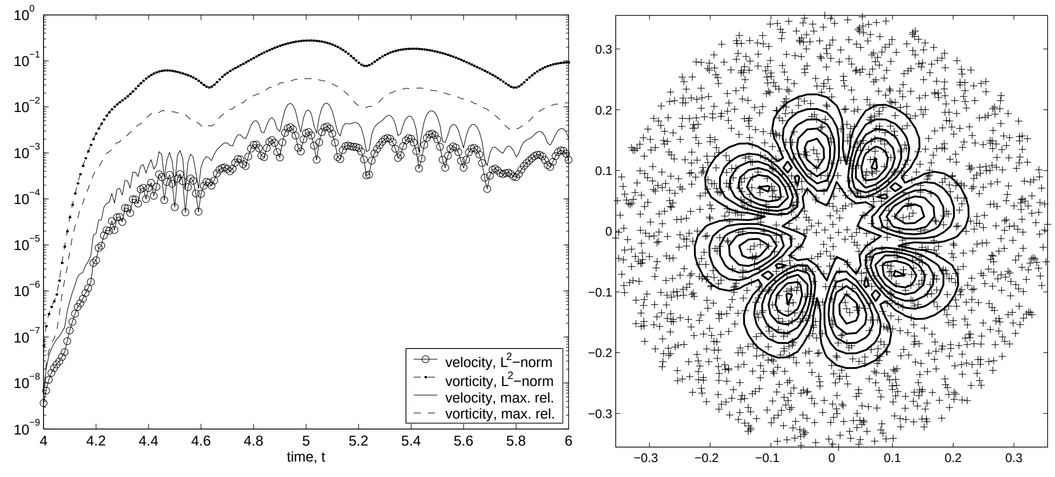
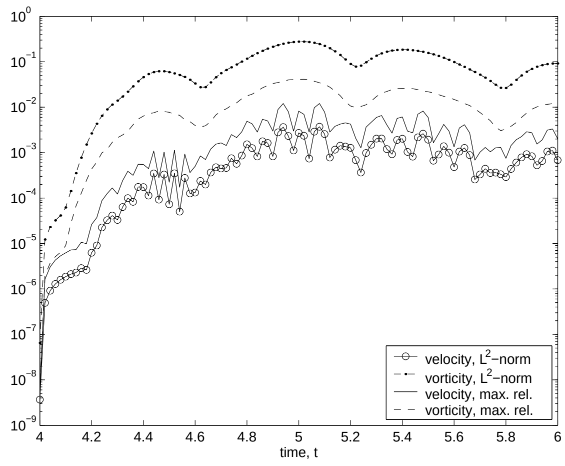


Figure 3.9: Calculation of a Lamb vortex: $\Gamma_o = 1$, $t_o = 4.0$ and $\nu = 0.0005$; Gaussian blobs with $\sigma = 0.02$, $h = 0.0160$, $N = 1560$. RK4 time-stepping, $\Delta t = 0.01$. (a) Evolution of errors in vorticity and velocity; (b) vorticity error contours and particle locations at final time.

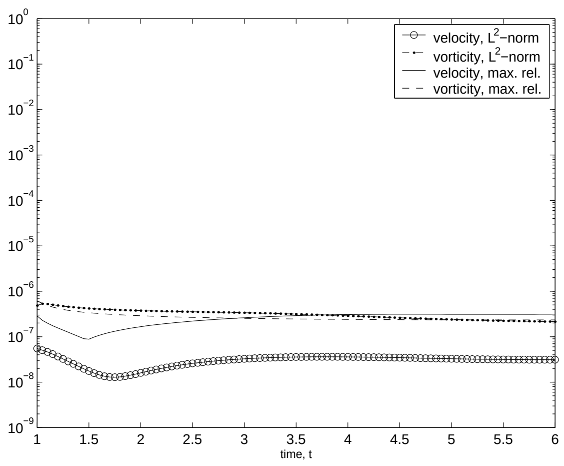
t	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
4.0	4.85E-09	6.02E-09	6.48E-08	3.67E-09
6.0	1.23E-02	2.10E-03	9.23E-02	7.09E-04

Summary of errors for the experiment shown in Figure 3.9.

For the experiment just described, the errors in vorticity and velocity at initialization ($t = t_o = 4.0$) and at the final time of the calculation are shown in the table above. About six orders of magnitude are lost in the accuracy, quite quickly in fact. With every other parameter kept the same, if this experiment is carried out with a larger time step of $\Delta t = 0.02$, one observes almost the same final errors, whereas a difference is obvious at the beginning of the calculation, the first 10 time steps or so, indicating time-stepping errors. See Figure 3.10(a). A number of numerical experiments with relatively small values of viscosity, giving $Re \sim 10^3$, produced similar results. In contrast, one can produce a calculation of the evolution of the Lamb vortex where the initial accuracy is maintained (without applying any sort of spatial adaption) throughout the calculation. For example, an experiment with a rather viscous Lamb vortex using $\nu = 0.01$ produced a final velocity error of order 10^{-8}; the evolution of errors in this case are plotted in Figure 3.10(b). However, the fact is that in this more viscous case the vortex cores are growing fast enough (due to core spreading) that one can expect that overlap is maintained even though the particles become disordered. In the case shown, for example, the cores grow from an initial value of $\sigma = 0.02$ to a final value of 0.3169, which is an increase of fifteenfold. This really is of no use in practice, due to



(a)



(b)

Figure 3.10: Evolution of errors in vorticity and velocity; calculations of a Lamb vortex: $\Gamma_o = 1$. Gaussian blobs with $\sigma = 0.02$, $h = 0.0160$, both cases. (a) $t_o = 4.0$ and $\nu = 0.0005$; $N = 1560$. RK4 time-stepping, $\Delta t = 0.02$. (b) $t_o = 1.0$ and $\nu = 0.01$; $N = 7704$. RK4 time-stepping, $\Delta t = 0.05$.

the necessity of core size control for convergence of the core spreading method, but it is nevertheless interesting to note. It will also be helpful for studying time-stepping effects (next section).

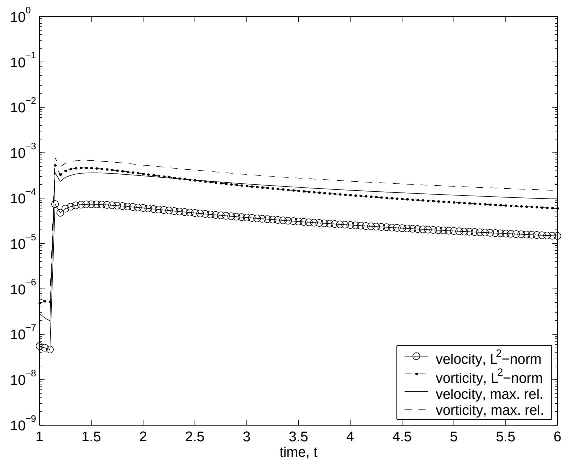
t	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
1.0	6.04E-07	2.87E-07	4.83E-07	5.51E-08
6.0	2.34E-07	3.14E-07	2.14E-07	3.13E-08

Summary of errors for the experiment shown in Figure 3.10(b).

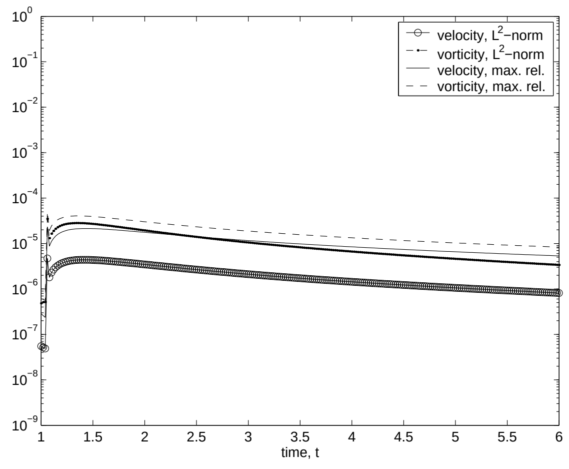
3.4 Aspects of Time-Stepping

A concise analysis of the effect of time-stepping will be presented here. Consider first the result of repeating the experiment presented in Figure 3.10(b), but using an Adams-Bashford scheme of third order (AB3) instead of Runge-Kutta of fourth order (RK4). The use of Adams-Bashford schemes is advocated by some workers due to the fact that only one velocity evaluation is required per time step. In contrast, RK4 requires four velocity evaluations, and this is the most expensive part of most time-marching computations, a fact that is especially true for vortex methods when no fast summation method is used for the Biot-Savart law.

The errors obtained with the AB3 scheme are shown on the table below, while the evolution of errors is plotted in Figure 3.11(a). One can see that three orders of magnitude were lost in comparison with RK4, most of which once the AB3 starts at the third time step (two initial steps



(a)



(b)

Figure 3.11: Evolution of errors in vorticity and velocity; calculations of a Lamb vortex: $\Gamma_o = 1$, $t_o = 1.0$ and $\nu = 0.01$; $N = 7704$. Gaussian blobs with $\sigma = 0.02$, $h = 0.0160$, both cases. (a) AB3 time-stepping, $\Delta t = 0.05$, with two RK4 steps to start. (b) AB3 time-stepping, $\Delta t = 0.02$.

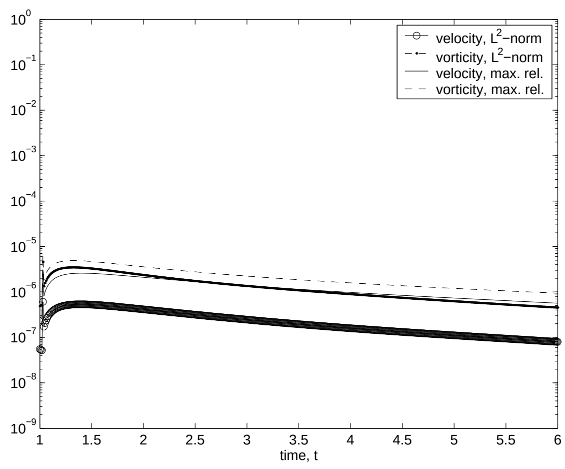
of RK4 are used as the AB schemes are not self-starting). One would have expected, sensibly, that a smaller time step would be needed when using AB3, to obtain a similar accuracy than with RK4. It is not solely due to the lower order of the AB3 scheme being tested in this case, but to the fact that the constant factor in front of that order can be large enough to require using smaller time steps. So, next an experiment is performed with a time step $\Delta t = 0.02$, resulting in approximately a one order-of-magnitude decrease in the errors in comparison with the case above. This is still not enough, however, to equal the RK4 calculation. The errors are detailed on the second table, below.

t	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
1.0	6.04E-07	2.87E-07	4.83E-07	5.51E-08
6.0	1.48E-04	9.53E-05	5.93E-05	1.47E-05

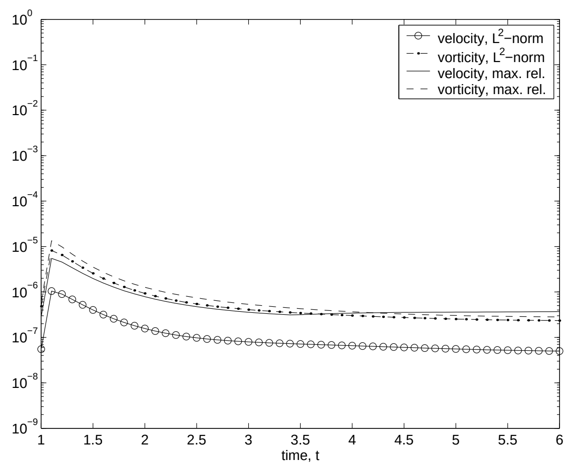
Summary of errors for the experiment in Figure 3.11(a), using AB3.

t	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
1.0	6.04E-07	2.87E-07	4.83E-07	5.51E-08
6.0	8.33E-06	5.35E-06	3.38E-06	8.16E-07

Errors for the calculation using AB3 and $\Delta t = 0.02$, Figure 3.11(b)



(a)



(b)

Figure 3.12: Evolution of errors in vorticity and velocity; Lamb vortex: $\Gamma_o = 1$, $t_o = 1.0$ and $\nu = 0.01$; $N = 7704$. Gaussian blobs: $\sigma = 0.02$, $h = 0.0160$. (a) AB3 time-stepping, $\Delta t = 0.01$ (500 steps), two RK4 steps to start. (b) RK4 time-stepping throughout, $\Delta t = 0.1$ (50 steps).

Decreasing the time step to $\Delta t = 0.01$, at last one observes a final error which is similar to the one obtained with RK4; this calculation required 500 time steps. But in fact, one is able to increase the time step to $\Delta t = 0.1$ with RK4, and still obtain an L^2-norm velocity error of order $\sim 10^{-8}$; although a jump is observed in the errors upon the first time step of almost an order of magnitude, the errors relax back to their initial levels. This calculation required only 50 time steps, which means that 200 velocity evaluations were necessary. Indeed, if one compares this case with the one shown in Figure 3.11(b), where AB3 was used with $\Delta t = 0.02$ (250 time steps), one can conclude that RK4 is more accurate *and* more efficient than AB3.

t	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
1.0	6.04E-07	2.87E-07	4.83E-07	5.51E-08
1.03	6.63E-06	2.98E-06	4.60E-06	6.08E-07
6.0	9.38E-07	5.66E-07	4.54E-07	7.98E-08

Summary of errors for the case shown in Figure 3.12(a).

t	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
1.0	6.04E-07	2.87E-07	4.83E-07	5.51E-08
1.1	1.34E-05	5.50E-06	8.18E-06	1.04E-06
6.0	2.83E-07	3.71E-07	2.34E-07	5.00E-08

Summary of errors for the case shown in Figure 3.12(b)

The numerical experiments presented above indicate that a step size ten times smaller is needed to attain with AB3 the same accuracy as with RK4. Although it is admissible to think that this result may be problem-dependent, the difference is considerable and it is enough to frustrate the arguments in favor of a one-evaluation time stepping scheme *versus* a many-evaluation method.

Another concern in regards to time-stepping refers to the relevance of criteria for the choice of a step size, which one can occasionally find mentioned in the literature (surprisingly, the subject is very often neglected or barely touched upon). One can compare the time step with $\omega_{\max}^{-1}$, for example (when quantities are made non-dimensional, sometimes the value of $\sqrt{\nu}$ is mentioned for viscous flows, in which case $\omega \sim \sqrt{Re}$).

The $\omega_{\max}^{-1}$ criterion is based on estimating the time step such that the particles can follow a curved path accurately. One possible way to look at this is to consider a simple local description of the flow in the incompressible case. Using a Taylor series expansion of the velocity field $\mathbf{u}(\mathbf{x}, t)$ around a point $\mathbf{x}_0$, see [125], one can show that, to linear order in $|\mathbf{x} - \mathbf{x}_0|$, a smooth velocity field is the sum of three terms

$$\mathbf{u}(\mathbf{x}, t_0) = \mathbf{u}(\mathbf{x}_0, t_0) + \frac{1}{2}\boldsymbol{\omega} \times (\mathbf{x} - \mathbf{x}_0) + \mathcal{D}(\mathbf{x} - \mathbf{x}_0), \tag{3.4}$$

where $\mathcal{D}$ is the symmetric deformation matrix. The first term on the right-hand side of (3.4) describes the velocity of an infinitesimal translation, the second term is an infinitesimal rotation in the direction of ω, and the third term represents an infinitesimal deformation velocity. Considering only the contribution of the rotation term, one can estimate then that for an infinitesimal motion $\delta t \sim \left(\frac{1}{2}\omega\right)^{-1}$. Using the maximum vorticity in the field to estimate the time step should then assure that the rotational motions can be integrated accurately. This may be enough in flows dominated by vorticity, but one should probably consider criteria that include the effects of deformation in the more general case.

Examination of the numerical experiments presented in this section permits the following observation. For the calculations presented in Figures 3.10(b) to 3.12(b), the initial condition is such that the value of $\omega_{\max}^{-1}$ is 0.1261, which would indicate that a choice of $\Delta t = 0.1$ is adequate. This time step did indeed give good results in the case shown in Figure 3.12(b), where the RK4 scheme was used. It seems, however, that if the AB scheme is used, the criterion for choosing a time step could produce disappointing results.

3.5 Experiments Using Standard Remeshing

Tests Using Inviscid Vortex Patch: Initialization. To demonstrate the capabilities of the standard remeshing schemes, we start with numerical experiments using an axisymmetric, inviscid vortex patch of order $k = 3$, with vorticity given by Equation (1.12), using a rather low-resolution discretization with $h = 0.1$. This classic test problem is used in [48] (p. 28) to demonstrate different initialization approaches, also using $h = 0.1$ (resulting in about 300 particles, as in the experiments presented here). They report velocity errors at discretization (time-zero) of order $\sim 10^{-2}$, for three different initializations: on a square lattice, on a random and on a "quasi-random" lattice (the coefficient in each case is —reading from the plot, approximately— 2, 8 and 3, respectively). Presently, a square lattice is used, which when initialized using the standard formula (2.72) results in the errors shown on the first row of the table below. This apparently corresponds very well to the example in [48] just cited, although they used a different cutoff function (fourth order linear combination of Gaussians, with $\sigma = 2h$). Note, however, that using Beale's method of circulation processing, the initial errors can be considerably decreased; the result shown on the second row of the table below was obtained with only 5 iterations.

method	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
standard	2.05E-01	1.67E-01	7.18E-02	1.96E-02
Beale's	1.01E-02	2.13E-03	9.87E-03	2.05E-04

Initialization errors for the low-resolution discretization of the inviscid vortex patch using Gaussian blobs, $\sigma = 0.2$ and $h = 0.1$.

Growth of Discretization Errors. Time-stepping the improved initialization obtained with circulation processing, using RK4 and a step $\Delta t = 1$, one can see that the errors soon start to oscillate and grow. The maximum observed L^2-norm error in the velocity is 0.0017 and in vorticity it is 0.0504, both one order of magnitude larger than initially. Furthermore, if one examines the field error of vorticity as it varies with time, one can see that there are spurious structures, which exhibit a periodicity consistent with the initial Cartesian lattice, see Figure 3.13.

Control of Discretization Errors with Standard Remeshing. Calculations were performed next with remeshing using the M_4' kernel; at a frequency of either 10 or 5 time steps, the oscillations and growth of errors were still present, but these were inhibited when remeshing at a frequency of 2 time steps. The time-evolution of the errors for the case with no remeshing is shown in Figure 3.14(a) and for the case with remeshing every 2 time steps in Figure 3.14(b). In addition, if one looks at the field vorticity error in the remeshed case (not shown here), one finds that the spurious structures almost have disappeared. The maximum values, during the long of the experiments (80 time steps), of the different error measurements are detailed on the table below. There is a substantial improvement in accuracy thanks to the remeshing processes, beside the fact that the spurious structures in the vorticity field have been controlled. One inconvenience of the classical

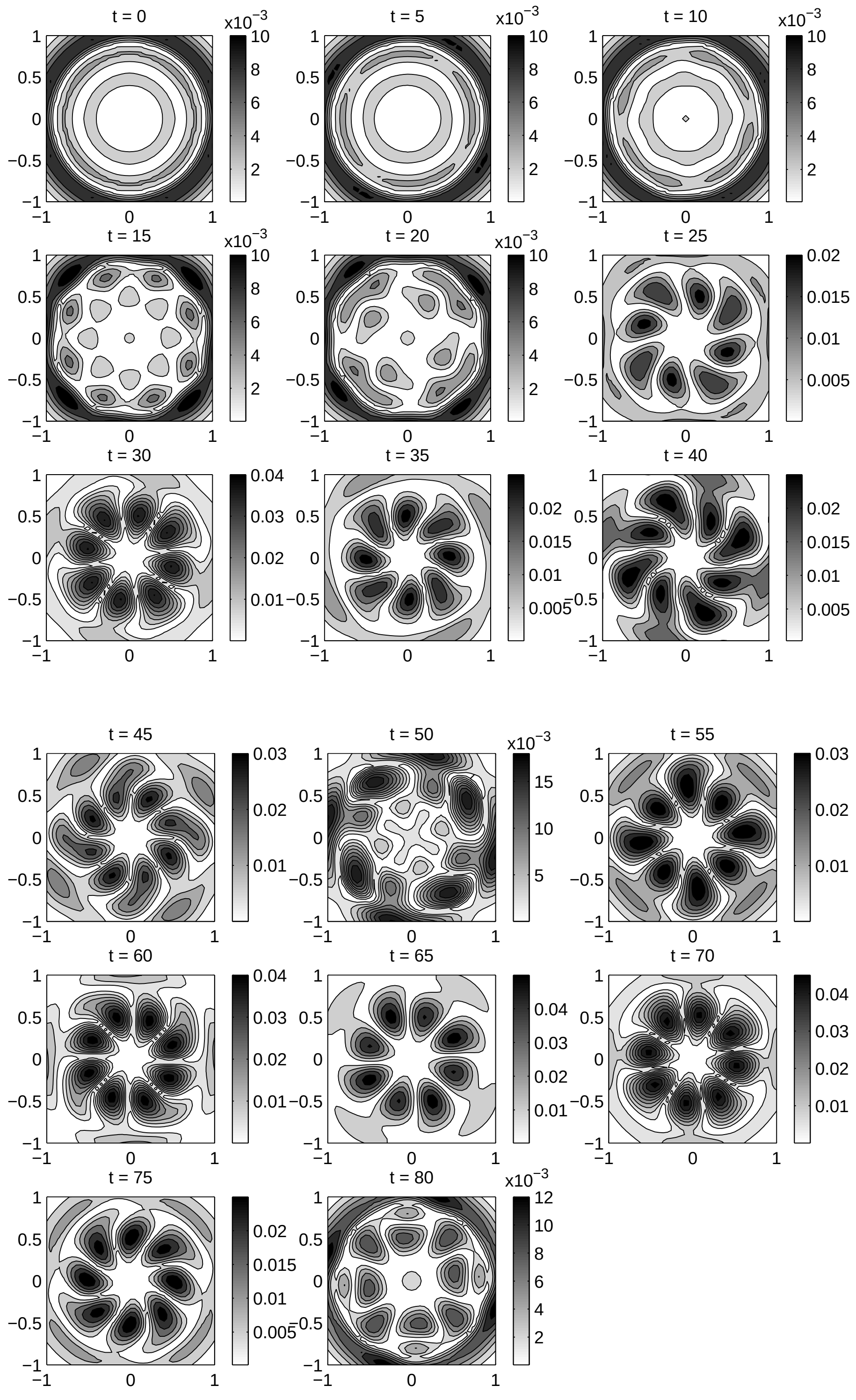
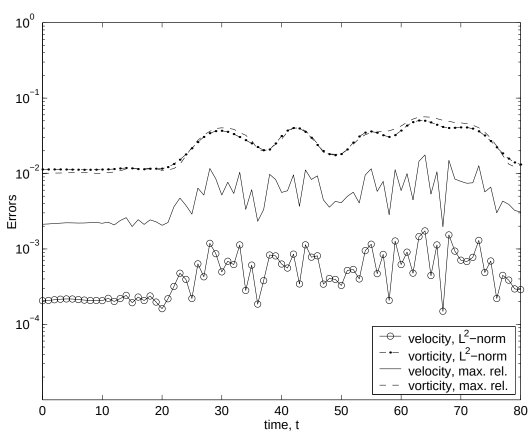
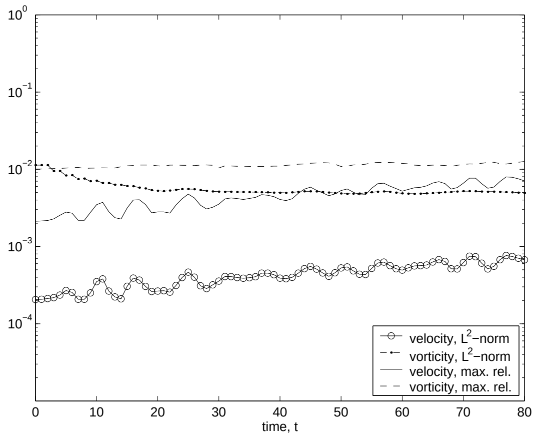


Figure 3.13: Evolution of vorticity error field; inviscid vortex patch. Gaussian blobs: $\sigma = 0.2$, $h = 0.1$. No remeshing.



(a)



(b)

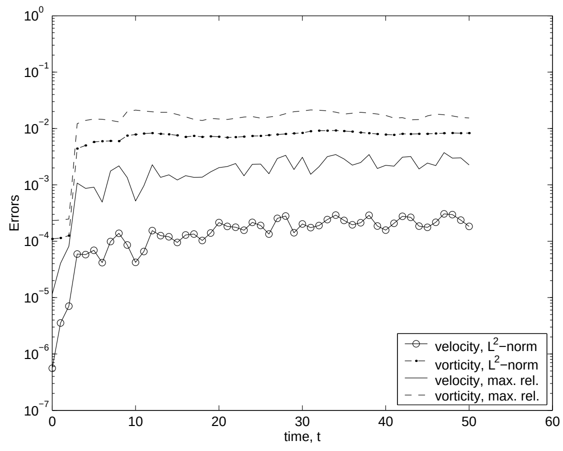
Figure 3.14: Evolution of errors in vorticity and velocity (measured on $\frac{h}{2}$-mesh); inviscid vortex patch. Gaussian blobs: $\sigma = 0.2$, $h = 0.1$. (a) No remeshing. (b) Remeshing every 2 steps, using M_4'.

remeshing scheme becomes apparent, however, as the problem size grows from $N = 305$ to $N = 2960$ during the 40 remeshing processes; this is due to the need for a stencil of particles on the edge of the initial domain, to redistribute the circulation according to the M_4' scheme.

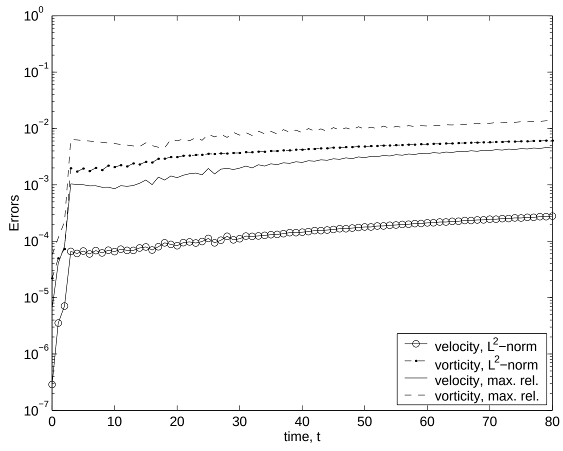
spatial adaption	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
no remeshing	5.64E-02	1.76E-02	5.04E-02	1.73E-03
M_4' every 2 steps	1.26E-02	7.96E-03	1.13E-02	7.64E-04

Maximum errors for the calculations of the inviscid vortex patch using Gaussian blobs ($\sigma = 0.2$, $h = 0.1$), corresponding to Figure 3.14(a) and (b).

Initial Remesh Error and Need for Population Control. Increasing the resolution of the discretization, and thanks to the improved accuracy of initialization using Beale's method of circulation processing, another limitation of the standard remeshing scheme seems to become evident. Using $h = 0.05$, obtained by the combination $\sigma = 0.05$ and $h/\sigma = 1.0$, one can obtain an initial L^2-norm error in velocity of 5.6E-7, and in vorticity of 1.1E-4. With the same time-stepping conditions as before (RK4, $\Delta t = 1.0$), and remeshing every 2 time steps, the errors evolve as seen in Figure 3.15(a). Especially in the case of the vorticity, one can see that after the first remeshing process there is a sharp jump in the errors. In this calculation, the number of particles grows from $N = 1245$ to $N = 3897$ in the 25 remeshing processes (only 50 time steps were advanced in this case). Note the following conflict: on the one hand, the initial vorticity error is of order $\sim 10^{-4}$, which added the fact that $h^2 \sim 10^{-3}$ means that the resolution in terms of circulation is of order $\sim 10^{-7}$. On



(a)



(b)

Figure 3.15: Evolution of errors in vorticity and velocity; inviscid vortex patch. Gaussian blobs: $\sigma = 0.05$, $h = 0.05$. (a) Square lattice; remeshing every 2 steps, using M_4' (measured on $\frac{h}{2}$-mesh). (b) Triangular lattice; hexagonal redistribution every 2 steps, (measured on h-mesh).

the other hand, due to the remeshing stencil and its effect on growing the computational domain, the minimum value of particle circulation at the end of the run is 2.16E-16. Indeed, there are 2284 particles whose absolute value of circulation is less than 1E-7 (almost 60% of the total number of particles). This clearly justifies employing some form of population control.

Tests Using Hexagonal Redistribution Scheme. Subsequently, numerical experiments were performed using a triangular lattice and the hexagonal redistribution scheme of Chatelain. To obtain an accurate initialization, instead of using Beale's method of circulation processing (which assumes a square lattice), the circulation strengths were obtained by solving the linear system (2.75), using —as in [102]— successive over-relaxation. A relaxation parameter of 0.20 was used (chosen by trial and error), and 25 iterations were imposed, to obtain the initial errors shown on the table below. Population control was enforced by deleting particles whose circulation strength fell below 2E-8, hence the problem size grows moderately from $N = 1261$ at $t = 0$, to $N = 1996$ at T.

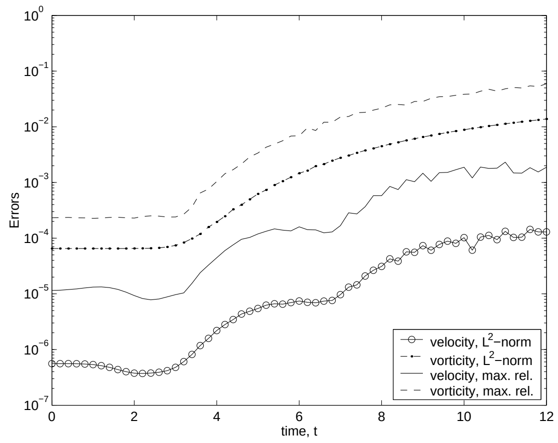
initialization/adaption	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
$t = 0$: Beale's method, 5 its.	2.33E-04	1.15E-05	1.10E-04	5.57E-07
M_4' every 2 steps; $T = 50$:	2.13E-02	3.73E-03	9.23E-03	3.05E-04
$t = 0$: SOR, 15 its.	5.66E-05	6.36E-06	2.20E-05	2.87e-07
hex. redist. every 2 steps; $T = 80$:	1.26E-02	7.96E-03	1.13E-02	7.64E-04

Errors for the calculations of the inviscid vortex patch using Gaussian blobs ($\sigma = 0.05$, $h = 0.05$), corresponding to Figure 3.15(a) and (b).

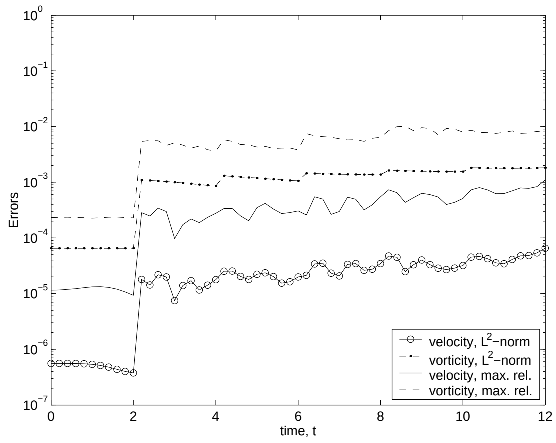
Results. A number of observations are possible from these experiments. The most noteworthy is that both using the third-order M_4' remeshing scheme and the hexagonal redistribution (also third-order), there is a jump on the errors upon the first remeshing process: an "initial remesh error". This can be seen only when the resolution is moderately high, and then, only with a sufficiently accurate initialization. Note that if standard initialization had been used, the time-zero errors would have been such to make it impossible to see this initial remesh error. On the other hand, the usual quantities used to measure the performance of remeshing schemes, *i.e.*, conservation of total circulation and linear impulse, would indicate excellent results in these remeshing experiments. In the case shown in Figure 3.14(b), the maximum change in total circulation during a remeshing process was 8.7E-15, and in linear impulse it was 1.04E-16. Similarly, for the more resolved experiment shown in Figure 3.15(a), total circulation was conserved to 9.3E-15, and impulse to 2.5E-16, on each remeshing process. In the experiment shown using the hexagonal redistribution scheme, the measured maximum change in total circulation was 1.38E-6 and in linear impulse it was 5.0E-8; however these larger values are due to the effect of population control. Other numerical experiments with this scheme, not presented here, conserved total circulation to order $\sim 10^{-15}$, just like M_4' —but produced, *e.g.*, a fivefold increase in the number of particles in 8 remeshing processes with $h = 0.1$, which justifies population control—. If one compares the actual vorticity and velocity errors, in contrast, one observes slightly more accurate results with the hexagonal redistribution scheme, in comparison with the M_4' scheme (that is comparing the errors at the same time, $t = 50$; note that the triangular lattice used a 'stretched' value of h for an equivalent resolution to the square lattice).

Figure 3.16 shows the errors of experiments using the same discretization parameters as above, $\sigma = 0.05$, $h/\sigma = 1.0$ (square lattice), but with a smaller step $\Delta t = 0.2$ and for a shorter time. One can observe more clearly how the errors start growing, and what happens upon remeshing. Note that the turn-around time for the initial condition is $T_v \approx 20$ (the maximum tangential velocity is 0.1737, at a radius of 0.575). Initialization was performed using 5 iterations of circulation processing (but the errors are slightly different than the case in Figure 3.15(a) because the errors were measured on a coarser grid). Remeshing does provide an improvement in accuracy at $t = T = 12.0$; the initial remesh error at $t = 2$, however, is of almost 2 orders of magnitude. Once again, note that this jump in the errors would not have been visible if circulation processing had not been performed to improve the accuracy of initialization (by almost four orders of magnitude, with 5 iterations).

To further emphasize the importance of an accurate initialization to be able to see the error introduced by remeshing, consider similar calculations presented in the literature. For example, using the same test problem, an example is provided in [48], p. 235, where different remeshing schemes are compared, with $h = 0.05$ as here (the type of cutoff is not specified). Reading approximately from the plot, one sees that at time-zero the velocity error was 1E-3, and without any remeshing the final error was 2^+E-2. Remeshing at every time step with the Λ_2 scheme, the final error is 2^-E-2,



(a) No remeshing.



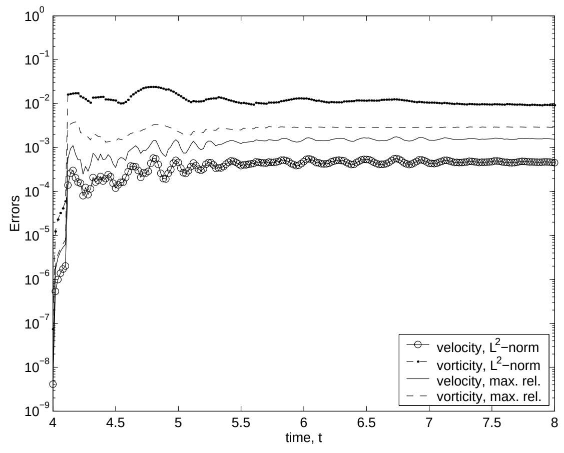
(b) Remeshing every 10 steps with M_4'.

Figure 3.16: Experiments with an inviscid vortex patch and Gaussian blobs, $\sigma = 0.05$, $h/\sigma = 1.0$ and $N = 1245$, RK4 $\Delta t = 0.2$ (errors measured on h-mesh). Population control at $\Gamma_i = 2\text{E-}10$.

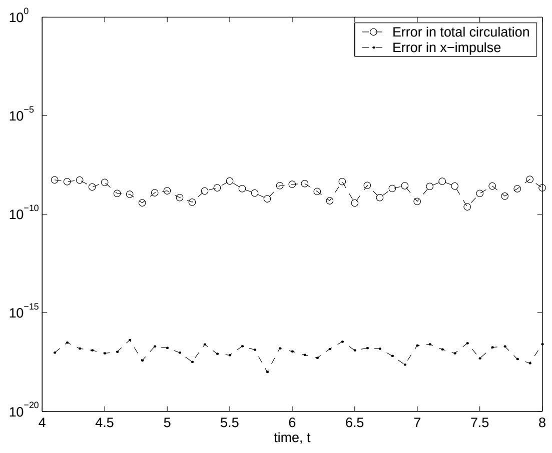
representing a slight improvement. With M_4', in contrast, the final error is 2E-3, thus maintaining the initial accuracy quite well.

Similar Remeshing Experiments in the Literature. There is an experiment in [52], where, using the same test problem, different remeshing schemes are compared. The initial particle spacing is $h = 0.05$ and the final time $T = 50$; one can read from the plot that the time step was 0.5, but no mention is made of the time marching scheme used. The authors state that a second order interpolant must be considered too dissipative and higher order formulas are required. But between the two third-order schemes, their experiment again shows M_4' to give much better results, and they choose this scheme for the applications presented (including a vortex-wall interaction problem, vortex shedding behind a circular cylinder, and behind an array of two cylinders). To compare the experiment of [52] with the ones presented here, we are forced to extract more details from their plot. One can read that the initial velocity error was around 4E-4; they do not specify what type of cutoff was used, but we suspect that this experiment was part of the thesis of [151], in which case a fourth-order algebraic cutoff was used. Reading from the error plot in [52], the final L^2-norm velocity error without any remeshing is 4E-2, hence two orders of magnitude in accuracy were lost with respect to initialization. Remeshing at every step using the Λ_2 kernel, the final error decreases slightly to 2E-2. With both the Λ_3 and the M_4' kernels, in contrast, the final error is 2E-3, a one order of magnitude improvement. Note that the final errors are very similar to the example in [48], p. 235, for the three cases, even though the initial error was one order of magnitude smaller in [52].

The initial L^2-norm velocity error in the experiment shown here in Figure 3.16(b) is 5.57E-7. In an experiment with M_4' remeshing every 5 time steps using $\Delta t = 0.5$, the final error was 3E-4, which



(a) Evolution of errors, measured on h-mesh.



(b) Standard remeshing error measurements.

Figure 3.17: Experiments with a Lamb vortex: $\Gamma_o = 1$, $t_o = 4.0$ and $\nu = 0.0005$; Gaussian blobs with $\sigma = 0.02$, $h = 0.0160$. RK4 time-stepping, $\Delta t = 0.02$. Remeshing every 10 steps with M_4': compare with Figure 3.10(a).

decreases to 1.4E-4 with remeshing every 2 steps (these cases are not shown in the figures). This is a one order of magnitude improvement over the case in [52]. An interesting observation is that in these experiments there was once again a noticeable jump in the errors on the first remesh, bringing them immediately to order 10^{-5}. In the experiment of [52] the initial error was already one order of magnitude larger than this, so no jump can be seen.

Test Using Lamb-Oseen Vortex. Finally, a remeshing experiment using the Lamb vortex is presented in Figure 3.17, where the M_4' scheme was used every 5 steps on the same initial condition as that of Figure 3.10(a). The use of remeshing improved the vorticity accuracy by about one order of magnitude. But once again, there is a significant increase in all the errors upon the first remeshing process. In Figure 3.17(b) are presented the absolute errors in total circulation and linear impulse on each remeshing process, and one can be satisfied that these are indeed small (population control was enforced by deleting particles whose strength was below 2E-10). A summary of error measurements for this experiments is presented in the table below.

t	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
4.0	5.95E-09	7.01E-09	7.35E-08	4.12E-09
4.02	2.00E-06	1.64E-06	1.23E-05	5.33E-07
4.1	3.10E-03	5.59E-04	1.61E-02	1.40E-04
8.0	2.88E-03	1.55E-03	9.20E-03	4.55E-04

Summary of errors for the experiment shown in Figure 3.17.

3.6 Experiments with Rezoning and a Higher-Order Cutoff

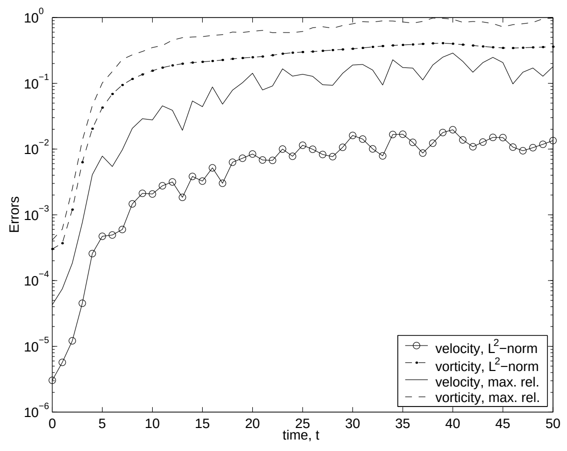
Tests Using Inviscid Vortex Patch: Faster Growth of Discretization Errors with High-Order Cutoff. Consider again the experiments presented at the beginning of the previous section, using Gaussian blobs on the inviscid vortex patch, with $h = 0.1$. If one uses instead the eighth-order cutoff function of Nordmark, with the same value of h and hence the same number of particles, and with $\sigma = 1.6\sqrt{h}$, there is an improvement of about two orders of magnitude in the accuracy of discretization at time zero (using the standard formula for initialization, with no circulation processing). The initial accuracy is lost very rapidly, however, and at the fourth time step the velocity error is about the same as it was initially with the Gaussian blobs (see Figure 3.18(a), and compare with Figure 3.14(a)). Thereafter, the errors are considerably larger than with the lower-order Gaussians. It was observed consistently in further numerical experiments that the higher-order blobs are much more vulnerable to the Lagrangian distortion of the particle locations. This was likely the motivation for the introduction of the rezoning schemes with the high-order kernels presented in [16]. Without some form of spatial adaption, there is no advantage in using higher order blobs in comparison with simple Gaussians, on the contrary. The observation of the field of vorticity errors as it evolves with time reveals that the high-order blobs produce smaller scale (or higher frequency) spatial structures, see Figure 3.19 and compare with 3.13.

Standard Remeshing. With the addition of standard remeshing every two time steps using the M_4' kernel, an improvement of two orders of magnitude is observed in all errors by the end of the run. The errors in this case are shown in Figure 3.18(b), where once again one can see that there is a noticeable jump upon the first remeshing event. The table below shows the initial errors, the errors at the fifth time step, and the maximum errors during the run for the case without remeshing and the case with M_4' remeshing every 2 steps.

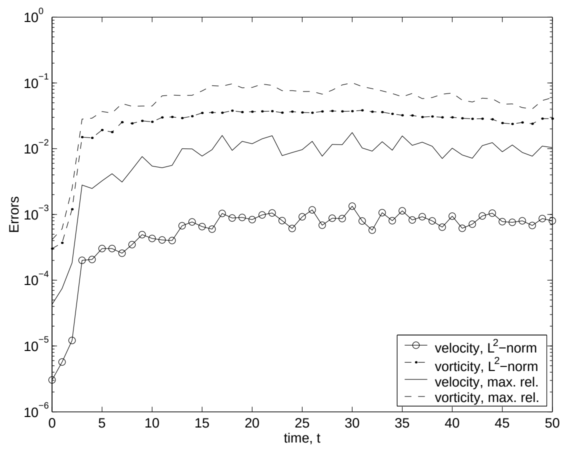
spatial adaption	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
at $t = 0$:	4.14E-04	4.29E-05	3.02E-04	3.05E-06
no remeshing, $t = 5$	0.1012	0.0078	0.0427	4.71E-04
no remeshing, max.	0.9834	0.2880	0.4076	0.0197
M_4' every 2 steps, max.	0.1005	0.0176	0.0383	0.0013

Errors for the calculations of the inviscid vortex patch using Nordmark blobs ($\sigma = c\sqrt{h}$, $h = 0.1$, $c = 1.6$), corresponding to Figure 3.18(a) and (b).

Hexagonal Redistribution Scheme. When initializing on a triangular lattice with the Nordmark blobs, and using the hexagonal redistribution scheme of Chatelain, one obtains very similar results to the ones presented above for the square lattice and the M_4' remeshing scheme. The evo-



(a) No remeshing.



(b) M_4' remeshing every 2 steps.

Figure 3.18: Experiments with an inviscid vortex patch using Nordmark blobs with $\sigma = c\sqrt{h}$, $h = 0.1$, $c = 1.6$. RK4 time-stepping, $\Delta t = 1.0$ (errors measured on $\frac{h}{2}$-mesh).

