Spectral Method for the Vortex Tripole

by

Helmut Wahanik Duran

Diary

of

Research

SCAT Project

Supervisor: Lorena Barba

University of Bristol
Bristol, UK
December, 2006

Contents

Preface

This report resumes the research undertaken at University of Bristol-UK, on the period September-December 2006. This visit is being sponsored and has been organized by the *SCAT* Project: "an international collaboration initiative for the advancement of scientific computing in Europe and Latin America".

Acknowledgements

I owe special gratitude to my supervisor Dr. Lorena Barba, for her enthusiasm and patience, and to Dr. André Nachbin: a person who has given me wonderful counselling during my studies at *IMPA*, and who introduced me to the fantastic field of Fluid Dynamics. I want to thank Boris Drappier for being so helpful and cheerful always, and for making so many arrangements for me when it was so necessary!

The *SCAT* Project is being made possible due to the immense dedication of its members and sponsors.

Chapter 1

Introduction

Our main objective consists of developing a spectral method for studying the evolution of 2-D turbulent structures. We aim to obtain results similar to the ones appearing in previous studies ([1],[2] and [8]) where cases of non-axisymmetrization were found to evolve when a 2-D monopole is added a sufficiently strong quadrupolar perturbation (this case is called *vortex tripole* or *tripolar vortex*).
A useful way of restating the equations for 2-D incompressible flow is obtained by introducing the streamfunction ψ and the vorticity ω. After appropriate rearrangements (see Chorin *et al.* [5] Ch 1.2) of the 2-D Navier-Stokes Equations, the Vorticity-Streamfunction formulation is obtained: throughout this report (Chapter 2) we state a Fourier method for solving it in its discretized (AB/BDI2) representation.
In Chapter 3 we describe the implementation of the spectral method within MATLAB's specifications; additionally we show in detail how to evaluate the Fourier coefficients of Non-linear Terms, and we highlight initialization tips.
Chapter 4 contains the results of the computations of the *tripolar vortex* in a regime specially adapted for the application of the spectral technique: as the framework has been transformed, extra adjustments must be applied for obtaining our desired *Output*.
In Chapter 5 the reader can find the codes used in our computations.

Chapter 2

Spectral method for the Vorticity-Streamfunction Equations

The advantages of using the Vorticity-Streamfunction equations include an "automatic" divergence-free velocity field which gives us an easy-framework of less equations (for notation convenience we will refer to these as the (ω, ψ)-Eqs).
Spectral Fourier methods can be applied straightforward for solving Navier-Stokes equations in this presentation, which appears useful for the simulation of 2-D 2π-Periodic Turbulent flow.
The equations for each mode are discretized with respect to time with the semi-implicit scheme AB/BDI2, and Nonlinear Terms are calculated by means of the pseudospectral technique *free of aliasing* (obtained by the "3/2" rule). We associate a first-order starting scheme for the initialization.

2.1 Formulation

The 2-D Vorticity-Streamfunction equations are:

$$\partial_t \omega + B(\vec{u}, \omega) = \frac{1}{Re}\Delta\omega \tag{2.1}$$

$$\omega = -\Delta\psi \tag{2.2}$$

Where $\vec{u} = (u, v)$ and:

$$\begin{aligned} u &= \partial_y \psi \\ v &= -\partial_x \psi \end{aligned} \qquad B(\vec{u}, \omega) := \vec{u} \cdot \nabla\omega$$

Assuming 2-D 2π-periodicity in the initial conditions, the solution (ω, ψ) has then the same period and is sought in truncated Fourier series representation of the form:

$$\phi(x,y,t) = \frac{1}{4\pi^2} \sum_{-K \le k_1,k_2 \le K} \hat{\phi}_{k_1,k_2}(t) e^{i\,(k_1 x + k_2 y)} \tag{2.3}$$

where ϕ represents either ω or ψ. (For notation convenience we will usually write $\mathbf{k} := k_1, k_2$, $\mathbf{x} := x, y$ and $k_1 x + k_2 y = \mathbf{k} \cdot \mathbf{x}$). Equation (2.3) is discretized with respect to time with the semi-implicit scheme AB/BDI2. This one is deduced from the general two step scheme for the time discretization of the (ω, ψ)-Eqs:

$$\begin{aligned} \frac{(1+\varepsilon)\omega^{n+1} - 2\varepsilon\omega^n - (1-\varepsilon)\omega^{n-1}}{2\Delta t} + \Big[\gamma_1 B^{n+1} + \gamma_2 B^n + (1-\gamma_1-\gamma_2)B^{n-1})\Big] \\ = \frac{1}{Re}\Delta\Big[\theta_1\omega^{n+1} + \theta_2\omega^n + (1-\theta_1-\theta_2)\omega^{n-1}\Big] \end{aligned} \tag{2.4}$$

where ω^n and B^n denote approximate values for ω and $B(\vec{u}\,, \psi)$ at time $n\Delta t$.
Second-order accuracy requires

$$\varepsilon/2 = 2\gamma_1 + \gamma_2 - 1 = 2\theta_1 + \theta_2 - 1$$

(Consistency and order of accuracy are determined by Taylor's expansions).

