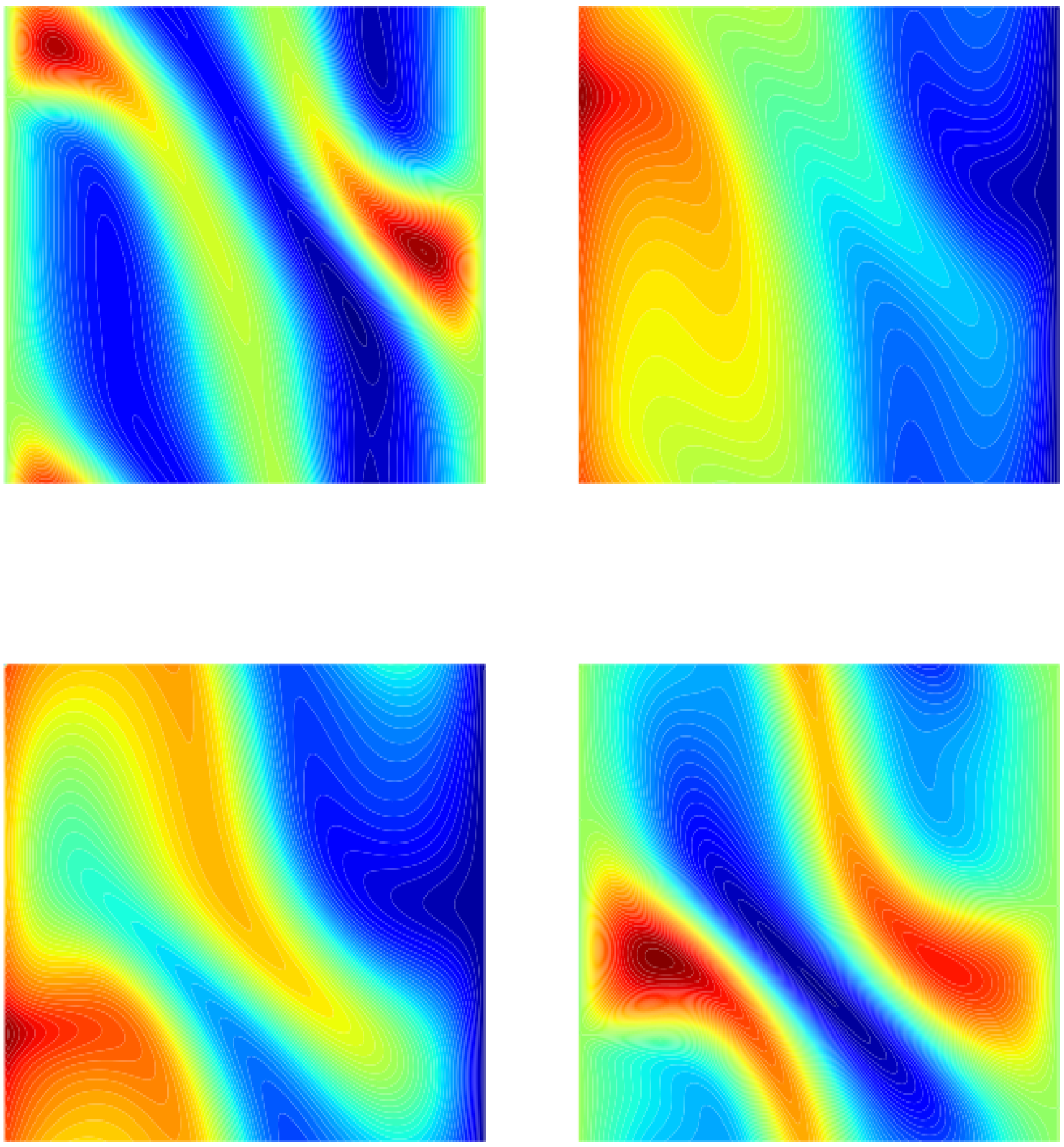
Faculty of Science and Technology
Department of Physics and Technology

Numerical computations of turbulent motions in magnetized plasmas

—

Gregor Decristoforo

FYS-3900 Master's thesis in physics May 2016



Abstract

Intermittent fluctuations and turbulence-induced transport of magnetically confined fusion plasmas are investigated by numerical computations. A reduced fluid model describing the evolution of plasma pressure and electric drift vorticity in a two-dimensional plane perpendicular to the magnetic field is derived. A numerical simulation code implemented on graphical processing units is presented. We observe significant speedup compared to sequential Fortran implementations.

The convective motions are driven by a constant incoming heat flux at the inner radial boundary of the domain. We identify different transport and confinement states. We observe stationary convection and self-sustained shared flows for low heat flux drive. For increasing drive, oscillatory motion with sheared flows arise until the system enters a state of turbulent convection.

At the onset of turbulent convection we observe that the probability density functions of the normalized radial velocity, pressure and flux fluctuations show nearly Gaussian form. This distributions get increasingly non-Gaussian and develop exponential tails for increasing heat flux drive. We observe quasi-periodic bursts at the state of intermittent convection, separated by quiescent periods. The waiting time and amplitude distribution of these bursts take a nearly exponential form. The conditionally averaged waveform and the auto-correlation function of the normalized pressure fluctuations are discussed. We further compare those results to experimental measurements and predictions from stochastic modelling and find very good agreement.

Acknowledgements

First, I want to thank my co-supervisor, Ph.D. Ralph Kube. Ralph introduced me to CUDA- and object orientated programming and always found time to discuss mostly numeric related issues, regardless how complex or trivial they turned out to be. This thesis wouldn't have been possible without him.

I would also like to thank my supervisor Professor Odd Erik Garcia for his excellent guidance. I have learned a lot through the last year while working on this thesis and want to thank him for his time and effort.

In addition a big thank you to Bjørn Fjukstad and Erlend Graff is appropriate, whose UiT thesis LaTeXtemplate I used.

On a personal note, many thanks to my friends and family who supported me throughout my studies in Norway. It has been two very exciting years that gave me a lot of experience and joy.

Contents

Abstract i

Acknowledgements iii

List of Figures vii

List of Tables xi

1 Introduction 1

2 Model Equations 7

2.1 Model setting 7

2.2 Drift terms in toroidal geometry 9

2.3 Two field equations 14

2.4 Dimensionless variables 15

3 Numerical Methods 19

3.1 Simulation domain 20

3.2 Finite differences 21

3.3 Ghost points 22

3.4 Spectral transformations 23

3.5 Elliptic equations 26

3.6 Time integration 30

3.7 Arakawa scheme 33

3.8 Matrix factorisations 34

3.9 Computational complexity 35

4 Parallel Computing 37

4.1 Parallel programming concepts 39

4.2 Graphical processing units 41

4.3 Programming in CUDA 41

4.4 Speedup of 2dads code 46

5 Code Testing 49

5.1 Blob diffusion 49
5.2 Non-linear advection 50
5.3 Simple blob simulation 51

6 Transport in flux-driven convection models 57
6.1 Stationary convection 59
6.2 Convection with sheared flows 60
6.3 Oscillatory motion with shared flows 63
6.4 Onset of turbulent convection 67
6.5 Intermittent convection 68

7 Statistical analysis of fluctuations 73
7.1 Statistical concepts 73
7.2 Probability density functions 76
7.3 Waiting time distribution 80
7.4 Amplitude distribution 82
7.5 Conditionally averaged waveform 83
7.6 Autocorrelation function 85

8 Conclusion and Outlook 87

Bibliography 91

List of Figures

1.1 Progress towards harnessing fusion as a power source compares very favourably with the progress in other high-technologies such as computing performance and particle accelerators [1]. . . 2

1.2 Shematic structure of a tokamak, showing the trajectory of charged particles [4]. 3

1.3 Poloidal cross-section of a tokamak including divertors and the SOL [5]. 4

1.4 Plasma streaming into the SOL of the NSTX expreiment in the form of blob-like structures with an area of $(23\text{cm})^2$ and 10 μs between each frame. Image from Princeton Plasma Physics Laboratory [8]. 5

1.5 Time series of ion saturation current fluctuations of large amplitude bursts [14]. 6

2.1 Charged particles in presence of a magnetic field on the left side and in absence on the right side [15]. 8

2.2 Illustration of the simulation domain in a toroidally magnetized plasma [17]. 8

3.1 Illustration of the two dimensional simulation domain using a cell centered grid and ghost points. The grid points inside the physical domain are represented as filled circles and the ghost points by open circles [20]. 20

3.2 Discretisation of a sin-function with the period $L = 2$ on a cell centered grid with 8 gridpoints in the domain (0,2) [17]. . . 21

3.3 Convergence rate of the finite difference method at the example of a first order derivative of a sin function. 22

3.4 Convergence rate of the finite difference method at the example of a second order derivative of a sin function. 23

3.5 Convergence rate of the spectral derivation scheme at the example of a first order derivative of a Gaussian function. . . . 26

3.6 Convergence rate of the spectral derivation scheme at the example of a second order derivative of a Gaussian function. . 27

3.7 Convergence rate of QR-factorisation at the example of a second order derivative in x- and y-direction of a Gaussian function. 35

4.1 Performance increase at the example of Intel CPUs [31]. . . . 38
4.2 Comparison of the increase of floating-point operations per second for CPUs and GPUs [35]. 42
4.3 Illustration of the processor architectures of conventional CPUs and GPUs [35]. 42
4.4 Runtime t of the CUDA/Fortran code for different numbers of grid points. The runtime is measured for 1000 time iterations and averaged over 5 executions. 47

5.1 Comparison of the numerical result (blue points) of the diffusion equation 5.2 with starting conditions 5.1 and the analytic solution 5.3 (green line). 50
5.2 The average error of the simulation of the diffusion equation 5.2 as a function of discretization points. This is compared to the expected rate of convergence for a second order scheme. 51
5.3 The average and standard deviation of critical parameters . 52
5.4 Vertical profile of the plasma density for different t. 53
5.5 Radial variation of the plasma density for different t. 54
5.6 Radial advection of a localized blob structure initialized at $\mathbf{x}_0 = (5, 35)$ showing the plasma density n in the left column and the potential ϕ in the right column for different t. 55

6.1 Evolution of the kinetic energy in fluctuating motions measured at (x,y) = (0.5,0.5) for the diffusion coefficients $\kappa = \mu = 0.0183$. The fluctuation amplitude saturates at $t \approx 750$. . 59
6.2 Evolution of the kinetic energy (blue line) and the radial heat flux (green line) measured at (x,y) = (0.5,0.5) for the diffusion coefficients $\kappa = \mu = 0.0183$. 60
6.3 Spatial structure of the pressure and electrostatic potential, pressure fluctuations and radial heat flux in the stationary state at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.0183$. 61
6.4 Pressure profile p_0 at $t = 1000$. 62
6.5 Potential profile ϕ_0 at $t = 1000$. 62
6.6 Vorticity profile Ω_0 at $t = 1000$. 62
6.7 Evolution of the kinetic energy integrals $\mathcal{K}$ (green) and $\mathcal{U}$ (blue) for the diffusion coefficients $\kappa = \mu = 0.012$. The energy integral of the fluctuating motion decreases after the appearance of the sheared mean flow at $t \approx 750$. 63

6.8 Spatial tilted structure of the pressure and electrostatic potential, pressure fluctuations and radial heat flux in the stationary state at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.012$. 64
6.9 Evolution of the kinetic energy integrals $\mathcal{K}$ (green line) and $\mathcal{U}$ (blue line) for the diffusion coefficients $\kappa = \mu = 0.011$. . . . 65
6.10 Evolution of the plasma pressure and the electrostatic potential over one period of oscillation for the diffusion coefficients $\kappa = \mu = 0.011$. 66
6.11 Evolution of the integrated convective heat flux Γ_p (green line) and the integrated pressure $\mathcal{E}$ (blue broken line) for the diffusion coefficients $\kappa = \mu = 0.011$, showing the oscillatory convection from $t \approx 750$ on. 67
6.12 Evolution of the kinetic energy integrals $\mathcal{K}$ (green line) and $\mathcal{U}$ (blue line) for the diffusion coefficients $\kappa = \mu = 0.0016$. . . . 68
6.13 Evolution of the integrated convective heat flux Γ_p (green line) and the integrated pressure $\mathcal{E}$ (blue broken line) for the diffusion coefficients $\kappa = \mu = 0.0016$. The system shows no clear periodicity after $t \approx 100$. 68
6.14 Spatial structures of the pressure and electrostatic potential, pressure fluctuations and radial heat flux at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.0016$. 69
6.15 Evolution of the kinetic energy integrals $\mathcal{K}$ (green line) and $\mathcal{U}$ (blue line) for the diffusion coefficients $\kappa = \mu = 0.0007$. . . . 70
6.16 Evolution of the integrated convective heat flux Γ_p (green line) and the integrated pressure $\mathcal{E}$ (blue broken line) for the diffusion coefficients $\kappa = \mu = 0.0007$. 71
6.17 Spatial structures of the pressure and electrostatic potential, pressure fluctuations and radial heat flux at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.0007$. 72

7.1 Time series of the normalized radial flux for the diffusion coefficients $\kappa = \mu = 0.0016$. The threshold at $\widetilde{\Gamma}_x = 1.5$ is represented by the reed broken line. The local maxima are displayed by the black circles. 74
7.2 Time series of the normalized radial flux for the diffusion coefficients $\kappa = \mu = 0.0007$. The threshold at $\widetilde{\Gamma}_x = 1.5$ is represented by the reed broken line. The local maxima are displayed by the black circles. 75
7.3 Normalized PDF for the radial velocity fluctuation $\widetilde{v}_x$ measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$, showing nearly Gaussian statistics. 77

7.4 Normalized PDF for the radial velocity fluctuation $\widetilde{v}_x$ measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$, revealing exponential tails for large fluctuation amplitudes. 77
7.5 Probability distribution functions of the normalized pressure $\widetilde{p}$, potential $\widetilde{\phi}$, vorticity $\widetilde{\Omega}$ and the radial heat flux $\widetilde{\Gamma}$, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$. 79
7.6 Probability distribution functions of the normalized pressure $\widetilde{p}$, potential $\widetilde{\phi}$, vorticity $\widetilde{\Omega}$ and the radial heat flux $\widetilde{\Gamma}$, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$. 80
7.7 Waiting time distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$. 81
7.8 Waiting time distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$. 81
7.9 Amplitude distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$. 82
7.10 Amplitude distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$. 83
7.11 Conditionally averaged waveform of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$. 84
7.12 Conditionally averaged waveform of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$. 84
7.13 Autocorrelation function of the normalized pressure measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$. 85
7.14 Autocorrelation function of the normalized radial velocity measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$. 86

List of Tables

3.1 Coefficients of the stiffly stable time integration for K-th order accuracy. 32

1 Introduction

During the last couple of decades there has been a growing need for alternative energy sources. CO_2 emissions from combustions of fossil fuels contribute significantly to global warming and shortages of these fuels have the potential to cause enormous political problems. Fusion power has the potential to provide large-scale energy without contributing to global warming or generating long-term radioactive waste as nuclear fission [1].

Since the second world war it has been a goal of the scientific community to harvest the energy released by fusion reactions of light atomic nuclei. The progress toward this goal is so far comparable to the progress in computer performance and particle accelerators, see figure 1.1. In order to fuse into heavier elements, the particles must overcome the Coulomb barrier to get close enough that the strong nuclear force can fuse the particles. For thermonuclear fusion, extremely high temperatures are required. In a medium where these conditions are met all particles are ionized and can therefore be confined by strong magnetic fields. Since the beginning of fusion research, several different types of fusion devices have been developed. One of the most promising designs is the tokomak, developed by Soviet physicists Igor Tamm and Andrei Sakharov [2].

A tokamak is a candidate for the world's first fusion reactor design, due to its excellent performance. It is a toroidal device designed to confine a high-temperature plasma using strong magnetic fields, as shown in figure 1.2. The principal magnetic field is toroidal and generated by the field coils around the

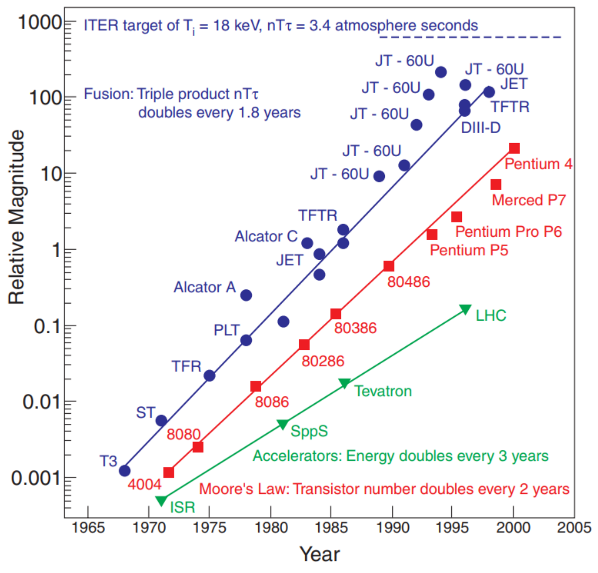


Figure 1.1: Progress towards harnessing fusion as a power source compares very favourably with the progress in other high-technologies such as computing performance and particle accelerators [1].

vacuum vessel walls. In addition, a toroidal electric current is induced in the plasma by a central solenoid. This current induces a poloidal magnetic field which, together with the toroidal field, results in a helical magnetic field. This field confines the charged particles in the plasma along helical paths within the vacuum chamber [3].

The magnetic field strength produced by the primary field coils is commonly in the range of several Tesla. To achieve such strong magnetic fields without enormous ohmic heating losses some present devices and future reactors use superconducting magnets, making these coils the most costly components of the whole device.