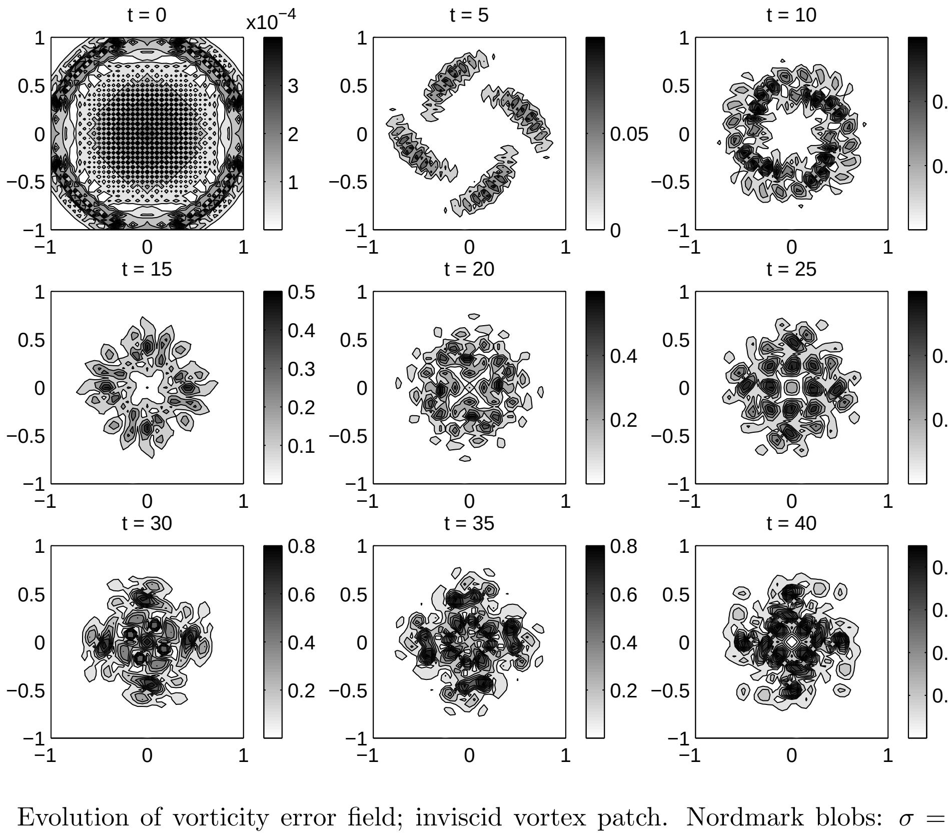
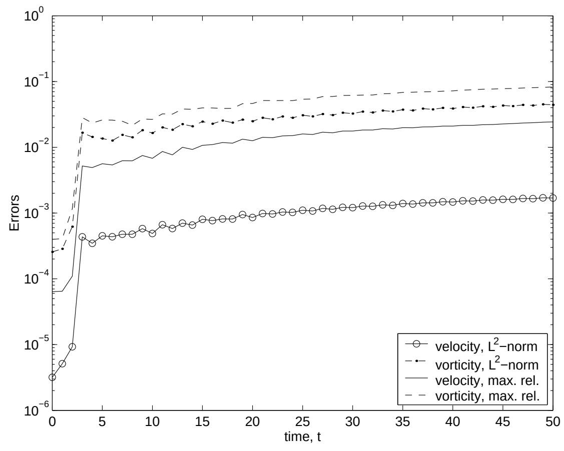
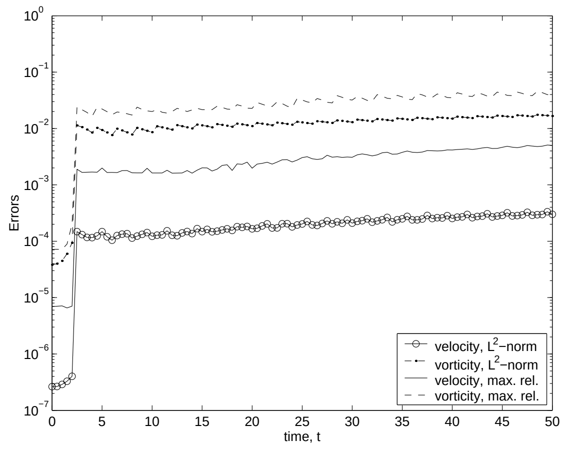


Figure 3.19: Evolution of vorticity error field; inviscid vortex patch. Nordmark blobs: $\sigma = c\sqrt{h}$, $h = 0.1$, $c = 1.6$. No remeshing.



(a)



(b)

Figure 3.20: Experiments with an inviscid vortex patch using Nordmark blobs with $\sigma = c\sqrt{h}$, on a triangular lattice. RK4 time-stepping (errors measured on $\frac{h}{2}$-mesh). (a) $h = 0.1$, $c = 1.6$, $\Delta t = 1.0$; hexagonal redistribution every 2 steps. (b) $h = 0.05$, $c = 1.3$, $\Delta t = 0.5$; hexagonal redistribution every 4 steps.

lution of errors in this case in shown in Figure 3.20(a), and the values of errors at initialization and the end of the calculation are given on the table below. Increasing the resolution by halving the value of h and Δt, one sees a slight improvement in final errors; however, the initial remesh error in this case is about 3 orders of magnitude, see Figure 3.20(b).

time	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
$t = 0$:	3.98E-04	6.37E-05	2.56E-04	3.17E-06
$t = 3$	2.83E-02	5.20E-03	1.67E-02	4.35E-04
$t = 50$	8.25E-02	2.43E-02	4.43E-02	1.70E-03

Errors for the calculations of the inviscid vortex patch using Nordmark blobs on a triangular lattice, corresponding to Figure 3.20(a).

time	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
$t = 0$:	7.18E-05	6.93E-06	3.82E-05	2.63E-07
$t = 2.5$	2.43E-02	1.89E-03	1.13E-02	1.48E-04
$t = 50$	3.96E-02	5.02E-03	1.67E-02	2.99E-04

Errors for the calculations of the inviscid vortex patch using Nordmark blobs on a triangular lattice, corresponding to Figure 3.20(b) ($h = 0.05$).

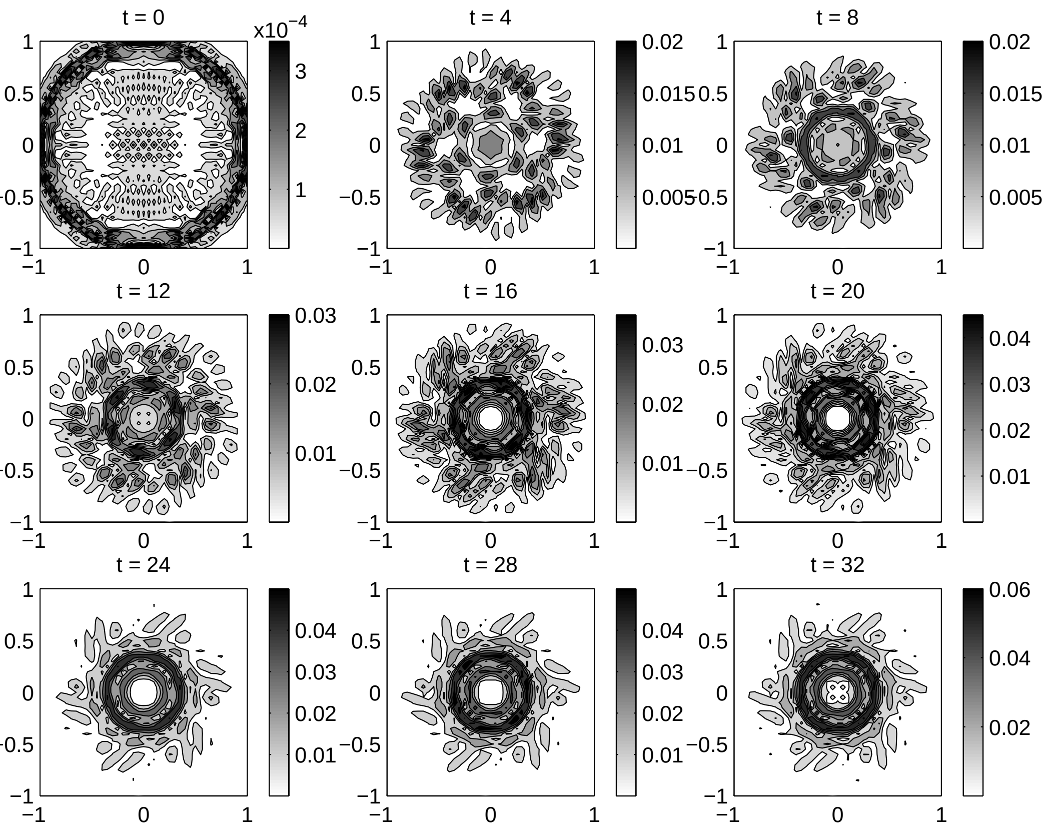
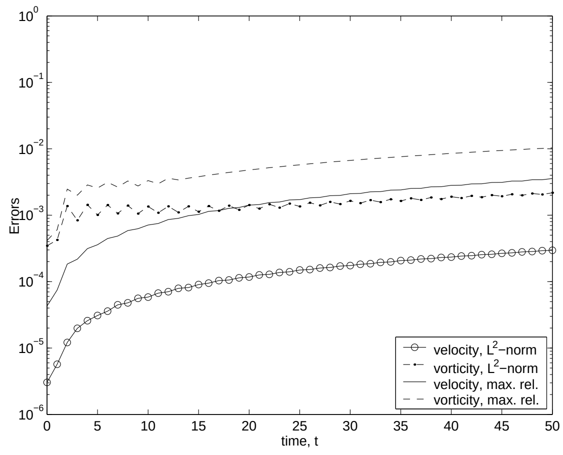


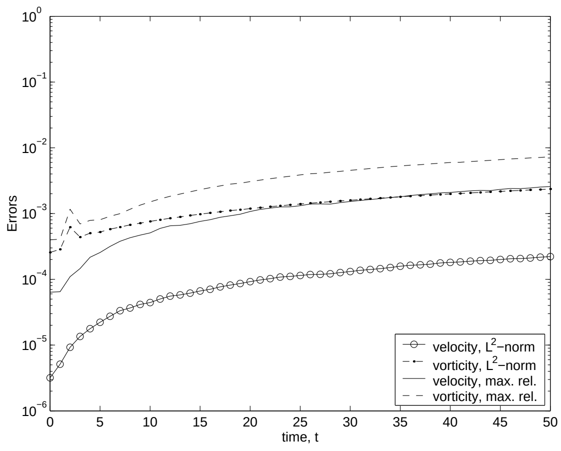
Figure 3.21: Evolution of vorticity error field; inviscid vortex patch. Nordmark blobs: $\sigma = c\sqrt{h}$, $h = 0.1$, $c = 1.6$. Hexagonal redistribution performed every 2 steps.

Note, finally, that the use of a triangular lattice with the hexagonal redistribution leads to a vorticity error field that has an hexagonal pattern, see Figure 3.21 which corresponds to time-slices of vorticity error for the case of Figure 3.20(a).

Results. At this stage, one would be tempted to conclude that there is no advantage in using higher-order cutoff functions. The initial discretization is indeed more accurate than with simple Gaussian blobs, but not only is this accuracy lost dramatically quickly once time-stepping begins, but also the remeshing schemes introduce the same amount of error as with the low-order cutoff. Hence, in the end of a moderately long calculation the errors are very similar with both a second-order and an eighth-order blob. In the experiments presented above, *e.g.*, at a relatively low resolution obtained with $h = 0.1$, the Nordmark blobs with standard initialization provided a two-order-of-magnitude improvement in the accuracy at time-zero compared to Gaussian blobs initialized with circulation processing. This difference is lost in as little as five time steps, due to the fact that higher-order blobs are much more vulnerable to loss of overlap. And when using one of the remeshing schemes, the final errors are not smaller with the higher-order blobs; in fact, for the case of the vorticity, they are slightly larger.



(a) Square lattice.



(b) Triangular lattice.

Figure 3.22: Experiments with an inviscid vortex patch using Nordmark blobs with $\sigma = c\sqrt{h}$, $h = 0.1$, $c = 1.6$. RK4 time-stepping with $\Delta t = 1.0$, and classic rezoning every 2 time steps (errors measured on $\frac{h}{2}$-mesh).

Classic Rezoning. Experiments were performed next using the classic rezoning strategy of Beale and Majda [16], as described on §2.2.2. Note that in obtaining the circulation values of the new particles after spatial adaption by multiplying the local value of vorticity by the cell area, the rezoning scheme is subject to the same errors as the standard initialization. Thus, when using low-order blobs this scheme is not very accurate and produces results which are very much inferior to standard remeshing using a kernel such as M_4' (several experiments were performed, but they are not detailed here). However, seen that the initialization errors are so much smaller with the higher-order blobs, classic rezoning should provide better results in this case.

In Figure 3.22(a) is shown the evolution of the errors for the case of Nordmark blobs initialized on a square lattice, with $h = 0.1$ as before, using classic rezoning every 2 time steps. This calculation results in an order-of-magnitude improvement over the case with M_4' remeshing every 2 steps, and hence two orders of magnitude over no-remeshing. Note, however, that for the case with M_4' most of the accuracy is lost on the first remeshing event, while here the errors seem to slowly creep up along the calculation. Possibly, for longer times, the rezoning option may accumulate more errors.

Rezoning on a Triangular Lattice. Figure 3.22(b) shows the results of an experiment of classic rezoning used on the triangular lattice. The nominal value of h is once again 0.1, but in the calculation this is stretched to obtain an equivalent resolution to the square lattice. In the formula for the particle strengths after rezoning, the cell area for the triangular lattice is used, and thus one is able to obtain good results with rezoning on this lattice. The results show that all errors are slightly smaller in comparison with the equivalent resolution case on the square lattice. And, once again, in comparison with the hexagonal redistribution scheme, there is a one-order-of magnitude improvement with rezoning on high-order blobs. Note that most of the error occurs on the first remeshing event when the hexagonal redistribution scheme was used. A summary of the error values for the two rezoning experiments is given on the tables below.

time	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
$t = 0$:	4.14E-04	4.29E-05	3.46E-04	3.05E-06
$t = 3$	2.00E-03	2.18E-04	8.38E-04	1.98E-05
$t = 50$	1.03E-02	3.57E-03	2.18E-03	2.96E-04

Errors for the calculations of the inviscid vortex patch using Nordmark blobs, rezoning on a square lattice, corresponding to Figure 3.22(a).

time	$E_\omega^{\text{rel,max}}$	$E_v^{\text{rel,max}}$	$E_\omega^{L^2}$	$E_v^{L^2}$
$t = 0$:	3.98E-04	6.37E-05	2.56E-04	3.17E-06
$t = 3.0$	6.96E-04	1.46E-04	4.38E-04	1.35E-05
$t = 50$	7.31E-03	2.57E-03	2.36E-03	2.22E-04

Errors for the calculations of the inviscid vortex patch using Nordmark blobs, rezoning on a triangular lattice, corresponding to Figure 3.22(b).

Results. In conclusion, one may suggest that the classic rezoning strategy of Beale and Majda should probably only be attempted when using high-order blobs. In this case, this technique can result in considerably improved accuracy in comparison with standard remeshing schemes applied with the high-order blobs. Finally, note that classic rezoning requires having a regular mesh of points for the particle locations, such that one can accurately calculate the area associated to a particle.

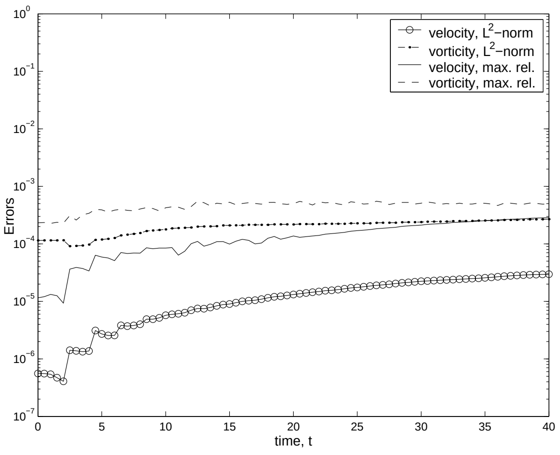
Chapter 4

Formulation and Testing of Fully Mesh-less Vortex Method

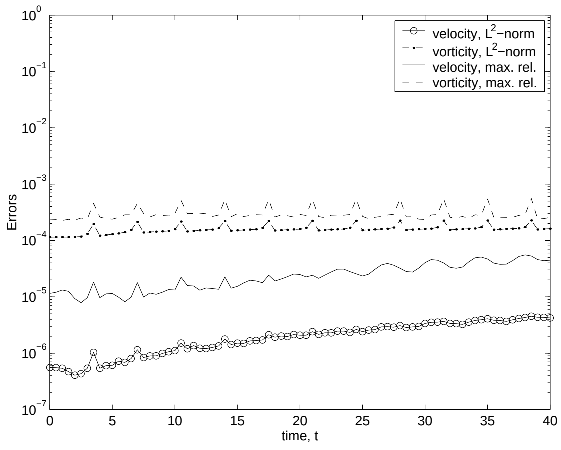
4.1 Preliminary Experiments Using SOR

Motivation. As discussed in the previous section, the classic rezoning strategy requires a regular arrangement of particles, where one can calculate accurately the cell area associated with each particle. This works on any rectangular lattice, including the lattice formed when particles are the vertices of equilateral triangles. This must be considered as a very restrictive geometry, however, and naturally one can expect problems when complex boundaries are present. Furthermore, rezoning is only really appropriate when using high-order vortex blobs. The standard remeshing schemes, on the other hand, are formulated to interpolate each particle's circulation onto a stencil of locations on a grid. In addition, as demonstrated by the numerical experiments presented in the previous section, they introduce noticeable interpolation errors. These experiments evidence that remeshing is a strategy that, although providing control of the errors due to Lagrangian distortion of the particles, limits the accuracy that one can get with the vortex method.

In §2.3, it was related how the function approximation community has carried out considerable work on mesh-less interpolation with radial basis functions, where the formulation has several similarities with the vortex blob spatial discretization. In fact, the vortex method is formulated as a mesh-less method, but is burdened with grids only because of the need for spatial adaption. In view of this, one can formulate a spatial adaption scheme that is akin to the concept of rezoning, but where the circulation strengths of the new particles are obtained by solving the RBF interpolation problem, so that a regular mesh is not needed. To explore this idea, a few experiments were performed where successive over-relaxation (SOR) was used with an under-relaxation parameter to solve for the circulation values. This scheme was chosen due to it having proved successful previously for obtaining accurate initializations, *e.g.*, in [102], and also here, for the case shown in Figure 3.15(b) with a triangular lattice.



(a) RBF spatial adaption every 4 steps;



(b) every 7 steps, less aggressive population control.

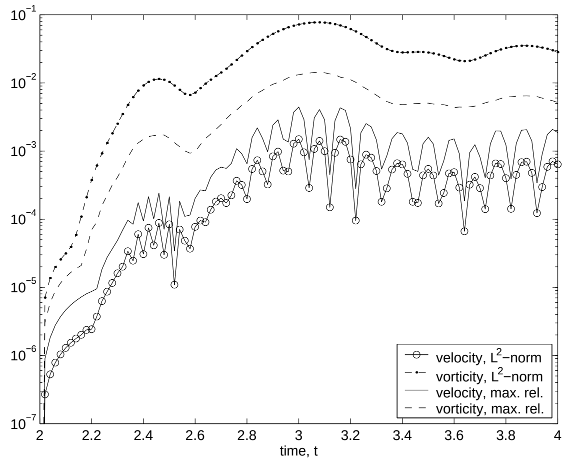
Figure 4.1: Experiments with an inviscid vortex patch using Gaussian blobs and square lattice, $h = 0.05$, $\sigma = 0.05$. RK4 time-stepping, $\Delta t = 0.5$ (errors measured on h-mesh).

Tests Using Inviscid Vortex Patch. Using the inviscid vortex patch test problem, numerical experiments were performed in which the particle locations were restarted on a square lattice every few time steps (as in remeshing and classic rezoning) and the values of circulation were obtained with SOR. The regular lattice is used only for simplicity at this stage, and is not required by the formulation. The vorticity value at each node of the lattice (new blob locations) is obtained by summing the effect of all the existing (old) particles, and then the new particle strengths are solved for, as in the initialization problem given by (2.75). The relaxation parameter was 0.2 and 15 iterations were imposed (hard-coded). In Figure 4.1(a) are plotted the errors obtained when this form of spatial adaption was performed every 4 time steps, with $\Delta t = 0.5$ —giving the same time-frequency of spatial adaption as the case with M_4' shown in Figure 3.16(b), where $\Delta t = 0.2$ and remeshing was performed every 10 steps— and in addition population control is enforced at Γ_i = 2E-6. There is a small jump of the velocity errors on the first few spatial adaption processes, but the final *vorticity* error shows a one-order-of-magnitude improvement over the remeshing case shown in 3.16(b). Note that this latter computation was only carried out to a time of $T = 12$, whereas the case of Figure 4.1(a) goes to $T = 40$. If one compares with the case shown in Figure 3.15(a), where the time step was $\Delta t = 1.0$, remeshing was performed every 2 steps, and $T = 50$, one detects an improvement of one order of magnitude in all error measurements, except the maximum vorticity error, which is improved by *two* orders of magnitude. The final maximum error of vorticity for the case shown in Figure 4.1(a) is 4.9E-4.

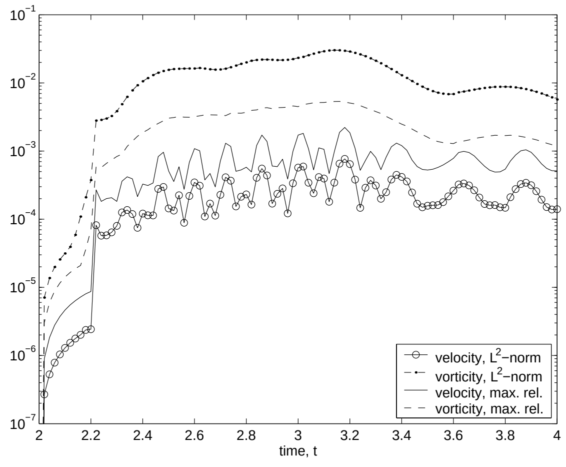
The case shown in Figure 4.1(b) corresponds to a less frequent spatial adaption (every 7 steps) and a less aggressive population control, enforced at Γ_i = 2E-8. In this case, the final L^2-norm velocity error is 4.22E-6, demonstrating a two-order-of-magnitude improvement over a calculation with standard remeshing using M_4' (the case shown in Figure 3.15(a)). In this case, the frequency of spatial adaption seems to be at the limit, as it is performed just as the errors start to tip upwards. We found that if spatial adaption was carried out every 10 time steps, the errors were not controlled.

Results. These preliminary exercises using the RBF ideas for spatial adaption indicate that it is possible to obtain long-time calculations with the vortex method of considerable better accuracy than is obtained by remeshing using one of the customary (high-order) interpolation schemes. We emphasize that nothing in this approach forces one to use a square or indeed a regular lattice. This is precisely the most notable feature of RBF interpolation, that it is *mesh-free*, in the sense that no logically ordered lattice is required. Hence, based on this approach one could contemplate a spatial adaption scheme which restarts the particle field onto a well-overlapped but not necessarily regular lattice, in this way allowing for easily conforming to boundaries. A square lattice is, in contrast, obligatory for any remeshing scheme based on tensor products, as it is also for classic rezoning.

Tests Using Lamb-Oseen Vortex. A calculation is performed using the Lamb vortex, with core spreading for diffusion, with initial condition given by: $\Gamma_o = 1.0$, $t_o = 2.0$ and $\nu = 0.001$ ($\omega_{\max} = 39.2$). An initialization using (2.74) with $h = 0.0175$ results in very small errors, but the time step chosen is not small enough to preserve it; after the first time step the L^2-norm velocity error is 2.7E-7, which is acceptable for the present purposes (RK4 is used for time stepping). Without remeshing or rezoning, the final error after 100 time steps is 6.4E-4 (Figure 4.2(a)). Adding M_4' remeshing every 10 time steps there is a fair improvement (about 50% in the velocity, and 60% in vorticity), but one sees a jump on the initial remeshing of one order of magnitude (Figure 4.2(b)). Next, the same calculation was performed with half the time step, so that the accuracy is noticeably improved in the beginning of the run, before the first remeshing event (the velocity error upon the first time step is 5.3E-9), but a much larger jump is seen on the first remeshing. This run is shown in Figure 4.3 on the same plot with a calculation using the RBF-based spatial adaption: they can easily be distinguished although the same line styles were used since the RBF-spatial adaption preserves the initial accuracy extremely well, with no jump on the first event and with a final velocity error of 3E-8. This demonstrates a four-order-of-magnitude improvement with respect to standard remeshing. To obtain the same accuracy with the third-order M_4' scheme, in theory h would have to be 23 times smaller, which means that N would have to be 526 times larger. In three dimensions, the problem size would have to be four orders of magnitude larger! In addition, with the RBF-spatial adaption we have *resized the blobs* to their original small σ upon each processing, so that the spatial adaption scheme has provided the correction for the core spreading method. The core sizes vary from $\sigma = 0.025$ to 0.029 in between each adaption process.



(a) No remeshing.



(b) M_4' remeshing every 10 time-steps.

Figure 4.2: Evolution of the errors in the calculation of a Lamb vortex with Gaussian blobs, $\Gamma_o = 1.0$, $t_o = 2.0$ and $\nu = 0.001$ and $\sigma = 0.025$, $h/\sigma = 0.7$ and $N = 2177$, $\Delta t = 0.02$.

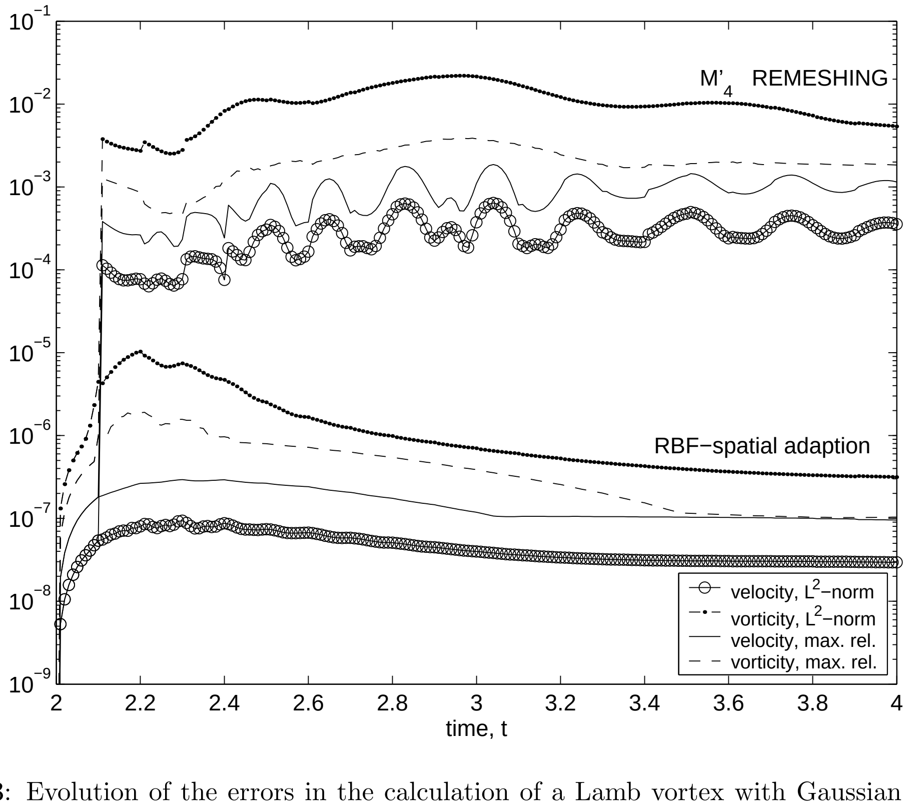


Figure 4.3: Evolution of the errors in the calculation of a Lamb vortex with Gaussian blobs, $\sigma = 0.025$, $h/\sigma = 0.7$ and $N = 2177$, $\Delta t = 0.01$. M_4' remeshing and RBF-based spatial adaption with SOR every 10 steps.

Results and Perspectives. These experiments are very encouraging, as it is demonstrated that there is a potential for highly accurate vortex method calculations, where the accuracy is not limited by the remeshing error, while at the same time an accurate means for controlling core sizes is provided for the application of the core spreading method. Furthermore, the formulation does not require a logically ordered lattice and would potentially allow for variable resolution in the physical domain —this is a very desirable prospect, in particular for wake dominated flows—. Different core sizes in the domain cannot be dealt with directly in the usual remeshing approach based on exchanging circulation; instead, some workers have devised schemes based on carrying out the remeshing on a mapped domain [65, 49, 156, 154], thus allowing for spatially varying cores. An RBF-based spatial adaption method, in contrast, would easily allow for variable resolution in the physical domain. All that is needed is for the particles to carry the information of their individual core size, *i.e.*, they would have one more degree of freedom, and the design of problem-specific rules to enforce the variable resolution during the spatial adaption processing. For example, in a calculation of cylinder flow, one could assign larger and larger maximum core sizes to the particles as they are convected away from the cylinder.

4.2 RBF Spatial Adaption Using Iterative Solution Methods

Motivation. The RBF interpolation used for spatial adaption in the experiments above results in a dense linear system that was solved with SOR, as mentioned. We note that in our first experiments, the same under-relaxation parameter was used as in the previous exercises with the inviscid vortex patch test problem. This resulted in errors that were larger than the cases using standard remeshing schemes (the Λ_3 scheme was also used, resulting in only a slight improvement over M_4'). Subsequently, several tests were performed with different relaxation, and finally a parameter $\varrho = 0.4$ provided the best results; this value was used in the case shown in Figure 4.3, with a hard-coded maximum of 100 iterations and a maximum norm of the residual in the iterations set to 1E-5.

The choice of a relaxation parameter is known to be problematic when using SOR, as it can be problem-dependent. In addition, this solution scheme is not practicable with large numbers of unknowns, more than, say, a few thousand. In particular, it would not be a practical scheme for the solution of flow problems of real fluid dynamical interest. Furthermore, SOR does not lend itself to parallelization. Fortunately, there is considerable experience in the RBF community using iterative methods and preconditioning to solve the difficult linear systems that arise.

Implementation of Iterative Method. As discussed for example in [27], Krylov subspace methods have proved to perform very well in the fitting with global basis function methods. The two principal exponents of this class of methods have been applied with success: the conjugate gradient

(CG) method [87], and the generalized minimal residual (GMRES) method [179]. The first of these is applied when the system matrix is symmetric and positive definite, which is true in many RBF applications, but ceases to be the case when some preconditioners are used. The authors of [27] have made freely available their code for the fitting of geophysical data [1] which uses the Kelley implementation of GMRES [94]. We modified this code to use the Gaussian basis function, as defined in (1.3), with $k = 2$, and incorporated it to the vortex method to be used in the spatial adaption processes.

Numerical Experiments. Using the GMRES method both for initialization and spatial adaption, the numerical experiment previously performed with SOR and shown in Figure 4.3 has been reproduced. The errors at the end of the calculation are of the same order of magnitude as in the experiment using SOR in the spatial adaption step, with a final L^2-norm velocity error of 2.2E-8. This is quite positive considering that no preconditioning was used in this experiment (the condition number was recorded on each spatial adaption process, being always of order 10^8; the maximum number of iterations was set to 500), although the SOR calculations were not preconditioned either. Once again, the core sizes were restarted to their original size of 0.025 upon each RBF-based spatial adaption step, so that the maximum value of σ was 0.0287.

Comparison with Standard Remeshing. Next, numerical experiments were performed attempting to produce comparable accuracies with both the RBF-spatial adaption and the standard remeshing, and find the decrease in h that was necessary in the latter case to obtain this accuracy. For these experiments, a slightly coarser resolution was used as the base case, with $\sigma = 0.03$, $h/\sigma = 0.8$, $h = 0.024$. Population control was enforced at a value of $\Gamma_i = $ 2E-13, resulting in $N = 859$ at initialization for the same initial condition of the Lamb vortex as used above ($\Gamma_o = 1.0$, $t_o = 2.0$, $\nu = 0.001$). A triangular lattice was used, where the remeshing cases utilized the hexagonal redistribution scheme of Chatelain. For the same value of h, the errors obtained using the hexagonal redistribution are plotted in Figure 4.5(a), whereas Figure 4.5(b) shows the results when using the RBF spatial adaption with GMRES as the solution method. The final L^2-norm velocity errors are 2.5E-4 and 1.4E-6, respectively; note that there is a one order of magnitude jump in the velocity error upon the first remeshing event.

A remeshing case is performed next where the value of h has been halved, with $\sigma = 0.015$. This improves the final velocity error to a value of 8.85E-5, while once again there is an initial remesh error of one order of magnitude (Figure 4.6(a)). An experiment with RBF spatial adaption using $\sigma = 0.045$ is shown in Figure 4.6(b), where the final velocity error is 1.4E-4, an increase of about 60% with respect to the previous case. In conclusion, it can be estimated that a value of h between

[1] Available under the terms of the GNU General Public Licence, version 2 or later, with ©2002 by S. Billings. See `http://www.geop.ubc.ca/~sbilling/cgs.html`.

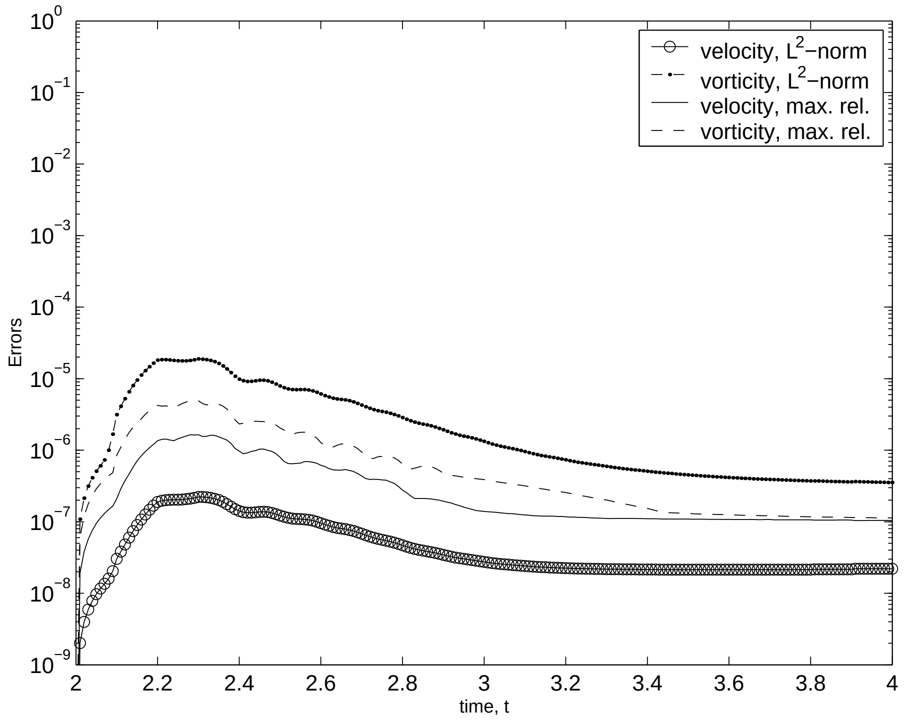
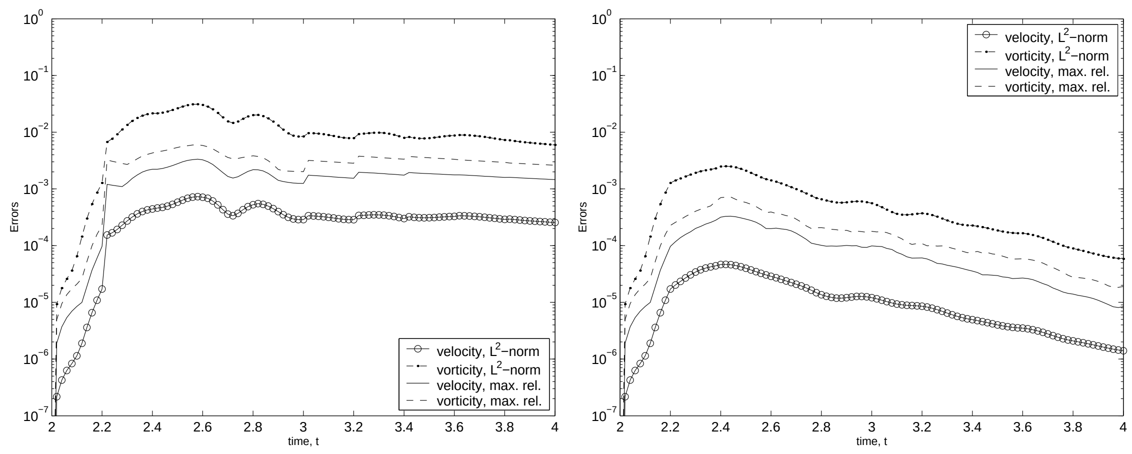


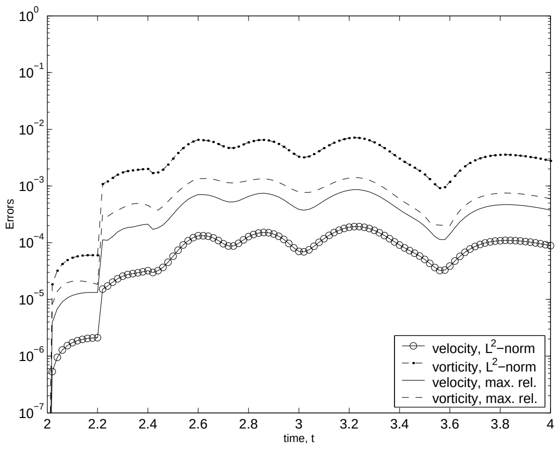
Figure 4.4: Evolution of the errors in the calculation of a Lamb vortex with Gaussian blobs, $\Gamma_o = 1.0$, $t_o = 2.0$ and $\nu = 0.001$ and $\sigma = 0.025$, $h/\sigma = 0.7$ and $N = 2177$, $\Delta t = 0.01$. Spatial adaption using GMRES to solve the RBF linear system for particle strengths, performed every 10 steps.



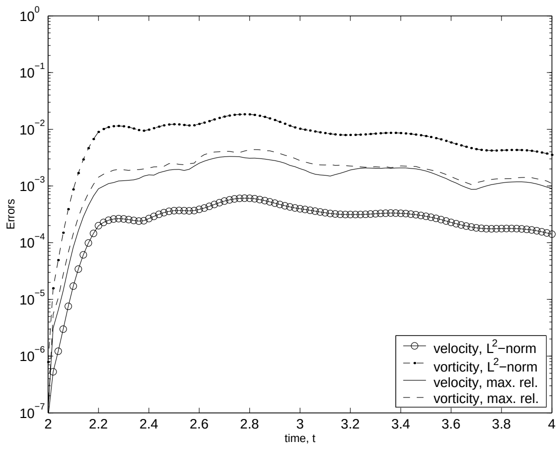
(a) Hexagonal redistribution scheme.

(b) RBF spatial adaption (using GMRES).

Figure 4.5: Evolution of the errors in the calculation of a Lamb vortex with Gaussian blobs, $\Gamma_o = 1.0$, $t_o = 2.0$ and $\nu = 0.001$ and $\sigma = 0.03$, $h/\sigma = 0.8$ and $N(t = 0) = 859$, $\Delta t = 0.02$. Spatial adaption every 10 time steps.



(a) Hexagonal redistribution scheme, $h = 0.012$.



(b) RBF spatial adaption (GMRES), $h = 0.036$.

Figure 4.6: Evolution of the errors in the calculation of a Lamb vortex with Gaussian blobs, $\Gamma_o = 1.0$, $t_o = 2.0$ and $\nu = 0.001$; $\Delta t = 0.02$. Spatial adaption every 10 time steps.

two and three times smaller was necessary to obtain the same accuracy with remeshing compared to the RBF spatial adaption, in this test problem. This is, however, only indicative and a more detailed study at different resolutions would be required for a more conclusive assessment of the comparative accuracy. To do this exhaustively would entail considerable work, taking into account that different parameters affect the results simultaneously. For example, repeating the coarsest case, shown in Figure 4.6(b), but with more frequent spatial adaption —every 5 time steps—, results in a final L^2-norm velocity error of 4.9E-5, which is almost three times smaller than when the adaption was performed every 10 steps, and almost two times smaller that the case with hexagonal remeshing shown in Figure 4.6(a).

Instead of pursuing further a detailed comparative study of the new mesh-less spatial adaption method with respect to the standard remeshing schemes used in vortex methods, we set about to perform a numerical study of the convergence properties of the new method. A first step in this direction is presented in the following section.

Summary of Algorithm. The basic implementation of the viscous vortex method, with core spreading and RBF spatial adaption (as used in the Lamb-Oseen tests above) consists in the following algorithmic steps.

1. Distribute particles in a computational box chosen to be large enough to contain the initial vorticity distribution. These particles do not necessarily have to be on a regular arrangement, but in this section we have used square and triangular lattices for simplicity.

2. Calculate the initial vorticity on each particle location. Obtain an approximate value of the

circulation strength of the initial particles using the standard initialization formula (local vorticity times h^2).

3. Eliminate particles whose approximate circulation strength is smaller than the chosen threshold for population control.

4. Solve an RBF interpolation system (using an iterative method) to obtain the strengths of the initial particles, after population control.

5. One basic time step — Calculate the particle velocities using the Biot-Savart law, and spread the cores to satisfy the diffusion term (note that there is no fractional step here: performing these steps in any order produces the same result!).

6. If spatial adaption is required (determined by a given adaption frequency, which is an input parameter), then: *(i)* distribute new particles as in the initialization step, but first determine the computational box with the maximum and minimum particle coordinates; *(ii)* calculate approximate circulation strengths using the standard initialization formula; *(iii)* eliminate particles whose approximate circulation strength is smaller than the chosen threshold for population control; *(iv)* using the left-over particles, solve the RBF interpolation system for the new particle strengths, as during the initialization, but first reset the particle sizes to their original small value.

4.3 First Convergence Studies

Following the first stage of testing (and a set of calculations corresponding to the first application presented later, in §6.1), the vortex method with mesh-less spatial adaption was implemented in parallel, using the PETSc library [6] in a C++/MPI code. In this implementation, the RBF interpolations are solved using the built-in pre-conditioners and GMRES solver of PETSc. More details of the parallel implementation will be given in Appendix A. Using this parallel code, a number of calculations were performed that use the Lamb-Oseen vortex test case to provide a numerical study of convergence. To produce this study, time marching was performed with a very small time step of $\Delta t = 0.002$, with the purpose of minimizing the effects of time-stepping in the error measurements. The calculations were advanced for 200 time steps, and spatial adaption was performed every 5 steps. The value of the initial inter-particle spacing h was very slowly varied, and the errors were measured on each time step. A summary of the results is shown in the table below, where the vorticity errors are shown at initialization, upon the first time step, and the maximum measured value during the run. As previously, the error measurements correspond to a discrete L^2-norm, and a discrete maximum norm error —that is, the maximum value of the point-by-point difference between the discretized and exact vorticity, normalized by the maximum vorticity—. According to these results, a difference of four orders of magnitude is experienced in the final errors, with a variation in h which is less than twofold. This is a dramatic difference, which is illustrated on the plot of these results given in Figure 4.7. Using a plot of the maximum-norm error measurement *versus* the initial inter-particle spacing h, a functional fit was sought by trial and error; the closest fit found is shown in Figure 4.8.

h	$\sigma_{\max}$	$E_\omega^{L^2}(t_0)$	$E_\omega^{\max}(t_0)$	$E_\omega^{L^2}(\Delta t)$	$E_\omega^{\max}(\Delta t)$	$E_\omega^{L^2}(\max)$	$E_\omega^{\max}(\max)$
0.01725	0.0220	2.2E-11	6.9E-12	3.9E-11	1.2E-11	4.3E-10	1.1E-10
0.0187	0.0238	1.9E-11	1.1E-11	6.2E-11	2.0E-11	5.3E-10	1.5E-10
0.02015	0.0256	2.5E-11	1.2E-11	1.2E-10	3.0E-11	3.6E-09	1.1E-09
0.0216	0.0274	2.6E-11	1.1E-11	2.8E-10	7.8E-11	1.7E-08	4.7E-09
0.02305	0.0292	3.5E-11	2.4E-11	6.8E-10	2.1E-10	8.2E-08	2.3E-08
0.0245	0.0309	3.7E-11	1.6E-11	1.6E-09	4.7E-10	3.2E-07	9.2E-08
0.02595	0.0327	6.3E-11	2.7E-11	3.5E-09	1.1E-09	1.3E-06	3.8E-07
0.0274	0.0345	1.8E-10	9.5E-11	8.5E-09	2.2E-09	4.0E-06	1.1E-06
0.02885	0.0363	5.6E-10	3.0E-10	2.9E-08	5.6E-09	1.0E-05	2.9E-06

Numerical convergence of the mesh-less vortex method, using Lamb-Oseen tests.

