CITY UNIVERSITY OF NEW YORK

Averaged dynamics of the advection-diffusion equation and applications to ocean flows.

by
Yauheni Dzedzits

A dissertation submitted to the Graduate Faculty in Physics in partial fulfillment of the requirements for the degree of Doctor of Philosophy

2012

This manuscript has been read and accepted for the Graduate Faculty in Physics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Prof. Tobias Schäfer, College of Staten Island, CUNY

Date _____ _____
Chair Of Examining Committee

Prof. Igor L. Kuskovsky

Date _____ _____
Executive Officer

_____
Prof. Andrew Poje, College of Staten Island, CUNY

_____
Prof. Sultan Catto, Baruch College, CUNY

_____
Prof. Paula Whitlock, Brooklyn College, CUNY

_____
Prof. Peter Gordon, New Jersey Institute of Technology

Abstract

AVERAGED DYNAMICS OF THE ADVECTION-DIFFUSION EQUATION AND APPLICATIONS TO OCEAN FLOWS

by Yauheni Dzedzits
Adviser: Professor Tobias Schäfer

This dissertation presents some aspects of an advection-diffusion equation and its applications to physical oceanography. We propose a perturbative scheme of averaging the advection-diffusion equation in the limit of vanishing diffusivity. Under the restriction that the time-dependence of the advective field is completely separable we construct an exact solution of the purely advective part via action-angle coordinates and treat diffusion as a perturbation using Lie transform techniques. The developed method is applied to a regularized vortical flow field which is periodically modulated in time. Numerical simulations of the vortical flow advection in presence of small diffusion are discussed. We present numerical evidence that the spectrum of of the averaged time-independent advection-diffusion operator converges to the spectrum of the operator with fully enabled time dynamics. A formal generalization of the method for three-dimensional time-periodic flows is discussed.

We also discuss the importance of advection and diffusion in problems of transport and mixing in complicated dynamical systems, such as hydrodynamical systems, in particular describing ocean currents. We propose a method to visualize and analyze the structure of complex flows using data from HYbrid Coordinate Ocean Model (HYCOM) as an example. We present results of simulations obtained with highly parallel Co-array Fortran code that can be run on modern computing systems that support partitioned global address space (PGAS) programming model.

Acknowledgements

I acknowledge with deep and sincere gratitude all the help, support and encouragement of my advisor Professor Tobias Schäfer. Working with Professor Schäfer was a great pleasure as for he is a bright scientist always full of ideas and ready to share them. I am honored to be one of his first graduate students and thank him for his patience throughout this endeavor. Above all I thank him for being an inexhaustible fountain of optimism. His sincere smile and positive thinking always fill me with enthusiasm.

I thank Professor Andrew Poje for the great amount of time and effort he spent teaching me fluid dynamics, both theoretical and computational aspects of it. I appreciate his attitude that is always spiced with a little bit of irony and sarcasm. He taught me one important philosophical concept — do not philosophize about the problem, instead solve it here and now. I am grateful to Professor Poje for introducing me to the world of Matlab — great piece of software that allows to obtain non-trivial results with just a few lines of code.

I express my deepest gratitude to Dr. Michael Kress for his support and for providing me with opportunity to join the CUNY High Performance Center and participate in its growth and development. It surely was an extremely interesting and fascinating experience.

I am indebted to Professor Anatoly Kuklov and Professor William Schreiber from the department of Engineering Science and Physics. A. Kuklov is undoubtedly a talented

physicist who taught me passion and enthusiasm in research and set an example of persistence and commitment. I am thankful to Professor Schreiber for helping me to squeeze through the dense forest of bureaucratic procedures.

I specially thank the director of the CUNY Hight Performance Computing Center Paul Muzio for his support. His intelligence and broad spectrum of interests always was a great source of discussions and debates. Physics, philosophy, politics or music — whatever we discuss over a cup of coffee or glass of wine — Paul would turn it into an intellectual exercise that I always found entertaining and yet useful. In particular, I want to thank Paul for teaching me to love opera, fine italian cooking, and of course the ocean.

I am thankful to Dr. Robert Numrich for his help with Co-array Fortran and for discussions that are much appreciated.

Most of the computations I performed at the CUNY HPC Center, which would not be possible without excellent staff members Richard Walsh and Nikolaos Trikoupis. I am glad to have such a great friends not only because I can always rely on their help but because they are very intelligent, bright and interesting people and it always is a pleasure to be around them.

Many thanks to my friend Dr. Pavel Ivanushkin. He hosted me when I was a newbie lost in the jungle of New York City skyscrapers and I will always appreciate our friendship.

I want to thank Liudmila Zagusta and Alexander Gorbatsievitch. L. Zagusta was my first teacher of physics and I am thankful to her for helping me to choose the right direction in

life. Dr. A. Gorbatsievitch from Belarusian State University will always be an example of an ideal researcher, lecturer, teacher that I will follow in my career.

Finally, I express my warmest thanks to my parents, Liudmila and Yury Dzedzits and my sister Kristina. Despite the large distance they were always nearby. And, of course, I cannot overstate my gratitude to my dear wife Tatiana, who was my shadow in the desert and my light in the dark night. I owe her billions of things “Yes, after my thesis, dear”. Their love, care and support, their understanding and patience is what kept me afloat all these years.

Contents

List of Figures

List of Tables

Chapter 1

Averaged dynamics of advection-diffusion equation with time-periodic advection field.

Advective evolution of passive tracers is a problem of great practical importance in many applications. Problems of transport induced by fluid advection emerge in engineering, astrophysics, plasma physics, turbulence and, of course, ocean/atmospheric science. Real systems are usually subjected to diffusive forces of different nature. It is not surprising that understanding of the advective dynamics of passive scalars in the presence of diffusion has been a subject of intensive research reaching back to at least as far as Batchelor [1].

The processes of advective and diffusive motion are described by a linear differential equation, however the complete analytic description of the dynamics is still problematic even

for smooth planar flows. Tracer trajectories in the real systems of interest (plasma flows, oceanic/atmospheric currents, etc...) usually are turbulent with velocity fluctuations in time and space. In cases when the length scales of these turbulent excitations are small in comparison with a typical length scales of advective forces (l_v) rigorous homogenization techniques can be applied [2, 3] which leads to renormalization of the diffusion. For large turbulent variations of the velocity field, reaching the characteristic lengths of both advection l_v and domain size L, a different approach is required.

In this chapter we discuss a time-periodic advection field with fully separable time-dependence $u(t, x, y) = \bar{u}(x, y)f(t)$ in the so-called Batchelor regime, when the advective spacial scale is assumed to be much larger then the diffusive length scale: $l_v \gg l_d$. Additionally the velocity field is suggested to be time-periodic and mean-free. The particular form of the time dependence of the advective term allows the original equation to be rewritten in action-angle coordinates. By applying a Lie transform we derive an approximate averaged equation with time-independent coefficients, thus dramatically simplifying the original problem. The technique we use was first developed for the finite-dimensional problem [4] and then extentedned to systems with infinite number of degrees of freedom [5, 6].

We first provide a general outline of the transformation that rewrites the original advection-diffusion in stream-lines coordinates. A Lie transform is then used to average the equation. We apply the developed methods to a particular vector field, namely, a time-periodic regularized vortex. Numerical simulations of this vortex field using Chebyshev methods are

presented to support the analytic results. At the end of the chapter, a generalization for a three dimensional case is discussed.

1.1 Introduction

We start discussion of the advection-diffusion equation by postulating, following the phenomenological first Fick's law [7], that the flux of the diffusing substance is proportional to the local density gradient (we write a one-dimensional formulation for simplicity):

$$j_x = -\kappa(x)\frac{\partial c(x,t)}{\partial x} \tag{1.1}$$

which is simply the statement that the substance tends to spread from the region with high concentration to regions with small concentration. In the above equation $c(x,t)$ is a local density, which is a function of time and posotion, and $\kappa(x)$ is the local diffusivity. Following the derivation in [8], consider a small volume $\delta V = \delta x \delta y \delta z$ shown in Fig. 1.1.

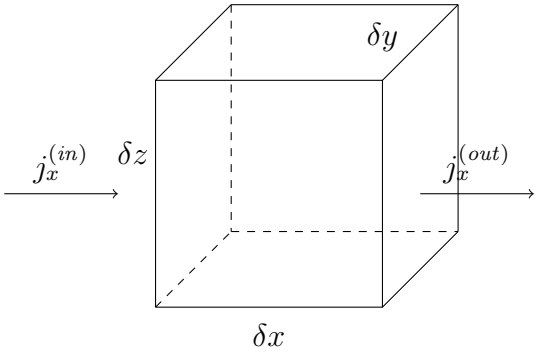


FIGURE 1.1: Fickian diffusion.

Flow of the mass through this volume is given by the difference between the incoming and outgoing fluxes

$$\begin{aligned}\frac{\partial(\delta m)}{\partial t} = \delta y \delta z (j_x^{(in)} - j_x^{(out)}) = -\kappa \delta y \delta z \left(\frac{\partial c}{\partial x}|_{x=x} - \frac{\partial c}{\partial x}|_{x=x+\delta x} \right) = \\ \text{using Taylor expansion} \\ = \kappa \delta x \delta y \delta z \frac{\partial^2 c}{\partial x^2}\end{aligned} \tag{1.2}$$

Keeping in mind that $\delta m = c\delta V = c\delta x \delta y \delta z$, we immediately obtain that, generalizing for the three-dimensional system,

$$\frac{\partial c}{\partial t} = \kappa \nabla^2 c \tag{1.3}$$

The last equation is, probably, the most famous (and the simplest) equation of fluid dynamics, known as the *diffusion* or *heat* equation.

In a very similar manner we can consider flux through the volume δV, that occurs not only due to the diffusion, but simultaneously with advection. In this situation total flux is a combination of the advection and diffusion-induced Fickian flux (1.1) and is given by

$$j_x = uc - \kappa(x) \frac{\partial c(x,t)}{\partial x}, \tag{1.4}$$

where u is the component of the velocity $\vec{V} = \{u, v, w\}$. Again, considering the flux through a small volume along the x-direction and computing a change of mass inside the volume we come to the following equation:

$$c_t + \nabla \cdot (\vec{V} c) = \kappa \nabla^2 c. \tag{1.5}$$

This equation is known as the advection-diffusion equation since evolution of the scalar

c is dictated by both of these effects. In the case of an incompressible flows $\nabla \cdot \vec{V} = 0$ this equation can be simplified since $\nabla \cdot (\vec{V}c) = (\nabla \cdot \vec{V})c + \vec{V} \cdot \nabla c$ and incompressibility implies that

$$c_t + \vec{V} \cdot \nabla c = \kappa \Delta c. \tag{1.6}$$

We are going to make certain assumptions about the time dependence of the velocity field and develop a method that handles periodic flows under small diffusive perturbations.

1.2 Stream-lines coordinates.

We start off with the advection-diffusion equation in the form

$$c_t + (u \cdot \nabla)c = \kappa \Delta c, \tag{1.7}$$

where the scalar c and the velocity vector field u are functions of spatial coordinates (x, y) and time t. This is an initial value problem with given initial state of the scalar:

$$c(0, x, y) = c_0(x, y) \tag{1.8}$$

We consider a divergence-free incompressible velocity field which means that there exists a stream function Ψ such that:

$$u(t, x, y) = \nabla \times \Psi(t, x, y), \tag{1.9}$$

where cross in the above expression denotes two-dimensional curl: $\nabla \times = (\partial_y, -\partial_x)$. We further assume that the time dependence of the stream function Ψ is completely separable

so that

$$\Psi(t,x,y) = H(x,y)f(t), \tag{1.10}$$

where $f(t)$ is a mean-free periodic function of time with period T:

$$\langle f \rangle = \frac{1}{T}\int_0^T f(t) = 0. \tag{1.11}$$

In non-dimensional variables, equation (1.7) takes the form

$$c_t + \frac{1}{St}(u \cdot \nabla)c = \epsilon \Delta c. \tag{1.12}$$

Here $St = L/UT$ is a Strouhal number – the ratio of an advective time-scale to a forcing period [9], and $\epsilon = T\kappa/L^2 \ll 1$, the ratio of the forcing period to diffusive time-scale $T_d = L^2/\kappa$. From now on we will assume $St = \mathcal{O}(1)$ and for simplicity use $u = u/St$.

To proceed we introduce a function F such that $\dot{F} = f(t)$ or, in other words

$$F(t) = \int_0^t f(t')dt' \tag{1.13}$$

and write the tracer coordinates (x, y) as functions of F. Differentiating (x, y) with respect to F we obtain

$$\begin{aligned} \frac{dx}{dF} &= \frac{dx}{dt}\cdot\frac{dt}{dF} = \frac{\dot{x}}{f(t)} \\ \frac{dy}{dF} &= \frac{dy}{dt}\cdot\frac{dt}{dF} = \frac{\dot{y}}{f(t)} \end{aligned} \tag{1.14}$$

and therefore

$$u = (\dot{x}, \dot{y}) = \nabla \times \Psi = (\nabla \times H)f(t) = (\frac{dx}{dF}, \frac{dy}{dF})f(t) \tag{1.15}$$

leads to

$$\frac{d(x,y)}{dF} = \nabla \times H. \tag{1.16}$$

The latter can be now viewed as a Hamiltonian system where F plays the role of time:

$$\begin{aligned} \frac{dx}{dF} &= \frac{\partial H(x,y)}{\partial y} \\ \frac{dy}{dF} &= -\frac{\partial H(x,y)}{\partial x} \end{aligned} \tag{1.17}$$

It is well know from classical mechanics that such an autonomous integrable Hamiltonian system allows for a canonical transformation to action-angle variables [10]

$$\mathcal{C} : (x,y) \to (J,\theta). \tag{1.18}$$

In these coordinates, the evolution of the system (1.17) can be written in extremely simple form

$$\begin{aligned} J &= J_0, \\ \theta &= \theta_0 - \omega(J)F(t), \end{aligned} \tag{1.19}$$

where $\omega(J)$ is some function of action coordinate and depends on the particular choice of the stream function H. The original advection-diffusion equation (1.7) in these coordinates is written as

$$c_t - f(t)\omega(J)c_\theta = \epsilon(\Gamma : \nabla\nabla + \delta \cdot \nabla)c. \tag{1.20}$$

Here

$$\begin{aligned} \Gamma : \nabla\nabla &= \Gamma_{11}\partial_{\theta\theta} + \Gamma_{12}\partial_{\theta J} + \Gamma_{21}\partial_{J\theta} + \Gamma_{22}\partial_{JJ}, \\ \delta \cdot \nabla &= \delta_1\partial_\theta + \delta_2\partial_J. \end{aligned} \tag{1.21}$$

We can now use stream-lines $\bar{J} = J$ and $\bar{\theta} = \theta + \omega(J)F(T)$ as new coordinates via the transformation

$$c(t, J, \theta) \longrightarrow v(t, \bar{J}, \bar{\theta}) \tag{1.22}$$

with the following transformation rules:

$$\begin{aligned} c_t &= v_t + v_{\bar{\theta}}\omega f \\ c_\theta &= v_{\bar{\theta}}, \quad c_{\theta\theta} = v_{\bar{\theta}\bar{\theta}} \\ c_J &= v_J + v_{\bar{\theta}}\omega' F \\ c_{JJ} &= v_{JJ} + v_{\bar{\theta}}\omega'' F + 2v_{J\bar{\theta}}\omega' F + v_{\bar{\theta}\bar{\theta}}(\omega' F)^2. \end{aligned} \tag{1.23}$$

This transformation to stream-lines coordinates is nothing but a transformation to a new "co-moving" reference frame. In these coordinates, the advective term in equation (1.20) disappears and we finally obtain an equation for $\bar{c}$ of the form

$$v_\tau = (\bar{\Gamma} : \nabla\nabla + \bar{\delta} \cdot \nabla)v. \tag{1.24}$$

Here the rescaled time $\tau = \epsilon t$ is used. All effects of the influence of the advective field are now contained in the time-dependent coefficients $\bar{\Gamma}$ and $\bar{\delta}$ and, therefore, equation (1.24) is now suitable for averaging.

1.3 Lie transform averaging.

In order to average equation (1.24) we explore the idea of applying a near-identity Lie transform that eliminates the explicit time dependence of the coefficients in the equation

for

$$v_\tau = X(v, \tau), \tag{1.25}$$

and instead leads to an equation with time-independent coefficients of the form

$$V_\tau = Y(v). \tag{1.26}$$

This is achieved by transform of the type

$$v = \exp(\phi \cdot \nabla_L) V \tag{1.27}$$

where the operator $\phi \cdot \nabla_L$ is chosen to eliminate the time dependence in (1.25).

Since the operators X and Y depend on v and all its spatial derivatives, the operator $\phi \cdot \nabla$ can be defined as

$$\phi \cdot \nabla_L = \sum_{n,m} \phi_{nx,my} \frac{\partial^{(n+m)}}{\partial V_{nx,my}}, \tag{1.28}$$

where $\phi_{nx,my} = \partial^{(n+m)}\phi / \partial x^n \partial y^m$ and $V_{nx,my} = \partial^{(n+m)} V / \partial x^n \partial y^m$. The subscript L on ∇_L distinguishes this operator from the usual ∇. The generating function ϕ also depends on V and all its derivatives. The idea is to hide all explicit time dependence of the equation for V in the generating function ϕ, which therefore will also be periodic in time. The general transformation rule that uses (1.27) to transform (1.25) to (1.26) is [11]

$$Y \cdot \nabla_L + \Big(\frac{\partial}{\partial \tau} e^{\phi \cdot \nabla_L} \Big) e^{-\phi \cdot \nabla_L} = e^{\phi \cdot \nabla_L} (X \cdot \nabla_L) e^{-\phi \cdot \nabla_L}. \tag{1.29}$$

Both terms can be conveniently expanded using the Campbell-Baker-Hausdorff formulae [12, 13]

$$\left(\frac{\partial}{\partial\tau}e^{\phi\cdot\nabla_L}\right) = \left(\phi_\tau + \frac{1}{2!}[\phi,\phi_\tau]_L + \frac{1}{3!}[\phi,[\phi,\phi_\tau]_L]_L + ...\right)\cdot\nabla_L, \tag{1.30}$$

$$e^{\phi\cdot\nabla_L}(X\cdot\nabla_L)e^{-\phi\cdot\nabla_L} = \left(X + [\phi,X]_L + \frac{1}{2!}[\phi,[\phi,X]_L]_L + ...\right)\cdot\nabla_L \tag{1.31}$$

where the Lie commutator is defined through $[A,B]_L = (A\cdot\nabla_L)B - (B\cdot\nabla_L)A$. We now expand both Y and ϕ in a series in a small parameter ϵ as

$$Y = Y_0 + \epsilon Y_1 + \epsilon^2 Y_2 + ..., \quad \phi = \epsilon\phi_1 + \epsilon^2\phi_2 + ... \tag{1.32}$$

where $\mathcal{O}(Y_n) = \mathcal{O}(\phi_n) = \epsilon^n$ and differentiation by τ lowers the order of ϕ_n by one

$$\mathcal{O}(\frac{\partial\phi_n}{\partial\tau}) = \epsilon^{n-1} \tag{1.33}$$

Using (1.32) we can rewrite (1.30) and (1.31) just for the first few orders in ϵ

$$\begin{aligned}\left(\frac{\partial}{\partial\tau}e^{\phi\cdot\nabla_L}\right)\approx \Bigg(&\phi_{1\tau} + \epsilon\left\{\phi_{2\tau} + \frac{1}{2!}[\phi_1,\phi_{1\tau}]_L\right\} + \\ &\epsilon^2\left\{\phi_{3\tau} + \frac{1}{2!}([\phi_2,\phi_{1\tau}]_L + [\phi_1,\phi_{2\tau}]_L) + \frac{1}{3!}[\phi_1,[\phi_1,\phi_{1\tau}]_L]_L\right\}\Bigg)\cdot\nabla_L,\end{aligned} \tag{1.34}$$

$$\begin{aligned}e^{\phi\cdot\nabla_L}(X\cdot\nabla_L)e^{-\phi\cdot\nabla_L} \approx \Bigg(&X + \epsilon[\phi_1,X]_L + \\ &\epsilon^2\left\{[\phi_2,X]_L + \frac{1}{2!}[\phi_1,[\phi_1,X]_L]_L\right\}\Bigg)\cdot\nabla_L\end{aligned} \tag{1.35}$$

Here $\phi_{n\tau} = \partial\phi_n/\partial\tau$. These expansions can now be used to solve (1.29) order by order. We will explicitly write down the first three orders in ϵ

$$\epsilon^0: \quad Y_0 + \phi_{1\tau} = X \tag{1.36}$$

$$\epsilon^1: \quad Y_1 + \phi_{2\tau} + \frac{1}{2}[\phi_1, \phi_{1\tau}]_L = [\phi_1, X]_L \tag{1.37}$$

$$\epsilon^2: \quad Y_2 + \phi_{3\tau} + \frac{1}{2}\Big([\phi_2, \phi_{1\tau}]_L + [\phi_1, \phi_{2\tau}]_L\Big) + \frac{1}{6}[\phi_1, [\phi_1, \phi_{1\tau}]_L]_L =$$

$$[\phi_2, X]_L + \frac{1}{2}[\phi_1, [\phi_1, X]_L]_L \tag{1.38}$$

...

Averaging of the first order equation (1.36) immediately yields the result $Y_0 = \langle X \rangle$ since ϕ_1 is periodic and a zero-mean function of time. Since the original advection-diffusion equation was written in a form

$$v_\tau = \bar{L}v, \quad \bar{L} = \bar{\Gamma} : \nabla\nabla + \bar{\delta} \cdot \nabla, \tag{1.39}$$

the averaged version at leading order becomes

$$V_\tau = \left\langle \bar{L} \right\rangle V \tag{1.40}$$

where

$$\left\langle \bar{L} \right\rangle = \left\langle \bar{\Gamma} \right\rangle : \nabla\nabla + \left\langle \bar{\delta} \right\rangle \cdot \nabla. \tag{1.41}$$

As follows from (1.40), for the first order in ϵ, the time-dependent coefficients of the full equation are simply replaced by their time averages. At the same order in ϵ, the

generating function ϕ_1 is given by

$$\phi_1 = L_1 V \equiv \left(\int_0^\tau \bar{L} - \left\langle \bar{L} \right\rangle \right) V. \tag{1.42}$$

Introducing Γ_1 and δ_1 as

$$\frac{d\Gamma_1}{d\tau} = \bar{\Gamma} - \left\langle \bar{\Gamma} \right\rangle, \quad \frac{d\delta_1}{d\tau} = \bar{\delta} - \left\langle \bar{\delta} \right\rangle \tag{1.43}$$

we can write L_1 explicitly as

$$L_1 = \Gamma_1 : \nabla\nabla + \delta_1 \cdot \nabla. \tag{1.44}$$

The second-order correction may be derived using the next term in the expansions of the Campbell-Baker-Hausdorff formulae, namely the expression (1.37). Using (1.40) and (1.42) we find for the next order for Y

$$Y_1 = \frac{1}{2} \left\langle [L_1 V, L_{1\tau} V]_l \right\rangle + [\langle L_1 \rangle V, \left\langle \bar{L} \right\rangle V]_l \tag{1.45}$$

where the last equality follows from the definition of the Lie commutator and integration by parts. Finally, collecting first and second orders, we arrive at the averaged equation for V in the form

$$V_\tau = \left(\left\langle \bar{L} \right\rangle + \epsilon \Big(\left\langle \bar{L} L_1 \right\rangle - \langle L_1 \rangle \left\langle \bar{L} \right\rangle \Big) \right) V. \tag{1.46}$$

1.4 Regularized vortical flow field.

In order to study this theory and check it against numerical simulations, we consider a regularized vortical flow field with a stream function given by

$$H(t,x,y) = \ln\Big(\sqrt{a^2+x^2+y^2}\Big) f(t) = \Psi(x,y) f(t) \tag{1.47}$$

Introducing $F(t)$ as in (1.13), we derive that the stream-lines equations are given by

$$\begin{aligned} \frac{dx}{dF} &= \frac{\partial \Psi(x,y)}{\partial y} = \frac{y}{r^2+a^2} = \omega(r) y, \\ \frac{dy}{dF} &= -\frac{\partial \Psi(x,y)}{\partial x} = -\frac{x}{r^2+a^2} = -\omega(r) x, \end{aligned} \tag{1.48}$$

where the following notation was introduced

$$\omega(r) = \frac{1}{a^2+r^2}. \tag{1.49}$$

Since $r^2 = x^2 + y^2$ is an integral of motion (which is easy to verify, for example, by demonstrating that Poisson brackets $\{\Psi, r\} = 0$), solutions of the above stream-lines equations are obviously given by

$$\begin{aligned} x(F(t)) &= x_0 \cos(\omega(r)F(t)) + y_0 \sin(\omega(r)F(t)), \\ y(F(t)) &= -x_0 \sin(\omega(r)F(t)) + y_0 \cos(\omega(r)F(t)), \end{aligned} \tag{1.50}$$

The procedure of transforming to stream-lines coordinates described in section 1.2 can now be followed by performing the canonical transformation to action angle variables. For the particular vortical flow considered in this section, the transformation (1.18) is simply the usual transformation to polar coordinates $(x,y) \to (r,\theta)$. In (r,θ) coordinates, the

advection-diffusion equation (1.7) is written as

$$c_t - \omega(r)f(t)c_\theta = \epsilon\Big(\frac{1}{r}c_r + c_{rr} + \frac{1}{r^2}c_{\theta\theta}\Big) = \epsilon\Delta c \tag{1.51}$$

Furthermore, we should conduct a transformation to stream-lines coordinates as suggested by (1.22). Using the transformation rules (1.23), we rewrite equation (1.51) as

$$v_\tau = \left(\Delta v + F\left(\Big(\frac{\omega'}{r} + \omega''\Big)v_{\bar\theta} + 2\omega' v_{\bar\theta r}\right) + F^2(\omega')^2 v_{\bar\theta\bar\theta}\right) \tag{1.52}$$

Here, we are using t instead of the rescaled time τ. We can now exploit the result obtained in section 1.3 and derive the averaged counterpart of (1.70) to the leading order by simply replacing the time-dependent coefficients by their time averages as

$$V_\tau = \left(\Delta V + \langle F\rangle\left(\Big(\frac{\omega'}{r} + \omega''\Big)V_{\bar\theta} + 2\omega' V_{\bar\theta r}\right) + \left\langle F^2\right\rangle(\omega')^2 V_{\bar\theta\bar\theta}\right) \tag{1.53}$$

It is not difficult to compute second-order corrections to an averaged equation for this flow field. In order to clarify our notations we introduce two operators $\mathcal{L}_1$ and $\mathcal{L}_2$

$$\mathcal{L}_1 \equiv \Big(\frac{\omega'}{r} + \omega''\Big)\partial_{\bar\theta} + 2\omega'\partial_{\bar\theta}\partial_r \tag{1.54}$$

$$\mathcal{L}_2 \equiv (\omega')^2\partial_{\bar\theta}\partial_{\bar\theta} \tag{1.55}$$

and introduce $F_1 \equiv F$ and $F_2 \equiv F^2$. The first-order averaged equation (1.53) becomes then

$$V_t = \epsilon(\Delta V + \langle F_1\rangle\,\mathcal{L}_1 V + \langle F_2\rangle\,\mathcal{L}_2 V). \tag{1.56}$$

For further simplification of notations we introduce functions G_1 and G_2 that can be found explicitly from F_1 and F_2 as

$$G_j(t_0) = \int_0^{t_0} (F_J(\tau) - \langle F_j \rangle) d\tau, \quad j = 1, 2. \tag{1.57}$$

Finally, the averaged advection-diffusion equation to second order can be written down as

$$\begin{aligned} V_t = {} & \epsilon(\Delta V + \langle F_1 \rangle \mathcal{L}_1 V + \langle F_2 \rangle \mathcal{L}_2 V) \\ & + \epsilon^2 \Bigg([\Delta, \langle G_1 \rangle \mathcal{L}_1] V + [\Delta, \langle G_2 \rangle \mathcal{L}_2] V \\ & + \big(\langle F_1 G_1 \rangle - \langle F_1 \rangle \langle G_1 \rangle \big) \mathcal{L}_1^2 V + \big(\langle F_2 G_2 \rangle - \langle F_2 \rangle \langle G_2 \rangle \big) \mathcal{L}_2^2 V \\ & + \big(\langle F_1 G_2 \rangle - \langle F_2 \rangle \langle G_1 \rangle \big) \mathcal{L}_1 \mathcal{L}_2 V + \big(\langle F_2 G_1 \rangle - \langle F_1 \rangle \langle G_2 \rangle \big) \mathcal{L}_2 \mathcal{L}_1 V \Bigg), \end{aligned} \tag{1.58}$$

where $[A, B] \equiv AB - BA$ denotes the usual commutator.

1.5 Numerical simulations.

It is important to justify our results by numerical simulations. We want to integrate numerically both the original equation and the averaged equation to the first order. In other words, we want to compare solutions of equations (1.70) and (1.53). This can be done in Cartesian coordinates by transforming differential operators according to

$$\begin{aligned} r\partial_r &= x\partial_x + y\partial_y, \quad \partial_\theta = x\partial_y - y\partial_x, \\ \partial_\theta \partial_\theta &= -x\partial_x - y\partial_y + y^2 \partial_x \partial_x + x^2 \partial_y \partial_y - 2xy \partial_x \partial_y \\ r\partial_r \partial_r &= x\partial_y - y\partial_x - xy\partial_x \partial_x + (x^2 - y^2)\partial_x \partial_y + xy\partial_y \partial_y \end{aligned} \tag{1.59}$$

For more details of simulations in Cartesian coordinates see [14]. However, taking into account symmetries of the regularized vortical flow field that we considered as an example in section 1.4, it is a more natural choice to keep working in polar coordinates.

Numerical simulations of a 2D system on $\mathbb{R}^2$ is problematic for obvious reasons. Dealing with unbounded domains involves, for example, the usage of the method of local corrections [15, 16] in the finite differences context or infinite elements for finite element methods [17, 18]. Another possible approach is to use mapped Chebychev spectral methods [19–21].