Unfortunately this geometry still does not entirely avoid losses of plasma and heat across magnetic field lines. Plasma-wall interactions lead to erosion of the vessel walls and release impurities into the plasma. A design to keep these interactions away from the main vessel wall to a remote region is by diverting the outer part of the magnetic field such that field lines intersect specially designed material surfaces. These surfaces are called the divertor targets. A schematic structure of the poloidal cross-section of a tokamak with divertor plates is shown in 1.3. The interior of the plasma can be divided into three different regions: the centre of the plasma is the so-called *plasma core,* where the magnetic field lines lie on toroidal surfaces of constant pressure and are closed. This region is bordered by the *scrape off layer* (SOL). In this region, the magnetic field lines intersect with the divertor plates. These areas are separated by the so-called separatrix, or last closed flux surface (LCFS). Closer to the wall is the *wall shadow*, where the field lines intersect the vessel wall instead of the divertor. In an ideal case, all plasma that crosses the separatrix will stream along the field lines towards the divertor target without reaching the vacuum vessel walls.

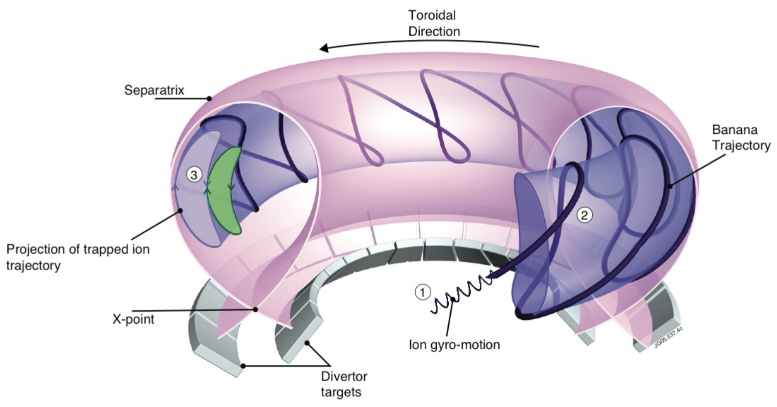


Figure 1.2: Shematic structure of a tokamak, showing the trajectory of charged particles [4].

In the following we discuss the SOL region in further detail. As mentioned before, it is designed to scrape off plasma which is then transported on to the divertor plates. The physical properties of the SOL plasma differs strongly from the core plasma. Particle densities and temperatures in the SOL are lower by several orders of magnitude than in the core. Despite the divertor geometry some plasma will escape the magnetic confinement and may interact with the vessel wall. This transport across the magnetic field lines is dominated by the radial motion of filament structures. These structures contain excess particles and heat compared to the background plasma and are therefore referred to as plasma blobs [6, 7]. Figure 1.4 shows the radial propagation of such a blob-like structure through the SOL into the wall shadow. The solid line represents the location of the separatrix and the dotted line stands for the beginning of the wall shadow. This phenomena is universally observed in all toroidal confinement devices.

The radial transport of plasma through the SOL has therefore been of great interest in the last years [7, 9]. Turbulent structures moving towards the main chamber wall have been studied in many tokamak experiments. An accepted mechanism for radial transport of coherent structures is given by the interchange mechanism [10, 11]. Curvature drifts and gradient-B drifts, produced by non-uniform magnetic fields, are charge dependent and cause

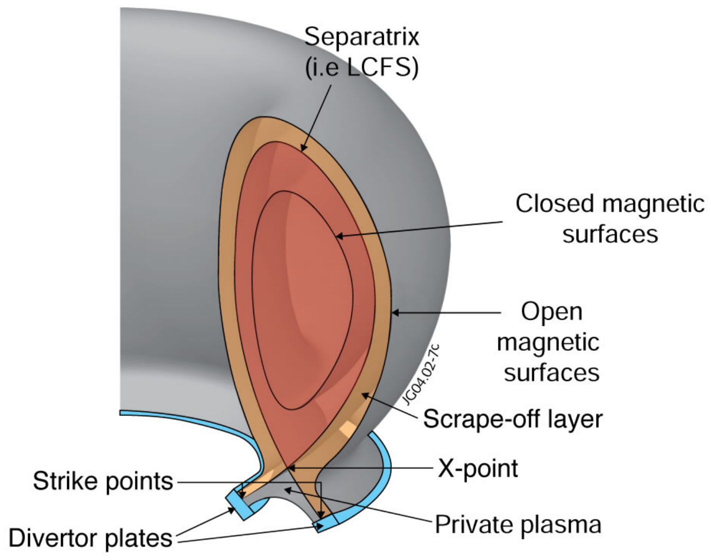


Figure 1.3: Poloidal cross-section of a tokamak including divertors and the SOL [5].

charge polarisation. The so produced electric field results in a electric drift pointing radially outwards the vacuum vessel. Numerical investigations of the interchange model [12] as well as experimental observastions [13] have been considered in previous studies.

Blob dynamics and SOL turbulence have a significant effect on plasma losses in present-day and future fusion plasma devices. It is therefore necessary to develop a theoretical understanding of the dominant physical mechanisms behind blob motion and SOL turbulence. In addition it is of big importance to reveal the statistical properties of plasma fluctuations in the SOL. The average particle density and radial flux in the SOL strongly depends on the amplitude distribution of blob-like structures and their frequency of occurrence. Statistical studies of SOL fluctuations are therefore crucial for predictions of plasma-surface interactions [14]. As an example the raw time series of the ion saturation current fluctuation $\tilde{J}$ of large amplitude bursts is shown in 1.5. We see a frequent appearance of bursts with peak amplitudes several times higher than the root men square (rms) value J_{rms}. In addition to the raw time series it is necessary to investigate other properties such as amplitude and waiting time distributions.

In order to gain statistically conclusive results we desire long data time series. To achieve this goal with numerical studies it is thus crucial to develop efficient numerical solvers to run long term simulations. In this thesis we develop implementations of numerical solvers suitable for studying simple models of SOL

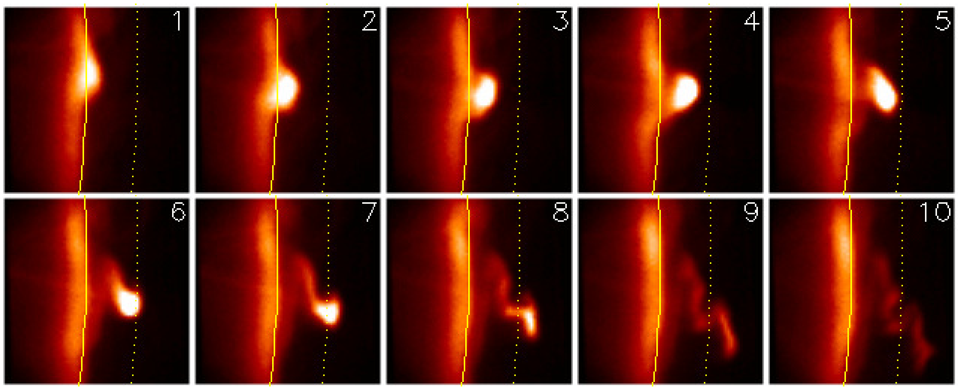


Figure 1.4: Plasma streaming into the SOL of the NSTX expreiment in the form of blob-like structures with an area of $(23\text{cm})^2$ and 10 μs between each frame. Image from Princeton Plasma Physics Laboratory [8].

turbulence and blob dynamics. These solvers are implemented on graphical processing units which offer high performance for scientific applications.

This thesis is structured as the following: In chapter two we present the model equations that describe the collective motion of a magnetized plasma and are derived from the fluid description of a plasma. We describe the numerical methods implemented in chapter three and present code validation. In chapter four we give an overview on parallel computing and programming on graphical processing units and present speed comparisons to previous implementations. We benchmark the code on simplified models such as blob diffusion and non-linear advection models in chapter five. We present the simulation results for different transport and confinement states in flux driven thermal convection in chapter six. In the last chapter we perform statistical analysis of long time series from single-point recordings.

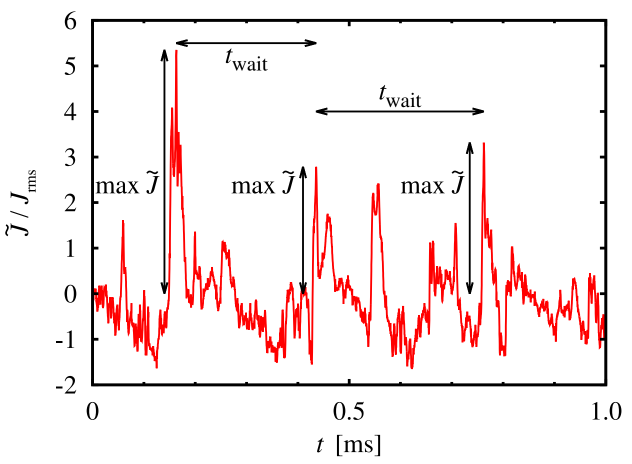


Figure 1.5: Time series of ion saturation current fluctuations of large amplitude bursts [14].

2 Model Equations

In this chapter we present the model equations to be solved numerically. The model equations describe collective motions of non-uniformly magnetized plasmas and are derived from the fluid description of a plasma. In the following we present the derivation of the used model equations step by step.

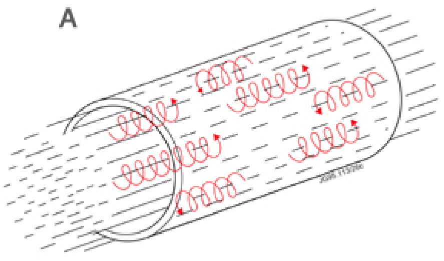
We start with the momentum equation and derive the resulting drifts in toroidal geometry assuming local slab coordinates. The model equations are expressed in terms of the particle continuity equations. In addition we present the normalization used for numerical simulations.

2.1 Model setting

We start the derivation by considering a plasma in a purely toroidal magnetic field, using a cylindrical coordinate system (R, Θ, Z). We further assume that the magnetic field $\mathbf{B}$, given by

$$\mathbf{B} = \frac{B_0 R_0}{R}\mathbf{b}, \tag{2.1}$$

is inhomogeneous and assume low-β plasma, which means that the magnetic field due to internal currents can be neglected. This regime resembles the SOL of large aspect ratio tokamaks. In this magnetized plasma the charged particles are affected by the Lorentz force. If the particle velocity has a component perpendicular to the magnetic field the particles are deflected. That leads to the characteristic gyro-motion of the particles around the magnetic field lines, shown in figure 2.1. We assume that the plasma contains only electrons and



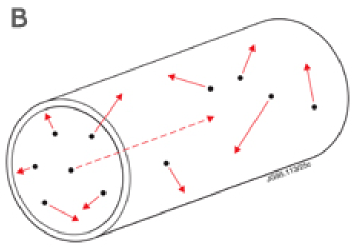


Figure 2.1: Charged particles in presence of a magnetic field on the left side and in absence on the right side [15].

one ion species.

Due to its toroidal geometry the magnetic field is not homogeneous. Consider the curvature vector $\boldsymbol{\kappa} = \mathbf{b} \cdot \nabla \mathbf{b}$ which points radially inwards and opposite to ∇R. The magnetic field decreases as $1/R$ due to the toroidicity which leads to ∇B pointing inwards in the same direction as $\boldsymbol{\kappa}$. We now introduce a slab coordinate system with $\mathbf{x}$ pointing along the $\mathbf{R}$ direction and $\mathbf{y}$ pointing along the $\mathbf{Z}$-direction. This geometry is shown in figure 2.2. The simulation domain is marked by the dashed rectangle. Note that the simulation domain shown in figure 2.2 is not to scale, the characteristic filament length in the SOL in the radial-poloidal plane is usually of the order of 1 cm or smaller [16].

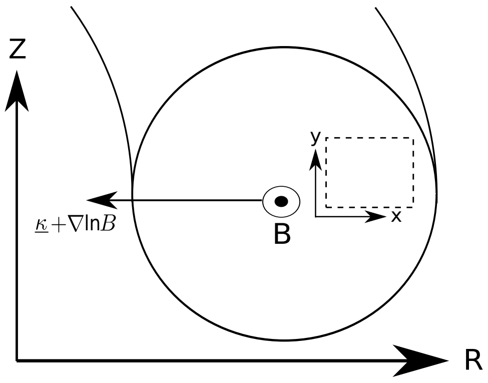


Figure 2.2: Illustration of the simulation domain in a toroidally magnetized plasma [17].

The momentum equation for particle species α is given by

$$\begin{aligned} m_\alpha n_\alpha \left(\frac{\partial}{\partial t} + \mathbf{u}_\alpha \cdot \nabla\right) \mathbf{u}_\alpha = {} & q_\alpha n_\alpha \left(\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}\right) - \nabla p_\alpha \\ & -\nabla \cdot \boldsymbol{\pi}_\alpha - \sum_\beta m_\alpha n_\alpha \nu_{\alpha\beta} \left(\mathbf{u}_\alpha - \mathbf{u}_\beta\right), \end{aligned} \tag{2.2}$$

where m_α labels the mass of one particle of species α, n_α the particle density, $\mathbf{u}_\alpha$ the velocity, q_α the particle charge, $\mathbf{E}$ and $\mathbf{B}$ for the electric and magnetic field respectively, p_α the pressure, $\boldsymbol{\pi}_\alpha$ the viscous stress tensor and $\nu_{\alpha\beta}$ the collision frequency between particle species α and β. The particle species is indexed by $\alpha =$ e,i , denoting electrons and ions, respectively. We further introduce the subscripts $\perp$ and $||$ on vector quantities to symbolize the components perpendicular and parallel to the magnetic field unit vector $\mathbf{b}$, $\mathbf{u}_{||} = \mathbf{b}(\mathbf{u} \cdot \mathbf{b})$ and $\mathbf{u}_\perp = \mathbf{b}{\times}(\mathbf{u}{\times}\mathbf{b})$. In the same way we split up the ∇-operator into $\nabla_\perp$ and $\nabla_{||}$.

The term $\nabla \cdot \boldsymbol{\pi}_\alpha$ describes the momentum transfer due to the change in velocity along different directions. Because of its complex structure we look for a way to approximate this term in an elegant way. In case of strongly magnetized plasmas the gyration frequency ω_c, given by $\omega_c = qB/m$, is large compared to the collision frequency $\nu_{\alpha\beta}$. In this case the gyrating particles complete Larmor gyrations before they collide with another particle. In this particular case we approximate the viscous stress tensor term by

$$\nabla \cdot \boldsymbol{\pi}_\alpha \approx \eta_{\alpha\perp} \nabla_\perp^2 \mathbf{u}_\alpha + \eta_{\alpha||} \nabla_{||}^2 \mathbf{u}_\alpha \tag{2.3}$$

with the viscosity coefficients $\eta_{||}$ and $\eta_\perp$ in parallel and perpendicular direction, respectively. A more detailed discussion of this term is given in [18].

2.2 Drift terms in toroidal geometry

We now determine the dominant cross-field drifts of the plasma in a toroidal geometry. We obtain the expressions of those drifts by crossing the momentum equation 2.2 by the magnetic field vector $\mathbf{B}$ and obtain an expression for the velocity of particle species α perpendicular to the magnetic field in the form

$$\begin{aligned} \mathbf{B} \times \left(m_\alpha n_\alpha \frac{d}{dt} \mathbf{u}_\alpha\right) = {} & \mathbf{B} \times \left[q_\alpha n_\alpha \left(\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}\right)\right] - \mathbf{B} \times \nabla p_\alpha - \mathbf{B} \times \nabla \boldsymbol{\pi}_\alpha \\ & + \mathbf{B} \times m_\alpha n_\alpha \nu_{\alpha\beta} \left(\mathbf{u}_\alpha - \mathbf{u}_\beta\right). \end{aligned} \tag{2.4}$$

We now divide this expression by $m_\alpha n_\alpha$ and introduce the electrostatic potential ϕ, assuming electrostatic perturbations via $\mathbf{E} = -\nabla\phi$. We furthermore divide the fluid velocity into its parallel and perpendicular components. In addition, we use $p_\alpha = n_\alpha T_\alpha$ and assume an isothermal plasma, $\nabla T_\alpha = \mathbf{0}$. Equation 2.4 then reads

$$\begin{aligned} \mathbf{B} \times \frac{d}{dt}\mathbf{u}_\alpha = &- \frac{q_\alpha}{m_\alpha}\left(\mathbf{B} \times \nabla\phi\right) - \frac{T_\alpha}{m_\alpha n_\alpha}\left(\mathbf{B} \times \nabla n_\alpha\right) + \frac{q_\alpha B^2}{m_\alpha}\mathbf{u}_{\alpha\perp} \\ &- \mathbf{B} \times \left(\nabla \cdot \boldsymbol{\pi}_\alpha\right) - \nu_{\alpha\beta}\mathbf{B} \times \left(\mathbf{u}_\alpha - \mathbf{u}_\beta\right) \end{aligned} \tag{2.5}$$

which we rewrite as an expression for the perpendicular velocity

$$\begin{aligned} \mathbf{u}_{\alpha\perp} = &\frac{\left(\mathbf{b} \times \nabla\phi\right)}{B} + \frac{T_\alpha\left(\mathbf{b} \times \nabla n_\alpha\right)}{q_\alpha n_\alpha B} + \frac{\mathbf{b} \times \left(\nabla \boldsymbol{\pi}_\alpha\right)}{\omega_{c\alpha}} \\ &+ \frac{m_\alpha n_\alpha \nu_{\alpha\beta}}{\omega_{c\alpha}}\mathbf{b} \times \left(\mathbf{u}_\alpha - \mathbf{u}_\beta\right) + \frac{1}{\omega_{c\alpha}}\left(\mathbf{b} \times \frac{d}{dt}\mathbf{u}_\alpha\right) \end{aligned} \tag{2.6}$$

where $\omega_{c\alpha}$ is the cyclotron frequency of particle species α. Each term on the right hand side corresponds to a drift perpendicular to the magnetic field which we will now discuss in further detail.