The results of this numerical convergence exercise, as shown in Figure 4.8, suggest spectral-like convergence. In fact, after trying many different functions in terms of powers of h to produce an approximate fit to the data, functions of exponential type were tried because of the fact that the literature on radial basis function interpolation offers an error estimate bounded by $e^{-\delta/h^2}$, for the Gaussian RBF [187] —where, in that case, h represents the fill distance, as defined by (2.99)—. It must be recognized, however, that the test problem of the Lamb-Oseen vortex is a very benign test, and possibly it is the reason why this high convergence rate is achieved. In particular, as will be further discussed in §5.1, the axisymmetric flow does not suffer the effects of convection errors. Hence, this experiment is possibly showing the best accuracy that could be obtained. On the other hand, it does demonstrate that the new *spatial adaption* scheme is not a limitation on the accuracy of the vortex method —in contrast with the case of the standard remeshing schemes—.

4.4 Practical Considerations of Efficiency

The use of RBF interpolation for spatial adaption of the particle field, as described in the previous two sections, entails the evaluation of the vorticity at each new location, induced by the old particle set (which is also the case when classic rezoning is used). This is an $O(N \cdot M)$ evaluation, for M

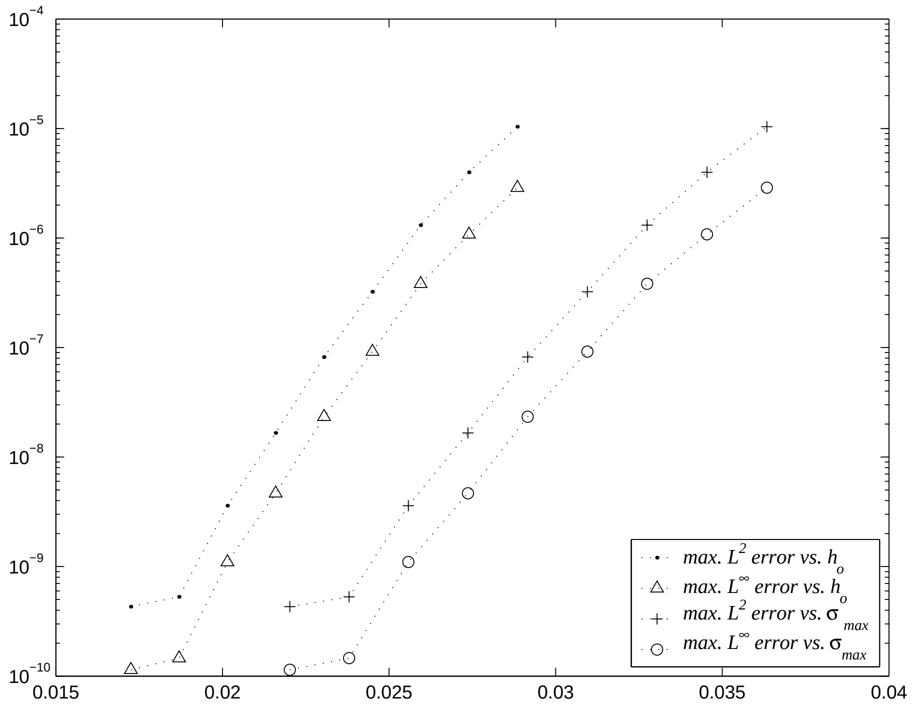


Figure 4.7: Plot of the maximum of two vorticity error measures during 200 time steps *vs.* the inter-particle spacing h and the maximum core size $\sigma_{\max}$.

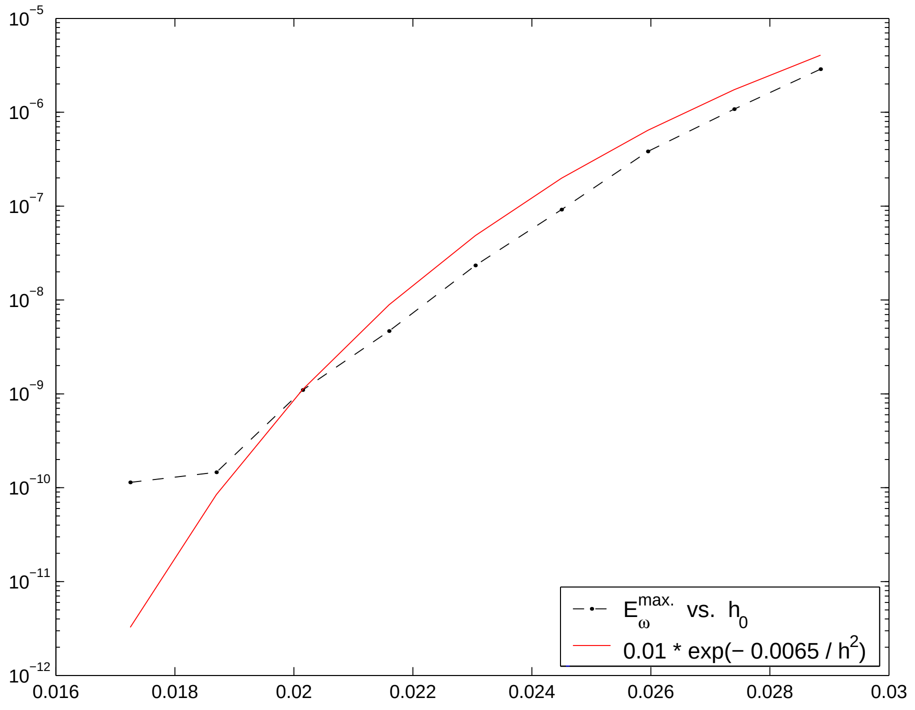


Figure 4.8: Plot of the maximum vorticity errors during 200 time steps *vs.* the inter-particle spacing h and an exponential function of h that shows an approximate fit to the error behavior.

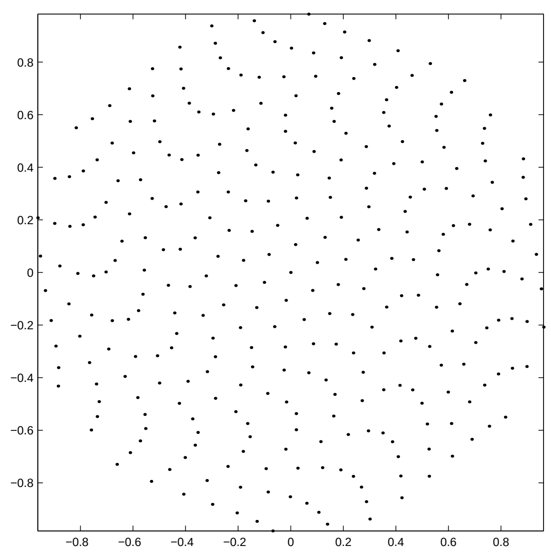
new particles and N old ones, which is approximately N^2 if the vorticity support is not growing very much. Although this appears to be a high computational cost, this evaluation is liable to be performed to arbitrary approximation using a fast summation method, in the same way as the Biot-Savart velocity evaluation. Indeed, the applicability of the FMM for the evaluation of RBF interpolations has been recognized for some time [20]. A method that was specifically designed for the fast evaluation of RBF interpolation functions was presented in [21], which instead of the far field expansions used in the FMM utilizes expressions in terms of the moments of the data. This has many advantages for the applications of scattered data interpolation, where many of the basis functions used do not have the asymptotic expansions that the FMM requires. For the purposes of the vortex method, considering the pervasiveness of the Gaussian blobs, the standard FMM may be sufficient. In any case, there are a number of techniques available that may be used to reduce the complexity of the evaluation to a more tractable number of operations.

The fast evaluation of the RBF interpolations, plus the use of preconditioning and iterative methods of solution of the RBF linear systems result in a feasible method for large numbers of particles. This combined use of advanced analytical and computational techniques has been demonstrated in interpolation problems with up to 5 million points in two dimensions and a quarter of a million in three dimensions in [19].

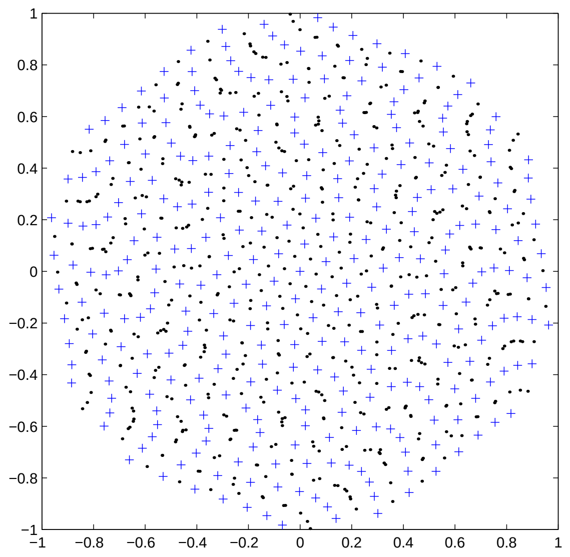
Presently, it has been demonstrated that the idea of RBF interpolation can be applied for the spatial adaption of particles in a Lagrangian method, but efforts at reducing the computational complexity have not been initiated. Nevertheless, the use of the GMRES iterative method with success, and the considerable improvements in accuracy in comparison with the standard remeshing schemes, indicate that this approach is not only viable for practical applications, but also very attractive. Finally, the opportunity for applying the simplest (and exact) viscous diffusion scheme, namely, the core spreading method, and providing the required core size control is notable.

4.5 Truly Adaptive Spatial Refinement

The testing of the RBF spatial adaption has been performed here in a simple way by re-discretizing the vorticity field placing particles on a regular lattice. Nevertheless, the opportunity exists for a truly adaptive spatial refinement technique. Ideas for truly adaptive strategies, for example, have been proposed in [23] to be applied in conjunction with the semi-Lagrangian method that is used for solving advection problems (the semi-Lagrangian method is described in [137], chapter 7). This proposal consists of the coarsening and refinement of computational points by the application of *a posteriori* error estimations for RBF interpolation. Briefly, a "significance" value $\eta(\xi)$ is assigned to each point in the set, $\xi \in \Xi$, which expresses the quality of the interpolation around ξ. The value of $\eta(\xi)$ is used to identify where refinement or coarsening is needed. Coarsening is effected by removal



(a) Particle locations.



(b) Voronoi points, plus particles in blue crosses.

Figure 4.9: Particle locations and Voronoi points, produced with a vortex method calculation of an inviscid vortex patch, after only 5 time steps, with $h = 0.1$ and $\Delta t = 1.0$, $N = 313$, initial triangular lattice.

of ξ, whereas refinement is performed by inserting the Voronoi points of ξ. A strategy such as this would clearly be applicable to the case of vortex blobs, as long as an error estimate can be derived and linked to local measures of density (such as those discussed in §2.3).

Figure 4.9(a) shows the locations of particles placed initially on a triangular lattice, after five time steps of a calculation of an inviscid vortex patch. This illustrates the effect of the differential rotation in opening gaps in the particle field. Figure 4.9(b) shows the same particle locations, this time in blue crosses, plus all the Voronoi points (inside the convex hull of the particle set). It is clear that adding all the Voronoi points would effect a significant *refinement* of the particle set. It would also introduce many unnecessary points that are very close together; hence a subsequent *coarsening*, by eliminating points by some criterion of closeness would be called for.

One must ask oneself, however, whether the use of Voronoi diagrams to generate adaptive spatial refinement algorithms would add excessive computational costs. Indeed, the naïve algorithm for constructing a 2D Voronoi diagram has $O(N^3)$ complexity on a first approach, but can be decreased with only slightly more sophisticated techniques to $O(N^2 \log N)$ [149]. But fortunately, there exist more advanced algorithms that can compute the Voronoi diagram in $O(N \log N)$ computations, namely the "plane sweep" algorithm, also known as Fortune's algorithm [178], and also algorithms based on "divide-and-conquer" approaches, as first proposed in [191]. The $O(N \log N)$ complexity is optimal, however. For more details, see [149].

Chapter 5

Some Numerical Analysis Topics

5.1 Convection Error

What is referred to as "convection error" —physically interpreted, the error originating from convecting the finite-size blobs without deformation with the velocity at their center— is formally a local spatial truncation error. The relevance of this truncation error in the context of the proposed higher accuracy spatial adaption by radial basis function interpolation is the following. The assessment of the accuracy of both the standard remeshing schemes and the new RBF spatial adaption scheme has so far been limited to numerical experiments using simple, axisymmetric test problems. These axisymmetric flows are used by numerous workers to study the accuracy of their codes due to the fact that they allow the calculation of precise error measurements, both in the velocity and the vorticity, thanks to the existence of an analytical solution. There is a drawback, however, and it is that these axisymmetric flows are immune to convection error. For this reason, it is legitimate to ask whether in applications, without axisymmetry, the convection error might be large enough that the high-accuracy spatial adaption is futile. This possibility is discussed in the present section. For completeness, we start by presenting the derivation of an estimate of the convection error, as developed in [113].

In transporting the vortex blobs with the velocity of their center, an error is incurred due to the neglect of shape changes of the fluid element (due to flow strain). This spatial truncation error is estimated by writing the discrete nonlinear convective term as

$$(\mathbf{u}\cdot\nabla)\omega = \nabla\cdot(\mathbf{u}\omega) \approx \nabla\cdot\left[\sum_{i=1}^{N}\Gamma_i\,\zeta_\sigma(\mathbf{x}-\mathbf{x}_i)\frac{d\mathbf{x}_i}{dt}\right] \tag{5.1}$$

The error is then obtained by subtracting the discrete velocity to the exact velocity, thus

$$\varepsilon(\mathbf{x}) = \nabla\cdot\left[\sum_{i=1}^{N}\Gamma_i\,\zeta_\sigma(\mathbf{x}-\mathbf{x}_i)\left(\mathbf{u}(\mathbf{x})-\frac{d\mathbf{x}_i}{dt}\right)\right] \tag{5.2}$$

which, after Taylor expanding the expression inside the big curved brackets, becomes (to first order in $|\mathbf{x}-\mathbf{x}_i|$)

$$\varepsilon(\mathbf{x}) \approx \nabla \cdot \left\{ \sum_{i=1}^{N} \Gamma_i \; \zeta_\sigma(\mathbf{x}-\mathbf{x}_i) \left[(\mathbf{x}-\mathbf{x}_i) \cdot \nabla \right] \mathbf{u}(\mathbf{x}) \right\}. \tag{5.3}$$

Note that for Gaussian blobs the following relation holds:

$$\begin{aligned} \nabla \zeta_\sigma &= \nabla \left[\frac{1}{2\pi\sigma^2} \exp\left(\frac{-|\mathbf{x}|^2}{2\sigma^2} \right) \right] && (5.4) \\ &= \frac{1}{2\pi\sigma^2} \exp\left(\frac{-|\mathbf{x}|^2}{2\sigma^2} \right) \left(-\frac{2x}{2\sigma^2}, -\frac{2y}{2\sigma^2} \right) && (5.5) \\ &= -\frac{\mathbf{x}}{\sigma^2} \zeta_\sigma && (5.6) \end{aligned}$$

Hence, the error functional can be re-written for the particular case of Gaussian blobs with constant core sizes as

$$\begin{aligned} \varepsilon(\mathbf{x}) &\approx \nabla \cdot \left\{ \sum_{i=1}^{N} \Gamma_i \; \sigma^2 \left[\nabla \zeta_\sigma(\mathbf{x}-\mathbf{x}_i) \cdot \nabla \right] \mathbf{u}(\mathbf{x}) \right\} && (5.7) \\ &= \sigma^2 \sum_{i=1}^{N} \Gamma_i \; \nabla \cdot \left[(\nabla \zeta_\sigma \cdot \nabla) \, \mathbf{u}(\mathbf{x}) \right] && (5.8) \\ &= \sigma^2 \sum_{i=1}^{N} \Gamma_i \; \frac{\partial^2 \zeta_\sigma}{\partial x_j \partial x_k} \frac{\partial u_j}{\partial x_k} && (5.9) \\ &= \sigma^2 \frac{\partial^2 \omega}{\partial x_j \partial x_k} \frac{\partial u_j}{\partial x_k}. && (5.10) \end{aligned}$$

This shows that the truncation error is second order in σ, *i.e.*, the modified partial differential equation, actually being solved by the vortex blob method, differs from the vorticity transport equation in a term of $O(\sigma^2)$ when simple Gaussian blobs are used. For this reason, the core parameter, representing the spread of the blobs, is a critical parameter in the vortex method and must be kept small. Note that the error is not proportional to a Laplacian, hence the vortex blob method is not numerically diffusive.

In the particular case of axisymmetric flows, the expression above estimating the convection error turns out to be zero. It can be seen that the expression on the right hand side of (5.10) is symmetric with respect to the vorticity derivatives, so only the symmetric part of the velocity gradient tensor has a contribution. In other words, one can write

$$\varepsilon(\mathbf{x}) \approx \sigma^2 \frac{\partial^2 \omega}{\partial x_j \partial x_k} \frac{1}{2} \left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} \right) = \sigma^2 \frac{\partial^2 \omega}{\partial x_j \partial x_k} D_{jk}. \tag{5.11}$$

In an incompressible flow, D_{jk} has diagonal form in coordinates which are rotated $\pm 45°$ with respect to the radial direction, with eigenvalues of equal magnitude and opposite sign, λ and $-\lambda$, say. In principal coordinates then the result of (5.11) is zero. Hence, when using axisymmetric test problems, the errors that one measures do not include the convection error.

The problem of not including convection error effects in numerical tests is that one could be making extra efforts to obtain more accurate results with the vortex method, only to find that in real non-axisymmetric applications the convection error can dominate. If this were the case, there would be two possible courses of action, aiming at diminishing the convection error. One possibility is using higher-order blob functions, another may be using deformable and rotating blobs. Incorporating these variations into the vortex method and performing studies of the accuracy gained are beyond the scope of this thesis. We note, however, that higher order blob functions, up to spectral order, are proposed in [113], whereas elliptical blobs have been proposed by Rossi [172] to obtain fourth-order accuracy.

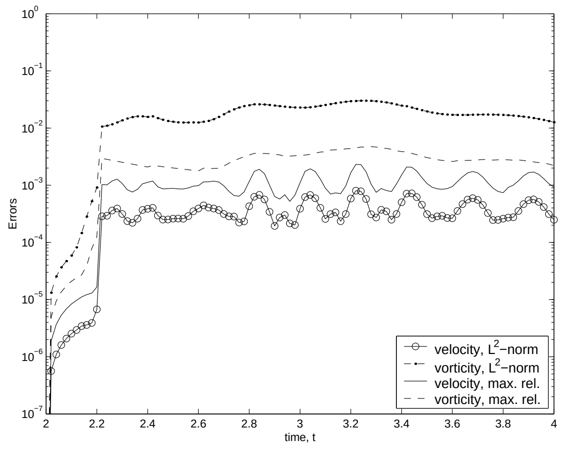
It is acknowledged here that performing numerical experiments using only test problems with axisymmetry could possibly be misleading, due to the absence of convection error. The use of axisymmetric test problems, however, is a widespread practice, as discussed also in §1.2. Although a profound study of the convection error effects is desirable, this is not performed here. Instead, computations are performed of the interactions of viscous vortices without symmetry, where comparisons with other published results are available. These are presented in Chapter 6.

5.2 Choice of Initial Particle Locations

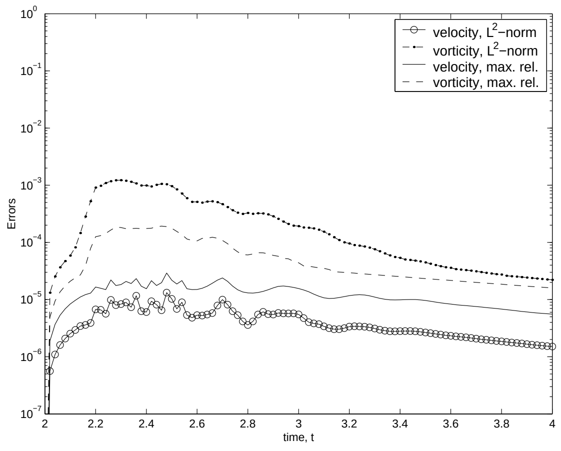
The study presented in §3.2 revealed that the initial lattice of particles can have a significant effect on the error of discretization with vortex blobs. It was shown with the results in Figure 3.5 how an initial triangular lattice of particles provides an increased accuracy in comparison with a square lattice. For equivalent resolutions (*i.e.*, same number of particles per unit area) at a nominal overlap ratio of $h/\sigma = 1.0$ about one order of magnitude improvement was observed at initialization merely by the use of the different —more densely packed, say— lattice.

On the other hand, in the introduction on the subject of radial basis function (RBF) interpolation included in §2.3 it was discussed how the approximation quality of this scattered data interpolation technique depends on a measure of how the nodes cover the domain, given by the *fill distance*, h_X, where X represents the set of nodes. Error estimates exist for the different basis functions which are functions of h_X. For example, when using Gaussian bases, the error functional is bounded by a function $F(h) = \exp(-\delta/h^2)$, with $\delta > 0$ (see [187], Table 2, p. 205).

It is natural to expect that the discretization error of the vortex method will deteriorate as one uses more disordered particle fields, at any time. This is a subject that merits more attention, if only because one of the purported advantages of using RBF interpolation is that no regular arrangement



(a) M_4' remeshing every 10 time-steps.



(b) RBF-based spatial adaption, using SOR.

Figure 5.1: Evolution of the errors in the calculation of a Lamb vortex with Gaussian blobs, $\Gamma_o = 1.0$, $t_o = 2.0$ and $\nu = 0.001$ and $\sigma = 0.03$, $h/\sigma = 0.8$ and $N(t_o) = 1153$, $\Delta t = 0.02$.

of particles is needed. If one were to fully take advantage of this mesh-less formulation, possibly in addition considering a fully adaptive spatial refinement technique with local modification of non-regular particle placements, then one would benefit from a greater understanding of the trade-off that this flexibility brings with respect to accuracy.

As a first approach to studying the effect of particle placement, consider the following numerical experiments using the Lamb-Oseen vortex test problem. The initial condition is given by the same parameters as were used in the experiments of §4.1, see Figures 4.2(a), 4.2(b) and 4.3: $\Gamma_o = 1.0$, $\nu = 0.001$, $t_o = 2.0$. The discretization will be coarser than the previous experiments, with parameters of the base case given by: $h = 0.024$, $\sigma = 0.03$ ($h/\sigma = 0.8$), resulting in $N = 1153$. Population control is provided by deleting particles whose circulation strength is smaller than about 10^{-13}. Time stepping is performed using fourth-order Runge-Kutta with $\Delta = 0.02$. Upon the first time step, the discrete L^2-norm error in velocity is 5.6×10^{-7}, which grows rapidly without any spatial adaption to reach a final value of 3.5×10^{-4}, after 100 time steps. Performing standard remeshing with the M_4' scheme, there is a moderate improvement of about 70% in the vorticity errors and 55% in the velocity errors. As before, a jump in the error measurements is observed upon the first remeshing event, see Figure 5.1(a). When using RBF interpolation for the spatial adaption processes, with the same frequency, a two-order of magnitude improvement is obtained with respect to remeshing, with a final L^2-norm error in velocity of 1.5×10^{-6}, see Figure 5.1(b).

In the next experiment, when performing the spatial adaption processes, a perturbation is made to the square lattice, by adding a sort of random walk to the particles, with a step of maximum size $h/4$. That is, the locations on the lattice were shifted on each coordinate location by a random

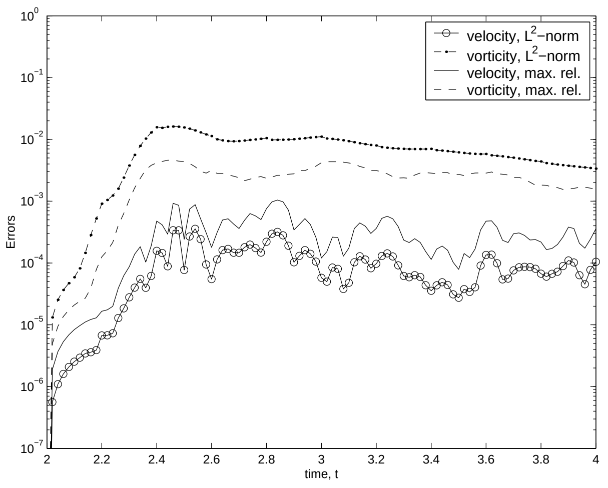


Figure 5.2: Errors for a spatial adaption onto a quasi-scattered lattice; same parameters as Figures 5.1(a) and (b); see more details in text.

fraction of $h/4$, before proceeding to solve the RBF interpolation problem. As a result, the spatial adaption now gives a comparable final accuracy to the standard remeshing case, with a velocity error at T of 1.05×10^{-4}, see Figure 5.2.

There are two reasons for the deteriorated accuracy on the quasi-random spatially adapted particle locations. On the one hand, the fill distance will be larger, because somewhere in the domain there is a larger gap being left by the quasi-scattered lattice. Thus, the approximation quality of the radial basis function interpolation suffers. Secondly, there will also be a decrease in the *separation radius*, q_X, caused by two particles being close together in the quasi-random field. This has the effect of increasing the ill-conditioning of the distance matrix, and thus the SOR method used in this example is less able to find a good solution.

The following set of experiments was conducted next: using the GMRES method to solve the RBF interpolation problem, initializations were obtained for a Lamb-Oseen vortex (this time, $\nu = 0.01$, $t_o = 0.25$) using Gaussian blobs with $\sigma = 0.02$, for varying overlap ratio *and* a variety of quasi-scattered particle placements. This time, the base case was generated with a triangular lattice, and scatter magnitude levels were assigned as a percentage of h. The result, presented in Figures 5.3 and 5.4, reveals the strong dependence of the accuracy on the particle placement. As the overlap ratio is decreased passing through the value of 1, the interpolation onto the regular lattice is more accurate by several orders of magnitude. The interpolation onto a quasi-scattered lattice, in contrast, exhibits a more moderate improvement.

A similar set of experiments was performed using the vorticity distribution of a seventh-order vortex patch, given by (1.12) with $k = 7$. Initialization was performed with Gaussian blobs of size $\sigma = 0.05$. In this case, the difference between the regular lattice and the quasi-scattered ones is less dramatic (see Figure 5.5), due to the fact that the perfectly regular lattice does not provide such a

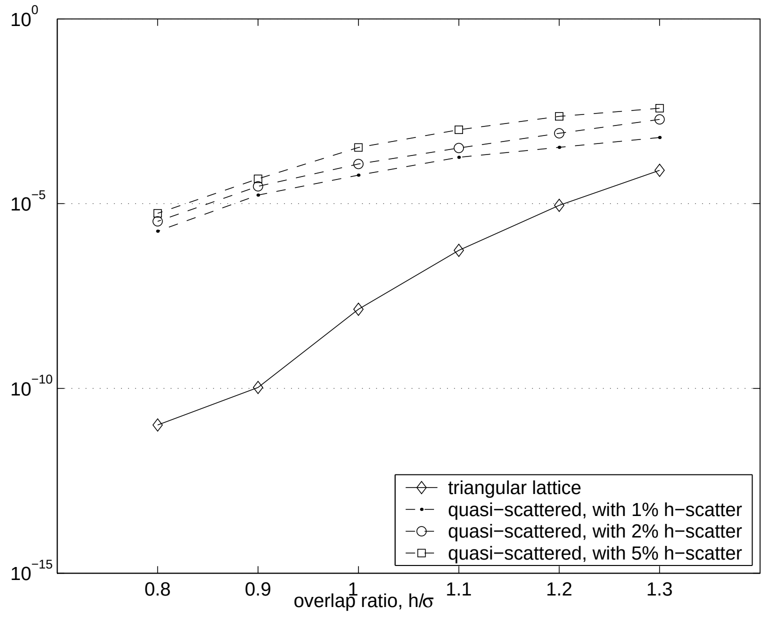


Figure 5.3: Initialization errors of vorticity on regular (triangular) and quasi-scattered lattices, obtained with the GMRES method to solve the RBF interpolation. Lamb-Oseen vortex test problem.

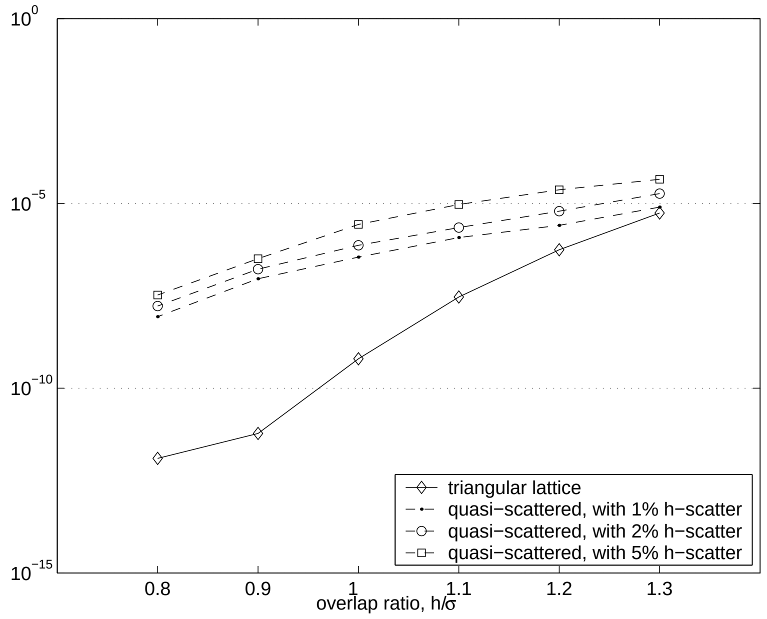


Figure 5.4: Initialization errors of *velocity* on regular (triangular) and quasi-scattered lattices, obtained with the GMRES method to solve the RBF interpolation. Lamb-Oseen vortex test problem.

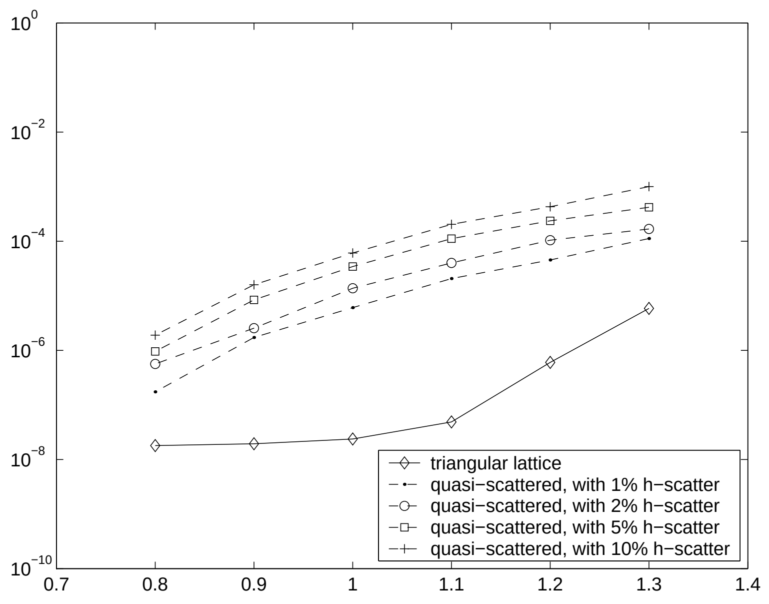


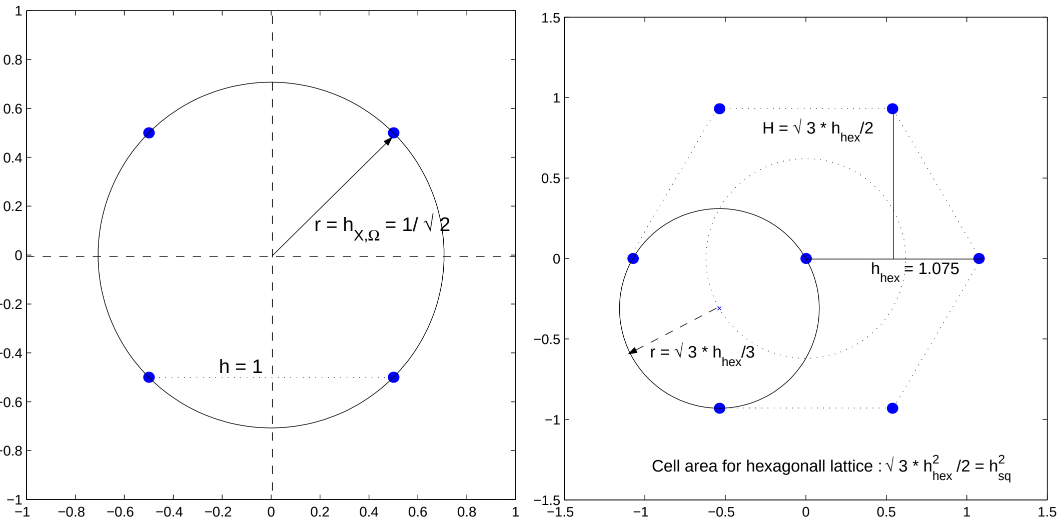
Figure 5.5: Vortex patch (order 7): Initialization errors of vorticity for particles on different regular (triangular) and quasi-scattered locations, obtained with the GMRES method to solve the RBF interpolation.

high accuracy as in the case of a Lamb-Oseen vortex test problem. This responds to the fact that the approximation quality of RBF interpolation also depends on the *regularity* of the function being approximated. In the case of the vortex patch, there is discontinuity of a high derivative, which has an impact on the interpolation quality. Nevertheless, the difference in the accuracy obtained with a perfectly regular lattice and with the quasi-scattered particle placements is considerable.

The table below lists the errors of initialization for the case of the vortex patch (Figure 5.5), using a discrete L^2-norm measured on a grid of spacing $h/2$, for the different initial particle fields. The scatter is measured as a percentage of h, which means that the locations which were initially on the regular lattice are shifted on each coordinate direction by a random fraction of the given percentage of h. On the rightmost column, the table lists a measure of the *uniformity* of the particle set, defined by

$$\rho_X = \frac{q_X}{h_X}, \tag{5.12}$$

where q_X is the separation radius of the set of points X, and h_X is the fill distance, both defined in §2.3. The quantity ρ_X was defined in [72], where it is also pointed out that the set of nodes in a triangular lattice is an example of a set of high uniformity. As illustrated in Figures 5.6(a) and (b), and performing some straightforward trigonometric calculations, the triangular lattice has a value of uniformity of $\sqrt{3}/2 = 0.8660$, whereas a perfectly square lattice has $\rho_X = \sqrt{2}/2 = 0.7071$. This observation elucidates the reason for obtaining higher accuracy with vortex methods based on triangular lattices, as found through numerical experiments in §3.2.



(a) (b)

Figure 5.6: Illustration of the measures of fill distance, h_X, and separation distance, q_X, for square (a) and triangular (b) lattices.

% h-scatter	$E_v^{L^2}$	$E_\omega^{L^2}$	ρ_X
0	1.90E-09	2.37E-08	0.8660
1	9.09E-08	6.07E-06	0.8475
2	2.19E-07	1.37E-05	0.8303
5	7.44E-07	3.43E-05	0.7783
10	2.22E-06	6.08E-05	0.6982

Initialization errors for a seventh-order vortex patch, using Gaussian particles on different quasi-scattered locations.

To obtain the observed values of uniformity reported in the table above, the difficulty lies in calculating the fill distance, as the separation radius is straightforward to obtain by pair-wise comparisons of the particle locations. The fill distance was already associated with the radius of the largest empty circle in §2.3; this is a well-studied concept in computational geometry, see for example [150]. The potential centers of largest empty circles are the interior Voronoi vertices, *i.e.*, those inside the convex hull of the point set. Thus, to obtain the fill distance, three significant algorithms from computational geometry are needed: *(i)* one to obtain the Voronoi diagram with topology, that is, one that gives the Voronoi vertices and information about the associated Voronoi cell; *(ii)* an algorithm to obtain the convex hull of the point set; *(iii)* an algorithm to determine whether a point or set of points is inside or outside a polygonal region.

In summary, this study of the effect of the choice of particle locations on the accuracy of the vortex method discretization reveals that there is great importance to be placed on this aspect. Although the technique of radial basis function interpolation is meant to be applied to *scattered* data locations, the fact is that to obtain good quality of approximation, uniformly distributed or nearly uniformly distributed locations are desirable. This has been recognized in [91], a work which considered the deliberate variation of the data locations with the goal of improving the quality of approximation and conditioning of the RBF interpolation problems. In the application to vortex methods, this is not possible, because the particles must follow the trajectories of fluid elements. It is possible to imagine, however, applying these ideas in the context of a truly adaptive (mesh-less) spatial refinement. This is subject for future research.

5.3 Analysis of Radial Basis Function Interpolation

The introduction given in §2.3 briefly mentioned some aspects of the numerical analysis of radial basis functions, in particular those concepts were introduced that were necessary to carry on the discussions that followed. These basic concepts of the theory of RBF's can be summarized as follows.

- A set of nodes for interpolation $X = \{x_i\}$ is characterized by two density measures: the separation radius, q_X, given by (2.98), and the fill distance, h_X, given by (2.99).
- The separation radius measures the minimum (half) distance from a node in X to its nearest neighbor, also in X. The importance of this density is that it determines the conditioning of the distance matrix, and thus the stability of the solution method.
- The fill distance measures the maximum distance from any point in the domain to its nearest data point in X, and its importance is that it determines the quality of the approximation.
- There is an intimate relationship between the quality of approximation and the conditioning, termed *uncertainty principle*, and phrased: "either one goes for a small error and gets a bad sensitivity, or one wants a stable algorithm and has to take a comparably large error" [186].
- A radial basis function is generally characterized by being of compact or non-compact support, and by being positive definite or conditionally positive definite.
- Existence of a solution is guaranteed for positive definite basis functions, and for conditionally positive definite functions by appending a polynomial, a result proved in [132].

The present section will delve slightly deeper in this subject, for completeness and with the goal of shedding some light on issues that have implications regarding the accuracy, convergence and general performance of the vortex blob discretization. To this effect, some formal definitions and theoretical results will be presented here. As in §2.3, $\Phi : \Omega \times \Omega \to \mathbb{R}$ is a function satisfying translation invariance and radiality, *i.e.*, $\Phi(x,y) = \phi(\|x-y\|_2)$ with $\phi : [0,\infty) \to \mathbb{R}$. The distance matrix, also called interpolation matrix, is built by $\mathbf{\Phi}_{ij} = \phi(\|x_i - x_j\|)$, for points in $X = \{x_1, \ldots, x_N\} \subset \Omega \subset \mathbb{R}^d$ which are scattered. Immediately we note that the data points must necessarily be distinct if the interpolation matrix is to be non-singular. The following is a basic definition in RBF theory.

Definition 1 *A continuous function $\Phi : \Omega \times \Omega \to \mathbb{R}$ is conditionally positive definite of order m on Ω if $\forall N \in \mathbb{N}$, all distinct $\{x_i\} \in \Omega$, and all vectors $\alpha \in \mathbb{R}^N \setminus \{0\}$ satisfying $\sum_{j=1}^{N} \alpha_j p(x_j) = 0$ for all polynomials p of degree less than m, the following quadratic form is positive:*

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k \Phi(x_j, x_k). \tag{5.13}$$

The function Φ is positive definite if it is conditionally positive definite of order $m = 0$.

The importance of this definition lies in its use in the following theorem. Here we follow [188]. The space of d-variate polynomials of degree at most m is denoted by $\pi_m(\mathbb{R}^d)$.

Theorem 1 *Let Φ be conditionally positive definite of order m on $\Omega \subseteq \mathbb{R}^d$, and let $X = \{x_i\} \subseteq \Omega$ be unisolvent in $\pi_{m-1}(\mathbb{R}^d)$,* i.e., *the only polynomial in $\pi_{m-1}(\mathbb{R}^d)$ that vanishes on X is the zero polynomial. Then, given $\{f_i\}$ there is one and only one function $s_{f,X}$ of the form (2.92) with a polynomial $p \in \pi_{m-1}(\mathbb{R}^d)$ that satisfies the interpolation conditions: $s_{f,X}(x_i) = f_i$, $1 \le i \le N$ and $\sum_{i=1}^{N} \alpha_i q(x_i) = 0$, $\forall q \in \pi_{m-1}(\mathbb{R}^d)$.*

In other words, the existence and uniqueness of the solution to the RBF interpolation problem is ensured when the basis function is conditionally positive definite, a result due to [132]. For example, the Gaussians and the inverse multi-quadrics are positive definite on $\mathbb{R}^d$, $\forall d \ge 1$. The generalized multiquadrics $\phi(r) = (-1)^{\lceil \beta \rceil}(c^2 + r^2)^\beta$, c, $\beta > 0$, $\beta \notin \mathbb{N}$, are conditionally positive definite of order $m \ge \lceil \beta \rceil$ on $\mathbb{R}^d$, $\forall d \ge 1$; where $\lceil \beta \rceil$ represents the integer ceiling of β. Proofs are given in [188], but the original result is given in [132] where the the conditional positive definiteness of a function Φ is related to complete monotonicity of derivatives of ϕ, which allows the characterization of these basis functions.

Another important concept in the theory of approximation using RBF's is that of function spaces generated by conditionally positive definite functions. In the simplest formulation, superposition of functions of the form (2.92), with null polynomial part, results in a function space $\mathcal{F}_{X,\Phi}$ given by

$$\mathcal{F}_{X,\Phi} := \left\{ \sum_{j=1}^{M} \alpha_j \Phi(x, x_j) \;/\; \alpha_j \in \mathbb{R} \right\}. \tag{5.14}$$

In other words, the functions Φ define an interpolation method and also define an inner-product space of functions, $\mathcal{F}_\Phi$, when all possible sets of N_X data points are considered which satisfy the unisolvency condition given in Theorem 1. These ideas originate in [122]. The function space $\mathcal{F}_\Phi$, which is a vector space, contains all finite linear combinations of translates of Φ, and it can be completed to form a Hilbert space. Each function $f \in \mathcal{F}_\Phi$ has a unique representation as follows, for $p \in \pi_m(\mathbb{R}^d)$ and a given X:

$$f = p + f_{\alpha,X} \tag{5.15}$$

and thus, the following bilinear form is defined on the space $\mathcal{F}_\Phi$:

$$(p + f_{\alpha,X}, q + f_{\beta,Y})_\Phi := \sum_{j=1}^{N_X} \sum_{k=1}^{N_Y} \alpha_j \beta_k \Phi(x_j - y_k) \tag{5.16}$$

where $p, q \in \pi_m(\mathbb{R}^d)$, and X and Y are two admissible data sets. The bilinear form $(\cdot,\cdot)_\Phi$ above induces a seminorm $|\cdot|_\Phi$, and this is useful for the derivation of error bounds of RBF interpolation. In particular, if f is a function being approximated by the form (2.92), then the bound on the error has the form

$$|f(x) - s_f(x)| \le |f|_\Phi \cdot P(x), \tag{5.17}$$

where the function $P(x)$, dubbed "power function", is the norm of the error functional on $\mathcal{F}_\Phi$ evaluated at x [186]. The power function is bounded by a function of the local fill distance $h(x)$ as

$$P^2(x) \le F\left(h(x)\right). \tag{5.18}$$

For multiquadrics, $F(h) = e^{-\frac{\delta}{h}}$, whereas for Gaussians, $F(h) = e^{-\frac{\delta}{h^2}}$ [124].