For our purposes these technical complications, however, are unnecessary and, therefore, we consider the advection-diffusion equation on a unit disc ($0 \leq r \leq 1, 0 \leq \theta \leq 2\pi$) with zero Dirichlet boundary conditions $v(r = 1, \theta) = 0$. We compare two solutions of the equations (1.70) and (1.53) at Poincaré sections where $F = 0$. We use Chebychev spectral methods to numerically approximate spatial differentiation operators and a second order Crank-Nicolson finite difference scheme in time. Details of the numerical scheme and algorithm are discussed in chapter 3.

From now on we assume an initial condition of the form

$$v_0(r, \theta) = re^{-br^2} \cos(\pi r/2) \tag{1.60}$$

where the parameters a from the expression for angular velocity (1.49) and b are chosen to be $a = 0.05, b = 20$ and $\cos(\pi r/2)$ is introduced to match the boundary conditions. As follows from (1.70) and (1.53), the impact of the averaging procedure is dictated by

the choice of time-dependence $f(t)$. We explore the case $f(t) = \sin(2\pi t/T)$ such that the period of the driving advective force is $T = 1$. For this choice of $f(t)$, the averaged $\langle F \rangle$ and $\langle F^2 \rangle$ are found to be

$$\begin{aligned} F &\equiv \int_0^t \sin(2\pi t'/T)dt' = \frac{T}{2\pi}\left(1 - \cos(2\pi t/T)\right) \\ \langle F \rangle &\equiv \frac{1}{T}\int_0^T F(t)dt = \frac{1}{T}\int_0^T \frac{T}{2\pi}\left(1 - \cos(2\pi t/T)\right)dt = \frac{T}{2\pi} \\ \left\langle F^2 \right\rangle &\equiv \frac{1}{T}\int_0^T F^2(t)dt = \frac{1}{T}\int_0^T \left(\frac{T}{2\pi}\left(1 - \cos(2\pi t/T)\right)\right)dt = \frac{3T^2}{8\pi^2} \end{aligned} \tag{1.61}$$

It is clear that, in the case when the advection field is absent, the evolution of the scalar

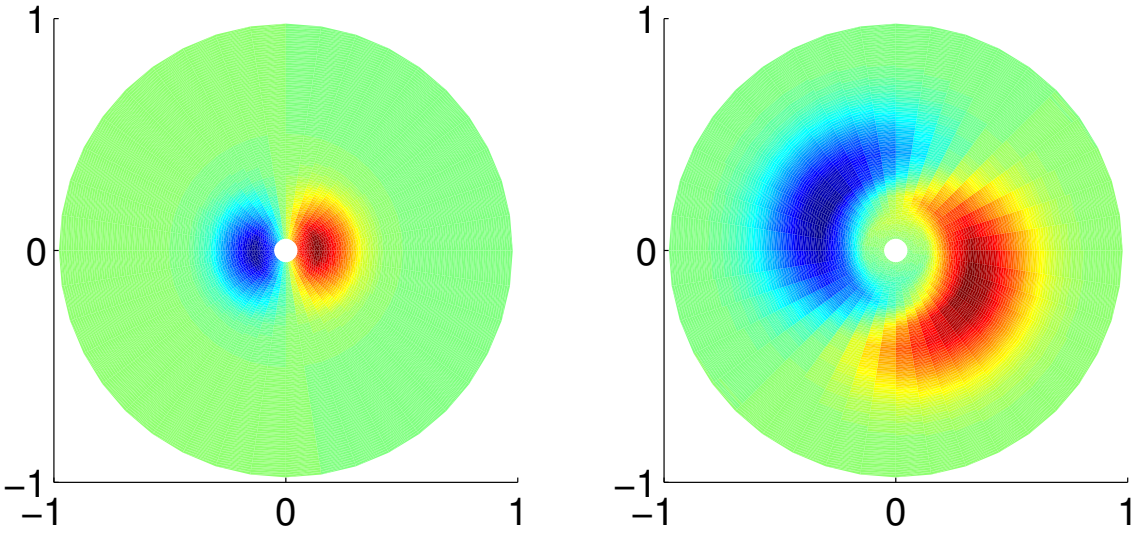


FIGURE 1.2: Evolution of the scalar tracer field in a time-dependent vortical velocity field. The figure on the left represents the initial condition, the figure on the right shows the state of the system after 10 periods.

tracer field will be described by a purely diffusive equation:

$$v_t = \epsilon \Delta v \tag{1.62}$$

The effect of pure diffusion will appear as simple broadening and widening of the initial condition with the preservation of the spatial symmetries of the initial state of the tracer

field. Considering the action of an exclusively advective force alone, particles would move alongside stream-lines trajectories returning exactly to their original positions after each period. This motion would obviously preserve symmetries as well. However the combination of these two types of motion destroys the original symmetries due to the interplay between advection and diffusion. The given initial condition develops a "twist" that is clearly seen on Fig. 1.2.

In order to provide a visual validation of similarities between solutions of equations (1.70) and (1.53), we plot the difference between the solution of the advection-diffusion equation and the pure diffusion equation $v_{\text{ad}} - v_{\text{diff}}$ for both cases of the fully time-dependent equation (1.70) and its averaged companion (1.53). This is presented in Fig. 1.3. The

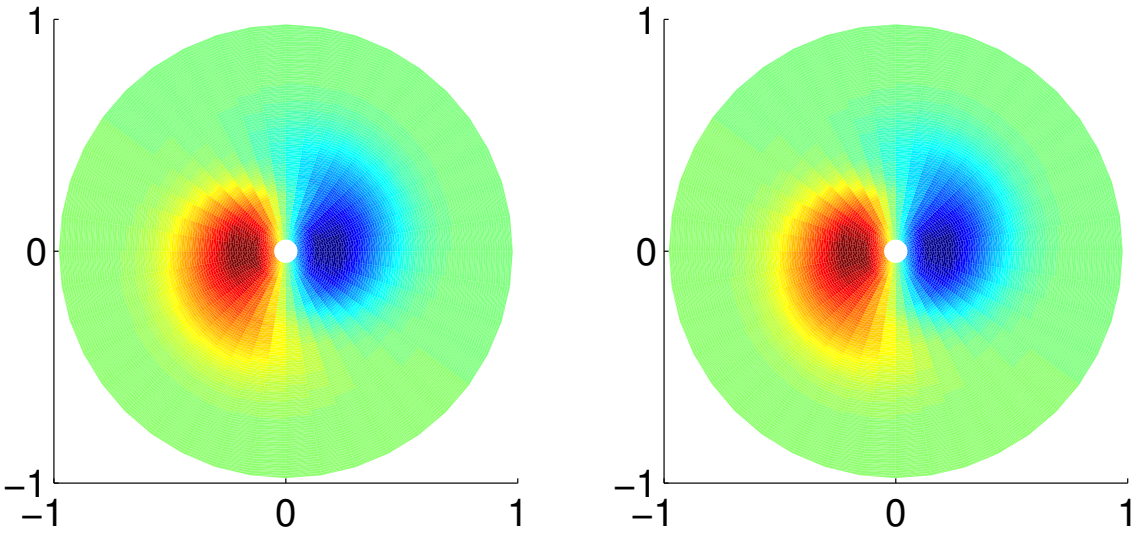


FIGURE 1.3: Left figure shows the difference between equation (1.70) and the pure diffusion equation $v_{\text{ad}} - v_{\text{diff}}$. The figure on the right shows the same difference between two solutions $V_{\text{ad}} - V_{\text{diff}}$ but for equation (1.53).

averaged equation is obviously capable of qualitatively demonstrating the effect of a

time-dependent velocity field. Comparing actual solutions of the full equation with time-dependent advection and the averaged equation as displayed in Fig. 1.4, we confirm again that the averaged equation captures very well the dynamics of the time-dependent equation.

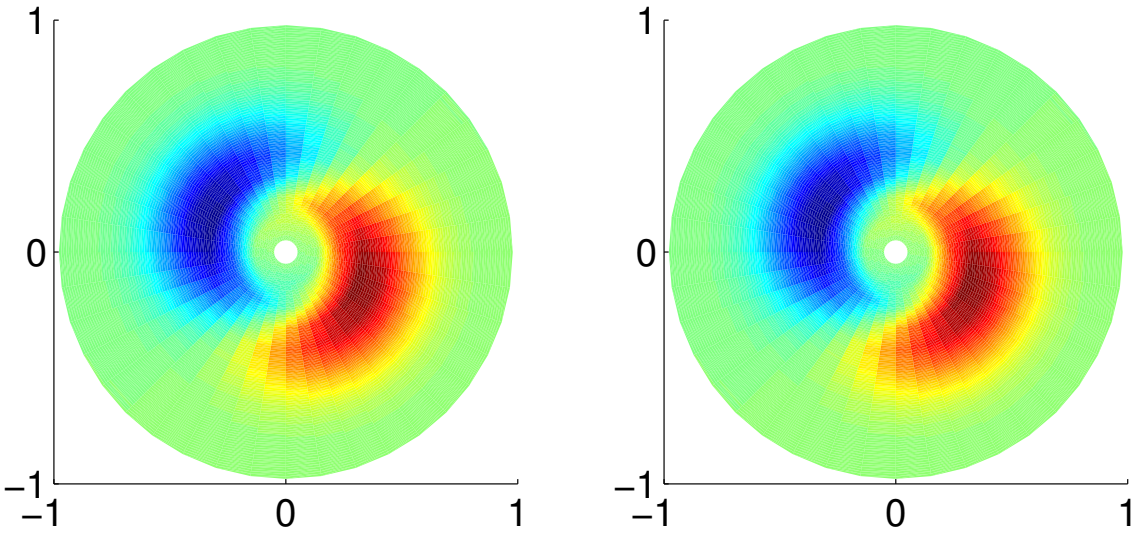


FIGURE 1.4: Scalar field after 20 periods as computed by the full time-dependent equation (1.70) (left) and the averaged equation (1.53) (right). Parameter ϵ here was chosen to be $\epsilon = 10^{-3}$.

As a quantitative measure of the accuracy of the averaged approximation we introduce the L^2-norm of the difference as

$$||v - v_{\text{av}}|| = \left(\frac{\int_\Omega |v - v_{\text{av}}|^2 \, dx \, dy}{\int_\Omega |v|^2 \, dx \, dy} \right)^{1/2} \tag{1.63}$$

Assuming that $||v - v_{\text{av}}|| \sim \epsilon^n$ we find from the analysis of the data presented in the Fig. 1.5 that the convergence rate is $\sim \epsilon^{0.82}$.

In order to get a better understanding of the differences between the behavior of the full and approximate equations we introduce the following operator that maps the scalar field

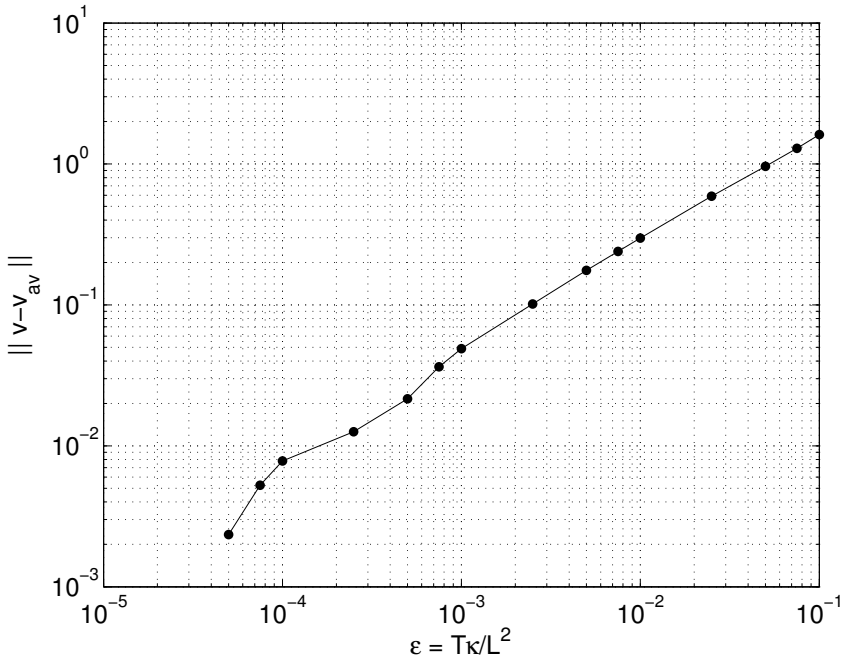


FIGURE 1.5: L^2-norm of a difference between solutions to full and approximate equations as a function of parameter ϵ. Convergence rate is found to be $\sim \epsilon^{0.82}$.

between two consequent Poincaré sections:

$$\begin{aligned} \mathcal{Q}: &\quad u(r,\theta,t+T) = \mathcal{Q}u(r,\theta,t) \\ \mathcal{Q}_{\mathrm{av}}: &\quad u_{\mathrm{av}}(r,\theta,t+T) = \mathcal{Q}_{\mathrm{av}}u_{\mathrm{av}}(r,\theta,t) \end{aligned} \tag{1.64}$$

We can now study how the eigenvalues λ_j defined via

$$\mathcal{Q}\psi_j = \lambda_j \psi_j \tag{1.65}$$

change with ϵ. Fig. 1.7 shows the first 40 eigenvalues (their absolute values) of the operator $\mathcal{Q}_{\mathrm{av}}$ whereas Fig. 1.6 depicts those of the operator $\mathcal{Q}$ for the chosen set of values of ϵ. First of all we see that eigenvalues corresponding to the same "quantum number" decrease with increasing value of ϵ. This simply means that solutions of the equation with higher ϵ decay faster. Secondly we observe that the structures of the

spectra of both operators are extremely similar and it is impossible to distinguish one from the other by looking at the plot. This serves as another indication of the validity of the approximate equation. Finally, we note that many of the eigenvalues are doubly degenerate.

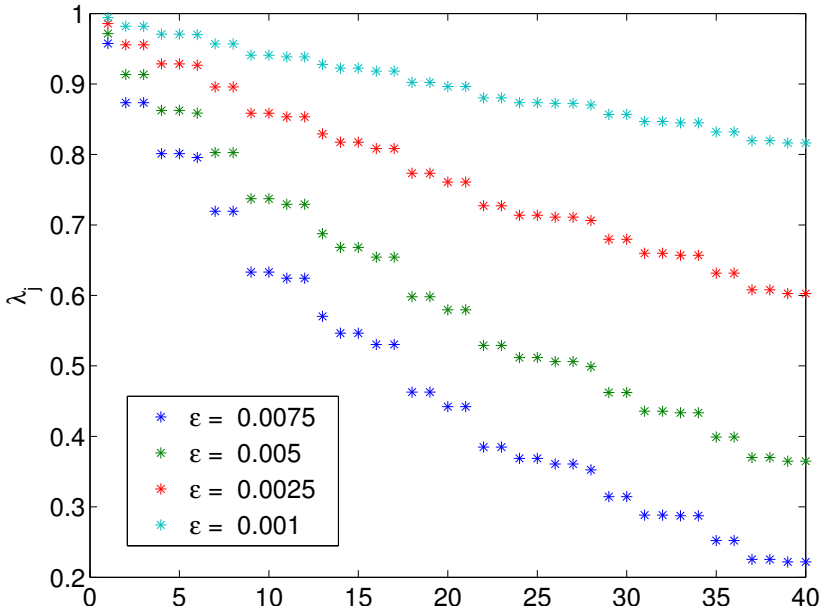


FIGURE 1.6: First 40 eigenvalues of the full one-period evolution operator $\mathcal{Q}$.

If we consider the map (1.64) applied to eigenstates of ψ_j we will immediately conclude that in the limit of time $t \to \infty$ (or in other words after applying map (1.64) many times) only modes corresponding to larger eigenvalues will survive, $\mathcal{Q}^N \psi_j = \lambda_j^N \psi_j$, and all the modes with smaller eigenvalues will decay exponentially faster. Therefore, it is interesting to see how the first eigenmode depends on ϵ for both operators. These data are presented in Fig. 1.8

There are two important things to note here. First, the largest eigenmode is exactly the same (up to machine precision) for both operators $\mathcal{Q}$ and $\mathcal{Q}_{\rm av}$! Second, the dependence of

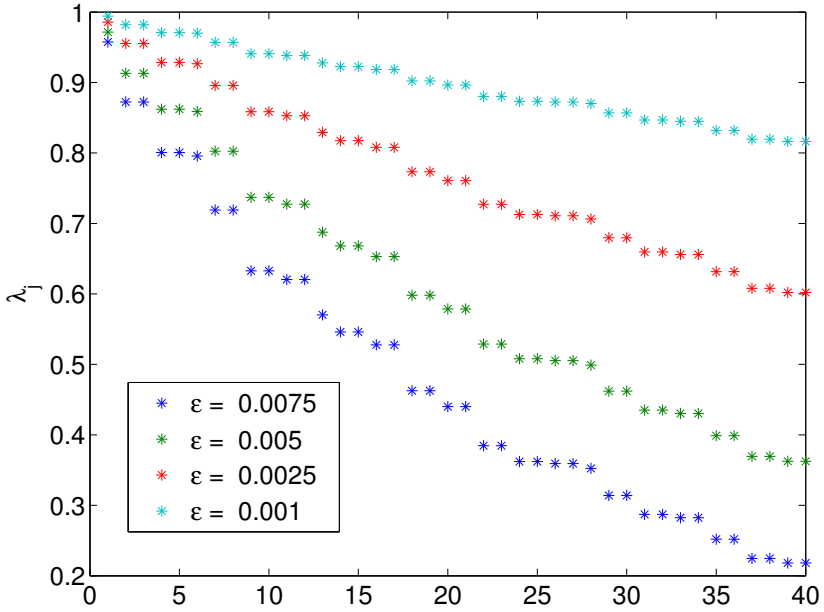


FIGURE 1.7: First 40 eigenvalues of the averaged one-period evolution operator $\mathcal{Q}_{\mathrm{av}}$.

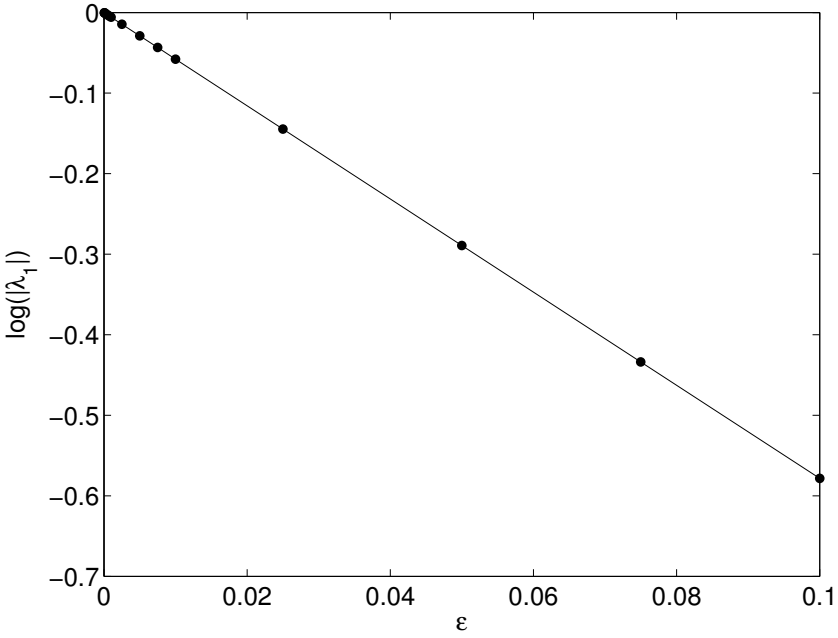


FIGURE 1.8: Logarithm of the largest eigenvalue of the one-period evolution operator as a function of ϵ. It is a straight line that is fitted by $y = -5.7832x + 0.0$.

the logarithm of the first eigenvalue on ϵ is linear, $\log(\lambda_1(\epsilon)) \sim \epsilon$, with a proportionality coefficient $k = -5.7832$. To understand this behavior we need to recall the solution of the heat equation on a unit disc. The ground state eigenfunction of the Laplacian in polar coordinates with zero Dirichlet boundary conditions on a unit disc is given by $J_0(\lambda_1^0 r)$ – the 0^{th}-order Bessel function with corresponding eigenvalue $(\lambda_1^0)^2$. Here λ_1^0 is the first root of $J_0(r)$ defined as $J_0(\lambda_1^0) = 0$. This value is known to be $\lambda_1^0 \approx 2.40482$. If used as the initial condition for the diffusion equation, its time evolution is described by $u = u_0 \exp(-\epsilon(\lambda_1^0)^2 t)$. Returning back to the world of Poincaré maps of period $T = 1$ we state that such map for the heat equation considered above is written as

$$\mathcal{Q}\psi_j = \lambda_j \psi_j = \exp(-\epsilon(\lambda_1^0)^2)\psi_j \tag{1.66}$$

This explains the linearity observed in Fig. 1.8 and explains the coefficient of proportionality k as $(\lambda_1^0)^2 \approx 5.7832$. It is also clear why the lowest eigenvalues of both operators $\mathcal{Q}$ and $\mathcal{Q}_{\text{av}}$ are equal to each other up to 12 digits. The lowest eigenvalue corresponds to the eigenfunction with a maximal symmetry – axial symmetry in this case. Obviously, axially-symmetric advection does not play any role in the dynamics of the axially symmetric distribution of the passive tracer.

Thus, the dynamics of such a system is described equivalently by both full the time-dependent operator $\mathcal{Q}$ and the average operator $\mathcal{Q}_{\text{av}}$ (and, as a matter of fact, is simply a dynamics of the heat equation). Eigenmodes that correspond to real eigenvalues will possess axial symmetry. For the reason that the real eigenstates are not affected by advection, such modes of the full operator and of the average operator will be *exactly* the

same independently of ϵ. We present first the 16 eigenmodes of operator $\mathcal{Q}$ in Fig. 1.9 and Fig. 1.10, and first 16 eigenmodes of operator $\mathcal{Q}_{\text{av}}$ on Fig. 1.11 and Fig. 1.12. Both cases are shown for $\epsilon = 0.01$

As we can see from figures 1.9 - 1.12, most of the eigenmodes come in complex-conjugate pairs. Those that do not have a conjugate partner correspond to a real eigenvalue and possess axial symmetry. Obviously, the procedure of averaging introduces an error to the result of the eigenvalues calculation. However, the average operator $\mathcal{Q}_{\text{av}}$ has exactly the same structure of its eigenmodes.

In this consideration an important question needs to be addressed. How much of the error between eigenvalues of the full and average operators is generated by the numerical discretization imprecision? To answer this question we study how the difference between the eigenvalues of both operators depends on the time steps used in the numerical difference scheme. We illustrate the contribution of the time-step error by plotting $|\lambda_j(\Delta t) - \lambda_j(\Delta t/2)|$ for several eigenvalues λ_j as a function of the time-step Δt. As seen from the Fig. 1.13, the higher the "quantum number" is, the more sensitive are the computations to the time discretization. Whereas the numerical scheme used to compute eigenvalues of the time-dependent operator is stable up to $\Delta t = 0.01$, it becomes unstable for the average operator at certain values of Δt, and the results become compromised. The tendency of the scheme for the average operator to become unstable with increasing Δt is more obvious for higher values of ϵ. In other words, to provide stability of the scheme for bigger values of ϵ, one has to decrease the time discretization Δt. Similar observations for the dependence of the results on spatial discretization show that there

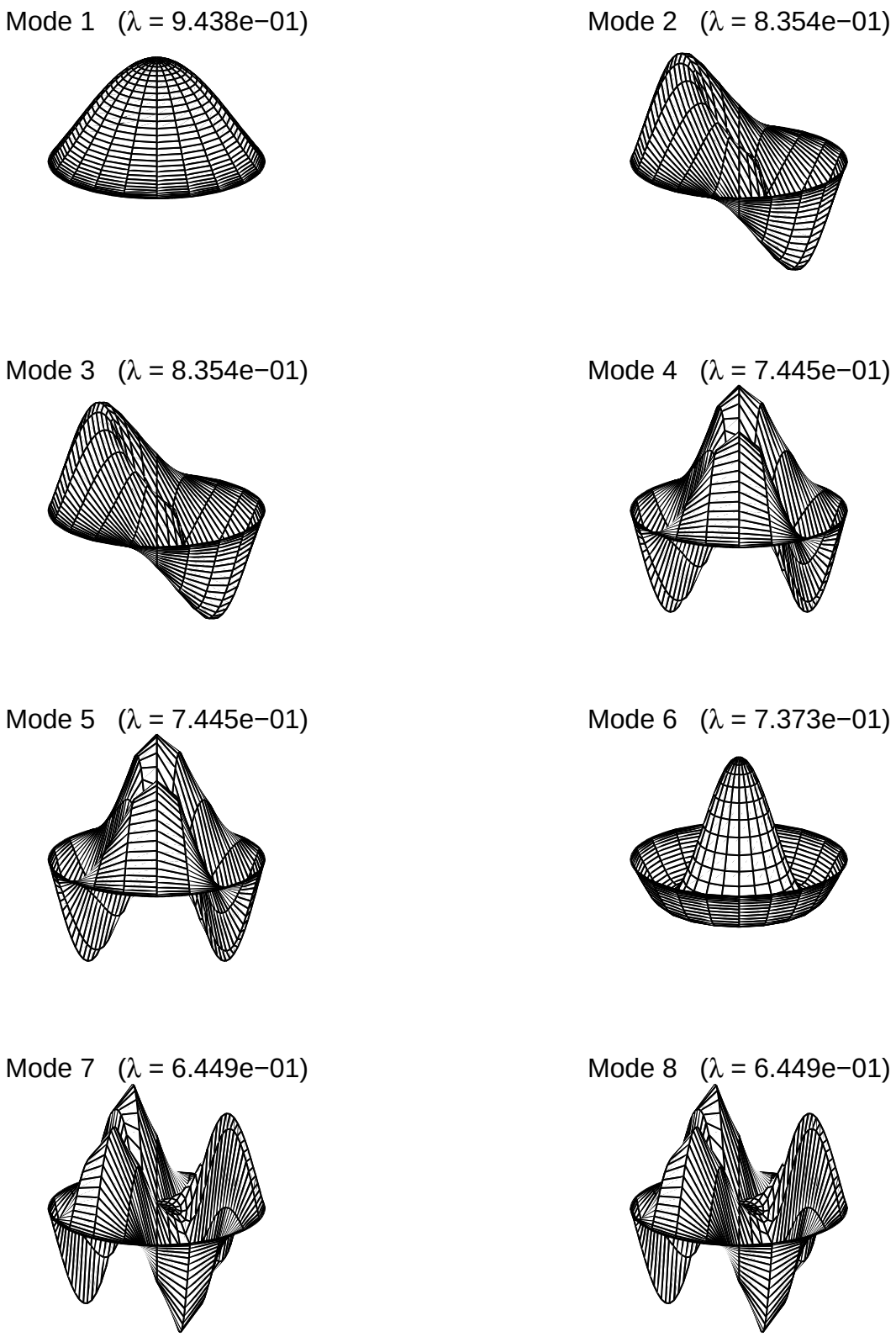


FIGURE 1.9: Eigenmodes 1-8 of the full time-dependent one-period evolution operator $\mathcal{Q}$.

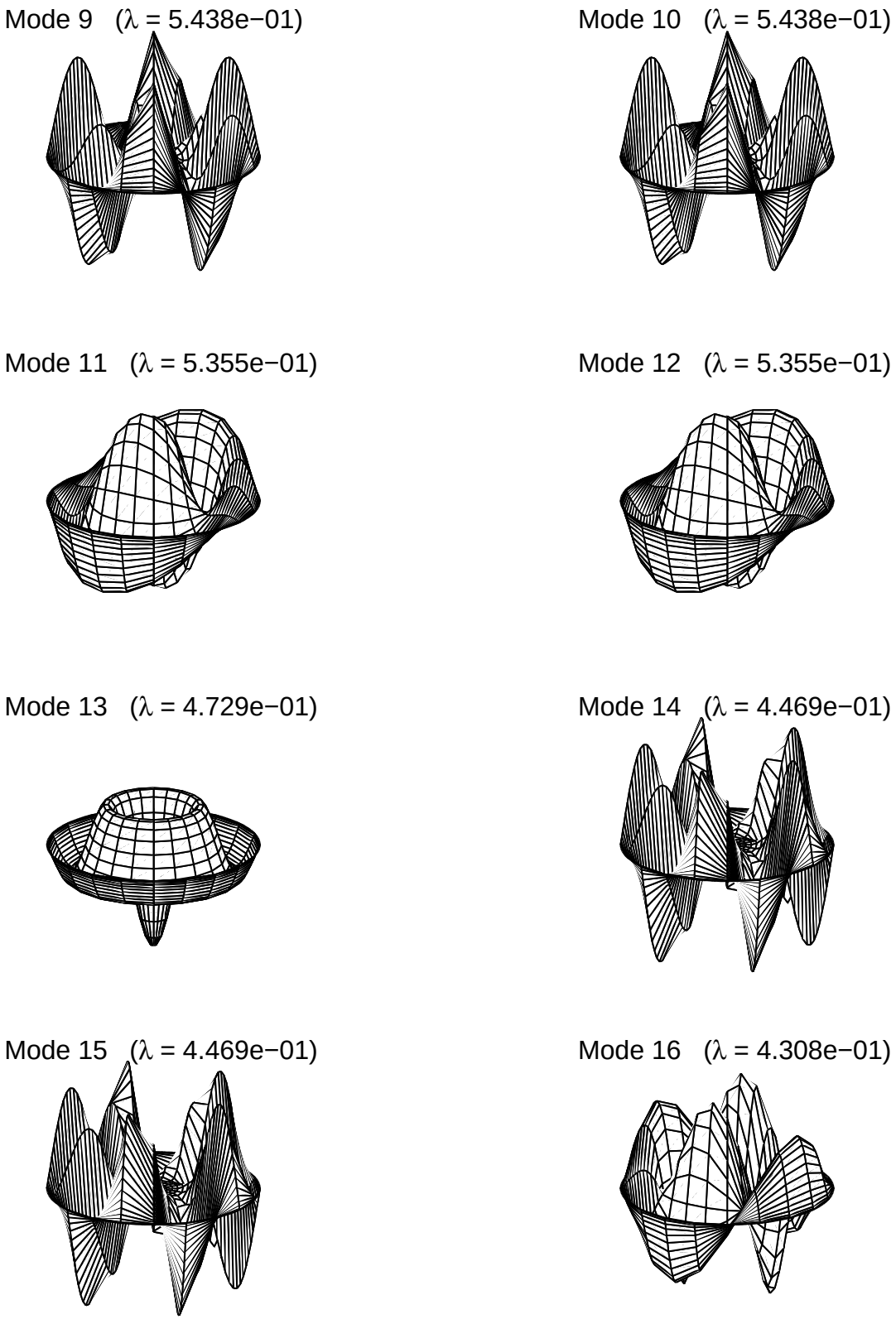


FIGURE 1.10: Eigenmodes 9-16 of the full time-dependent one-period evolution operator $\mathcal{Q}$.