In particular the semi-implicit Adams-bashfort/Backward Differentiation 2^{nd}-order scheme is obtained with $\varepsilon = 2$, $\gamma_1 = 0$, $\gamma_2 = 2$, $\theta_1 = 1$, $\theta_2 = 0$. After replacing this values into (2.4) and applying the Galerkin technique we obtain, for each value of $\mathbf{k}$,

$$\frac{3\hat{\omega}_{\mathbf{k}}^n - 4\hat{\omega}_{\mathbf{k}}^n + \hat{\omega}_{\mathbf{k}}^{n-1}}{2\Delta t} + \Big[2\hat{B}_{\mathbf{k}}^{n+1} - \hat{B}_{\mathbf{k}}^{n-1}\Big] = \frac{1}{Re} k^2 \hat{\omega}_{\mathbf{k}}^{n+1} \tag{2.5}$$

$$k^2 \hat{\psi}_{\mathbf{k}}^{n+1} = \hat{\omega}_{\mathbf{k}}^{n+1} \tag{2.6}$$

where $k^2 := |\mathbf{k}|^2 = k_1^2 + k_2^2$. Equation (2.6) is obtained evaluating equation (2.2) at time $(n+1)\Delta t$.
In the next section we will see how to calculate $\hat{B}_{\mathbf{k}}^n$, the Fourier coefficient of $\vec{u} \cdot \nabla\omega$ by means of the pseudospectral technique *free of aliasing*, using the values of $\hat{\omega}_{\mathbf{k}}^n$ and $\hat{\psi}_{\mathbf{k}}^n$.

The Fourier coefficients $\left(\hat{u}_{\mathbf{k}}^{n+1}, \hat{v}_{\mathbf{k}}^{n+1}\right)$ of the velocity are,

$$\hat{u}_{\mathbf{p}}^{n+1} = \underline{i}\, p_2\, \hat{\psi}_{\mathbf{p}}^{n+1} \tag{2.7}$$

$$\hat{v}_{\mathbf{p}}^{n+1} = -\underline{i}\, p_1\, \hat{\psi}_{\mathbf{p}}^{n+1} \tag{2.8}$$

and for the gradient of the vorticity,

$$\partial_x \hat{\omega}_{\mathbf{q}}^{n+1} = \underline{i}\, q_1\, \hat{\omega}_{\mathbf{q}}^{n+1} \tag{2.9}$$

$$\partial_y \hat{\omega}_{\mathbf{q}}^{n+1} = \underline{i}\, q_2\, \hat{\omega}_{\mathbf{q}}^{n+1} \tag{2.10}$$

Given initial conditions $\omega(0)$ and $\psi(0)$ we put,

$$\hat{\omega}_{\mathbf{k}}^0 := \hat{\omega}_{\mathbf{k}}(0)$$
$$\hat{\psi}_{\mathbf{k}}^0 := \hat{\psi}_{\mathbf{k}}(0)$$

2.1.1 Starting Scheme

The starting scheme can be one-order accuracy less than the general scheme, who in this case is of 2^{nd}-order: in fact, it has been shown (Gear,1971) that the approximation error $O(\Delta t^k)$ becomes $O(\Delta t^{k+1})$ when $t = m\Delta t$, with $m \lll 1$. Consequently, it is sufficient to associate a 1^{st}-order starting scheme for equation (2.5).
Suppose that we know $\hat{\omega}_{\mathbf{k}}^0$ and $\hat{\psi}_{\mathbf{k}}^0$ (we will see in the next section how to obtain $\hat{B}_{\mathbf{k}}^0$ from this values). Now observe that when $n = 0$ we are in lack the values of $\hat{\omega}_{\mathbf{k}}^{-1}$ and $\hat{\psi}_{\mathbf{k}}^{-1}$ in order to find $\hat{\omega}_{\mathbf{k}}^1$ and $\hat{\psi}_{\mathbf{k}}^1$ for moving to the next time-step. For this reason we pose $\hat{\omega}_{\mathbf{k}}^{-1} \equiv \hat{\omega}_{\mathbf{k}}^0$ and $\hat{\psi}_{\mathbf{k}}^{-1} \equiv \hat{\psi}_{\mathbf{k}}^0$ (which makes $\hat{B}_{\mathbf{k}}^{-1} \equiv \hat{B}_{\mathbf{k}}^0$).

We have then, for each $\mathbf{k}$,

$$\frac{\hat{\omega}_{\mathbf{k}}^1 - \hat{\omega}_{\mathbf{k}}^0}{\frac{2}{3}\Delta t} + \hat{B}_{\mathbf{k}}^0 = \frac{1}{Re} k^2 \hat{\omega}_{\mathbf{k}}^1 \tag{2.11}$$

$$k^2 \hat{\psi}_{\mathbf{k}}^1 = \hat{\omega}_{\mathbf{k}}^1 \tag{2.12}$$

Chapter 3

Implementation

Some inconveniences arise when implementing the method for the (ω, ψ)-Eqs in the computer.
Changing -for good theoretical reasons- a little bit discretization (2.3), we will satisfy MATLAB's specifications for storing wavenumbers when using the FFT2 algorithm.
Instead of evaluating $\hat{B}^n_{\mathbf{k}}$ through convolution sums (which would be too expensive!), we invoke the power of the FFT2 algorithm to move back and forth from Fourier to Physical space, performing operations in the second rather than the first. Although in this process "alias" terms appear; this can happily be fixed through the elegant and quick application of the "3/2" rule when using this *pseudospectral technique*.