- The first drift on the right hand side given by $\left(\mathbf{b} \times \nabla\phi\right)/B$ is the so-called $E \times B$ or electric drift. It appears in the presence of an electric field and leads to a drift perpendicular to both $\mathbf{E}$ and $\mathbf{B}$. Note that this drift is independent of the particle mass and charge and is therefore the same for all particle species.

- The second drift on the right hand side, $T_\alpha\left(\mathbf{b} \times \nabla n_\alpha\right)/q_\alpha n_\alpha B$, results from the inhomogeneous particle density. This so-called diamagnetic drift does not equal a guiding center motion in contrast to the electric drift but is related to gradient and curvature drifts.

- $\mathbf{b} \times \left(\nabla \cdot \boldsymbol{\pi}_\alpha\right)/\omega_{c\alpha}$ is the viscous drift due to the viscous stress in the plasma.

- The drift $(m_\alpha n_\alpha \nu_{\alpha\beta}/\omega_{c\alpha})\mathbf{b} \times \left(\mathbf{u}_\alpha - \mathbf{u}_\beta\right)$ is the resistive drift resulting from the momentum transfer due to collisions between different particle species.

- The term $1/\omega_{c\alpha}\left(\mathbf{b} \times \mathrm{d}\mathbf{u}_\alpha/\mathrm{d}t\right)$ is called the polarization drift. For a characteristic scale of the perpendicular dynamics ω_α this term is of $\mathcal{O}(\omega_\alpha/\omega_{\alpha c})$.

For a strongly magnetised plasma we assume that the Larmor gyration is the dominant motion perpendicular to the magnetic field, which means that any change in the perpendicular particle velocity takes place on larger time scales than gyration. Under this assumption we order the resistive drift, the viscous drift and the time derivative in equation 2.6 as

$$\mathbf{u}_{\alpha\perp} = \mathbf{u}_{\alpha E} + \mathbf{u}_{\alpha d} + \mathcal{O}(\omega_\alpha/\omega_{c\alpha}) \tag{2.7}$$

which implied drift ordering. Here $\mathbf{u}_E$ denotes the electric drift and $\mathbf{u}_{\alpha d}$ as the diamagnetic drift. Note that the viscous drift $\mathbf{u}_{\pi\alpha}$ is not of lowest order because the tensor itself is of order $\mathcal{O}(\nu/\omega_c)$. In addition the viscous drift is proportional to the particle mass which is the reason why we include this term when we discuss the ion momentum equation. We now insert these lowest order drifts for $\mathbf{u}_\alpha$ into the time derivative term in equation 2.6, to obtain the so called polarisation drift given by

$$\mathbf{u}_{\alpha p} = \frac{m_\alpha}{q_\alpha B}\mathbf{b} \times \left(\frac{\partial}{\partial t} + \mathbf{u}_E \cdot \nabla + \mathbf{u}_{d\alpha} \cdot \nabla + \mathbf{u}_{\alpha\|} \cdot \nabla\right)\left(\mathbf{u}_E + \mathbf{u}_{d\alpha} + \mathbf{u}_{\alpha\|}\right), \tag{2.8}$$

where $\mathbf{u}_{\alpha\|}$ stands for the plasma velocity in parallel direction.

We now take a separate look at the electron and ion momentum equations. In the parallel direction we assume that the ions have no velocity component i.e., $\mathbf{u}_{i\|} = 0$. For Boltzmann distributed electrons of the form $n = n_0 \exp(e\phi/T)$ we can show that the parallel motion of the electrons is zero as well. Another possible assumption for the electron motion is force balance for the electrons $m_e n_e \frac{d}{dt}\mathbf{u}_{e\|} = 0$. We ignore the electron mass and find under the assumption of expression 2.7

$$\mathbf{u}_{e\|} = \frac{e\nabla_\|\phi - T\nabla_\|\ln n}{m_e \nu_{ei}}. \tag{2.9}$$

For the parallel component we use the mass and temperature difference between electrons and ions to simplify the expression 2.6. The electron mass is given by 9.11×10^{-31} kg, the mass of the lightest ion, a proton, is 1.67×10^{-27} kg, which implies that the ion mass is 1839 times larger than the electron mass. In the electron momentum equation we therefore neglect the polarisation drift because it is proportional to the electron mass and thus transfers only little momentum. The electron momentum equation then reads

$$\mathbf{u}_{e\perp} = \frac{1}{B}\mathbf{b} \times \nabla\phi - \frac{T_e}{e n_e B}\mathbf{b} \times \nabla n_e. \tag{2.10}$$

We assume cold ions as to neglect the diamagnetic drift in the ion momentum equation. Writing out the ion momentum equation we get

$$\mathbf{u}_{i\perp} = \frac{1}{B}\mathbf{b} \times \nabla\phi - \frac{e}{m_i B^2}\left(\frac{\partial}{\partial t} + \mathbf{u}_E \cdot \nabla\right)\nabla_\perp\phi + \frac{\mathbf{b} \times (\nabla \cdot \boldsymbol{\pi}_i)}{\omega_{ci}}, \tag{2.11}$$

where the polarisation drift takes the form

$$\mathbf{u}_{pi} = \frac{q}{mB}\mathbf{b} \times \left(\frac{\partial}{\partial t} + \mathbf{u}_E \cdot \nabla\right)\mathbf{u}_E = -\frac{q}{mB^2}\left(\frac{\partial}{\partial t} + \mathbf{b} \times \nabla\phi \cdot \nabla\right)\nabla_\perp\phi \tag{2.12}$$

Note that the cross product commutes with both differential operators for a uniform magnetic field that is constant in time.

We now consider the the charge continuity equation given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = 0. \tag{2.13}$$

Here ρ symbolizes the charge density summed over all particle species and $\mathbf{j}$ is the current density given by

$$\mathbf{j} = \sum_\alpha \mathbf{j}_\alpha = \sum_\alpha n_\alpha \mathbf{u}_\alpha q_\alpha. \tag{2.14}$$

The continuity equation for both particle species now read

$$\frac{\partial n_\mathrm{e}}{\partial t} + \nabla \cdot (n_\mathrm{e}\mathbf{u}_E + n_\mathrm{e}\mathbf{u}_{d\mathrm{e}} + n_\mathrm{e}\mathbf{u}_\mathrm{e})_{||} = 0, \tag{2.15}$$

$$\frac{\partial n_\mathrm{i}}{\partial t} + \nabla \cdot \left(n_\mathrm{i}\mathbf{u}_E + n_\mathrm{i}\mathbf{u}_{p\mathrm{i}} + n_\mathrm{i}\mathbf{u}_{\pi\mathrm{i}}\right) = 0 \tag{2.16}$$

We now assume quasi-neutrality i.e. $n_\mathrm{i} \approx n_\mathrm{e}$ which is valid for $\lambda_s \nabla \ll 1$, where λ_s is the Debye length. In addition we neglect the explicit appearance of space charges. This leads to $\frac{\partial \rho}{\partial t} = 0$. We subtract the continuity equations for electrons and ions from each other and obtain

$$\nabla \cdot \left(n\mathbf{u}_{d\mathrm{e}} + n\mathbf{u}_{||\mathrm{e}} - n\mathbf{u}_{p\mathrm{i}} - n\mathbf{u}_{\pi\mathrm{i}}\right) = 0 \tag{2.17}$$

Because of quasi-neutrality equation 2.17 expresses that the electric current in the plasma is divergence free i.e. $\nabla \cdot \mathbf{j} = \nabla \cdot \left(\mathbf{j}_\perp + \mathbf{j}_{||}\right) = 0$.

We start with the electron continuity equation 2.15 and include this time the collision term from equation 2.2. This comes from the fact that collision between electrons and ions result in a random deflection of the electrons whereas the heavy ions momentum feels almost no change. This net drift acting on the electrons is called the resistive drift and is given to lowest order by

$$\mathbf{u}_{r\perp} = m_\mathrm{e} n \nu_\mathrm{ei} \left(\mathbf{u}_{\mathrm{e}\perp} - \mathbf{u}_{\mathrm{i}\perp}\right) = m_\mathrm{e} \nu_\mathrm{ei}\, \mathbf{b} \times \mathbf{u}_{d\mathrm{e}}, \tag{2.18}$$

where ν_ei is the collision frequency between electrons and ions and $\mathbf{u}_\mathrm{i}$ and $\mathbf{u}_\mathrm{e}$ are approximated by the lowest order drifts. Under this approximations we can derive an expression for the divergence of the resistive drift which takes the form

$$\nabla \cdot (n\mathbf{u}_{r\perp}) = \nabla \cdot \left(m_\mathrm{e} n^2 \nu_\mathrm{ei}\, \mathbf{b} \times \frac{T_\mathrm{e}}{-enB} (\mathbf{b} \times \nabla n)\right) \simeq \frac{m_\mathrm{e}\nu_\mathrm{ei} T_\mathrm{e}}{eB} \nabla_\perp^2 n \tag{2.19}$$

where we neglect the term $(\nabla n)^2$ because we assume that the density fluctuation amplitudes are much smaller than n. As we see the divergence of the resistive drift hives rise to diffusion in the perpendicular plane for the particle density and that its coefficient is of order $\mathcal{O}(\nu_{\mathrm{ei}}/\omega_{\mathrm{ce}})$.

Equation 2.15 can be rewritten in the form

$$
\begin{aligned}
\frac{\partial n}{\partial t} + \nabla \cdot \left(n\mathbf{u}_E + n\mathbf{u}_{d\mathrm{e}} + n\mathbf{u}_{\mathrm{e}\|}\right) &= \left(\frac{\partial}{\partial t} + \mathbf{u}_E \cdot \nabla\right) n + n\nabla \cdot \mathbf{u}_E + \nabla \cdot (n\mathbf{u}_{d\mathrm{e}}) + \nabla \cdot \left(n\mathbf{u}_{\mathrm{e}\|}\right) \\
&= \left(\frac{\partial}{\partial t} + \frac{1}{B}\mathbf{b} \times \nabla\phi \cdot \nabla\right) n + \nabla \cdot (n\mathbf{u}_{d\mathrm{e}}) + \nabla \cdot \left(n\mathbf{u}_{\mathrm{e}\|}\right) \\
&= 0.
\end{aligned}
\tag{2.20}
$$

As we see the divergence of the diamagnetic particle flux occurs. In the following we derive an explicit expression for that. We obtain for a low-β plasma

$$\nabla \cdot (n\mathbf{u}_{d\mathrm{e}}) = \nabla \cdot \left(\frac{T_{\mathrm{e}}}{eB}\mathbf{b} \times \nabla n\right) = \frac{2T_{\mathrm{e}}}{eB}\mathbf{b} \times \nabla \ln B \cdot \nabla n. \tag{2.21}$$

Note that the expression $\nabla \times \mathbf{b}$ can be rewritten as

$$\nabla \times \mathbf{B} = \nabla \times (\mathbf{b}B) = B\,(\nabla \times \mathbf{b}) + (\nabla B) \times \mathbf{b} \simeq \mathbf{0}, \tag{2.22}$$

which is equivalent to

$$\nabla \times \mathbf{b} = -\left(\nabla B \times \mathbf{b}\right)/B = \mathbf{b} \times (\nabla B/B) = \mathbf{b} \times \nabla\, \ln B, \tag{2.23}$$

with

$$\nabla \times \mathbf{b} = \mathbf{b} \times \boldsymbol{\kappa}. \tag{2.24}$$

In case of a purely toroidal magnetic field the field is given by

$$\mathbf{B} = -B\,\widehat{\boldsymbol{\Theta}}. \tag{2.25}$$

The $\nabla \ln B$ then reads

$$\nabla \ln B = \frac{\nabla B}{B} = \frac{1}{B}\frac{\partial B}{\partial R}\widehat{\mathbf{R}} = -\frac{1}{R}\widehat{\mathbf{R}}. \tag{2.26}$$

For a inhomogeneous toroidal field in this geometry with the magnetic field in the negative $\boldsymbol{\theta}$-direction the divergence of the diamagnetic electron drift takes the form

$$\nabla \cdot (n\,\mathbf{u}_{d\mathrm{e}}) = \frac{2T_{\mathrm{e}}}{eBR}\frac{\partial n}{\partial Z}. \tag{2.27}$$

Next we calculate the divergence of the ion polarisation drift from equation 2.12:

$$
\begin{aligned}
\nabla \cdot \left(n\,\mathbf{u}_{p\mathrm{i}}\right) &= \nabla \cdot \left(-\frac{m_{\mathrm{i}}n}{eB^2}\left(\frac{\partial}{\partial t} + \mathbf{u}_E \cdot \nabla\right)\nabla_{\perp}\phi\right) \\
&= \frac{m_{\mathrm{i}}n}{eB^2}\left(\nabla \ln n + \nabla\right) \cdot \left(\frac{\partial}{\partial t} + \frac{1}{B}\mathbf{b} \times \nabla_{\perp}\phi \cdot \nabla\right)\nabla_{\perp}\phi.
\end{aligned}
\tag{2.28}
$$

Note that the nabla operator acts on both the particle density n and on ϕ where it commutes to lowest order with the total time derivative.

In the same manner we take a look at the divergence of the viscous drift, which results in

$$\nabla \cdot (n\,\mathbf{u}_{\boldsymbol{\pi}}) = \nabla \cdot (n\eta_\perp\, \mathbf{b} \times \mathbf{u}_E) = \frac{n}{B}\eta_\perp \nabla_\perp^4 \phi \tag{2.29}$$

where we neglect the density gradients and replace it by the approximation $\nabla \cdot \mathbf{u}_{\boldsymbol{\pi}} \approx \eta_\perp \nabla_\perp^4 \phi$.

Next we calculate the divergence of the electric drift and find

$$\nabla \cdot \mathbf{u}_E = \nabla \cdot \left(\frac{1}{B} \times \nabla\phi\right) = \frac{1}{B}\mathbf{b} \times \nabla \ln B \cdot \nabla\phi + \frac{1}{B}\nabla \times \mathbf{b} \cdot \nabla\phi \tag{2.30}$$

where the first term descends from the divergence of the current and the second term results from the field curvature in the toroidal geometry. For a toroidal field along the negative $\boldsymbol{\Theta}$-axis and find

$$\nabla \cdot \mathbf{u}_E = -\frac{2}{BR}\frac{\partial\phi}{\partial Z}. \tag{2.31}$$

With these expressions for the compressions of the different drifts we are able to assemble them to the model equations.

2.3 Two field equations

We now introduce the vorticity Ω, which is defined as $\Omega = \mathbf{b} \cdot \nabla \times \mathbf{u}_E$. This quantity is often used in fluid dynamics to describe the rotational motion of fluids and plasmas. For the electric drift velocity we can show that $\Omega = \mathbf{b}(\nabla^2\phi)/B$ holds to lowest order:

$$\begin{aligned} \nabla \times \mathbf{u}_E &= \nabla \times \left(\frac{1}{B}\mathbf{b} \times \nabla\phi\right) \\ &= \frac{1}{B}(\nabla \cdot \nabla_\perp\phi)\,\mathbf{b} - \nabla_\perp\phi\left(\nabla \cdot \frac{1}{B}\mathbf{b}\right) + (\nabla_\perp\phi \cdot \nabla)\,\frac{1}{B}\mathbf{b} - \left(\frac{1}{B}\mathbf{b} \cdot \nabla\right)\phi \\ &\simeq \frac{1}{B}\nabla_\perp^2\phi \end{aligned} \tag{2.32}$$

because we assumed a homogeneous magnetic field, which implies that the characteristic length scale of B is much larger than the scale on which the electrostatic potential varies, i.e. $\mathcal{O}(\nabla B) = 1/L, \mathcal{O}(\nabla\phi) = 1/\ell$ with $\nabla = \frac{\partial}{\partial \mathbf{x}} \sim 1/\text{length}$ and $\frac{\ell}{L} \ll 1$ where ℓ is the characteristic length scale for the electrostatic potential where we assume $\ell/R \ll 1$. Additionally we used that the gradients of the field are of the same magnitude as the fields themselves.

If we now insert equations 2.27 and 2.28 into 2.15 and 2.17 to get the two-field model for strongly magnetised plasma

$$\left(\frac{\partial}{\partial t} + \mathbf{u}_E \cdot \nabla\right) n + \frac{2n}{BR}\frac{\partial \phi}{\partial Z} + \frac{2T}{eBR}\frac{\partial n}{\partial Z} = \frac{T_\mathrm{e}}{m_\mathrm{e}\nu_\mathrm{ei}}\nabla_\parallel^2\left(\frac{e\phi}{T} - \ln n\right) + \frac{m\nu_\mathrm{ei}T}{qB}\nabla_\perp^2 n \tag{2.33a}$$

$$\frac{m_\mathrm{i}n}{eB^2}(\nabla \ln n + \nabla)\cdot\left(\frac{\partial}{\partial t} + \mathbf{u}_E \cdot \nabla\right)\nabla_\perp\phi + \frac{2T}{eBR}\frac{\partial n}{\partial Z} = \frac{T_\mathrm{e}}{m_\mathrm{e}\nu_\mathrm{ei}}\nabla_\parallel^2\left(\frac{e\phi}{T} - \ln n\right) + \eta_\perp\nabla_\perp^4\phi. \tag{2.33b}$$

As we see these two coupled non-linear partial differential equations for the evolution of the physical fields of the particle density n and the the electrostatic potential ϕ is fairly complex. We therefore assume that the length on which the density varies is much smaller than the length scale of the electrostatic potential. We therefore can neglect the first term on the left hand side of equation 2.33b.

We now take a closer look at the electric drift advection terms $\mathbf{u}_E \cdot \nabla n$ and $\mathbf{u}_E \cdot \nabla_\perp^2\phi$. These non-linear terms introduce a coupling between all length scales of the system. These terms therefore require special treatment in the numerical simulation that conserves energy to produce accurate results. We will introduce a solution for this case in the chapter **Numerical Methods**.