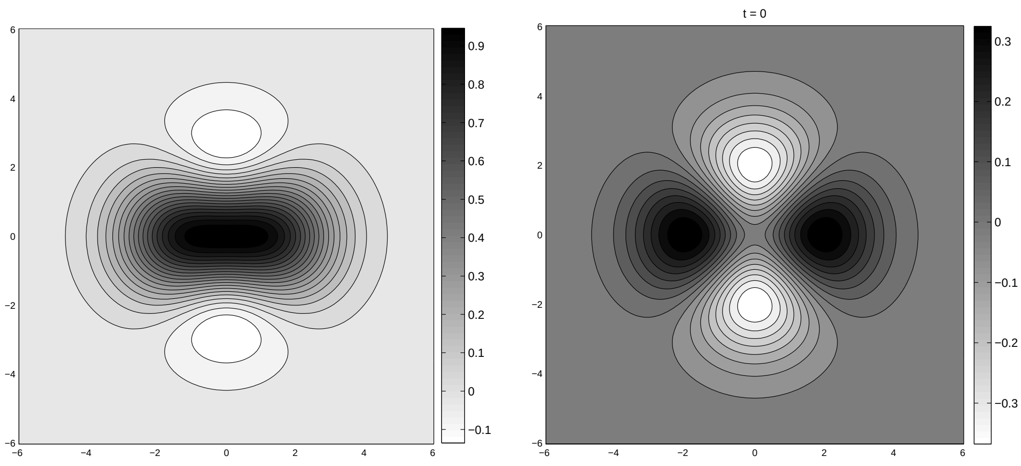
Chapter 6

Computations of Viscous Vortex Interactions

6.1 Perturbed Monopole with Tripole Attractor

Description. When a two-dimensional monopolar vortex is added a quadrupolar perturbation, the principal effect is a localized elliptical deformation of the main vortex (Figure 6.1). One of the relevant questions apropos of two-dimensional vortices with elliptical shapes is whether they decay to an axisymmetric state. The process of "axisymmetrization" is recognized as one of two fundamental processes in the evolution of two-dimensional turbulent flows, the other being vortex merging [130]. These processes participate in the notorious evolution of two-dimensional turbulence to form isolated, "coherent" vortices, that live for many eddy turn-over periods [129]. In [26], the relaxation of linearly perturbed, large-Re Lamb-Oseen vortices was studied numerically using finite difference methods, and it was seen that the non-axisymmetric perturbations decay much faster than the viscous timescale. A mechanism of shear-diffusion averaging [121] is active, whereby the shearing along streamlines causes the winding-up of the non-axisymmetric vorticity into spiral structures, which are then rapidly homogenized due to viscous diffusion. In this case, the axisymmetric state is approached on a $Re^{1/3}$ timescale, as shown in [121]. Note: In this section, the Reynolds number will be defined as $Re = \Gamma/\nu$, where Γ is the total vorticity of the base vortex and ν is the viscosity.

Previous Restults. Fully nonlinear simulations of this flow were presented in [174], where a vortex blob method is used with core spreading and vortex particle splitting [170]. In that work, it was observed that whereas for small-amplitude non-axisymmetric perturbations the flow relaxes to an axisymmetric state, for large enough amplitudes of the perturbation, the flow relaxes instead to a quasi-steady, rotating tripole. The shear-diffusion mechanism in this case is still active, but only the positive portions of the quadrupolar perturbation are mixed while the negative parts form persistent inclusions. This is explained by the creation of a separatrix of the streamlines, when seen in a frame



(a) Total vorticity.

(b) Perturbation vorticity.

Figure 6.1: Initial condition for the perturbed monopole calculations, normalized by $\omega_{o,\max}$.

of reference rotating with the vortical structure. This has the effect of blocking the shear-diffusion mixing between the negative inclusions and the base vortex. Thus, it was suggested that there exists a threshold amplitude separating the domains of attraction of the monopole and tripole.

Relevance. Tripolar vortices have been observed in experiments [202, 203], and they have also been observed in numerical simulations of evolving turbulence [112]. A vortex tripole was observed by satellite imaging on the Bay of Biscay, surviving for several days [153]. Thus, the (rotating) tripole is now recognized as one more "elemental" coherent vortical structure of geostrophic and two-dimensional turbulent flow, together with the (translating) dipole and the (stationary) monopole, the latter being the most pervasive.

Proof-of-concept Calculations. Presently, the vortex method with mesh-less spatial adaption has been used to compute the evolution of a Gaussian vortex monopole, with a quadrupolar perturbation of sufficient amplitude, according to the studies of [174], to produce a quasi-steady tripole. Calculations were performed for Reynolds numbers $Re = 500, 10^3, 5000, 10^4$. The initial condition is given by the base vorticity ω_o with perturbation ω', given below (where $\theta = \arg \mathbf{x}$):

$$\omega_o(\mathbf{x}) = \frac{1}{4\pi} \exp\left(\frac{-|\mathbf{x}|^2}{4}\right), \qquad \omega'(\mathbf{x}) = \frac{\delta}{4\pi}\, |\mathbf{x}|^2 \exp\left(\frac{-|\mathbf{x}|^2}{4}\right) \cos 2\theta. \tag{6.1}$$

For the case $R = 10^4$, the evolution of the total vorticity, normalized by the maximum vorticity at $t = 0$, is shown in Figure 6.2. As observed in the experiments of Rossi [174], the perturbed monopole does not relax to an axisymmetric state. Instead, as the vortex structure rotates, the positive parts of the perturbation are sheared and spiral arms are ejected which then mix, whereas

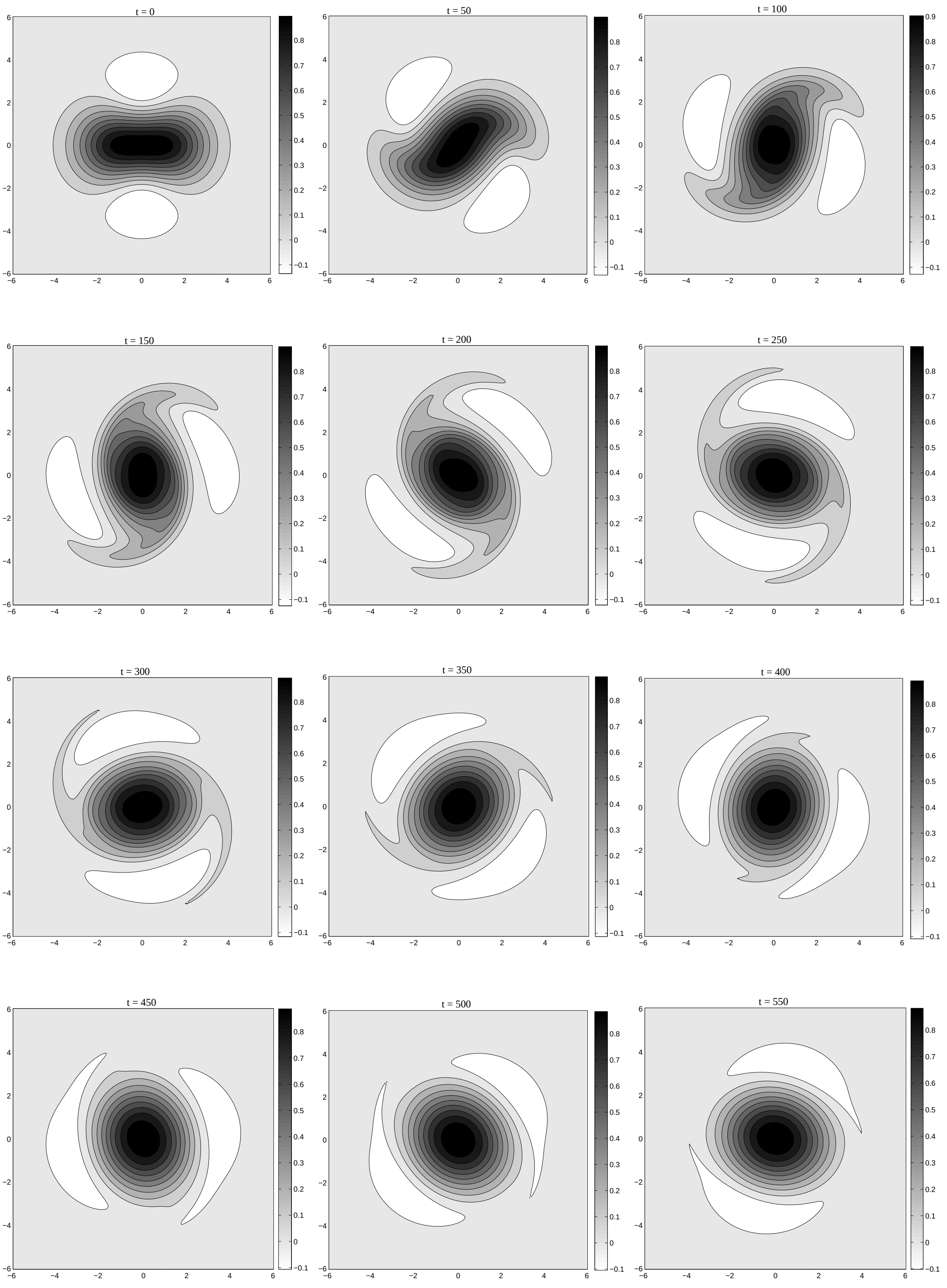


Figure 6.2: Perturbed monopole relaxing to a tripole attractor. Reynolds number is 10^4, strength of the perturbation is $\delta = 0.25$. 10 equally spaced contours, vorticity normalized by $\omega_{\max}(0)$.

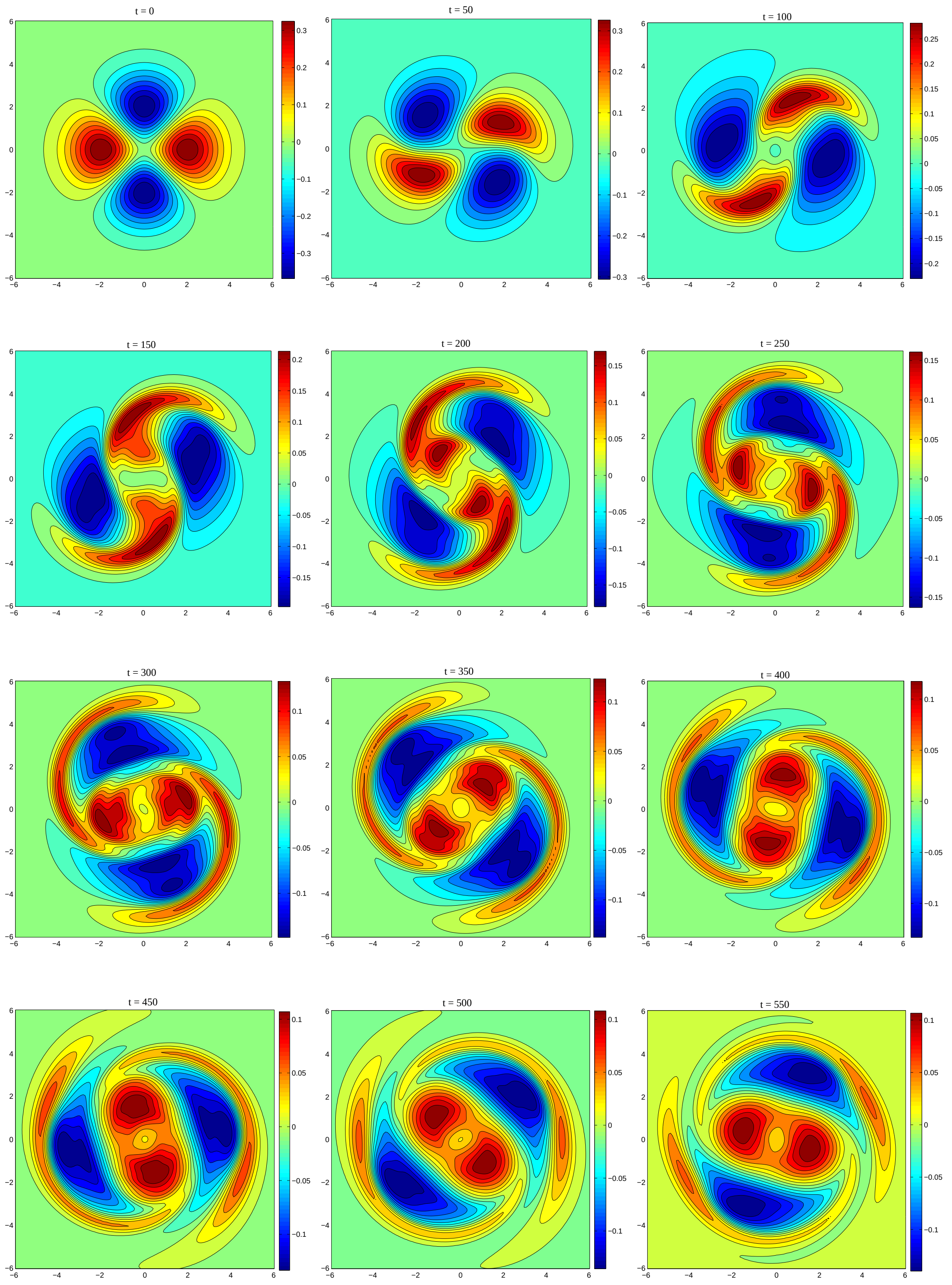


Figure 6.3: Perturbation vorticity, normalized by $\omega_{\max}(0)$. $Re = 10^4$, $\delta = 0.25$. 16 equally spaced contours.

the negative parts remain largely untouched by the shear-diffusion mixing. The vortical structure approaches a state where the positive vorticity is concentrated in a slightly elliptically deformed core, while two satellites of negative vorticity remain: a rotating tripole. This structure then decays due to viscosity on the Re timescale.

The evolution of the perturbation vorticity, shown in Figure 6.3 for the $Re = 10^4$ case, reveals more clearly how the positive part of the perturbation is quickly sheared, producing spiral arms as it also merges towards the core of the vortex. The negative perturbation, on the other hand, remains concentrated in two satellites. In addition, as the overall perturbation amplitude slowly decays, the relative amplitude of negative and positive parts remains comparable. As the structure reaches a quasi-steady state, the positive spiral arms mix and diffuse, leaving most of the positive perturbation vorticity concentrated towards the core.

Details of Numerical Experiments. For the calculation shown above (and others to be presented subsequently, unless otherwise noted), the initial vorticity was discretized by arranging vortex particles in a triangular lattice, covering an initial box of dimensions $[-6, 6] \times [-6, 6]$. Particles outside a circle of radius equal to 6 were then deleted, and the vorticity of the remaining particles was assigned according to the initial condition evaluated at their locations $\mathbf{x}_i$. The circulation strengths of the particles were then found by solving the RBF interpolation problem with Gaussian basis functions —the blob function given by (1.3)— using the GMRES solution method, as described in §4.2. The initial inter-particle spacing for an equivalent resolution square lattice was $h = 0.18$ (stretched in the triangular lattice to $h = 0.19342$), and the overlap ratio was $h/\sigma = 0.9$, resulting in an initial core size $\sigma_o = 0.2$. Time-stepping was provided by a fourth-order Runge-Kutta scheme, with $\Delta t = 1.0$, and spatial adaption was performed every 10 time steps using, once again, the GMRES method to obtain the new particle strengths. For the GMRES method, the relevant parameters were a relative residual reduction factor of 10^{-12}, and a maximum number of iterations fixed at 400. The viscous method used, as in the rest of this thesis, was core spreading, resulting in a maximum core size of $\sigma_{\max} = 0.2049$ for the $Re = 10^4$ case. The number of particles at the end of 600 time steps was $N = 4600$. One should point out that this is not an exceptionally fine resolution calculation, as we were limited by the fact that it was performed with the first version of the codes, in serial mode (and on modest desktop machines). The initialization error in the vorticity was measured on a maximum norm to be 0.03% (maximum value on the whole domain).

Results. In Figure 6.4, the total vorticity field is presented for three time slices of the calculation performed with the presently developed method, side-by-side to results obtained by Rossi with his splitting vortex method, and published in [174], Fig. 1, p. 2330 (reproduced here with kind permission from the first author). This is the higher Reynolds number case, with $Re = 10^4$ and $\delta = 0.25$. One

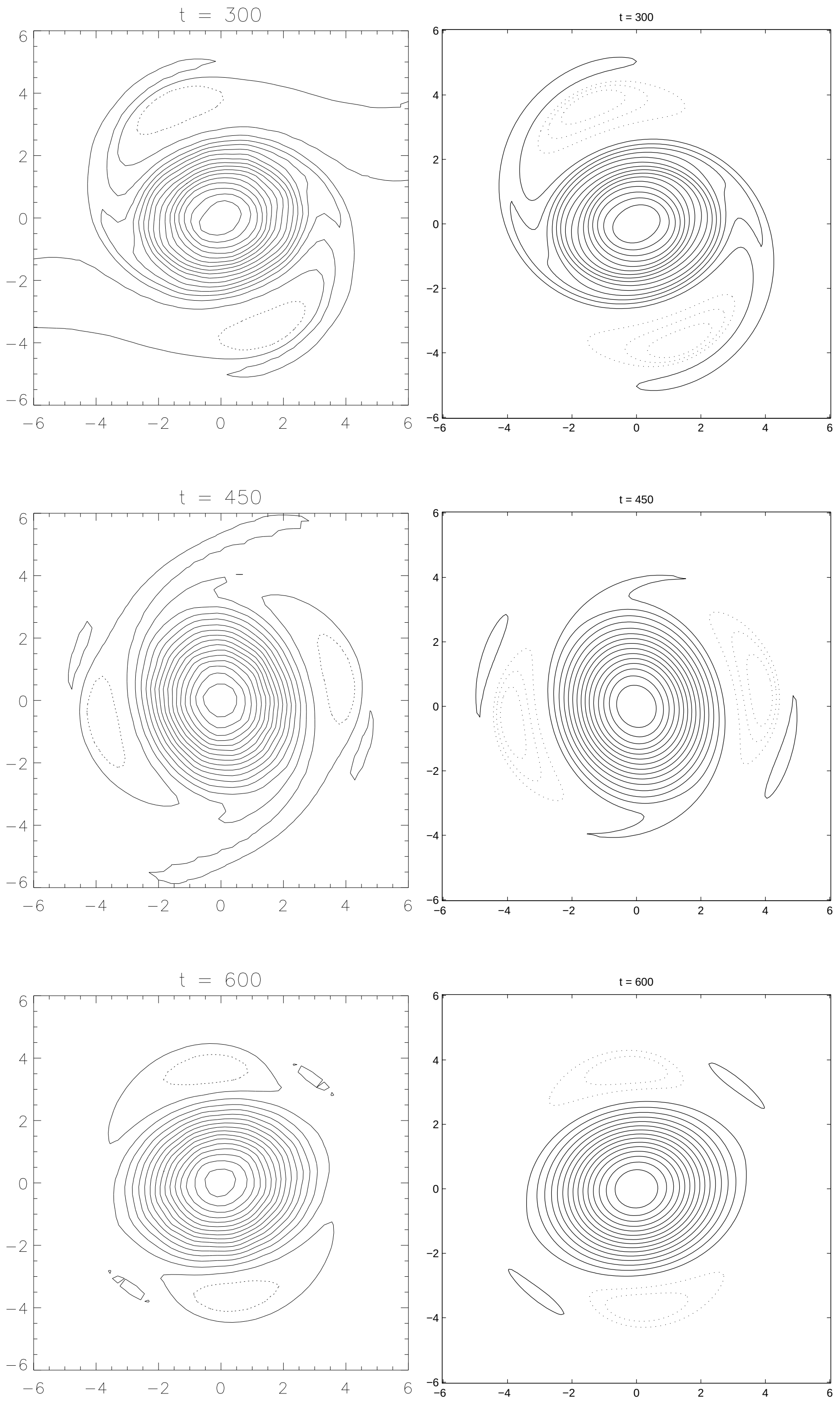


Figure 6.4: Perturbed monopole relaxing to a tripole attractor, total vorticity field. $Re = 10^4$, $\delta = 0.25$. Left: Rossi's splitting vortex method, from [174] (reproduced with permission). Right: present method; 15 equally spaced positive contours, and 3 equally spaced negative contours (dotted).

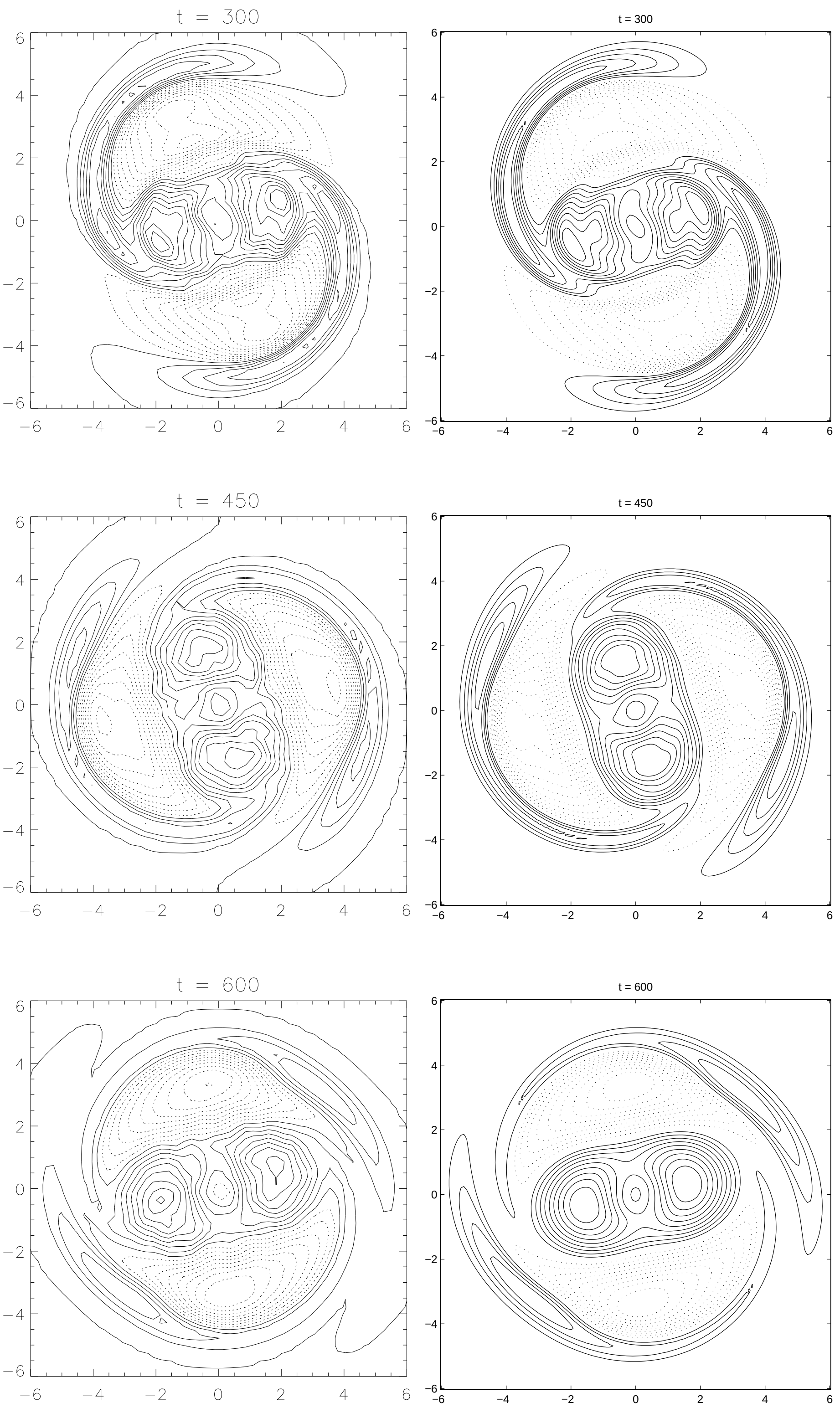


Figure 6.5: Perturbation vorticity; $Re = 10^4$, $\delta = 0.25$. Left: Rossi's splitting vortex method, from [174] (reproduced with permission). Right: present method; 20 equally spaced contours (dotted for negative values).

can see that the present method is able to reproduce very well the results of [174], in terms of the rotation angle of the structure and shape of contours, but exhibiting much smoother contours with sharper fine details. Unfortunately, there are no numerical parameters reported in [174], but the first author informally stated to us that the number of vortex elements in the computations were in the "low to middle 10's of thousands" (L. F. Rossi, private communication 2004). Their calculations were performed quite some years ago, and indeed they required a mainframe computer (Cray C90). The present results were obtained on a common desktop machine, and since the maximum number of particles was $N = 4600$, one can reasonably conclude that Rossi used more than double the number of particles for the calculations in [174]. It is proposed that the splitting scheme may be a source of noise and numerical diffusion in the calculations of [174]; this is avoided in the present method by the control of core sizes in the RBF interpolation during each spatial adaption process, without the need for vortex splitting.

Figure 6.5 shows the perturbation vorticity for the same calculation and at the same times as the total vorticity contour plots of Figure 6.4. Once again, the agreement with [174] is satisfactory, and the results obtained with the present method are considerably smoother.

In Figure 6.6 are shown three plots of the vorticity perturbation at a time $t = 500$, for different Reynolds numbers. Once again, we reproduce the plots published in [174], corresponding to Fig. 8 p. 2337 (by permission) and place them side-by-side with the results obtained with the present method. It can be seen that once again the present results are much smoother, but more importantly, the general agreement applies only to the two higher values of Re. For $Re = 1000$ we obtain significant differences with the results of [174], where it was observed in this case that the tripole structure is significantly eroded at $t = 500$ (apart from the fact that in making this plot, Rossi seems to have inadvertently switched the positive and negative contour line styles). Our result shows that the perturbation is indeed being decreased in magnitude due to diffusion (as well as the base vortex), but the positive center and the negative inclusions continue to have comparable strength, and the tripole is still clearly visible until the end of the computation. This is even more clear in the time-slices of vorticity perturbation presented in Figure 6.7. Indeed, we performed a calculation at $Re = 500$ and the same observation holds: although the perturbation amplitude has decayed, the relative strength of the positive center and negative inclusions is still close to 1. The perturbation vorticity for the case $Re = 500$ is shown in Figure 6.8.

Discussion. It is our view that the differences observed with the results of [174] for the lower Re case reflect that in the latter case there is considerable numerical diffusion introduced by the vortex particle splitting scheme. In the more viscous calculations, an imposed value for the maximum core size results in particle splitting being performed much more often. Hence, splitting errors accumulate for the low Re case. The implication of our calculations, where the spatial adaption effects core

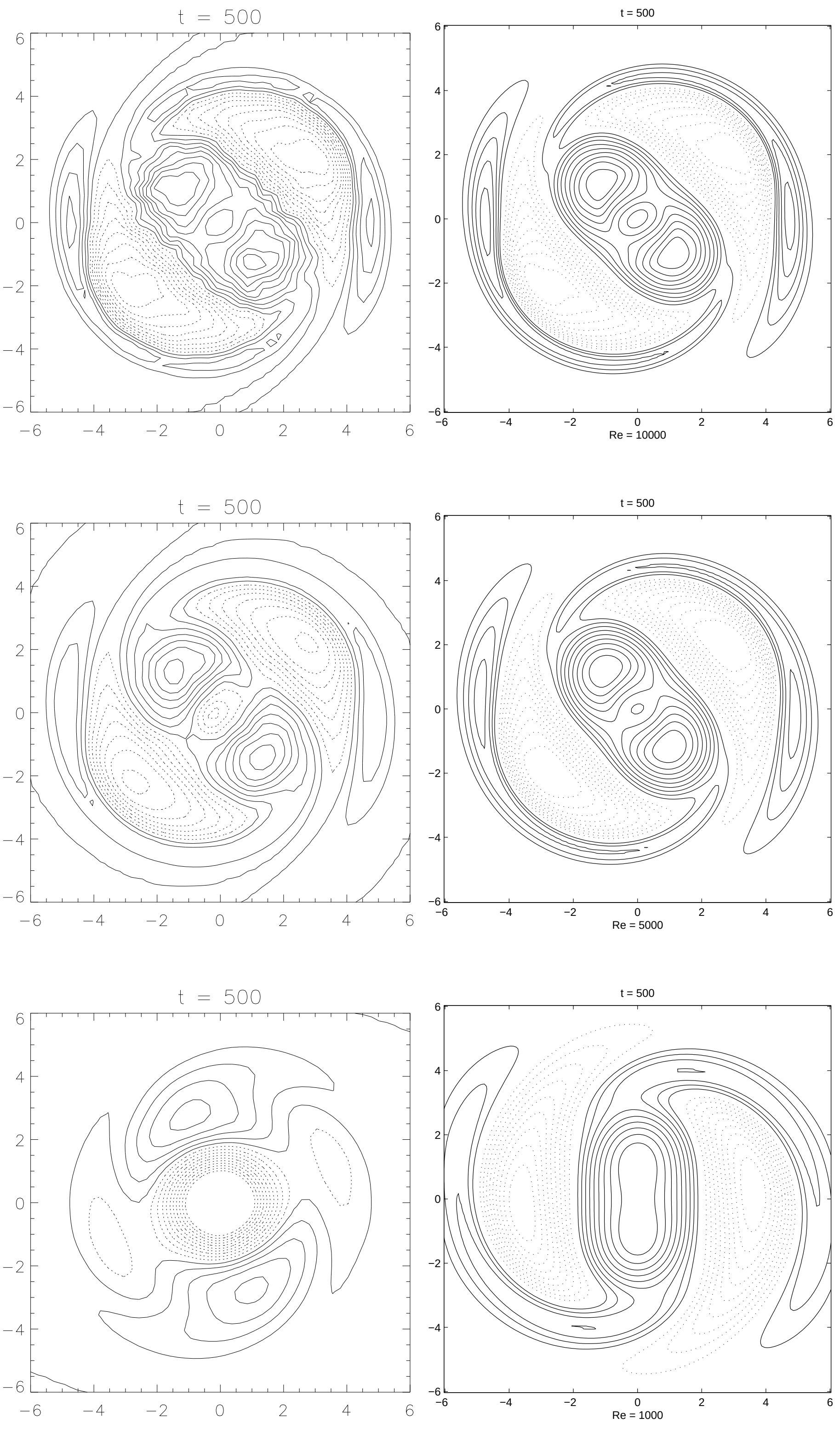


Figure 6.6: Perturbation vorticity at $t = 500$; from top to bottom, $Re = 10^4$, 5000, 10^3. Left: Rossi's splitting vortex method, from [174] (reproduced by permission). Right: present method; maximum blob core radius, respectively, 0.205, 0.21, 0.245; 20 equally spaced contours (dotted are negative).

size control automatically without splitting, is that the tripole structure is more persistent than previously thought, and is not eroded at the low $Re = 10^3$, nor even at $Re = 500$.

Since there is not an analytic expression for a vortex tripole, as pointed out in [174], there are difficulties in measuring the decay of the vortical structure relative to its quasi-steady state. For example, when measuring the relaxation to a known axisymmetric state, the quantity of non-axisymmetric enstrophy is useful [26]. In [174], the decay of the tripole structure is illustrated by plotting the ratio of maximum vorticity to minimum vorticity (absolute value), with respect to the viscous timescale, t/Re. We present a similar result for our calculations, including the additional case of $Re = 500$, in Figure 6.9(a), where the axes have the same limits as those of the plot included in [174], Fig. 9, p. 2337, for the purposes of comparison. Careful reading of the plots reveals that the case $Re = 10^3$ is decaying more rapidly in Rossi's calculation, consistent with our previous observations. In Figure 6.9(b), the same plot is presented with adjusted axes, to allow for the results for $Re = 500$ to be fully visible. Figure 6.9 seems to support the conclusion that the Gaussian monopole is the asymptotic state, since the magnitude of the negative vorticity inclusions is vanishing. This decay occurs on the viscous timescale, however, whereas the relaxation towards the quasi-steady state (*i.e.*, the rotating tripole, for large enough perturbation amplitudes) occurs on the much faster shear-diffusion timescale, $\sim Re^{1/3}$. The eventual decay of the vortical structure is more clearly revealed when the plot of Figure 6.9 is re-drawn using logarithmic axes; this is shown on Figure 6.10. Another possible way to look at the decay behavior of the flow is shown in Figure 6.11, which plots the amplitude of positive and negative perturbation vorticity with respect to time, for the different Reynolds number calculations. It can be observed that the amplitude of the positive and negative perturbations decay very slowly after an initial stage of re-organization. Also, the relative magnitude of the negative and positive perturbations remains comparable, which would support the observation that although the perturbation decays, it does not lose structure during this decay.

6.2 Co-rotating Vortices Before Merging

Description. The interaction of two co-rotating vortices is characterized by the fundamental process of vortex merging. For inviscid vortex patches, the interaction is governed by the "merger criterion", which stipulates the critical distance between the vortices under which merging will occur [180]. Above this distance, the vortices exhibit a deformed shape, until when they are sufficiently far apart they maintain their approximately circular shape. In the case of viscous flow, the sizes of the vortices grow with time as they diffuse, so that eventually they will reach the critical distance, and merge [131]. At higher Reynolds numbers, the time necessary to reach the critical distance will be larger, and thus it is possible to observe the early interaction of the vortices in some detail.

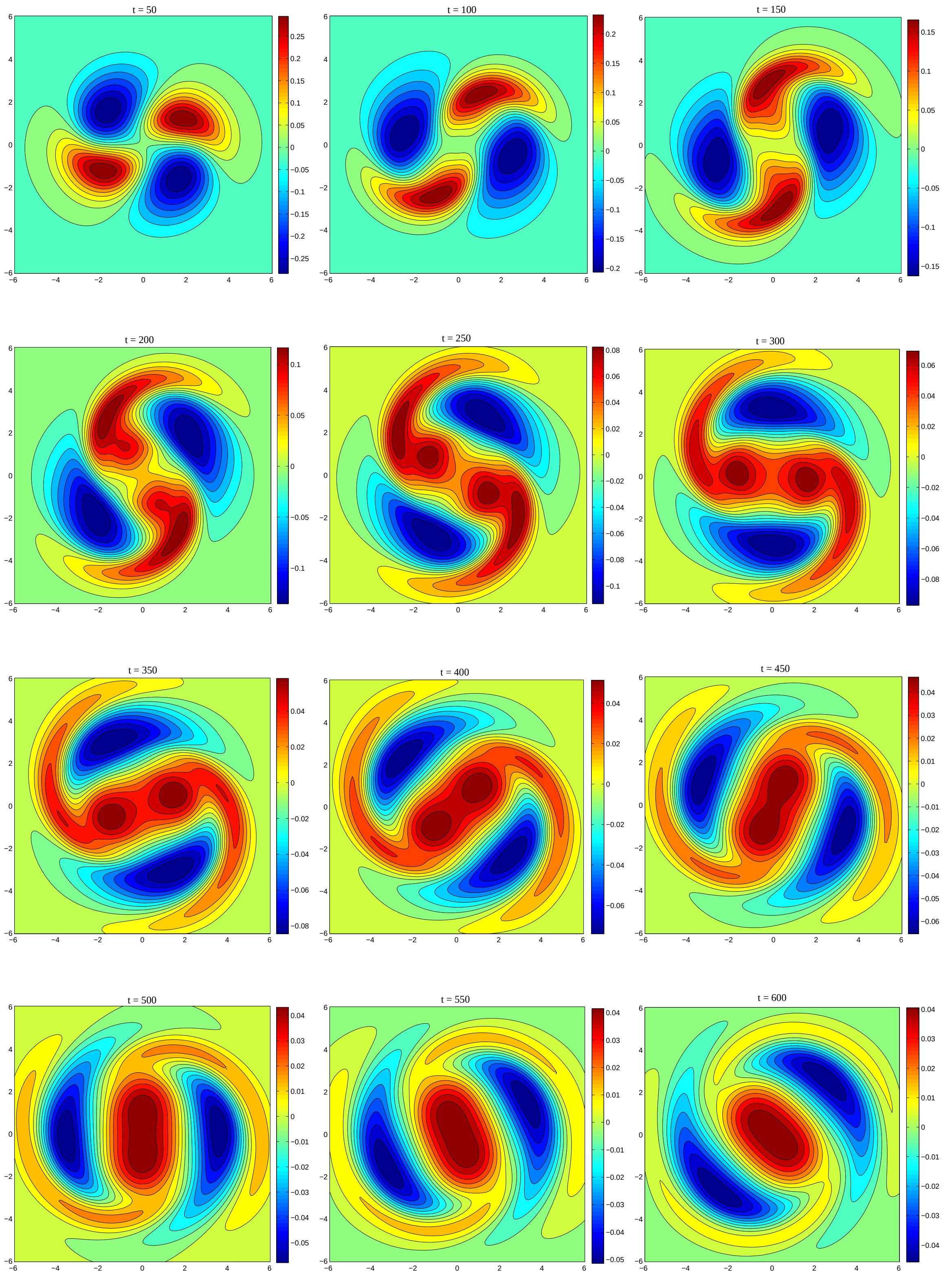


Figure 6.7: Perturbation vorticity, normalized by $\omega_{\max}(0)$. $Re = 10^3$, $\delta = 0.25$. 16 equally spaced contours.

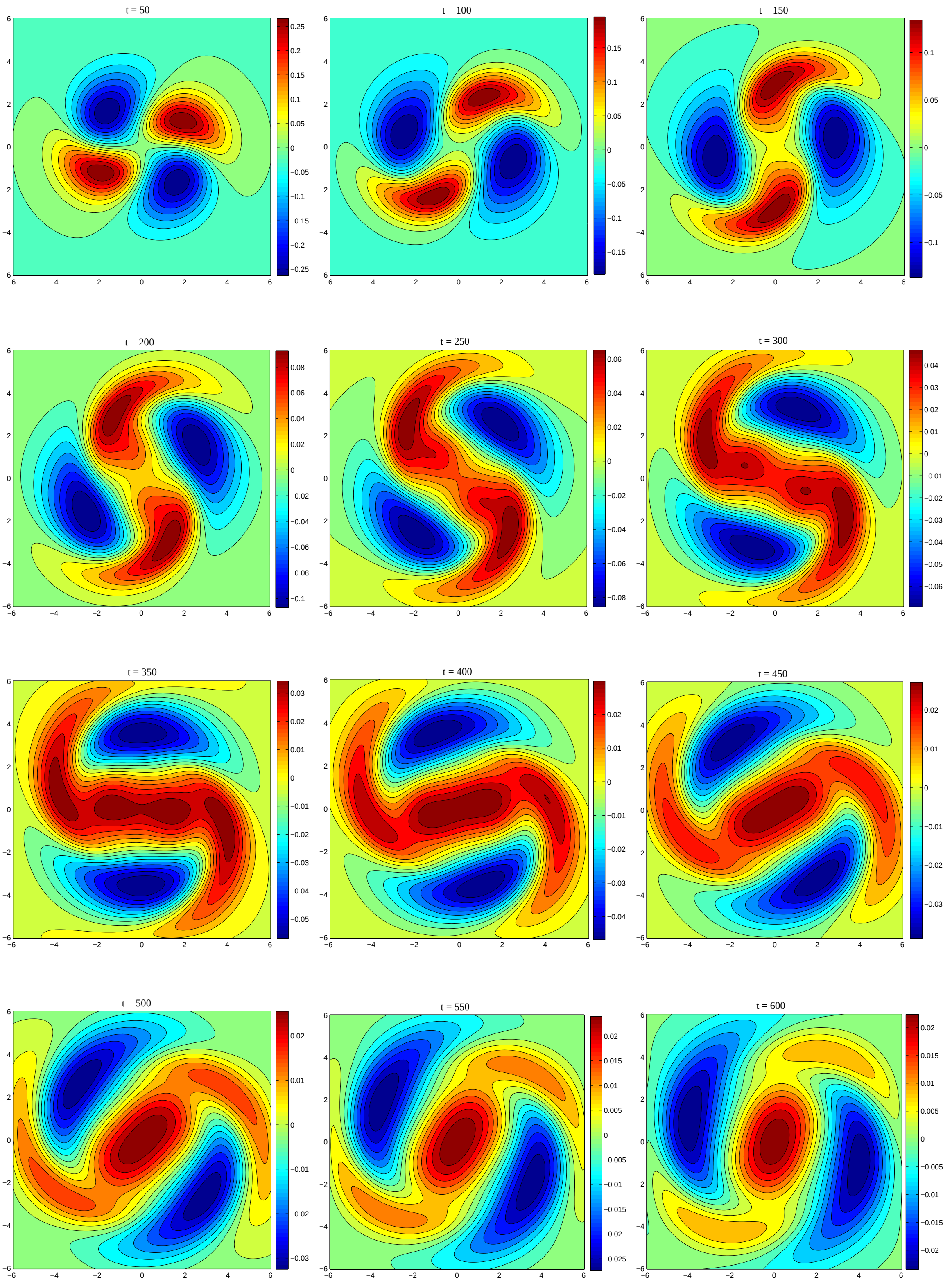
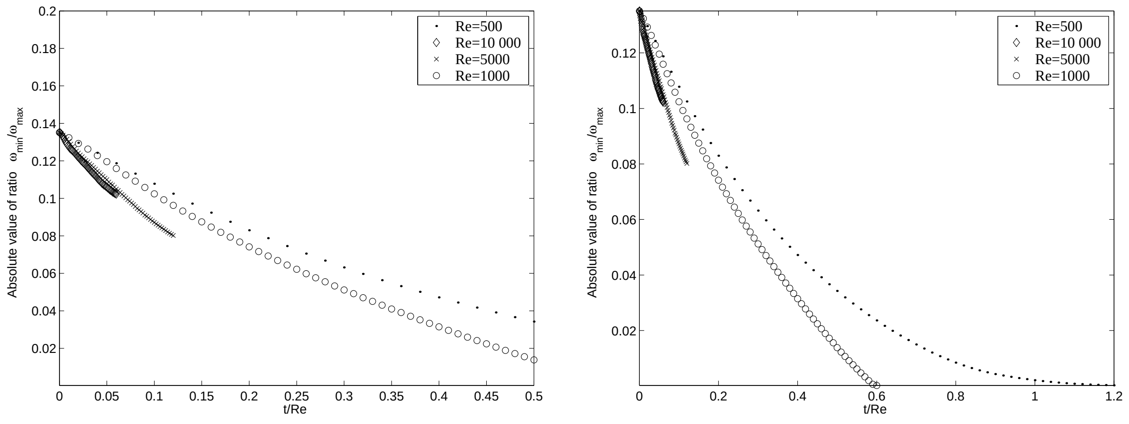


Figure 6.8: Perturbation vorticity, normalized by $\omega_{\max}(0)$. $Re = 500$, $\delta = 0.25$. 16 equally spaced contours.



(a) (b)

Figure 6.9: Variation on the viscous timescale of the ratio $|\omega_{\min}/\omega_{\max}|$, for the perturbed vortex with $\delta = 0.25$, for varying Reynolds number.

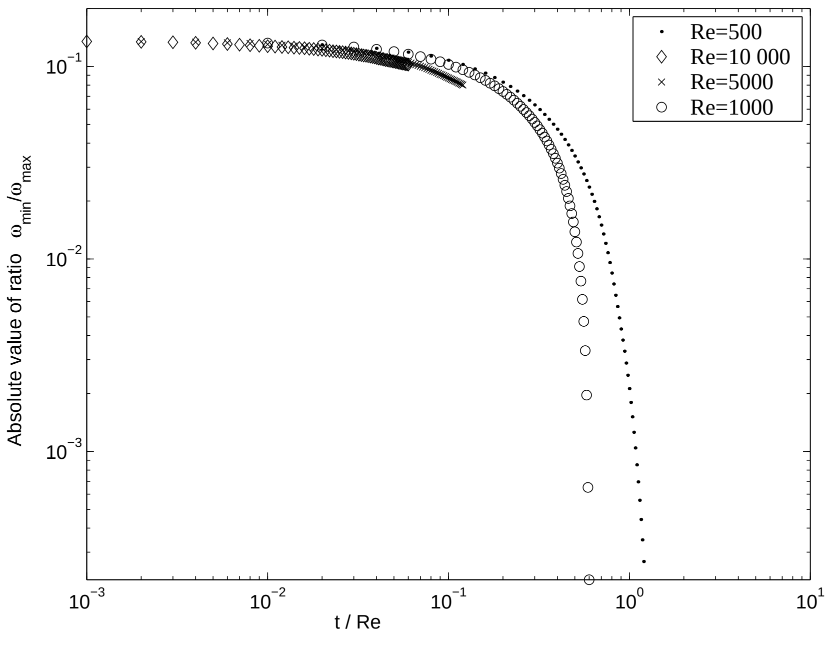


Figure 6.10: Variation on the viscous timescale of the ratio $|\omega_{\min}/\omega_{\max}|$, for the perturbed vortex with $\delta = 0.25$, for varying Reynolds number.