3.1 The FFT2 and iFFT2 Algorithms

Our basic periodic grid will be a discrete subset of $[0, 2\pi] \times [0, 2\pi]$. Specifically we use equally spaced grid points:

$$\mathbf{x}_\iota := (x_i, y_j) = \left(\frac{2\pi i}{N}, \frac{2\pi j}{N} \right)$$

where $i, j \in 1, \ldots, N$ and $N \equiv 0 \,(\mathrm{mod}\, 4)$. The interspacing is $h := \frac{2\pi}{N}$.
Expansion (2.3) must satisfy, for fixed t,

$$\phi(\mathbf{x}_\iota, t) = \frac{1}{4\pi^2} \sum_{-K \leq \mathbf{k} \leq K} \hat{\phi}_{\mathbf{k}}(t) e^{\frac{i}{-} \mathbf{k} \cdot \mathbf{x}_\iota} \tag{3.1}$$

The coefficients $\hat{\phi}_{\mathbf{k}}(t)$ are determined using the orthogonality relation,

$$\frac{1}{4\pi^2} \sum_{\iota=1}^{N} e^{\frac{2\pi i}{N} (\mathbf{k}-\mathbf{l}) \cdot \iota} = \begin{cases} \frac{1}{h^2} & \text{if } \mathbf{k} - \mathbf{l} = mN,\ m \in \mathbb{Z}, \\ 0 & \text{in the other cases.} \end{cases} \tag{OR}$$

where $\mathbf{l} := (l_1, l_2)$, the greek letter ι stands for the pair i, j, and the expression $\mathbf{k} - \mathbf{l} = mN$ stands for the equality $\mathbf{k} - \mathbf{l} = mN(1,1)$ (we will often abuse of this notation!).
Multiplying (3.1) by $e^{-\underline{i}\,\mathbf{l}\cdot\mathbf{x}_\iota}$ and adding from $\iota = 1, \ldots, N$ we obtain,

$$\hat{\phi}_{\mathbf{k}}(t) = h^2 \sum_{\iota=1}^{N} \phi(\mathbf{x}_\iota, t) e^{-\underline{i}\,\mathbf{k}\cdot\mathbf{x}_\iota} \quad \text{for } -K \leq \mathbf{k} \leq K \tag{3.2}$$

Observe that if $\mathbf{k}_\alpha - \mathbf{k}_\beta \equiv 0 \pmod{\frac{2\pi}{h}}$ then

$$e^{\underline{i}\,\mathbf{k}_\alpha\cdot\mathbf{x}_\iota} = e^{\underline{i}\,\mathbf{k}_\beta\cdot\mathbf{x}_\iota} e^{2\pi\underline{i}\,\delta\cdot\lambda} = e^{\underline{i}\,\mathbf{k}_\beta\cdot\mathbf{x}_\iota}$$

where $\delta, \lambda \in \mathbb{Z}^2$. In words, this means that wavenumbers differing by an integer multiple of $2\pi/h$ are equal on the grid. We conclude that it is senseless to use values of $\mathbf{k}$ outside of an interval of size $2\pi/h$. Specifically we imply $\mathbf{k} \in \left(-\frac{\pi}{h}, \frac{\pi}{h}\right]^2$ *i.e.*,

$$k_1, k_2 \in \{-N/2+1, -N/2+2, \ldots, 0, 1, \ldots, N/2\}$$

Therefore from now on we will have $K = N/2$ and the range of the wavenumbers in the expressions (2.3), (3.1) and (3.2) will be modified. These three equations will be replaced respectively by,

$$\phi(\mathbf{x}, t) = \frac{1}{4\pi^2} \sum_{-K+1 \leq \mathbf{k} \leq K} \hat{\phi}_{\mathbf{k}}(t) e^{\underline{i}\,\mathbf{k}\cdot\mathbf{x}} \tag{TFS}$$

$$\phi(\mathbf{x}_\iota, t) = \frac{1}{4\pi^2} \sum_{-K+1 \leq \mathbf{k} \leq K} \hat{\phi}_{\mathbf{k}}(t) e^{\underline{i}\,\mathbf{k}\cdot\mathbf{x}_\iota} \tag{iDFT2}$$

$$\hat{\phi}_{\mathbf{k}}(t) = h^2 \sum_{\iota=1}^{N} \phi(\mathbf{x}_\iota, t) e^{-\underline{i}\,\mathbf{k}\cdot\mathbf{x}_\iota} \quad \text{for } -K+1 \leq \mathbf{k} \leq K \tag{DFT2}$$

In doing this we are standarizing our problem for MATLAB's specifications: expressions iDFT2 and DFT2 are calculated via the iFFT2 and FFT2 algorithms (when N is a product of small prime factors these algorithms require $O(N \log N)$ operations). MATLAB's FFT2 algorithm receives as an entry an $N \times N$ matrix of real numbers (A) representing a plotting grid, and returns an $N \times N$ matrix of Fourier coefficients $\widehat{A} := \{\hat{a}_{k_1,k_2}\}$, where k_1 and k_2 are stored in the order $0, 1, \ldots, N/2, -N/2+1, \ldots, -1$. The iFFT2 just works in the opposite way.
Observe that some trouble arises with the modes $\hat{\phi}_{K,k_2}$ and $\hat{\phi}_{K,k_2}$. In fact the terms,

$$\hat{\phi}_{K,k_2} e^{\underline{i}\,(Kx+k_2y)}$$

$$\hat{\phi}_{k_1,K} e^{\underline{i}\,(k_1x+Ky)}$$

cannot be associated with their analog

$$\hat{\phi}_{-K,k_2} e^{\frac{i}{}\,(-Kx+k_2y)}$$

$$\hat{\phi}_{k_1,-K} e^{\frac{i}{}\,(k_1x-Ky)}$$

This leads to instabilities in the time-dependent problem. For handling it, we just set these terms equal to 0 whenever they appear during the implementation of the FFT2 algorithm.