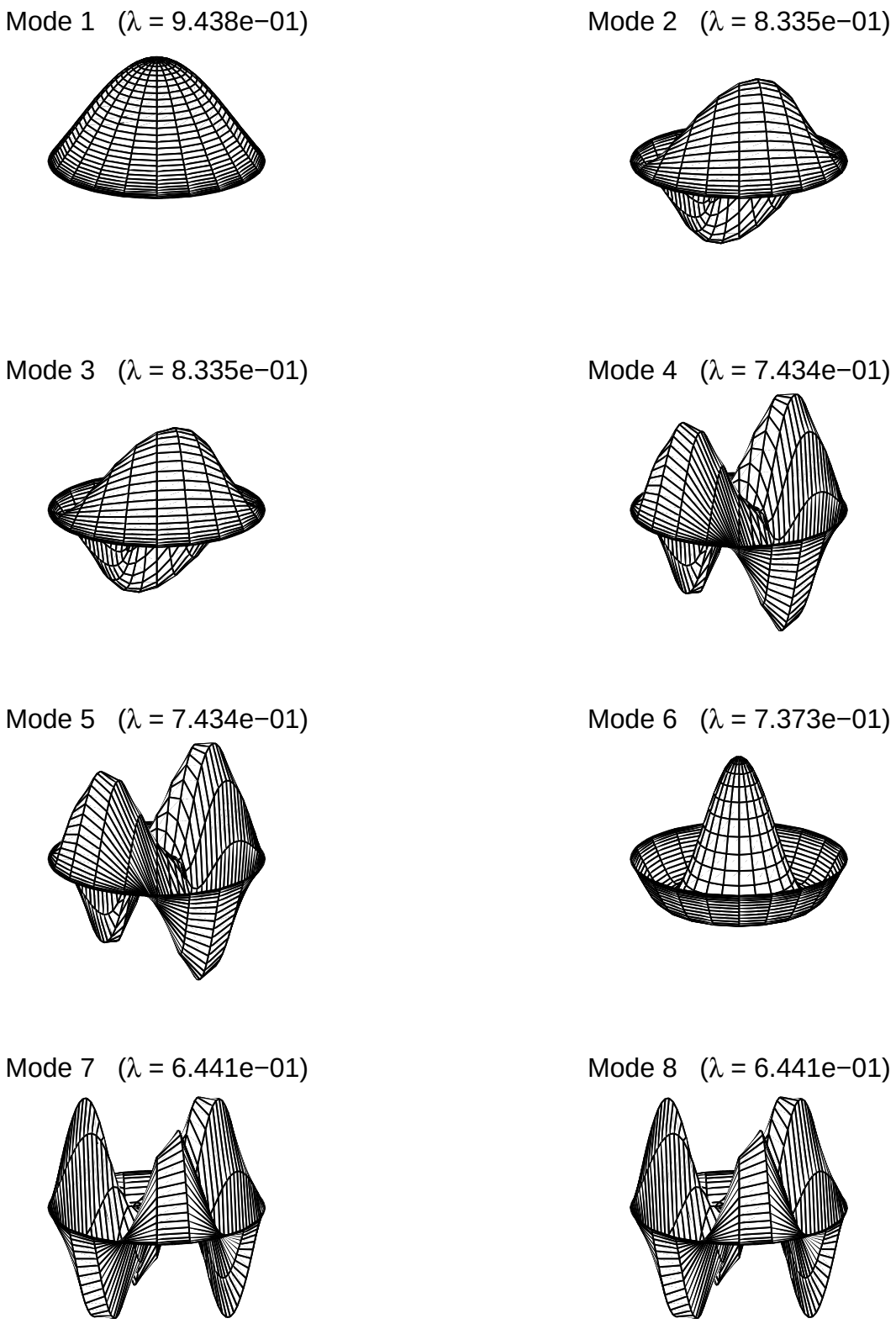


FIGURE 1.11: Eigenmodes 1-8 of the one-period evolution operator $\mathcal{Q}_{\mathrm{av}}$ that describes averaged dynamics.

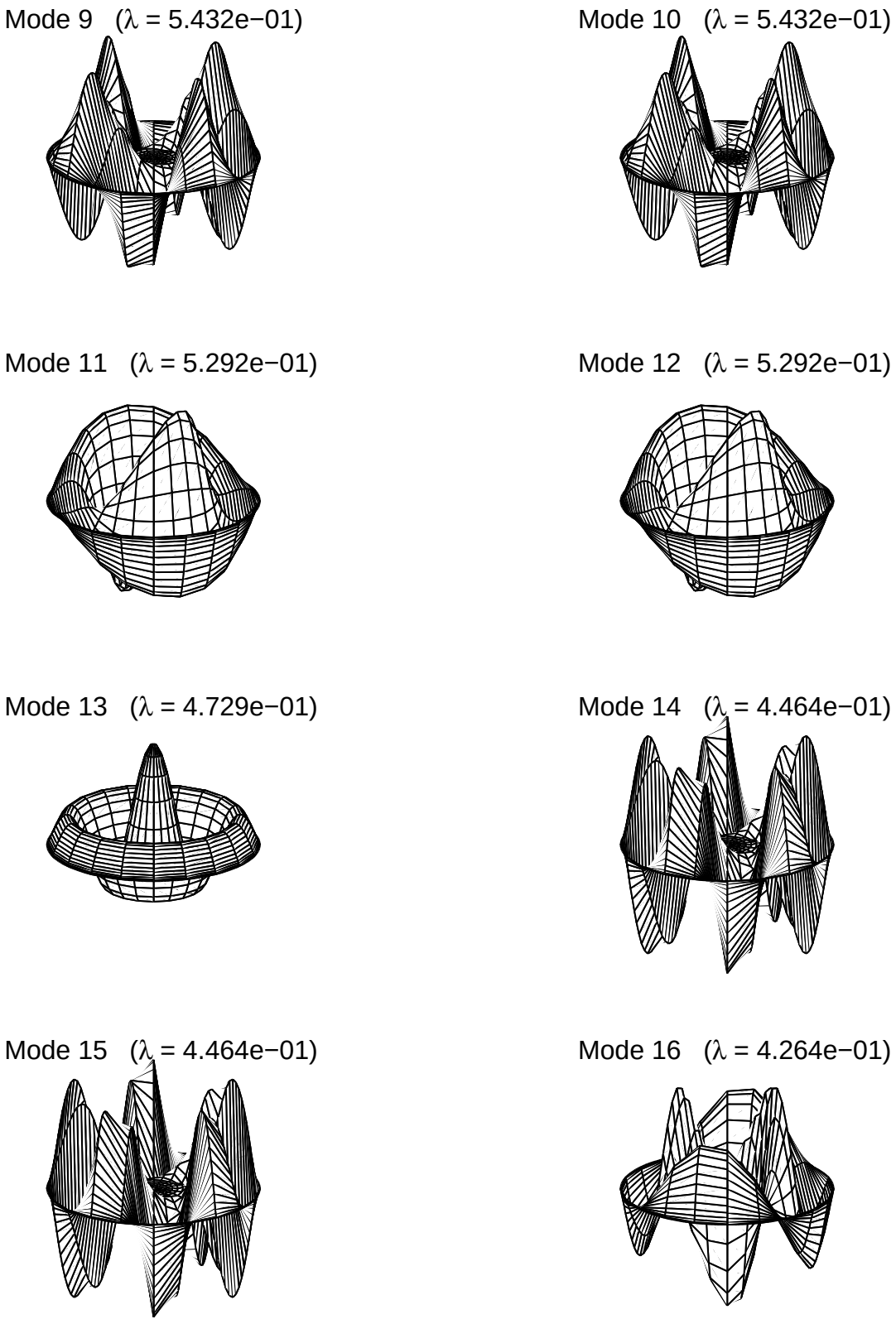


FIGURE 1.12: Eigenmodes 9-16 of the one-period evolution operator $\mathcal{Q}_{\mathrm{av}}$ that describes averaged dynamics.

are no signs of instability of the scheme as long as the number of Chebychev points is kept reasonably high. We performed simulations for $N = 61$ and $N = 31$ and did not see any significant difference in result. For details of the numerical scheme please refer to Chapter 3.

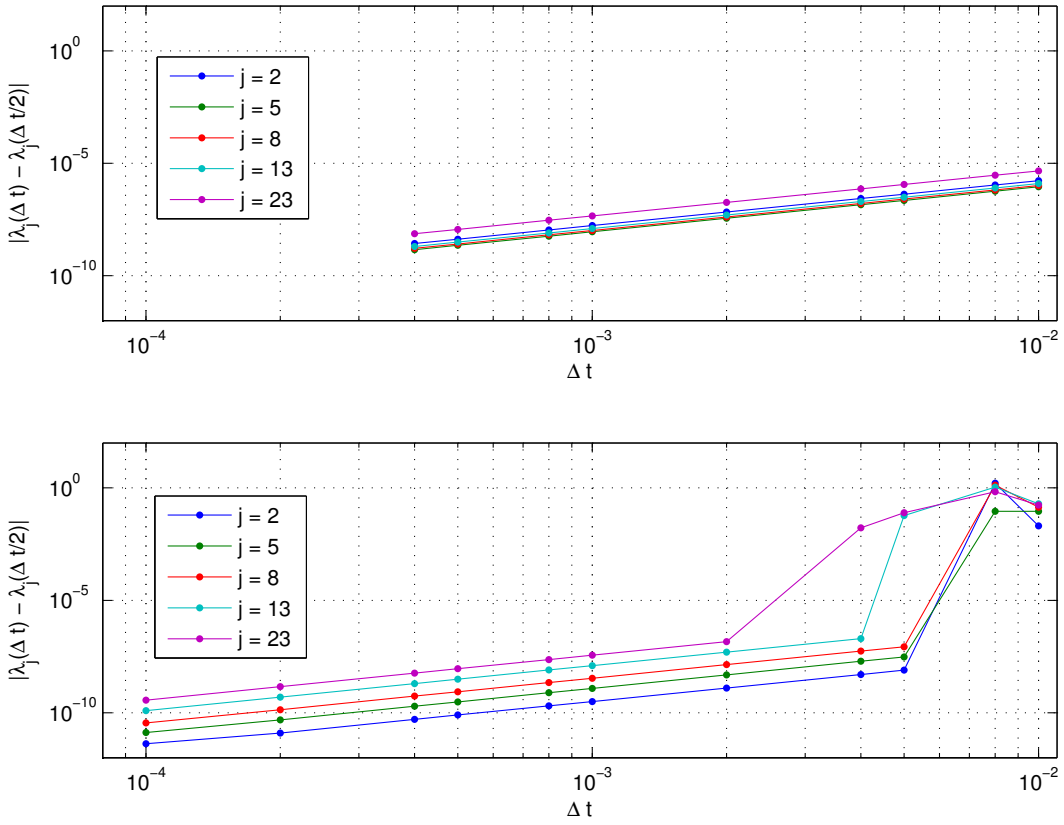


FIGURE 1.13: Influence of the time discretization error on computations of eigenvalues λ_j. Upper plot shows λ_j of the full operator $\mathcal{Q}$ and lower plot shows those of the operator $\mathcal{Q}_{\text{av}}$. The parameter ϵ used here is $\epsilon = 0.01$.

We present the dependence of the relative difference in the eigenvalues of operators $\mathcal{Q}$ and $\mathcal{Q}_{\text{av}}$ for the j^{th} mode as given by

$$\delta\lambda_j = \frac{|\lambda_j - \lambda_j^{(\text{av})}|}{\lambda_j} \tag{1.67}$$

in Fig. 1.14. This is to be viewed as a function of ϵ. It is immediately seen from this plot that modes 1 and 13 possess axial symmetry, and, therefore, the procedure of averaging does not have any effect on the one-period evolution. We also note that for $\epsilon \gtrsim 0.05$ the numerical scheme becomes unstable. This indicates the effect of the time-discretization error. We demonstrate it by measuring the same dependence (1.67), but with smaller time discretization $\Delta t = 0.0001$. As seen in Fig. 1.15, the error introduced by averaging is well-behaved on the whole range $5 * 10^{-5} \leq \epsilon \leq 10^{-1}$, which supports our reasoning about the time-discretization error introduced into calculations of eigenmodes.

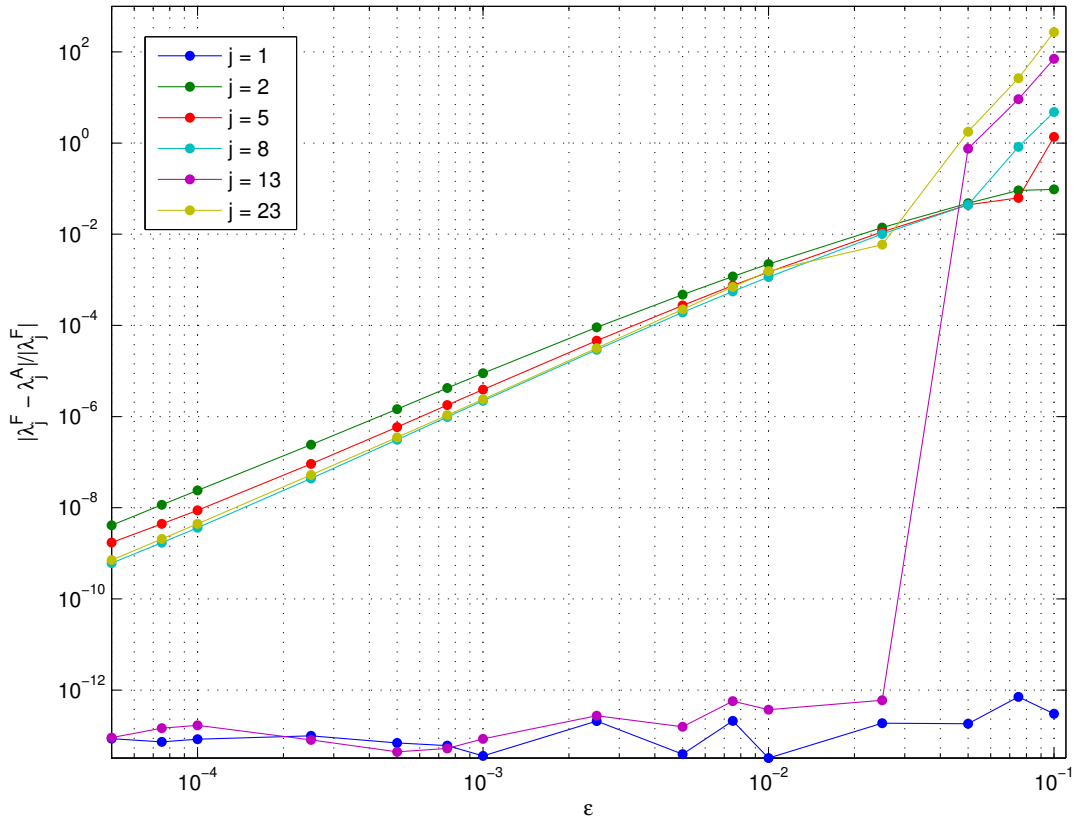


FIGURE 1.14: Difference in the calculation of eigenvalues of the full and average operators (as given by (1.67)) as a function of parameter ϵ. These data were obtained using time-discretization $\Delta t = 0.001$.

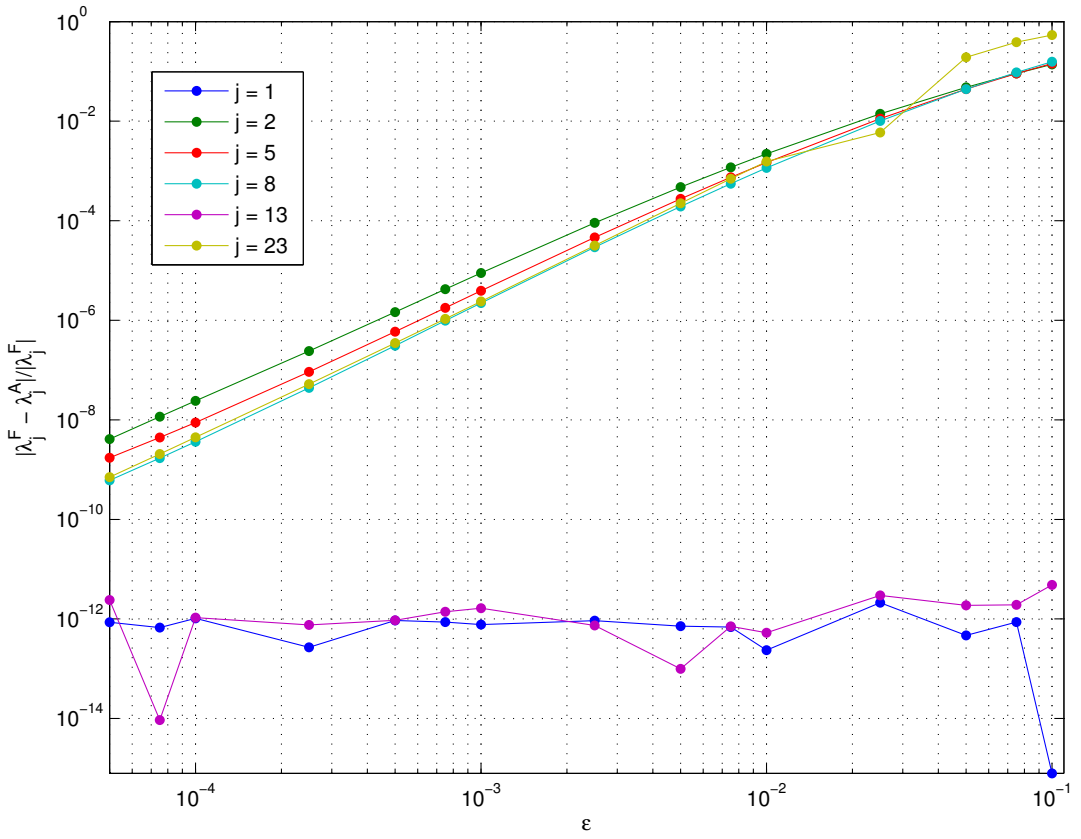


FIGURE 1.15: Difference in the calculation of eigenvalues of the full and average operators (as given by (1.67)) as a function of parameter ϵ obtained with finer time-discretization $\Delta t = 0.0001$. Compare to Fig 1.14

.

To conclude, we illustrate how the averaging procedure works for different types of the time dependence of the velocity field. Considering $f(t) = \cos(2\pi t)$, we easily find that in this case $\langle F \rangle = 0$ and $\langle F^2 \rangle = T^2/8\pi^2$. Evolution of the same initial condition for 10 periods is shown on Fig. 1.16 (here and to the end of this section, states of the scalar field are computed for $\epsilon = 10^{-3}$). We show that the convergence of the error introduced by averaging in this case is similar to the case of using a sin-force (see Fig. 1.17).

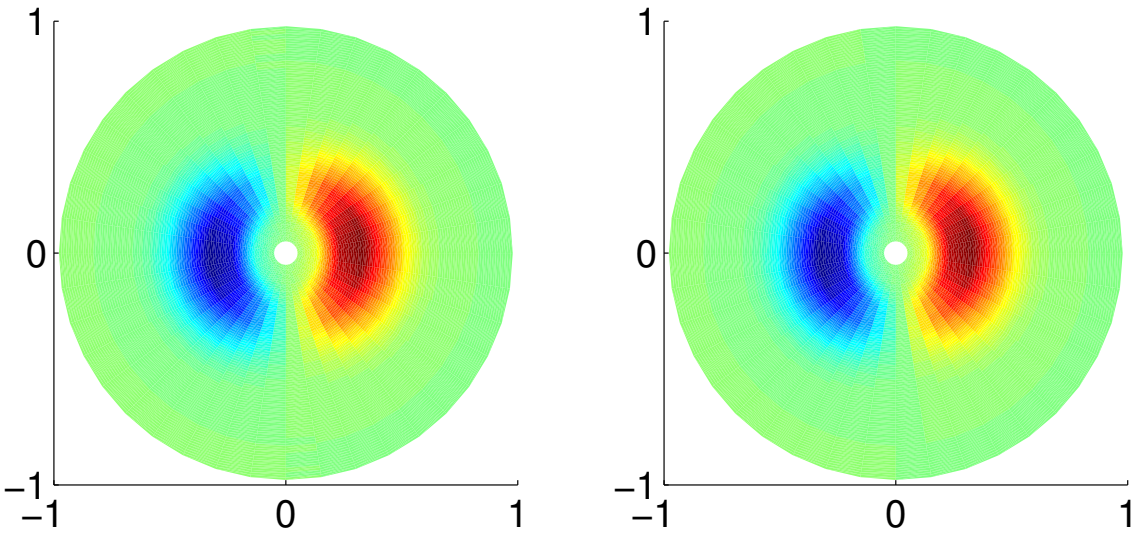


FIGURE 1.16: Evolution of the tracer field subjected to $\cos(2\pi t)$ advection after 10 periods computed by the full time-dependent equation (left) and the averaged equation (right).

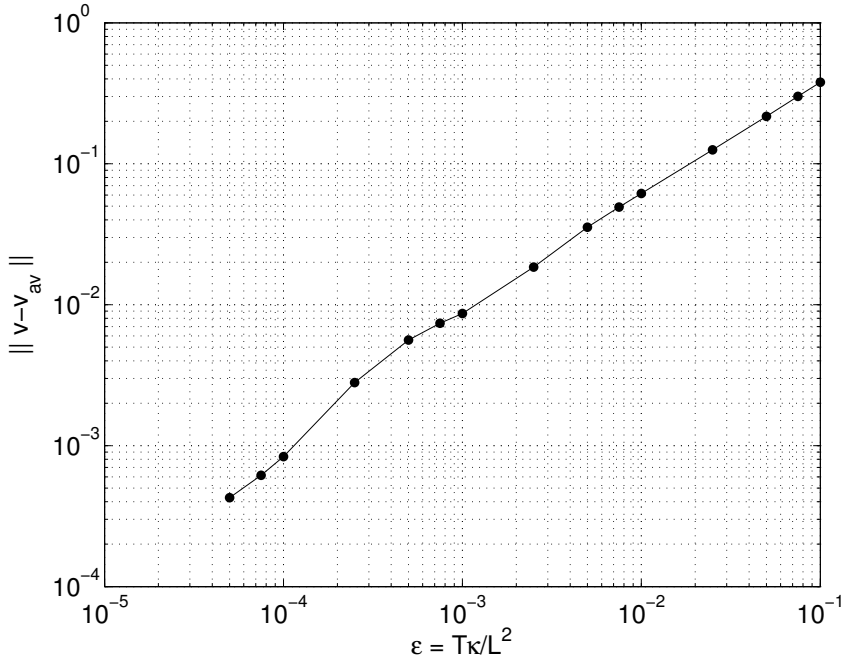


FIGURE 1.17: L^2-norm of a difference between the solutions to thre full and the approximate equations as a function of parameter ϵ for the case $f(t) = \cos(2\pi t)$. Convergence rate is found to be $\sim \epsilon^{0.88}$.

Finally, we consider time dependence given by

$$f(t) = \begin{cases} 2 & : \quad 0 \le t \le T/4 \\ \\ -2/3 & : \quad T/4 \le t \le T \end{cases} \tag{1.68}$$

It obviously is a zero-mean function as required in (1.11). This choice implies $\langle F \rangle = T/4$ and $\langle F^2 \rangle = T^2/12$. The tracer's state after 10 periods and L^2-norm are shown in Fig. 1.18 and Fig. 1.19 respectively. For this case, the convergence of the averaged solution to the true solution is slower, namely it goes as $\epsilon^{0.65}$. Thereby, one has to go to the second order approximation as given in (1.58) if better accuracy is needed.

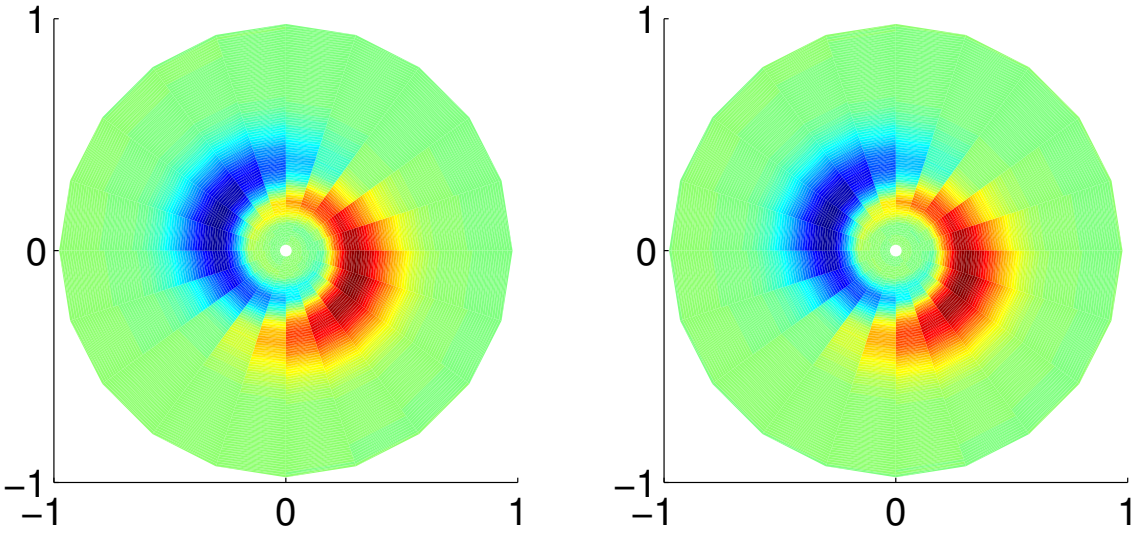


FIGURE 1.18: Evolution of the tracer under the non-harmonic force (1.68) after 10 periods computed by the full time-dependent equation (left) and the averaged equation (right).