We insert the definition of the electric drift and the vorticity Ω, and neglect the parallel current since we are only interested in the perpendicular motion and obtain

$$\begin{aligned}\left(\frac{\partial}{\partial t} + \frac{1}{B}\mathbf{z}\times\nabla\phi\cdot\nabla\right) n + \frac{2}{BR}\frac{\partial \phi}{\partial Z} + \frac{2T}{eBR}\frac{\partial n}{\partial Z} &= \chi\nabla_\perp^2 n \\ \left(\frac{\partial}{\partial t} + \frac{1}{B}\mathbf{z}\times\nabla\phi\cdot\nabla\right)\Omega + \frac{2T}{mnR}\frac{\partial n}{\partial Z} &= \eta\nabla_\perp^2\Omega.\end{aligned} \tag{2.34}$$

Here the dissipation coefficient η stands for the kinematic viscosity of the fluid and the other dissipation coefficient χ is the collisional diffusivity. Note that both equations include the non-linear advection by the electric drift.

2.4 Dimensionless variables

In order to reduce the number of model parameters we introduce dimensionless variables. We therefore take advantage of the fact that each physical quantity can be normalized by a characteristic value.

We introduce the following dimensionless variables:

$$\begin{gathered}\widehat{t} = \gamma t \quad \widehat{Z} = \frac{Z}{\ell} \quad \widehat{\phi} = \frac{\phi}{\gamma B \ell^2} \\ \widehat{\nabla} = \ell \nabla \quad \widehat{n} = \frac{n}{N} \quad \widehat{R} = \frac{R}{\ell},\end{gathered} \tag{2.35}$$

where γ is the characteristic freqency, ℓ is a characteristic length scale and N a characteristic particle density. We substitute these variables into equations 2.34 and divide the particle continuity equation by $N\gamma$ to normalize the first time derivative-term of the density function which leads for the second term to

$$\frac{1}{N\gamma} n \frac{2}{BR} \frac{\partial \phi}{\partial Z} = \frac{1}{N\gamma} \frac{\gamma B \ell^2}{\ell} \frac{2n}{BR} \frac{\partial \widehat{\phi}}{\partial \widehat{Z}} = \frac{2\,\ell}{R} \widehat{n} \frac{\partial \widehat{\phi}}{\partial \widehat{Z}}. \tag{2.36}$$

The factor $2\ell/R$ is small as we assume that the characteristic length scale ℓ of the system is much smaller than the major radius R. We therefore neglect this term in the model equations, which arises from compression of the electric drift.

The third term of the density equation becomes

$$\frac{1}{N\gamma} \frac{2T_\mathrm{e}}{eBR} \frac{\partial n}{\partial Z} = \frac{2T m_\mathrm{i}}{\gamma eBR\,\ell m_\mathrm{i}} \frac{\partial \widehat{n}}{\partial \widehat{Z}} = \frac{\gamma}{\omega_\mathrm{ci}} \frac{\partial \widehat{n}}{\partial \widehat{Z}} = \sqrt{\frac{2\rho_s^2}{R\,\ell}} \frac{\partial \widehat{n}}{\partial \widehat{Z}}, \tag{2.37}$$

where $(2\rho_s^2/R\ell)^{1/2}$ is small and will therefore be neglected as well.

For the vorticity expression we obtain for the second term

$$\frac{2T_\mathrm{e}}{m_\mathrm{i}R} \frac{1}{\gamma^2 \ell^2 N} \frac{\partial n}{\partial Z} = \frac{2T_\mathrm{e}}{m_\mathrm{i}R\gamma^2\ell} \frac{\partial \widehat{n}}{\partial \widehat{Z}} \tag{2.38}$$

which leads us to the dimensionless model equations

$$\begin{aligned}\left(\frac{\partial}{\partial \widehat{t}} + \widehat{\mathbf{z}} \times \widehat{\nabla} \widehat{\varphi} \cdot \widehat{\nabla}\right) \widehat{n} &= \widehat{\kappa}\, \widehat{\nabla}_\perp^2 \widehat{n} \\ \left(\frac{\partial}{\partial \widehat{t}} + \widehat{\mathbf{z}} \times \widehat{\nabla} \widehat{\varphi} \cdot \widehat{\nabla}\right) \widehat{\Omega} + \frac{2T_\mathrm{e}}{\gamma^2 m_\mathrm{i} \ell R} \frac{\partial}{\partial \widehat{Z}} \widehat{n} &= \widehat{\mu}\, \widehat{\nabla}_\perp^2 \widehat{\Omega}\end{aligned} \tag{2.39}$$

where we summarize the remaining pre-factors on the right side of the equations in $\hat{\kappa}$ respectively $\hat{\eta}$. We chose γ^2 to be

$$\gamma^2 = \frac{2T_\mathrm{e}}{m_\mathrm{i}\ell R} \tag{2.40}$$

so that we obtain no pre-factor for the collective dynamics. We now introduce local slab coordinates as shown in figure 2.2 and insert the Poisson bracket as

$$\{\omega, \phi\} = \left(\frac{\partial \omega}{\partial x} \frac{\partial \phi}{\partial y} - \frac{\partial \omega}{\partial y} \frac{\partial \phi}{\partial x} \right). \tag{2.41}$$

For simplicity reasons we drop the hat notation in equation 2.39 which leads to the final model equations:

$$\begin{aligned} \frac{\partial n}{\partial t} + \{\phi, n\} &= \kappa \nabla_\perp^2 n, \\ \frac{\partial \Omega}{\partial t} + \{\phi, \Omega\} + \frac{\partial n}{\partial y} &= \mu \nabla_\perp^2 \Omega. \end{aligned} \tag{2.42}$$

Note that for each additional term the same normalizations as in expression 2.35 have to be applied. These two coupled differential equations can now be solved as an initial value problem with suitable boundary conditions.

This model is analogous to the Rayleigh-Bénard convection model which uses other coefficients, the so called Rayleigh number Ra and the Prandtl number Pr. These coefficients are related to the friction coefficients for each variable used in Equations 2.42 as

$$Ra = \frac{1}{\kappa \mu} \qquad Pr = \frac{\kappa}{\mu}. \tag{2.43}$$

The concrete meaning of the Rayleigh number is the ratio between diffusion which acts as a stabilizing force to the system, and the buoyancy which tends to destabilize the system. The Prandtl number represents the ration between viscosity and collisional diffusion [19].

3
Numerical Methods

In this chapter we describe the numerical methods implemented in the two-dimensional advection-diffusion solver, abbreviated 2dads [20]. We use the code to integrate the model equations 2.42 in time. They are both advection-diffusion equations of the form

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = \kappa \nabla^2 u + \mathcal{L}(u) \tag{3.1}$$

where $\mathcal{L}$ is a well defined differential operator acting on the variable u. We require that u is periodic in one direction. The code uses finite difference approximations in one spatial direction and spectral expansion in the other to solve the model equations on a rectangular domain. As part of the work presented in this thesis, the majority of the methods have been re-implemented on graphical processing units. We use the NVIDIA CUDA Fast Fourier Transform library (cuFFT) [21] for spectral transformations and the NVIDIA cuSOLVER library [22] for matrix factorisations. We further present a stiffly stable scheme for the time integration that treats diffusion implicitly. In addition we present the energy and enstrophy conserving finite difference scheme used for non-linear advection terms and discuss both Dirichlet and Neumann boundary conditions.

A large portion of the following methods are derived from the documentation for the 2dads code written by Odd Erik Garcia and originally implemented in a low level Fortran code [20].

3.1 Simulation domain

The model equations 2.42 are discretized on a rectangular domain of the size $(-L_x/2, L_x/2)\times(-L_y/2, L_y/2)$. We use a cell centered grid with N_x equidistant grid points in x-direction and M_y equidistant grid points in y-direction. The position of the grid points is given by

$$x_n = -\frac{L_x}{2} + (n - \frac{1}{2})\,\Delta_x \text{ for } n = 0, \ldots, N_x + 1, \tag{3.2a}$$

$$y_n = -\frac{L_y}{2} + (n - \frac{1}{2})\,\Delta_y \text{ for } n = 0, \ldots, M_y + 1, \tag{3.2b}$$

where $\Delta_x = L_x/N_x$ and $\Delta_y = L_y/M_y$. Here we have implicitly introduce ghost points just outside of the physical domain as shown in Figure 3.1, so that we have $(N_x + 2) \times (M_y + 2)$ discretization points in total.

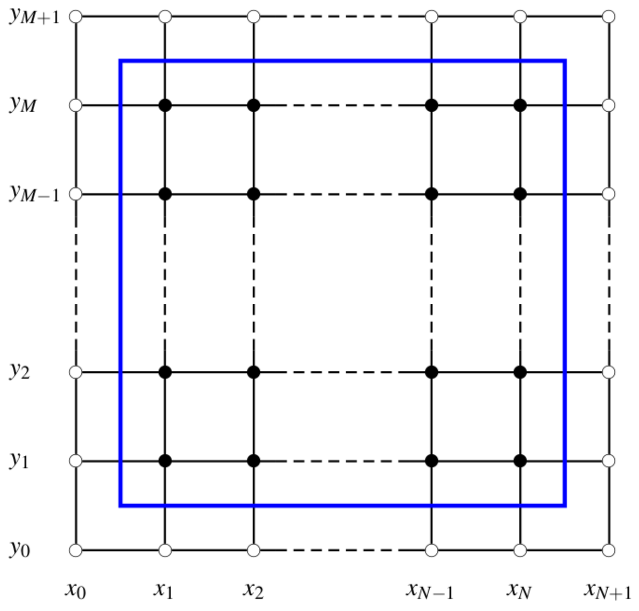


Figure 3.1: Illustration of the two dimensional simulation domain using a cell centered grid and ghost points. The grid points inside the physical domain are represented as filled circles and the ghost points by open circles [20].

Figure 3.2 shows how a one-dimensional function is discretisized on a cell centered grid with the ghost points $x_0 = x_N$ and $x_{N+1} = x_1$. This concept can easily be expanded to two dimensions.

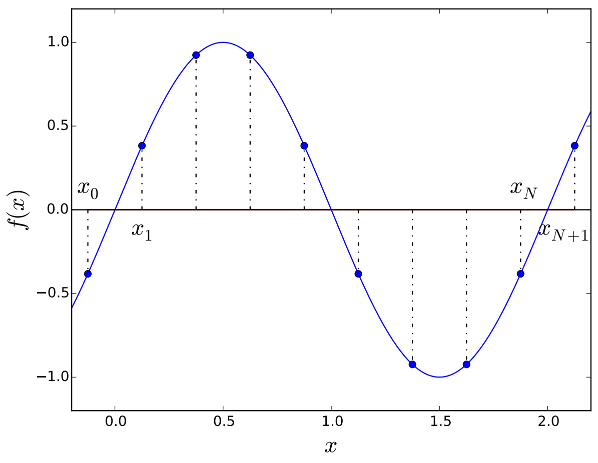


Figure 3.2: Discretisation of a sin-function with the period $L = 2$ on a cell centered grid with 8 gridpoints in the domain (0,2) [17].

3.2 Finite differences

The 2dads code uses finite difference approximations in the x-direction, which have the advantage that they are easy to implement, computationally fast and easily adaptable to boundary conditions [23]. We define the centered difference approximation for the first order derivative as

$$\frac{\partial u}{\partial x}(x_n) = \frac{1}{2\Delta_x}\left[u_{n+1} - u_{n-1}\right] + \mathcal{O}(\triangle_x^2), \tag{3.3}$$

where u_n is the discretized function value at the grid point x_n. Note that this scheme is of second order accuracy. Analogously we approximate the second order derivative by centered differences as

$$\frac{\partial^2 u}{\partial x^2}(x_n) = \frac{1}{\Delta_x^2}\left[u_{n-1} - 2u_n + u_{n+1}\right] + \mathcal{O}(\triangle_x^2). \tag{3.4}$$

Note that both approximations only involve function values at x_n and the neighbour points x_{n-1} and x_{n+1}.

The schemes 3.6 and 3.4 have been implemented for graphical processing units. To verify the implementation we compare the numerically obtained derivative to the analytical derivative of the test function $f(x) = \sin(2\pi x)$ with $x \in (0, 1)$. We determine the average error by calculating the error at each grid point by $|f_{num} - f_{an}|$ and determine the average value. We wary the resolution of the numerical derivative in order to verify that 3.6 has a quadratically decreasing error as shown in figure 3.3.

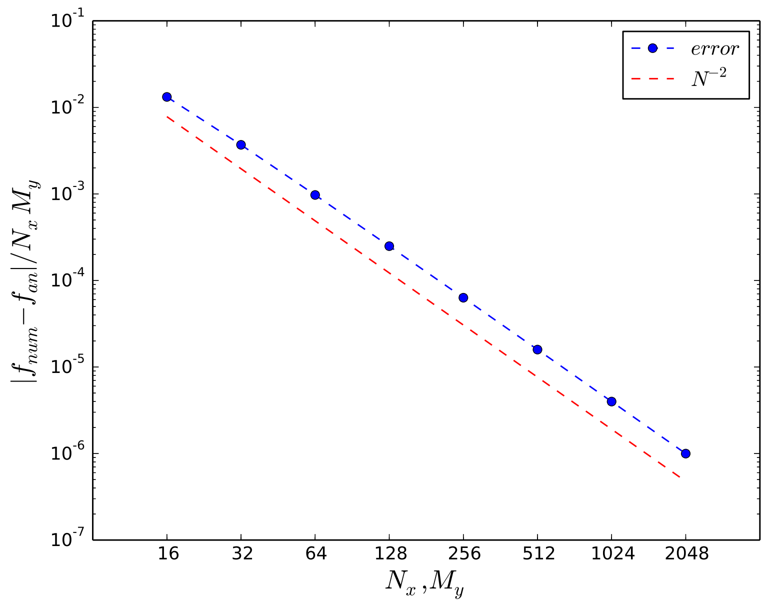


Figure 3.3: Convergence rate of the finite difference method at the example of a first order derivative of a sin function.

We now test the convergence rate of the finite difference method at the example of a second order derivative 3.4 in figure 3.4 tested with the same test function as in figure 3.3. Again, the error decreases quadratically.

3.3 Ghost points

We now discuss the boundary conditions. For Dirichlet boundary conditions the values on the boundary are specified. We use a linear extrapolation across the boundary of the domain to relate the values of the domain boundaries to the value at the cell centers by

$$U_{1/2} = \frac{1}{2}(u_0 + u_1) + \mathcal{O}(\triangle_x^2), \tag{3.5a}$$

$$U_{N+1/2} = \frac{1}{2}(u_N + u_{N+1}) + \mathcal{O}(\triangle_x^2). \tag{3.5b}$$

From these equations we evaluate the function values at the ghost points to be

$$u_0 = 2U_{1/2} - u_1 + \mathcal{O}(\triangle_x^2), \tag{3.6a}$$

$$u_{N+1} = 2U_{N+1/2} - u_N + \mathcal{O}(\triangle_x^2). \tag{3.6b}$$

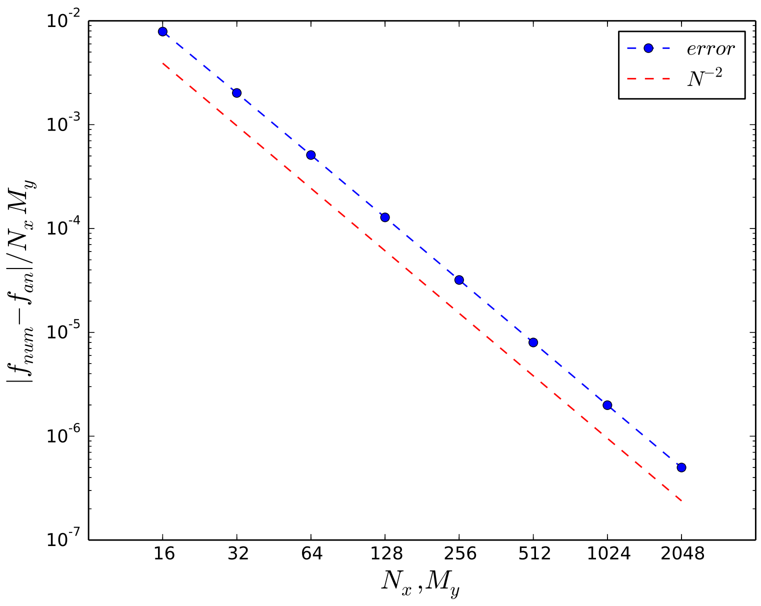


Figure 3.4: Convergence rate of the finite difference method at the example of a second order derivative of a sin function.

In case of Neumann boundary conditions where the gradient of the function at the boundary is specified we use equation 3.6 to obtain

$$U'_{1/2} = \frac{1}{\Delta_x}(u_1 - u_0) + \mathcal{O}(\triangle_x^2), \tag{3.7a}$$

$$U'_{N+1/2} = \frac{1}{\Delta_x}(u_{N+1} - u_N) + \mathcal{O}(\triangle_x^2). \tag{3.7b}$$

This yields

$$u_0 = u_1 - \Delta_x U'_{1/2} + \mathcal{O}(\triangle_x^2), \tag{3.8a}$$

$$u_{N+1} = u_N + \Delta_x U'_{N+1/2} + \mathcal{O}(\triangle_x^2) \tag{3.8b}$$

for the numerical values at the ghost points.

Given a numerical solution for the internal domain and the boundary conditions, the solution at the ghost points can be found from the above equations.

3.4 Spectral transformations

The only natural boundary conditions in y-direction is to assume periodic boundaries. As part of this thesis work, we implemented a spectral transformation along the y-direction. Spectral transformations make it relatively easy to implement derivatives and the rate of convergence is better than the one of finite differences schemes, partly explaining the widespread use of periodic boundary conditions in numerical simulations [23].