3.2 Calculation of Nonlinear Terms

In this section we show how to efficiently evaluate the nonlinear term $\hat{B}^n_{\mathbf{k}}$ through the application of the ***P**seudospectral **T**echnique **F**ree of **A**liasing.* This technique was developed independently by Orszag (1969, 1970, 1971) and Eliasen, Machenhauer and Rasmussen (1970).

3.2.1 The Pseudospectral Technique

Let

$$\hat{B}^n_{\mathbf{k}} := \widehat{\left[u\partial_x\omega + v\partial_y\omega\right]}^n_{\mathbf{k}} = \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ -K+1\le \mathbf{p},\mathbf{q}\le K}} \hat{u}^n_{\mathbf{p}}\widehat{\partial_x\omega}^n_{\mathbf{p}} + \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ -K+1\le \mathbf{p},\mathbf{q}\le K}} \hat{v}^n_{\mathbf{p}}\widehat{\partial_y\omega}^n_{\mathbf{q}} \tag{3.3}$$

for $-K+1 \le \mathbf{k} \le K$. A direct calculation of (3.3) takes $O(N^3)$ operations. In the absence of another tool to perform this computation, spectral methods would be too expensive given the fact that calculation of 2-D non-linear terms through finite difference algorithms take $O(N^2)$ operations. However, the pesudospectral technique requires $O(N^2 \log N)$ operations (although some corrections have to be applied for its application in order to remove the *aliasing* error).
We will illustrate the technique's application on a generic 2-D convolution sum of the type,

$$\hat{c}_{\mathbf{k}} := \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ -K+1\le \mathbf{p},\mathbf{q}\le K}} \hat{a}_{\mathbf{p}}\hat{b}_{\mathbf{q}} \quad \text{for } -K+1 \le \mathbf{k} \le K \tag{3.4}$$

Consider the matrices $\hat{c} := \{\hat{c}_{\mathbf{k}}\}$, $\hat{a} := \{\hat{a}_{\mathbf{p}}\}$ and $\hat{b} := \{\hat{b}_{\mathbf{q}}\}$, organized within MATLAB's specifications for the application of the FFT2 algorithm.
We use the iFFT2 algorithm to transform $\hat{a}$ and $\hat{b}$ to the 2-D plotting grid (*i.e.* the physical space) perform a "multiplication" and then applying the FFT2 we determine $\widehat{C}$, an approximate value for matrix $\hat{c}$.

Algorithm 3.2.1.

i) Start with calculating $a := \text{iFFT2}\,(\hat{a})$ and $b := \text{iFFT2}(\hat{b})$.

ii) Let $C := a. * b$ (element-wise multiplication).

iii) Finish with $\widehat{C} := \text{FFT2}(C)$.

Due to the *aliasing* phenomenon, matrix $\widehat{C}$ is different from matrix $\hat{c}$. In fact, using (OR) we have,

$$\begin{aligned}
\widehat{C}_{\mathbf{k}} &= h^2 \sum_{\iota=1}^{N} c(\mathbf{x}_\iota) e^{-\underline{i}\,\mathbf{k}\cdot\mathbf{x}_\iota} \\
&= h^2 \sum_{\iota=1}^{N} \left(\sum_{-K+1\le \mathbf{p} \le K} \hat{a}_{\mathbf{p}} e^{\underline{i}\,\mathbf{p}\cdot\mathbf{x}_\iota} \right) \left(\sum_{-K+1\le \mathbf{q} \le K} \hat{b}_{\mathbf{q}} e^{\underline{i}\,\mathbf{q}\cdot\mathbf{x}_\iota} \right) e^{-\underline{i}\,\mathbf{k}\cdot\mathbf{x}_\iota} \\
&= h^2 \sum_{\iota=1}^{N} \sum_{-K+1\le \mathbf{p},\mathbf{q} \le K} \hat{a}_{\mathbf{p}} \hat{b}_{\mathbf{q}} e^{\underline{i}\,(\mathbf{p}+\mathbf{q}-\mathbf{k})\cdot\mathbf{x}_\iota} \\
&= h^2 \sum_{-K+1\le \mathbf{p},\mathbf{q} \le K} \hat{a}_{\mathbf{p}} \hat{b}_{\mathbf{q}} \sum_{\iota=1}^{N} e^{\underline{i}\,(\mathbf{p}+\mathbf{q}-\mathbf{k})\cdot\mathbf{x}_\iota} \\
&= h^2 \sum_{-K+1\le \mathbf{p},\mathbf{q} \le K} \hat{a}_{\mathbf{p}} \hat{b}_{\mathbf{q}} \sum_{\iota=1}^{N} e^{\frac{2\pi i}{N}\,(\mathbf{p}+\mathbf{q}-\mathbf{k})\cdot\iota} \\
&= \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ -K+1\le \mathbf{p},\mathbf{q} \le K}} \hat{a}_{\mathbf{p}} \hat{b}_{\mathbf{q}} + \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k}\pm N \\ -K+1\le \mathbf{p},\mathbf{q} \le K}} \hat{a}_{\mathbf{p}} \hat{b}_{\mathbf{q}} \\
&= \hat{c}_{\mathbf{k}} + \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k}\pm N \\ -K+1\le \mathbf{p},\mathbf{q} \le K}} \hat{a}_{\mathbf{p}} \hat{b}_{\mathbf{q}} \quad \text{for } -K+1 \le \mathbf{k} \le K
\end{aligned} \tag{3.5}$$

Now we can clearly see the aliasing error! We conclude that evaluation of the convolution $\widehat{C}$ through Algorithm (3.2.1) is not at all correct. Modifying it we will eliminate the *alias* terms achieving our final goal!