Finally we want to make a remark about the type of the flow time-dependence restrictions. So far we only discussed time-periodic flows with zero mean: $f(t+T) = f(t)$ and $\int_0^T f(t)dt = 0$. However, the method discussed above may be expanded to a slightly more

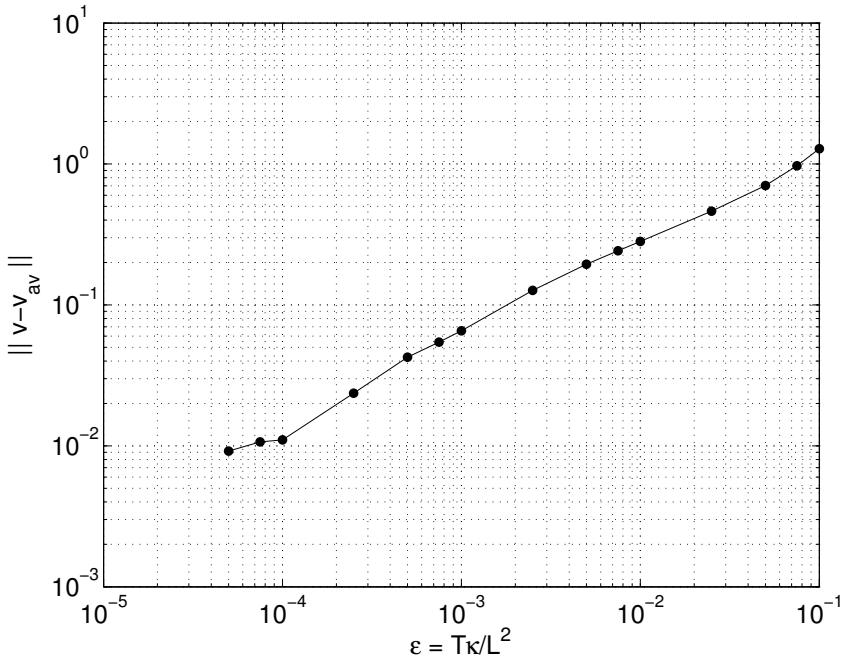


FIGURE 1.19: L^2-norm of a difference between the solutions to the full and the approximate equations as a function of parameter ϵ for the case of a non-harmonic force given by (1.68). Convergence rate is $\sim \epsilon^{0.65}$.

general class of flow fields, namely, we can consider flow with time dependence in the form

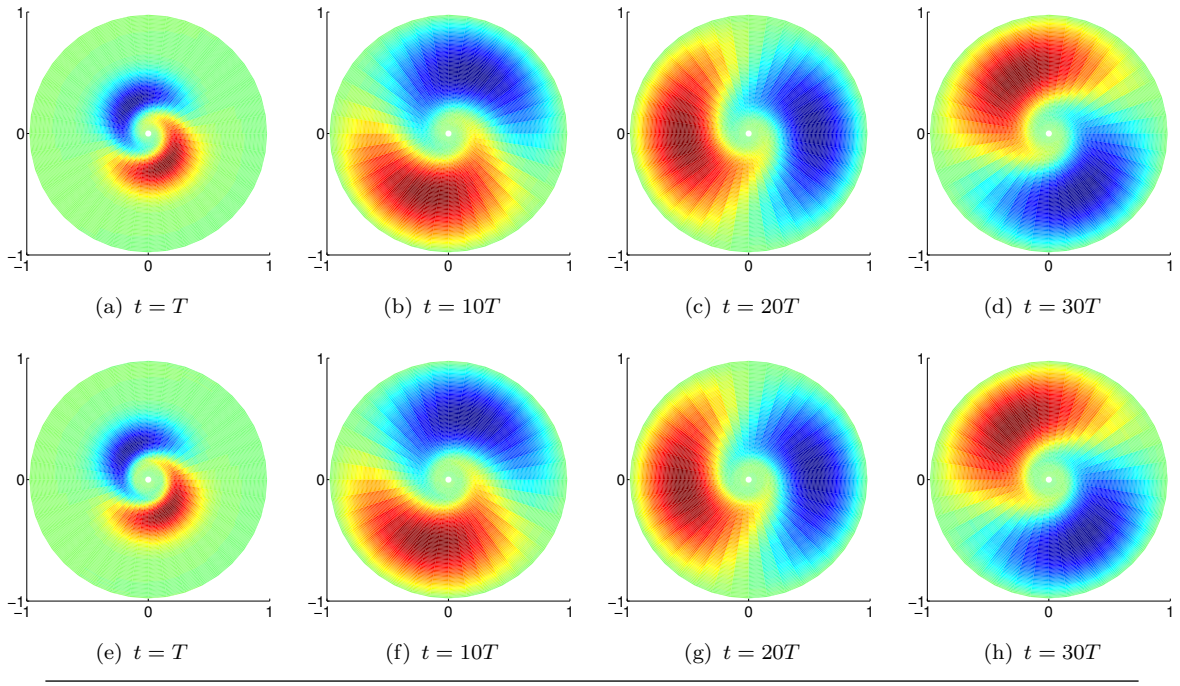
$$f(t) = f_0(t) + f_1, \tag{1.69}$$

where $f_0(t)$ is periodic and mean-free function of time and f_1 is a constant. Using this time-dependence in the equation (1.47) we obtain equations, analogous to (1.70) and (1.53) in the form

$$v_\tau = \left(\Delta v + F\left(\left(\frac{\omega'}{r} + \omega'' \right) v_{\bar{\theta}} + 2\omega' v_{\bar{\theta}r} \right) + F^2 (\omega')^2 v_{\bar{\theta}\bar{\theta}} \right) + f_1 \omega v_{\bar{\theta}}, \tag{1.70}$$

$$V_\tau = \left(\Delta V + \langle F \rangle \left(\left(\frac{\omega'}{r} + \omega'' \right) V_{\bar{\theta}} + 2\omega' V_{\bar{\theta}r} \right) + \left\langle F^2 \right\rangle (\omega')^2 V_{\bar{\theta}\bar{\theta}} \right) + f_1 \omega V_{\bar{\theta}}, \tag{1.71}$$

with F being in this case $F(t) = \int_0^t f_0(t')dt'$. This motion corresponds to the rotation



(a) $t = T$ (b) $t = 10T$ (c) $t = 20T$ (d) $t = 30T$

(e) $t = T$ (f) $t = 10T$ (g) $t = 20T$ (h) $t = 30T$

FIGURE 1.20: Full time-dependent and averaged dynamics of the tracer under the influence of the constant mean flow shown for 1 period, 10, 20 and 30 periods. The top raw shows results for full operator ans bottom raw — averaged evolution.

of the system as a whole with a constant angular velocity ω (which still is a function of r) and periodic oscillations superposed with this rotational motion. Because of the fact that rotational motion is dependent upon r, large gradients are constantly created in the scalar field. These gradient are exposed to the action of diffusive smearing. The enhanced stretching of the tracer field creates somewhat richer dynamics and provides for faster mixing. We demonstrate evolution of the initial state (1.60) for the case of the flow (1.69) and $\epsilon = 0.01$ in the Fig. 1.20. Clearly, states as computed using full time-dependent operator and averaged operator are almost indistinguishable. In fact, convergence of the averaged solution to the true solution, defined by (1.63) is as good as for mean-free field and in this particular case is $\sim \epsilon^{0.88}$. The L^2-norm is presented in Fig. 1.21

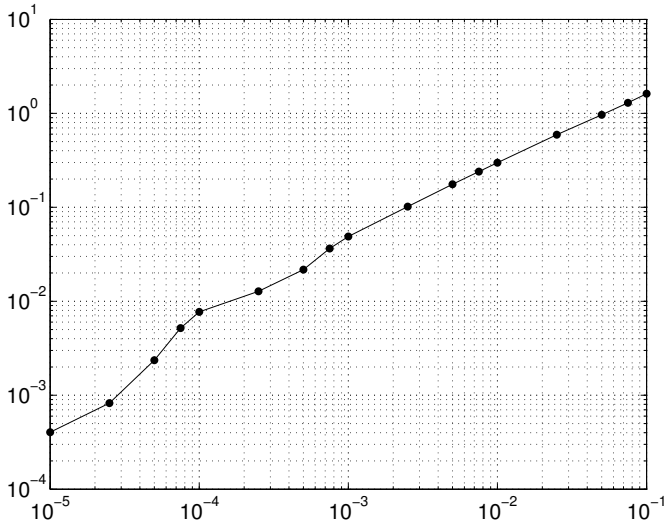
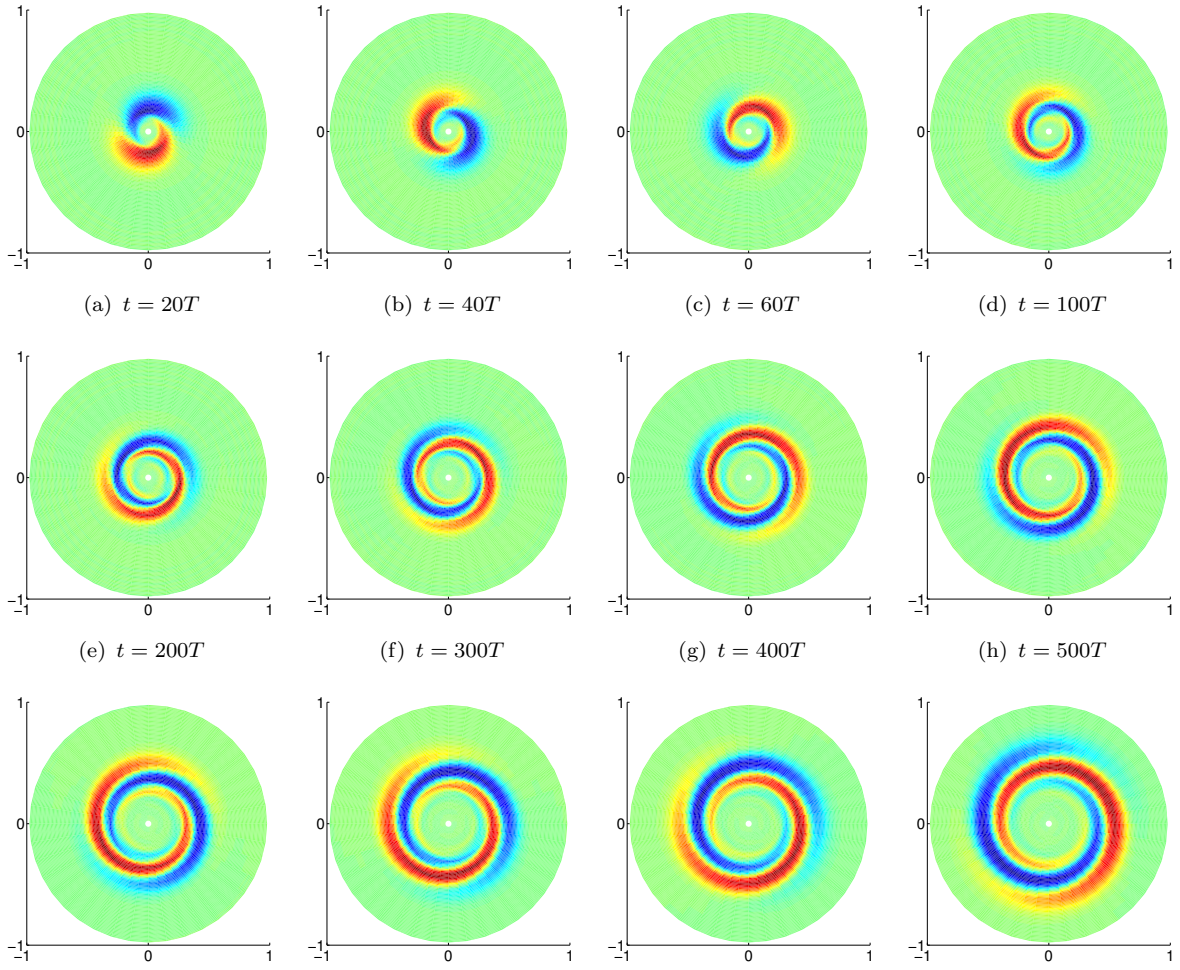


FIGURE 1.21: L^2-norm of a difference between solutions to full and approximate equations as a function of parameter ϵ. Convergence rate is found to be $\sim \epsilon^{0.88}$.

We finally show evolution of the system for the case of very small effective diffusivity $\epsilon = 10^{-5}$. For such small value of the parameter ϵ, tracer field does not diffuse trough the boundary for a long time, and, therefore, large twists can be created by the mean component of the circular flow. We illustrate it in the Fig. 1.22.



(a) $t = 20T$ (b) $t = 40T$ (c) $t = 60T$ (d) $t = 100T$

(e) $t = 200T$ (f) $t = 300T$ (g) $t = 400T$ (h) $t = 500T$

(i) $t = 600T$ (j) $t = 700T$ (k) $t = 800T$ (l) $t = 1000T$

FIGURE 1.22: Large-time evolution if the tracer field for the case of extremely small diffusivity $\epsilon = 10^{-5}$.

1.6 Example of the averaging procedure for 3D advection-diffusion equation.

Consider a volume-preserving system of ordinary differential equations

$$\frac{dx_i}{dt} = f_i(x_1, x_2, x_3, t), \quad i = 1, 2, 3 \tag{1.72}$$

that has a one-parameter spatial volume-preserving symmetry group. It was shown [22, 23] that there exists a local change of variables

$$x_i = \phi_i(z_1, z_2, z_3), \quad i = 1, 2, 3 \tag{1.73}$$

that transforms systems (1.72) to the form

$$\begin{aligned} \frac{dz_1}{dt} &= \frac{\partial H(z_1, z_2, t)}{\partial z_2} \\ \frac{dz_2}{dt} &= -\frac{\partial H(z_1, z_2, t)}{\partial z_1} \\ \frac{dz_3}{dt} &= k_3(z_1, z_2, t) \end{aligned} \tag{1.74}$$

If the latter is autonomous, H is a first integral of motion. Restricting ourselves to a class of autonomous advection fields (or those that allow separation of time and spatial variables), we consider the system

$$\begin{aligned} \frac{dz_1}{dF} &= \frac{\partial H(z_1, z_2)}{\partial z_2} \\ \frac{dz_2}{dF} &= -\frac{\partial H(z_1, z_2)}{\partial z_1} \\ \frac{dz_3}{dF} &= k_3(z_1, z_2) \end{aligned} \tag{1.75}$$

where F is time or parametric time as it was introduced in (1.13) - (1.17). The variables z_1 and z_2 of such a flow field are independent of z_3 and, therefore, the first pair of equations (1.75) can be transformed to action-angle variables such that $(z_1, z_2) \to (J, \theta)$. Moreover, there exists a subsequent transformation of variables $(J, \theta, z_3) \to (I, \phi_1, \phi_2)$ that takes

(1.75) into the form

$$\begin{aligned} \dot{I} =& 0 \\ \dot{\phi_1} =& \Omega_1(I) \\ \dot{\phi_2} =& \Omega_2(I) \end{aligned} \tag{1.76}$$

For details of this transformation, see [22]. Here "˙" denotes the derivative with respect to time or the derivative with respect to parametric time as discussed above, depending on the properties of the original flow. Summing up, there exists a way to write streamlines equations for a volume-preserving flow (1.72) in action-angle-angle coordinates form as given by (1.76).

In order to demonstrate the application of methods developed in section 1.3 to a three dimensional problem, let us now consider (with some loss of generality) a particular flow u in action-angle-angle form

$$\begin{aligned} u &= \{u_r, u_\theta, u_z\} \\ u_r &= 0 \\ u_\theta &= -\omega(r) f(t) \\ u_z &= -\Omega(r) g(t). \end{aligned} \tag{1.77}$$

Both $f(t)$ and $g(t)$ are assumed to be time-periodic functions with period T. Coordinates θ and z are periodic angular coordinates. The solution of these equations of motion is

trivial and given by

$$\begin{aligned} r &= r_0 \\ \theta &= \theta_0 - \omega(r)F(t) \\ z &= z_0 - \Omega(r)G(t), \end{aligned} \tag{1.78}$$

where $F(t) = \int_0^t f(t')dt'$ and $G(t) = \int_0^t g(t')dt'$. (Compare these expression to (1.13) and (1.21)). Expressions for the stream-lines therefore are

$$\begin{aligned} \bar{r} &= r \\ \bar{\theta} &= \theta + \omega(r)F(t) \\ \bar{z} &= z + \Omega(r)G(t). \end{aligned} \tag{1.79}$$

Using (1.77), the 3D counterpart of the original advection-diffusion equation (1.7) is rewritten as:

$$c_t - \omega(r)f(t)c_\theta - \Omega(r)g(t)c_z = \epsilon\Delta c. \tag{1.80}$$

Here Δ is the three-dimensional Laplacian in cylindrical coordinates

$$\Delta = \frac{1}{r}\frac{\partial}{\partial r}\Big(r\frac{\partial}{\partial r}\Big) + \frac{1}{r^2}\frac{\partial^2}{\partial\theta^2} + \frac{\partial^2}{\partial z^2} \tag{1.81}$$

Now, we use the stream-lines (1.79) as new coordinates via the transformation $c(t, r, \theta, z) \to v(t, r, \bar{\theta}, \bar{z})$ with the following transformation rules:

$$\begin{aligned} c_t &= v_t + v_{\bar{\theta}}\omega f + v_{\bar{z}}\Omega g \\ c_\theta &= v_{\bar{\theta}}, \quad c_{\theta\theta} = v_{\bar{\theta}\bar{\theta}}, \quad c_z = v_{\bar{z}}, \quad c_{zz} = v_{\bar{z}\bar{z}} \\ c_r &= v_r + v_{\bar{\theta}}\omega' F + v_{\bar{z}}\Omega' G \\ c_{rr} &= v_{rr} + v_{\bar{\theta}}\omega'' F + v_{\bar{z}}\Omega'' G + \\ &\quad + 2v_{r\bar{z}}\Omega' G + 2v_{r\bar{\theta}}\omega' F + 2v_{\bar{\theta}\bar{z}}\Omega'\omega' FG + \\ &\quad + v_{\bar{\theta}\bar{\theta}}(\omega' F)^2 + v_{\bar{z}\bar{z}}(\Omega' G)^2 \end{aligned} \tag{1.82}$$

Using the transformation rules (1.23), we can rewrite the original equation (1.7) in the form

$$\begin{aligned} v_t = \epsilon\Big[\Delta v + F\Big(\Big(\frac{\omega'}{r} + \omega''\Big)v_{\bar{\theta}} + 2\omega' v_{r\bar{\theta}}\Big) + G\Big(\Big(\frac{\Omega'}{r} + \Omega''\Big)v_{\bar{z}} + 2\Omega' v_{r\bar{z}}\Big) + \\ + (F\omega')^2 v_{\bar{\theta}\bar{\theta}} + (G\Omega')^2 v_{\bar{z}\bar{z}} + 2FG(\omega\Omega')v_{\bar{\theta}\bar{z}}\Big] \end{aligned} \tag{1.83}$$

This equation is written in the form suitable for averaging. It is straightfroward to notice that the structure of this equation is exactly the same as the structure of equation (1.70). Obviously, because of the extra dimension, there are more terms in the equation. However, again, we were able to hide all time-dependence of the advection field into the coefficients of the linear operator. This equation can now be averaged using the technique developed in previous sections. Obviously, the averaged approximate equation to leading order in ϵ is given by replacing the time-dependent coefficients in (1.83) by their time averages. In order to obtain the next order correction and rewrite it in a compact form, we introduce

operators

$$\begin{aligned}
\mathcal{L}_f &\equiv \Big(\frac{\omega'}{r} + \omega''\Big)\partial_\theta + 2\omega'\partial_r\partial_\theta \\
\mathcal{L}_g &\equiv \Big(\frac{\Omega'}{r} + \Omega''\Big)\partial_z + 2\Omega'\partial_r\partial_z \\
\mathcal{L}_{ff} &\equiv (\omega')^2\partial_\theta\partial_\theta, \quad \mathcal{L}_{gg} \equiv (\Omega')^2\partial_z\partial_z \\
\mathcal{L}_{fg} &\equiv 2\omega'\Omega'\partial_\theta\partial_z
\end{aligned} \tag{1.84}$$

We also introduce variables

$$\begin{aligned}
A_f &= F, \quad A_g = G \\
A_{ff} &= F^2, \quad A_{gg} = G^2 \\
A_{fg} &= 2FG
\end{aligned} \tag{1.85}$$

And, finally, we need to introduce functions

$$B_\alpha(t) = \int_0^t (A_\alpha(t') - \langle A_\alpha \rangle)dt' \tag{1.86}$$

Here, α takes the values (f, g, ff, gg, fg). Using these notations, we write a first order averaged equation in the form

$$V_t = \epsilon\Big(\Delta V + \sum_\alpha \langle A_\alpha \rangle \mathcal{L}_\alpha\Big) \tag{1.87}$$

(Compare it to (1.53)). At second order in ϵ, the equation is

$$\begin{aligned}
V_t = \epsilon\Big(\Delta V + \sum_\alpha \langle A_\alpha \rangle \mathcal{L}_\alpha\Big) + \\
+ \epsilon^2\Big\{\sum_\alpha \langle B_\alpha \rangle [\Delta, \mathcal{L}_\alpha] + \sum_{\alpha,\beta}\Big(\langle A_\alpha B_\beta \rangle - \langle A_\beta \rangle \langle B_\alpha \rangle \mathcal{L}_\alpha \mathcal{L}_\beta\Big)\Big\}
\end{aligned} \tag{1.88}$$

This equation is to be compared to the analogous result obtained for the 2D problem (1.88).

Chapter 2

Numerical simulation of advective-diffusive transport in the ocean.

2.1 Introduction

The problem of transport of a scalar field is of extreme practical and theoretical interest in many scientific and engineering areas, such as astrophysics [24, 25], plasma physics [26, 27], turbulence [28] and, of course, ocean/atmospheric science [29, 30]. One of the key problems of physical oceanography is the problem of transport and mixing of large water masses (and at the same time transport of heat, salinity, plankton, contaminants and other *tracers*). These processes may be important in many ways: they may have great

influence on the weather and climate, flora and fauna environments, affecting biodiversity of a region and eventually having impact on public health and wellness.

In many cases it is reasonable to assume that the advected physical entity does not affect its fluid environment. If additionally the particle being advected inherits the phase velocity of the flow without any significant retardation such that inertial effects can be neglected, they are said to be *passive scalars* (or tracers, or Lagrangian particles) and the advective flow itself is called *passive advection*. A passive scalar has extremely simple equation of motion:

$$\frac{d\vec{r}}{dt} = \vec{v}(\vec{r}, t), \tag{2.1}$$

where $\vec{r} = (x, y, z)$ and $\vec{v} = (u, v, w)$ are the position of the particle and the velocity vector. Traditionally, in physical oceanography, the problem is considered to be solved when the hydrodynamical equations of motion are solved. However, from the point of view of the scalar transport problem, this is just the beginning. The Eulerian velocity field is taken as a known parameter and then is used in the right-hand side of equation (2.1) for passive tracers dynamics. The velocity field can be found experimentally, from some kinematic considerations or as a result of numerical simulations of some more or less consistent flow models. Therefore, the dynamics of tracers is described by a system of nonlinear differential equations with fully determined right-hand side.

In the real physical systems, in particular in oceanic flows, advective dynamics is always accompanied by diffusion. First of all, this is due to simple molecular diffusion that only depends on the nature of the passive tracer. However, typical fluid flows of geophysical

interest are usually extremely complex and are excited on a very wide range of length and time scales. This has great impact on the choice of spatial disctretization of the grid used in computations (and therefore on the temporal disctretization due to stability requirements of the numerical scheme being used). Therefore, very often, small scale fluctuations are integrated out such that the resulting mean velocity field is smoothed on the typical scales of interest but is still realistic enough to possess some of the real physical properties of the flow. This averaging or *homogenization* leads to an effective change of the diffusion [2, 3]. The new effective, renormalized diffusion incorporates not only regular molecular diffusion but also effects of averaging small-scale excitations of the flow. This approach was successfully applied to a wide range of problems, including flows in porous media [31, 32], composite materials [33, 34], turbulence [35, 36] etc...

It is well known from the theory of dynamical systems that even solutions of such averaged and "smooth" systems can exhibit chaotic behavior in the sense that they are exponentially sensitive to small variations of initial conditions. Dynamical chaos is not an exotic phenomenon and is well studied, found even in extremely simple systems and observed in laboratory experiments. In the main part, the motion in phase space is determined by so-called invariant manifolds — geometric structures that are the subject of study in the theory of nonlinear dynamics, such as stationary points, attractors (including strange attractors), invariant tori and so on. In fluid dynamics these structures are sometimes called *Lagrangian structures*. There exists an extensive literature regarding the subject (for example see [37–39]). Analysis of Lagrangian coherent structures emerging from solutions of equations of motion (2.1) gives a lot of useful information about transport

barriers, zones of active mixing etc...

The term "transport" itself in the theory of dynamical systems is simply used to describe the motion of fluid particles in space. It is measured quantitatively by the total mass of liquid passing through a given cross-section of the flow and spatial sizes of the tracer field distribution. Under certain circumstances, the material surfaces formed by the trajectories of the liquid particles emerge in the flow. They act as barriers to transport effects since trajectories of liquid particles cannot cross these surfaces. For example, so-called Kolmogorov-Arnold-Moser (KAM) tori act as absolute barriers for transport [37–39].

The term "mixing" is one of the key concepts in fluid dynamics and can be given a rigorous mathematical definition. Consider a basin A and some contaminant occupying region B such that at $t = 0$ it's volume is $V(B_0)$. For some different region $C \in A$ the amount of contaminant in it at some time t is $V(B_t \cap C)$ and the concentration of contaminant is $V(B_t \cap C)/V(C)$. Full mixing occurs at time t when for $\forall C$ in A, the concentration of the contaminant in C is the same as in the whole A: $V(B_t \cap C)/V(C) = V(B_0)/V(A)$.

The Gulf Stream — a powerful warm Atlantic ocean flow that starts at the southern end of Florida and extends all the way to northern Europe — has an enormous effect on the climate of the American east coast and Europe. The Gulf Stream gathers all of its water from the warm tropical Caribbean basin and the Gulf of Mexico and transports it along its path affecting climate of surrounding areas, making them warmer and more hospitable. Florida, for example, has mild and warm weather all year round. The Gulf

Stream has a huge impact on the climate in Europe. It brings large masses of warm water into the North Atlantic Current and keeps European waters much warmer than they would be otherwise at such a high latitude. For example, the average low in London in December is 42°F (5°C) while in St. Johns, Newfoundland, the average is 27°F (-3°C). The Gulf Stream and its warm winds and waters also have a big impact on the weather system of Scandinavian mountains, keeping Norway's coast free of ice and snow. Besides this, the warm waters of the Gulf Stream increase rates of cyclones generation and affect their strengths.

This is just one (probably the most famous) example of how important a meso-scale ocean phenomena can be. Together with the Gulf Stream there are many other different instances of meso- and submeso-scale *coherent structures* in the ocean. Coherent structures are stable formations that exist on time scales that are sufficiently larger then any local Eulerian time characteristics of the flow. Modern techniques of flow visualization by means of buoys, drifters, satellite imaging help to reveal and explore large-scale coherent structures such as planetary scale eddies, major streams, meso-scale eddies and rings, meanders and so on. On Fig. 2.1, a satellite image of the western North Atlantic (courtesy of NASA) is presented. We clearly see large formations such as mesoscale swirls and meanders.

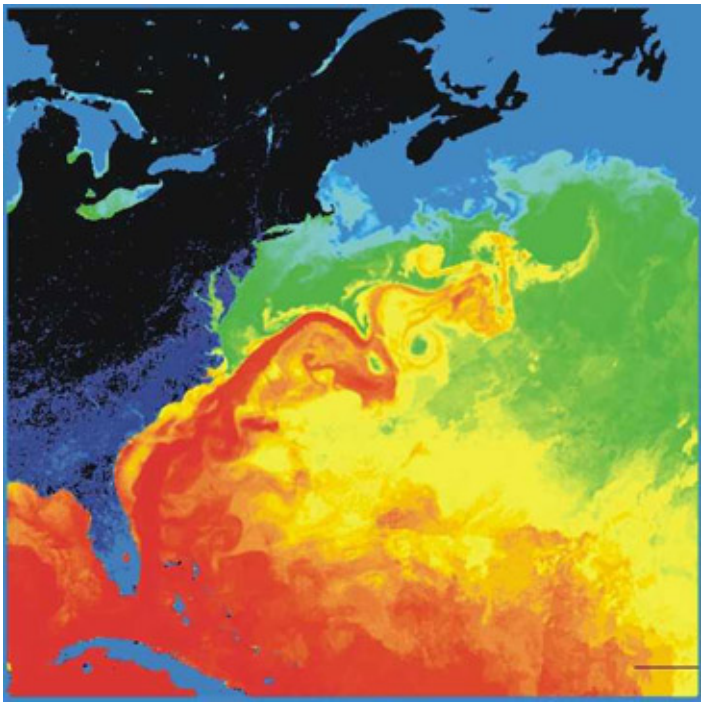


FIGURE 2.1: Distribution of surface temperature in the western North Atlantic. Source: NASA.

2.2 Results of numerical simulations for HYCOM ocean flows.

In this work we want to suggest a tool that allows visualization and the study of advective-diffusive dynamics of some self-consistent velocity field that comes from either an analytical solution of the equations of motion, from experiments, or as a result of numerical simulations. Once the external velocity field (possibly extremely complicated) given as a parameter is known in one or another form (as an analytic expression or in the form of a space-time grid) — what are the possible approaches to analyze it, study the structure of the field, or make conclusions about coherent Lagrangian structures that might or might

not exist? Figure 2.2 shows the structure of a typical ocean flow. One of the most typical approaches is to simulate a large number of hard particles that are advected by the velocity field which is known as the method of smoothed-particle hydrodynamics [40, 41]. This method however requires the simulation of an enormous number of particles to successfully resolve and visualize complicated flows and therefore is very computationally demanding. Instead we directly solve the advection-diffusion equation (equation (2.1) with an extra diffusive term and possibly source terms on the right-hand side) as a Cauchy problem for given initial condition.

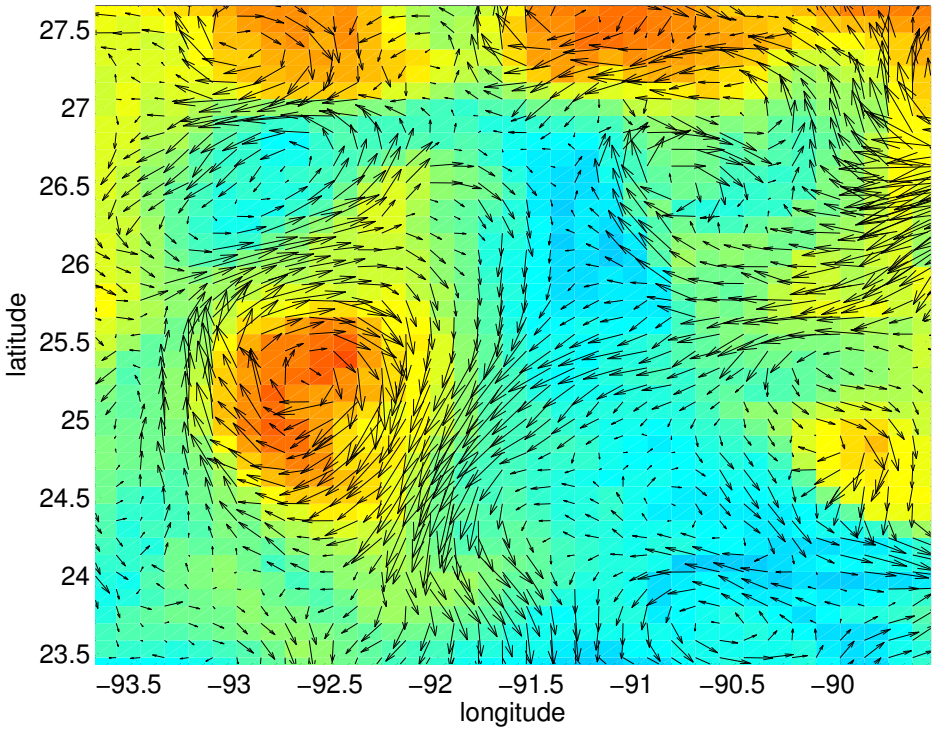


FIGURE 2.2: A snapshot of the horizontal vector field generated by HYCOM and used as a parameter in the simulations of advective-diffusive transport. Color indicates temperature. One can see multiple cyclone-anticyclone dipoles.

In our simulation, as an example of such an advective field, we used the tree-dimensional

velocity field generated by HYbrid Coordinate Ocean Model (HYCOM). The HYCOM model is a data-assimilative hybrid isopycnal-sigma-pressure (generalized) coordinate ocean model. The hybrid coordinate is one that is isopycnal in the open, stratified ocean, but smoothly reverts to a terrain-following coordinate in shallow coastal regions, and to z-level coordinates in the mixed layer and/or unstratified seas. The HYCOM effort is funded by the National Ocean Partnership Program (NOPP), as part of the U. S. Global Ocean Data Assimilation Experiment (GODAE). HYCOM data is available at HYCOM dataserver [42] in NetCDF format [43] on a longitude-latitude grid with the step of $1/12^{\circ}$.

The numerical scheme used for these computations was based on the MPDATA (multidimensional positive definite advection transport algorithm) which we discuss in Chapter 3. Diffusion was simulated by a usual second-order accurate finite difference scheme. The general algorithm that we used for simulations incorporates the following procedure:

- Read velocity data from HYCOM data files.
- Interpolate all three components to a finer grid using a B-spline routine (IMSL library was used [44]).
- Partition prepared data for parallel computations. The decomposition is done in the $X - Y$ directions only such that each computational core handles tree dimensional arrays: its portion of the $X - Y$ grid and all of the values in Z direction. Interaction between domains is implemented using a standard approach known as "halo exchange".