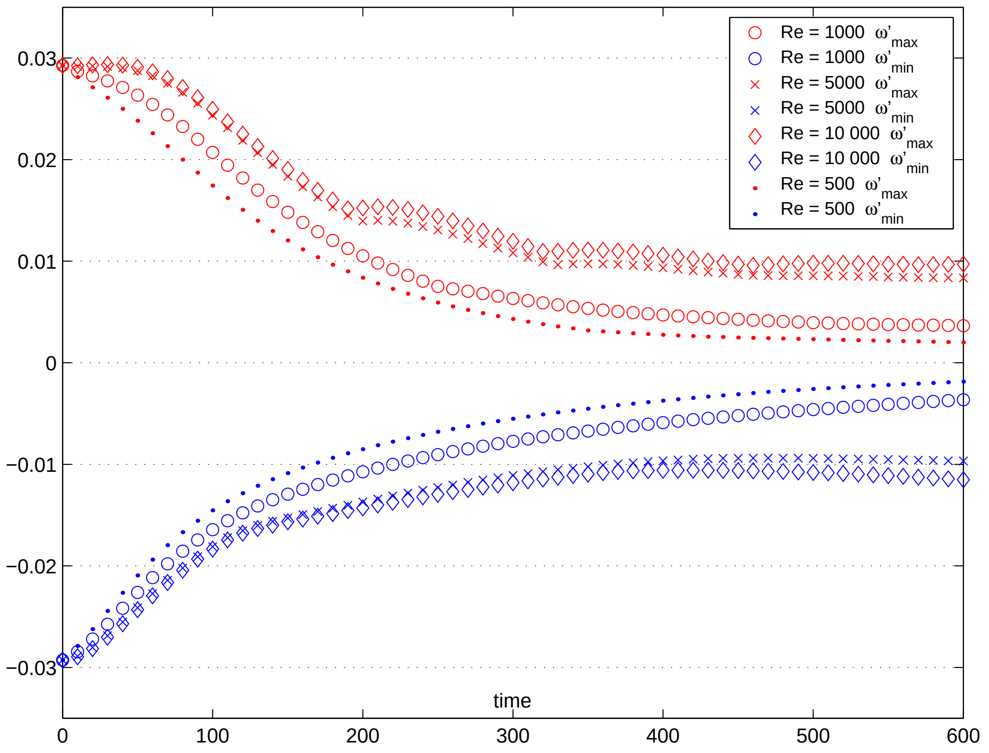


Figure 6.11: Perturbation vorticity amplitudes, ω'_{min} and ω'_{max}, for the perturbed vortex with $\delta = 0.25$, *versus* time, for varying Reynolds number.

In the merging process of viscous vortices, different stages have been distinguished [131]. The first is an *adaptation* stage, when each vortex is elliptically deformed due to the strain field produced by the other. This is an important interaction due to the fact that in three dimensions the elliptically deformed vortices are subject to the so-called "elliptic instability" [10, 110, 95]. In the second stage, the vortices are in a *metastable state* and evolve on a viscous time scale. Their distance remains relatively constant, and they rotate around each other at a frequency very close to that of two point vortices of same circulation. In the third stage, the vortices reach their critical state, and finally, they rapidly *merge* on the convective time scale.

Previous Results. The adaptation stage of two viscous co-rotating vortices was studied numerically using spectral methods in [111]. This paper includes results in terms of vorticity contours during the early evolution of the vortices at $Re = 8 \times 10^4$, showing logarithmic contour levels down to a value of 10^{-6}. These results illustrate very well the deformation suffered by each vortex core under the effect of the strain produced by the other vortex. In addition to the elliptical deformation of the cores, deformations of the weak contours of vorticity show small-scale "wiggles", as a result of the differential rotation, and also indicate vorticity gradient intensification.

Parallel Implementation of the Vortex Method. We considered this to be a very exacting test for any numerical method, and set about to reproduce these results with the present vortex method. To this end, the method was implemented in parallel, using the PETSc library in a C++/MPI code. The PETSc library [6] provided the preconditioners and GMRES solution method, used both at initialization and during each RBF spatial adaption process, as well as vector and matrix assembly and manipulation in parallel (more details are given in Appendix A). With this new implementation of the method, a high-accuracy calculation was performed, with up to 2×10^4 vortex particles. The results, shown in Figure 6.12, reproduce remarkably well those presented on [111], where a 1024×1024 computational mesh was used (thus, the problem size is two orders of magnitude larger than in our calculation). Thanks to the courtesy of Dr. S. Le Dizès (and with the permission of Cambridge University Press), we reproduce the results presented in [111] for the calculation at $Re = 8 \times 10^4$ in Figure 6.13, for the purposes of comparison with our result in Figure 6.12.

Details of Calculations and Results. The initial condition for the calculation that produced the results in Figure 6.12 is given by placing two Gaussian vortices, each of total circulation $\Gamma = \pi$ and radius $a_o = 1.0$, a distance of $10a_o = b_o$ apart. For a given total circulation of each vortex (and vorticity distribution, in this case, Gaussian), the evolution of the two-vortex system is dictated by two parameters, the ratio of the radius to the separation distance, a_o/b_o, and the Reynolds number, defined as $Re = \Gamma/\nu$. For the case in Figure 6.12 (as well as Figure 6.13) $a_o/b_o = 0.1$ and $Re = 8 \times 10^4$. The initial vorticity was discretized by placing vortex particles on a triangular lattice covering an initial domain of size $[-b_o, b_o] \times [-\frac{b_o}{2}, \frac{b_o}{2}]$, with an inter-particle spacing on the equal-resolution square lattice given by $h_{sq} = 0.075$ and an overlap ratio $h/\sigma = 0.8$, resulting in $\sigma_o = 0.09375$. The co-rotating vortices have their centers on $(-\frac{b_o}{2}, 0)$ and $(\frac{b_o}{2}, 0)$. Thus, the initial vorticity at any location is given by

$$\omega(\mathbf{x}, 0) = \exp\left(-\frac{(x + \frac{b_o}{2})^2 + y^2}{a_o^2}\right) + \exp\left(-\frac{(x - \frac{b_o}{2})^2 + y^2}{a_o^2}\right). \tag{6.2}$$

Note that the calculations presented in [111] are performed in a domain of dimensions $2\pi \times 2\pi$, with the vortices placed at symmetric locations on the x-axis and at a distance $b_o = \pi/4$ from each other, such that the width of the computational domain is 8 times b_o. This larger computational box is needed for their calculations with spectral methods to avoid the effects of periodic images. In our spatial scaling, $a_o = 1.0$ and $b_o = 10$, so that the resolution used in [111] would be equivalent to $h = 8b_o/1024 = 0.078125$; this is very close to our chosen $h_{sq} = 0.075$, and thus one can say that the spatial resolutions are comparable. The vortex method results in a two-order of magnitude economy in problem size due to the fact that the computational elements are concentrated in the domain of interest, where the vorticity is non-zero. This is illustrated in Figure 6.14, where two snapshots are

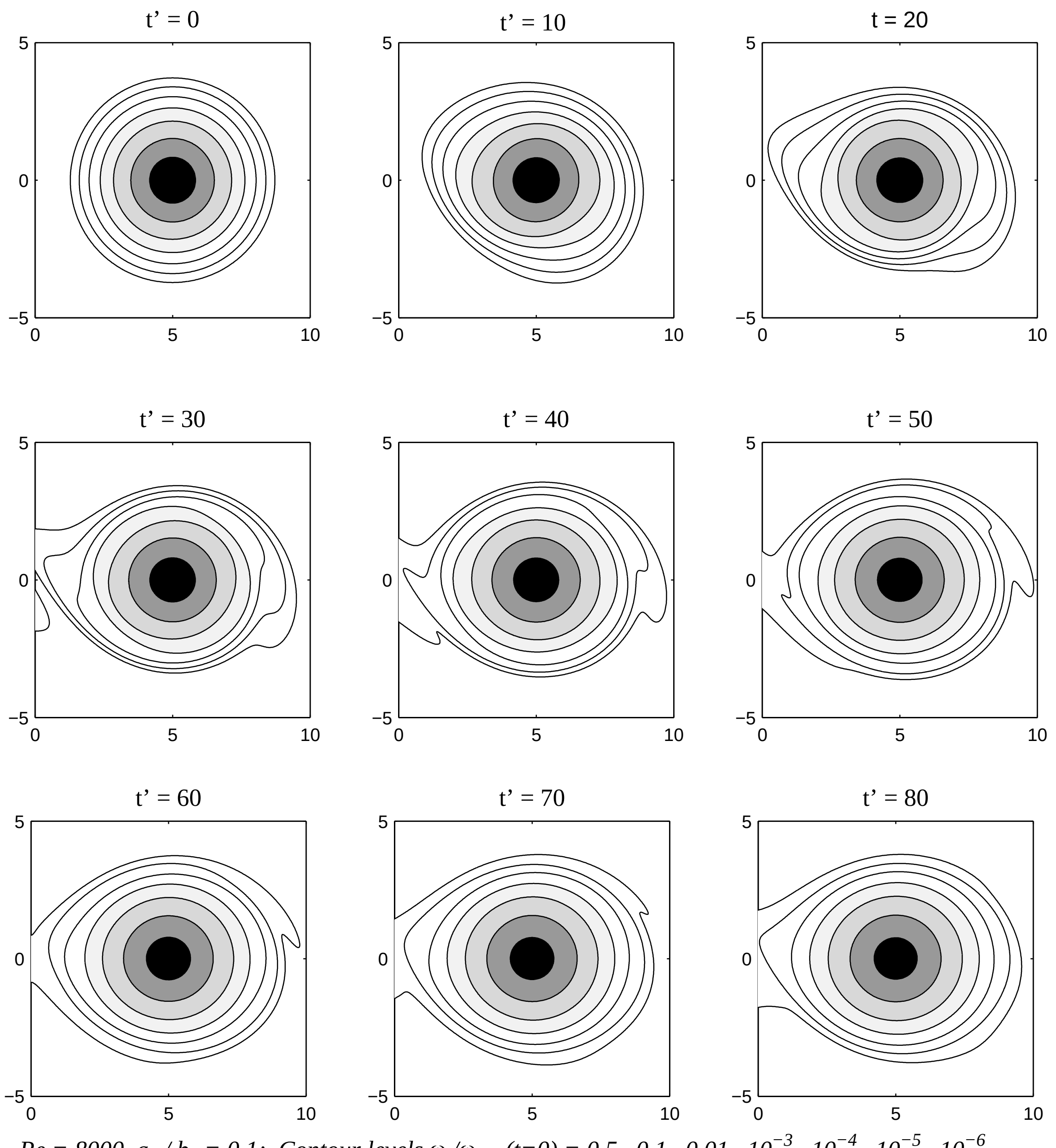


Figure 6.12: Vorticity contours of the right side of a symmetric two-vortex system (initially Gaussian) as they adapt to each other's strain field (on a frame rotating with the vortices); $t' = t\Gamma/(2\pi a_0^2)$, $Re = 8 \times 10^4$, $a_o/b_o = 0.1$.

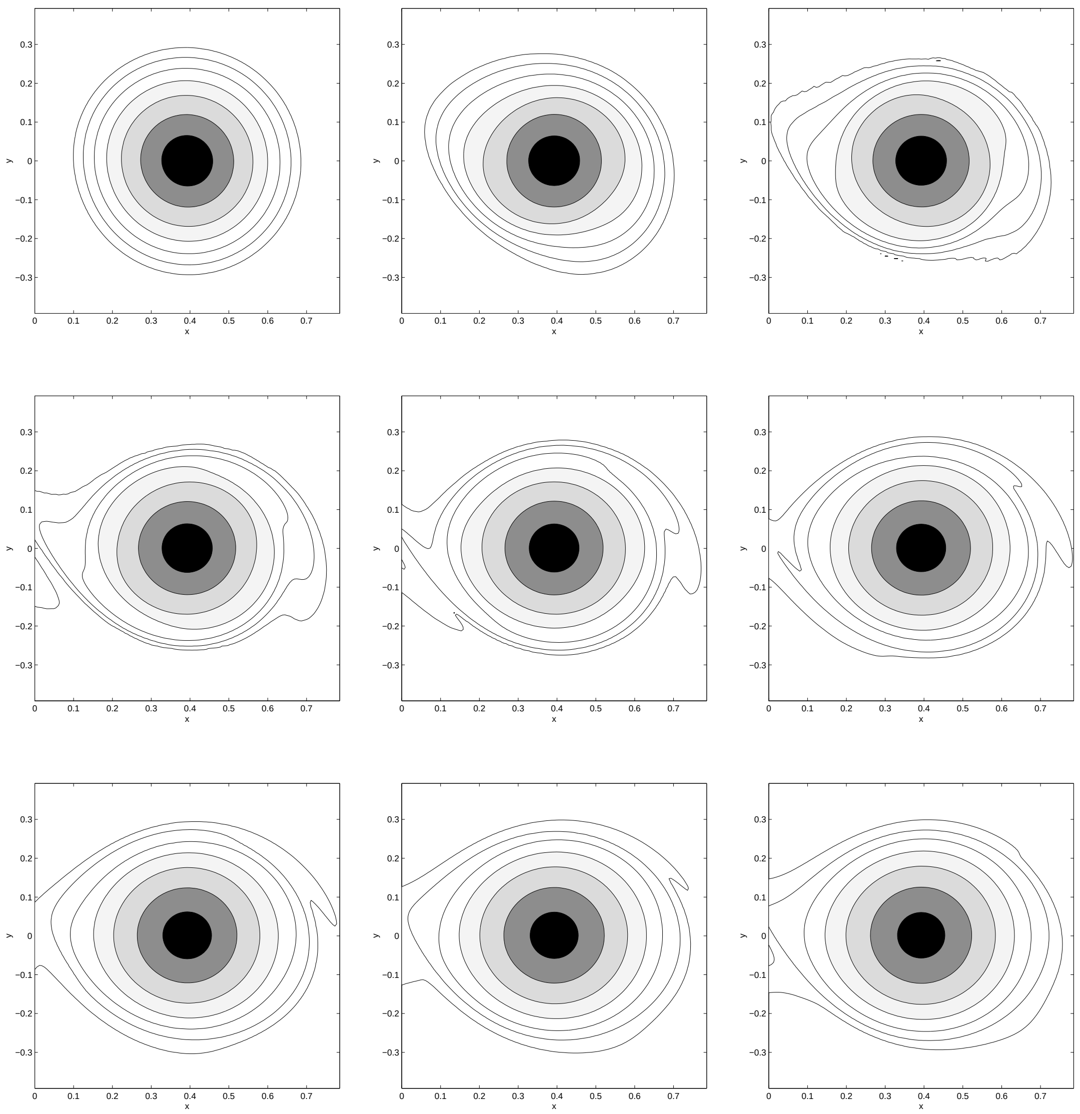


Figure 6.13: Vorticity contours during relaxation process for two co-rotating vortices with Gaussian profile; from right to left, top to bottom, $t' = t\Gamma/(2\pi a_0^2) = 0, 10, 20, 30, 40, 50, 60, 70, 80$. Corresponds to Figure 3 of [111], p. 395. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}. Reprinted with the permission of Cambridge University Press.

shown of the vorticity contours down to level 10^{-6} and the particle locations at that time.

An approximate value of circulation strength for the initial particles was obtained using the standard initialization formula, $\Gamma_i = \omega_i\, h^2$, after which population control was enforced by deleting all particles whose approximate circulation was less than 10^{-10}. In this way, the particle population was reduced from an initial value of 4×10^4 to $N = 19940$. The remaining particles were used in the RBF interpolation problem to solve for their strengths, where the approximate Γ_i's were used as initial guess to the GMRES solution method. The relevant parameters for the iterative solver were: a relative convergence tolerance (relative decrease in the residual norm) of 10^{-10}; an absolute convergence tolerance (relative decrease in the residual norm) of 10^{-20}; a maximum number of iterations set to the current number of particles, N. With this convergence tolerances, the initialization of the case shown in Figure 6.12, for example, converged in 216 iterations; the non-normalized 2-norm of the vector-difference between the initial guess and the solution was measured to be 0.00167441. The maximum number of iterations in the total of 800 spatial adaption processes (performed every 10 time steps) during the computation was 238.

Finally, time-stepping was provided using a fourth-order Runge-Kutta scheme with $\Delta t = 0.02$. Note that for a final time $T = 160$ (giving $t' = 80$) the number of time steps needed is 8000. Thus, this calculation is in every way quite demanding. Running on a Beowulf cluster on 20 processors —with each processor an Intel Pentium III 1 GHz, 4 x 256 Mb SDRAM 72 bit— it required 11 days and 18 hours, approximately (but note that direct summation of the Biot-Savart law was performed). The maximum core size due to core spreading, before the spatial adaption processes, is $\sigma_{\max} = 0.094584$, and the vortex particle population at the end of the run is $N(T) = 24'471$.

During the beginning stage of adaptation and the meta-stable state, the vortices rotate around each other at constant frequency given by

$$f_o = \frac{\Gamma}{\pi b_o^2}. \tag{6.3}$$

Thus, the time variable can be non-dimensionalized by the turn-over time, to give

$$t^* = \frac{t\Gamma}{2\pi^2 b_o^2}. \tag{6.4}$$

This time normalization is useful when studying the interaction of the vortices up to the onset of merging. In this time scale, the typical evolution of f and b is that both remain constant up to a critical time, when f rapidly increases while b rapidly decreases to zero, *i.e.*, the actual merging. For example, in [111] is shown a result for $Re = 8000$ and $a_o/b_o = 0.05$, where the critical time is $t^* \approx 5$. Note, however, that the non-dimensionalization of time in Figures 6.12 to 6.14 is with respect to the vortex radius a_o; this is the non-viscous time scale. For a calculation with $a_o/b_o = 0.05$, $t' = t^* \times 400$,

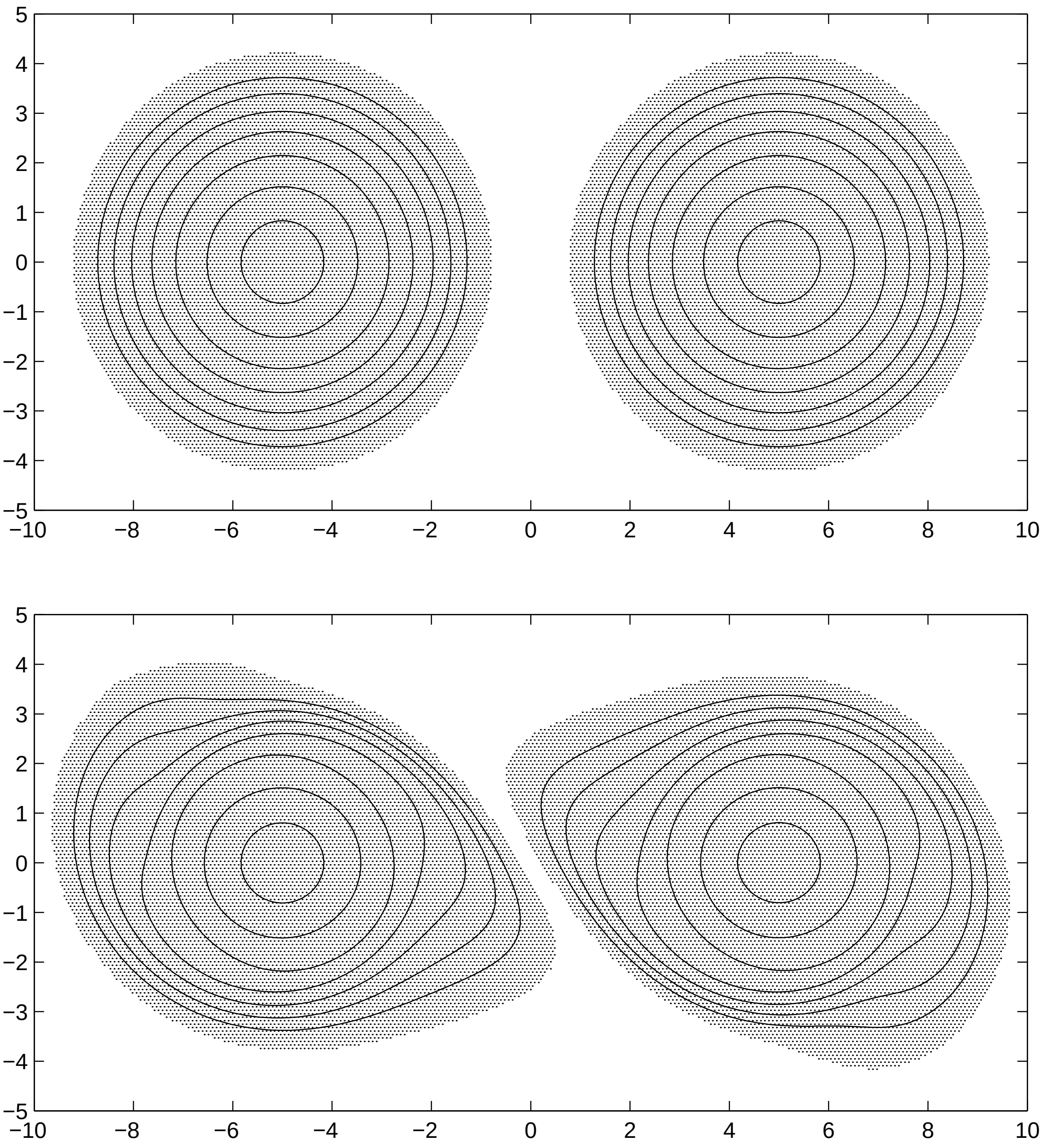


Figure 6.14: Vorticity contours of the symmetric two-vortex system at $Re = 8\times10^4$ and $a_o/b_o = 0.1$, for $t' = t\Gamma/(2\pi a_0^2) = 0$ (top) and 20 (bottom), including locations of the vortex particles at that time. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}.

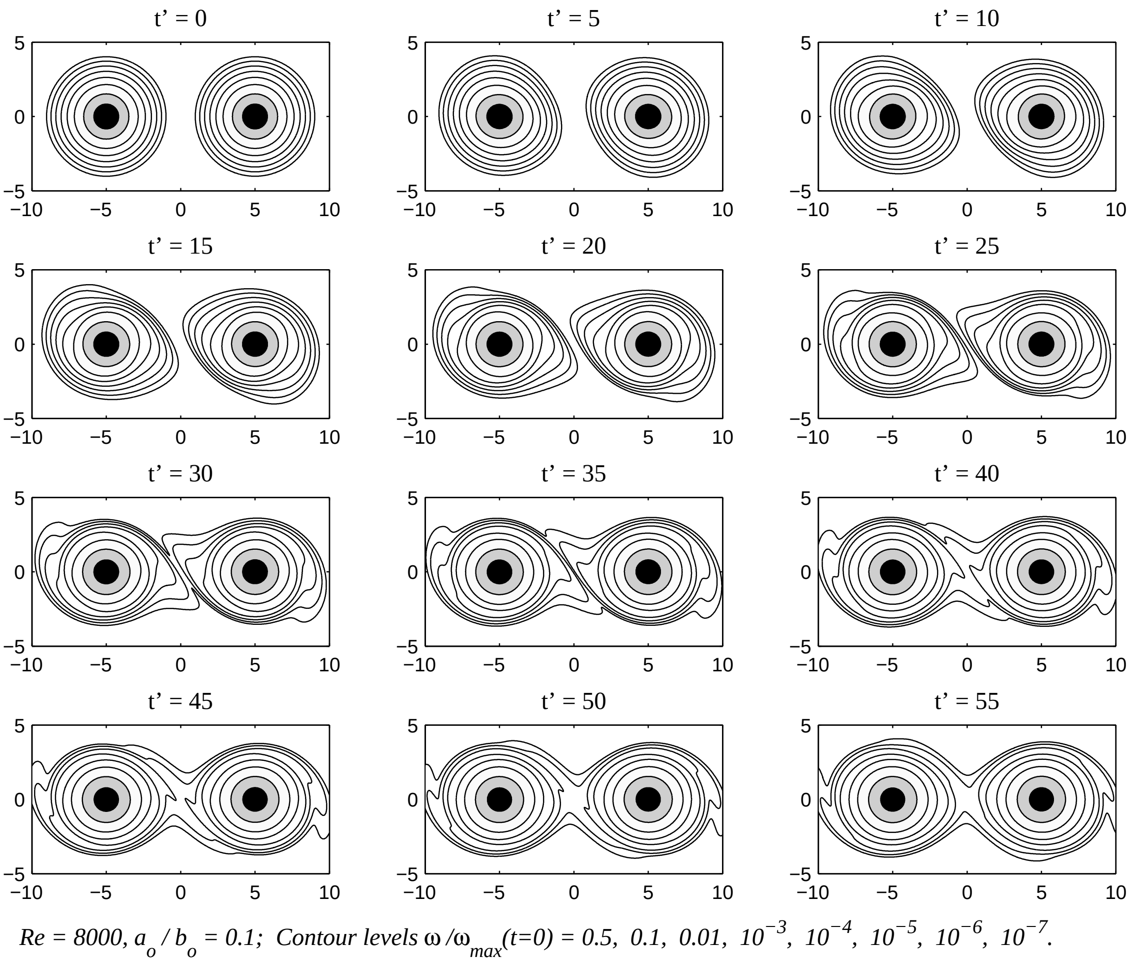


Figure 6.15: Vorticity contours of the symmetric two-vortex system, same case as Figure 6.12, but including one more logarithmic contour level.

so that with $\Delta t = 0.02$ more than 6×10^5 time steps would be needed to observe the merging process!

For observation of greater details of the early interaction of the co-rotating vortices, Figure 6.15 presents more frequent snapshots of the vorticity than those shown in Figure 6.12, including both vortices this time, and down to a further logarithmic contour of 10^{-7}.

Next, the calculation previously described was repeated with a different resolution; first a coarser value of $h_{sq} = 0.15$ is used, where every other numerical parameter has been kept the same. This results in an initial core size of $\sigma_o = 0.1875$ and initial number of particles of $N = 5382$ after population control. The final number of particles was $N(T) = 7583$. As shown in Figure 6.16, the coarser resolution results in the fine scales being smoothed and some of the details being lost. Figure 6.17 shows a direct comparison of the vorticity at a given time, as obtained by both resolutions. This allows one to see that the coarser resolution calculation also loses some of the 'tightness' of the outer contours, in the areas of vorticity gradient intensification.

A more resolved calculation was attempted as well, with a value of $h_{sq} = 0.055$ (recall that this

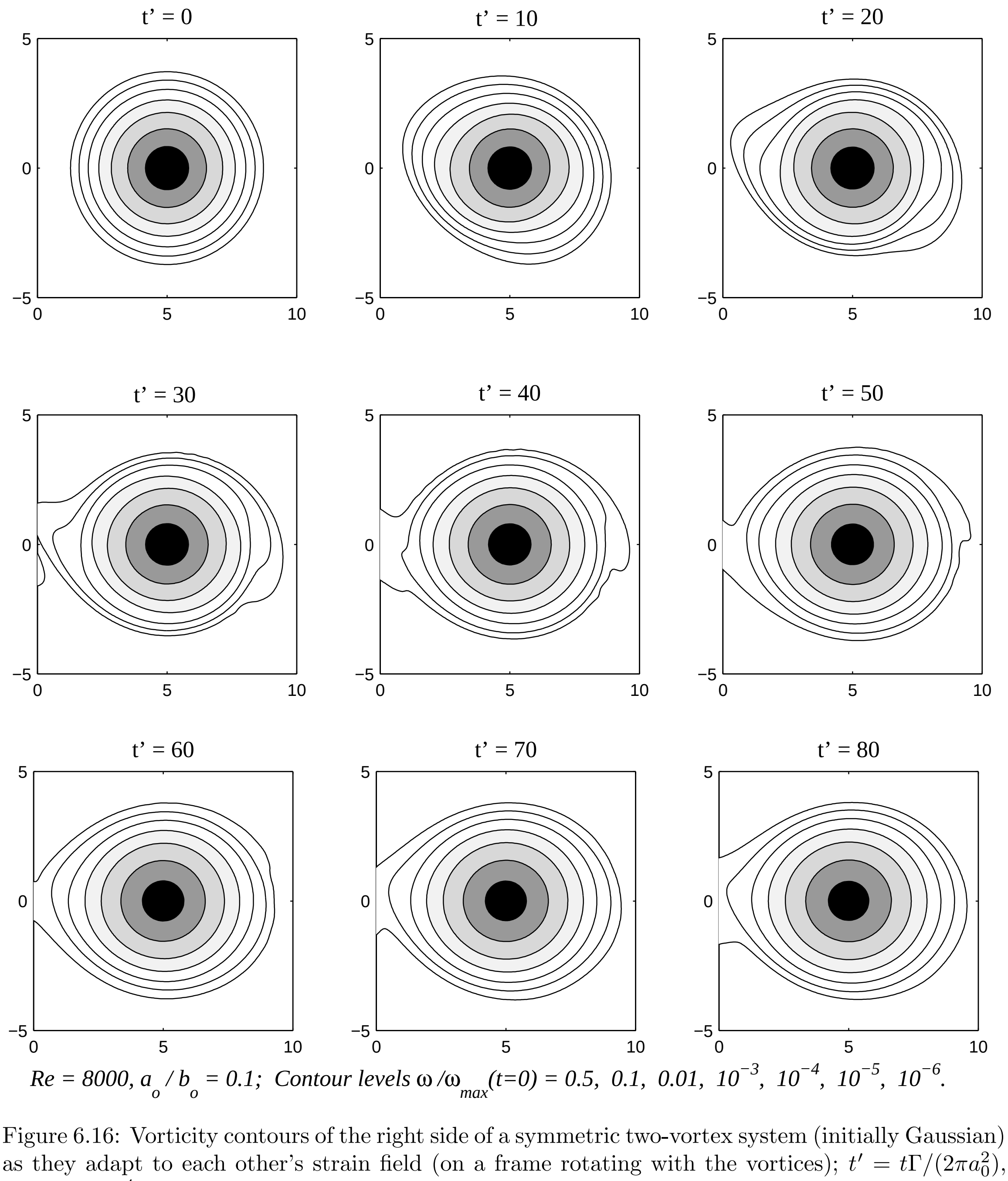


Figure 6.16: Vorticity contours of the right side of a symmetric two-vortex system (initially Gaussian) as they adapt to each other's strain field (on a frame rotating with the vortices); $t' = t\Gamma/(2\pi a_0^2)$, $Re = 8 \times 10^4$, $a_o/b_o = 0.1$. Coarser resolution with $h = 0.15$.

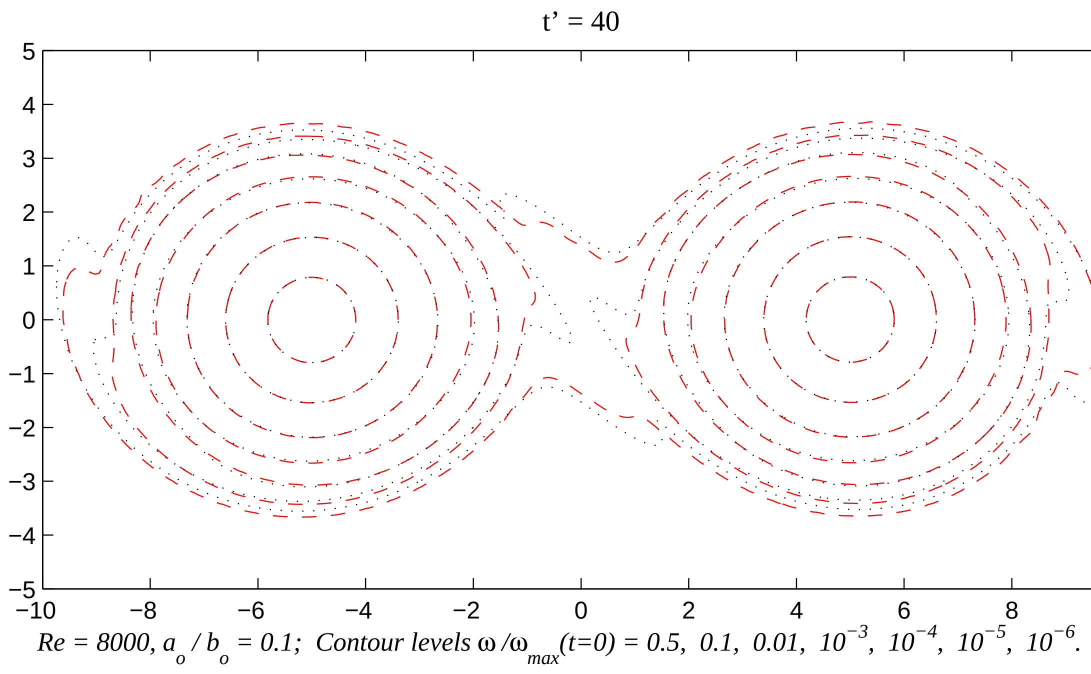


Figure 6.17: Vorticity contours of the symmetric two-vortex system for $t' = t\Gamma/(2\pi a_0^2) = 40$. Coarser resolution with $h = 0.15$ in red dashes, and fine resolution with $h = 0.075$ in black dots.

is the value of inter-particle spacing on an equivalent resolution square lattice). Unfortunately, with an initial number of vortex particles of almost 40'000, this computation could only be carried for a short time with the computing resources available at the time. With the present implementation of the vortex method, these calculations are limited by memory requirements, due to the lack of a matrix-free implementation. Using 44 processors, this highly-resolved calculations was carried on for a week, and the final time was $t' = 22$. A plot of the vorticity contours down to a level of 10^{-7} is shown in Figure 6.18 at $t' = 20$ for the two calculations using $h_{sq} = 0.055$ and 0.075, and a zoomed-in image is shown in Figure 6.18. It can be seen that the difference between both calculations is extremely small, being barely noticeably at vorticity levels of between 10^{-5} and 10^{-7}. Nevertheless, the same observation as previously made with the coarse calculation is valid, namely, that the fine-scale wiggles are more pronounced in the finer resolution, and the vorticity contours are "tighter" on the weak levels, where there is vorticity gradient intensification.

Concluding Remarks. These experiments have demonstrated a high-accuracy calculation of viscous vortex interaction, using a fully mesh-less method. The comparison with previously published results using spectral methods is thoroughly encouraging. Although a comparison cannot be made in wholly quantitative terms, it has been made with the help of the visualization of the vorticity to levels as weak as 10^{-6}, and the agreement is excellent. The potential of the vortex method of producing these results with a problem size two orders of magnitude smaller is quite auspicious.

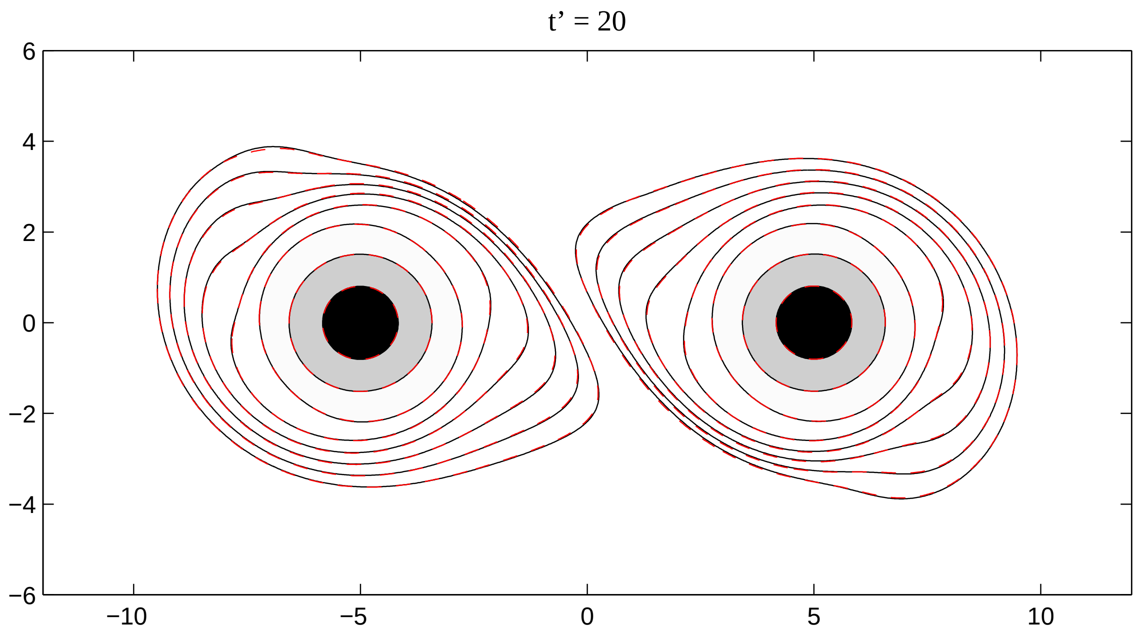


Figure 6.18: Vorticity contours of the symmetric two-vortex system for $t' = t\Gamma/(2\pi a_0^2) = 20$; as before, $Re = 8 \times 10^4$, $a_o/b_o = 0.1$. Very fine resolution with $h = 0.055$ in black; red dashes correspond to the previous case using $h = 0.075$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}.

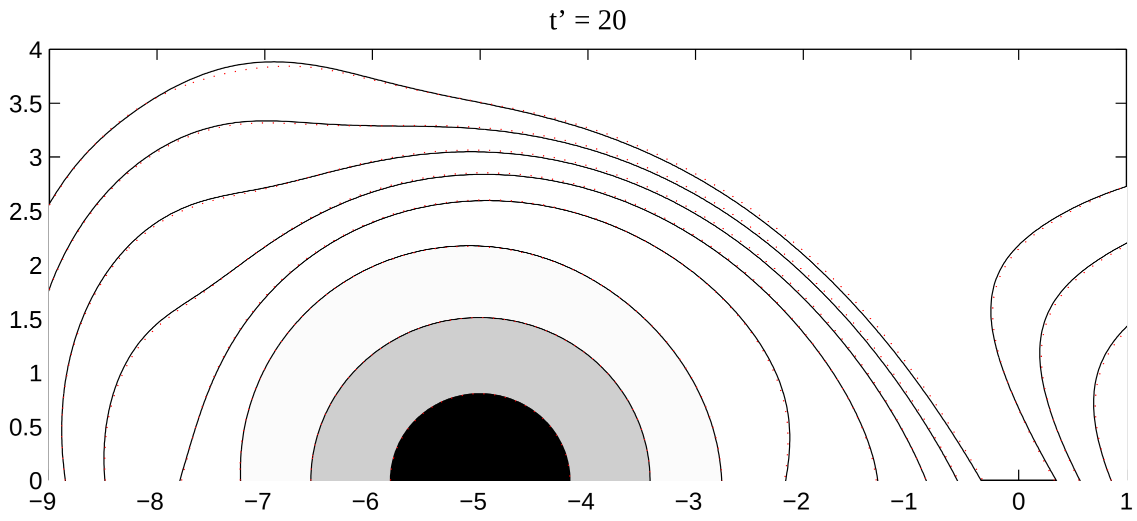


Figure 6.19: Zoom-in of the plot in Figure 6.18.

6.3 Non-symmetric Burgers vortices

Basic Burgers Vortex Solution. The Burgers vortex is, like the Lamb-Oseen vortex, an axisymmetric Gaussian distribution of vorticity which provides an exact solution to the Navier-Stokes equation. But unlike the Lamb-Oseen vortex, it is steady, preserving its size due to an equilibrium between the spreading due to viscosity, and the vorticity concentration provided by an axisymmetric strain field. The Navier-Stokes equation in an axisymmetric strain is written

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \gamma \omega + \nu \nabla^2 \omega, \tag{6.5}$$

having the following axisymmetric, steady, exact solution, given by Burgers [34]:

$$\omega(r) = \frac{\Gamma \gamma}{4\pi\nu} \exp\left(-\frac{\gamma r^2}{4\nu}\right). \tag{6.6}$$

This simple analytic solution has garnered considerable interest in the study of turbulence, where it constitutes a model for the concentrated, stretched vortex tubes that have been observed to emerge in numerical simulations of turbulence, *e.g.*, [59, 204]. The first to use stretched vortices to model turbulence was Townsend [201]; another classic vortex model in turbulence corresponds to the strained spirals of Lundgren [121].

Generalized Burgers Vortices. The basic Burgers vortex was generalized to the case of non-symmetric strain in [167], where in addition some results were provided to questions of stability in the axisymmetric case. Noting that a uni-directional vorticity (in the z direction) is consistent with a velocity field of the form

$$\begin{aligned} u_x &= -\alpha x + u(x,y,t), && (6.7)\\ u_y &= -\beta y + v(x,y,t), && (6.8)\\ u_z &= (\alpha+\beta) z, && (6.9) \end{aligned}$$

where $\alpha, \beta > 0$ are constant, Robinson and Saffman [167] classify the axisymmetric case by $\alpha = \beta > 0$. The case $\beta = 0$, $\alpha > 0$, on the other hand, is called a Burgers vortex layer. They introduced non-dimensional variables, re-writing the vorticity equation with the Reynolds number as a parameter, defined by $Re = \Gamma/(2\pi\nu)$ where Γ is the total vorticity of the vortex, and a parameter in terms of the strain components defined as $\epsilon = (\alpha - \beta)/(\alpha + \beta)$. Using a pseudo-spectral method, they were able to obtain numerical solutions with $\epsilon \leq 0.75$ for low values of Re, in the form of elliptically shaped strained vortices. As the strain parameter approaches the value of 1, they observed that the vorticity is elongated towards infinity, so that numerical solutions could not be computed. The further observations were provided that as the value of Re increases the orientation axes of the elliptical vortices rotate counterclockwise, and also the vortex shape tends more and more towards the axisymmetric shape. These conclusions were corroborated by numerical experiments in [97]. The results given in [167] were for $Re < 100$ in all cases, whereas the experiments in [97] go up to $Re = 500$.

In most later works on stretched vortices, *e.g.* [134, 161], the negative sign has been absorbed into the strain α, and the problem of strained flow is defined by a background velocity field due to strain given by $\mathbf{U} = (\alpha x, \beta y, \gamma y)$, where $\alpha + \beta + \gamma = 0$, $\alpha < 0 \leq \gamma$, $\beta \geq \alpha$. Thus, the vorticity equation is written as

$$\frac{\partial \omega}{\partial t} + (\alpha x + u)\frac{\partial \omega}{\partial x} + (\beta y + v)\frac{\partial \omega}{\partial y} = \gamma\omega + \nu\nabla^2\omega, \tag{6.10}$$

and the strained flow situations are characterized by the following positive parameter:

$$\lambda = \frac{\alpha - \beta}{\alpha + \beta}. \tag{6.11}$$

In terms of this parameter, the following classification is made [134]:

$$\begin{aligned} \lambda &= 0, && \text{axisymmetric axial strain;} && (6.12)\\ 0 < \lambda &< 1, && \text{axial strain;} && (6.13)\\ \lambda &= 1, && \text{plane strain;} && (6.14)\\ 1 < \lambda &< 3, && \text{biaxial strain with } 0 < \beta < \gamma; && (6.15)\\ \lambda &= 3, && \text{axisymmetric biaxial strain } (\beta = \gamma); && (6.16)\\ \lambda &> 3, && \text{biaxial strain with } \beta > \gamma. && (6.17) \end{aligned}$$

Some Important Results in the Literature. A pioneering computation of strained vortices was performed by Buntine and Pullin in [33], where interactions of these vortices were considered (mergers and cancellations). This work utilized a hybrid spectral/finite-difference method in vorticity formulation.

The non-symmetric Burgers vortices were studied by means of asymptotic techniques on the limit of large Reynolds number in [134]. This study was valid for all ranges of the strain parameter λ, including the case of biaxial strain, for which calculations could not be performed in [167] (where only low values of Re were considered, for which the vorticity becomes infinitely strained as $\lambda \to 1$). It was found in [134] that strong enough vortices can survive in the biaxial regime, but they experience a slow decay of circulation, a sort of (viscous) stripping.

Using a pseudo-spectral method based on that of [167], solutions for larger values of Re (up to

1000) and λ (up to 150) were obtained in [161]. These solutions are apparently steady for the larger values of Re and $\lambda > 1$. This last work has been used presently as a reference to test the capabilities of the vortex method to compute strained vortices.

Vortex Method for Strained Vortices. The vortex method developed here provides viscous effects in two dimensions by means of core spreading. To use this two-dimensional method for the computation of flow with uni-directional vorticity and three-dimensional strain, a correction to the core spreading for out-of-plane strain can be proposed in the following way (for a Gaussian with $k = 2$):

$$\frac{d\sigma^2}{dt} = 2\nu - \gamma\sigma^2, \tag{6.18}$$

whereas the other two components of strain are added directly in each velocity evaluation. The above correction is devised to cancel the component of the truncation error which is due to the strain and which is resultant in a diffusion-like term. Consider the expression for the truncation error given in (5.10). The contribution of the strain field to the velocity gradient is given by

$$\frac{\partial u_j}{\partial x_k} = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix} = \begin{pmatrix} \frac{\alpha+\beta}{2} & 0 \\ 0 & \frac{\alpha+\beta}{2} \end{pmatrix} + \begin{pmatrix} \frac{\alpha-\beta}{2} & 0 \\ 0 & -\frac{(\alpha-\beta)}{2} \end{pmatrix}. \tag{6.19}$$

The contribution to the truncation error of the diagonal tensor term in (6.19) is thus given by

$$\sigma^2 \left(\frac{\alpha+\beta}{2}\right) \frac{\partial^2 \omega}{\partial x_1^2} + \sigma^2 \left(\frac{\alpha+\beta}{2}\right) \frac{\partial^2 \omega}{\partial x_2^2} = -\sigma^2 \frac{\gamma}{2} \Delta\omega, \tag{6.20}$$

which is a diffusion-like term with a "viscosity" of $-\sigma^2 \frac{\gamma}{2}$. This component of the truncation error can then be cancelled in the core spreading as shown in (6.18), while the other component, which multiplies the right-most term in (6.19), remains as a non-diffusive truncation error. As a result, in an axisymmetric test the overall truncation error will cancel as discussed in §5.1.