3.2.2 The "3/2" Rule

Aliasing removal is performed using the "3/2" rule. Orszag (1971) was the first to consider this computationally inexpensive method: its principle consists of *extending* matrices $\hat{a}$ and $\hat{b}$ before applying 3.2.1, and then *extract* a submatrix of the *Output* who would be our desired convolution sum $\hat{c}$. The details can be found in the following,

Algorithm 3.2.2 (*PTFA*)**.**

i) Let $K' := 3K/2$.

ii) Extend matrices $\hat{a}, \hat{b}$ into $\hat{a}', \hat{b}'$ following the rule:

$$\hat{a}'_{\mathbf{p}} = \begin{cases} \hat{a}_{p_1,\,p_2} & \text{if } -K+1 \le p_1, p_2 \le K, \\ 0 & \text{if } -K'+1 \le p_r \le K, \text{ or } K < p_r \le K' \text{ for some } r \in \{1,2\} \end{cases}$$

$$\hat{b}'_{\mathbf{q}} = \begin{cases} \hat{b}_{q_1,\,q_2} & \text{if } -K+1 \le q_1, q_2 \le K, \\ 0 & \text{if } -K'+1 \le q_s \le K, \text{ or } K < q_s \le K' \text{ for some } s \in \{1,2\} \end{cases}$$

iii) Compute $a' := \text{iFFT2}\,(\hat{a}')$ and $b' := \text{iFFT2}(\hat{b}')$.

iv) Let $C' := a'.*\,b'$.

v) Find $\widehat{C}' := \text{FFT2}(C')$. The result is a square matrix of size $2K'$.

vi) Extract the submatrix $\hat{c}$ of size $2K$, from $\widehat{C}'$, following the rule:

$$\hat{c}_{\mathbf{k}} = \left\{ \widehat{C}'_{k_1,\,k_2} \quad \text{if } -K+1 \le k_1, k_2 \le K \right.$$

Proposition 3.2.3. *Matrix $\hat{c}$ corresponds to the desired convolution sum (3.4).*

Proof. Define $N' := 2K'$.
In a similar way as we obtained (3.5), we find:

$$\widehat{C}'_{\mathbf{k}} = \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ -K'+1 \le \mathbf{p},\mathbf{q} \le K'}} \hat{a}'_{\mathbf{p}}\hat{b}'_{\mathbf{q}} + \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k}\pm N' \\ -K'+1 \le \mathbf{p},\mathbf{q} \le K'}} \hat{a}'_{\mathbf{p}}\hat{b}'_{\mathbf{q}} \quad \text{for } -K+1 \le \mathbf{k} \le K \tag{3.6}$$

Step ii) from Algorithm *PTFA* indicates how (3.6) can be replaced by:

$$\widehat{C}'_{\mathbf{k}} = \hat{c}_{\mathbf{k}} + \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k}+N' \\ -K+1 \le \mathbf{p},\mathbf{q} \le K}} \hat{a}'_{\mathbf{p}}\hat{b}'_{\mathbf{q}} + \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k}-N' \\ -K+1 \le \mathbf{p},\mathbf{q} \le K}} \hat{a}'_{\mathbf{p}}\hat{b}'_{\mathbf{q}} \quad \text{for } -K+1 \le \mathbf{k} \le K \tag{3.7}$$

The *alias* terms in equation (3.7) are 0. In fact,

$$\begin{aligned} \mathbf{q} &= 2K' + \mathbf{k} - \mathbf{p} = 3K + \mathbf{k} - \mathbf{p} \\ &> 3K - K + 1 - K \\ &> K \end{aligned}$$

in the second sum. In the third one we have:

$$\begin{aligned} \mathbf{q} &= -2K' + \mathbf{k} - \mathbf{p} = -3K + \mathbf{k} - \mathbf{p} \\ &< -3K + K + K - 1 = -K - 1 \\ &< -K + 1 \end{aligned}$$

□

3.3 Initialization

Discretization (2.11) requires the values of:

$$\widehat{B}^0 := \left\{ \widehat{B}^0_{\mathbf{k}} \right\}_{-K+1 \leq \mathbf{k} \leq K}$$

For finding them we need the values of:

$$\hat{\omega}^0 := \left\{ \hat{\omega}^0_{\mathbf{k}} \right\}_{-K+1 \leq \mathbf{k} \leq K} \qquad \hat{\psi}^0 := \left\{ \hat{\psi}^0_{\mathbf{k}} \right\}_{-K+1 \leq \mathbf{k} \leq K}$$