- Initialize initial state of a scalar tracer field and partition it for parallel computations.
- On each computational core perform the first MPDATA donor-cell step (MPDATA algorithm is discussed in the section 3).
- On each computational core compute anti-diffusive velocities and perform corrective MPDATA donor-cell step.
- On each computational core perform diffusive step.
- Write result to the file and repeat the computations proceeding to the next upwind donor-cell step.

This algorithm was implemented in Fortran with the parallel part written with Co-array Fortran (CAF). Co-Array Fortran is a PGAS (Partitioned Global Address Space) language, a class of parallel programming languages. It was created by Robert Numrich and John Reid in 1990s to be the smallest change required to convert Fortran into a robust and efficient parallel language [45, 46]. It allows one to efficiently write highly parallel codes following the SPMD (single program, multiple data) parallelization scheme. Each process (called image) has its own private variables. Variables which have a so-called *co-dimension* are addressable from other images. This extension is part of the Fortran 2008 standard (ISO/IEC 1539-1:2010).

2.2.1 Deep Water Horizon

First, we present the results of the simulation of an advective-diffusive transport with a line source term. We place the source at the coordinates of the Deep Water Horizon — oil drilling rig notorious for its failure in 2010 that caused the largest off-shore oil spill in the history of U.S. The source was placed at coordinates (28.737, -88.3869) from the surface to the bottom. Figure 2.3 depicts the set-up.

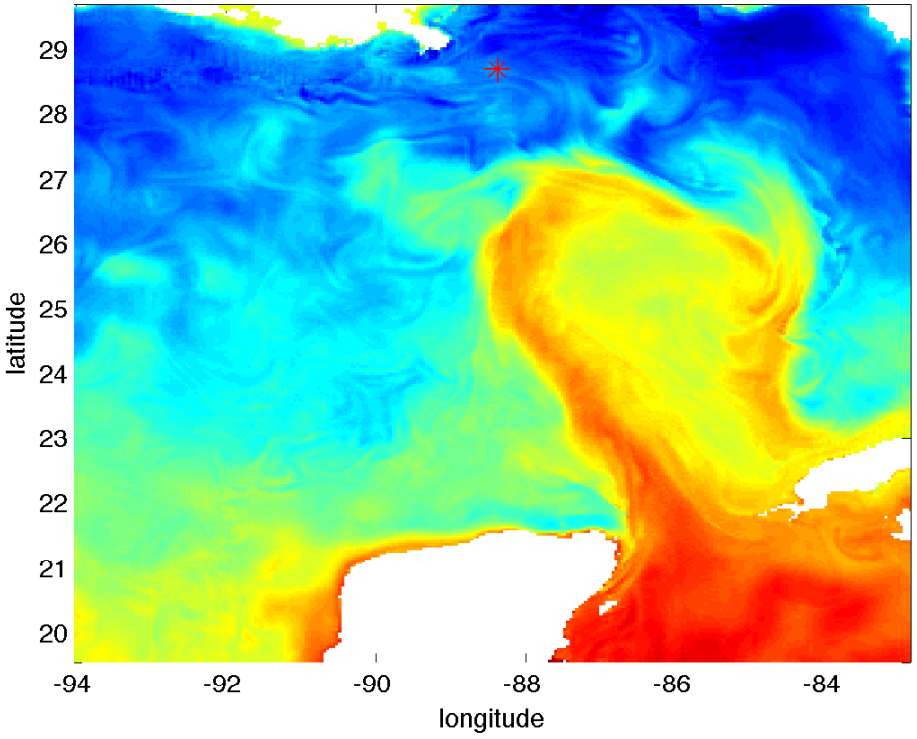


FIGURE 2.3: Deep Water Horizon set-up. Red asterisks denotes the position of the rig (28.737, -88.3869). Color indicates the temperature of the water.

We show the evolution of the tracer field generated by the line source on Fig. 2.5. One of the interesting questions is the effect of the vertical component of the velocity on the dynamics of the tracer. The HYCOM model computes velocities in the isopycnal coordinates

and the vertical component is obtained after interpolation onto the "longitude-latitude-depth" based grid. Usually, the vertical component is orders of magnitude smaller than the horizontal components, and it is important to estimate the impact of vertical velocity. We can easily set the vertical velocity to zero and run simulations with otherwise exactly the same parameters. In the absence of diffusion, the solution exhibits fully stratified behavior. On the Fig. 2.4 we show results of the simulations of the tracer's dynamics for the case when vertical velocities are completely suppressed, but there exists diffusive coupling along Z-axis. On the countrary, Fig. 2.5 presents the same system but with the vertical component of the velocity. It is easy to see that the effect of the vertical velocity is rather dramatic.

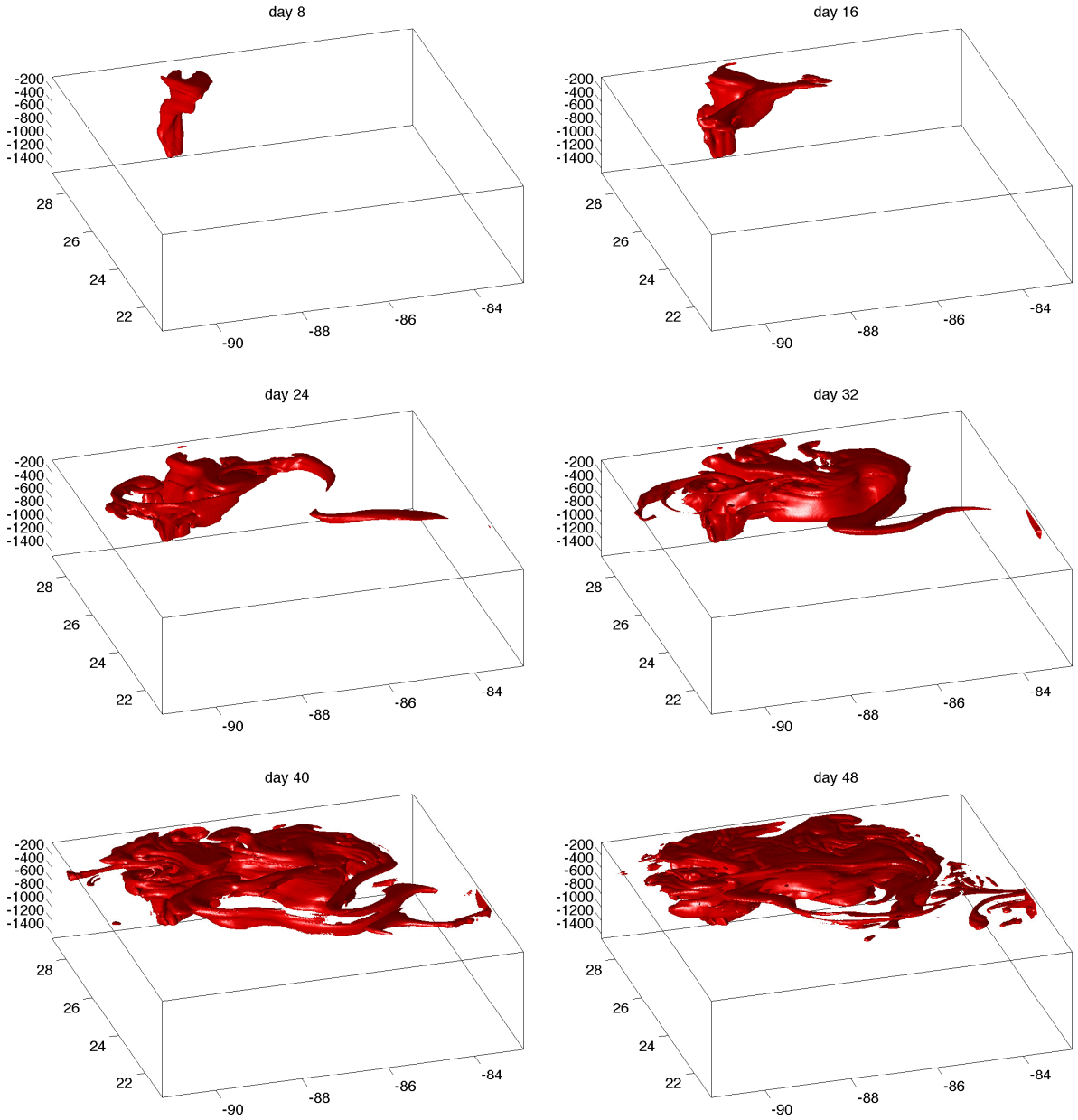


FIGURE 2.4: Evolution of the tracer field. Vertical component of the velocity field is turned off.

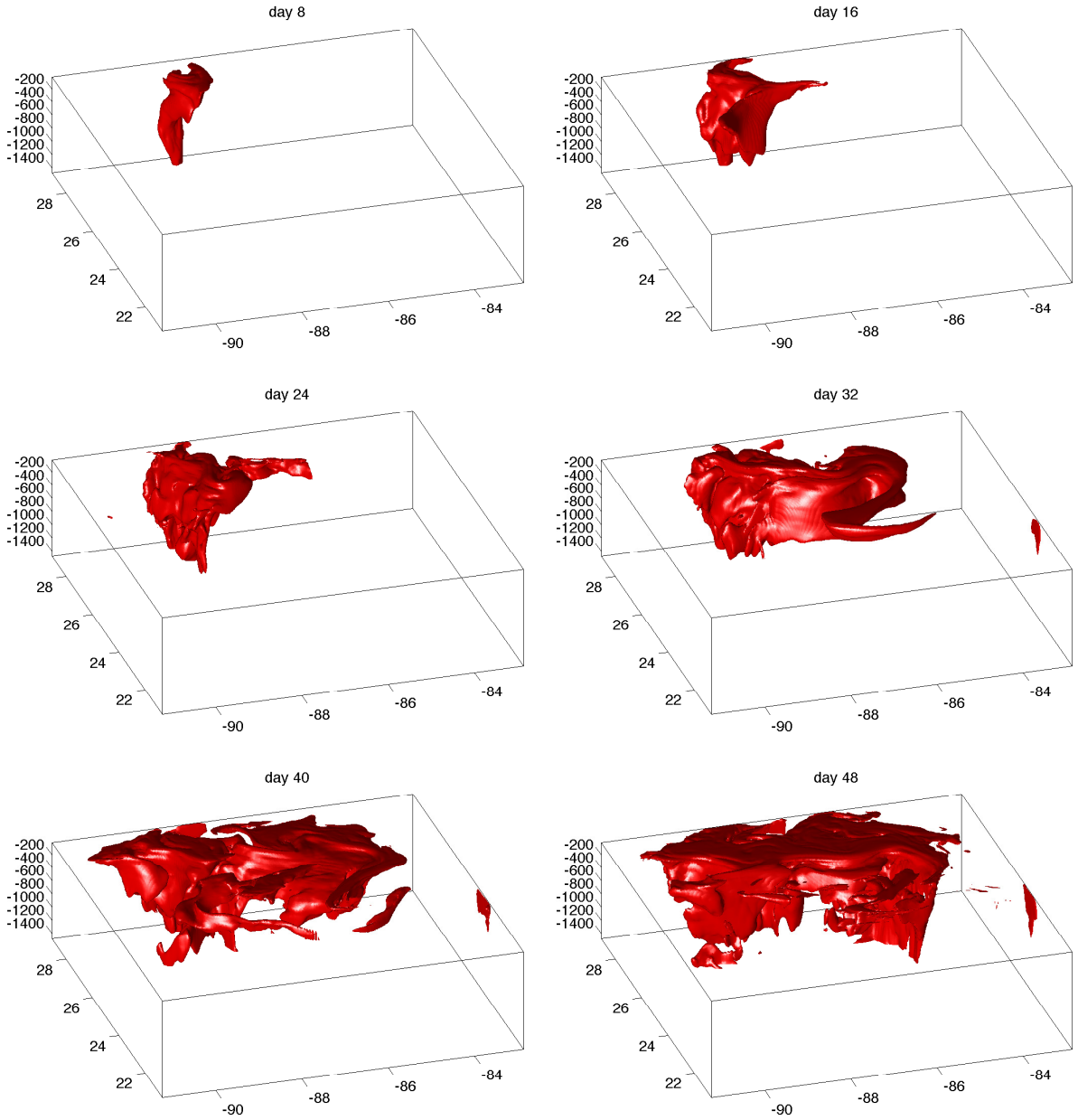


FIGURE 2.5: Evolution of the tracer field. Vertical component of the velocity field is present.

2.2.2 Loop Current

One of the most powerful currents in the Gulf of Mexico has the name of *Loop Current*. It originates in the channel between the Yucatan Peninsula and Cuba (as a Yucatan Current), flows northward, loops east and south and exits east through the Florida Straits joining the Gulf Stream. It plays an important role in hurricane creation [47, 48] and is known for the mechanism of creating large rings of warm water that pinch off the main flow and drift at speeds of about $5cm/s$ towards Texas or Mexico [49, 50]. We can study this current by setting a source of the tracer field on the path of the flow as depicted in Fig. 2.6.

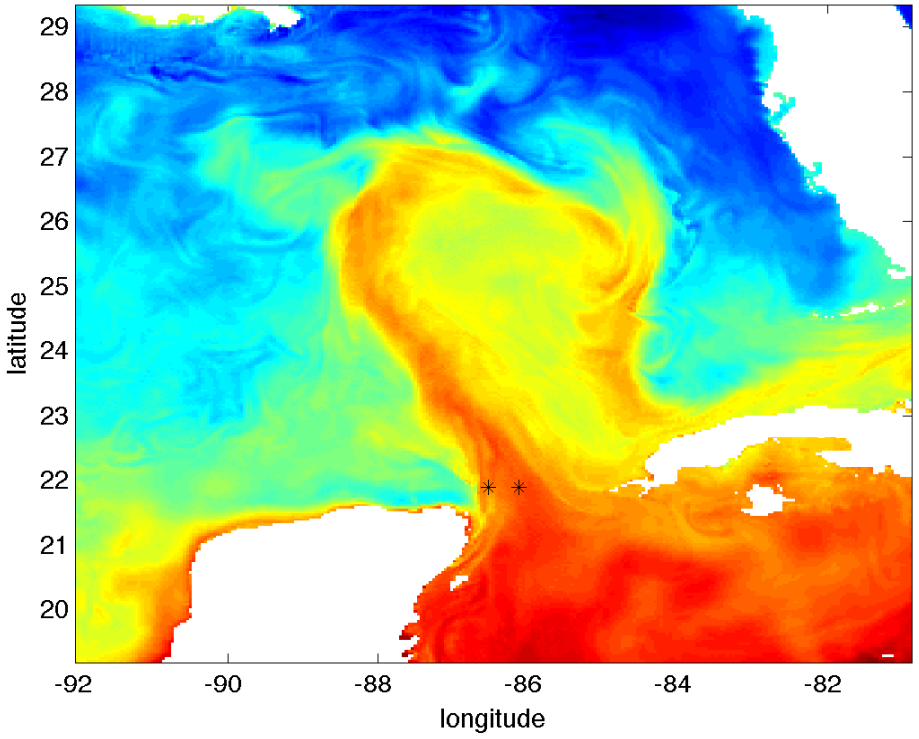


FIGURE 2.6: Loop Current set-up. The source is placed in the middle of the meso-scale flow entering the Gulf of Mexico between Yucatan Peninsula and Cuba — the Yucatan Current. Color indicates temperature of the water. Two black asterisks denote the position of the source term.

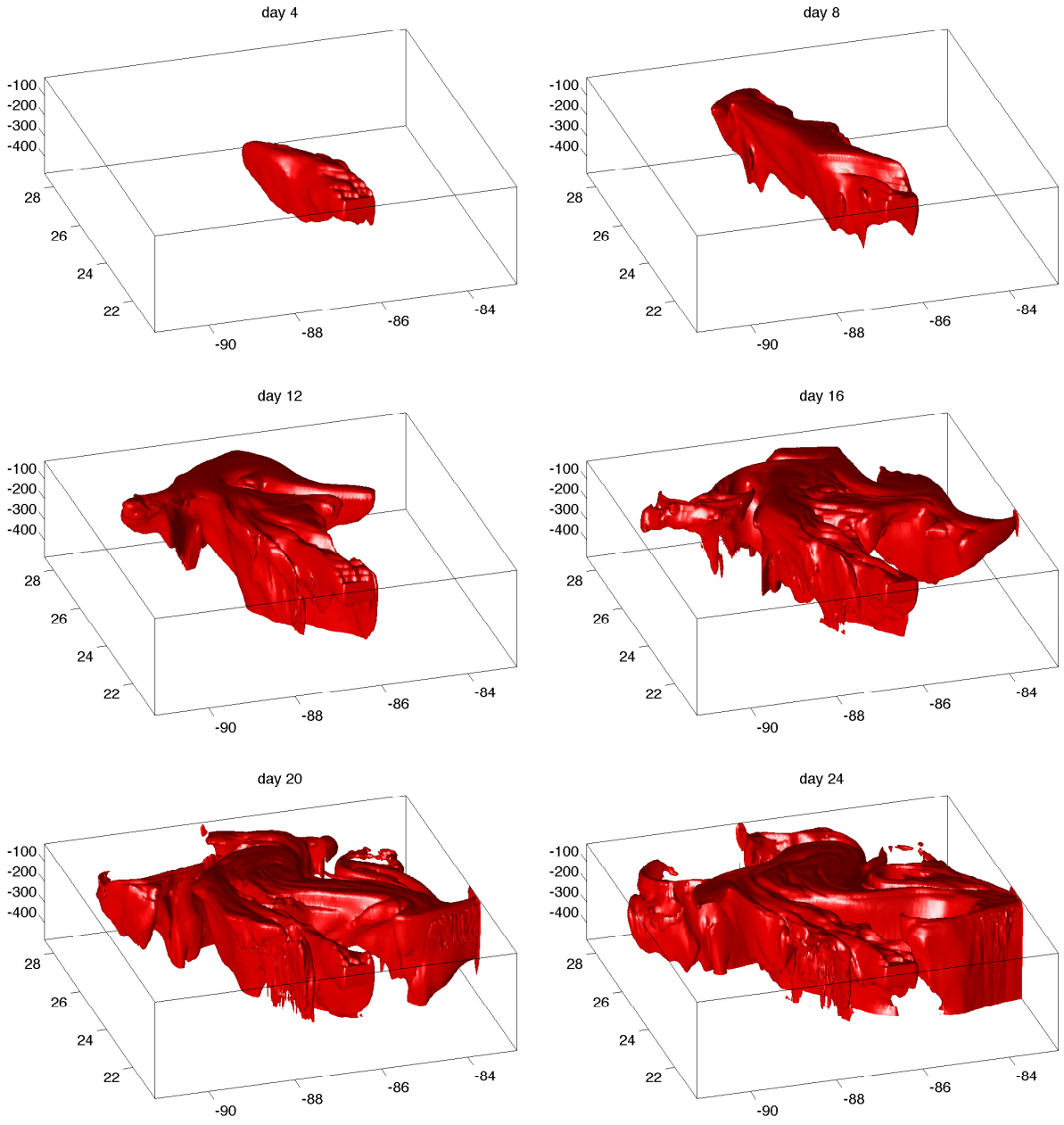


FIGURE 2.7: Advection in the Loop Current. Full 3D case with $v_z \neq 0$.

Results of these simulations are presented on Fig. 2.7 (for full 3D case with $v_z \neq 0$) and on Fig 2.8 (for stratified dynamics with v_z set to zero).

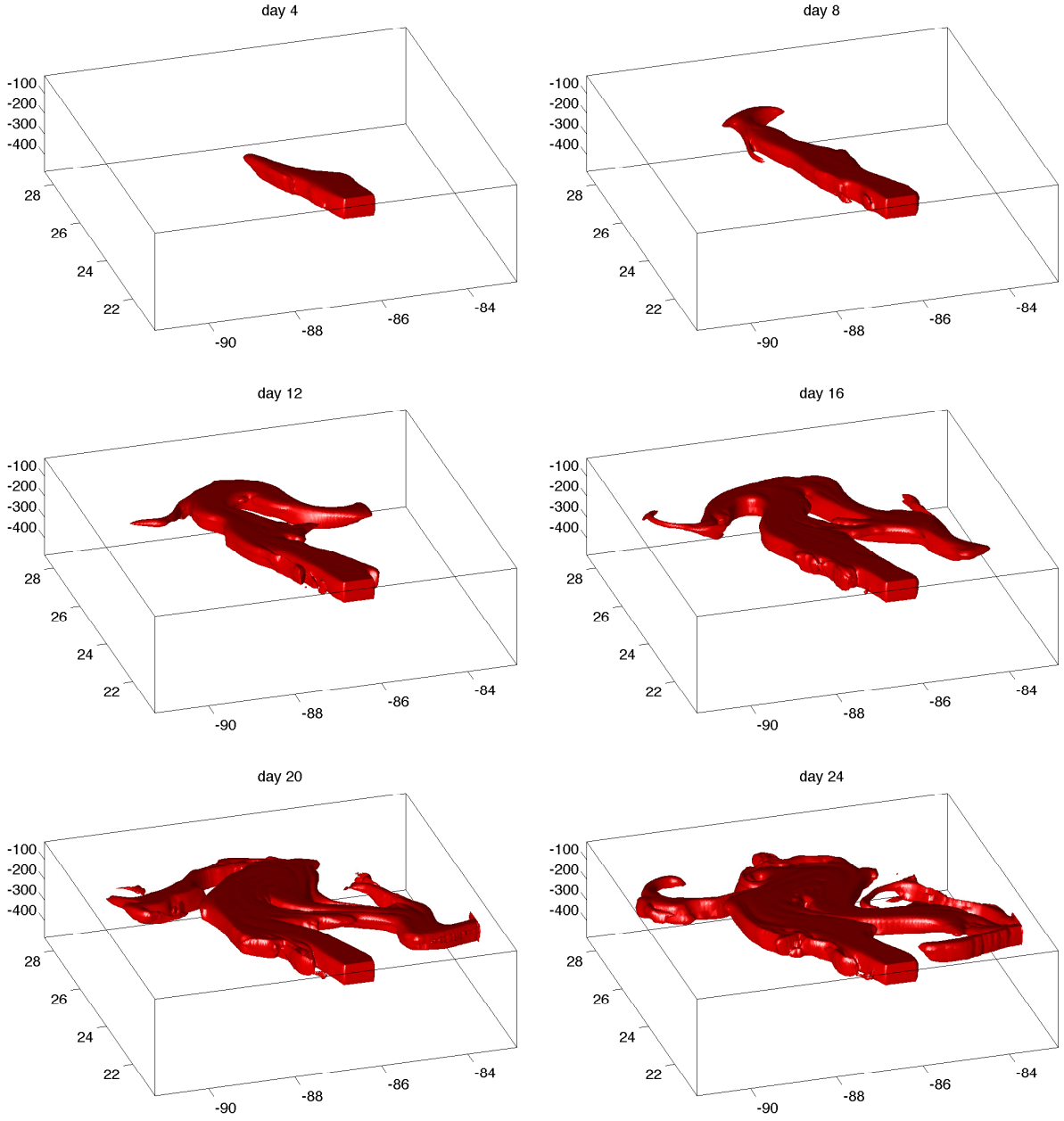


FIGURE 2.8: Advection in the Loop Current. Stratified case of 2D advection $v_z \neq 0$ and full 3D diffusion.

2.2.3 Interaction of two eddies.

Finally, we want to apply this method to the problem of mixing introduced by large scale structures such as large persistent flows of eddies. Large swirls that break off the Loop Current may reach up to about 200 kilometers across and may last for many months. Such structures drift across ocean basins at speeds of about 4-5 kilometers per day, and, depending on their source, stimulate mixing of either warmer or colder water than their surroundings. Besides the temperature, such eddies may facilitate mixing of water masses with different levels of salinity, or solutions of other chemicals (including pollutants) critical to long- and short-term climate variations and the health of the environment.

In this simulation we present the initial distribution of a scalar field and evolve it in time without additional source terms. To do this, we studied HYCOM data to find the cyclone-anticyclone dipole and carefully measured sizes of both eddies, Fig. 2.9. At time $t = 0$, we partially filled the volume of both eddies with contaminants. Evolution of the tracers that shows interaction between the two swirls is presented in Fig. 2.10.

In Fig. 2.11 we present the displacement of the center of mass of the initial distribution of the contaminant as a function of time in three directions. Obviously, for the case $v_z = 0$, the vertical displacement is exactly equal to 0 (Fig. 2.11(c)), whereas turning the vertical component on leads to substantial downward motion of the tracers (we observe that the contaminant of type "red" that was originally placed closer to the surface undergoes a much bigger displacement than the "blue" contaminant; the curve flattens after tracers reach a certain depth that may serve as an indication that there exists some kind of barrier

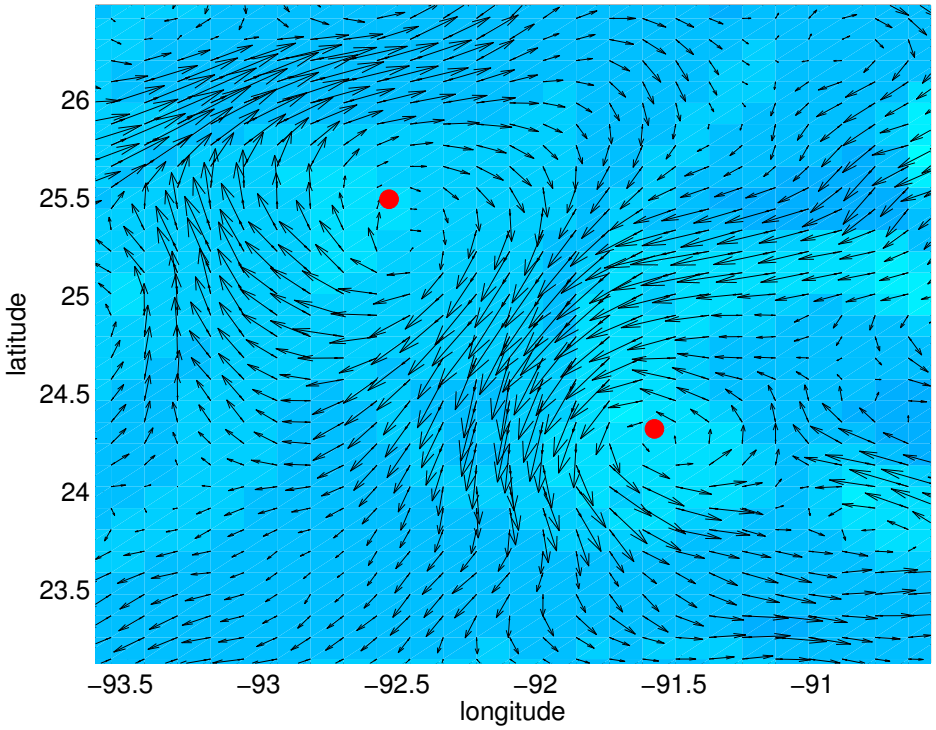


FIGURE 2.9: Example of the vector field with two interacting eddies. Slice for $z = 400m$. This cyclone-anticyclone system is clearly distinguishable from 20-30 to 1300m deep. Red dots depict centers of vortices.

for transport). On Fig. 2.11(a) and 2.11(b), we see that, for almost a month, a longitudinal motion of the "red" contaminant would have exhibited truly periodic properties without a vertical component of the flow field. Turning the vertical velocity on quickly kills these periodic patterns. At the same time we can conclude that the overall displacement of the center of mass in horizontal directions is not affected by the vertical component of the flow to a very big extent. These observations illustrate how the proposed method can be used to to visualize, study and analyze properties of various flows.