Application of the Vortex Method for Strained Vortices. Using this vortex method, with core spreading corrected for out-of-plane strain, a calculation was performed for the case of plane strain, and $Re = 100$. This case is also included in [161], where the quasi-steady state vorticity contours are shown in Fig. 2(b) (p. 207), demonstrating that a bounded vorticity distribution is possible under plane strain (in contrast with the Burgers vortex layer).

The result shown in Figure 6.20 corresponds to the relaxation under strain of an initially axisymmetric Gaussian vortex upon which strain is imposed with $\lambda = 1.0$. The initial vorticity is actually provided by a Lamb-Oseen vortex with the same radius as the equilibrium Burgers vortex for the given strain. The Lamb-Oseen vortex has total circulation $\Gamma_o = 1.0$, and is initialized at $t_o = 2.5$; with a Reynolds number $Re = \Gamma/(2\pi\nu) = 100$, a value of viscosity is obtained of $\nu = 0.0016$, and

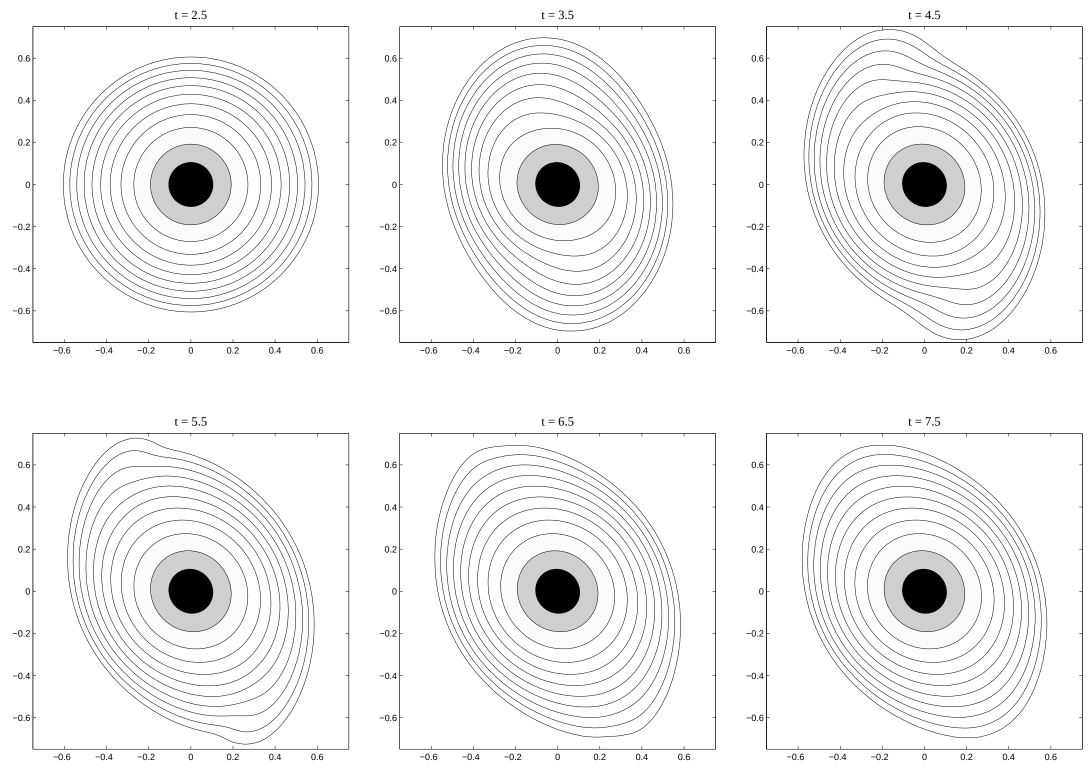


Figure 6.20: Vorticity contours, normalized by $\omega_{\max}(t_o) = 20$, during relaxation of a Gaussian vortex under a plane strain: $\lambda = 1.0$, $Re = 100$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}.

thus $\omega_{\max}(t_o) = 20$. The Lamb vortex radius is then $a_L = \sqrt{4\nu t_o} = 0.1265$. For a Burgers vortex, the radius is given by $a_B = \sqrt{4\nu/\gamma}$, so that using the value of 0.1265 we obtain a corresponding strain rate of $\gamma = 0.4$. Using this value, and $\lambda = 1.0$, the other components of strain are calculated from the relations,

$$\alpha = -\frac{1}{2}(1+\lambda)\gamma, \qquad \beta = -\frac{1}{2}(1-\lambda)\gamma. \tag{6.21}$$

The initial vorticity is discretized by placing vortex particles on a triangular lattice inside an initial domain $[-0.6, 0.6] \times [-0.6, 0.6]$, with inter-particle spacing of an equivalent resolution square lattice given by $h_{sq} = 0.0175$, and an overlap ratio $h/\sigma = 0.8$. Approximate particle strengths are obtained by the standard initialization formula $\Gamma_i = \omega_i\, h^2$, after which population control was enforced by deleting all particles whose approximate circulation was less than 10^{-12}. In this way, the particle population was reduced from an initial value of 5200 to $N = 3685$. These remaining particles were used in the RBF interpolation problem to solve for their strengths, where the approximate Γ_i's were used as initial guess to the GMRES solution method. The relevant parameters for the iterative solver were: a relative convergence tolerance (relative decrease in the residual norm) of 10^{-10}; an absolute convergence tolerance (relative decrease in the residual norm) of 10^{-20}; a maximum

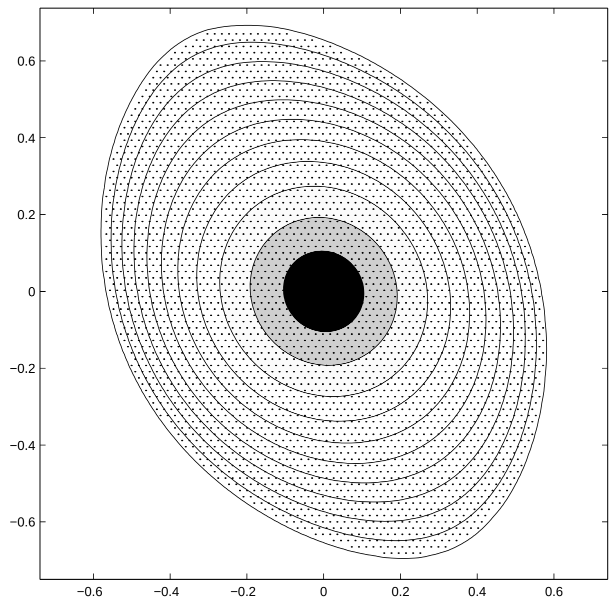


Figure 6.21: Contours of normalized vorticity and particle locations at the final simulation time, $t = 8.5$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}.

number of iterations set to the current number of particles, N. With this convergence tolerances, the initialization of the case shown in Figure 6.20, for example, converged in 237 iterations; the non-normalized 2-norm of the vector-difference between the initial guess (standard initialization) and the RBF solution was measured to be 0.00246448. Time stepping was performed by fourth-order Runge-Kutta with $\Delta t = 0.02$, and spatial adaption was carried out every 5 time steps, such that the core sizes reached a maximum value of $\sigma_{\max} = 0.0278017$, from an initial $\sigma_o = 0.021875$.

With the initial condition just described, one can estimate the radius of maximum tangential velocity to be $r_1 = 2.24\sqrt{\nu t} = 0.14167$ (using the formula given in [180], p. 254), so that one obtains a tangential velocity $v(r_1) = 0.80297$, and hence a turn-around period $t_1 \approx 1.11$. The calculation shown in Figure 6.20 seems to have reached its (quasi-) steady state after about four turn-around periods, and there is no visible difference in the vorticity during the last 50 time steps. The final vorticity distribution is shown in Figure 6.21, where in addition the particle locations at the final time are plotted.

Figure 6.22(a) shows the evolution of the value of maximum vorticity, normalized by $\omega_{\max}(t_o)$, for the calculation just described. It can be seen that there is no vorticity intensification, in view of the balance between diffusion and stretching. Upon careful scrutiny, however, the plot in Figure 6.22(a) seems to evidence a very slight *decrease* of $\omega_{\max}$ over time. This could possibly be evidence of the "stripping" mechanism detected in the asymptotic solution of [134].

To illustrate the result of applying a correction to the core spreading for out-of-plane strain, Figure 6.22(b) shows the plot of maximum vorticity normalized by $\omega_{\max}(t_o)$ for a calculation started

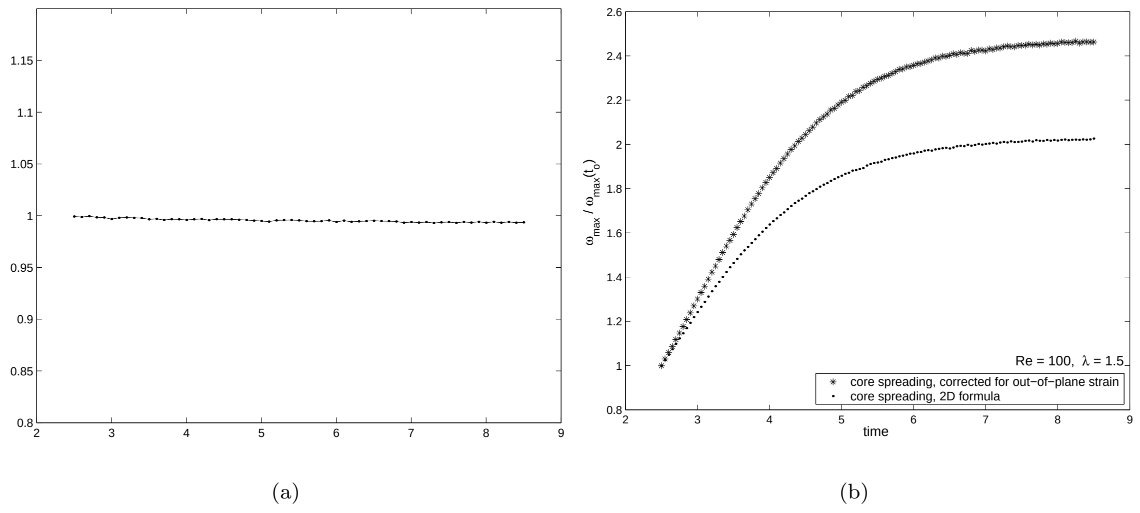


Figure 6.22: Discretely sampled $\omega_{\max}$ divided by $\omega_{\max}(t_o)$ *versus* time. (a) Core spreading corrected for out-of-plane strain with $\gamma = 0.4$; $Re = 100$, $\lambda = 1.0$; simulation starting from an equilibrium vortex (more details given in text). (b) Comparison of corrected and standard core spreading for a simulation starting from a non-equilibrium vortex with $\gamma = 1.0$.

from a non-equilibrium vortex. The initial condition was the same as that previously described, but the evolution was performed with a value of $\gamma = 1.0$, which results in an equilibrium Burgers vortex of radius $a_B = \sqrt{4\nu/\gamma} = 0.08$. Thus, the initial condition corresponds to a Gaussian vortex whose radius is 60% larger than the equilibrium Burgers vortex for the given strain, and as a result there is vorticity intensification. As shown in Figure 6.22(b), the core spreading corrected for out-of-plane strain results in enhanced vorticity intensification (as it clearly acts in contraposition to the viscous spreading of the elemental vortices).

As a further test on the vortex method with strain, the equilibrium vortex was evolved with an axisymmetric strain ($\lambda = 0$), using once again $\gamma = 0.4$ and the initial condition described above. Figure 6.23 shows that, as expected, the vortex remains in its axisymmetric steady-state.

Next, a different initial condition will be used, such that the vortex is more spread out and its maximum vorticity is four times less than previously, *i.e.*, $\omega_{\max}(t_o) = 5$. In addition, it will be discretized with many more vortex particles to attain a higher resolution. The initial condition is given by an equilibrium Burgers vortex with $a_B = 1.0$, using $\gamma = 0.1$ and $\nu = 0.025$. The Reynolds number will be the same as before (on the first instance), $Re = 100$, so that a total circulation for the vortex is given by $\Gamma = 5\pi$. An equivalent Lamb-Oseen vortex to this initial condition is given by the same circulation and viscosity, plus $t_o = 10$ (*i.e.*, $t_o = 1/\gamma$). This vortex has a maximum tangential velocity at $r_1 = 1.12$ equal to $v(r_1) = 1.595$, so that the turn-over time is 4.41, approximately. The vortex is evolved for 4 turn-over periods, using $\Delta t = 0.1$. The discretization is performed by the same procedure described previously, using $h_{sq} = 0.075$ with $h/\sigma = 0.8$, which results in 13'438 vortex

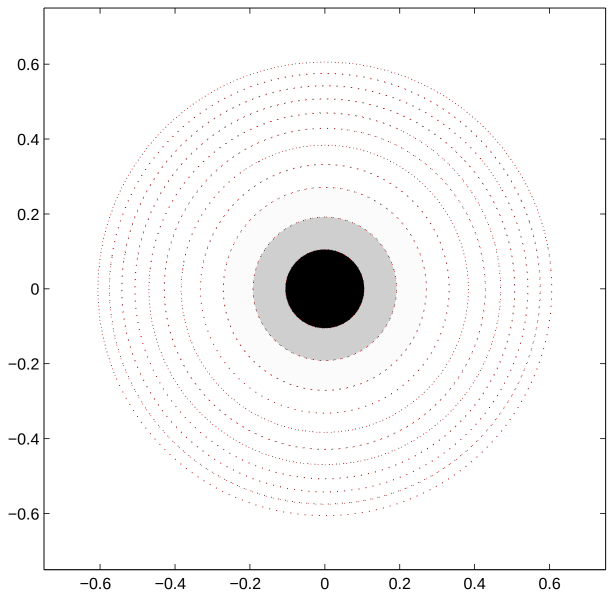


Figure 6.23: Contours of normalized vorticity for the axisymmetric strain case ($\lambda = 0$), using the equilibrium Burgers solution as initial condition, with $\gamma = 0.4$. Filled and dotted black contours correspond to the initial condition, and dotted red contours are for $t = 8.5$. Contours levels: 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}.

particles at initialization, after population control (the initialization GMRES solution converges in 221 iterations). The initial value of vortex core size is $\sigma_o = 0.09375$, which grows to a maximum of $\sigma_{\max} = 0.181282$ due to core spreading, in the five time steps between spatial adaptions.

As can be seen in Figure 6.24, where six time-slices of vorticity are shown omitting the (axisymmetric) initial condition, the evolution towards the elliptical Burgers vortex is the same as before. We note that, as previously, a slight decrease was detected in the maximum vorticity; it was measured to be a 0.6% decrease in the 4 turn-over periods.

Using this second initial condition and discretization, an experiment was performed with biaxial strain. Using $\lambda = 1.5$ and $Re = 100$ —a case which was included in [161], Fig. 2(c), p. 207—, the vortex was evolved for 240 time steps. Three snapshots of the vorticity are shown in Figure 6.25, down to a level of 10^{-8}, for no other reason than to compare with the result in [161], where this is the smallest contour level shown. It can be seen that the vorticity is indeed approaching a state like that shown in [161], but the experiment needs to go further, to obtain more stretching of the weaker contour in the y-direction. Figure 6.26 shows the vorticity contours with the particle positions plotted as well, for the final time of this calculation. The before-mentioned decrease in the maximum vorticity was here detected to be about 1% by the end of the 240 time steps (about 5.4 turn-over periods).

A continuation run of the previous calculation was carried out for a further 200 time steps. The vorticity distribution obtained at the final time $t = 54$ is shown in Figure 6.27, where a closer

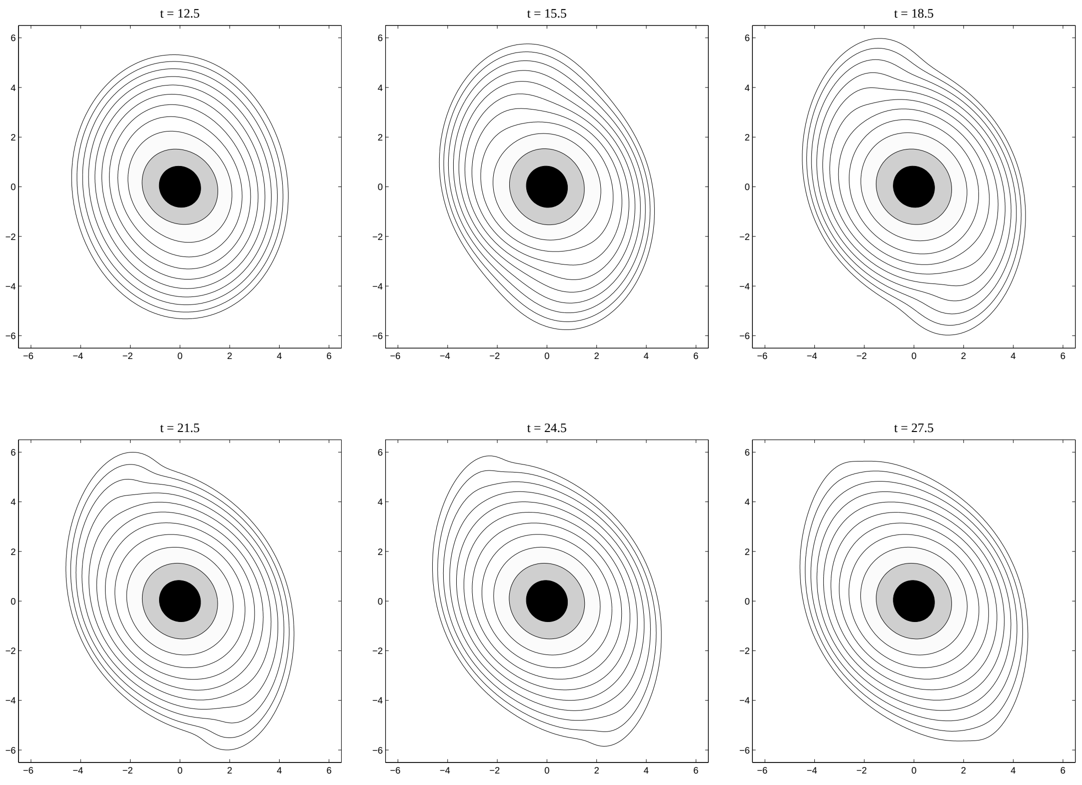


Figure 6.24: Vorticity contours, normalized by $\omega_{\max}(t_o) = 5$, during relaxation of a Gaussian vortex under a plane strain: $\lambda = 1.0$, $Re = 100$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}.

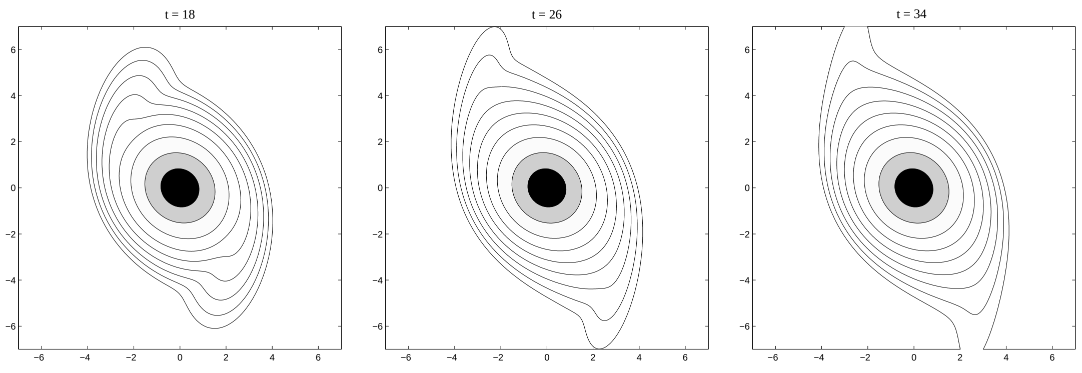


Figure 6.25: Vorticity contours, normalized by $\omega_{\max}(t_o) = 5$, during relaxation of a Gaussian vortex under biaxial strain: $\lambda = 1.5$, $Re = 100$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}.

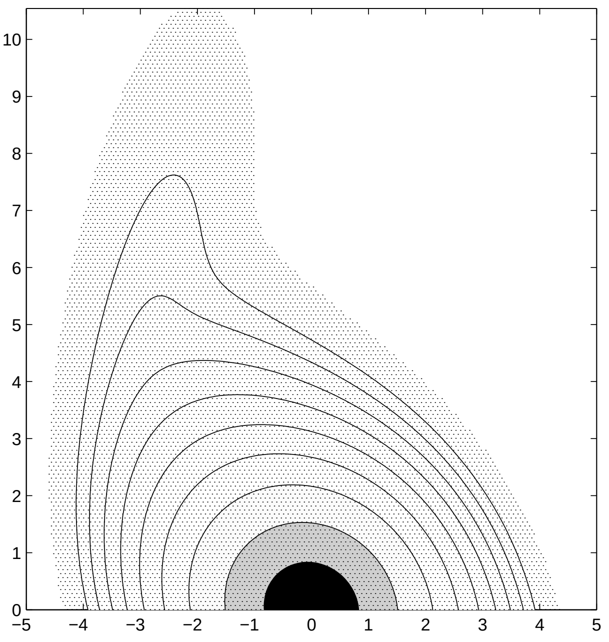


Figure 6.26: Contours of vorticity, normalized by $\omega_{\max}(t_o) = 5$, and particle locations at the final simulation time, $t = 34$, for the case under biaxial strain: $\lambda = 1.5$, $Re = 100$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}.

agreement with the result of [161] is appreciated.

A final test of the vortex method for strained vortices is performed for a higher Reynolds number, $Re = 1000$ and a value of $\lambda = 6.0$ corresponding to "extreme biaxial strain". In this case, similarly to the case of plane strain ($\lambda = 1.0$) and moderate Reynolds number $Re = 100$, one obtains a bounded, elliptical vortex. The relaxation towards that state can be seen in the snapshots of vorticity in Figure 6.28.

Although in [161] only the final, quasi-steady states are shown (apparently not fully reached by the end of the computation illustrated in Figure 6.28), we find that in this case it is quite revealing to look at the time evolution. In particular, notice that there are frank similarities between the evolution of this strained vortex at high Reynolds number, and the evolution of the co-rotating vortices in the adaptation phase, discussed in §6.2. Observing the structural similarities in the evolution of the weak areas of vorticity for these two flow situations is satisfying in ways. Especially, one of the motivations for the study of vortices under prescribed strain is the approximation made that, up to leading order, this is the same effect as the one produced by other vortices. It has long been noted, *e.g.*, in [136] and [96], that the study of vortices under strain or shear is useful for the general greater understanding of vortex interactions.

In summary, this investigation has demonstrated the capabilities of the vortex method for the study of strained, viscous vortices. A method of core spreading corrected for out-of-plane strain has

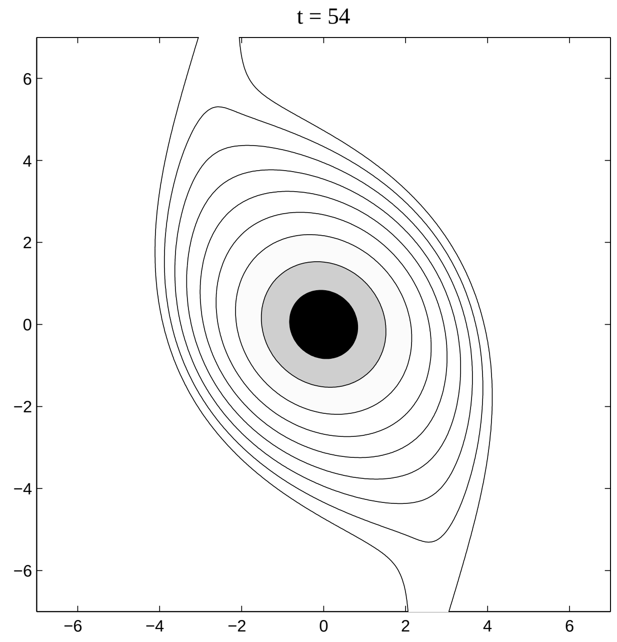


Figure 6.27: Contours of vorticity, normalized by $\omega_{\max}(t_o) = 5$, at $t = 54$, for the case under biaxial strain: $\lambda = 1.5$, $Re = 100$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}.

been tested using non-symmetric Burgers vortices, with good results as evidenced by agreement with previous investigations using pseudo-spectral methods. There are many reasons to advocate the use of vortex methods for these types of applications. For example, for problems such as that studied in §6.1 where mechanisms of shear-diffusion are active, it should be preferable to utilize a method with Newtonian viscosity instead of the prevalent hyperviscosity used in this field. In addition, one may prefer vortex methods due to the fact that the boundary conditions at infinity are satisfied, and there are no periodic images. On the other hand, for problems such as the Burgers vortex studied in this section, there is an added advantage in the fact that the vortex particles adapt to the changing vorticity support, and follow the strained areas of vorticity without reaching a boundary which is only imposed by the computational method (as is the case with spectral methods, as well as any other mesh-based method).

6.4 Observed order of convergence

It was seen in §4.3 that a numerical convergence study performed by using different spatial resolutions on the Lamb-Oseen vortex test problem suggested spectral-like convergence of the vortex method with mesh-less spatial adaption. It was also acknowledged that the axisymmetric Lamb-Oseen vortex, although its use for testing viscous vortex methods is widespread, is not a severe test problem. One reason for the benign nature of this test is its infinite regularity, but more importantly for the case of Lagrangian vortex (blob) methods is the fact that in an axisymmetric flow there is a

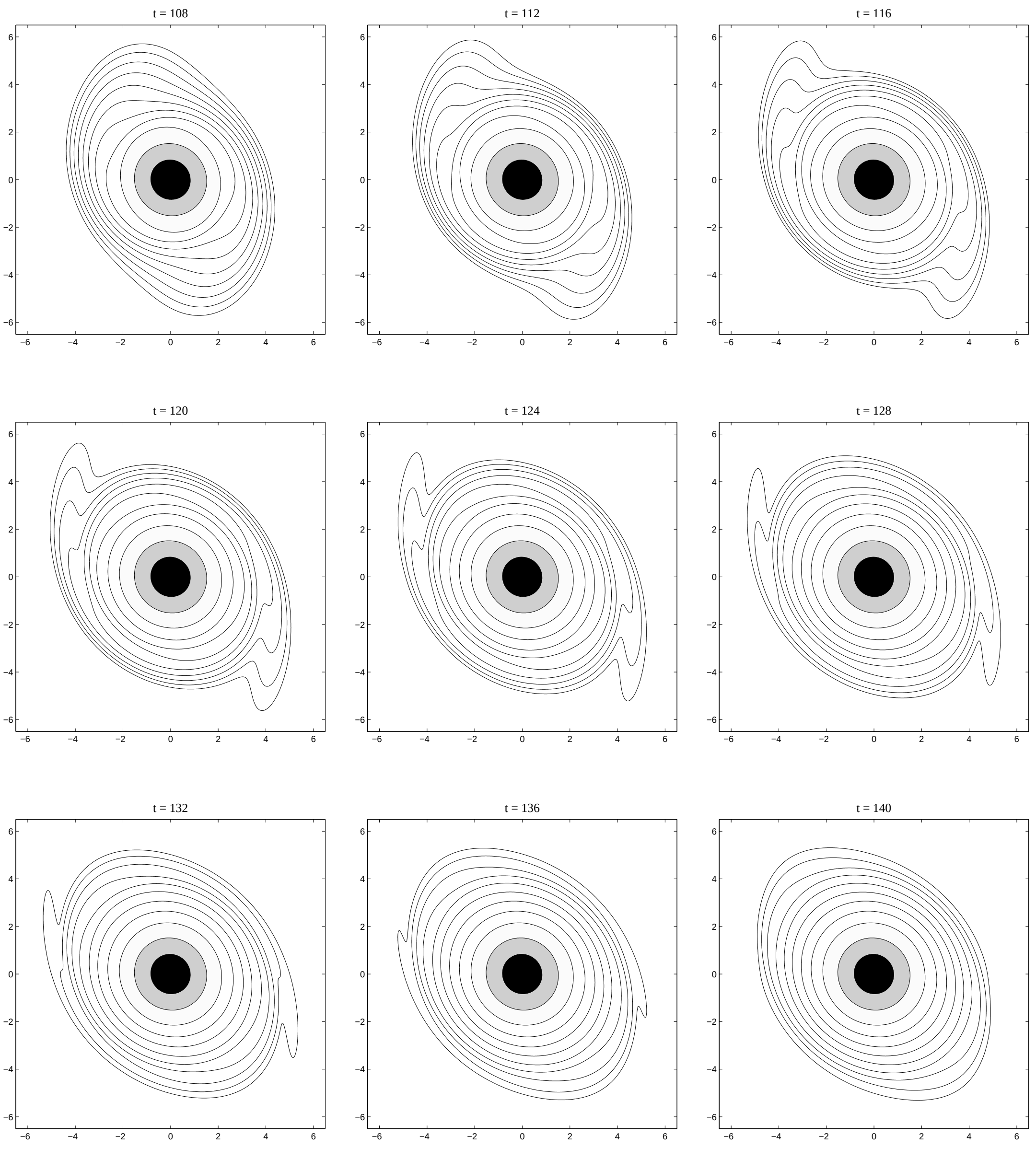


Figure 6.28: Vorticity contours, normalized by $\omega_{\max}(t_o) = 5$, during relaxation of a Gaussian vortex under an extreme biaxial strain: $\lambda = 6.0$, $Re = 1000$. Contours at 0.5, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}.

cancellation of one error component: the convection error. This was discussed in §5.1, where it was acknowledged that using exclusively test problems with axisymmetry could be misleading, due to the absence of convection error.

On the other hand, the numerical convergence exercise of §4.3 has the value of verifying the high-convergence rate of the spatial adaption process based on radial basis function interpolation. It is likely that the observation of these convergence properties depends on the regularity of the flow as well. It is nevertheless an indication that the vortex method presently developed can provide increased accuracy in comparison with standard remeshing schemes (of which the prevalent type is of third order). Furthermore, the convergence result indicates that the core spreading viscous method with adequate core size control allows this high rate of convergence as well.

The previous sections of this chapter have presented extensive numerical experiments where flow problems of significant fluid dynamical interest have been used. In these cases, we have been able to make comparisons with other published works and thus demonstrate the capabilities of the vortex method developed here to produce high-accuracy computations, in practice. It it desirable to also perform convergence studies with these more severe test problems. This is generally hampered by the lack of an analytical solution, however. But, as explained in [166], it is possible to extract the *observed order of convergence* from a grid-convergence study, using three grid solutions.

When using conventional CFD methods, where the formal (theoretical) order of convergence is known, it is also often necessary to perform a grid-convergence study. For example, if one is not certain that a solution produced is in the asymptotic range, then one can use such a study to verify the rate of convergence. Also, there are many situations when the formal order is not achieved in a certain problem being solved (for example, due to the existence of singularities), and thus a grid-convergence study will give assessment of the rate being achieved in practice.

It was first proposed in [56] that if one obtains three numerical solutions of a given problem with the same code, using three different spatial discretizations given by h_1, h_2, and h_3 with a fixed grid-refinement ratio $r = h_3/h_2 = h_2/h_1$ (taking the subscript 1 to refer to the finest resolution), then one can obtain an *empirical* convergence order p by the following relation:

$$p = \ln\left(\frac{u_{h_3} - u_{h_2}}{u_{h_2} - u_{h_1}}\right) / \ln(r). \tag{6.22}$$

Most commonly, in mesh-based methods, the grid-refinement ratio is chosen as $r = 2$. This is generally due to the desire of making economies in grid generation, by removing every-other grid point in each coordinate direction in an already generated mesh. It is not necessary, however, to have $r = 2$; on the other hand, some workers recommend that this ratio be at least $r \geq 1.1$ to positively identify discretization errors[1].

[1]NPARC Alliance CFD Verification and Validation Website, `http://www.grc.nasa.gov/WWW/wind/valid/`

To carry out a grid-convergence study of our vortex method, a new implementation of the code for computing the perturbed monopole problem of §6.1 has been developed. This is a parallel implementation, using the PETSc library as described for the numerical experiments of the co-rotating vortices and the non-symmetric Burgers vortices. The parallel code allows much larger problem sizes, and thus it is possible to vary h to smaller values without the computational resource being such a limitation. Using this new code, three solutions were computed for the same problem: a perturbed monopole (same as in §6.1) with $\delta = 0.25$ and $Re = 10^3$. The time step was considerably reduced in comparison with the previous calculations, with the goal of minimizing time-stepping errors; the runs were carried for 740 time steps with $\Delta t = 0.05$. In the three cases, the vorticity was sampled on the same mesh, corresponding to the finer value of h, and a grid-refinement ratio was chosen of $r = 1.4$. The table below shows the relative sizes of the computational experiments, in terms of number of particles (after population control enforced when approximate particle circulation was less than 10^{-10}). The particles were initialized on a triangular lattice, with an equivalent resolution square lattice having the spacing h shown in the table. Spatial adaption was performed every 10 time steps, using the same procedures described in the previous two sections.

particle spacing	$N(t = 0)$	$N(t = 37)$
$h_1 = 0.09$	20389	21978
$h_2 = 0.126$	10361	11418
$h_3 = 0.1764$	5309	5957

Problem sizes for the grid convergence study.

By means of a comparison of the point-by-point value of vorticity at each three time-slices, $t = 10, 20, 35$, and taking both a discrete L^2-norm and a maximum norm (normalized by the maximum vorticity), measurements of the observed order of convergence were performed by applying (6.22). The results are presented in the table below, where the table values correspond to the value of p.

time-slice \ norm used:	$E_\omega^{L^2}$	$E_\omega^{\text{rel,max}}$
$t = 10$	2.0282	2.0271
$t = 20$	2.026	2.027
$t = 35$	2.0175	2.0191

Observed order of convergence at three different time-slices,
using two different error norms.

This grid-convergence study indicates that the vortex method has a *second-order* rate of convergence. Since the principal difference between this test problem, and the Lamb-Oseen vortex used in §4.3 is the lack of axisymmetry in the present case, we conjecture that the observed order of convergence is fundamentally due to convection error.

One should not conclude from this result, however, that the mesh-less spatial adaption method is not able to provide an advantage over the standard vortex method with remeshing. The extensive numerical experiments performed, both with standard remeshing and with the radial basis function interpolation, demonstrate that the remeshing schemes impose an accuracy limitation on the vortex method. The RBF interpolation removes the accuracy limitation of the spatial adaption process, and leaves convection error to dominate on the rate of convergence. Therefore, higher order vortex methods are possible with techniques to reduce this error, such as higher order cutoff functions, or concepts such as using deformable basis function. Or possibly other, not yet imagined, approaches. These are subjects that merit further research.

Chapter 7

Conclusion

The Case for Mesh-less Computational Methods. The computation of viscous flow at high Reynolds numbers is severely hindered by the numerical dissipation present in the numerical methods used for computational fluid dynamics (CFD). There are many problems in fluid dynamics, such as unsteady vortical and massively separated flows, which are by and large intractable with the conventional mesh-based methods, such as finite elements and finite volumes. Such is the case for example of vortices generated in an aircraft trailing wake, vortex systems around delta wings, problems of vortex breakdown, vortex-blade interaction, dynamic stall, flows with moving boundaries such as biological flows, and many others. Likewise, in the field of research related to two-dimensional and geostrophic turbulence, most workers have determined that the work-horses that are the finite element and finite volume methods are not sufficiently accurate and thus spectral methods are generally preferred in these applications. Spectral methods, although providing high rates of convergence, are very limited in the geometries that they can deal with and often it is difficult to minimize unwanted effects of periodic images. Finally, the conventional methods of CFD suffer a significant economical burden in the need for mesh-generation. In fact, a huge body of research and development is dedicated to the subject of grid generation, which is recognized by workers in the field to be one of the most expensive phases of the CFD process.

The above are some very persuasive arguments to support the need for research involving alternative methods for CFD, adequate for high-Reynolds number flows. In particular, mesh-less methods of computation offer far-reaching opportunities for applying CFD to problems which are today very difficult to tackle with mainstream methods. Similarly, the Lagrangian approach can offer the freedom from numerically diffusive truncation errors, which are inevitable in Eulerian grid-based methods. For these reasons, among others, there is great potential in vortex methods.

Contributions of this Research to the Vortex Method. Vortex methods are a mesh-less approach for simulating unsteady, viscous flows. Their basic formulation is free from numerical diffusion, as the nonlinear term of the Navier-Stokes equations is replaced by a set of ordinary differential equations for the vortex particle trajectories. In practice, however, they have been burdened by the use of a mesh in processes of spatial adaption of the Lagrangian particles (remeshing). In this research, a fully mesh-less vortex method has been implemented, where spatial adaption is provided using radial basis function interpolation. No other vortex method has been put forth which is able to provide overlap control of the particles, thus maintaining small discretization errors throughout a computation, and at the same time can preserve the grid-free formulation.

In the past, vortex methods have faced difficulties related to the inconvenience of expressing the viscous diffusion terms in a Lagrangian framework. The determination of opportunities for improvement in ways of providing viscous effects in vortex methods has been one of the goals of the present investigation. It is believed that the method that offers greatest potential for the applications of high-Reynolds number flows is the *core spreading* method. This approach has been used only scantily, one of the reasons being an early discredit due to mathematical objections. This was surpassed less than ten years ago, by the application of a scheme for core size control, namely, the vortex splitting idea. Although proven to be convergent, it seems to be of rather low accuracy, possibly comparable to the veteran random walk method. It is also ascertained to be numerically diffusive and have low convergence rate. For this reason, the potential of core spreading methods for high-Reynolds number applications will likely be achieved by providing core size control without splitting. The greatest motivation of this work has been the implementation of the vortex method in a form that preserves its mesh-less nature, but provides accurate simulation of high-Reynolds number flows. It is perhaps surprising that the means to provide this desideratum has also made it possible to effect core size control for the application of the core spreading method without splitting. By using the method of radial basis function interpolation, the core sizes can be set to a small value during the mesh-less spatial adaption, thus satisfying the consistency requirements of core spreading.

The Need for Spatial Adaption. Particle methods suffer from an increase of discretization error over time, due to the Lagrangian distortion of the particle field. Extensive numerical experimentation performed in this thesis with the vortex method has demonstrated the crucial importance of maintaining particle overlap. Although this was known to be a required assumption in the convergence proofs of the vortex method, presently it has been shown in practice how the accuracy of discretization deteriorates by several orders of magnitude as overlap is lost. The widespread use of the particle strength exchange (PSE) viscous scheme has brought with it a solution in the form of remeshing methods, although previously other approaches were implemented such as rezoning and circulation processing schemes. The standard remeshing schemes as well as the older rezoning and circulation processing ideas all require a regular lattice of new particles to be built. In addition, thorough numerical experiments carried out as part of this investigation have demonstrated that the remeshing schemes do introduce noticeable error. In fact, they burden the other-wise mesh-less vortex method with a grid-dependent spatial adaption scheme, and thus it can be expected that numerical dissipation is introduced. Spatial adaption by means of radial basis function interpolation, in contrast, does not require any regular particle arrangement and in addition provides high convergence rates.

Accuracy of the New Vortex Method. By means of extensive numerical experiments, it has been demonstrated that the mesh-less spatial adaption for vortex methods, based on radial basis function interpolation, has the potential for considerably increased accuracy, in comparison with the standard remeshing schemes. In the literature on radial basis function interpolation it has been proved that using Gaussian bases, as here, spectral-like convergence can be achieved. Using a numerical convergence study, this result was here corroborated for the case of an axisymmetric test problem. It has to be acknowledged, however, that the Lagrangian vortex method may be limited by the existence of convection error, which is bounded by second order terms with respect to the core size of the elemental vortices. Indeed, a grid refinement study performed in this investigation using one of the non-trivial application problems resulted in an *observed second order* of convergence. Therefore, an avenue has been opened for future research, in regards to means of improving the accuracy of vortex methods by reducing convection errors. One starting point for this topic is the use of high-order kernels for the vortex blobs, already in existence for many years but only rarely used. Another is the possibility of providing deformation of the bases, thus more accurately describing the flow strain in the Lagrangian framework.

Numerical experiments were performed to compare the new method to the standard remeshing approach used in vortex methods. These indicate that a value of inter-particle spacing about three times smaller is necessary (in 2D) to obtain with remeshing (using a third order kernel) an accuracy which compares with that obtained using the radial basis function interpolation. This implies that in three dimensions it may be possible to obtain the same accuracy using a one-order of magnitude smaller problem size with the present method.

Difficulties of the Radial Basis Function Method. The solution of a radial basis function interpolation problem involves a number of computational challenges. In the first place, setting up the problem involves assembling an $N \times N$ matrix, for N computational elements (vortex particles). For this reason, the computations performed in this investigation were mainly limited by memory. For a production code (and for any efforts to produce the extension to three dimensions), it would be necessary to implement a matrix-free solution method. In addition, the straightforward implementation requires a matrix-vector multiplication within each iteration of the solution method (GMRES), which implies that the complexity is order N^2. This has been surmounted by other authors by means of applying the multipole method to implement a fast mat-vec multiply function;

in the case of a rapidly decaying basis (like the Gaussian), simpler approaches are also possible for the fast mat-vec algorithms. These techniques can make the present vortex method more attractive in terms of efficiency (possibly scaling with N).

Applications. The mesh-less vortex method developed here has been tested in three non-trivial applications of relevance in fluid dynamics. The first of these corresponds to a problem previously studied by means of another vortex method, specifically, a method using core spreading and vortex blob splitting. The problem consists of the relaxation of vortex monopoles perturbed non-linearly such that they experience a locally elliptic deformation of the core. The previously published work on this problem has shown that the flow possesses two different "attractors" or quasi-steady states that it relaxes to. If the amplitude of the quadrupolar perturbation is large enough, it was found to relax to a rotating tripole, while if it is not, the flow relaxes back to an axisymmetric state. The calculations performed here have generally reproduced the previous result very well, indeed the visualizations show much smoother vorticity. There was discrepancy, however, in the case of the lower Reynolds number calculation. This is argued to reflect the introduction of numerical diffusion by the vortex blob splitting scheme used by the other authors, and to support the claim of higher-accuracy for the present method.

In the second application, the adaptation phase of the interaction of two viscous, co-rotating vortices was studied. This problem has been recently computed using spectral methods, and we were able to reproduce these high-accuracy results with a considerably reduced problem size. This is a consequence of the need for a very large computational domain when using spectral methods, when in contrast the vortex method can concentrate the computational effort in the regions of interest. Examples were provided of the excellent adaptivity of the particle fields to the vorticity support as it deforms responding to the flow strain.

The final application, namely the study of non-symmetric Burgers vortices, required the modification of the core spreading vortex method to correctly account for three-dimensional strain in a two-dimensional calculation (possible thanks to the vorticity field being uni-directional). Previous results were also available in which pseudo-spectral methods were used. Thus, agreement with the previous work has permitted the conclusion that the vortex method can provide a useful alternative to spectral methods for the computation of strained vortices. It has the advantage of not being limited by a computational box, and adapting readily to the change of shape of the vortices as they become strained.

Closing Remarks and Future Research. In summary, the vortex method developed here has shown great promise for its application in the fields of vortex dynamics, and high Reynolds number flows. The potential for a mesh-less method to perform high-accuracy computations is quite auspicious, as CFD has been unremittingly burdened by the computational mesh for all its history. There have always been a few researchers willing to venture in some non-traditional approaches to numerical simulation, one can cite for example the trickling of efforts with methods such as smoothed particle hydrodynamics, which seems to finally be gaining ground. The field of computational mechanics, largely dominated by the finite element method for decades, is also showing a growing interest in mesh-less methods. In this area of applications, it has been slowly acknowledged that problems with large deformations or discontinuities (*e.g.*, fracture) become intractable with grid-based methods.

Contributing to the advancement of the mesh-less "paradigm" (with a slight abuse of semantics) is likely to be a worthwhile research effort. It is necessary to do more than prove the concept, indeed one needs to produce significant calculations of problems of physical, aeronautical or industrial relevance for mesh-less methods to become accepted by the mainstream CFD community. In the particular case of the vortex method presented here, using radial basis function ideas, the obvious next step is to improve the efficiency. Work in regards to application of fast methods for the velocity evaluation has not been initiated, and in addition the method would require a matrix-free implementation to be able to tackle larger problem sizes.