The value of $\hat{\omega}^0$ is found by computing FFT2(ω^0), where $\omega^0 := \left\{ \omega^0(\mathbf{x}_\iota) \right\}_{\iota=1,\ldots,N}$
Furthermore, using relation (2.12), $\hat{\psi}^0_{\mathbf{k}}$ can be found for every $\mathbf{k}$ except for $\mathbf{k} = 0$. Fortunatelly we do not require the missing Fourier coefficient! In fact, in order to find $\widehat{B}^0$ we will apply Algorithm *PTFA* using the matrices,

$$\hat{u}^0 := \left\{ \underline{i}\, p_2\, \hat{\psi}^0_{\mathbf{p}} \right\}_{-K+1 \leq \mathbf{p} \leq K} \qquad \hat{v}^0 := \left\{ -\underline{i}\, p_1\, \hat{\psi}^0_{\mathbf{p}} \right\}_{-K+1 \leq \mathbf{p} \leq K}$$

with the first column (*respectively* row) of null entries.
As we can observe in discretization (2.6) the issue concerning the missing Fourier coefficient of ψ will repeat for the next time steps as well. As above, we do not require the missing term for computing $\widehat{B}^n := \left\{ \widehat{B}^n_{\mathbf{k}} \right\}_{-K+1 \leq \mathbf{k} \leq K}$

Chapter 4

Computations of a Perturbed Monopole with a Tripole Attractor

In this chapter we use the method for the Vorticity-Streamfunction equations for simulating the evolution of the vortex tripole. In this particular case the *Output* gives evidence of how evolution of 2-D turbulence does not always approach axisymmetrization (which was initially proposed as one of the universal processes of relaxation). In fact, the presence of non-linear interactions may be an important cause of re-organization into other ensembles, such as the tripolar vortex.
This situation has already been studied by Barba [1] [2] using mesh-less vortex methods, and by Rossi *et al.* [8] by means of viscous methods based on core spreading and vortex particle splitting: is was highlighted how under small-amplitude non-axisymmetric perturbations, the flow relaxes to an axisymmetric state, but for sufficiently large amplitudes it reorganizes into a quasi-steady, rotating vortex tripole.
For the implementation of the Fourier method we must *rescale* and *shift* the initial conditions in order to satisfy the periodicity assumption made. Furthermore, this changes will force us to change all the framework!

4.1 Rescaling and Shifting

Initial conditions for previous studies have been composed of a Gaussian vortex with a superposed quadrupolar perturbation $\omega_0(\mathbf{x}) + \omega'(\mathbf{x})$, with

$$\omega_0(\mathbf{x}) = \frac{1}{4\pi} e^{-|\mathbf{x}|^2/4} \tag{4.1}$$

$$\omega'(\mathbf{x}) = \frac{\delta}{4\pi} e^{-|\mathbf{x}|^2/4} \cos m\theta \tag{4.2}$$

where ω_0 and ω' are the base and perturbation vorticity respectively and $\theta := \arg(\mathbf{x})$.

Barba [1],[2] and Rossi [8] performed numerical simulations with different values for the parameters: taking $\delta = 0.25$ and $Re = 10^4$, where $Re = \Gamma/\nu$ (total circulation divided by the viscosity), the initial state re-organizes to a quasi-steady structure composed of one core of positive vorticity, and two satellites of negative vorticity
Observe that the method for the (ω, ψ)-Eqs as described in Chapter 3 is sought for a 2π periodic flow defined on $[0, 2\pi] \times [0, 2\pi]$. The method can also be applied to flows who satisfy the no-slip condition, *i.e.* such that the velocity of the particles approaches 0 in the boundary. For satisfying this we must *rescale* and *shift* initial conditions (4.1) and (4.2). This is easily made through a change of variables:

$$\mathbf{x} \Longrightarrow \varphi(\mathbf{x}) := C(\mathbf{x} - \mathbf{x}_0) \tag{4.3}$$

where $C \geq 3$. The initial conditions are then replaced by $\omega_0(\varphi(\mathbf{x}))$ and $\omega'(\varphi(\mathbf{x}))$.

4.2 A New Framework

The process of adaptation described in Section 4.1 affects the initial conditions of the vortex tripole. In fact, whereas *shifting* would not be an important factor in the evolutionary feature, *rescalling* reduces the *size* of our framework. Therefore, for expecting similar outcomes as the ones depicted in [1],[2] and [8], we must apply some adjustments: we use $C\omega_0(\varphi(\mathbf{x}))$ and $C\omega'(\varphi(\mathbf{x}))$, increasing by a factor of C the initial vorticity.
A simulation was performed with values of $\delta = 0.5$ and $Re = 10^4$. The method for the (ω, ψ)-Eqs was implemented with 2-D plotting grids of 10^6 points, *i.e.* with $N = 10^3$. The time-step used was $\Delta t = 0.1$ and $\Delta t_1 = 10^{-8}$ for the initial time-step.
As observed in the cited previous works, the perturbed monopole relaxes to a non-axisymmetric state. A simultaneous process of counter-clockwise *rotation* of the negative "blobs" and *shearing* of the positive core takes place. It is surprising how the satellites are almost intact after the evolution occurs!