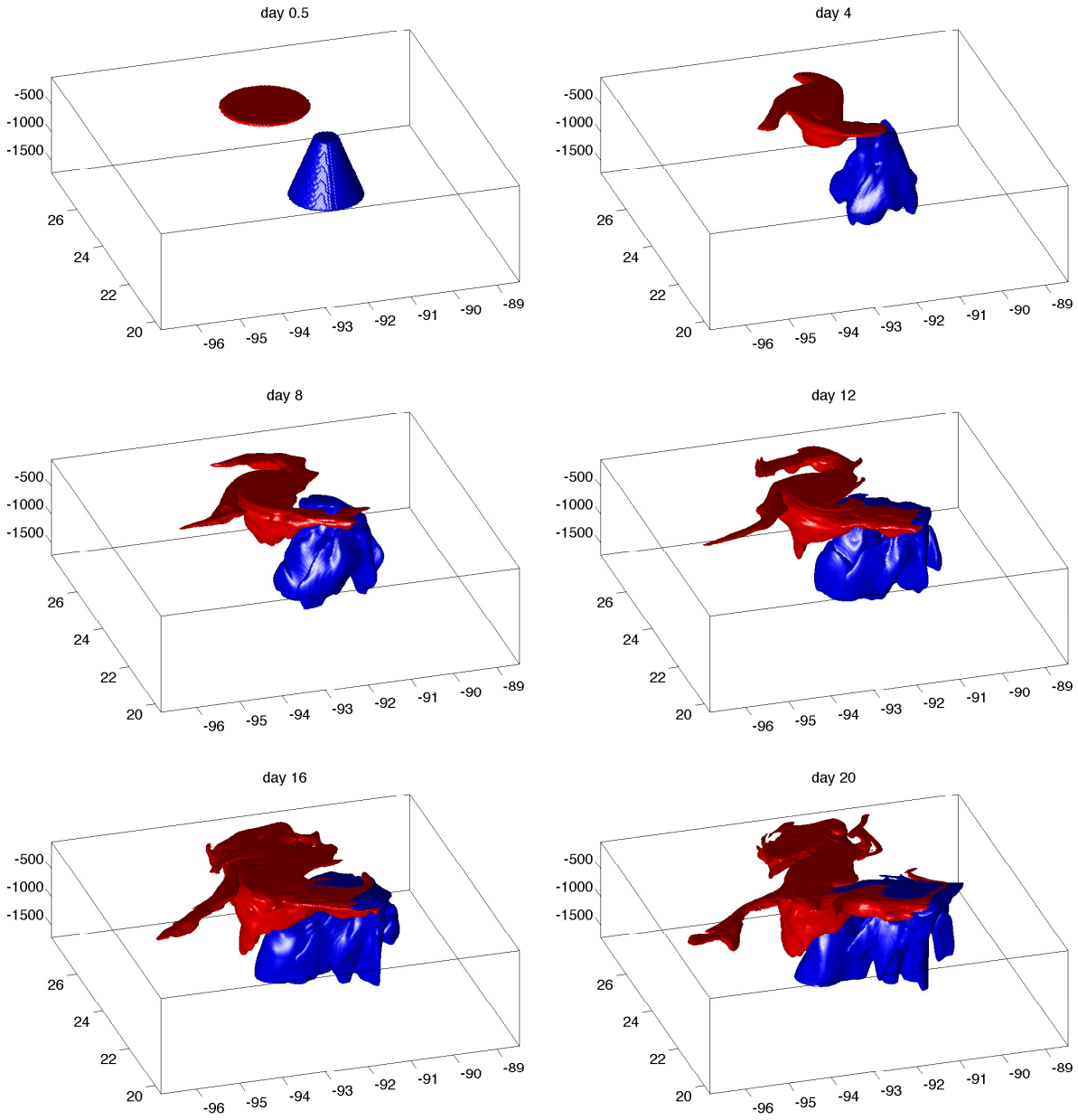
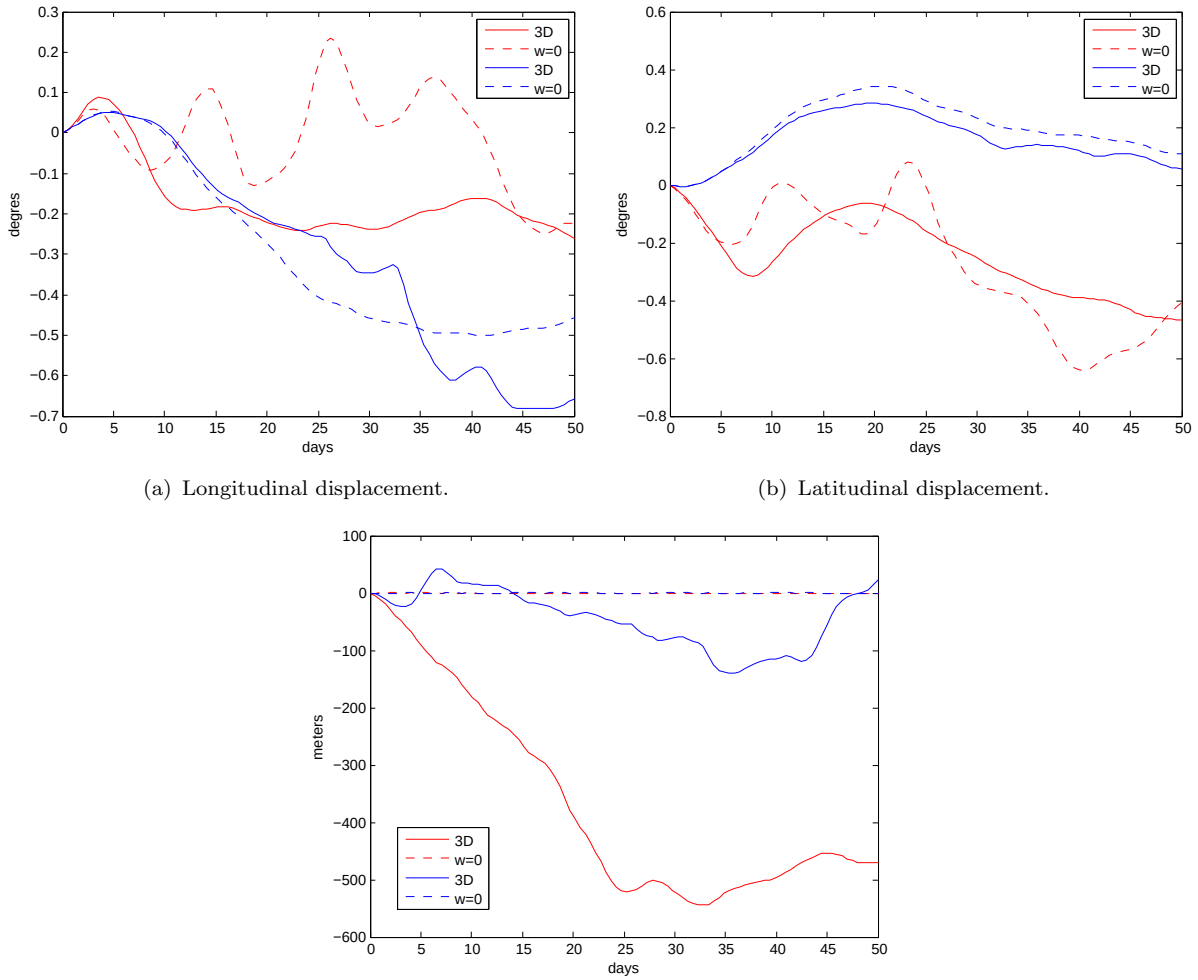


FIGURE 2.10: Interaction between two eddies with opposite vorticities. Full 3D case with $v_z \neq 0$.



(a) Longitudinal displacement.

(b) Latitudinal displacement.

(c) Displacement along Z axis

FIGURE 2.11: Displacement of the center of mass of the initial distribution of the contaminant in longitudinal (subfigure (a)), latitudinal (subfigure (b)) and vertical (subfigure (c)) directions as function of time. Red and blue colors correspond to red and blue tracers shown in Fig. 2.10. The solid line shows results of full 3D simulations (with $v_z \neq 0$) and dashed lines show the result of simulations for the case of no vertical velocities $v_z = 0$.

2.3 Co-array Fortran

Co-array Fortran (CAF) is one of three language extensions that support explicit parallel programming together with Unified Parallel C (UPC), developed at George Washington University [51] and Lawrence Berkeley National Laboratory [52], and Titanium — a

Java extension developed at University of California, Berkeley [53]. These programming languages are called Partitioned Global Address Space (PGAS) languages for that they have a common feature of partitioning the data and affinity of the data with processors.

The CAF model first appeared as a small extension of Fortran 90 (F^{--}) that would enable the programmer to create programs that run on multiple computing processors with minimal changes to the standard Fortran syntax. CAF is a *Single Program, Multiple Data* (SMPD) programming model, where all processors simultaneously execute the same program. In Co-Array terminology, these independent replications are referred to as *images*. SPMD is most often implemented using the Message Passing Interface (MPI) — a C library (with bindings to Fortran) that is unaware that the program has multiple instances and implements communication between the images by passing messages via library calls on both the sending and receiving image. In this model, both sending and receiving images must be aware of the communication, which creates additional burden for the programmer.

In CAF all off-processor (off-image) data is available via special bracket notation. At the same time, all data that is local to the image is accessible via standard Fortran syntax. The programmer is responsible for explicit data decomposition. Affinity between data and physical processors is established by a run-time system. All images execute the program asynchronously and all data and all computations are local to the image. The programmer is responsible for explicit synchronization. The programer accesses and moves remote data to local data through, and only through, explicit co-array syntax.

During the execution of the program, the number of images is fixed and is retrieved during run-time. The number of the images may be obtained using the function num_images() ≥ 1. Each image has its own index that is assigned during run-time: $1 \leq$ this_image() ≤ 1 num_images(). The declaration of co-arrays and regular arrays and their properties are presented in the Table 2.1

Regular array	Co-array
real :: x(n)	**real** :: y(n)[*]
Every image has one of these objects	Every image has one of these objects
The name is the same on all images	The name is the same on all images
The size is the same on all images	The size is the same on all images
Elements of the array in local memory are indexed by the dimension	Elements of the array in local memory are indexed by the dimension
Arrays on remote image are invisible	Arrays on remote image are visible through co-dimension

TABLE 2.1: Properties of co-arrays compared with regular arrays.

The only difference lies in the visibility of the data that is remote with respect to the particular image. Otherwise the programmer can treat regular arrays and co-arrays in a similar manner, simply keeping in mind that value[p] is the value on p-th processor. Similarly, the declaration **real** :: x(n)[*] means that there is an array x(n) on each of num_images() processors.

There can be more than one co-array dimension. Consider for example the following declaration: **real** :: **data**(nx,ny)[p,*]. This replicates the instance **data**(nx,ny) — a 2-dimensional

array of size $n_x \times n_y$ — on each image (as many as are available at run-time). The last co-dimension computes as num_images()/p. A schematic of the idea of the 2-dimensional decomposition is presented on the Fig. 2.12.

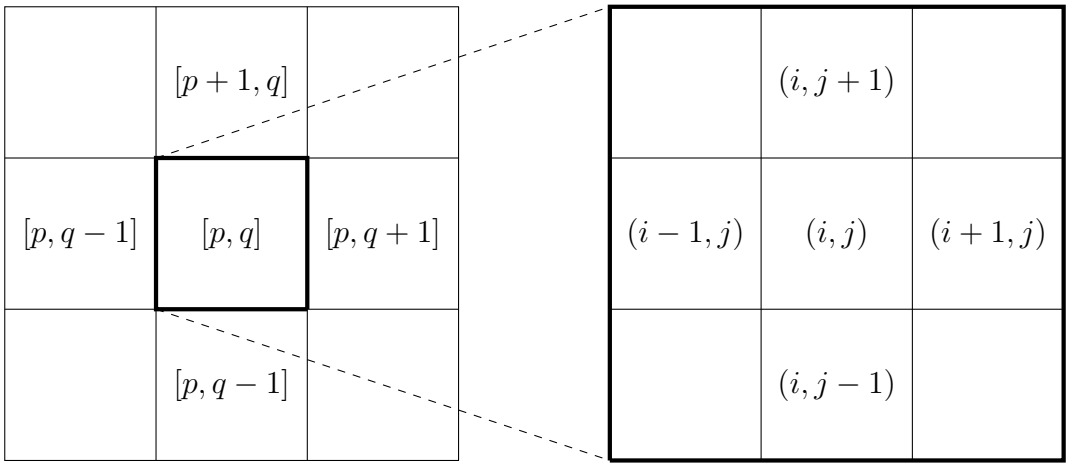


FIGURE 2.12: Decomposition of a two dimensional grid using co-arrays. Each image $[p, q]$ of $P \times Q$ total images handles a computational grid (i, j) of the size $n_x \times n_y$.

The code that was used for the solving advection-diffusion equation as described in the previous sections was based on exactly the same idea of 2D decomposition. All computational domains were split among multiple cores in such a way that one core handles one array **data**(nx,ny,nz) — a portion of the $X - Y$ domain and all of the values along the direction Z. The actual Fortran code that implements a similar 2D decomposition is shown on listing 2.1.

In conclusion we mention that since every image executes independently of all other images in an asynchronous manner, there is a need to control synchronization between images. Luckily, co-array Fortran provides the programmer with tools that allow to control execution of the code. This is done through barriers: *hidden barriers* (allocation

of a co-array automatically forces synchronization of images), *explicit barriers* — full (sync all) and partial (sync images(list (:))), *locks* and *critical regions*.

Listing 2.1: Example of 2D decomposition.

```
1   allocate(data_local(n,m)[q,p]) ! number of images must be p*q
2   if(this_image() == 1) then
3      allocate(data_global(p*n, q*m)) ! allocate global array on image = 1
4      call initialize(data_global)      ! call routine that will initialize
       it
5      do r=1,p
6         do s=1,q
7            data_local(1:n, 1:m)[r,s] = &
8                 data_global((r-1)*n+1 : r*n, (s-1)*m+1 : s*m)
9         end do
10     end do
11  end if
```

We illustrate the difference between the MPI-based implementation and CAF implementation of the halo exchange problem in Appendix A. It is seen that the co-array model indeed provides the programmer with a very elegant, easy-to-use and economical way to write highly parallel applications or to parallelize existing codes.

Chapter 3

Numerical Methods.

3.1 Introduction.

It is hard to under-estimate the role of numerical simulations in various fields: from engineering science and physics to finance [54, 55] and biology [56, 57]. A variety of problems that require a numerical approach includes numerical integration, nonlinear equations, differential equations, both ordinary and partial differential equations (PDEs) with complicated boundaries, signal processing, images processing and so on. The number of areas where numerical approximate methods find their use is limitless. There are many reasons for such great importance of numerical analysis. Very often, an analytical solution simply does not exist or it is extremely hard to obtain one. Simulations and numerical modeling are a very good and in many areas de-facto tool that can be used to check analytical results. Experimental verifications may be too expensive (aerodynamics,

automotive crash tests, etc...), or they may be simply impossible (one example of such an area is cosmology where numerical modeling is extensively used to test theories describing early stages of the universe [58–60]).

Numerical simulations constitute an entire self-contained branch of mathematics. Because of the extreme importance of numerical methods, there exists an enormous number of methods and techniques used for solving various mathematical problems on computers. Among others we are interested in computational approaches of solving PDEs. A detailed discussion of this problem alone is impossible within the scope of this work; however, we provide a brief discussion of some of the most famous numerical schemes. We first discuss possible ways to approximate derivatives of functions and give an overview of the so-called "Chebychev spectral" method. Then, we mention a few numerical schemes that could be used to integrate partial differential equation. And finally we discuss MPDATA — the more elaborate method that was used for simulations of the mass transport in the ocean.

3.2 Numerical differentiation.

In order to solve partial differential equations numerically one has to provide an efficient method of computing derivatives of the function. We discuss some typical ways to handle this task.

3.2.1 Finite Differences.

Our starting point is a very basic question: given a set of points x_i and a set of values of some function f at these points $f(x_i)$, what is the way to approximate the derivative of the function $f'(x)$? From here on, we consider a uniform grid $x_1, ..., x_N$, with $x_{i+1} - x_i = h$ for every i. The first and most obvious way to answer this question is to introduce a finite difference approximation

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \approx \frac{f(x+h) - f(x)}{h} \tag{3.1}$$

Using Taylor expansion $f(x + \Delta x) = f(x) + f'(x)\Delta x + 1/2 f''(x)(\Delta x)^2 + \mathcal{O}((\Delta x)^3)$ in the above equation we see that $f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + \mathcal{O}(\Delta x)$. This finite difference scheme correctly accounts for the first order of Δx term. The truncation error is proportional to $(\Delta x)^2$ and the approximation (3.1) is said to be a first order approximation. In order to get a better approximation, consider two Taylor series:

$$\begin{aligned} f(x + \Delta x) =& f(x) + f'(x)\Delta x + \frac{(\Delta x)^2}{2} f''(x) + \frac{(\Delta x)^3}{3!} f'''(x) + \mathcal{O}((\Delta x)^4) \\ f(x - \Delta x) =& f(x) - f'(x)\Delta x + \frac{(\Delta x)^2}{2} f''(x) - \frac{(\Delta x)^3}{3!} f'''(x) + \mathcal{O}((\Delta x)^4) \end{aligned}$$

Subtracting the second equation from the first, we get

$$f'(x) = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} + \mathcal{O}((\Delta x)^3). \tag{3.2}$$

We see that this finite difference equation correctly accounts for the term proportional to the $(\Delta x)^2$ and the truncation error is proportional to the third power of Δx. Therefore, this finite difference is called a second order approximation of the derivative. We will come back to discussions of finite differences later in this chapter.

3.2.2 Periodic grid. Spectral differentiation.

Another approach — spectral differentiation — can be developed for the approximation of a derivative of the function defined on a periodic grid. To introduce this method, let us first impose periodicity $u_1 = u_N$ and rewrite equation (3.2) in a slightly different form, namely

$$\begin{pmatrix} \\ f_1' \\ \\ f_2' \\ \\ \vdots \\ \\ \\ f_N' \\ \\ \end{pmatrix} = \frac{1}{h} \begin{pmatrix} & & & & \\ 0 & \frac{1}{2} & \cdots & & -\frac{1}{2} \\ & & & & \\ -\frac{1}{2} & 0 & \cdots & & \\ & & & & \\ & & \ddots & & \\ & & & & \\ & & \cdots & 0 & \frac{1}{2} \\ & & & & \\ \frac{1}{2} & & \cdots & -\frac{1}{2} & 0 \\ & & & & \end{pmatrix} \begin{pmatrix} \\ f_1 \\ \\ f_2 \\ \\ \vdots \\ \\ \\ f_N \\ \\ \end{pmatrix}. \tag{3.3}$$

Here, we assume N to be even for simplicity. The matrix in (3.3) is a Toeplitz matrix with constants along diagonals that "wrap" around the matrix due to the periodicity of the problem [61, 62].

The same result may be obtained if $f(x)$ is locally interpolated by a polynomial. Let p_i for every i be a polynomial of a degree ≤ 2 such that $p_i(x_{i-1}) = f_{i-1}, p_i(x_i) = f_i$ and $p_i(x_{i+1}) = f_{i+1}$. Once p_i is found, we can obtain the first derivative as $f_i' = p_i'(x_i)$. For

the simple case (3.3), the sought polynomial is given by

$$p_i(x) = Af_{i-1} + Bf_i + Cf_{i+1},$$

where

$$\begin{aligned} A &= (x - x_i)(x - x_{i+1})/2h^2, \\ B &= -(x - x_{i-1})(x - x_{i+1})/h^2, \\ C &= (x - x_i)(x - x_{i-1})/2h^2. \end{aligned} \tag{3.4}$$

Taking the derivative of (3.4) at $x = x_i$, we obviously recover (3.2). In a similar way we can construct a matrix that will approximate derivatives to fourth order:

$$\begin{pmatrix} \\ f_1' \\ \\ f_2' \\ \\ \\ \vdots \\ \\ \\ \\ f_N' \\ \\ \end{pmatrix} = \frac{1}{h} \begin{pmatrix} & & \ddots & & & \frac{1}{12} & -\frac{2}{3} \\ & & \ddots & -\frac{1}{12} & & & \frac{1}{12} \\ & & \ddots & \frac{2}{3} & \ddots & & \\ & & \ddots & 0 & \ddots & & \\ & & \ddots & -\frac{2}{3} & \ddots & & \\ -\frac{1}{12} & & & \frac{1}{12} & \ddots & & \\ \frac{2}{3} & -\frac{1}{12} & & & \ddots & & \end{pmatrix} \begin{pmatrix} \\ f_1 \\ \\ f_2 \\ \\ \\ \vdots \\ \\ \\ \\ f_N \\ \\ \end{pmatrix}. \tag{3.5}$$

An analogous interpolation procedure may be performed with polynomials of the degree $\leq$ 4. This process can be continued to the infinite limit. In the end we should obtain a dense matrix that represents an infinite order accurate differentiation matrix. As discussed in [21], for a bounded periodic grid, a different interpolant (called the Fourier interpolant

for the choice of basic functions) should be used:

$$\begin{aligned} p(x) &= \sum_{i=1}^{N} f_i S_h(x - x_i), \quad \text{where} \\ S_h(x) &= \frac{h}{2\pi} \frac{\sin(\pi x/h)}{\tan(x/2)} \end{aligned} \tag{3.6}$$

The first derivative of the interpolant $S_h(x)$ at grid points $x = x_i$ is given by

$$S_h'(x_i) = \begin{cases} 0 & : \quad i = 0(\mathrm{mod} N) \\ (-1)^i \cot(ih/2)/2 & : \quad i \neq 0(\mathrm{mod} N) \end{cases} \tag{3.7}$$

Therefore, the first differentiation matrix is given by

$$D_N = \begin{pmatrix} 0 & & & -\frac{1}{2}\cot\frac{1h}{2} \\ -\frac{1}{2}\cot\frac{1h}{2} & \ddots & \ddots & \frac{1}{2}\cot\frac{2h}{2} \\ \frac{1}{2}\cot\frac{2h}{2} & \ddots & & -\frac{1}{2}\cot\frac{3h}{2} \\ -\frac{1}{2}\cot\frac{3h}{2} & & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{1}{2}\cot\frac{1h}{2} \\ \frac{1}{2}\cot\frac{1h}{2} & & & 0 \end{pmatrix}. \tag{3.8}$$

To compute the second derivative, we differentiate the interpolant (3.6) twice and obtain:

$$S_h''(x_i) = \begin{cases} -\frac{\pi^2}{3h^2} - \frac{1}{6} & : \quad i = 0(\mathrm{mod} N) \\ -\frac{(-1)^i}{2\sin^2(ih/2)} & : \quad i \neq 0(\mathrm{mod} N) \end{cases} \tag{3.9}$$

The double differentiation matrix therefore takes the form

$$D_N^{(2)} = \begin{pmatrix} & & \\ \ddots & \vdots & \\ & & \\ \ddots & -\frac{1}{2}\csc^2(\frac{2h}{2}) & \\ & & \\ \ddots & \frac{1}{2}\csc^2(\frac{h}{2}) & \\ & & \\ & -\frac{\pi^2}{3h^2} - \frac{1}{6} & \\ & & \\ & \frac{1}{2}\csc^2(\frac{h}{2}) & \ddots \\ & & \\ & -\frac{1}{2}\csc^2(\frac{2h}{2}) & \ddots \\ & & \\ & \vdots & \ddots \\ & & \end{pmatrix}. \tag{3.10}$$

We illustrate the accuracy of this method by computing the first and second derivatives of a periodic function defined on domain $[-\pi, \pi]$

$$\begin{aligned} f(x) &= \cos(x) e^{\sin(x)}, \\ f'(x) &= \left(\cos^2(x) - \sin(x)\right) e^{\sin(x)}, \\ f''(x) &= \cos(x) \left(\cos^2(x) - 3\sin(x) - 1\right) e^{\sin(x)}. \end{aligned} \tag{3.11}$$

Results of numerical computations of the first and second derivatives are presented in Fig. 3.1 and Fig. 3.2 respectively. We are plotting $f'(x)_{\text{analytical}} - f'(x)_{\text{numerical}}$ (and similarly for the second derivative). It is clear that good convergence can be reached already for 20-30 grid points for both first and second derivatives.

Another possible (and extremely popular) approach for approximating a derivative of a periodic function is, of course, the Fourier transform that is given by

$$\begin{cases} F(\omega) &= \int_{-\infty}^{+\infty} e^{-i\omega x} f(x)\, dx \\ f(x) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{i\omega x} F(\omega)\, d\omega \end{cases} \tag{3.12}$$

Differentiating the second expression with respect to x, we find that the Fourier transformation is given by $f'(x) \to i\omega F(\omega)$. The algorithm of finding the n^{th} derivative is trivial:

- transform $f(x)$ to Fourier space
- multiply $F(\omega)$ by $(i\omega)^n$
- transform the result back to the direct space

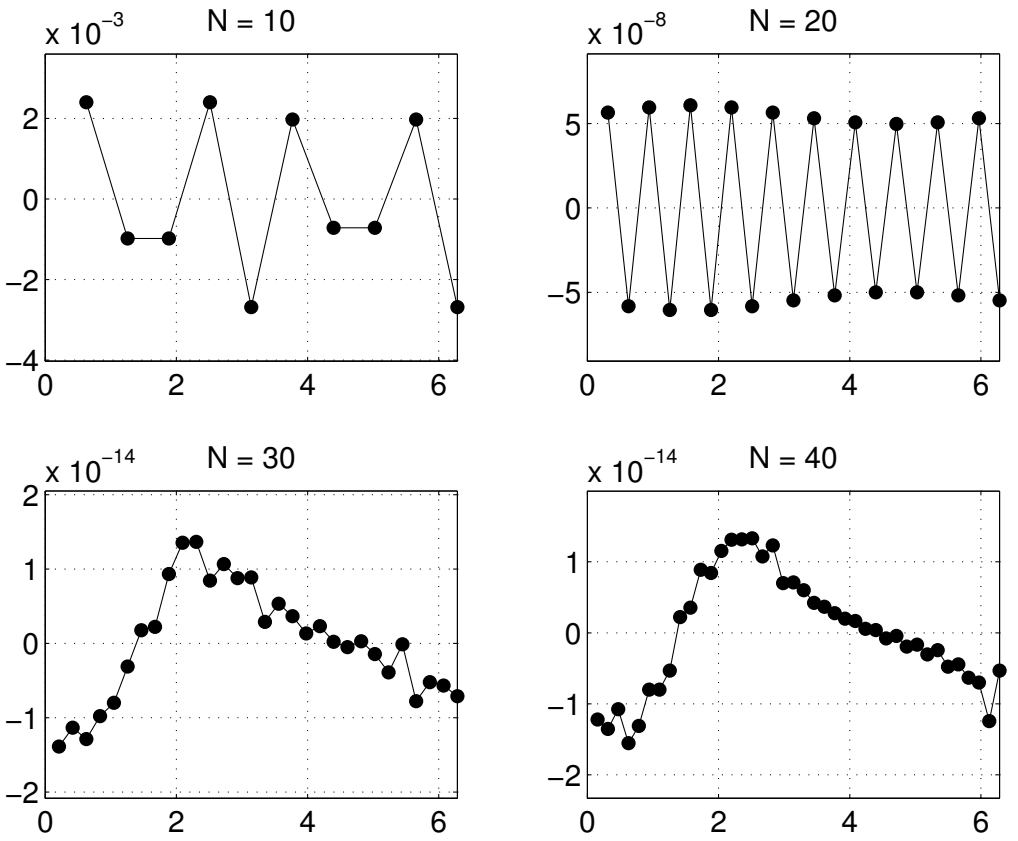


FIGURE 3.1: Numerical approximation of the first derivative on a periodic domain obtained by using the differentiation matrix (3.8). Here N is the number of grid points.

The discretized version of the Fourier transform — the discrete Fourier transform (DFT) — is probably the most often used algorithm in computational analysis [63, 64]. The reason for such great popularity is the fact that the direct and inverse DFT can be efficiently computed using the so-called fast Fourier transform that dates back to a famous paper published by John Tukey of Princeton University and John Cooley of IBM Research in 1965 [65]. The FFT algorithm allows one to perform the discrete Fourier transform in $N \log(N)$ operations whereas a direct usage of DFTs requires N^2 operations. One of the very widely used implementations of FFTs is "The Fastest Fourier Transform in the West", a software library, developed by Matteo Frigo and Steven G. Johnson at the

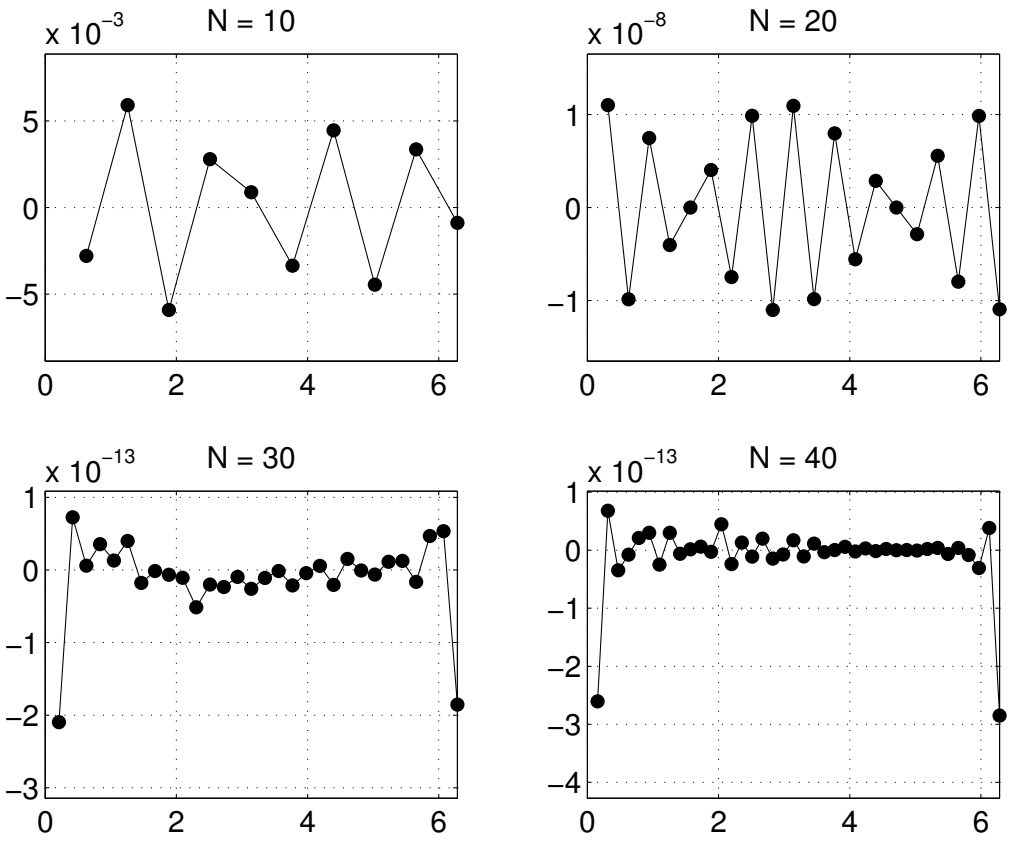


FIGURE 3.2: Numerical approximation of the second derivative on a periodic domain obtained using the differentiation matrix (3.10). Here N is the number of grid points.

Massachusetts Institute of Technology [66, 67]. This and related questions are discussed in a great details in [68]

3.2.3 Non-periodic domain. Chebychev differentiation matrices.

Suppose we need to evaluate a numerical approximation of a derivative of the function $f(x)$ for a function that is not periodic and is defined on a finite domain $[-1, 1]$. A first idea would be to try to periodically extend $f(x)$ onto whole $\mathbb{R}$ so that it becomes a piecewise

smooth, periodic function with jump discontinuities and to apply Fourier interpolation as described above. However, by introducing a discontinuity at the boundary, we loose accuracy which appears as an "overshoot" in the DFT representation of a function near the boundary. This effect is known as the Gibbs phenomenon [69, 70].

Another possible approach would be to use algebraic polynomials to interpolate $f(x)$. It also happens to be bad decision because polynomial interpolation is known to generate high-amplitude oscillations at the edges which is known as the Runge phenomenon [71, 72]. This divergence of the polynomial interpolation has even more dramatic effects than the Gibbs effect, however there are some techniques developed to suppress Runge divergence (for example see [73] or [74]).

An alternative approach is to use polynomial interpolation on a non-uniform grid. Different non-uniform grids were suggested [75], however one of the most popular ones is a grid of so-called "Chebychev points" or "Chebychev nodes", that are given by

$$x_j = \cos(j\pi N), \quad j = 0, 1, \cdots, N. \tag{3.13}$$

Geometrical interpretation of the set x_j of Chebychev nodes is presented on Fig. 3.3.

We see that grid points cluster around domain edges which has an dramatic effect on the accuracy of polynomial interpolation [19, 21]. In fact, polynomial interpolation on a Chebychev grid (or on any grid with asymptotic $N \to \infty$ density of points on the interval $[-1, 1]$ behaving like $1/\sqrt{1-x}$) is completely free of Runge's phenomenon.