The next obviously necessary step is to introduce the capability of computing bounded flows. The standard way to introduce solid boundaries in vortex methods is the use of boundary elements on the surface (*i.e.*, panel methods), in conjunction with the model of vorticity generation. The use of flat panels to approximate a complex geometry, however, introduces some error due to the fact that the boundary conditions are only satisfied at the control points of the panels. An alternative formulation of the boundary geometry could be imagined that is a further application of radial basis functions, *i.e.*, using nodal functions instead of panels. These ideas have been used already to model complicated surfaces, in applications such as reconstructing surfaces from point clouds obtained from 3D range scanners. Providing boundary conditions without panels would be an exciting development, and is an avenue of research opened by the present thesis.

Another avenue for future research which appears very promising is the development of truly adaptive refinement, that is, providing the spatial adaption in a scheme that is local, applied only where needed, and based on some error estimates or measurements. This is probably a challenging project, involving considerably advanced techniques of computational geometry and theory of function approximation. It would, however, offer great potential in some applications with varying degrees of dynamics in different regions of the domain. Finally, it is envisaged that the mesh-less paradigm may be much better suited to develop multi-scale computational methods. Indeed, as a first step it would not be hard to extend the present method to variable resolution in the physical domain. This would entail basically an added degree of freedom for the particles, whereas presenting no problem in the spatial adaption process. These are some of the possible continuation topics of the present investigation.

Appendix A

Details of Implementation

Most of the preliminary calculations for this research were performed in serial codes programmed in MATLAB. These include the experiments of Chapter 3 and those of Chapter 4, except for the initial convergence study of §4.3, which was carried out with a parallel version of the codes. MATLAB codes were also used for the calculations of §6.1, whereas the following two applications, in §§6.2 and 6.3, and also the grid-refinement study of §6.4 were performed with parallel codes.

A flow diagram of the MATLAB code for numerical experiments using the Lamb-Oseen test problem is presented in Figures A.1 and A.2, which gives some idea of the volume of programming involved. The implementation in parallel, as expected, is considerably more complex and long. As a general rule, we found that translating a MATLAB routine into our parallel version required at least a ten-fold increase in the number of lines of code! This, even though our parallel implementations benefitted considerably from the use of available libraries, in particular PETSc which makes much of the message passing with MPI transparent for the user.

PETSc [6], an acronym for Portable, Extensible Toolkit for Scientific Computation, is a suite of data structures and routines which are used as building blocks in a large-scale parallel application. It has been under development at the Argonne National Laboratory for many years, funded primarily by the MICS (Mathematics, Information, and Computational Sciences) Division of the United States Department of Energy.

The PETSc library consists of a variety of components which manipulate families of mathematical objects, such as vectors, matrices, solvers, and operations of common use on these objects. It has the particularity of (relatively) easily permitting comparison of different algorithms, for example different preconditioners or solution methods.

Consider as an example the initialization function for the program to calculate the evolution of a Lamb-Oseen vortex. This function is called `INITlole_sq()` for the MATLAB version which initializes the vortex method discretization by placing Gaussian vortex particles on a square lattice, and calculating particle circulation strengths by using the time-shift correction (see §2.2.1). The input variables to this function are `rmax`, `h`, `G_cut`, corresponding respectively to: the half-range for

Flow Diagram of Matlab script Lole14.m (June 2003) : latest version of Lamb vortex evolution code

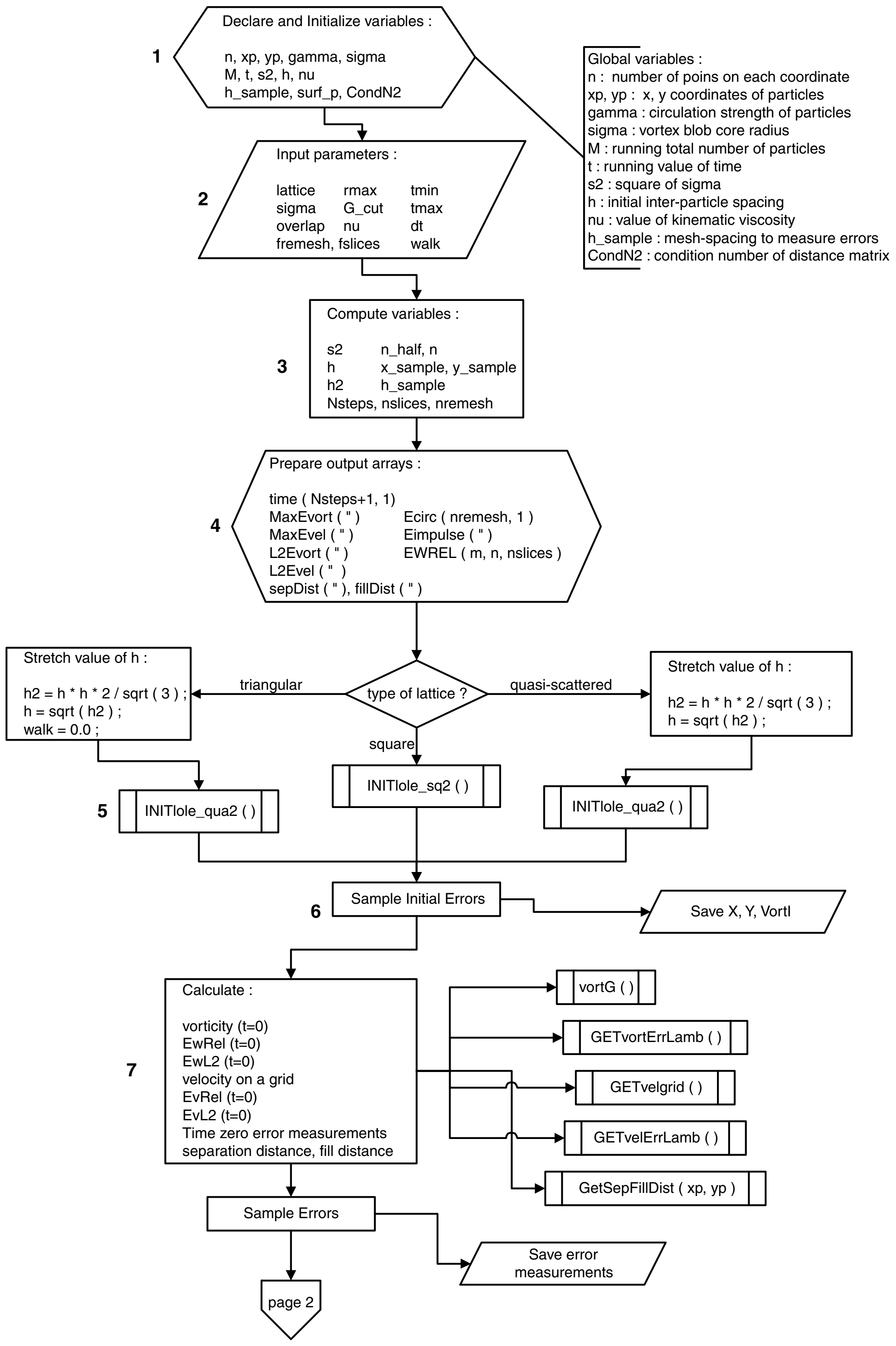


Figure A.1: Flow diagram, part 1.

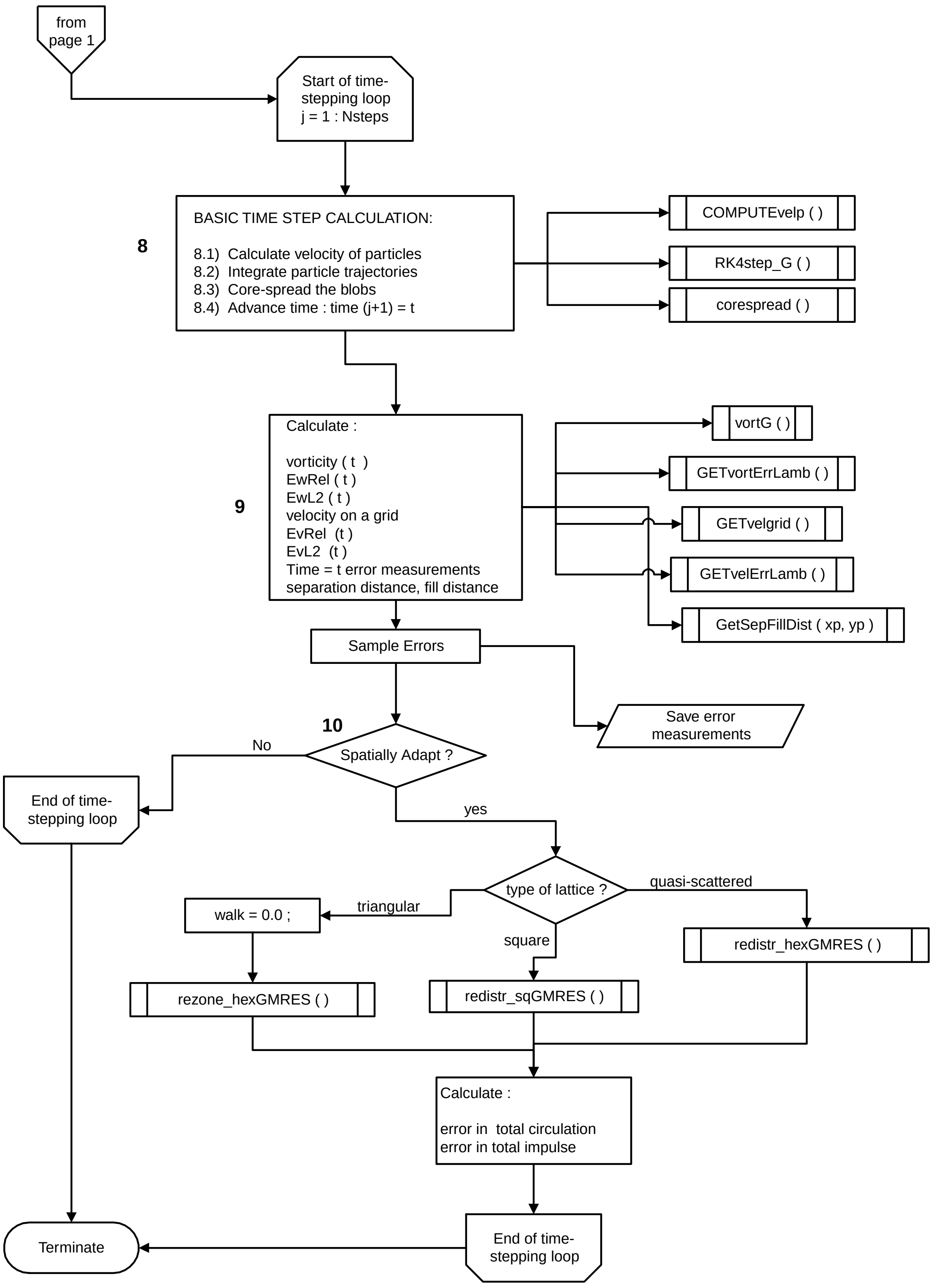


Figure A.2: Flow diagram, part 2.

the initial locations on the x and y coordinate directions, the inter-particle spacing, and the limit value of circulation strength below which to delete particles (for population control). The **Matlab** code for this function is shown in Figure A.3. In the version of the code which utilizes the radial basis function (RBF) interpolation concept for obtaining the particle strengths, we cheat slightly in the case of the Lamb-Oseen vortex by using the time-shift correction to find circulations to be able to apply population control (in other applications, the standard initialization formula (2.72) is used to find approximate values of circulation, which are used for population control). The strengths are subsequently calculated all over again with the surviving particles using the GMRES method of S. Billings (see the footnote in §4.2), and this is performed by the additional lines of code shown in Figure A.4 (added after population control and before `return`).

The initialization function just described assigns the values of three arrays containing the vortex particle information: `xp` and `yp` containing the spatial coordinates of the particle locations, and `gamma` containing the particle strengths. It also assigns the scalar `M` which is the total number of particles. These are global variables in the MATLAB version of the codes. In the production version of the codes, written in C++, we define a class `ParticleSet`, containing only public members, which correspond to the vectors holding the particle information. The class implementation is shown in Figure A.6, and it uses the PETSc vector object, denoted by `Vec`. (This is a rather simplistic implementation of the class; ideally, one would program constructors and destructors at least, plus other methods.) We also implement a class, called `Loleparams`, which is a container for all the numerical parameters defining the initial state of the Lamb-Oseen vortex, and the initial resolution requirements of the vortex blob discretization. This is subsequently used to pass the parameters into the initialization function, and others, by one name only.

With these antecedents, and giving an intimidating picture of the programming effort for the whole range of codes, the complete listing of the parallel initialization function for the Lamb-Oseen vortex, using the PETSc library, is provided in the end of this Appendix.

```
function [] = INITlole_sq(rmax,h,G_cut)
% INITLOLE  Initialize a Lamb-Oseen vortex with Gaussian vortex blobs.

global xp yp gamma M nu t s2 surf_p

   tini = t - s2/(2*nu);     % shifted initial time

 deltax = h;
 surf_p = h*h;
 deltay = deltax;

   n_sq = 2*round(rmax/h) + 1;
      x = [-rmax:deltax:rmax+n_sq*deltax]';
      y = x;

  xtemp = [];    % empty array size [0 0]
  ytemp = [];

  for j = 1:size(y,1)
      for i = 1:size(x,1)
          ytemp(length(ytemp)+1) = y(j);
          xtemp(length(xtemp)+1) = x(i);
      end
  end

  xtemp = xtemp';  % turn row vectors into column vectors
  ytemp = ytemp';

  R2 = (xtemp.^2+ytemp.^2);  % calculate radius squared array

  % Circ -- corresponding circulation, vorticity times area,
    Circ = surf_p/(4*pi*nu*tini)*exp(-R2/(4*nu*tini));

  % Now we create the actual particles
     xp = [];
     yp = [];
  gamma = [];

  for k = 1:length(xtemp)
     if abs(Circ(k))>G_cut
         xp(length(xp)+1) = xtemp(k);
         yp(length(yp)+1) = ytemp(k);
         gamma(length(gamma)+1) = Circ(k);
     end
  end

     xp = xp';
     yp = yp';
  gamma = gamma';
      M = length(xp);

return
```

Figure A.3: MATLAB code for Lamb-Oseen initialization function on a square lattice of vortex particles, using time-shift correction.

```
% common factors used in sums:
      c2 = 1/(2*s2);
      c1 = 1/(2*pi*s2);
      c3 = 1/(2*pi);

% Obtain the VORTICITY at each particle location

  R2 = (xp.^2+yp.^2);  % calculate radius squared array
  vortip = 1/(4*pi*nu*t)*exp(-R2/(4*nu*t));

% here we use SBillings routines, with our function to create distance matrix.
gamma = find_gmres_house('gauss2_VM', vortip, [xp yp], 0, [], [1e-7 200], 1);
```

Figure A.4: MATLAB code for Lamb-Oseen initialization function on a square lattice of vortex particles: additional lines of code in the version utilizing RBF interpolation to obtain circulation strengths.

```
function [AMATRIX] = my_gauss2_VM(XEVALS,XCENTERS,delta_x,deg)
% Based on
% function [AMATRIX] = my_gauss2(XEVALS,XCENTERS,delta_x,deg)
% "Calculates the matrix A_nm = phi(XEVALS_m-XCENTERS_n) assuming we
% have the a double gaussian as a basis function" by SBillings.
% LBARBA 2003 : changed to obtain the vortex method version of the
% distance matrix, scaled automatically by sigma (s2 = sigma^2).
% Lines 28 & 29 based on my_gauss() by SBillings, with modified scaling.

   global s2 CondN2

[t1 t2] = size(XEVALS);
if t1 < t2 XEVALS = XEVALS'; end
[t1 t2] = size(XCENTERS);
if t1 > t2 XCENTERS = XCENTERS'; end
Ne = length(XEVALS);
Nc = length(XCENTERS);

   % common factors used in sums:
      c1 = 1/(2*pi*s2);
      c2 = 1/(2*s2);
      c3 = 1/(2*pi);

   % Calculate distance squared matrix:
AMATRIX = c1*exp(-c2*(XEVALS(:,1)*ones(1,Nc)-ones(Ne,1)*XCENTERS(1,:)).^2) .* ...
      exp(-c2*(XEVALS(:,2)*ones(1,Nc)-ones(Ne,1)*XCENTERS(2,:)).^2);

  CondN2 = cond(AMATRIX);
fprintf('Condition of distance matrix %e\n', CondN2);

save Amn.mat AMATRIX

% IF we have a polynomial term then add it in
if (deg >= 0)
   AMATRIX = poly_terms(AMATRIX, XEVALS, XCENTERS, deg);
end
```

Figure A.5: MATLAB code for calculating the distance matrix in the vortex method application.

```
#include "petscvec.h"

class ParticleSet { public:
  Vec      xpositions;   // holds vector of x-coordinates
  Vec      ypositions;   // final vector of y-coordinates

  Vec      partvorticity;// vector of vorticity at particle locations
  Vec      strengths;    // solution of the Linear system.

  Vec      xvelocity;    // particle velocities
  Vec      yvelocity;
};
```

Figure A.6: Class `ParticleSet`, holding the information of the vortex particles

```
/* $Id: InitLole_sq.C  v.1.2,  2003 LA Barba $ */

/* This function initializes the vortex blob discretization of the
   Lamb-Oseen vortex, using Gaussian blobs on a square lattice.
*/

/* Include the class Loleparams and global variables/constants */
#include "Loleparams.h"
#include "LoleGlob.h"
#include "ParticleSet.h"
/*
  Include "petscsles.h" so that we can use SLES solvers.  Note that this file
  automatically includes:
     petsc.h       - base PETSc routines   petscvec.h - vectors
     petscsys.h    - system routines       petscmat.h - matrices
     petscis.h     - index sets            petscksp.h - Krylov subspace methods
     petscviewer.h - viewers               petscpc.h  - preconditioners
*/
#include "petscsles.h"

#undef __FUNCT__
#define __FUNCT__ "InitLole_sq" int InitLole_sq(
                 Loleparams params, PetscReal Xmax, PetscReal Xmin,
                 PetscReal Ymax, PetscReal Ymin, ParticleSet &parts )
{
  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                     Variable definitions/declarations
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
  /* --------------------basic system variables----------------------- */
 int ierr;              // error code
 const int root=0;      // identifier of root node
 int i, j, k;           // loop counters

  /* -------------------working variables----------------------------- */
  Vec      iniXvalues;   // initialization vector, for x-coordinates
  Vec      iniYvalues;   // y-coordinates
  Vec      Xvalues_sq;   // squares of x-coordinates
  Vec      Distance;     // holds values of x.^2 + y^2

  /* before population control -- then again, after generalized scatters:
     These will be the final vectors, stashed into object "parts"
  */
  Vec      xpositions;   // holds vector of x-coordinates
  Vec      ypositions;   // final vector of y-coordinates
  Vec      partstrength; // vector of particle strengths : initial guess
  Vec      partvorticity;// vector of vorticity at particle locations

  /* after population control BUT before generalized scatters
     These vectors will be destroyed before Linear Solve.
   */
  Vec      XPositions;   // holds vector of x-coordinates
  Vec      YPositions;   // final vector of y-coordinates
  Vec      PartStrength; // vector of particle strengths
  Vec      PartVorticity;// vector of vorticity at particle locations

  IS       base_index;   // index set from 0 to Nx_sq, stride 1
  IS       next_index;   // index set from ifrom to ifrom+Nx_sq
  IS       full_index;   // index set from 0 to Nparts (after P.C.), stride 1
```

```
VecScatter   vecscat;  // context used in general vector scatters

Mat         A;         // linear system matrix
SLES        sles;      // linear solver context
PC          pc;        // preconditioner context
KSP         ksp;       // Krylov subspace method context
PetscReal   norm;      // norm of difference between solution and initial guess


PetscReal *xarray;     // pointer to array of x-values, auxiliary variable
PetscReal *yarray;     // pointer to array of y-values, auxiliary variable
PetscReal *Darray;     // pointer to array of Distance values
PetscReal *ylocal;     // pointer to array of locally owned y-values
PetscReal *gammas;     // pointer to array of approximate circulation values
PetscReal *vorticity;  // pointer to array of vorticity values
int       *indices;    // pointer to array of indices

PetscReal      sigma = params.SIGMA;

PetscReal     deltax = params.H;
PetscReal     surf_p = deltax * deltax;
PetscReal     deltay = deltax;
PetscReal   gammafactor1 = 1.0 / (4.0*params.NU*params.TMIN);
PetscReal   vortfactor  = gammafactor1 / pi;
PetscReal   gammafactor2 = surf_p * vortfactor;
PetscReal   y_sq;
PetscReal   expfactor;

PetscReal   totalcirc;

/* ---------variables which will be computed by the program--------- */
 int Nparts;          // number of vortex particles at any given time


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                    Initialization of vortex blobs
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
 ierr = PetscSynchronizedPrintf( PETSC_COMM_SELF,"[%d] Initializing ... \n",
                                 RANK ); CHKERRQ(ierr);

   /* number of lattice points in the x-direction */
   /* this will be the number of nodes per row */
 int  Nx_sq = int( floor( (Xmax-Xmin)/ deltax )) + 2;

   /* number of lattice points in the y-direction */
   /* this will be the number of rows */
 int  Ny_sq = int( floor( (Ymax-Ymin)/ deltay )) + 2;

   /* Here we fill a temporary array with the values of the x-coordinate
      for the initial nodes.  This task could be divided among the nodes,
      but it seems unnecessary as all nodes will need the whole array anyway,
      so it would have to passed around in a message.  It is not that big
      (100 to 1000 elems) so for now we do it serial.
   */
 ierr = PetscMalloc( Nx_sq*sizeof(PetscReal), &xarray ); CHKERRQ(ierr);

    for ( i=0; i <= Nx_sq-1; i=i++ )
    {
        xarray[i] = Xmin + deltax * double(i);
     }

   /* create in all processors the vector of initial x-values,
      as a sequential vector, using the working array.
   */
 ierr = VecCreateSeqWithArray( MPI_COMM_SELF, Nx_sq, xarray, &iniXvalues );
      CHKERRQ(ierr);

/* Note:  According to the online manual pages,
   PETSc does NOT free the array when the vector is destroyed via VecDestroy().
   The user should not free the array until the vector is destroyed.
*/

   /*  Assemble vector, using the 2-step process:
     VecAssemblyBegin(), VecAssemblyEnd()
       Computations can be done while messages are in transition
       by placing code between these two statements.
   */
 ierr = VecAssemblyBegin( iniXvalues );CHKERRQ(ierr);
 ierr = VecAssemblyEnd( iniXvalues );CHKERRQ(ierr);

 int n;
 ierr = VecGetSize( iniXvalues, &n );CHKERRQ(ierr);
 ierr = PetscPrintf( PETSC_COMM_WORLD,"Number of x-nodes = %d \n\n", n );
      CHKERRQ(ierr);

   /*  Fill up working array with the y-coordinates
       First, calculate the number of rows to be owned per node: m
       Since this may not be exact, as a result of integer division, need to
       recalculate Ny_sq
   */
 int m = Ny_sq / NUMPROCS + 1;  // integer division
 Ny_sq = m * NUMPROCS;

 ierr = PetscPrintf( PETSC_COMM_WORLD,"Number of y-values, %d \n\n",Ny_sq);
      CHKERRQ(ierr);
 ierr = PetscPrintf( PETSC_COMM_WORLD,"Number of rows owned per node,
                     %d \n\n", m );   CHKERRQ(ierr);

   /* Fill in working array for y-values, note that each processor
      holds a different array, which will contribute a subset of a vector
   */
 ierr = PetscMalloc( Ny_sq*sizeof(PetscReal), &yarray ); CHKERRQ(ierr);

     for ( i=0; i < m; i=i++ )
     {
        yarray[i] = Ymin + deltay * double( i + RANK*m );
     }

   /* create the vector of initial y-values,
      as a parallel vector, using the working array.
   */
 ierr = VecCreateMPIWithArray( PETSC_COMM_WORLD, m, Ny_sq, yarray,
                              &iniYvalues ); CHKERRQ(ierr);

   /*  Assemble vector, using the 2-step process:
     VecAssemblyBegin(), VecAssemblyEnd()
```

```
        Computations can be done while messages are in transition
        by placing code between these two statements.
    */
  ierr = VecAssemblyBegin( iniYvalues ); CHKERRQ(ierr);
  ierr = VecAssemblyEnd( iniYvalues ); CHKERRQ(ierr);

    /* This is just to check that we get the same size as above */
  ierr = VecGetSize( iniYvalues, &Ny_sq ); CHKERRQ(ierr);
  ierr = PetscPrintf( PETSC_COMM_WORLD,"Number of y-values, after assembly,
                   %d \n\n",Ny_sq);
       CHKERRQ(ierr);

    /*
      Duplicate iniXvalues into  Xvalues_sq (they have the same format and
      partitioning ).  Then initialize Xvalues_sq and makint it equal
      to the pointwise square of iniXvalues.
    */
  ierr = VecDuplicate( iniXvalues, &Xvalues_sq ); CHKERRQ(ierr);
  //ierr = VecSet( &one, Xvalues_sq ); CHKERRQ(ierr);
  ierr = VecPointwiseMult( iniXvalues, iniXvalues, Xvalues_sq ); CHKERRQ(ierr);


    /*  From PETSc manual page:
       int VecGetOwnershipRange(Vec x,int *low,int *high)
       Returns the range of indices owned by this processor, assuming that
       the vectors are laid out with the first n1 elements on the first processor,
       next n2 elements on the second, etc. For certain parallel layouts this range
       may not be well defined.
    */
  int istart, iend;
  ierr = VecGetOwnershipRange( iniYvalues, &istart, &iend); CHKERRQ(ierr);

  ierr = PetscSynchronizedPrintf( PETSC_COMM_SELF,"[%d] Row-ownership :
                                %d, %d \n", RANK, istart, iend );
       CHKERRQ(ierr);


    /* -----------------------------------------------------------------
      The total number of locations, before applying population control: Nparts
    */
  Nparts = Nx_sq * Ny_sq;

  ierr = PetscPrintf( PETSC_COMM_WORLD,"Number of locations, before creating particles,
                   %d \n\n", Nparts);
       CHKERRQ(ierr);

    /* Create a parallel vector of size equal to the number of locations
       to hold the x-coordinates. This vector must hold consecutive
       copies of iniXvalues, Ny_sq times.
       We pass the global size (Nparts) and use PETSC_DECIDE for the
       local size; PETSc will choose a reasonable partition trying
       to put nearly an equal number of elements on each processor.
       The vector is initialized to have all elements equal to 0.0
    */
  ierr = VecCreateMPI( PETSC_COMM_WORLD, PETSC_DECIDE, Nparts, &xpositions );
        CHKERRQ(ierr);
  ierr = VecSet( &zero, xpositions );CHKERRQ(ierr);

```

```
      /*
         Duplicate vector xpositions into other vectors of the same format and
         partitioning as the initial vector.  Fill up with constant value 0.
      */
    ierr = VecDuplicate( xpositions, &ypositions ); CHKERRQ(ierr);
    ierr = VecDuplicate( xpositions, &partstrength ); CHKERRQ(ierr);
    ierr = VecDuplicate( xpositions, &partvorticity ); CHKERRQ(ierr);
    ierr = VecCopy( xpositions, ypositions ); CHKERRQ(ierr);
    ierr = VecCopy( xpositions, partstrength ); CHKERRQ(ierr);
    ierr = VecCopy( xpositions, partvorticity ); CHKERRQ(ierr);

      /* -------------------------------------------------------------------
        Access the vector values of "iniYvalues" directly. Each processor has
        access only to its portion of the vector. For default PETSc vectors
        VecGetArray() does NOT involve a copy.
      */
    ierr = VecGetArray( iniYvalues, &ylocal );CHKERRQ(ierr);


      /*  Create Distance vector by duplicating Xvalues_sq, later
          on we will add the local value of y squared to each element,
          and thus obtain the value of distance squared x^2+y^2:
      */
    ierr = VecDuplicate( Xvalues_sq, &Distance ); CHKERRQ(ierr);
    ierr = VecCopy( Xvalues_sq, Distance ); CHKERRQ(ierr);


      /*
         Create needed index sets: base_index, next_index
         Then, will do generalized scatters to copy iniXvalues into xpositions
         repeatedly, and to fill ypositions with the constant value of ylocal
         repeated Nx_sq times, and finally fill the vectors with vorticity
         and approximate circulation values.  This all occurs in the for loop
         with index between 0 and m = number of locally owned rows.
      */
    int istep = 1;
    int ifrom = 0;
    ierr = ISCreateStride( PETSC_COMM_WORLD, Nx_sq, ifrom, istep, &base_index );
         CHKERRQ(ierr);

      /* allocate working arrays to hold circulation and vorticity values
      */
    ierr = PetscMalloc( Nx_sq*sizeof(PetscReal), &gammas ); CHKERRQ(ierr);
    ierr = PetscMalloc( Nx_sq*sizeof(PetscReal), &vorticity ); CHKERRQ(ierr);

        for ( i=0; i < m; i++ )                 // for each locally owned row
        {
            y_sq = ylocal[i]*ylocal[i] ;        // obtain the square of the y-value
            ierr = VecShift( &y_sq, Distance ); // add to each elem. of vector of x.^2
                 CHKERRQ(ierr);
            /* ------------ get locally owned values of the distance squared
                this is needed because we want to take the exp() of the distances
                and presently, PETSc does not have a pointwise exp() function.
                Since the vector is sequential, all values are owned locally.
            */
            ierr = VecGetArray( Distance, &Darray ); CHKERRQ(ierr);

```

```
for ( j=0; j < Nx_sq; j++ )           // for each x-value
{
    /* --------get the approximate value of circulation
        and also the value of vorticity given by the Lamb vortex
        (this is being done in all nodes:  later try to parallelize)
    */
       expfactor = exp( -Darray[j] * gammafactor1 );
       gammas[j] = gammafactor2 * expfactor;
    vorticity[j] = vortfactor * expfactor;

}  //end of internal for-loop

ierr = VecRestoreArray( Distance, &Darray ); CHKERRQ(ierr);


 /* --------- Now create the locations in this row --------------- */
 /* We can use the vector Distance as an auxiliary vector here,
    to save memory.  It will be reset to the needed values for
    the next row in the last statement of this for-loop.
    Here, we need to assign the ypositions, xpositions, partstrength,
    and partvorticity, which are vectors of size Nparts.
 */
 /* first we set the subset of ypositions to the value ylocal[i],
    using the vector Distance as temporary auxiliary variable
 */
ierr = VecSet( &ylocal[i], Distance ); CHKERRQ(ierr);

ifrom = (i + RANK*m) * Nx_sq;   // start value of the destination index set

ierr = ISCreateStride( PETSC_COMM_WORLD, Nx_sq, ifrom, istep, &next_index );
      CHKERRQ(ierr);
ierr = VecScatterCreate( Distance, base_index, ypositions,
                         next_index, &vecscat ); CHKERRQ(ierr);
ierr = VecScatterBegin( Distance, ypositions, INSERT_VALUES,
                        SCATTER_FORWARD, vecscat );
      CHKERRQ(ierr);
ierr = VecScatterEnd( Distance, ypositions, INSERT_VALUES,
                      SCATTER_FORWARD, vecscat );
      CHKERRQ(ierr);

 /* next we set the subset of xposition to a copy of the vector iniXvalues
    using the same VecScatter object, because the vectors have the same
    parallel data layout
 */
ierr = VecScatterBegin( iniXvalues, xpositions, INSERT_VALUES,
                        SCATTER_FORWARD, vecscat );
      CHKERRQ(ierr);
ierr = VecScatterEnd( iniXvalues, xpositions, INSERT_VALUES,
                      SCATTER_FORWARD, vecscat );
      CHKERRQ(ierr);

ierr = VecScatterDestroy( vecscat ); CHKERRQ(ierr);


 /* now we need to fill the subsets of vectors of partstrength and
    partvorticity, using the arrays gammas[] and vorticity[]
 */
ierr = ISGetIndices( next_index, &indices ); CHKERRQ(ierr);
```

```
            ierr = VecSetValues( partstrength, Nx_sq, indices, gammas, INSERT_VALUES );
                 CHKERRQ(ierr);
            ierr = VecSetValues( partvorticity, Nx_sq, indices, vorticity,
                                INSERT_VALUES ); CHKERRQ(ierr);
            ierr = ISRestoreIndices( next_index, &indices ); CHKERRQ(ierr);


            ierr = ISDestroy( next_index ); CHKERRQ(ierr);

            ierr = VecCopy(Xvalues_sq,Distance); CHKERRQ(ierr);   // reset for the row

        }   // end of the external for-loop, for each locally owned row

             /*
               Assemble vectors, using the 2-step process:
               VecAssemblyBegin(), VecAssemblyEnd()
               Computations can be done while messages are in transition
               by placing code between these two statements.
               values may be cached, so VecAssemblyBegin() and VecAssemblyEnd()
               MUST be called after ALL calls to VecSetValues() have been completed
             */
        ierr = VecAssemblyBegin( partstrength ); CHKERRQ(ierr);
        ierr = VecAssemblyBegin( partvorticity); CHKERRQ(ierr);
        ierr = VecAssemblyEnd( partstrength ); CHKERRQ(ierr);
        ierr = VecAssemblyEnd( partvorticity ); CHKERRQ(ierr);

        ierr = ISDestroy( base_index ); CHKERRQ(ierr);
        ierr = VecRestoreArray( iniYvalues, &ylocal ); CHKERRQ(ierr);


    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                     Free workspace
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
      /*
      All PETSc objects should be destroyed when they
      are no longer needed.
     */
    ierr = VecDestroy( iniXvalues ); CHKERRQ(ierr);
    ierr = VecDestroy( iniYvalues ); CHKERRQ(ierr);
    ierr = VecDestroy( Xvalues_sq ); CHKERRQ(ierr);
    ierr = VecDestroy( Distance ); CHKERRQ(ierr);

      /* Free the work arrays */
    ierr = PetscFree( xarray ); CHKERRQ(ierr);
    ierr = PetscFree( yarray ); CHKERRQ(ierr);

    ierr = PetscFree( gammas ); CHKERRQ(ierr);
    ierr = PetscFree( vorticity ); CHKERRQ(ierr);


    /* ----------------------------------------------------------------------
       At this point, we have the particle information in three vectors
       of size Nparts:  xposition, yposition, partvorticity
       plus we have an approximation to the values of circulation in
       the vector partstrengths, that we can use as initial guess for
       the solution of the linear system.
       The vector partvorticity corresponds to the r.h.s. of the
       linear system.
```

```
        However, there are many particles with very tiny circulation
        strengths.

        First we want to make the problem size smaller by deleting locations
        with very small value of approximate circulation.
        --------------------------------------------------------------------
        Get the total vorticity and circulation, before population control:
     */
     ierr = VecSum( partvorticity, &totalcirc ); CHKERRQ(ierr);
     ierr = PetscPrintf( PETSC_COMM_WORLD,"Total vorticity before population control =
                    %e \n", totalcirc );
          CHKERRQ(ierr);

     ierr = VecSum( partstrength, &totalcirc ); CHKERRQ(ierr);
     ierr = PetscPrintf( PETSC_COMM_WORLD,"Total circulation before population control =
                    %e \n", totalcirc );
          CHKERRQ(ierr);


       /*
         Access the vector values of "partstrength" directly. Each processor has
         access only to its portion of the vector. For default PETSc vectors
         VecGetArray() does NOT involve a copy.
       */
     ierr = VecGetArray( partstrength, &gammas ); CHKERRQ(ierr); //we re-use gammas[]

       /*  From PETSc manual page:
          int VecGetOwnershipRange(Vec x,int *low,int *high)
          Returns the range of indices owned by this processor, assuming that
          the vectors are laid out with the first n1 elements on the first processor,
          next n2 elements on the second, etc. For certain parallel layouts this range
          may not be well defined.
       */
     ierr = VecGetOwnershipRange( partstrength, &istart, &iend ); CHKERRQ(ierr);

     ierr = PetscSynchronizedPrintf( PETSC_COMM_SELF,"[%d] Ownership of locations:
                                  %d, %d \n", RANK, istart, iend );
          CHKERRQ(ierr);

       /* the number of locations owned per processor should be the same
          because of the way the initial locations were constructed:
       */
     int locgammas = iend - istart;  // same number in every node

     PetscReal tmp;
     j = 0;
         /* in each node, count the number of locations to discard*/
         for ( i=0; i<locgammas; i++ )
         {   tmp = gammas[i];
             if ( fabs(tmp) <= params.G_CUT )
             {   j= j+1;  // count the number of nodes with tiny strength
             }
         }
     ierr = PetscSynchronizedPrintf( PETSC_COMM_SELF,
                     "[%d] number of nodes with tiny strength, %d \n", RANK, j );
          CHKERRQ(ierr);
      int *jj;           // array to hold number of elements to discard in each node
      int ndiscard;      // will hold the total number of elements to discard
      ierr = PetscMalloc( NUMPROCS*sizeof(int), &jj ); CHKERRQ(ierr);

         MPI_Gather( &j, 1, MPI_INT, jj, 1, MPI_INT, root, PETSC_COMM_WORLD );

         if ( RANK==root ) {
         ndiscard = 0;
             for ( i=0; i< NUMPROCS; i++ )
             { ndiscard = ndiscard + jj[i];  // adding the contribution of each node
             }
         }
         MPI_Bcast( &ndiscard, 1, MPI_INT, root, PETSC_COMM_WORLD );
         ierr = PetscFree( jj ); CHKERRQ(ierr);

      /* in each node, the number of elements to keep is locgammas-j */
     int locNparts = locgammas-j;  // j is different in each node


       /*  Number of particles after population control */
     Nparts = Nparts - ndiscard;

     ierr = PetscPrintf( PETSC_COMM_WORLD, "Number of particles,
                    after population control %d \n\n", Nparts );
     CHKERRQ(ierr);

       /* -----------------------------------------------------------------
             Here we create the final particle set
          -----------------------------------------------------------------*/
     /* new auxiliary arrays:
     */
     PetscReal *xarray2;
     PetscReal *yarray2;
     PetscReal *gammas2;
     PetscReal *vorticity2;
     ierr = PetscMalloc( locNparts*sizeof(PetscReal), &xarray2 ); CHKERRQ(ierr);
     ierr = PetscMalloc( locNparts*sizeof(PetscReal), &yarray2 ); CHKERRQ(ierr);
     ierr = PetscMalloc( locNparts*sizeof(PetscReal), &gammas2 ); CHKERRQ(ierr);
     ierr = PetscMalloc( locNparts*sizeof(PetscReal), &vorticity2 ); CHKERRQ(ierr);

         /*
         Access the vector values directly. Each processor has
         access only to its portion of the vector. For default PETSc vectors
         VecGetArray() does NOT involve a copy.
       */
     ierr = VecGetArray( xpositions, &xarray ); CHKERRQ(ierr);
     ierr = VecGetArray( ypositions, &yarray ); CHKERRQ(ierr);
     ierr = VecGetArray( partvorticity, &vorticity ); CHKERRQ(ierr);
     k = 0;
          for ( i=0; i<locgammas; i++ )
         {   tmp = gammas[i];
             if ( fabs(tmp) > params.G_CUT )
             {
                 xarray2[k] = xarray[i];
                 yarray2[k] = yarray[i];
                 gammas2[k] = gammas[i];
              vorticity2[k] = vorticity[i];
              k = k + 1;
             }
         }
     ierr = VecRestoreArray( partstrength, &gammas ); CHKERRQ(ierr);
     ierr = VecRestoreArray( xpositions, &xarray ); CHKERRQ(ierr);
     ierr = VecRestoreArray( ypositions, &yarray ); CHKERRQ(ierr);
     ierr = VecRestoreArray( partvorticity, &vorticity ); CHKERRQ(ierr);

   /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                           Free workspace
       - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    ierr = VecDestroy( xpositions ); CHKERRQ(ierr);
    ierr = VecDestroy( ypositions ); CHKERRQ(ierr);
    ierr = VecDestroy( partstrength ); CHKERRQ(ierr);
    ierr = VecDestroy( partvorticity ); CHKERRQ(ierr);

       /* Create parallel vectors of size equal to the new number of particles
          to hold the x-coordinates, y-coordinates, approximate circulation
          strengths, and vorticity values, using the working arrays.
       */
      ierr = VecCreateMPIWithArray( PETSC_COMM_WORLD, locNparts, Nparts, xarray2,
                                 &XPositions ); CHKERRQ(ierr);

      ierr = VecCreateMPIWithArray( PETSC_COMM_WORLD, locNparts, Nparts, yarray2,
                                 &YPositions ); CHKERRQ(ierr);

      ierr = VecCreateMPIWithArray( PETSC_COMM_WORLD, locNparts, Nparts, gammas2,
                                 &PartStrength ); CHKERRQ(ierr);

      ierr = VecCreateMPIWithArray( PETSC_COMM_WORLD, locNparts, Nparts, vorticity2,
                                 &PartVorticity ); CHKERRQ(ierr);

       /*
           Assemble vector, using the 2-step process:
         VecAssemblyBegin(), VecAssemblyEnd()
           Computations can be done while messages are in transition
           by placing code between these two statements.
       */
     ierr = VecAssemblyBegin( XPositions ); CHKERRQ(ierr);
     ierr = VecAssemblyBegin( YPositions ); CHKERRQ(ierr);
     ierr = VecAssemblyBegin( PartStrength ); CHKERRQ(ierr);
     ierr = VecAssemblyBegin( PartVorticity ); CHKERRQ(ierr);

     ierr = VecAssemblyEnd( XPositions ); CHKERRQ(ierr);
     ierr = VecAssemblyEnd( YPositions ); CHKERRQ(ierr);
     ierr = VecAssemblyEnd( PartStrength ); CHKERRQ(ierr);
     ierr = VecAssemblyEnd( PartVorticity ); CHKERRQ(ierr);

     /*
        Get the change in circulation, due to population control:
     */

     PetscReal totalcirc2;
     PetscReal Ecirc;


     ierr = VecSum( PartStrength, &totalcirc2 );CHKERRQ(ierr);

     Ecirc = totalcirc - totalcirc2;
     ierr = PetscPrintf(PETSC_COMM_WORLD,
            "Total circulation lost due to population control = %e \n", Ecirc );
          CHKERRQ(ierr);

     ierr = VecGetOwnershipRange( XPositions, &istart, &iend ); CHKERRQ(ierr);

     ierr = PetscSynchronizedPrintf( PETSC_COMM_SELF,
            "[%d] Ownership of particle vectors: %d, %d \n", RANK, istart, iend );
          CHKERRQ(ierr);


     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         Compute the matrix that defines the linear system, Ax = b.
       - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

     /*    ---From the PETSc manual pages:
       Create parallel matrix, specifying only its global dimensions.
       When using MatCreate(), the matrix format can be specified at
       runtime. Also, the parallel partitioning of the matrix is
       determined by PETSc at runtime.
       Performance tuning note:  For problems of substantial size,
       preallocation of matrix memory is crucial for attaining good
       performance.  Since preallocation is not possible via the generic
       matrix creation routine MatCreate(), we recommend for practical
       problems instead to use the creation routine for a particular matrix
       format, e.g.,
           MatCreateMPIAIJ() - parallel AIJ (compressed sparse row)
           MatCreateMPIBAIJ() - parallel block AIJ
       ...also
           MatCreateMPIDense()
      */

     ierr = MatCreateMPIDense( PETSC_COMM_WORLD, PETSC_DECIDE, PETSC_DECIDE,
                              Nparts, Nparts, PETSC_NULL, &A ); CHKERRQ(ierr);
     ierr = PetscPrintf( PETSC_COMM_WORLD, "Creating parallel distance matrix ...\n\n" );
          CHKERRQ(ierr);

       /*  MatGetOwnershipRange()
           Returns the range of matrix rows owned by this processor,
           assuming that the matrix is laid out with the first n1 rows
           on the first processor, the next n2 rows on the second, etc.
           For certain parallel layouts this range may not be well defined.
       */
     int Istart;      // the global index of the first local row
     int Iend;       // one more than the global index of the last local row
     int locsize;

     ierr = MatGetOwnershipRange( A, &Istart, &Iend ); CHKERRQ(ierr);

       /* To fill up matrix elements, each processor needs to know the
          x-values and y-values of all nodes. Hence we need to scatter
          the parallel vectors XPositions and Ypositions to sequential
          vectors in all the nodes.  Then each node can calculate the
          values to insert in the matrix for the rows it owns.

          The matrix values are given by:
          A_{ij} = 1/(2\pi\sigma^2) \exp(-[(x_i-x_j)^2+(y_i-y_j)^2] / 2\sigma^2)
       */
     Vec seqXPos;
     Vec seqYPos;

     ierr = VecCreateSeq( PETSC_COMM_SELF, Nparts, &seqXPos ); CHKERRQ(ierr);
     ierr = VecDuplicate( seqXPos, &seqYPos ); CHKERRQ(ierr);

     istep = 1;
     ifrom = 0;

     ierr = ISCreateStride( PETSC_COMM_WORLD, Nparts, ifrom, istep, &full_index );
          CHKERRQ(ierr);
     // we do not destroy this index set in this block, to re-use it later below

     ierr = VecScatterCreate( XPositions, full_index, seqXPos, full_index, &vecscat );
          CHKERRQ(ierr);

     ierr = VecScatterBegin( XPositions, seqXPos, INSERT_VALUES,
                             SCATTER_FORWARD, vecscat ); CHKERRQ(ierr);
     ierr = VecScatterEnd( XPositions, seqXPos, INSERT_VALUES, SCATTER_FORWARD, vecscat );
          CHKERRQ(ierr);

     ierr = VecScatterBegin( YPositions, seqYPos, INSERT_VALUES,
                             SCATTER_FORWARD, vecscat );  CHKERRQ(ierr);
     ierr = VecScatterEnd( YPositions, seqYPos, INSERT_VALUES, SCATTER_FORWARD, vecscat );
          CHKERRQ(ierr);

     ierr = VecScatterDestroy( vecscat ); CHKERRQ(ierr);


       /*
         Access the vector values directly. Each processor has
         access only to its portion of the vector, but since these are
         sequential vectors, all values are available.
         For default PETSc vectors VecGetArray() does NOT involve a copy.

         Once again, we are re-using the auxiliary arrays xarray and yarray.
       */
     ierr = VecGetArray( seqXPos, &xarray ); CHKERRQ(ierr);
     ierr = VecGetArray( seqYPos, &yarray ); CHKERRQ(ierr);

       /* Now fill the matrix values using the auxiliary arrays.

         - Each processor needs to insert only elements that it owns
           locally (but any non-local elements will be sent to the
           appropriate processor during matrix assembly).
         - Always specify global rows and columns of matrix entries.

         The matrix values are given by:
           A_{ij} = 1/(2\pi\sigma^2) \exp(-[(x_i-x_j)^2+(y_i-y_j)^2] / 2\sigma^2)
       */
             expfactor = 1.0 / (2.0*sigma*sigma);
     PetscReal Afactor = expfactor / pi;
     PetscReal xdist, ydist;
     PetscReal value;

     for ( i=Istart; i<Iend; i++ ) // for each locally owned matrix row
     {
         for ( j=0; j<Nparts; j++ ) // for each x,y coordinate value
         {
           xdist = ( xarray[i] - xarray[j] )*( xarray[i] - xarray[j] );
           ydist = ( yarray[i] - yarray[j] )*( yarray[i] - yarray[j] );
           value = Afactor * exp( -(xdist + ydist)*expfactor );

           ierr = MatSetValues( A, 1, &i, 1, &j, &value, INSERT_VALUES );
                CHKERRQ(ierr);

         } // end internal for loop

     }  // end of for loop, for each locally owned matrix row

    /*
       Assemble matrix, using the 2-step process:
         MatAssemblyBegin(), MatAssemblyEnd()
       Computations can be done while messages are in transition
       by placing code between these two statements.
    */
     ierr = MatAssemblyBegin( A, MAT_FINAL_ASSEMBLY ); CHKERRQ(ierr);
     ierr = MatAssemblyEnd( A, MAT_FINAL_ASSEMBLY ); CHKERRQ(ierr);


     /* Restore working arrays */
     ierr = VecRestoreArray( seqXPos, &xarray ); CHKERRQ(ierr);
     ierr = VecRestoreArray( seqYPos, &yarray ); CHKERRQ(ierr);

   /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                           Free workspace
       - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    ierr = VecDestroy( seqXPos ); CHKERRQ(ierr);
    ierr = VecDestroy( seqYPos ); CHKERRQ(ierr);


    /*  ------------------------------------------------------------------
        - When solving a linear system, the vectors and matrices MUST
          be partitioned accordingly.  PETSc automatically generates
          appropriately partitioned matrices and vectors when MatCreate()
          and VecCreate() are used with the same communicator.

       ->   We need to scatter the vector PartVorticity, holding the r.h.s.
          so that it is partitioned correctly.

          In fact, it would be best if all the particle information were
          partitioned equally among the processors, so we use here the
          same VecScatter context for all our four vectors.

          For this purpose, we re-use the lower case vector names, but put
          these now in class ParticleSet (except for the initial guess
          in vector partstrength).
    */
   locsize = Iend - Istart;  // size of locally owned portion of final vectors

   /*  Here, the four vectors that make up the ParticleSet object "parts"
       are set to their distributed size, and set to type VECMPI.

       Note that if the type is set before the sizes, PETSc returns an error.
   */
```

```
ierr = VecSetSizes( parts.xpositions, locsize, Nparts ); CHKERRQ(ierr);
ierr = VecSetSizes( parts.ypositions, locsize, Nparts ); CHKERRQ(ierr);
ierr = VecSetSizes( parts.partvorticity, locsize, Nparts ); CHKERRQ(ierr);
ierr = VecSetSizes( parts.strengths, locsize, Nparts ); CHKERRQ(ierr);

ierr = VecSetType( parts.xpositions, VECMPI ); CHKERRQ(ierr);
ierr = VecSetType( parts.ypositions, VECMPI ); CHKERRQ(ierr);
ierr = VecSetType( parts.strengths, VECMPI ); CHKERRQ(ierr);
ierr = VecSetType( parts.partvorticity, VECMPI ); CHKERRQ(ierr);

 /* the vector partstrength will hold the initial guess */
ierr = VecDuplicate( parts.xpositions, &partstrength ); CHKERRQ(ierr);

   /*  We use generalized scatters to fill up the vectors making up
       the particle set, using the previous vectors which are distributed
       non-uniformly accross the processors, due to population control.

       We are re-using the index set "full_index" created above.
   */
   ierr = VecScatterCreate( XPositions, full_index, parts.xpositions,
                           full_index, &vecscat );  CHKERRQ(ierr);

   ierr = VecScatterBegin( XPositions, parts.xpositions, INSERT_VALUES,
                            SCATTER_FORWARD, vecscat );  CHKERRQ(ierr);
   ierr = VecScatterEnd( XPositions, parts.xpositions, INSERT_VALUES,
                          SCATTER_FORWARD, vecscat); CHKERRQ(ierr);

   ierr = VecScatterBegin( YPositions, parts.ypositions, INSERT_VALUES,
                            SCATTER_FORWARD, vecscat ); CHKERRQ(ierr);
   ierr = VecScatterEnd( YPositions, parts.ypositions, INSERT_VALUES,
                          SCATTER_FORWARD, vecscat ); CHKERRQ(ierr);

   ierr = VecScatterBegin( PartStrength, partstrength, INSERT_VALUES,
                            SCATTER_FORWARD, vecscat ); CHKERRQ(ierr);
   ierr = VecScatterEnd( PartStrength, partstrength, INSERT_VALUES,
                          SCATTER_FORWARD, vecscat ); CHKERRQ(ierr);

   ierr = VecScatterBegin( PartVorticity, parts.partvorticity,
                           INSERT_VALUES, SCATTER_FORWARD, vecscat );
       CHKERRQ(ierr);
   ierr = VecScatterEnd( PartVorticity, parts.partvorticity,
                         INSERT_VALUES, SCATTER_FORWARD, vecscat );
       CHKERRQ(ierr);

   ierr = VecScatterDestroy( vecscat ); CHKERRQ(ierr);

   ierr = ISDestroy( full_index ); CHKERRQ(ierr);

 //initial guess, solution vector:
ierr = VecCopy( partstrength, parts.strengths ); CHKERRQ(ierr);

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                    Free workspace
    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

 ierr = VecDestroy( XPositions ); CHKERRQ(ierr);
 ierr = VecDestroy( YPositions ); CHKERRQ(ierr);
 ierr = VecDestroy( PartStrength ); CHKERRQ(ierr);
 ierr = VecDestroy( PartVorticity ); CHKERRQ(ierr);

     /* Free the work arrays used to create the above vectors*/
  ierr = PetscFree( xarray2 ); CHKERRQ(ierr);
  ierr = PetscFree( yarray2 ); CHKERRQ(ierr);
  ierr = PetscFree( gammas2 ); CHKERRQ(ierr);
  ierr = PetscFree( vorticity2 ); CHKERRQ(ierr);


 /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
               Create the linear solver and set various options
    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
 /*
    Create linear solver context
 */
 ierr = SLESCreate( PETSC_COMM_WORLD, &sles ); CHKERRQ(ierr);

 /*
    Set operators. Here the matrix that defines the linear system
    also serves as the preconditioning matrix.
 */
 ierr = SLESSetOperators( sles, A, A, DIFFERENT_NONZERO_PATTERN ); CHKERRQ(ierr);

 /*
    Set linear solver defaults for this problem (optional).
    - By extracting the KSP and PC contexts from the SLES context,
      we can then directly call any KSP and PC routines to set
      various options.
    - The following two statements are optional; all of these
      parameters could alternatively be specified at runtime via
      SLESSetFromOptions().  All of these defaults can be
      overridden at runtime, as indicated below.
 */

 /*Returns the KSP context for a SLES solver */
 ierr = SLESGetKSP( sles, &ksp ); CHKERRQ(ierr);

 /* Sets the relative, absolute, divergence, and maximum iteration
    tolerances used by the default KSP convergence testers.

  KSPSetTolerances(KSP ksp,PetscReal rtol,PetscReal atol,PetscReal dtol,int maxits)
    ksp  - the Krylov subspace context
    rtol  - the relative convergence tolerance (relative decrease in the residual norm)
    atol  - the absolute convergence tolerance (absolute size of the residual norm)
    dtol  - the divergence tolerance (amount residual can increase before
          KSPDefaultConverged() concludes that the method is diverging)
    maxits  - maximum number of iterations to use

 */
 int maxits = Nparts;
 ierr = KSPSetTolerances( ksp, 1.e-10, 1.e-20,PETSC_DEFAULT, maxits ); CHKERRQ(ierr);

 /* Tells the iterative solver that the initial guess is nonzero;
    otherwise KSP assumes the initial guess is to be zero
    (and thus zeros it out before solving).
 */
 ierr = KSPSetInitialGuessNonzero( ksp, PETSC_TRUE ); CHKERRQ(ierr);