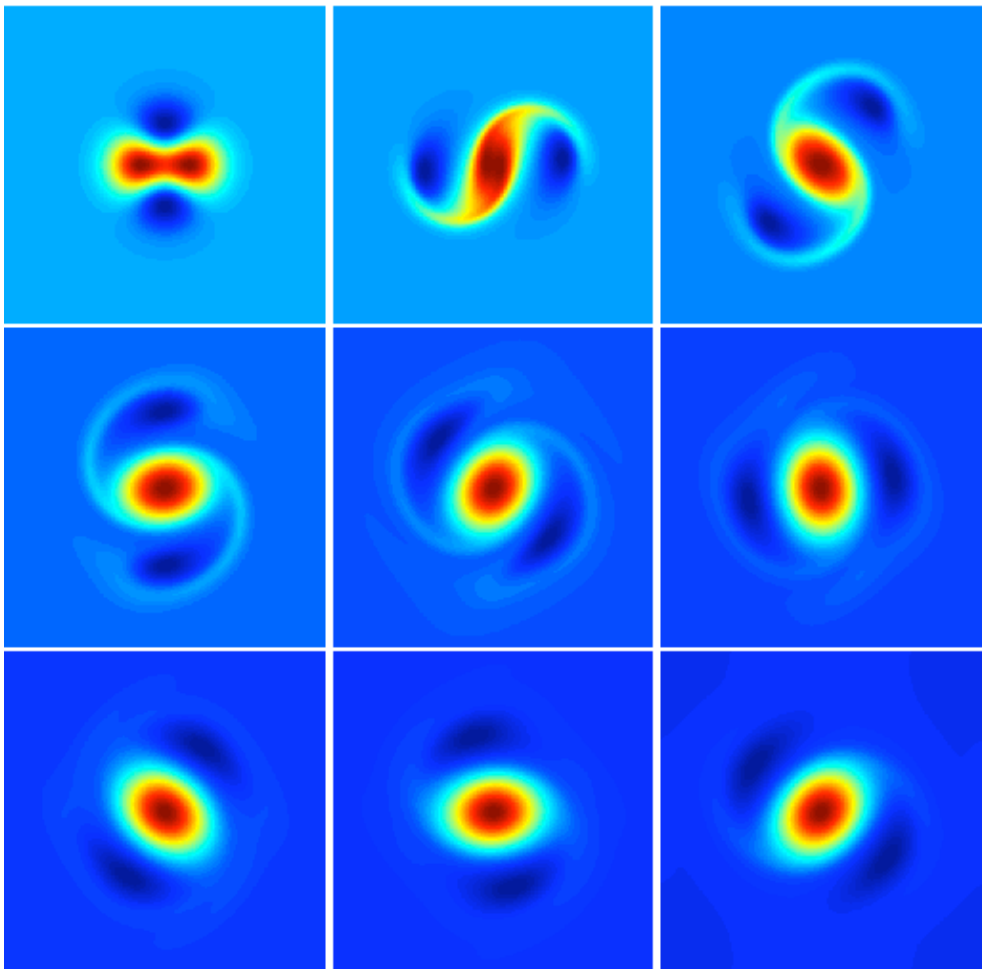

Figure 1. Perturbed monopole relaxing to a tripole attractor. The strength of the perturbation is $\delta = 0.5$ and the Reynolds number is $Re = 10^4$.
Here we observe the sequence $t = 0, 100, \ldots, 800$. Color-ranges for the vorticity start from black (negative), blue (zero), to intense red (positive).

Chapter 5

Programs

The following codes have been implemented in MATLAB: the main one computes the starting scheme described in Section 2.1.1 using initial conditions (4.1) and (4.2) with the arrangements described in Section 4.2; additionally it evaluates the subsequent steps of discretization (2.5) and plots the sequential "frames" of the evolution.
A code for calculating non-linear terms through the *PTFA* was written; it uses two sub-functions for *extending* and *extracting* spectra.

5.1 Main Code

```
%%%%% SPECTRAL METHOD FOR THE VORTEX TRIPOLE
%-------------------------------------------------------------------------
%%%%% ASSIGNING THE GRID AND THE STARTING CONDITIONS FOR THE
%%%%% VORTEX TRIPOLE

clear all;
clf ;

% Functions
Nonlinear = @PTFA;

% Parameter Space
    C = 3;
delta = 0.5;
   dt = 0.1 ;
    t = 800 ; n = t/dt ;  % time-intervals
    N = 1000; % N=0(mod 4)
    Re = 1e4;
```

```
    K = N/2;
    m = 2;

% Helping Matrices
L1 = repmat( [ 0:K -K+1:-1 ] , 2*K , 1 ) ;
normksq = L1.^2  +  L1'.^2 ;

% Little Filter for normksq
normksq2 = normksq ;
normksq2(1,1) = 1 ;

% Grid and initial conditions
[X,Y] = meshgrid( 2*pi/N : 2*pi/N : 2*pi  ,  2*pi : -2*pi/N : 2*pi/N );
[THETA,RHO] = cart2pol( C*( X-pi ) , C*( Y-pi ) ); % reescaling the grid

Omega_ini = ( 1 / ( 4*pi ) )*exp( -( RHO.^2 )./ (4) );
Omega_pert = ( delta )*Omega_ini.*( RHO.^2 ).*cos( m*THETA );
Omega_0 =  C*( Omega_ini  +  Omega_pert ) ;

% Plotting the initial condition
subplot(3,3,1)
surf( X, Y, Omega_0, 'EdgeColor', 'none', 'FaceLighting', 'phong' );
view( [0 90] )
axis off, axis square
%-------------------------------------------------------------------------

%%%%% STEP 1

Omegac_0 = fft2( Omega_0 ) ;

% K+1 mode filters
Omegac_0( K+1 , : ) = 0 ;
Omegac_0( : , K+1 ) = 0 ;

% Initial Stream Function's transform
Psic_0 = ( Omegac_0 ) ./ normksq2 ;

% Initial nonlinear term: matrix of 2D Fourier Components of
% -psi_y*w_x + psi_x*w_y ;
```

```
BNLc_0 = Nonlinear( 1i*L1.*Psic_0 , 1i*L1'.*Omegac_0 , K ) +
Nonlinear( -1i*L1'.*Psic_0 , 1i*L1.*Omegac_0 , K ) ;

h = 1e-8 ;  % h<<1 for the initial step
dt_0 = (2/3)*h*dt ;

% Evaluation
Omegac_1 =  ( Omegac_0-dt_0*(BNLc_0) ) ./ ( 1 + (dt_0/Re)*normksq ) ;
Psic_1 = Omegac_1./ normksq2 ;
%--------------------------------------------------------------------------

%%% SUBSEQUENT STEPS

Omegac_nback = Omegac_0 ;
Psic_nback = Psic_0 ;
BNLc_nback = BNLc_0 ;
Omegac_n = Omegac_1 ;
Psic_n = Psic_1 ;

for j = 2:n
    % Nonlinear terms
    BNLc_n = Nonlinear( 1i*L1.*Psic_n , 1i*L1'.*Omegac_n , K )  +
    Nonlinear( -1i*L1'.*Psic_n 1i*L1.*Omegac_n , K ) ;

    % Evaluation
    Omegac_nfront = ( 4*Omegac_n - Omegac_nback - 2*dt*( 2*BNLc_n - BNLc_nback ) )
    ./ ( 3 + ( 2*dt/Re )*normksq ) ;
    Psic_nfront   = ( Omegac_nfront )./ normksq2 ;

    % Assignation for the next step
    Omegac_nback = Omegac_n ;
    Psic_nback = Psic_n ;
    BNLc_nback = BNLc_n  ;
    Omegac_n = Omegac_nfront ;
    Psic_n   = Psic_nfront ;

    % Plotting Sequentially
    if mod( j*dt,100 ) == 0
```

```
      subplot( 3 , 3 , ( ( j*dt )/100 ) + 1 ,'v6')
      Omega_fin = real( ifft2(Omegac_n) );
      surf( X, Y, Omega_fin, 'EdgeColor', 'none', 'FaceLighting', 'phong' );
      view( [0 90] )
      axis off, axis square
    end
end
%%% END OF PROGRAM
%-------------------------------------------------------------------------
```