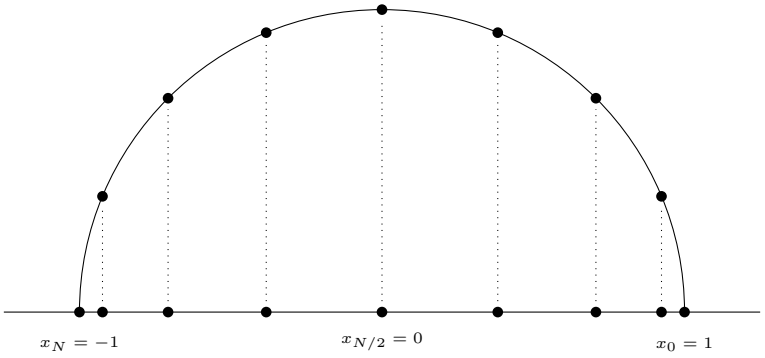


FIGURE 3.3: Chebychev points can be geometrically interpreted as projections of points uniformly spaced on the unit circle onto the x-axis.

In order to approximate the derivative, we therefore first interpolate $f(x)$ by a polynomial $p(x)$ of degree $\leq N$, i.e. we find $p(x)$ such that $p(x_k) = f_k$ and after that we set $f'_k = p'(x)|_{x=x_k}$. As before, this can be written as

$$f' = D_N f \tag{3.14}$$

(compare to (3.3) and (3.5)). Let's consider the case $N = 1$ to illustrate the principle. There are only 2 grid points $x_0 = 1$ and $x_1 = -1$ and the function $f(x)$ interpolates as

$$p(x) = \frac{1}{2}(1 + x)f_0 + \frac{1}{2}(1 - x)f_1. \tag{3.15}$$

Therefore, the derivative of the function is found to be

$$f'(x) \approx p'(x) = \frac{1}{2}f_0 - \frac{1}{2}f_1. \tag{3.16}$$

Which in matrix form (3.14) is written as

$$\begin{pmatrix} \\ f_0' \\ \\ \\ f_1' \\ \\ \end{pmatrix} = D_1 \begin{pmatrix} \\ f_0' \\ \\ \\ f_1' \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \frac{1}{2} & -\frac{1}{2} \\ \\ \\ \frac{1}{2} & -\frac{1}{2} \\ \\ \end{pmatrix} \begin{pmatrix} \\ f_0 \\ \\ \\ f_1 \\ \\ \end{pmatrix}. \tag{3.17}$$

Similarly, for case $N = 2$, there are three grid points $x_0 = 1, x_1 = 0$ and $x_2 = -1$. The interpolating function is

$$p(x) = \frac{1}{2}x(1+x)f_0 + (1+x)(1-x)f_1 + \frac{1}{2}x(x-1)f_2, \tag{3.18}$$

and the derivative is obviously

$$f'(x) \approx p'(x) = (x + \frac{1}{2})f_0 - 2xf_1 + (x - \frac{1}{2})f_2. \tag{3.19}$$

The differentiation matrix for this case looks as

$$\begin{pmatrix} \\ f_0' \\ \\ \\ f_1' \\ \\ \\ f_2' \\ \\ \end{pmatrix} = D_1 \begin{pmatrix} \\ f_0' \\ \\ \\ f_1' \\ \\ \\ f_2' \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \frac{3}{2} & -2 & \frac{1}{2} \\ \\ \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ \\ \\ -\frac{1}{2} & 2 & -\frac{3}{2} \\ \\ \end{pmatrix} \begin{pmatrix} \\ f_0 \\ \\ \\ f_1 \\ \\ \\ f_2 \\ \\ \end{pmatrix}. \tag{3.20}$$

In general case of arbitrary N, the components of the differentiation matrix are given by[21, 76] :

$$\begin{aligned}(D_N)_{ij} &= \frac{c_i}{c_j}\frac{(-1)^{i+j}}{(x_i - x_j)} \quad \text{for } \forall\, i,j \neq 0, N \quad \text{and} \quad i \neq j,\\ (D_N)_{ii} &= -\frac{x_j}{2(1-x^2)} \quad \text{for } \forall\, i \neq 0, N,\\ (D_N)_{00} &= \frac{2N^2+1}{6}, \quad (D_N)_{NN} = -\frac{2N^2+1}{6},\end{aligned} \tag{3.21}$$

where

$$c_i = \begin{cases} 2 & i = 0 \quad \text{or} \quad i = N \\ 1 & i = 1, 2, \cdots, N-1 \end{cases}$$

Now finding the approximation of a derivative of a function defined on a grid of Chebychev points (3.13) numerically is done by a matrix-vector multiplication (3.14), where the components of a matrix D_N are given by (3.21). Higher order (k^{th}-order) derivatives will be of course approximated by powers of the matrix, $(D_N)^k$. We illustrate the accuracy of this method by computing first and second derivatives of some function and comparing the numerical approximation as given by the Chebychev differentiation matrices to the analytical expression. Consider the "probe" function and its derivatives

$$\begin{aligned} f(x) &= e^x\left(x - x^3\right)\\ f'(x) &= e^x(1 + x - 3x^2 - x^3)\\ f''(x) &= -e^x(-2 + 5x + 6x^2 + x^3)\end{aligned} \tag{3.22}$$

The difference $f'(x)_{\text{analytical}} - f'(x)_{\text{numerical}}$ is presented in Fig. 3.4 and similarly for the second derivative in Fig. 3.5.

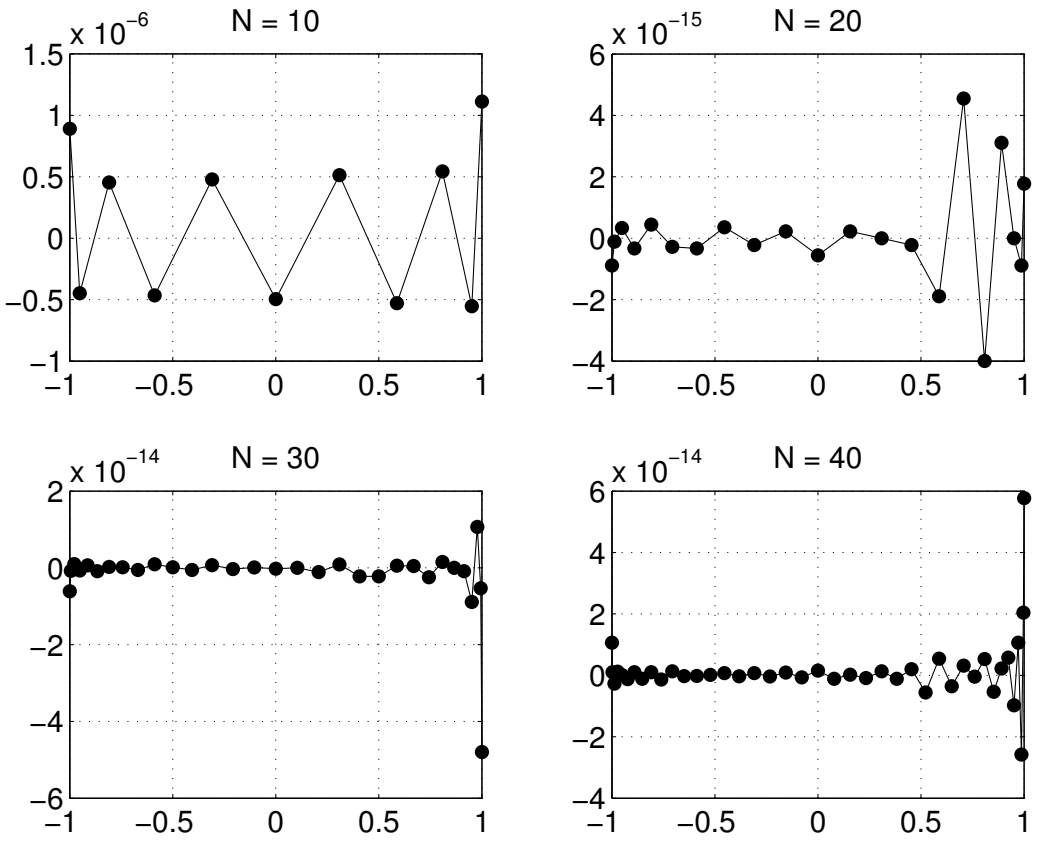


FIGURE 3.4: Numerical approximation of the first derivative on a non-periodic domain obtained using the Chebychev differentiation matrix (3.21) for different numbers of grid points.

We see that even for $N = 10$ somewhat reasonable accuracy is achieved! Increasing N to ~ 30 we get a more than satisfactory approximation of derivatives.

3.2.4 Approximations of partial derivatives.

It is clear now how to proceed in the case when we need to approximate more complicated derivatives. Consider for example some function defined in polar coordinates $f(r, \theta)$.

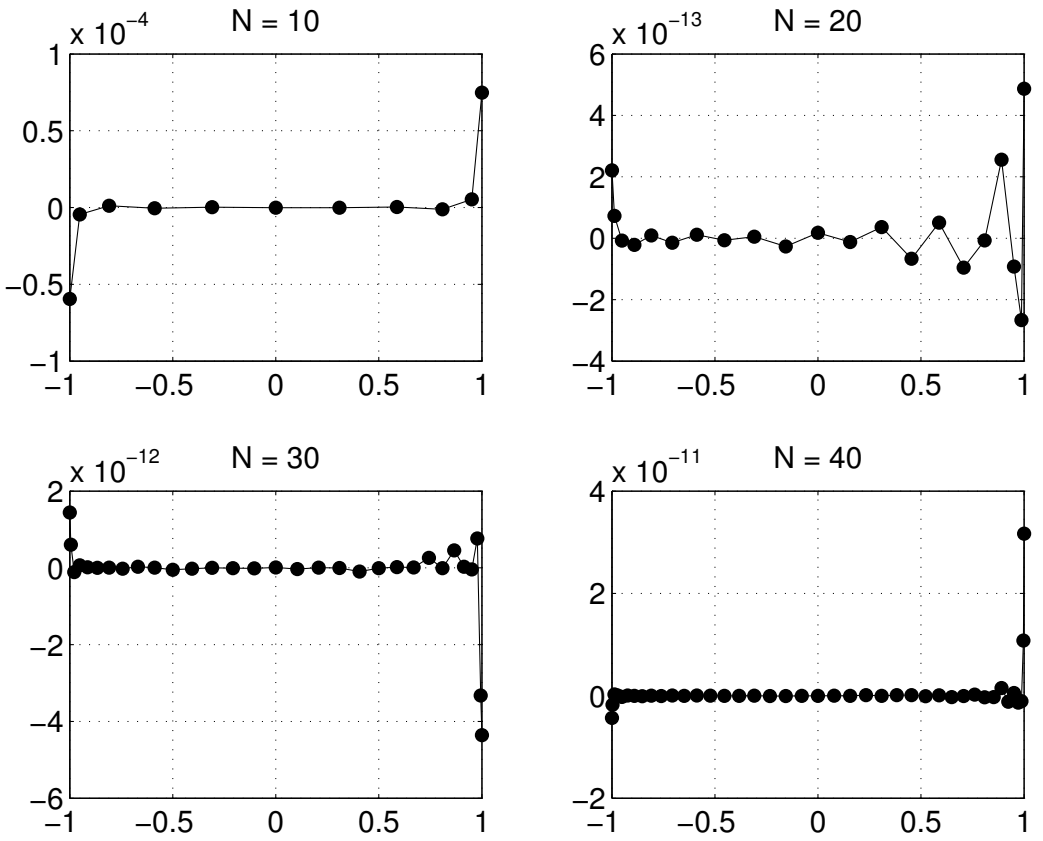


FIGURE 3.5: Numerical approximation of the second derivative on a non-periodic domain obtained using the square of the Chebychev differentiation matrix (3.21) for different numbers of grid points.

Imagine that we have to compute something like:

$$\Delta f = \frac{d^2 f(r,\theta)}{dr^2} + \frac{1}{r}\frac{df(r,\theta)}{dr} + \frac{1}{r^2}\frac{d^2 f(r,\theta)}{d\theta^2}, \tag{3.23}$$

which of course is recognized as the Laplacian of f. To compute it numerically on a disc of unit radius we should sample f at Chebychev grid points in r-direction and on regular uniform grid in θ-direction. However, the Chebychev grid as it is introduced above is defined on the interval $[-1, 1]$, whereas $r \in [0, 1]$. This complication can be resolved in many ways. One possibility is to simply map regular Chebychev grid from $x_j \in [-1, 1]$ to $r_j \in [0, 1]$ via the transformation $r = (x+1)/2$. Then $d/dr = (dx/dr) \cdot (d/dx) = 2(d/dx)$.

We see that this transformation will simply multiply the Chebychev differentiation matrix (3.21) by a constant. But the rest of the formalism is still applicable. Another method would be to use $r \in [-1, 1], \theta \in [0, 2\pi]$ and explore the symmetry [75, 77]

$$f(r, \theta) = f(-r, (\theta + \pi)(\text{mod } 2\pi)). \tag{3.24}$$

Various other methods that deal with polar or spherical coordinates are discussed in great details in [19].

Whatever approach we pick, the approximate Laplacian is written as

$$\Delta f(r_i, \theta_j) = \left[\bar{D}^2_{\text{cheb}} f(r, \theta_j)\right]_i + \frac{1}{r_i}\left[\bar{D}_{\text{cheb}} f(r, \theta_j)\right]_i + \frac{1}{r_i^2}\left[D^{(2)} f(r_i, \theta)\right]_j \tag{3.25}$$

here $\bar{D}_{\text{cheb}}$ is the Chebychev differetiation matrix (3.21) and the bar on top of it is used to indicate that it may possibly be multiplied by a coefficient emerging from the grid transformation; $D^{(2)}$ is a matrix, (3.10), for differentiation in the periodic (angular) direction; $f(r, \theta_j)$ is a vector of values of function f for fixed θ_j. The product $Df(r, \theta_j)$ is a vector itself and the notation $[Df(r, \theta_j)]_i$ is used to select i^{th} component of this vector.

3.3 Methods of solving partial differential equations.

Consider the differential equation

$$f_t = \mathcal{L} f \tag{3.26}$$

where $\mathcal{L}$ is some operator. It may be a differential operator, in Cartesian, polar, spherical *etc...* system of coordinates or it may simply denote multiplication by some (time-independent) factor. For example, if $\mathcal{L} = \Delta$, equation (3.26) becomes a well-studied parabolic partial differential equation known as the "heat equation". If $\mathcal{L} = a\cdot$ — simple multiplication by a constant, then (3.26) is an ordinary differential equation with the trivial solution $f(t) = f_0 \exp(at)$. In previous sections we discussed some of the possible ways to carry out numerical differentiation, so, if $\mathcal{L}$ is differential operator, we know how to perform computations of $\mathcal{L}f$. Nevertheless, we still want to use a simple ordinary differential equation with operator $\mathcal{L}$ being just a multiplication by a constant. This will still allow us to to briefly describe some of the most typical approaches to integrate differential equations numerically.

3.3.1 Euler method.

We are trying to integrate equation

$$f' = af, \tag{3.27}$$

where f' means df/dt. The first and the most natural idea is to use discretized time and to approximate the time-derivative by a finite difference as it is described in section 3.2.1. We use slightly different notation here and denote $f^{(n)} = f(n\Delta t)$ — the value of function f at the n^{th} time step. Equation (3.27) is approximated as

$$\frac{f^{(n+1)} - f^{(n)}}{\Delta t} = af^{(n)} \tag{3.28}$$

Knowing $f^{(0)}$ we can now find $f^{(1)}$ as $f^{(1)} = f^{(0)} + \Delta t a f^{(0)}$. Continuing this iterative process until the desired time is reached, we can obtain an approximate solution of (3.27). We know that the exact solution of this equation at time Δt is given by

$$\begin{aligned} f^{(1)} = f(\Delta t) = f^{(0)} e^{a\Delta t} \approx \\ \approx f^{(0)} + a f^{(0)} \Delta t + \frac{1}{2} a^2 f^{(0)} (\Delta t)^2 + \frac{1}{3!} a^3 f^{(0)} (\Delta t)^3 + \cdots \end{aligned} \tag{3.29}$$

Comparing this to our analytical solution we see that the first term that we got wrong is $const \cdot (\Delta t)^2$. This truncation error will be introduced at every computational iteration. To get to the final time we need to make $N_t \sim 1/\Delta t$ of such steps and, therefore, the total accumulated error will be $\sim \Delta t$. This is very slow convergence and, as we will show, much better convergence can be easily achieved. Another important aspect is the notion of stability. We see that the error grows by $(1 + a\Delta t)$ at every time step, so this scheme is only stable for $|1 + a\Delta t| \leq 1$.

We can slightly modify (3.28) and write is as

$$\frac{f^{(n+1)} - f^{(n)}}{\Delta t} = a f^{(n+1)} \tag{3.30}$$

It is easy to find that in this case

$$f^{(n+1)} = \frac{1}{1 - a\Delta t} f^{(n)} \approx (1 + a\Delta t + a^2 (\Delta t)^2) f^{(n)} \tag{3.31}$$

Again term $\sim (\Delta t)^2$ is wrong (compared to (3.29)) and, therefore, this method has the same convergence. However, we see that it has much better stability properties. In fact, it is stable for $\forall$ a and Δt. These two numerical schemes have names *forward* and *backward* (or alternatively *explicit* and *implicit*) Euler schemes.

3.3.2 Crank-Nicolson Scheme.

A quite natural idea that was first suggested by John Crank and Phyllis Nicolson in 1947 [78] is to combine both of the two methods described above and write down the numerical approximation of (3.27) as

$$\frac{f^{(n+1)} - f^{(n)}}{\Delta t} = a\frac{f^{(n)} + f^{(n+1)}}{2} \tag{3.32}$$

The right-hand side is taken as the average of values of f at n^{th} and $(n+1)^{\text{st}}$ time steps. Due to this symmetry, this scheme is also sometimes referred to as the *trapezoidal* method. Similarly to what we did for Euler methods we find that

$$f^{(n+1)} = \frac{1 + a\Delta t/2}{1 - a\Delta t/2} f^{(n)} \tag{3.33}$$

We see that the stability condition $|(1 + a\Delta t/2)/(1 - a\Delta t/2) \leq |1$ implies that scheme is stable for $\forall\ \Delta t$ and $a < 0$. Expanding it to a Taylor series we get

$$\begin{aligned} f^{(n+1)} &\approx \left(1 + \frac{a\Delta t}{2}\right)\left(1 + \frac{a\Delta t}{2} + \frac{(a\Delta t)^2}{4} + \frac{(a\Delta t)^3}{8} + \cdots\right) f^{(n)} \\ &\approx \left(1 + a\Delta t + \frac{(a\Delta t)^2}{2} + \frac{(a\Delta t)^3}{4} + \cdots\right) f^{(n)} \end{aligned} \tag{3.34}$$

Comparing this to the exact solution given by (3.29), we notice that the Crank-Nicolson scheme evaluates the approximate solution correctly up to the term $\sim (\Delta t)^2$. So this method is said to be of second order for much better accuracy. It is in fact an extremely popular method of numerical integration of PDEs exactly for this combination of second-order accuracy and great stability properties. However, its "semi-implicit" nature creates some computational expenses. In our simplistic example, the evaluation of subsequent

time-steps only costs a multiplication of previous time-step by a factor $1 + a\Delta t/2$ and a division by a factor $1 - a\Delta t/2$. If we were to solve a PDE instead of an ordinary differential equation, the expression (3.33) would become a system of linear algebraic equations for every $f_i^{(n)} = f(t_n, x_i)$.

3.3.3 Adams-Bashforth's method.

In a Crank-Nicolson scheme, the better accuracy ($\sim (\Delta t)^2$ as opposed to $\sim \Delta t$ for the Euler method) is achieved due to the fact that the estimate for $f^{(n+1)}$ is computed by considering the right-hand side of (3.27) an average of the values of f' at the n^{th} and $(n+1)^{\text{st}}$ time steps. This trick allowed to increase accuracy but introduced some exctra computational costs.

Another possible approach to reach good accuracy (at least to second order in Δt) and to keep at the same time the derivation fully explicit would be to use information about values of f at n^{th} and $(n-1)^{\text{st}}$ time steps while computing the $(n+1)^{\text{st}}$ value. Let's approximate the left-hand side as

$$\frac{f^{(n+1)} - f^{(n)}}{\Delta t} = a\left(Af^{(n-1)} + Bf^{(n)}\right) \tag{3.35}$$

The coefficients A and B are to be picked such that this approximation correctly restores the term proportional to $(\Delta t)^2$ in (3.29). By expanding $f^{(n-1)}$ into a Taylor series and comparing (3.35) to the exact solution, it is easy to show that these coefficients have to

be $A = -1/2$ and $B = 3/2$, such that (3.35) resolves for $f^{(n+1)}$ as

$$f^{(n+1)} = f^{(n)} + \Delta t \left(\frac{3}{2} a f^{(n)} - \frac{1}{2} a f^{(n-1)} \right) \tag{3.36}$$

We see that this method is fully explicit (which should be read as "easy-to-compute") and has second order accuracy. In a similar manner, one can create a method that would include more of the previous time-steps

$$\frac{f^{(n+1)} - f^{(n)}}{\Delta t} = a \left(A f^{(n-1)} + B f^{(n)} + C f^{(n-2)} + D f^{(n-3)} + \cdots \right) \tag{3.37}$$

Each of the terms (with correctly picked coefficients) will increase convergence. The family of these methods is known as a family of multistep Adams-Bashforth's methods [79, 80]. To determine the stability properties of the method (3.36), we rewrite it as

$$f^{(n+1)} - \left(1 + \frac{3}{2} a \Delta t \right) f^{(n)} + \frac{1}{2} a \Delta t f^{(n-1)} = 0 \tag{3.38}$$

In this finite difference equation, we furthermore substitute an ansatz $f^{(n)} = p^n$ and obtain (after canceling the common term p^{n-1})

$$p^2 - \left(1 + \frac{3}{2} a \Delta t \right) p + \frac{1}{2} a \Delta t = 0 \tag{3.39}$$

The two roots of this quadratic equation for p are given by

$$\begin{aligned} p_1 &= \frac{1}{2} \left\{ \left(1 + \frac{3}{2} a \Delta t \right) + \sqrt{\left(1 + \frac{3}{2} a \Delta t \right)^2 - 2 a \Delta t} \right\} \\ p_2 &= \frac{1}{2} \left\{ \left(1 + \frac{3}{2} a \Delta t \right) - \sqrt{\left(1 + \frac{3}{2} a \Delta t \right)^2 - 2 a \Delta t} \right\} \end{aligned} \tag{3.40}$$

The finite difference equation (3.36), in the limit of $\Delta t \to 0$, reduces to the actual differential equation (3.27). In this limit, the roots p_1 and p_2 are

$$\lim_{\Delta t \to 0} p_1 = 1 + a\Delta t, \quad \lim_{\Delta t \to 0} p_2 = \frac{1}{2} a\Delta t. \tag{3.41}$$

We see that the solution $f^{(n)}$ corresponding to p_1 (in the limit of $\Delta t \to 0$) becomes exactly the solution of the original equation (3.27), as

$$\lim_{\Delta t \to 0} p_1^n = \lim_{\Delta t \to 0} (1 + a\Delta t)^n = \lim_{\Delta t \to 0} (1 + a\Delta t)^{t/\Delta t} = e^{at}. \tag{3.42}$$

However, the solution of a difference equation that is obtained with the second root p_2 does not have a relation to any actual solution of ODE (3.27). For this reason, the root p_2 and its difference equation solution $f^{(n)} = p_2^n$ are called parasitic solutions. To require stability of a difference scheme (3.35) we should obviously need to require that both p_1 and p_2 are smaller than 1:

$$|p_1| \leq 1 \quad \text{and} \quad |p_2| \leq 1 \tag{3.43}$$

These are rather complicated conditions for $a\Delta t$ due to (3.40). However, the point we are making here is that (3.43) is a restriction on a and Δt. Another statement that we will not prove here is that the higher the order of the Adams-Bashforth's method, the smaller is the stability region. A detailed discussion of this question may be found in [81, 82].

3.3.4 Backward differentiation methods.

In order to illustrate a general statement about role of fully explicit and implicit methods let us briefly consider yet another numerical scheme. One can choose an implicit approach

to achieve second-order accuracy by considering the following difference method

$$\frac{3f^{(n+1)} - 4f^{(n)} + f^{(n-1)}}{2\Delta t} = af^{(n+1)} \tag{3.44}$$

This is an improvement of the implicit Euler scheme that we considered above. It is easy to verify that this scheme it second order by expanding $f^{(n-1)}$ into a Taylor series, collecting terms with powers of Δt and comparing the expression for $f^{(n+1)}$ to the Taylor series of the exact solution (3.29). A similar analysis shows that this method due to its implicit nature becomes stable for $\forall \ \ \Delta t$ and $a < 0$. This method can be improved by including values of f at older time-steps. Similarly to Adams-Bashforth's improvement (3.37), each additional term will increase accuracy by an extra power of Δt (see [83] for more on this subject).

Every n^{th}-order backward difference method will have a much wider stability region than an n^{th}-order explicit method. On the downside, there are additional computations that have to be done at each time-step of the implicit method. Therefore picking the correct approximation that will be implemented in the code is a constant trade-off between desired accuracy and stability requirements (which are also dictated by available computational resources).

3.3.5 MPDATA — multidimensional positive definite advection transport algorithm.

Unfortunately, the schemes listed in the previous chapters have problems with handling advection of sharp wave-fronts, or shocks. Those schemes are very usefull as long as we are dealing with fairly smooth wave-forms. However, they have poor performance at advecting wave-forms with sharp leading or trailing edges. Use of these methods for a numerical solution of a constant velocity advection of the initial condition results in distorted and oscillatory solutions. A typical illustration of this effect is presented in Fig. 3.6(a). We consider the following equation for simple advection with constant velocity u in the form

$$f_t = -uf_x \tag{3.45}$$

and integrate it numerically using the Crank-Nicolson formula. We see that this solution generates spurious oscillations, does not preserve maximal value and, what's more important, is not positively defined. Using higher-order methods, it is possible to substantially reduce the impact of these oscillations, however, it turns out that all central difference schemes for solving the advection equation suffer from a similar problem [84]. Further discussion and examples of this may be found in references [84–86]. This effect can be extremely important when the evolution of the positively defined scalar is studied. Sometimes results of such simulations are used as input parameter in a different problem, and having negative values may ruin the stability of the whole system. Therefore, it is extremely important to pay special attention to positiveness of the solution (for example

[87]).

A typical approach to solve hyperbolic partial differential equations constitutes another class of numerical schemes that is called *upwind methods*. The key idea of these methods is to construct a finite difference approximation based on the direction of the local velocities. To illustrate the method, let us again consider equation (3.45). A first-order upwind approximation for this equation is written as

$$\frac{f_i^{(n+1)} - f_i^{(n)}}{\Delta t} = -u \frac{f_i^{(n)} - f_{i-1}^{(n)}}{\Delta x} \quad \text{for} \quad u > 0 \tag{3.46}$$

$$\frac{f_i^{(n+1)} - f_i^{(n)}}{\Delta t} = -u \frac{f_{i+1}^{(n)} - f_i^{(n)}}{\Delta x} \quad \text{for} \quad u < 0 \tag{3.47}$$

If $u \neq const$, the condition $u < 0$ should be checked at each computational step (both time- and space-wise). Depending on the result, (3.46) or (3.47) must be used. This scheme is first order accurate and stable when the so-called Courant-Friedrichs-Lewy condition condition holds true [88]:

$$U = \left| \frac{u\Delta t}{\Delta x} \right| \leq 1. \tag{3.48}$$

Besides being only first order accurate, this method is known to suffer from strong implicit diffusivity. We demonstrate the diffusive behavior in Fig. 3.6(b). Clearly, the upwind scheme does not only advect initial condition it the direction of the flow, but also imposes substantial intrinsic (numerical) diffusion. Therefore, alternatives are required.

One such alternative — a numerical scheme called MPDATA (multidimensional positive definite advection transport algorithm) — was proposed in the early eighties by P. Smolarkiewich. It originated as a simple positive-definite advection scheme with small

implicit diffusion for problems of atmospheric cloud models [89]. Over last thirty years, MPDATA has found numerous applications in atmospheric simulations, geophysical fluid models, chemical processes modeling, fluid models from biomechanics and solar physics thus covering scales of motion from micro to stellar [85, 90]. It was also adapted to general time-dependent curvilinear coordinates [91, 92] and unstructured meshes [93, 94]. Application of MPDATA to the "toy problem" of constant advection is shown in Fig. 3.6(c) to illustrate the efficiency of this method.