We consider a function $U(y)$ which is periodic in its argument y with periodicity length L_y, $U(y) = U(y + L_y)$. Furthermore we define the discrete sampled values as $U_m = U(y_m)$ and $y_m = m\Delta_y$ for $m = 0, \dots, M_y - 1$. We now calculate the Fourier coefficients by a discrete Fourier transform (DFT) by

$$\widehat{U}_m = \sum_{n=0}^{M_y-1} U_n \exp\left(\frac{i2\pi nm}{M_y}\right) \quad \text{for} \quad m = 0, \dots, M_y - 1. \tag{3.9}$$

The inverse transform (IDFT) is given by

$$U_m = \frac{1}{M_y} \sum_{n=0}^{M_y-1} \widehat{U}_n \exp\left(-\frac{i2\pi nm}{M_y}\right) \quad \text{for} \quad m = 0, \dots, M - 1. \tag{3.10}$$

To see that 3.10 is indeed the inverse transformation of 3.9 we write

$$\begin{aligned}
\frac{1}{M_y} \sum_{n=0}^{M_y-1} \exp\left(-\frac{i2\pi nm}{M_y}\right) \widehat{U}_n &= \frac{1}{M_y} \sum_{n=0}^{M_y-1} \exp\left(-\frac{i2\pi nm}{M_y}\right) \sum_{\ell=0}^{M_y-1} U_\ell \exp\left(\frac{i2\pi n\ell}{M_y}\right) \\
&= \frac{1}{M_y} \sum_{\ell=0}^{M_y-1} U_\ell \sum_{n=0}^{M_y-1} \exp\left(\frac{i2\pi n(\ell - m)}{M_y}\right) \\
&= \frac{1}{M_y} \sum_{\ell=0}^{M_y-1} U_\ell \sum_{n=0}^{M_y-1} \left\{\exp\left(\frac{i2\pi(\ell - m)}{M_y}\right)\right\}^n \\
&= \frac{1}{M_y} \sum_{\ell=0}^{M_y-1} M_y U_\ell \delta_{m,\ell} \\
&= U_m.
\end{aligned} \tag{3.11}$$

On a computer the $\widehat{U}_n$ can be computed from the U_n, or vice versa, in $\mathcal{O}(N \log N)$ operations by the fast Fourier transform (FFT) [24], which makes working with Fourier series practical. The 2dads code uses the NVIDIA CUDA Fast Fourier Transform library (cuFFT) which provides routines for arbitrary size DFTs for graphical processing units [21].

We now describe how to calculate the derivatives $U'(y)$ and $U''(y)$. The first derivative of U evaluated at the sample point y_m is:

$$\begin{aligned}U'_m =&U'(mL_y/M_y)\\ =&\frac{1}{M_y}\sum_{0<m<M_y/2}\frac{2\pi i}{L_y}m\left(\widehat{U}_m\exp\left(-\frac{2\pi i}{M_y}mn\right)-\widehat{U}_{M_y-m}\exp\left(+\frac{2\pi i}{M_y}mn\right)\right)\\ =&\frac{1}{M_y}\sum_{m=0}^{M_y-1}\widehat{U}'_m\exp\left(-\frac{2\pi i}{M_y}nm\right).\end{aligned} \tag{3.12}$$

Note that the $\widehat{U}'_{N_y/2}$ term vanishes. If we want to calculate the first derivative $U'(y)$ from $U(y)$ we therefore use a FFT to compute $\widehat{U}_m$ for $0 \le m < M_y$. Then we multiply $\widehat{U}_m$ by $2\pi i/L_y m$ for $m < M_y/2$, by $2\pi i/L_y(m-M_y)$ for $m > M_y/2$, and zero for $m = M_y/2$ (if M_y is even). Finally we compute U'_n from $\widehat{U}'_n$ via an inverse FFT.

On the other hand, the second derivation of $U(y_m)$ is given by:

$$\begin{aligned}U''_m =&U''(mL_y/M_y)\\ =&-\frac{1}{M_y}\sum_{0<m<M_y/2}\left[\frac{2\pi}{L_y}m\right]^2\left(\widehat{U}_m\exp\left(-\frac{2\pi i}{M_y}mn\right)+\widehat{U}_{M_y-m}\exp\left(+\frac{2\pi i}{M_y}mn\right)\right)\\ &-\left[\frac{\pi}{L_y}M_y\right]^2\widehat{U}_{M_y/2}(-1)^n\\ =&\frac{1}{M_y}\sum_{m=0}^{M_y-1}\widehat{U}''_m\exp\left(-\frac{2\pi i}{M_y}nm\right).\end{aligned} \tag{3.13}$$

Note that the $\widehat{U}_{M_y/2}$ term does not vanish. To calculate the second derivative $U''(y)$ from $U(y)$ we use the same procedure as to calculate the first derivative but multiply $\widehat{U}_m$ by $-\left[\frac{2\pi}{L_y}m\right]^2$ for $m \le M_y/2$ and by $-\left[\frac{2\pi}{L_y}(m-M_y)\right]^2$ for $m > M_y/2$ to obtain $U''(y)$ [25].

We now again take a look at the convergence rate of the spectral derivation scheme at the example of the first derivation of a one-dimensional gaussian function $f(x) = \exp\left(-x^2/2\right)$ with $x \in (-10, 10)$. The numerical solution is compared to the analytical one and the average error given is plotted against the resolution of the simulation domain in Figure 3.5. The error converges against the machine epsilon, note that the error decreases much faster than at the finite difference scheme in Figure 3.4. For completeness we take a look at the second derivative for the same function as well. The outcome is given in

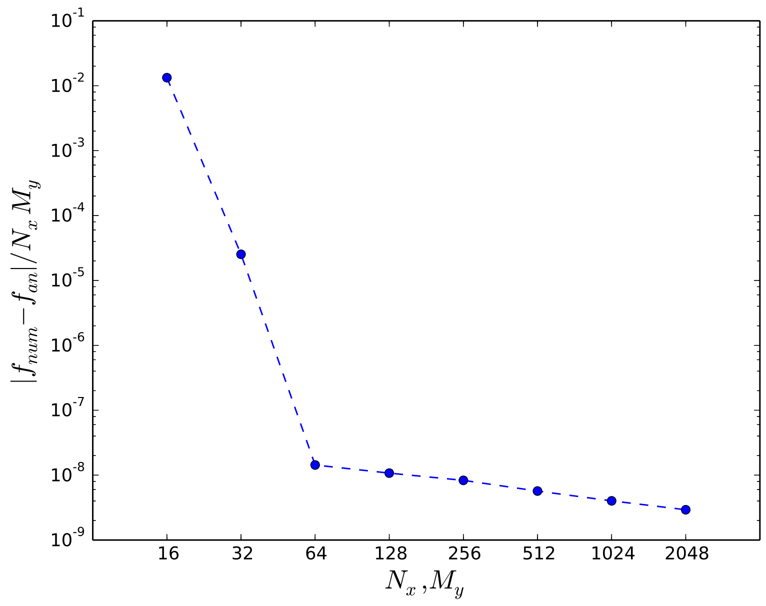


Figure 3.5: Convergence rate of the spectral derivation scheme at the example of a first order derivative of a Gaussian function.

Figure 3.6. As expected the convergence rate for the second derivative behaves the same as for the first derivative.

3.5 Elliptic equations

For calculating the electrostatic potential from the vorticity in equation 8.1 we have to solve the elliptic equation $\partial^2\Phi/\partial x^2 = \Omega$. We discretize this equation by approximating the second order derivative as in equation 3.4 and obtain $(\phi_{n-1} - 2\phi_n + \phi_{n+1}) = \triangle_x^2 \omega_n$ where ϕ and ω represent the numerical approximation to the exact function values. Now we rewrite this expression in matrix notation to

$$\begin{pmatrix} \ddots & \ddots & \ddots & & & & \\ & 1 & -2 & 1 & & & \\ & & 1 & -2 & & & \\ & & & 1 & -2 & 1 & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ \phi_{n-1} \\ \phi_n \\ \phi_{n+1} \\ \vdots \end{pmatrix} = \triangle_x^2 \begin{pmatrix} \vdots \\ \omega_{n-1} \\ \omega_n \\ \omega_{n+1} \\ \vdots \end{pmatrix}. \tag{3.14}$$

Thus, the elliptic equation may be solved by inverting the triangular matrix. To consider the boundary conditions we need to modify the first and the last

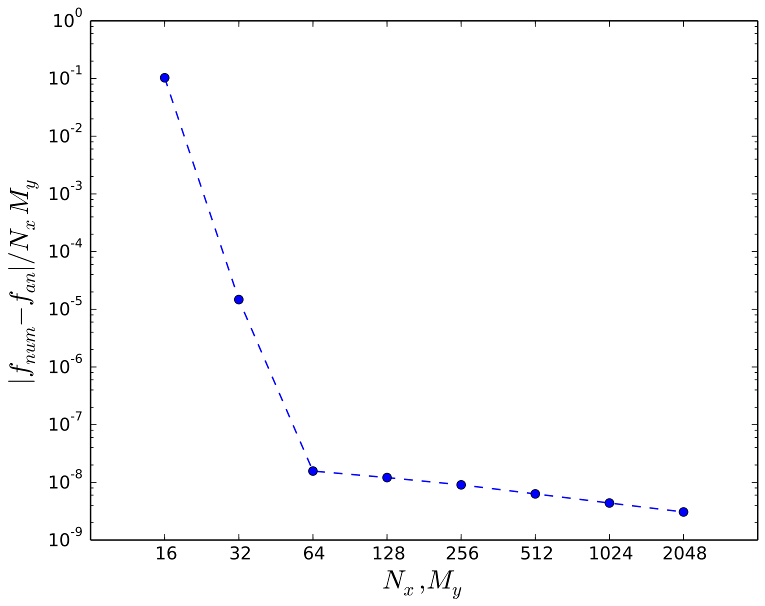


Figure 3.6: Convergence rate of the spectral derivation scheme at the example of a second order derivative of a Gaussian function.

row of the matrix. We use equation 3.6 for Dirichlet boundary conditions and obtain

$$-3\phi_1 + \phi_2 = \triangle_x^2 \omega_1 - 2\Phi_{1/2}, \tag{3.15a}$$

$$\phi_{N-1} - 3\phi_N = \triangle_x^2 \omega_N - 2\Phi_{N+1/2}. \tag{3.15b}$$

In matrix form the finite difference approximation of the elliptic equation with Dirichlet boundary conditions takes therefore the form

$$\begin{pmatrix} -3 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -3 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N-1} \\ \phi_N \end{pmatrix} = \triangle_x^2 \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_{N-1} \\ \omega_N \end{pmatrix} - \begin{pmatrix} 2\Phi_{1/2} \\ 0 \\ \vdots \\ 0 \\ 2\Phi_{N+1/2} \end{pmatrix}. \tag{3.16}$$

For Neumann boundary conditions the first and the last rows read

$$-\phi_1 + \phi_2 = \triangle_x^2 \omega_1 + \triangle_x \Phi'_{1/2}, \tag{3.17a}$$

$$\phi_{N-1} - \phi_N = \triangle_x^2 \omega_N - \triangle_x \Phi'_{N+1/2}. \tag{3.17b}$$

Again we write this in matrix form as

$$\begin{pmatrix} -1 & 1 & & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -1 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N-1} \\ \phi_N \end{pmatrix} = \triangle_x^2 \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_{N-1} \\ \omega_N \end{pmatrix} + \triangle_x \begin{pmatrix} \Phi'_{1/2} \\ 0 \\ \vdots \\ 0 \\ \Phi'_{N+1/2} \end{pmatrix}. \tag{3.18}$$

Note that the boundary conditions themselves appear explicitly as a source together with the vorticity.

In case of a two dimensional problem there are several possibilities to solve the elliptic equation. The approach we present in the following solves the problem approximating the second order derivative in x-direction by central differences as given in 3.4 and treating the y-direction spectrally. In this case the potential Φ is given by the two dimensional elliptic equation

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \Phi = \Omega. \tag{3.19}$$

Assuming Φ and Ω is periodic in y with periodicity length L_y we take a discrete spectral transformation as given in 3.9 in y-direction which leads us to

$$\left(\frac{\partial^2}{\partial x^2} - k_y^2 \right) \widehat{\Phi}_m = \widehat{\Omega}_m. \tag{3.20}$$

where the wave number k_y is given by $k_y = m \frac{2\pi}{L_y}$ and $m = -\frac{M_y}{2}+1, \ldots, 0, \ldots, \frac{M_y}{2}$. In matrix notation this can be written as

$$\begin{pmatrix} \ddots & \ddots & \ddots & & & \\ 1 & -2 - \triangle_x^2 k_y^2 & 1 & & & \\ & 1 & -2 - \triangle_x^2 k_y^2 & 1 & & \\ & & 1 & -2 - \triangle_x^2 k_y^2 & 1 & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ \widehat{\phi}_{n-1} \\ \widehat{\phi}_n \\ \widehat{\phi}_{n+1} \\ \vdots \end{pmatrix}$$
$$= \triangle_x^2 \begin{pmatrix} \vdots \\ \widehat{\omega}_{n-1} \\ \widehat{\omega}_n \\ \widehat{\omega}_{n+1} \\ \vdots \end{pmatrix} \tag{3.21}$$

which has to be solved for each wave number k_y.

As in the one dimensional case, the boundary conditions change the first and the last rows of the matrix. For Dirichlet boundary conditions we use 3.6 to obtain

$$-\left(3+\triangle_x^2 k_y^2\right)\widehat{\phi}_1+\widehat{\phi}_2=\triangle_x^2\widehat{\omega}_1-\delta_{k_y}2\widehat{\Phi}_{1/2}, \tag{3.22a}$$

$$\widehat{\phi}_{N-1}-\left(3+\triangle_x^2 k_y^2\right)\widehat{\phi}_N=\triangle_x^2\widehat{\omega}_N-\delta_{k_y}2\widehat{\Phi}_{N+1/2} \tag{3.22b}$$

which reads in matrix form

$$\begin{pmatrix} -3-\triangle_x^2k_y^2 & 1 & & & & \\ & 1 & -2-\triangle_x^2k_y^2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2-\triangle_x^2k_y^2 & 1 \\ & & & & 1 & -3-\triangle_x^2k_y^2 \end{pmatrix}\begin{pmatrix}\widehat{\phi}_1\\ \widehat{\phi}_2\\ \vdots\\ \widehat{\phi}_{N-1}\\ \widehat{\phi}_N\end{pmatrix}.$$

$$=\triangle_x^2\begin{pmatrix}\widehat{\omega}_1\\ \widehat{\omega}_2\\ \vdots\\ \widehat{\omega}_{N-1}\\ \widehat{\omega}_N\end{pmatrix}-\delta_{k_y}\begin{pmatrix}2\widehat{\Phi}_{1/2}\\ 0\\ \vdots\\ 0\\ 2\widehat{\Phi}_{N+1/2}\end{pmatrix}.$$

where δ_{k_y} is the Kronecker delta given by

$$\delta_{k_y}=\begin{cases}1, & \text{if } k_y=0,\\ 0, & \text{if } k_y\neq 0.\end{cases} \tag{3.23}$$

In the case of Neumann boundary conditions we use 3.7 to get

$$-\left(3+\triangle_x^2 k_y^2\right)\widehat{\phi}_1+\widehat{\phi}_2=\triangle_x^2\widehat{\omega}_1+\delta_{k_y}\triangle_x^2\widehat{\Phi}'_{1/2}, \tag{3.24a}$$

$$\widehat{\phi}_{N-1}-\left(3+\triangle_x^2 k_y^2\right)\widehat{\phi}_N=\triangle_x^2\widehat{\omega}_N+\delta_{k_y}\triangle_x^2\widehat{\Phi}'_{N+1/2} \tag{3.24b}$$

In matrix form the elliptic equation with fixed stream function gradients on the boundaries can therefore be written as

$$\begin{pmatrix} -3-\triangle_x^2 k_y^2 & 1 & & & \\ 1 & -2-\triangle_x^2 k_y^2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2-\triangle_x^2 k_y^2 & 1 \\ & & & 1 & -3-\triangle_x^2 k_y^2 \end{pmatrix} \begin{pmatrix} \widehat{\phi}_1 \\ \widehat{\phi}_2 \\ \vdots \\ \widehat{\phi}_{N-1} \\ \widehat{\phi}_N \end{pmatrix}.$$

$$= \triangle_x^2 \begin{pmatrix} \widehat{\omega}_1 \\ \widehat{\omega}_2 \\ \vdots \\ \widehat{\omega}_{N-1} \\ \widehat{\omega}_N \end{pmatrix} + \delta_{k_y} \triangle_x \begin{pmatrix} \widehat{\Phi}'_{1/2} \\ 0 \\ \vdots \\ 0 \\ \widehat{\Phi}'_{N+1/2} \end{pmatrix}. \tag{3.25}$$

Note that the boundary conditions only affect the zero-mode in Fourier space.