```

```
  /*
    Set runtime options, e.g.,
        -ksp_type <type> -pc_type <type> -ksp_monitor -ksp_rtol <rtol>
    These options will override those specified above as long as
    SLESSetFromOptions() is called _after_ any other customization
    routines.
  */
   ierr = SLESSetFromOptions( sles );CHKERRQ(ierr);

  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                      Solve the linear system
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
   int iterations;

   ierr = SLESSolve( sles, parts.partvorticity, parts.strengths, &iterations );
        CHKERRQ(ierr);

  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                      Check solution and clean up
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
   KSPConvergedReason reason;

   ierr = KSPGetConvergedReason( ksp, &reason ); CHKERRQ(ierr);
      PetscPrintf( PETSC_COMM_WORLD, "KSPConvergedReason: %d\n", reason );
   /*
     Check the difference w.r.t. initial guess
   */
   PetscScalar neg_one = -1.0;

   ierr = VecAXPY( &neg_one, parts.strengths, partstrength ); CHKERRQ(ierr);
   ierr = VecNorm( partstrength, NORM_2, &norm ); CHKERRQ(ierr);


   /*
     Print convergence information.
   */
   ierr = PetscPrintf( PETSC_COMM_WORLD, "Norm of difference w.r.t. initial guess
                  %A - iterations %d\n", norm, iterations ); CHKERRQ(ierr);

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                         Free workspace
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
   ierr = SLESDestroy( sles ); CHKERRQ(ierr);

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                         Free workspace
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

   ierr = MatDestroy( A );CHKERRQ(ierr);

   ierr = VecDestroy( partstrength );CHKERRQ(ierr);

    return Nparts;
}
```

Bibliography

[1] G. Alessandrini, A. Douglis, and E. Fabes. An approximate layering method for the Navier-Stokes equations in bounded cylinders. *Ann. Mat. Pura Appl.*, 135:329–347, 1983.

[2] C. B. Allen. Time-stepping and grid effects on convergence of inviscid rotor flows. AIAA #2002-2817, 20th Applied Aerodynamics Conference, St. Louis MO, June 2002.

[3] C. B. Allen. Convergence of steady and unsteady formulations for inviscid hovering rotor solutions. *Int. J. Num. Meth. Fluids*, 41(9):931–949, 2003.

[4] C. Anderson and C. Greengard. On vortex methods. *SIAM J. Num. Anal.*, 22:413–440, 1985.

[5] C. R. Anderson and C. Greengard, editors. *Vortex Dynamics and Vortex Methods*, volume 28 of *AMS Lectures in Appl. Math.*, 1991.

[6] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. Curfman-McInnes, B. F. Smith, and H. Zhang. PETSc User's Manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2002.

[7] K. Ball. Eigenvalues of Euclidean distance matrices. *J. Approx. Theory*, 68:74–82, 1992.

[8] K. Ball, N. Sivakumar, and J.D. Ward. On the sensitivity of radial basis interpolation to minimal distance separation. *J. Approx. Theory*, 68:401–426, 1992.

[9] G. K. Batchelor. *An Introduction to Fluid Dynamics.* Cambridge University Press, 1967.

[10] B. Bayly. Three-dimensional instability of elliptical flow. *Phys. Rev. Lett.*, 57:2160–2163, 1986.

[11] J. T. Beale. A convergent 3-D vortex method with grid-free stretching. *Math. Comp.*, 46:401–424, 1986.

[12] J. T. Beale. On the accuracy of vortex methods at large time. In B. Engquist et al., editors, *Computational Fluid Dynamics and Reacting Gas Flows*, pages 19–32. Springer-Verlag, New York, 1988.

[13] J. T. Beale and A. Majda. Rates of convergence for viscous splitting of the Navier-Stokes equations. *Math. Comp.*, 37:243–259, 1981.

[14] J. T. Beale and A. Majda. Vortex methods I: Convergence in three dimensions. *Math. Comp.*, 39:1–27, 1982.

[15] J. T. Beale and A. Majda. Vortex methods II: High-order accuracy in two and three dimensions. *Math. Comp.*, 39:29–52, 1982.

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

[17] R. K. Beatson, J. B. Cherrie, and C. T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Adv. Comp. Math.*, 11:253–270, 1999.

[18] R. K. Beatson, G. Goodsell, and M. J. D. Powell. On multigrid techniques for thin plate spline interpolation in two dimensions. In J. Renegar, M. Shub, and S. Smale, editors, *The Mathematics of Numerical Analysis*, pages 77–97. American Mathematical Society: Rhode Island, 1995.

[19] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740, 2000.

[20] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: I. *Comp. Math. Applic.*, 24(12):7–19, 1992.

[21] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: Moment-based methods. *SIAM J. Sci. Comput.*, 19(5):1428–1449, 1998.

[22] A. Beaudoin, S. Huberson, and E. Rivoalen. Simulation of anisotropic diffusion by means of a diffusion velocity method. *J. Comp. Phys.*, 186:122–135, 2003.

[23] Jörn Behrens, Armin Iske, and Stephan Pöhn. Effective node adaption for grid-free semi-Lagrangian advection. In Th. Sonar and I. Thomas, editors, *Discrete Modelling and Discrete Algorithms in Continuum Mechanics*, pages 110–119. Logos Verlag, Berlin, 2001.

[24] G. Benfatto and M. Pulvirenti. A diffusion process associated to the Prandtl equations. *J. Funct. Anal.*, 52:330–343, 1983.

[25] P. S. Bernard. A deterministic vortex sheet method for boundary layer flow. *J. Comp. Phys.*, 117:132–145, 1995.

[26] A. J. Bernoff and J. F. Lingevitch. Rapid relaxation of an axisymmetric vortex. *Phys. Fluids*, 6(11):3717–3723, 1994.

[27] S. D. Billings, R. K. Beatson, and G. N. Newsam. Interpolaton of geophysical data using continuous global surfaces. *Geophysics*, 67(6):1810–1822, 2002.

[28] G. Birkhoff. Helmholtz and Taylor instability. In *Proc. Symp. Appl. Math.*, volume XIII, pages 55–76. Amer. Math. Soc., 1962.

[29] G. Birkhoff and J. Fisher. Do vortex sheets roll up? *Rend. Circ. Mat. Palermo Ser. 2*, 8:77–90, 1959.

[30] S. H. Brecht and J. R. Ferrante. Vortex-in-cell calculations in three dimensions. *Comp. Phys. Comm*, 58:25–54, 1990.

[31] M. D. Buhmann. Radial basis functions. *Acta Numerica*, pages 1–38, 2000.

[32] M. D. Buhmann. *Radial Basis Functions. Theory and Implementations.* Cambridge University Press, 2003.

[33] J. D. Buntine and D. I. Pullin. Merger and cancellation of strained vortices. *J. Fluid Mech.*, 205:263–295, 1989.

[34] J. M. Burgers. A mathematical model illustrating the theory of turbulence. *Adv. Appl. Mech.*, 1:171–199, 1948.

[35] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the* 28th *Annual Conference on Computer Graphics and Interactive Techniques*, pages 67–76. ACM Press New York, NY, 2001.

[36] C. K. Chan, K. S. Lau, and B. L. Zhang. Simulation of a premixed turbulent flame with the discrete vortex method. *Int. J. Num. Meth. Eng.*, 48:613–627, 2000.

[37] C. C. Chang and R. L. Chern. A numerical study of flow around an impulsively started circular cylinder by a deterministic vortex method. *J. Fluid Mech.*, 233:243–263, 1991.

[38] P. Chatelain and A. Leonard. Face-centred cubic lattices and particle redistribution in vortex methods. *J. Turbulence*, 3:046, 2002.

[39] A. Y. Cheer. Unsteady separated wake behind an impulsively started cylinder in slightly viscous fluid. *J. Fluid Mech.*, 201:485–505, 1989.

[40] J. P. Choquin and B. Lucquin-Desreux. Accuracy of a deterministic particle method for Navier-Stokes equations. *Int. J. Num. Meth. in Fluids*, 8:1439–1458, 1988.

[41] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[42] A. J. Chorin. Numerical study of slightly viscous flow. *J. Fluid Mech.*, 57:785–796, 1973.

[43] J. P. Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. *J. Comp. Phys.*, 13:363–379, 1973.

[44] N. R. Clarke and O. R. Tutty. Construction and validation of a discrete vortex method for the two-dimensional incompressible Navier-Stokes equations. *Computers & Fluids*, 23:751–783, 1994.

[45] G.-H. Cottet. Convergence of a vortex-in-cell method for the two-dimensional Euler equations. *Math. Comp.*, 49(180):407–425, 1987.

[46] G.-H. Cottet. Large time behavior of deterministic particle approximations to the Navier-Stokes equations. *Math. Comp.*, 56:45–59, 1991.

[47] G.-H. Cottet, J. Goodman, and T. Y. Hou. Convergence of the grid-free point vortex method for the three-dimensional Euler equations. *SIAM J. Num. Anal.*, 28(2):291–307, 1991.

[48] G.-H. Cottet and P. Koumoutsakos. *Vortex Methods. Theory and Practice.* Cambridge University Press, 2000.

[49] G.-H. Cottet, P. Koumoutsakos, and M. L. Ould Salihi. Vortex methods with spatially varying cores. *J. Comp. Phys.*, 162:164–185, 2000.

[50] G.-H. Cottet and S. Mas-Gallic. Une méthode de décomposition pour une équation de type convection-diffusion combinant résolution explicite et méthode particulaire. *C. R. Acad. Sci. Paris*, 297:133–136, 1983.

[51] G.-H. Cottet and S. Mas-Gallic. A particle method to solve the Navier-Stokes system. *Numer. Math.*, 57:805–827, 1990.

[52] G.-H. Cottet, M.-L. Ould Salihi, and M. El Hamraoui. Multi-purpose regridding in vortex methods. In André Giovannini et al., editors, *ESAIM Proc., III Intl. Workshop on Vortex Flows and Related Num. Meth. Toulouse, France, Aug. 24–27, 1998*, volume 7 of *European Series in Applied and Industrial Mathematics*, pages 94–103. Société de Mathématiques Appliquées et Industrielles, 1999.

[53] G.-H. Cottet and P. Poncet. Particle methods for direct numerical simulations of three-dimensional wakes. *J. Turbulence*, 3:038, 2002.

[54] G.-H. Cottet and P. Poncet. Advances in direct numerical simulations of 3D wall-bounded flows by vortex-in-cell methods. *J. Comp. Phys.*, 193(1):136–158, 2004.

[55] B. Couet, O. Buneman, and A. Leonard. Simulation of three-dimensional incompressible flows with a vortex-in-cell method. *J. Comp. Phys.*, 39(2):305–328, 1981.

[56] G. de Vahl Davis. Natural convection of air in a square cavity: a benchmark numerical solution. *Int. J. Num. Meth. in Fluids*, 3:249–264, 1983.

[57] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. Part 1. The case of an isotropic viscosity. *Math. Comp.*, 53:485–507, 1989.

[58] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. Part 2. The anisotropic case. *Math. Comp.*, 53:509–525, 1989.

[59] S. Douady, Y. Couder, and M. E. Brachet. Direct observation of the intermittency of intense vorticity filaments in turbulence. *Phys. Rev. Lett.*, 67:983–986, 1991.

[60] J. Duchon. Fonctions-spline du type plaque mince en dimension 2. Report #231, Université de Grenoble, 1975.

[61] J. Duchon. Fonctions-spline à energie invariante par rotation. Report #27, Université de Grenoble, 1976.

[62] J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *Rev. Française Automat. Informat. Rech. Opér., Anal. Numér.*, 10:5–12, 1976.

[63] N. Dyn, D. Levin, and S. Rippa. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM J. Sci. Stat. Comput.*, 7:639–659, 1986.

[64] A. B. Ebiana and R. W. Bartholomew. Design considerations for numerical filters used in vortex-in-cell algorithms. *Comp. & Fluids*, 25(1):61–75, 1996.

[65] M. El Hamraoui. *Contributions à la Simulation d'Écoulements Tridimensionnels par Méthode de Vortex.* PhD thesis, Université Paul Sabatier, 1999.

[66] J. D. Eldredge, T. Colonius, and A. Leonard. A vortex particle method for compressible flows. AIAA 2001-2641, 2001.

[67] J. D. Eldredge, T. Colonius, and A. Leonard. A vortex particle method for two-dimensional compressible flow. *J. Comp. Phys.*, 179:371–399, 2002.

[68] A. C. Faul, G. Goodsell, and J. J. D. Powell. A Krylov subspace algorithm for multiquadric interpolation in many dimensions. Technical Report DAMTP 2002/NA05, University of Cambridge, April 2002.

[69] A. C. Faul and M. J. D. Powell. Proof of convergence of an iterative technique for thin plate spline interpolation in two dimensions. *Adv. Comp. Math.*, 11(2–3):183–192, 1999.

[70] D. Fishelov. A new vortex scheme for viscous flows. *J. Comp. Phys.*, 86:211–224, 1990.

[71] D. Fishelov. Simulation of three-dimensional turbulent flow in non-cartesian geometry. *J. Comp. Phys.*, 115:249–266, 1994.

[72] M. S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *J. Comp. Appl. Math.*, 73:65–78, 1996.

[73] R. Franke. Scattered data interpolation: Tests of some methods. *Math. Comp.*, 38(157):181–200, 1982.

[74] Yves Gagnon et al., editors. *ESAIM Proceedings, II International Workshop on Vortex Flows and Related Numerical Methods. Montréal, Canada, August 20–24, 1995*, volume 1 of *European Series in Applied and Industrial Mathematics.* Société de Mathématiques Appliqués et Industrielles, 1996.

[75] A. Gharakhani and A. F. Ghoniem. Massively parallel implementation of a 3D vortex-boundary element method. In Gagnon et al. [74], pages 213–223.

[76] A. Gharakhani and A. F. Ghoniem. Three-dimensional vortex simulation of time dependent incompressible internal viscous flows. *J. Comp. Phys.*, 134:75–95, 1997.

[77] J. Goodman. Convergence of the random vortex method. *Comm. Pure Appl. Math.*, 40:189–220, 1987.

[78] A. A. Gossler and J. S. Marshall. Simulation of normal vortex–cylinder interaction in a viscous fluid. *J. Fluid Mech.*, 431:371–405, 2001.

[79] C. Greengard. The core spreading vortex method approximates the wrong equation. *J. Comp. Phys.*, 61:345–348, 1985.

[80] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Phys.*, 73:325–348, 1987.

[81] O. Hald. Convergence of vortex methods for Euler's equations II. *SIAM J. Num. Anal.*, 16:726–755, 1979.

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

[83] O. Hald and V. Mauceri Del Prete. Convergence of vortex methods for Euler's equations. *Math. Comp.*, 32:791–801, 1978.

[84] R. L. Harder and R. N. Desmarais. Interpolation using surface splines. *J. Aircraft*, 9:189–191, 1972.

[85] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.*, 176:1905–1915, 1971.

[86] R. L. Hardy. Research results in the application of multiquadric equations to surveying and mapping problems. *Surveying and Mapping*, 35:321–332, 1975.

[87] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, 49:409–4365, 1952.

[88] T. Y. Hou, J. Lowengrub, and R. Krasny. Convergence of a point vortex method for vortex sheets. *SIAM J. Num. Anal.*, 28(2):308–320, 1991.

[89] S. A. Huyer and J. R. Grant. Incorporation of boundaries for 2D triangular vorticity element methods. In *Proceedings of the Forum on Vortex Methods for Engineering Applications*. Sandia National Labs., 1995. Albuquerque, NM.

[90] S. A. Huyer and J. R. Grant. Computation of incipient separation via solution of the vorticity equation on a Lagrangian mesh. In Gagnon et al. [74], pages 109–123.

[91] A. Iske. Optimal distribution of centers for radial basis function methods. Report TUM-M0004, Technische Universität München, June 2000.

[92] E. J. Kansa. Multiquadrics —A scattered data approximation scheme with applications to computational fluid-dynamics, I. Surface approximations and partial derivative estimates. *Computers Math. Applic.*, 19(8/9):127–145, 1990.

[93] E. J. Kansa. Multiquadrics —A scattered data approximation scheme with applications to computational fluid-dynamics, II. Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers Math. Applic.*, 19(8/9):147–161, 1990.

[94] C. T Kelley. *Iterative Methods for Linear and Non-Linear Equations*. Soc. Ind. Appl. Math., 1995.

[95] R. R. Kerswell. Elliptical instability. *Ann. Rev. Fluid Mech.*, 34:83–113, 2002.

[96] S. Kida. Motion of an elliptic vortex in a uniform shear flow. *J. Phys. Soc. Japan*, 50(10):3517–3520, 1981.

[97] S. Kida and K. Ohkitani. Spatio-temporal intermittency and instability of a forced turbulence. *Phys. Fluids*, A 4(5):1018–1027, 1992.

[98] T. Kida. Theoretical and numerical results of a deterministic two-dimensional vortex method. *Sādhanā*, 23(5-6):419–441, 1998.

[99] T. Kida and T. Nakajima. Core spreading vortex methods in two-dimensional viscous flows. *Comput. Methods Appl. Mech. Engrg.*, 160:273–298, 1998.

[100] T. Kida, T. Nakajima, and H. Suemitsu. Second-order core spreading vortex method in two-dimensional viscous flows. *JSME Intl. J.*, 41 B(2):441–446, 1998.

[101] P. Koumoutsakos. *Simulation of Unsteady Separated Flows Using Vortex Methods.* PhD thesis, California Institute of Technology, 1993.

[102] P. Koumoutsakos. Inviscid axisymmetrization of an elliptical vortex. *J. Comp. Phys.*, 138:821–857, 1997.

[103] P. Koumoutsakos and A. Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech.*, 296:1–38, 1995.

[104] P. Koumoutsakos, A. Leonard, and F. Pépin. Boundary conditions for viscous vortex methods. *J. Comp. Phys.*, 113:52–61, 1994.

[105] P. Koumoutsakos and D. Shiels. Simulations of the viscous flow normal to an impulsively started and uniformly accelerated flat plate. *J. Fluid Mech.*, 328:177–227, 1996.

[106] R. Krasny. *A Numerical Study of Kelvin-Helmholtz Instability by the Point Vortex Method.* PhD thesis, University of California Berkley, 1983.

[107] R. Krasny. Desingularization of periodic vortex sheet roll-up. *J. Comp. Phys.*, 65:292–313, 1986.

[108] R. Krasny. A study of singularity formation in a vortex sheet by the point-vortex approximation. *J. Fluid Mech.*, 167:65–93, 1986.

[109] K. Kuwahara and H. Takami. Numerical studies of two-dimensional vortex motion by a system of points. *J. Phys. Soc. Japan*, 34:247–253, 1973.

[110] M. J. Landman and P. G. Saffman. The three-dimensional instability of strained vortices in a viscous fluid. *Phys. Fluids*, 30(8):2339–2342, 1987.

[111] S. Le Dizès and A. Verga. Viscous interactions of two co-rotating vortices before merging. *J. Fluid Mech.*, 467:389–410, 2002.

[112] B. Legras, P. Santangelo, and R. Benzi. High-resolution numerical experiments for forced two-dimensional turbulence. *Europhys. Lett.*, 5:37–42, 1988.

[113] A. Leonard. Vortex methods for flow simulation. *J. Comp. Phys.*, 37:289–335, 1980.

[114] A. Leonard. Computing three-dimensional incompressible flows with vortex elements. *Ann. Rev. Fluid Mech.*, 17:523–559, 1985.

[115] A. Leonard, D. Shiels, J. K. Salmon, G. S. Winckelmans, and P. Ploumhans. Recent advances in high-resolution vortex methods for incompressible flows. AIAA 97-2108, 1997.

[116] M. J. Lighthill. Introduction: Boundary layer theory. In L. Rosenhead, editor, *Laminar Boundary Layers*, pages 46–113. Oxford University Press, 1963.

[117] P.-L. Lions and S. Mas-Gallic. Une méthode particulaire déterministe pour des équations diffusives non linéaires. *C. R. Acad. Sci. Paris, Série I*, 332:369–376, 2001.

[118] C. H. Liu. A three-dimensional vortex particle-in-cell method for vortex motions in the vicinity of a wall. *Int. J. Num. Meth. in Fluids*, 37:501–523, 2001.

[119] Chung Ho Liu and Denis J. Doorly. Vortex particle-in-cell method for three-dimensional viscous unbounded flow computations. *Int. J. Num. Meth. in Fluids*, 32:29–50, 2000.

[120] D. G. Long. Convergence of the random vortex method in two dimensions. *JAMS*, 1:779–804, 1988.

[121] T. S. Lundgren. Strained spiral vortex model for turbulent fine structures. *Phys. Fluids*, 25(12):2193–2203, 1982.

[122] W. R. Madych and S. A. Nelson. Multivariate interpolation and conditionally positive definite functions. *Constr. Approx.*, 4:77–89, 1988.

[123] W. R. Madych and S. A. Nelson. Multivariate interpolation and conditionally positive definite functions: II. *Math. Comp.*, 54(189):211–230, 1990.

[124] W. R. Madych and S. A. Nelson. Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation. *J. Approx. Theory*, 70:94–114, 1992.

[125] A. J. Majda and A. L. Bertozzi. *Vorticity and Incompressible Flow.* Cambridge University Press, 2002.

[126] J. S. Marshall and J. R. Grant. A Lagrangian collocation method for vorticity transport in viscous fluid flows. In *Proceedings of the Forum on Vortex Methods for Engineering Applications.* Sandia National Labs., 1995. Albuquerque, NM.

[127] J. S. Marshall and J. R. Grant. A Lagrangian vorticity collocation method for viscous, axisymmetric flows with and without swirl. *J. Comp. Phys.*, 138:302–330, 1997.

[128] J. S. Marshall, J. R. Grant, A. A. Gossler, and S. A. Huyer. Vorticity transport on a Lagrangian tetrahedral mesh. *J. Comp. Phys.*, 161:85–113, 2000.

[129] J. C. McWilliams. The emergence of isolated coherent vortices in turbulent flow. *J. Fluid Mech.*, 146:21–43, 1984.

[130] M. V. Melander, J. C. McWilliams, and N. J. Zabusky. Axisymmetrization and vorticity-gradient intensification of an isolated two-dimensional vortex through filamentation. *J. Fluid Mech.*, 178:137–159, 1987.

[131] M. V. Melander, N. J. Zabusky, and J. C. McWilliams. Symmetric vortex merger in two dimensions: causes and conditions. *J. Fluid Mech.*, 195:303–340, 1988.

[132] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constr. Approx.*, 2:11–22, 1986.

[133] F. Milinazzo and P. G. Saffman. The calculation of large Reynolds number two-dimensional flow using discrete vortices with random walk. *J. Comp. Phys.*, 23:380–392, 1977.

[134] H. K. Moffatt, S. Kida, and K. Ohkitani. Stretched vortices – the sinews of turbulence; large-Reynolds-number asymptotics. *J. Fluid Mech.*, 259:241–264, 1994.

[135] J. J. Monaghan. Extrapolating B-splines for interpolation. *J. Comp. Phys.*, 60:253–262, 1985.

[136] D.W. Moore and P. G. Saffman. Density of organized vortices in a turbulent mixing layer. *J. Fluid Mech.*, 69:465–473, 1975.

[137] K. W. Morton. *Numerical Solution of Convection-Diffusion Problems.* Chapman & Hall, 1996.

[138] T. Nakajima and T. Kida. A remark of discrete vortex method (derivation from Navier-Stokes equation). *Trans. Jpn. Soc. Mech. Eng.*, 56 B(531):3284–3291, 1990.

[139] Y. Nakanishi and K. Kamemoto. Numerical simulation of flow around a sphere with vortex blobs. *J. Wind Eng. and Ind. Aero.*, 46-47:363–369, 1993.

[140] F. J. Narcowich, N. Sivakumar, and J. D. Ward. On condition numbers associatied with radial-function interpolation. *J. Math. Anal. Appl.*, 186:457–485, 1994.

[141] F. J. Narcowich and J. D. Ward. Norm of inverses and condition numbers for matrices associated with scattered data. *J. Approx. Theory*, 64:69–94, 1991.

[142] F. J. Narcowich and J. D. Ward. Norm estimates for the inverses of a general class of scattered data radial function interpolation matrices. *J. Approx. Theory*, 69:84–109, 1992.

[143] M. Nitsche and J. H. Strickland. Extension of the gridless vortex method into the compressible flow regime. *J. Turbulence*, 3:050, 2002.

[144] H. Nordmark. Rezoning for higher-order vortex methods. *J. Comp. Phys.*, 97:366–397, 1991.

[145] H. Nordmark. Deterministic high order vortex methods for the 2D Navier-Stokes equation with rezoning. *J. Comp. Phys.*, 129:41–56, 1996.

[146] Y. Ogami and T. Akamatsu. Viscous flow simulation using the discrete vortex model —The diffusion velocity method. *Computers & Fluids*, 19:433–441, 1991.

[147] Y. Ogami and Y. Ayano. Flows around a circular cylinder simulated by the viscous vortex model –The diffusion velocity method. *CFD J.*, 4:383–399, 1995.

[148] A. Ojima and K. Kamemoto. Numerical simulation of unsteady flow around three dimensional bluff bodies by an advanced vortex method. *JSME Int. J.*, 43B:127–135, 2000.

[149] A. Okabe, B. Boots, K. Sugihara, and S. Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Ltd., second edition, 2000.

[150] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, second edition, 1998.

[151] M. L. Ould Salihi. *Couplage de Méthodes Numériques en Simulation Directe d'Écoulements Incompressibles.* PhD thesis, Université Joseph Fourier, 1998.

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

[153] R. D. Pingree and B. Le Cann. Anticyclonic eddy X91 in the southern Bay of Biscay, May 1991 to February 1992. *J. Geophys. Res.*, 97:14353–14367, 1992.

[154] P. Ploumhans. *Simulation of High-Reynolds Nnumber Flows Past Bluff Bodies Using Vortex and Boundary Element Methods.* PhD thesis, Universtité Catholique de Louvain, 2001.

[155] P. Ploumhans and G. S. Winckelmans. Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry. *J. Comp. Phys.*, 165:354–406, 2000.

[156] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Application to the sphere at Re=300, 500 and 1000. *J. Comp. Phys.*, 178:427–463, 2002.

[157] G. J. Poitras and Y. Gagnon. Mixed vortex-finite volume method for the computation of pressure fields in fluid flows. *Int. J. Comp. Fluid Dynamics*, 17(6):487–497, 2003.

[158] P. Poncet. Vanishing of mode B in the wake behind a rotationally oscillating circular cylinder. *Phys. Fluids*, 14(6):2021–2023, 2002.

[159] M. J. D. Powell. A new iterative method for thin plate spline interpolation in two dimensions. *Ann. Num. Math.*, 4:519–527, 1997.

[160] W. Präger. Die Druckverteilung an Körpern in ebener Potential strömung. *Physik. Zeitschr.*, XXIX:865, 1928.

[161] A. Prochazka and D. I. Pullin. Structure and stability of non-symmetric Burgers vortices. *J. Fluid Mech.*, 363:199–228, 1998.

[162] E. G. Puckett. Vortex methods: An introduction and survey of selected research topics. In M. D. Gunzburger and R. A. Nicolaides, editors, *Incompressible Computational Fluid Dynamics: Trends and Advances*, pages 335–408. Cambridge University Press, 1993.

[163] P.-A. Raviart. An analysis of particle methods. In F. Brezzi, editor, *Numerical Methods in Fluid Dynamics, Lecture Notes in Math.* Springer-Verlag: NY, Berlin, 1985.

[164] E. Rivoalen and S. Huberson. Numerical simulation of axisymmetric viscous flows by means of a particle method. *J. Comp. Phys.*, 152:1–31, 1999.

[165] E. Rivoalen, S. Huberson, and F. Hauville. Simulation numérique des équations de Navier-Stokes 3D par une méthode particulaire. *C. R. Acad. Sci. Paris, Série IIb*, 324:543–549, 1997.

[166] P. J. Roache. Quantification of uncertainty in computational fluid dynamics. *Ann. Rev. Fluid Mech.*, 29:123–160, 1997.

[167] A. C. Robinson and P. G. Saffman. Stability and structure of stretched vortices. *Stud. Appl. Math.*, 70:163–181, 1984.

[168] D. Roddier, S. W. Liao, and R. W. Yeung. Wave-induced motion of floating cylinders fitted with bilge keels. *Int. J. Offshore and Polar Engrg.*, 10(4):241–248, 2000.

[169] L. Rosenhead. The formation of vortices from a surface of discontinuity. *Proc. R. Soc. Lond. A*, A134:170–192, 1931.

[170] L. F. Rossi. Resurrecting core spreading vortex methods: A new scheme that is both deterministic and convergent. *SIAM J. Sci. Comput.*, 17:370–397, 1996.

[171] L. F. Rossi. Merging computational elements in vortex simulations. *SIAM J. Sci. Comput.*, 18:1014–1027, 1997.

[172] L. F. Rossi. High order vortex methods with deforming elliptical Gaussian blobs 1: Derivation and validation. Unpublished preprint, 2001.

[173] L. F. Rossi. High order vortex methods with deforming elliptical Gaussian blobs 2: Merging. Unpublished preprint, 2001.

[174] L. F. Rossi, J. F. Lingevitch, and A. J. Bernoff. Quasi-steady monopole and tripole attractors for relaxing vortices. *Phys. Fluids*, 9(8):2329–2338, 1997.

[175] J. W. Rottman, J. E. Simpson, and P. K. Stansby. The motion of a cylinder of fluid released from rest in a cross flow. *J. Fluid Mech.*, 177:307–337, 1987.

[176] G. Russo. A deterministic vortex method for the Navier-Stokes equations. *J. Comp. Phys.*, 108:84–94, 1993.

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

[178] Fortune S. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[179] Y. Saad and M. H. Schultz. GMRES: Generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[180] P. G. Saffman. *Vortex Dynamics.* Cambridge University Press, 1992.

[181] J. K. Salmon and M. S. Warren. Skeletons from the treecode closet. *J. Comp. Phys.*, 111:136–155, 1994.

[182] J. K. Salmon, M. S. Warren, and G. S. Winckelmans. Fast parallel tree codes for gravitational and fluid dynamical N-body problems. *Intl. J. Supercomput. Appl. High Perf. Comp.*, 8(2):129–142, 1994.

[183] T. Sarpkaya. Computational methods with vortices. *J. Fluids Eng.*, 11:5–52, 1989.

[184] R. Schaback. Comparison of radial basis function interpolants. In K Jetter and F. Utreras, editors, *Multivariate Approximation. From CAGD to Wavelets*, pages 293–305. World Scientific, London, 1993.

[185] R. Schaback. Lower bounds for norms of inverses of interpolation matrices for radial basis functions. *J. Approx. Theory*, 79:287–306, 1994.

[186] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Adv. Comput. Math.*, 3:251–264, 1995.

[187] R. Schaback. Improved error bounds for scattered data interpolation by radial basis functions. *Math. Comp.*, 68(225):201–216, 1999.

[188] R. Schaback and H. Wendland. Characterization and construction of radial basis functions. In N. Dyn, D. Leviatan, D. Levin, and A. Pinkus, editors, *Multivariate Approximation and Applications*, pages 1–24. Cambridge University Press, 2001.

[189] R. K. M. Seah and R. W. Yeung. Sway and roll hydrodynamics of cylindrical sections. *Int. J. Offshore and Polar Engrg.*, 13(3):241–248, 2003.

[190] J. A. Sethian, J. P. Brunet, A. Greenberg, and J. P. Mesirov. A parallel implementation of the random vortex method. *Comp. Phys. Comm.*, 65:231–237, 1991.

[191] M. I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science*, pages 151–162, 1975.

[192] S. Shankar and L. van Dommelen. A new diffusion procedure for vortex methods. *J. Comp. Phys.*, 127:88–109, 1996.

[193] D. Shiels. *Simulation of Controlled Bluff Body Flow with a Viscous Vortex Method.* PhD thesis, California Institute of Technology, 1998.

[194] D. Shiels and A. Leonard. Investigation of a drag reduction on a circular cylinder in rotary oscillation. *J. Fluid Mech.*, 431:297–322, 2001.

[195] P. A. Smith and P. K. Stansby. Generalized discrete vortex method for cylinders without sharp edges. *AIAA J.*, 25:199–200, 1987.

[196] P. A. Smith and P. K. Stansby. Impulsively started flow around a circular-cylinder by the vortex method. *J. Fluid Mech.*, 194:45–77, 1988.

[197] P. R. Spalart. Vortex methods for separated flows. Technical Memorandum 100068, NASA, 1988.

[198] G. Strang. On the construction and comparison of difference schemes. *SIAM J. Num. Anal.*, 5:506–517, 1968.

[199] J. H. Strickland, S. N. Kempka, and W. P. Wolfe. Viscous diffusion of vorticity in unsteady wall layers using the diffusion velocity concept. In *Proceedings of the Forum on Vortex Methods for Engineering Applications.* Sandia National Labs., 1995. Albuquerque, NM.

[200] J. H. Strickland, S. N. Kempka, and W. P. Wolfe. Viscous diffusion of vorticity using the diffusion velocity concept. In Gagnon et al. [74], pages 135–151.

[201] A. A. Townsend. On the fine-scale structure of turbulence. *Proc. R. Soc. Lond. A*, 208:534–542, 1951.

[202] G. J. F. van Heijst and R. C. Kloosterziel. Tripolar vortices in a rotating fluid. *Nature*, 338:569–571, 1989.

[203] G. J. F. van Heijst, R. C. Kloosterzierl, and C. W. M. Williams. Laboratory experiments on the tripolar vortex in a rotating fluid. *J. Fluid Mech.*, 225:301–331, 1991.

[204] A. Vincent and M. Meneguzzi. The spatial structure and statistical properties of homogeneous turbulence. *J. Fluid Mech.*, 225:1–25, 1991.

[205] Z.-M. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA J. Numer. Anal.*, 13:13–27, 1993.

[206] L. A. Ying. Convergence study of viscous splitting in bounded domains. *Lecture Notes in Math.*, 1297:184–202, 19857.

[207] L. A. Ying. Viscous splitting for the unbounded problem of the Navier-Stokes equations. *Math. Comp.*, 55:89–113, 1990.