In order to discuss this method, we again consider the advection equation (3.45) and a first-order upwind approximate scheme given by (3.46) and (3.47) (this is sometimes called *donor-cell* approximation). It can be written in flux form by introducing the flux function

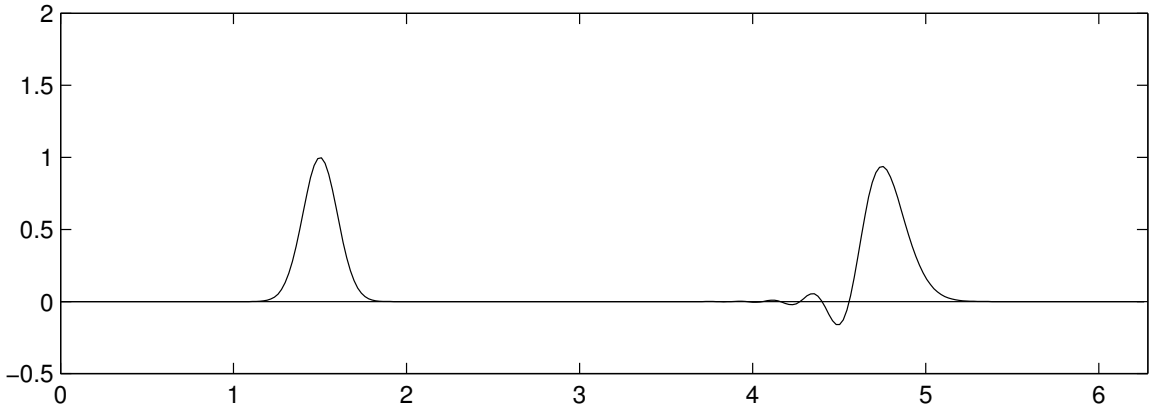
$$F(f_l, f_r, U) = U_+ f_l + U_- f_r, \tag{3.49}$$

where indices at f_l and f_r mean "left" and "right" and denote the position of the donor cell with respect to the cell of interest (where scalar field is currently computed); $U_{(+)}$ and $U_{(-)}$ are nonnegative and nonpositive parts of the local Courant number defined as

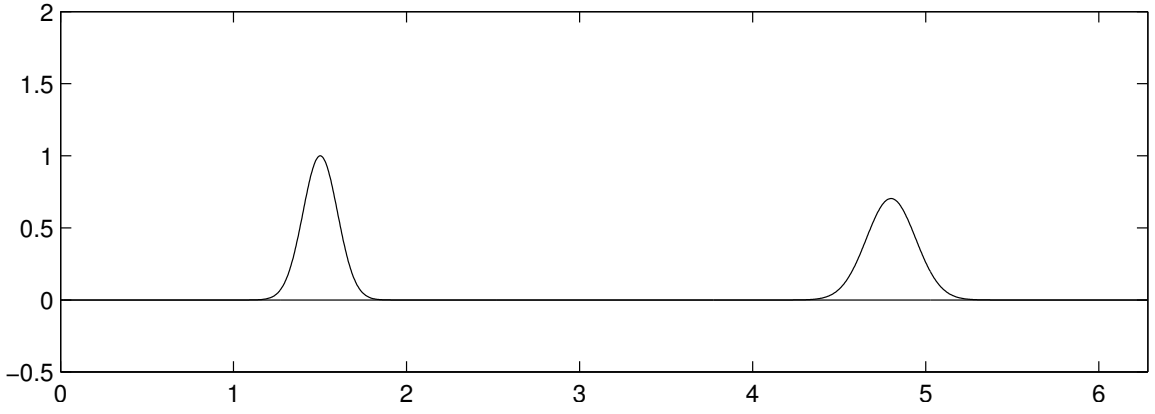
$$\begin{aligned} U &\equiv \frac{u\Delta t}{\Delta x}, \\ U_{(+)} &\equiv \frac{1}{2}(U + |U|), \\ U_{(-)} &\equiv \frac{1}{2}(U - |U|). \end{aligned} \tag{3.50}$$

The flux form of (3.46) and (3.47) now can be compactly written as

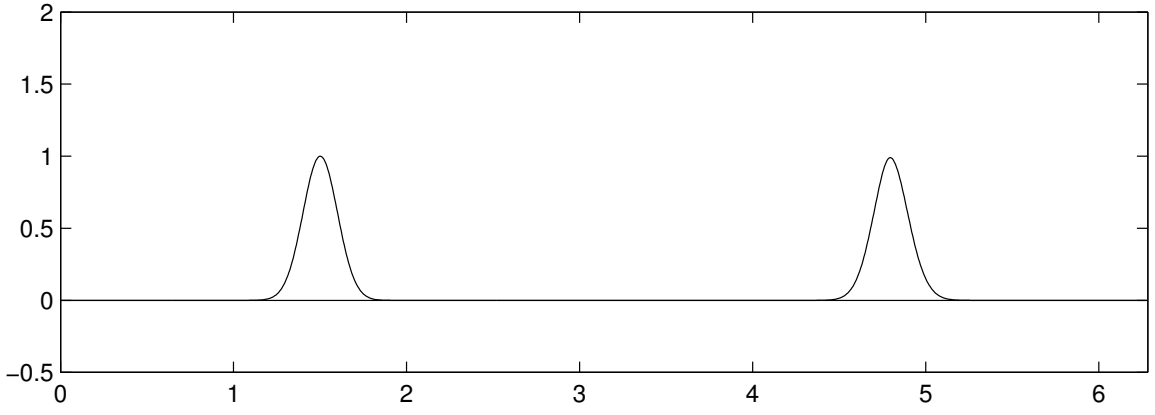
$$f_i^{(n+1)} = f_i^{(n)} - \Big(F(f_i^{(n)}, f_{i+1}^{(n)}, U_{i+1/2}) - F(f_{i-1}^{(n)}, f_i^{(n)}, U_{i-1/2})\Big). \tag{3.51}$$



(a) Crank-Nicolson method.



(b) First-order upwind scheme.



(c) MPDATA algorithm.

FIGURE 3.6: Problem of the simple advection with constant velocity (from left ot right) computed by three different methods. Left side of each graph depicts initial condition and right side — state after some time. Crank-Nicolson scheme exhibits spurious oscillations (subfigure 3.6(a)). Upwind algorithm introduces severe diffusion into the dynamics of the system (subfigure 3.6(b)). The best result is demonstrated by the MPDATA algorithm (subfigure 3.6(c)).

The idea of the method is illustrated in Fig. 3.7.

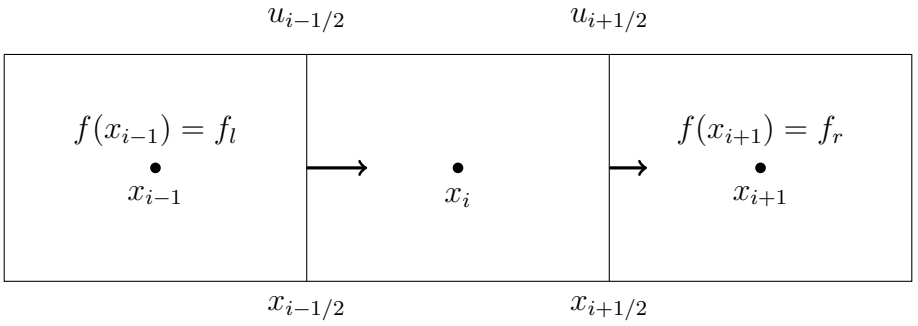


FIGURE 3.7: Schematic of MPDATA. The tracer field is defined at integer points $(x_i, i = 1, 2, 3...)$ — the cell centers. The velocity vector is defined at half-integer points — the cell walls. We compute fluxes from one cell to a neighboring cell based on the direction of the advective flow.

Expanding $f^{(n+1)}, f^{(n)}$ and $f^{(n-1)}$ into a Taylor series around $(x_i, t^{(n)})$, the approximate equation (3.51) may be written as

$$\frac{\partial}{\partial t} f\Big|_i^{(n)} = -\frac{\partial}{\partial x}(uf)\Big|_i^{(n)} + \frac{\partial}{\partial x}\Big(\frac{(\Delta x)^2}{2\Delta t}(|u| - u^2)\frac{\partial f}{\partial x}\Big)\Big|_i^{(n)}. \tag{3.52}$$

The last expression is a second-order approximation to the advection-diffusion equation

$$f_t + uf_x = -\frac{\partial}{\partial x}\Big(-K_{impl}\frac{\partial f}{\partial x}\Big), \tag{3.53}$$

where

$$K_{impl} = \frac{(\Delta x)^2}{2\Delta t}\left(|U| - U^2\right) \tag{3.54}$$

is the implicit diffusion that we mentioned above. In the limit when both Δx and Δt approach zero, (3.53) approaches to (3.45). However, actual computations always have finite non-zero $\Delta x, \Delta t$. This, on the one hand, makes the upwind scheme stable but, on the other hand, does not describes equation (3.45) itself but rather an advective-diffusive equation with some effective diffusion. The idea of MPDATA is to rewrite the diffusive

term in the form of a flux $-K_{impl}(\partial f/\partial x) \equiv u_d f$. This new pseudo velocity u_d is called *diffusive* and is given by

$$u_d = -\frac{(\Delta x)^2}{2\Delta t}\left(|U| - U^2\right)\frac{1}{f}\frac{\partial f}{\partial x}. \tag{3.55}$$

Now, the effect of the implicit diffusive flux of the scheme (3.51) can be compensated by an additional upwind step with anti-diffusive velocity $\hat{u} \equiv -u_d$. The term $(1/f)(\partial f/\partial x)$ in (3.55) can be approximated as

$$\frac{1}{f}\frac{\partial f}{\partial x} \approx \frac{2}{\Delta x}\,\frac{f_{i+1}^{(n+1)} - f_i^{(n+1)}}{f_{i+1}^{(n+1)} + f_i^{(n+1)} + \epsilon} \tag{3.56}$$

An arbitrary small $\epsilon > 0$ has to be introduced to guarantee stability of the corrective step.

This scheme easily generalizes to a multidimensional advection equation. Consider the three-dimensional analog of the equation (3.45):

$$f_t = -uf_x - vf_y - wf_z. \tag{3.57}$$

The first order upwind approximation (3.51) becomes

$$\begin{aligned} f_{i,j,k}^{(n+1)} = f_{i,j,k}^{(n)} &- \Big(F(f_{i,j,k}^{(n)}, f_{i+1,j,k}^{(n)}, U_{i+1/2,j,k}) - F(f_{i-1,j,k}^{(n)}, f_{i,j,k}^{(n)}, U_{i-1/2,j,k})\Big) \\ &- \Big(F(f_{i,j,k}^{(n)}, f_{i,j+1,k}^{(n)}, V_{i,j+1/2,k}) - F(f_{i,j-1,k}^{(n)}, f_{i,j,k}^{(n)}, V_{i,j-1/2,k})\Big) \\ &- \Big(F(f_{i,j,k}^{(n)}, f_{i,j,k+1}^{(n)}, W_{i,j,k+1/2}) - F(f_{i,j,k-1}^{(n)}, f_{i,j,k}^{(n)}, W_{i,j,k-1/2})\Big), \end{aligned} \tag{3.58}$$

where separate local Courant numbers are used:

$$U \equiv \frac{u\Delta t}{\Delta x}, \quad V \equiv \frac{v\Delta t}{\Delta x}, \quad W \equiv \frac{w\Delta t}{\Delta x}, \tag{3.59}$$

Similarly, the truncation error analysis via expanding $f^{(n+1)}, f^{(n)}$ and $f^{(n-1)}$ into the Taylor series around $(x_i, y_j, z_k, t^{(n)})$ shows that (3.58) is a second-order approximation to the advection-diffusion equation:

$$\begin{aligned} &f_t + uf_x + vf_y + wf_z = \\ &\frac{(\Delta x)^2}{2\Delta t}(|U| - U^2)\frac{\partial^2 f}{\partial x^2} + \frac{(\Delta y)^2}{2\Delta t}(|V| - V^2)\frac{\partial^2 f}{\partial y^2} + \frac{(\Delta z)^2}{2\Delta t}(|W| - W^2)\frac{\partial^2 f}{\partial z^2} - \\ &\frac{UV\Delta x\Delta y}{2\Delta t}\frac{\partial^2 f}{\partial x \partial y} - \frac{UW\Delta x\Delta z}{2\Delta t}\frac{\partial^2 f}{\partial x \partial z} - \frac{VW\Delta y\Delta z}{2\Delta t}\frac{\partial^2 f}{\partial y \partial z}. \end{aligned} \tag{3.60}$$

The compensation of diffusive flux can now be done by an additional upwind step with 3 components of the antidiffusive velocity given by

$$\hat{u} = \frac{(\Delta x)^2}{2\Delta t}(|U| - U^2)\frac{1}{f}\frac{\partial f}{\partial x} - \frac{UV\Delta x\Delta y}{2\Delta t}\frac{1}{f}\frac{\partial f}{\partial y} - \frac{UW\Delta x\Delta z}{2\Delta t}\frac{1}{f}\frac{\partial f}{\partial z}, \tag{3.61}$$

$$\hat{v} = \frac{(\Delta y)^2}{2\Delta t}(|V| - V^2)\frac{1}{f}\frac{\partial f}{\partial y} - \frac{VU\Delta y\Delta x}{2\Delta t}\frac{1}{f}\frac{\partial f}{\partial x} - \frac{VW\Delta y\Delta z}{2\Delta t}\frac{1}{f}\frac{\partial f}{\partial z}, \tag{3.62}$$

$$\hat{w} = \frac{(\Delta z)^2}{2\Delta t}(|W| - W^2)\frac{1}{f}\frac{\partial f}{\partial z} - \frac{WU\Delta z\Delta x}{2\Delta t}\frac{1}{f}\frac{\partial f}{\partial x} - \frac{WV\Delta z\Delta y}{2\Delta t}\frac{1}{f}\frac{\partial f}{\partial y}. \tag{3.63}$$

Afterwards, the terms $(1/f)(\partial f/\partial x)$, $(1/f)(\partial f/\partial y)$ and $(1/f)(\partial f/\partial z)$ are approximated as in (3.56), and stability of this corrective iteration is guaranteed.

Chapter 4

Conclusion.

We have proposed the method of averaging time-periodic dynamics of the advection-diffusion equation akin to one suggested by Krol. The proposed scheme exploits a transformation to action-angle coordinates in which the original equation can be written in a form suitable for averaging. The diffusion is taken to be small and is treated as a perturbation to purely advective periodic motion via a Lie transform. We consider the application of this method to a particular flow field, namely a time-periodic regularized vortical flow, and provide numerical evidence that the dynamics of the averaged time-independent equation adequately approximates the dynamics of the complete advection-diffusion equation. We provide a detailed analysis of the spectra of both full and averaged equations, defined on a circular domain with the Dirichlet boundary conditions, using Chebychev spectral approach for the numerical approximations of finite differences. Spectral structures of

both equations are shown to converge with the decrease of the diffusion coefficient. A detailed discussion of the convergence and accuracy of the approximation is provided.

We illustrate that, for the flows with certain properties of time-dependence (periodic and mean-free), purely advective motion is conservative and symmetry-preserving (with the Poincaré map of the flow being an identity), whereas arbitrary small diffusion breaks the symmetry of the initial state of the tracer field.

We only considered the rather restricted class of advective fields with periodic tracer trajectories. For this type of flow, the transformation to action-angle coordinates enables the averaging procedure. Development of a similar approach for other classes of velocity fields will be important to understand general regularities of the interplay between advection and diffusion. We provide a brief discussion of a way to generalize the developed method to the three-dimensional case.

The advection-diffusion equation is one of the key equations in the problem of transport and mixing effects in complex flows. An important problem of the theory of dynamical systems is the detection of coherent structures, which can be responsible for enhancement or suppression of transport and mixing effects in the flow. We discussed a method to visualize flow properties that is based on a highly parallel finite-differences solver of the advection-diffusion equation implemented using co-array Fortran (CAF). CAF is the extension of the Fortran language, that supports the partitioned global address space (PGAS) programming model. CAF is a a representative of SPMD ("single program, multiple data") parallelism methodology and naturally maps to the algorithmic structure

of finite- difference methods. We apply the developed method to illustrate how it can be used to study oceanic flows using the example of the HYCOM data. In particular, we illustrate how the solver can be used to model and visualize the evolution of a contaminant spill, study the effect of diffusion on the tracers propagation, or detect visually possible coherent structures in the flow.

Appendix A

Comparison of co-array Fortran and MPI

Listing A.1 illustrates implementation of halo-exchange problem using Message-Passing Interface (MPI). Listing A.2 is the code that does the same but using co-array Fortran. It is easy to see how economical, compact and elegant co-array Fortran is.

Listing A.1: Halo exchange. MPI

```
1   real :: data(0:nx+1, 0:ny+1, 0:nz+1)
2   real :: local_result, global_result
3   integer :: mype, ier, nx, myright, myleft
4   integer :: stag, rtag, status, iz
5
6   call MPI_init(ier)
7   call MPI_comm_rank(MPI_COMM_WORLD, mype, ier)
8
9   (setup)
10
11  ! exchange halo cell data with "left" and "right" processors
12  do iz = 1, nz
13     stag = stag + 1
14     rtag = rtag + 1
15     call MPI_sendrecv(data(1,ny,iz), nx, MPI_REAL8, myright, stag, &
16          & data(1,0,iz), nx, MPI_REAL8, myleft, rtag, &
17          & MPI_COMM_WORLD, status, ier)
18     stag = stag + 1
19     rtag = rtag + 1
20     call MPI_sendrecv(data(1,1,iz), nx, MPI_REAL8, myleft, stag, &
21          & data(1,ny+1,iz), nx, MPI_REAL8, myright, rtag, &
22          & MPI_COMM_WORLD, status, ier)
23
24  enddo
25
26  ! Do some useful work on my new halo cell data then sum results
27  local_result = use_data(data, nx, ny, nz)
28
29  call MPI_reduce(local_result, global_result, 1, MPI_REAL8, &
30       & MPI_SUM, 0, MPI_COMM_WORLD, ier)
31
32
33  if(mype .eq. 0) print *,"Final =", global_result
34
35  call MPI_finalize(ier)
```

Listing A.2: Halo exchange. Co-array Fortran

```
1   real :: data(0:nx+1, 0:ny+1, 0:nz+1)[*]
2   real :: local_result, global_result[*]
3   integer :: myright, myleft, me, ix, iz
4
5   me = this_image()
6
7   ! exchange halo cell data with "left" and "right" processors
8
9   do iz = 1, nz
10    do iz = 1, nx
11      data(ix, 0, iz) = data(ix,ny,iz)[myleft]
12      data(ix, ny+1, iz) = data(ix, 1, iz)[myright]
13    enddo
14  enddo
15
16
17  ! Do some useful work on my new halo cell data then sum results
18  local_result = use_data(data, nx, ny, nz)
19
20  cirtical
21    global_result[1] = global_result[1] + local_result
22  end critical
23
24  sync all
25
26  if(me .eq. 1) print *,"Final = ", global_result
```

Bibliography

[1] Batchelor G. K. Small-scale variations of convected quantities like temperature in turbulent fluid. *Journal of Fluid Mechanics.*, 5:113–133, 1959.

[2] G. A. Pavliotis. *Homoenization theory for advection-diffusion equation with the mean flow.* PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, 2002.

[3] R. M. McLaughlin J. Bonn. Sensitive enhanced diffusivities for flows with fluctuating mean winds: A two-parameter study. *Journal of Fluid Mechanics.*, 445:345 – 375, 2001.

[4] A.H. Nayfeh. *Perturbation methods.* Wiley International, 1973.

[5] S. K. Turitsyn I. Gabitov, T. Schäfer. Lie-transform averaging in nonlinear optical transmission systems with strong and rapid periodic dispersion variations. *Phys. Lett. A*, 265:274–281, 2000.

[6] Y. Kodama. Normal forms for weakly dispersive wave equations. *Phys. Lett. A*, 112: 193–196, 1985.

[7] A. Fick. Ueber diffusion. *Pogg. Ann. Phys. Chem.*, 1855.

[8] H. B. Fischer, J. E. List, C. R. Koh, J. Imberger, and N. H. Brooks. *Mixing in inland and coastal waters.* Academic Press, 1979.

[9] Frank M. White. *Fluid Mechanics.* McGraw Hill, 4th edition, 1999.

[10] V. I. Arnold. *Mathematical Methods of Classical Mechanics.* Springer, New York, 1989.

[11] Y. Kodama A. Hasegawa. *Solitons in Optical Communications.* Oxford University Press, 1995.

[12] Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction.* Springer, 2003.

[13] W. Rossmann. *Lie Groups: An Introduction through Linear Groups.* Oxford University Press, 2002.

[14] Tobias Schäfer, Andrew C. Poje, and Jesenko Vukadinovic. Averaged dynamics of time-periodic advection diffusion equations in the limit of small diffusivity. *Physica D: Nonlinear Phenomena*, 238:233–240, 2009.

[15] Christopher R Anderson. A method of local corrections for computing the velocity field due to a distribution of vortex blobs. *Journal of Computational Physics*, 62: 111–123, 1986.

[16] Gregory T. Balls and Phillip Colella. A finite difference domain decomposition method using local corrections for the solution of poisson's equation. *J. Comput. Phys.*, 180:25–53, 2002.

[17] Peter Bettess. *Infinite Elements*. Penshaw Press, 1992.

[18] Renato Pavanello Euclides Mesquita. Numerical methods for the dynamics of unbounded domains. *Computational and Applied Mathematics*, 24, 2005.

[19] John P. Boyd. *Chebyshev and Fourier Spectram Methods*. Dover, New York, 2001.

[20] D. Gottlieb and S.A. Orszag. *Numerical analysis of spectral methods*. Industrial and Applied Mathematics, Philadelphia, 1977.

[21] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[22] Igor Mezić. *On the geometrical and statistical properties of dynamical systems: theory and applications.* PhD thesis, California Institute of Technology, 1994.

[23] Igor Mezić and S Wiggins. On the integrability and perturbation of three-dimensional fluid flows with symmetry. *Journal of Nonlinear Science*, 1994.

[24] V.M. Canuto. The physics of subgrid scales in numerical simulations of stellar convection: Are they dissipative, advective, or diffusive? *Astrophys. J.*, 541, 2000.

[25] E. Knobloch and W.J. Merryfield. Enhancement of diffusive transport in oscillatory flows. *Astrophys. J.*, 1991.

[26] Stephen Childress. Alpha-effect in flux ropes and sheets. *Physics of the Earth and Planetary Interiors*, 20:172 – 180, 1979.

[27] H K Moffatt. Transport effects associated with turbulence with particular attention to the influence of helicity. *Reports on Progress in Physics*, 46(5):621, 1983.

[28] Andrew J Majda. Vorticity, turbulence, and acoustics in fluid flow. *SIAM Review*, 33(3):349–388, 1991.

[29] Rodó X. and F. A. Comín, editors. *Global Climate*. Springer, 2003.

[30] G. T. Csanady. *Turbulent diffusion in the environment*. D. Reidel Pub. Co. Dordrecht, Boston, 1973.

[31] C. W. Clark. Derivation of microstructure of fluid flow by homogenization. *Math. Anal. Appl.*, 226(3):264–376, 1998.

[32] U. Hornung. *Homogenization in porous media.* Springer Verlag, 1997.

[33] N. S. Bakhvalov and Grigori P. Panasenko. *Homogenization: Averaging Processes in Periodic Media.* Kluwer, Dordrecht/Boston/London, 1989.

[34] Sanchez E. Palencia and A. Zaoui, editors. *Homogenization Techniques for Composite Media*, volume 272 of *Lecture Notes in Physics*. Springer-Verlag, Berlin, 1987.

[35] Andrew J. Majda and Peter R. Kramer. Simplified models for turbulent diffusion: Theory, numerical modelling, and physical phenomena. *Physics Reports*, 314(4-5): 237–574, 1999.

[36] A. J. Majda and T. Souganidis. The effect of turbulence on mixing in prototype reaction diffusion systems. *Comm. Pure Applied Math.*, 53:1284–1304, 2000.

[37] G. M. Zaslavskii and R. Z. Sagdeev. *Introduction to Nonlinear Physics: From the Pendulum to Turbulence and Chaos [in Russian].* Nauka, Moscow, 1998.

[38] Robert L. Devaney. *An Introduction to Chaotic Dynamical Systems.* Westview Press., 2003.

[39] J. P. Gollub and G. L. Baker. *Chaotic dynamics.* Cambridge University Press., 1996.

[40] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–89, 1977.

[41] J.J. Monaghan. An introduction to SPH. *Computer Physics Communications*, 48: 88–96, 1988.

[42] URL http://hycom.org/dataserver.

[43] URL http://www.unidata.ucar.edu/software/netcdf/.

[44] URL http://www.roguewave.com/products/imsl-numerical-libraries.aspx.

[45] Robert W. Numrich and John Reid. Co-array Fortran for parallel programming. *SIGPLAN Fortran Forum*, 17(2):1–31, 1998.

[46] Robert W. Numrich and John Reid. Co-arrays in the next Fortran standard. *SIGPLAN Fortran Forum*, 24(2):4–17, 2005.

[47] Philip J. Klotzbach. Trends in global tropical cyclone activity over the past twenty years (1986–2005). *Geophysical Research Letters*, 33(10), 2006.

[48] Kerry A. Emanuel. The dependence of hurricane intensity on climate. *Nature*, 326 (6112):483–485, 1987.

[49] Gabriela Athié, Julio Candela, José Ochoa, and Julio Sheinbaum. Impact of Caribbean cyclones on the detachment of Loop Current anticyclones. *Journal of Geophysical Research*, 117(C3):1–16, 2012.

[50] Wilton Sturges, Nicholas G Hoffmann, and Robert R Leben. A trigger mechanism for Loop Current Ring separations. *Journal of Physical Oceanography*, 40(5):900–913, 2010. URL http://journals.ametsoc.org/doi/abs/10.1175/2009JPO4245.1.

[51] URL http://upc.gwu.edu/.

[52] URL http://upc.lbl.gov/.

[53] URL http://titanium.cs.berkeley.edu/.

[54] R. A. Carmona, P. Del Moral, P. Hu, and N. Oudjane. *Numerical Methods in Finance*. Springer, 2010.

[55] P. Brandimarte. *Numerical Methods in Finance and Economics: A MATLAB-Based Introduction (Statistics in Practice)*. Wiley-Interscience, 2006.

[56] R. Schwartz. *Biological Modeling and Simulation: A Survey of Practical Models, Algorithms, and Numerical Methods (Computational Molecular Biology)*. The MIT Press, 2008.

[57] H. Prinz. *Numerical Methods for the Life Scientist*. Springer, 2011.

[58] C.J.A.P. Martins. *Quantitative String Evolution*. PhD thesis, University of Cambridge, 1997.

[59] Fethi M. Ramazanoglu and Frans Pretorius. Two-dimensional quantum black holes: numerical methods. *Classical and Quantum Gravity*, 27(24), 2010.

[60] C. S. S. Brandao and J. C. N. de Araujo. A recipe to probe alternative theories of gravitation via n-body numerical simulations. i. spiral galaxies. *The Astrophysical Journal*, 750(1), 2012.

[61] D. Bini. Toeplitz matrices, algorithms and applications. *ERCIM News*, 1995.

[62] A. Bottcher and B. Silbermann. *Introduction to Large Truncated Toeplitz Matrices*. Springer, New York, 1999.

[63] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1999.

[64] N. Morrison. *Introduction to Fourier Analysis*. Wiley Interscience, 1994.

[65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comput.*, 1965.

[66] M. Frigo and S.G. Johnson. Fftw: an adaptive software architecture for the fft. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 3, pages 1381 –1384 vol.3, 1998.

[67] URL http://www.fftw.org/.

[68] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[69] J. W. Gibbs. Fourier's series. *Nature*, 1899.

[70] A. Zygmund. *Trigonometric Series*. Cambridge Mathematical Library, 1959.

[71] C. Runge. Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten. *Zeitschrift für Mathematik und Physik*, 1901.

[72] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric lagrange interpolation. *SIAM Review*, 2004.

[73] John P. Boyd. Defeating the runge phenomenon for equispaced polynomial interpolation via tikhonov regularization. *Applied Mathematics Letters*, 5(6), 1992.

[74] John P. Boyd. Six strategies for defeating the runge phenomenon in gaussian radial basis functions on a finite interval. *Computers and Mathematics with Applications*, 60(12), 2010.

[75] B. Fornberg. *A practical guide to pseudospectral methods*. Cambridge University Press, 1996.

[76] Gottlieb D., Hussaini M. Y., , and S. A. Orzarg. *Introduction: Theory and Applications of Spectral Methods*. SIAM: Society for Industrial and Applied Mathematics, 1984.

[77] Bengt Fornberg. A pseudospectral approach for polar and spherical geometries. *SIAM J. Sci. Comput.*, 16(5), 1995.

[78] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6, 1996.

[79] F. Bashforth and J.C. Adams. *Theories of Capillary Action*. Cambridge University Press, 1883.

[80] H. Jeffreys and B. S. Jeffreys. *Methods of Mathematical Physics, 3rd ed.* Cambridge University Press, 1988.

[81] Hairer Ernst, Nørsett Syvert P., and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 1993.

[82] Germund G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3:27–43, 1963. ISSN 0006-3835. URL http://dx.doi.org/10.1007/BF01963532. 10.1007/BF01963532.

[83] David Mayers and Endre Süli. *An introduction to numerical analysis*. Cambridge Univ. Press, 2003.

[84] S.K. Godunov. A difference scheme for numerical computation of discontinuous solutions of equations in fluid dynamics. *Math. Sb.*, 47, 1959.

[85] Piotr K. Smolarkiewicz. Multidimensional positive definite advection transport algorithm: an overview. *International Journal for Numerical Methods in Fluids*, 50(10):1123–1144, 2006. ISSN 1097-0363. doi: 10.1002/fld.1071. URL http://dx.doi.org/10.1002/fld.1071.

[86] Craig J Tremback, James Powell, William R Cotton, and Roger A Pielke. The forward-in-time upstream advection scheme: extension to higher orders. *Monthly Weather Review*, 115(2):540–555, 1987.

[87] Su-Tzai Soong and Yoshimitsu Ogura. A comparison between axisymmetric and slab-symmetric cumulus cloud models. *Journal of the Atmospheric Sciences*, 30(5): 879–893, 1973. URL http://dx.doi.org/10.1175/1520-0469(1973)030<0879:ACBAAS>2.0.CO;2.

[88] C. Hirsch. *Numerical Computation of Internal and External Flows*. John Wiley & Sons., 1990.

[89] Piotr K. Smolarkiewicz. A simple positive definite advection scheme with small implicit diffusion. *Monthly Weather Review*, 111(3):479–486, 1983.

[90] Piotr K. Smolarkiewicz and Len G. Margolin. MPDATA: A finite-difference solver for geophysical flows. *Journal of Computational Physics*, 140(2):459–480, 1998.

[91] Joseph M. Prusa and Piotr K. Smolarkiewicz. An all-scale anelastic model for geophysical flows: dynamic grid deformation. *Journal of Computational Physics*, 190(2): 601 – 622, 2003. URL http://www.sciencedirect.com/science/article/pii/S0021999103002997.

[92] Piotr K. Smolarkiewicz and Joanna Szmelter. MPDATA: An edge-based unstructured-grid formulation. *J. Comput. Phys.*, 206(2):624–649, 2005.

[93] Bacon DP *et al.* A dynamically adapting weather and dispersion model: the operational environment model with grid adaptivity (omega). *Monthly Weather Review*, 128, 2000.

[94] Margolin LG and Shashkov M. Second-order sign-preserving conservative interpolation (remapping) on general grids. *Journal of Computational Physics*, 184, 2003.