3.6 Time integration

We start with the simple case of a one-dimensional diffusion problem given by

$$\frac{\partial U}{\partial t} = \kappa \frac{\partial^2 U}{\partial x^2} \tag{3.26}$$

with some initial conditions $U(x, t = 0) = U^0(x)$ and boundary conditions at $-L_x/2$ and $L_x/2$. In case of the simplest possible implicit scheme by a forward difference in time we get

$$\frac{\partial U}{\partial t} = \frac{1}{\triangle_t}\left(u^i - u^{i-1}\right) + \mathcal{O}(\triangle_t) \tag{3.27}$$

with $\triangle_t$ as the numerical time step and results in the temporally discretisized diffusion equation

$$u^i = u^{i-1} + \kappa \triangle_t \frac{\partial^2 U^i}{\partial x^2} + \mathcal{O}(\triangle_t). \tag{3.28}$$

For second order spatial derivative, given by equation , this expression is written as

$$-r_x u^i_{n-1} + (1 + 2r_x) u^i_n - r_x u^i_{n+1} = u^{i-1}_n \tag{3.29}$$

with $r_x = \kappa\triangle_t/\triangle_x^2$. In matrix form this equation takes the form

$$\begin{pmatrix} \ddots & \ddots & \ddots & & \\ & -r_x & 1+2r_x & -r_x & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ u^i_{n-1} \\ u^i_n \\ u^i_{n+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ u^{i-1}_{n-1} \\ u^{i-1}_n \\ u^{i-1}_{n+1} \\ \vdots \end{pmatrix}, \tag{3.30}$$

which has to be solved for u^i at time step t_i given the solution u^{i-1} at the previous step t_{i-1}. Boundary conditions can be implemented analogous to equation 3.6 and equation 3.7 which leads to the diffusion equation in matrix form

$$\begin{pmatrix} 1+3r & -r & & & \\ -r & 1+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & 1+2r & -r \\ & & & -r & 1+3r \end{pmatrix} \begin{pmatrix} u^i_1 \\ u^i_2 \\ \vdots \\ u^i_{N-1} \\ u^i_N \end{pmatrix} = \begin{pmatrix} u^{i-1}_1 + 2rU_{1/2} \\ u^{i-1}_2 \\ \vdots \\ u^{i-1}_{N-1} \\ u^{i-1}_N + 2rU_{N+1/2} \end{pmatrix}, \tag{3.31}$$

for Dirichlet boundary conditions and

$$\begin{pmatrix} 1+r & -r & & & \\ -r & 1+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & 1+2r & -r \\ & & & -r & 1+r \end{pmatrix} \begin{pmatrix} u^n_1 \\ u^n_2 \\ \vdots \\ u^n_{N-1} \\ u^n_N \end{pmatrix} = \begin{pmatrix} u^{n-1}_1 + r\triangle_x U'_{1/2} \\ u^{n-1}_2 \\ \vdots \\ u^{n-1}_{N-1} \\ u^{n-1}_N + r\triangle_x U'_{N+1/2} \end{pmatrix} \tag{3.32}$$

for Neumann boundary conditions. The extension to the two dimensional diffusion problem treating the y-direction spectrally is analogous to the two dimensional elliptic equation discussed above and is therefore not described in detail again.

We now introduce the K-th order stiffly stable integration, presented in [26], which is used in the 2dads code. We generalize Equation 3.26 to the form

$$\frac{\partial U}{\partial t} = \kappa \frac{\partial^2 U}{\partial x^2} + \mathcal{L}U \tag{3.33}$$

where $\mathcal{L}$ stands for a differential operator acting on the variable U. For u as a discretisation of U, the stiffly stable time integration scheme is given by

$$\frac{1}{\triangle_t}\left(\alpha_0 u^i - \sum_{k=1}^{K} \alpha_k u^{i-k}\right) = \kappa\delta_x^2 u^i + \sum_{k=1}^{K} \beta_k \mathcal{L}u^{i-k} + \mathcal{O}(\triangle_t^K) \tag{3.34}$$

where the discretization of the second order spatial derivative is symbolised by the operator δ_x^2. Approximating the differential operator δ_x^2 by the second order central difference scheme, as given in equation 3.4, we reorder the equation above as

$$-r_x u_{n-1}^i + (\alpha_0 + 2r_x) u_n^i - r_x u_{n+1}^i = \sum_{k=1}^{K} \left(\alpha_k u_n^{i-k} + \triangle_t \beta_k \mathcal{L} u_n^{i-k} \right) + \mathcal{O}(\triangle_t^K + \triangle_x^2), \tag{3.35}$$

where $r_x = \kappa \triangle_t / \triangle_x^2$ and the coefficients α_k and β_k are determined by the order of the scheme K and presented in table 3.1. We now write this equation in matrix form again and obtain

$$\begin{pmatrix} \ddots & \ddots & \ddots & & \\ & -r_x & \alpha_0 + 2r_x & -r_x & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ u_n^i \\ \vdots \end{pmatrix} = \sum_{k=1}^{K} (\alpha_k + \beta_k \mathcal{L}) \begin{pmatrix} \vdots \\ u_n^{i-k} \\ \vdots \end{pmatrix}. \tag{3.36}$$

As we see in equation 3.35 the stiffly stable integration scheme treats diffusion implicitly and any other terms explicitly. For $K = 1$ we get the simplest case which is used in equation 3.28. Because of the higher accuracy of higher order integration schemes we use the third order integration scheme wherever possible. For the first two time steps we are obviously forced to use the first respectively second order integration scheme.

K	α_0	α_1	α_2	α_3	β_1	β_2	β_3
1	1	1	0	0	1	0	0
2	3/2	2	-1/2	0	2	-1	0
3	11/6	3	-3/2	1/3	3	-3	1

Table 3.1: Coefficients of the stiffly stable time integration for K-th order accuracy.

As in the previous scheme the first and last row of the matrix require special treatment because of the boundary conditions. In the case of Dirichlet boundary conditions we use equation 3.15a to get

$$(\alpha_0 + 3r_x)\, u_1^i - r_x u_2^i = \sum_{k=1}^{K} \left(\alpha_k u_1^{i-k} + \beta_k \mathcal{L} u_1^{i-k} \right) + 2r_x U_{1/2}, \tag{3.37a}$$

$$-r_x u_{N-1}^i + (\alpha_0 + 3r_x)\, u_N^i = \sum_{k=1}^{K} \left(\alpha_k u_N^{i-k} + \beta_k \mathcal{L} u_N^{i-k} \right) + 2r_x U_{N+1/2}. \tag{3.37b}$$

In case of Neumann boundary conditions, equation 3.17a leads to

$$(\alpha_0 + r_x)\, u_1^i - r_x u_2^i = \sum_{k=1}^{K} \left(\alpha_k u_1^{i-k} + \beta_k \mathcal{L} u_1^{i-k}\right) - \triangle_x r_x U'_{1/2}, \tag{3.38a}$$

$$-r_x u_{N-1}^i + (\alpha_0 + r_x)\, u_N^i = \sum_{k=1}^{K} \left(\alpha_k u_N^{i-k} + \beta_k \mathcal{L} u_N^{i-k}\right) + \triangle_x r_x U'_{N+1/2}. \tag{3.38b}$$

Note that the extension to the two dimensional problem treating the y-direction spectrally is analogous to the two dimensional elliptic equation discussed above [20].

3.7 Arakawa scheme

The non-linear advection terms are computed in configuration space, and are calculated using the Arakawa scheme [27]. The Arakawa scheme has the advantage that it conserves energy and enstrophy exactly. Particularly this scheme calculates the advection terms via

$$\begin{aligned}
\{\omega, \phi\} = -\frac{1}{12\triangle_x\triangle_y}\Big[&\left(\phi_{i,j-1} + \phi_{i+1,j-1} - \phi_{i,j+1} - \phi_{i+1,j+1}\right)\left(\omega_{i+1,j} + \omega_{i,j}\right) \\
&- \left(\phi_{i-1,j-1} + \phi_{i,j-1} - \phi_{i-1,j+1} - \phi_{i,j+1}\right)\left(\omega_{i,j} + \omega_{i-1,j}\right) \\
&+ \left(\phi_{i+1,j} + \phi_{i+1,j+1} - \phi_{i-1,j} - \phi_{i-1,j+1}\right)\left(\omega_{i,j+1} + \omega_{i,j}\right) \\
&- \left(\phi_{i+1,j-1} + \phi_{i+1,j} - \phi_{i-1,j-1} - \phi_{i-1,j}\right)\left(\omega_{i,j} + \omega_{i,j-1}\right) \\
&+ \left(\phi_{i+1,j} - \phi_{i,j+1}\right)\left(\omega_{i+1,j+1} + \omega_{i,j}\right) \\
&- \left(\phi_{i,j-1} - \phi_{i-1,j}\right)\left(\omega_{i,j} + \omega_{i-1,j-1}\right) \\
&+ \left(\phi_{i,j+1} - \phi_{i-1,j}\right)\left(\omega_{i-1,j+1} + \omega_{i,j}\right) \\
&- \left(\phi_{i+1,j} - \phi_{i,j-1}\right)\left(\omega_{i,j} + \omega_{i+1,j-1}\right)\Big].
\end{aligned} \tag{3.39}$$

Note that the error of the Arakawa scheme is $\mathcal{O}(\triangle_x^4)$ and $\mathcal{O}(\triangle_y^4)$, two orders of magnitude smaller than the error of the finite differences schemes used to discretize spatial derivatives (equation 3.6). We have to evaluate this expression for each grid point inside the physical domain. As we see this is dependent on the nearest neighbours in x and y direction which demonstrates the usefulness of ghost points in the y-direction in the numerical simulation.

3.8 Matrix factorisations

Solving the elliptic equation and the time integration require a matrix factorisation to solve the equation system. The factorisation itself is solved by the cuSOLVER library. The cuSOLVER library is a package that provides common numerical linear algebra routines, such as matrix factorization and triangular solver routines for dense matrices, sparse least-squares solvers and eigenvalue solvers for GPUs [22].

In this thesis we employed the QR solver in the 2dads code to solve the equation systems in matrix form described in the previous sections such as 3.25 and 3.36. A QR-decomposition is a composition of a Matrix **A** into

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R}, \tag{3.40}$$

where **R** is a upper triangular matrix and **Q** an orthogonal matrix, that is $\mathbf{Q}^\top \cdot \mathbf{Q} = \mathbf{1}$, where $\mathbf{Q}^\top$ is the transpose matrix of **Q**. QR-decompositions can be used to solve systems of linear equations like

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}. \tag{3.41}$$

This can be done by forming $\mathbf{Q}^\top \cdot \mathbf{b}$ and then solving

$$\mathbf{R} \cdot \mathbf{x} = \mathbf{Q}^\top \cdot \mathbf{b} \tag{3.42}$$

by back substitution [28]. The QR-decomposition is the only solver included in the cuSOLVER library that provides a batch size bigger than one, which means that several equation systems can be executed in parallel.

Both the elliptic equation 3.25 and time integration 3.36 have to be solved for each time step. These matrices can be pre-factorized and stored at the start of the numerical simulation and reused at each time step. We therefore pre-calculate the matrix factorizations at the start of each numerical simulation.

We now take a look at the convergence rate at the example of

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) f = \Omega \tag{3.43}$$

where Ω is the second derivative in x- and y-direction of a gaussian function given by $f(x, y) = \exp\left(-(x^2 + y^2)/2\right)$ which takes the form

$$\Omega(x, y) = \exp\left(-(x^2 + y^2)/2\right)\left(x^2 + y^2 - 2\right) \tag{3.44}$$

with $x, y \in (-10, 10)$. The numerical solution f of the QR-factorisation is compared to the analytical one. We calculated the error at each grid point by $|f_{num} - f_{an}|$ and determine the average value which is plotted against the resolution of the simulation domain in Figure 3.7. As we expect the error decreases quadratically.

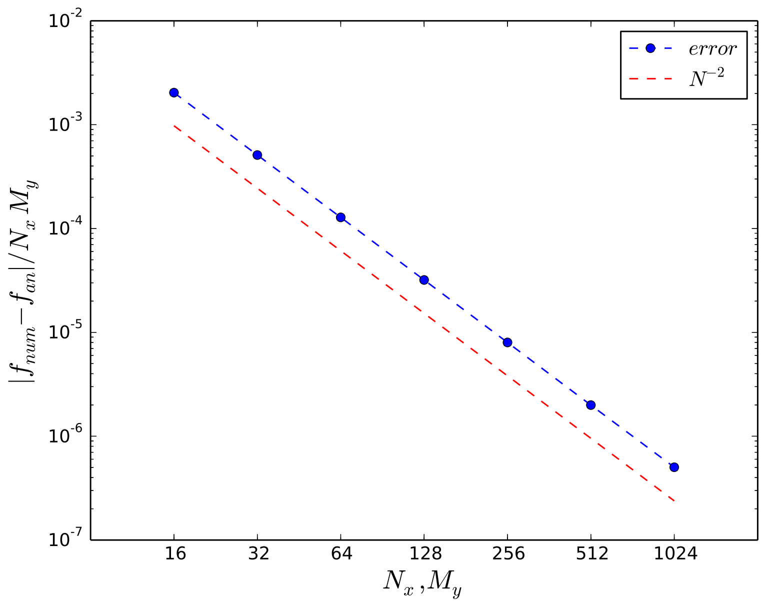


Figure 3.7: Convergence rate of QR-factorisation at the example of a second order derivative in x- and y-direction of a Gaussian function.

3.9 Computational complexity

In order to minimize the runtime of the numerical simulation it is necessary to consider the computational complexity of the numerical algorithms.

Finite differences algorithms have a computational complexity of $\mathcal{O}(n)$ and the spectral method, using fast fourier transformations, is of order $\mathcal{O}(n \ln n)$. This complexity makes this method suitable for numerical simulations considering its high accuracy for derivatives.

The highest computation complexity belongs to the matrix factorisation. The QR-decomposition implemented in this code is of complexity $\mathcal{O}(n^3)$ and the most computationally intensive algorithm of this simulation. This fact can easily be confirmed by taking a look at the runtime of the functions of the different algorithms. As an example we chose the model described in chapter 5.3 with $N_x = N_y = 1024$. The runtime measurements have been performed using the NVIDIA Visual Profiler [29].

The measurements show that the QR-decomposition demands around 90 % of the total runtime of the simulation. To achieve a significantly better performance of the code it therefore would be necessary to further optimize the QR-decomposition. If we consider that we used the cuSOLVER library to performe the decomposition, it would be substantially more time consuming to implement this method by our own, without the guaranty that our implementation would run faster than the cuSOLVER routine. We therefore consider ourself satisfied with the code performance.

4 Parallel Computing

In this chapter we give an overview on parallel computing and programming on graphical processing units.

Over the last decades central processing units (CPU) increased rapidly in performance, as shown in figure 4.1, while their manufacturing costs steadily decreased. This increase was obtained by increasing the clock speed of the processors. These improvements made most software developers rely on the advances in hardware to increase the speed of their application. The same piece of software ran faster on each new generation of processing units. This drive of improvement has slowed down in recent years. Especially energy consumption and heat dissipation issues create an insurmountable wall for the clock rate of new CPUs as frequencies beyond 5 GHz lead to melting microprocessors [30].

However, the number of transistors on an integrated circuit (IC) still doubles every 18 -24 months as shown in Figure 4.1. To keep up with previous advances in performance, IC manufacturers started exploiting parallel architectures. This allows single processors to maintain an increase in theoretical computing power with each new generation while not exceeding the clock rate limit of 5 GHz. In order to utilize these new types of processors efficiently it is necessary to implement algorithms that can be executed in parallel [31].

In the following section we will introduce the concepts of parallel computation and the architectures of graphical processing units (GPU).

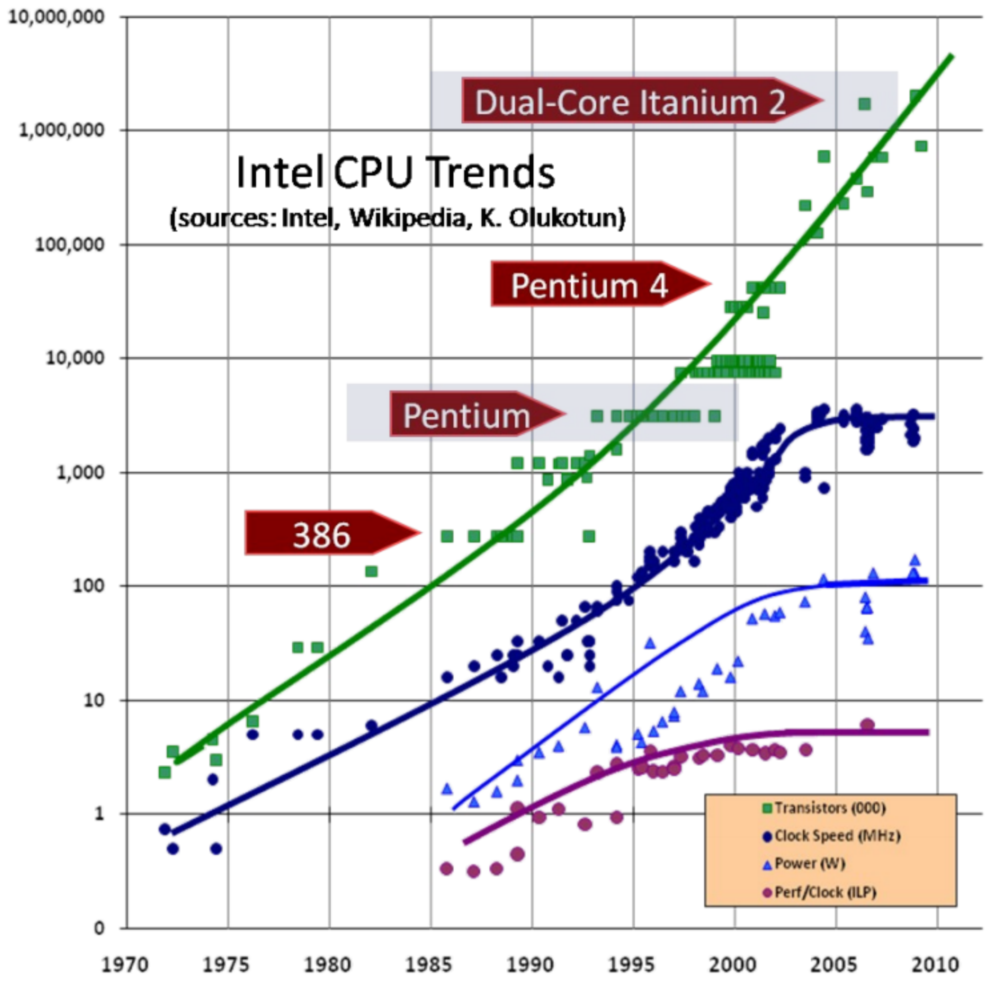


Figure 4.1: Performance increase at the example of Intel CPUs [31].