5.2 *PTFA*

```
% Function for calculating the PseudoSpectral Technique free
% of Aliasing (using the "3/2" rule)

function Cc = PTFA(Ac, Bc, K)

         Kp = 3*K/2; grow=@GROW; shrink=@SHRINK;

         % 1.Extend the matrices Ac and Bc into the matrices Acp, Bcp
         Acp = grow(Ac, K); Bcp = grow(Bc, K);

         % 2.Calculate a(xp_i, yp_j), b(xp_i, yp_j) i,j 1,....,2Kp=Np
         Ap = ifft2(Acp);  Bp = ifft2(Bcp);

         % 3. Multiply Ap and Bp pointwise
         Cp = Ap.*Bp;

         % 4. Calculate the Fourier Coefficients
         Cvp = fft2(Cp);

         % 5. Shrink for obtaining the desired Fourier Coefficients
         %    free of aliasing
         Cc = shrink(Cvp, K);

         % 6. Apply Filters
         Cc(K+1,:) = 0 ;
         Cc(:,K+1) = 0 ;
```

5.2.1 *Extending* with *GROW*

```
% Function for Expanding Xc into Xcp

function Xcp = GROW(Xc,K)

        Kp = 3*K/2;

        % Copying
        X11 = Xc(1:K+1 , 1:K+1);
        X12 = Xc(1:K+1 , K+2:2*K);
        X21 = Xc(K+2:2*K , 1:K+1);
        X22 = Xc(K+2:2*K , K+2:2*K);

        % Extending
        H1 = zeros(K+1, 2*(Kp-K)) ; H2 = zeros(2*(Kp-K),2*Kp);
        H3 = zeros(K-1, 2*(Kp-K)) ;
        Xcp = [X11 H1 X12; H2; X21 H3 X22];
```

5.2.2 *Extracting* with *SHRINK*

```
% Function for shrinking Xcp into Xc

function Xc = SHRINK(Xcp,K)

        Kp = 3*K/2;

        % Copying
        X11 = Xcp(1:K+1 , 1:K+1);
        X12 = Xcp(1:K+1 , 2*Kp-K+2:2*Kp);
        X21 = Xcp(2*Kp-K+2:2*Kp , 1:K+1);
        X22 = Xcp(2*Kp-K+2:2*Kp, 2*Kp-K+2:2*Kp);

        % Extracting
        Xc = [X11 X12; X21 X22];
```

Bibliography

[1] L.A. Barba, *Vortex method for computing high Reynolds number flows: increased accuracy with a fully mesh-less formulation*, PhD Thesis, Caltech, 2004.

[2] L.A. Barba, *Nonshielded multipolar vortices at high Reynolds number*, Physical Review E **73**, 065303, The American Physical Society, 2006.

[3] G.K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press 1967.

[4] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.

[5] A. Chorin, J.E. Marsden, *A Mathematical Introduction to Fluid Dynamics*, Universitext, Springer-Verlag, New York, 1979,1990,1994.

[6] Desmond J. Higham, Nicholas J. Higham, *Matlab Guide*, SIAM, 2005.

[7] Roger Peyret, *Spectral Methods for Incompressible Viscous Flow*, Springer-Verlag, New York, 2002.

[8] Louis F. Rossi, Joseph F. Lingevitch, Andrew J. Bernoff, *Quasi-steady monopole and tripole attractors for relaxing vortices*, Phys. Fluids 9 (8), American Institute of Physics, August 1997.

[9] Lloyd N.Trefethen, *Spectral Methods in Matlab*, SIAM, 2000.