4.1 Parallel programming concepts

A sequential program may be split up into several processes (instances of a computer program that are being executed) which can be executed simultaneously if certain conditions are fulfilled. This set of conditions was established by Bernstein in 1966 and relates to memory locations used by the processes to hold variables that are read and altered during the execution of the processes. Let us define I_i as the set of input and O_i as the set of output memory locations of the process P_i. For two processes P_1 and P_2 the Bernstein Conditions read

$$I_1 \cap O_2 = \emptyset \tag{4.1}$$
$$I_2 \cap O_1 = \emptyset \tag{4.2}$$
$$O_1 \cap O_2 = \emptyset \tag{4.3}$$

where $\emptyset$ is the empty set. If those three conditions are satisfied the two processes can be executed in parallel. If one of the first two conditions is violated the input memory location of one process may be overwritten by another process. This case leads to non-deterministic results because the runtime of the processes might vary between different executions of the program. In case of violation of the third condition several processes write to the same memory location. Here, obviously only the output of the last process will be saved and the output of the other processes is lost.

Maybe the most important question when developing solutions for parallel architectures is how much faster a parallel method solves the problem compared to a sequential method. This relative performance is described by the Speedup Factor. For a program running on p processors it is defined as

$$S(p) = \frac{t_s}{t_p}, \tag{4.4}$$

where t_s is the execution time of the sequential algorithm running on a single processor and t_p is the execution time of the parallel algorithm on a multiprocessor. The maximum speedup is p and is achieved when the computation can be divided into p equal-duration parts assuming no additional overhead in the parallel algorithm.

Usually several factors will appear as overhead in the parallel method that decrease the speedup. This is the case if the computation task is not dividable into p equally sized parts and if there is extra communication in the parallel algorithm that does not appear in the sequential version. Keeping that in mind we assume that a fraction of the the computation f cannot be parallelized. In this case the time to perform the computation on p processes is given by

$f\,t_s + (1-f)t_s/p$. The speedup factor is then given by

$$S(p) = \frac{t_s}{ft_s + (1-f)t_s/p} = \frac{p}{1+(p-1)f}, \tag{4.5}$$

which is also known as Amdahl's law. The speedup is therefore limited to $\lim\limits_{p\to\infty} S(p) = \frac{1}{f}$.

Another useful quantity is the efficiency of the parallel solution, defined by

$$E = \frac{t_s}{t_p \times p}. \tag{4.6}$$

This quantifies how long processes are being used on the computation and to which extent it is reasonable to increase the number of processors p. Obviously the efficiency equals 1 in case of linear speedup.

A classification for computers created by Michael Flynn in 1966 is based on the two dimensions of Instruction and Data. Each of the dimensions has two possible states: Single and Multiple. This results in four possible classes:

- **Single Instruction, Single Data (SISD):** This corresponds to a classical serial computer. During one clock cycle one instruction is executed by the CPU on one data stream. This is the oldest and most common type of a single core computer.

- **Single Instruction Multiple Data (SIMD):** This is a type of a parallel computer. Each processing unit executes the same instruction on a different data element. Graphical Processing Units (GPU) are a SIMD architecture and will be discussed in detail later. Other examples would be MMX and SSE extensions of pentium CPUs.

- **Multiple Instruction Single Data (MISD):** This class corresponds to computers using different introduction streams on the same data stream. There are only few actual examples such as multiple frequency filters or multiple cryptography algorithms.

- **Multiple Instructions Multiple Data (MIMD):** This describes the most common type of parallel computers. Each processor has its own instruction and data stream. Most current multi-core computers, supercomputers and clusters use this class [32].

In the following I will discuss graphical processing units more detailed. The solvers discussed in Chapter **Numerical Methods** are implemented on this architecture.

4.2 Graphical processing units

Since the performance increase of classical single core processors slowed down in 2003, processor vendors have settled on two different designs of microprocessors [33]. Firstly, multicore CPUs, typically using two to a few tens of cores, are designed to maintain the highest possible execution speed of sequential programs. In addition these processing units are designed to perform well on a broad variety of tasks. Architectural advances are for instance branch prediction and large caches. In contrast to this, the many-core trajectory, typically using hundreds of cores, follows a fundamentally different strategy. These multiprocessors are SIMD computers and are build specially for applications that present a large degree of data parallelism [34]. Another important issue is the memory bandwidth. GPUs bandwith is approximately 5 to 10 times the bandwidth of CPU chips. The reason for this is that GPUs feaure simpler memory models and are bound by fewer legacy constrains [30]. This makes GPUs suitable for numerical methods such as the Fast Fourier Transformations or the matrix factorization as described in the previous chapter.

While the performance of general-purpose CPUs increased slowly, GPUs improved relentlessly as shown in Figure 4.2. The reason for this large gap in theoretical GFLOPs between many-core GPUs and multicore CPUs lies in the design philosophies discussed above. Both processor architectures are illustrated in figure 4.3.

Nevertheless, one has to consider claims about GPUs delivering substantial speedups of up to 1000X over multi-core CPUs [36, 37, 38] have to be treated with caution. CPUs still provide numerous possibilities for optimization such as multithreading or cache blocking. The comparison of fully optimized code for CPUs and GPUs still delivers a 5X to 10X better performance of the GPUs for algorithms that are relevant for simulations as described in the previous chapter [34].

4.3 Programming in CUDA

In the following we present the basic concepts of programming on graphical processing units. As an application programming interface (API) for the GPU we use CUDA which is created by NVIDIA. CUDA is an acronym for Computer Unified Device Architecture and provides an extention to C [30].

As an example of how to use this API we compare the implementation of the finite difference scheme in chapter 3.6 for a serial computer and using CUDA. An implementation in conventional C could look like the following.

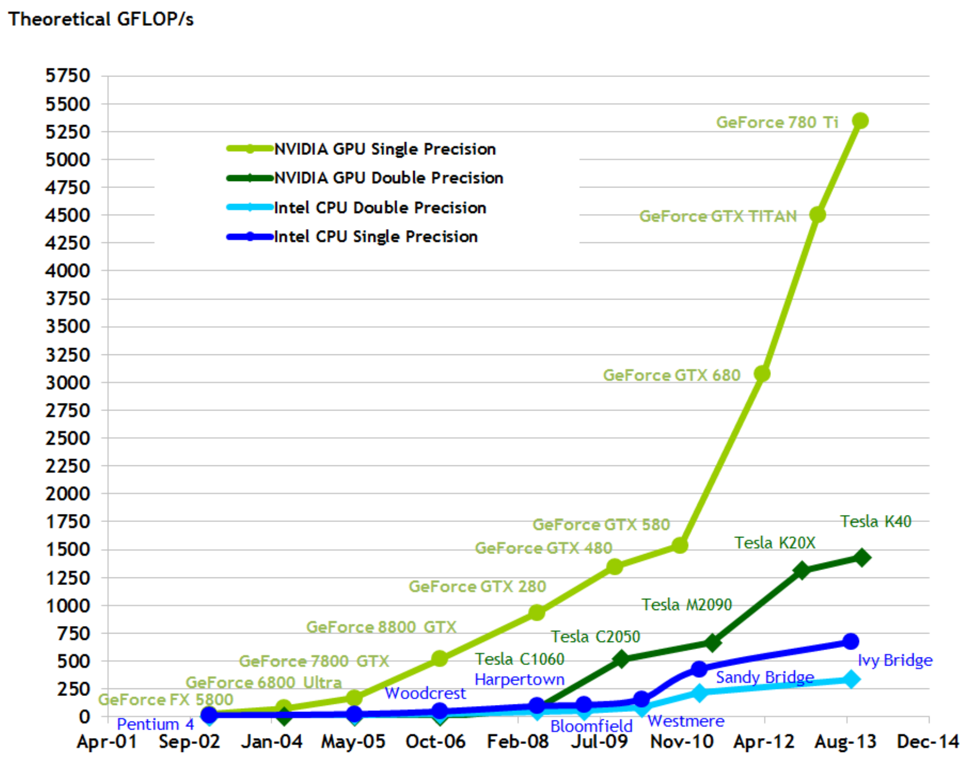


Figure 4.2: Comparison of the increase of floating-point operations per second for CPUs and GPUs [35].

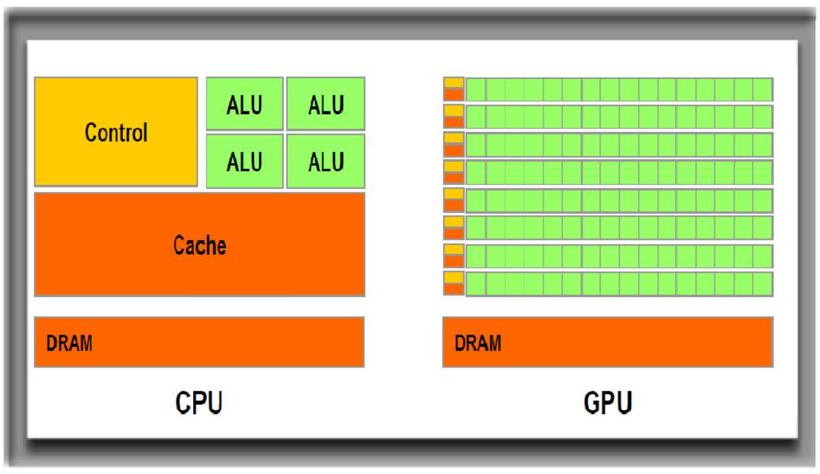


Figure 4.3: Illustration of the processor architectures of conventional CPUs and GPUs [35].

```
 1 #include<stdio.h>
 2
 3 void fd_CPU(double* in, double* out,
 4   int Nx, double delta_x);
 5
 6 double f(double x, double Lx);
 7
 8 int main(){
 9   const int Nx = 128;
10   const double Lx = 42;
11   double data[Nx];
12   double CPU_out[Nx-2];
13
14   double delta_x = Lx/Nx;
15
16   // initialize array
17   for(int i = 0; i < Nx; i++){
18     data[i] = f(i*delta_x, double Lx);
19   }
20
21   fd_CPU(data, CPU_out, Nx, delta_x);
22
23   return 0;
24 }
25
26 double f(double x, double Lx){
27   // assign function value
28
29     return value;
30 }
31
32 void fd_CPU(double* in, double* out,
33   int Nx, double delta_x){
34
35   for(int i = 1; i< Nx -1; i++){
36     out[i] = (in[i+1] - in[i-1])/(2*delta_x);
37   }
38 }
```

For simplicity we ignore boundary conditions. In this code we initialize the input for the computation in CPU memory in line 16 and calculate the first derivative by the function `finite_differences`, defined in line 24. The result of the computation is stored in CPU memory, pointed to by `CPU_out`.

To implement this finite difference scheme in CUDA several additional steps are necessary. One possible implementation could look like:

```
#include<stdio.h>
#include<cuda_runtime_api.h>

__global__
void fd_GPU(double* in, double* out, int Nx,
double delta_x);


  int main(){

  const int Nx = 128;
  const double Lx = 42;
  double data[Nx];
  //double CPU_out[Nx-2];
  double delta_x = Lx/Nx;

  double* d_Lx, d_data, GPU_out, d_delta_x;
  int* d_Nx;

  // initialize array
  for(int i = 0; i < Nx; i++){
    data[i] = f(i*delta_x);
  }

  cudaMalloc( (void**)&d_Lx, sizeof(double) );
  cudaMalloc( (void**)&d_data, sizeof(double)*Nx );
  cudaMalloc( (void**)&d_delta_x, sizeof(double) );
  cudaMalloc( (void**)&GPU_out, sizeof(double)*(Nx-2) );

  cudaMemcpy( d_Lx, Lx, sizeof(double),
  cudaMemcpyHostToDevice );
  cudaMemcpy( d_delta_x, delta_x, sizeof(double),
  cudaMemcpyHostToDevice );
  cudaMemcpy( d_data, data, sizeof(double)*Nx,
  cudaMemcpyHostToDevice );

  fd_GPU<<<dimGrid,dimBlock>>>(d_data, GPU_out,
      d_Nx, d_delta_x);

  cudaMemcpy( out, GPU_out, sizeof(double)*(Nx-2),
        cudaMemcpyDeviceToHost );
36
37      cudaFree(d_data);
38      cudaFree(GPU_out);
39      cudaFree(d_Lx);
40      cudaFree(d_delta_x);
41
42       return 0;
43    }
44
45    __global__
46    void fd_GPU(double* in, double* out, int Nx,
      double delta_x)
47    {
48      int index = blockIdx.x*blockDim.x + threadIdx.x;
49
50      if(index < 0 && index < Nx){
51        out[index] = (in[index+1] - in[index-1])
                       /(2*delta_x);
52      }
53    }
54
```

As we see this code contains a number of additional function calls compared to the previous example. The code can be divided into four parts:

1 initializing input data on the host (CPU)

2 copying data from host memory to device memory (GPU)

3 performing calculations on the device

4 copying results back to host memory

In the following we explain each step in further detail at the example above.

1. As in the previous example we initialize the input for the computation in CPU memory in line 19.

2. The first difference to the previous code is the function call `cudaMalloc` in line 22. `cudaMalloc` allocates memory on the GPU analogous to `malloc`. In line 27 we copy data from the CPU to the GPU using the function `cudaMemcpy`. The keyword `cudaMemcpyHostToDevice` specifies the direction of the copy.

3. In CUDA a function that is executed on the GPU is called kernel. The definition of a kernel requires an additional specifier, indicated in front of the name of the function, which declares where this function can be called from. In line 42 `__global__` defines a kernel that is called from host code. We call the kernel in line 31 where we specify the execution configuration between the triple chevrons. This defines how many instances of execution, so called threads, are executed in parallel and how they are structured. Threads are organized in blocks and blocks are organized in a grid. The logical separation between thread blocks and grids is a logical abstraction of the memory hierarchy implementd in GPU hardware. In the example the number of blocks is represented by `dimGrid` which we did not declare in the code for simplicity. The number of threads in one block is given by `dimBlock`. When a kernel is executed, each worker thread is assigned to a `bloxkIdx` and `threadIdx` and uses those to find the data element to operate on.

4. After the kernel is executed we copy the result back to host memory using the function `cudaMemcpy` (line 32). Finally we free the memory allocated at the device by `cudaFree`, analogous to `free`, in line 34 to 37.

This code is a typical example of SIMD computation. All threads have the same instruction stream but operate on different array elements.

4.4 Speedup of 2dads code

In this section we compare the runtime of our implementation using CUDA to a sequential Fortran code that uses mostly the same methods to solve model equations of the same shape [20].

We therefore measure the runtime of the two programs for different resolutions, shown in figure 4.4. We use a rectangular grid with 32^2 to 1024^2 grid points and solve a perform a simple blob simulation as described in chapter 5.3. For each program we perform 1000 time iterations and note the average run time of 5 program executions. We use a GeForce GTX 680 as the GPU and a Intel Xeon W3550 as a CPU. As we see the performance of the two different codes strongly depends on the number of grid points. For very low resolutions the sequential Fortran code is significantly faster than the parallelized CUDA implementation. With increasing N_x, M_y the CUDA code increases in performance and pass the Fortran version between $N_x = M_y = 64$ and $N_x = M_y = 128$. For bigger resolution the parallelization of the CUDA code leads to a significant runtime difference, according to equation 4.4 we gain a Speedup Factor of

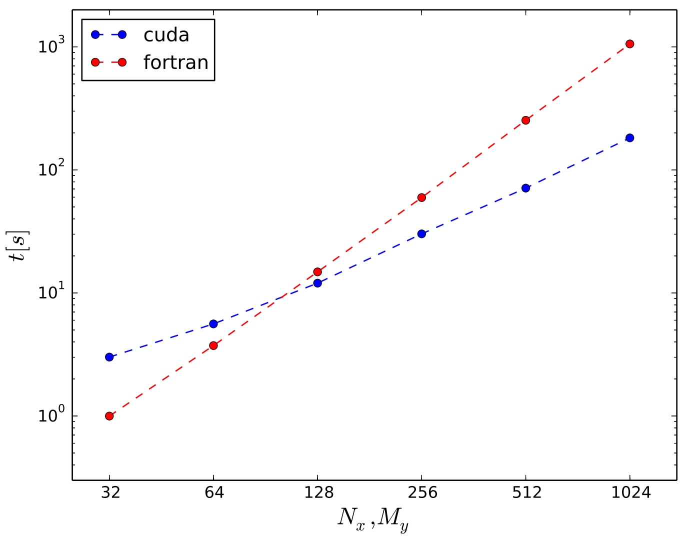


Figure 4.4: Runtime t of the CUDA/Fortran code for different numbers of grid points. The runtime is measured for 1000 time iterations and averaged over 5 executions.

$S(p) \approx 6$ for $N_x = M_y = 1024$.

As we see the CUDA implementation only pays off for large simulations. Only above $N_x = M_y = 128$ we gain significant speedup with the parallelized GPU version.

The 2dads code, which is stil steadily enhanced, is available at [39].

5 Code Testing

5.1 Blob diffusion

To verify the implementations presented in Chapter 3 and the frame work in which they are implemented we benchmark the code on simplified models. We start comparing the diffusion of a symmetric blob to its analytical solution. The blob structure is initialized with its peak in the center of the coordinate system of the simulation domain in the form

$$n(\mathbf{x}, t = 0) = \exp\left(-\frac{1}{2}\mathbf{x}^2\right). \tag{5.1}$$

The diffusion equation is given by

$$\frac{\partial n}{\partial t} = \kappa \nabla_\perp^2 n \tag{5.2}$$

where κ represents the diffusion coefficient and n is the particle density. We perform a numerical simulation over 1000 time steps with a time step of $\Delta t = 10^{-2}$ and a diffusion coefficient of $\kappa = 0.5$ on a quadratic simulation domain where $\mathbf{x} \in (-25, 25)$. The analytic solution of the diffusion equation 5.2 with the initial conditions 5.1 takes the form [12]

$$n(\mathbf{x}, t) = \frac{1}{1 + 2\kappa t}\exp\left[-\frac{\mathbf{x}^2}{2(1 + 2\kappa t)}\right]. \tag{5.3}$$

We now compare the analytic solution to the result from the numerical simulation. As an example we show the case of a resolution of 128×128 grid points

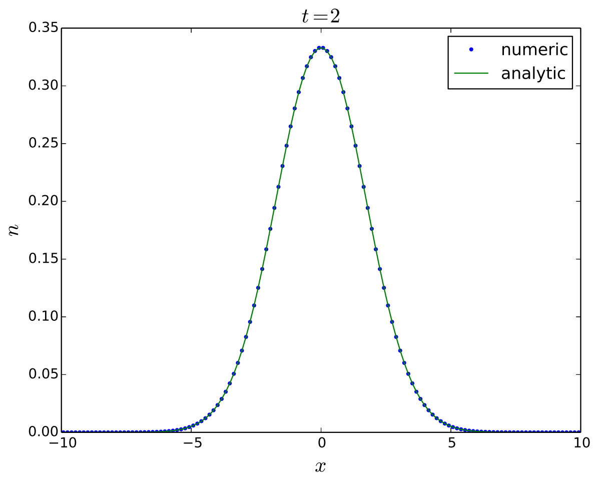


Figure 5.1: Comparison of the numerical result (blue points) of the diffusion equation 5.2 with starting conditions 5.1 and the analytic solution 5.3 (green line).

and the same parameters as mentioned above in figure 5.1. The average error at each grid point should decrease with the resolution of the simulation. To verify this we measure the average error at the end of the simulation, given by $|f_{num} - f_{an}|/N_x M_y$ for different resolutions. The results are shown in figure 5.2. We find that the error decreases almost quadratically with $(N_x \times M_y)^2$. For $N_x, M_y = 1024$ we see that the converges rate flattens out due to the error of the time derivative witch does not change for different resolutions.

5.2 Non-linear advection

We further test the code with a more complex model where we include a non-linear advection term. This model is given by

$$\frac{\partial n}{\partial t} + \{\phi, n\} = \kappa \nabla_\perp^2 n \tag{5.4}$$

where n stands for the plasma density and ϕ is the electrostatic potential which we chose time independent and of the form

$$\phi(x, y) = \phi_{21}\, sin\left(\frac{y}{L_y}\right) sin\left(\frac{x}{2\,L_x}\right) \tag{5.5}$$

with box length L_x and L_y and the amplitude $\phi_{21} = 1$. As an initial condition for the density we chose a linearly decreasing density profile in x-direction, $n(x, t = 0) = (L_x - x)/L_x$. We chose Dirichlet boundary conditions on both boundaries in x-direction with $U_{1/2} = 1$ and $U_{N+1/2} = 0$, a diffusion coefficient

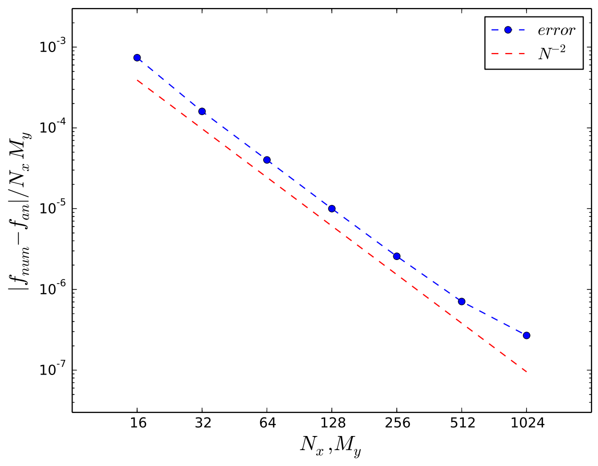


Figure 5.2: The average error of the simulation of the diffusion equation 5.2 as a function of discretization points. This is compared to the expected rate of convergence for a second order scheme.

of $\kappa = 0.5$ and $L_x = L_y = 1$.

The chosen potential gives rise to a electric drift of the form $\mathbf{v}_E = \widehat{\mathbf{z}} \times \nabla\phi$. We therefore expect the plasma to drift radially outwards in the middle part of the simulation domain and inwards at the borders.

The simulation results are shown in figure 5.3 for $t = 0$, $t = 0.01$, $t = 0.05$ and $t = 20$. We take a closer look at the vertical profile n_0 of the plasma density given by

$$\langle n \rangle = n_0\,(x, t) = \frac{1}{L_y} \int_0^{L_y} dy\, n\,(x, y, t)\,. \tag{5.6}$$

The vertical profile is shown in figure 5.4 for $t = 0$, $t = 0.05$, $t = 0.1$ and $t = 2$. As we expect the density profile flattens out as the simulation proceeds. Note that the flattening is symmetric in x.

5.3 Simple blob simulation

For the last test we chose a minimal model for interchange motions. This model is essentially a closure of the plasma vorticity equation, coupled with an advection-diffusion equation for the plasma density n. The nondimensional

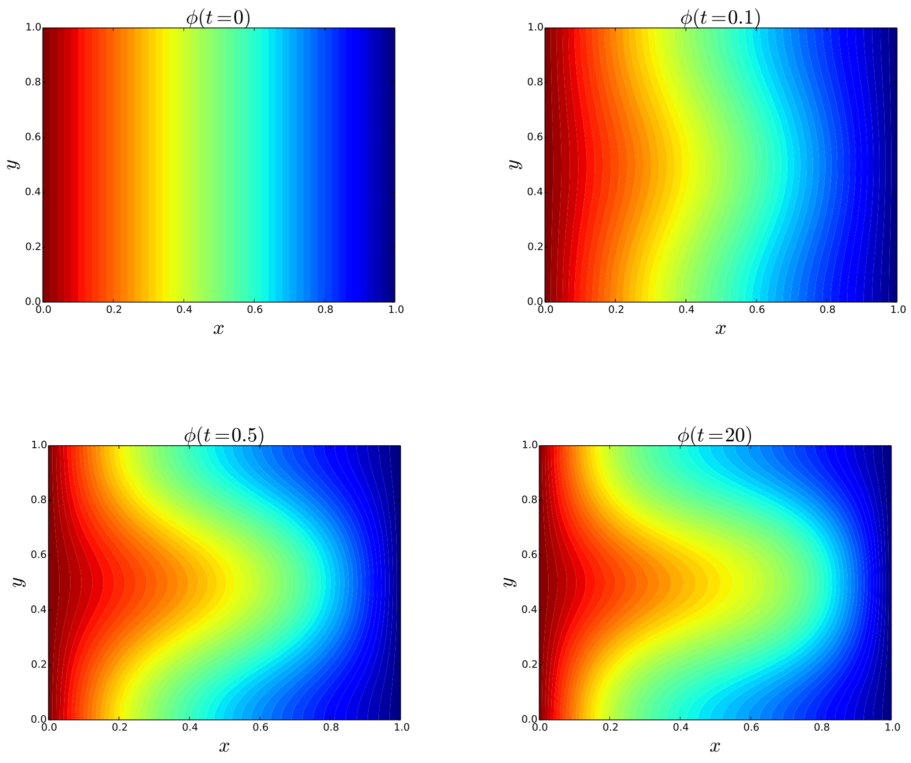


Figure 5.3: Evolution of the plasma density according to the model 5.4 with a diffusion coefficient $\kappa = 0.5$.

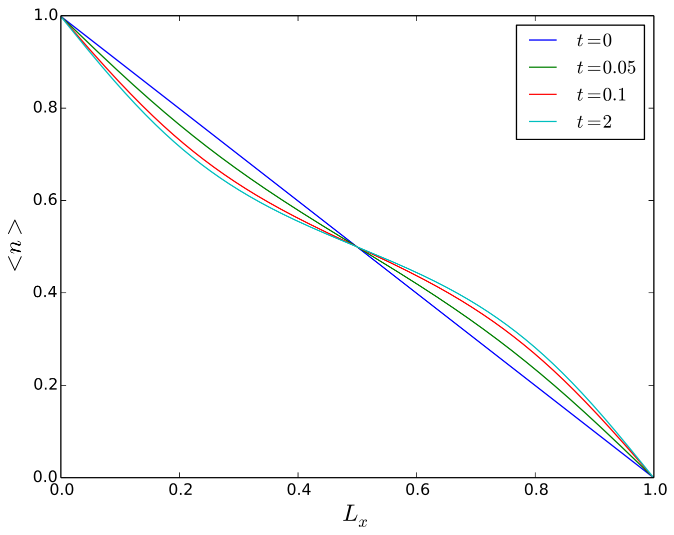


Figure 5.4: Vertical profile of the plasma density for different t.

model equations may be written as

$$\begin{aligned}\frac{\partial n}{\partial t} + \{\phi, n\} &= \kappa \nabla_\perp^2 n \\ \frac{\partial \Omega}{\partial t} + \{\phi, \Omega\} + \frac{\partial n}{\partial y} &= \mu \nabla_\perp^2 \Omega \\ \nabla_\perp^2 \phi &= \Omega.\end{aligned} \tag{5.7}$$

We chose a quadratic simulation domain with $L_x = L_y = 30$ with a resolution $M_x = N_y = 1024$. As Boundary conditions we chose Dirichlet conditions in x-direction with $U_{1/2} = U_{N+1/2} = 0$. for the diffusion coefficients we chose $\kappa = \mu = 0.0158$. The simulation is initialized with a symmetric blob structure for the plasma density centered at $\mathbf{x}_0 = (5, 35)$,

$$n(\mathbf{x}, t = 0) = \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^2\right]. \tag{5.8}$$

The starting conditions of potential ϕ and vorticity Ω are $\phi(t = 0) = \Omega(t = 0) = 0$.

At the beginning of the simulation there is no flow field, which so follows from the polarization of vorticity by the blob structure given by the $\partial n / \partial y$-term in equation 5.7. The blob structure is advected radially outwards, which corresponds the x-direction in the domain. At $t = 5$ a density peak with a steeper gradient at the front has developed, as vorticity is created through the $\partial n / \partial y$-term. The flow field is now dipolar and transports the plasma radially outwards at the position of the blob center. Subsequently, the plasma is circulated into the lobes by the dipolar flow field as it is transported along the equipotential lines. The advection of the blob structure is shown in figure 5.6. The radial cross section of the density field is shown in more detail in figure 5.5. As we see the the steep front is followed by a trailing wake while the amplitude decays with increasing t.

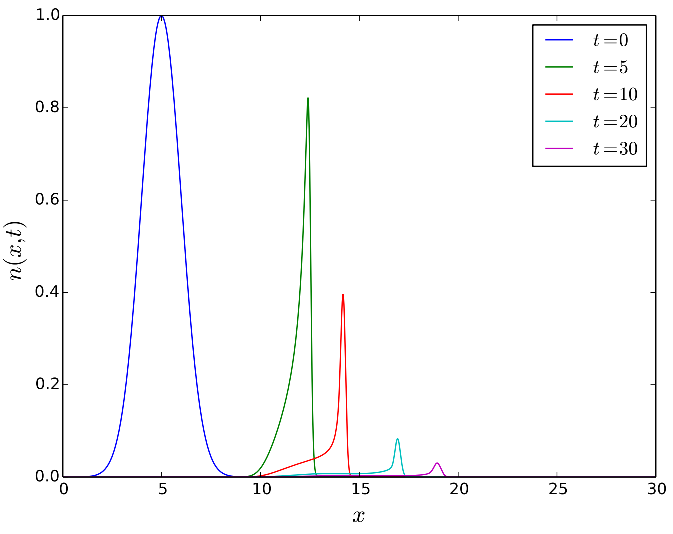


Figure 5.5: Radial variation of the plasma density for different t.

We observe excellent agreement with previous studies [10] which gives us reason to believe that the code solves the model equations 5.7 correctly.

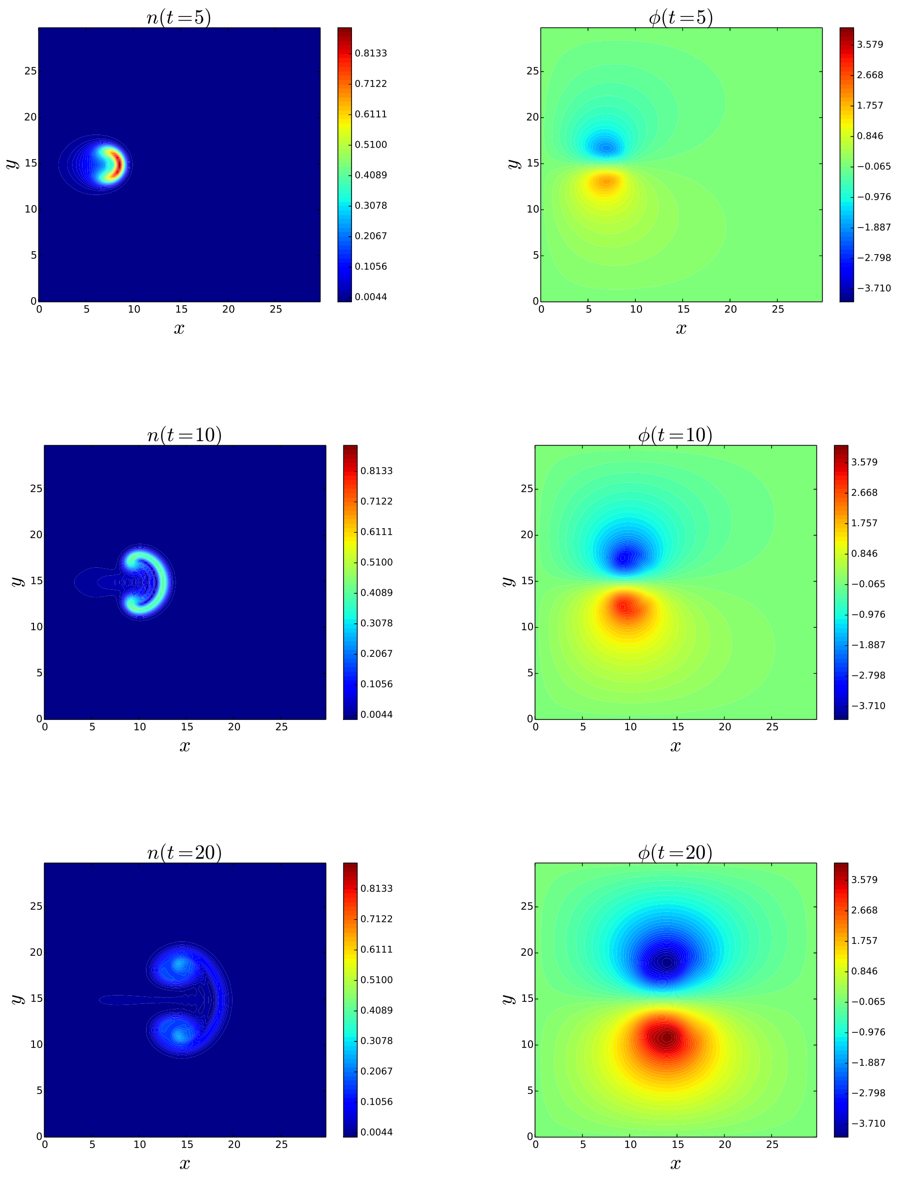


Figure 5.6: Radial advection of a localized blob structure initialized at $\mathbf{x}_0 = (5, 35)$ showing the plasma density n in the left column and the potential ϕ in the right column for different t.

6 Transport in flux-driven convection models

In this chapter we investigate the transport and confinement of a system driven by a constant incoming heat flux at the inner radial boundary. We identify different transport states, investigate the kinetic energy, the convective heat flux and integrated pressure for various model parameters. In addition the results are compared to previous studies [40].

Magnetized plasmas are usually driven far from thermodynamic equilibrium by forced quasi-stationary fluxes of particles and energy. These fluxes frequently lead to linear instabilities and fluctuation-induced transport which causes a degradation of the particle and heat confinement in the plasma. In addition quasi-two-dimensional fluctuations frequently lead to the generation of sheared flows, that reduce the particle and heat transport and therefore improve the plasma confinement. Therefore we investigate the basic physics of such a flux driven convection-shear flow system. The instability mechanisms described by this model are the most important mechanism of such diverse plasma systems as the edge and scrape-off layer in magnetic confinement experiments [41, 42].

The model comprises only the pressure and vorticity equations and is therefore the simplest possible self-cosistent model describing cross-phase and coherence between those fields [43]. We assume that the system is driven by a constant incoming heat flux Δ_p/L_x at the inner radial boundary, where L_x represents the radial width of the plasma layer and Δ_p the associated radial pressure variation. We further assume slab coordinates which leads to the two-dimensional convection model

$$\begin{aligned} \frac{\partial p}{\partial t} + \{\phi, p\} &= \kappa \nabla_\perp^2 p, \\ \frac{\partial \Omega}{\partial t} + \{\phi, \Omega\} + \frac{\partial p}{\partial y} &= \mu \nabla_\perp^2 \Omega, \end{aligned} \tag{6.1}$$

where ϕ stands for the electrostatic potential, p for the plasma pressure and Ω for the vorticity. The relation between ϕ and Ω is given by $\Omega = \nabla_\perp^2 \phi$. The model is normalized the way described in the chapter **Model Equations**. In the following we take the pressure and vorticity diffusion coefficients to be equal, $\kappa = \mu$. We use a Neumann condition for the left boundary of the domain for the plasma pressure of the form $\partial_x p = -1$ in non-dimensional units. For the right boundary we use Dirichlet conditions with $p = 0$. In the poloidal direction we chose periodic boundary conditions. The conditions for the potential and the vorticity are chosen $\phi = \Omega = 0$ at $x = 0, 1$.

Before we present the simulation results we define the physical properties that are examined. We define the poloidal average of any field as its profile, which we indicate with a zero sub-script. The pressure profile for instance takes the form

$$p_0(x,t) = \frac{1}{L_y} \int_0^{L_y} \mathrm{d}y \, p(x,y,t) \tag{6.2}$$

Where L_y is the domain size in the poloidal direction. Furthermore we define the spatial fluctuation by subtracting the poloidal average from the plasma pressure and donate this by an over-tilde $\widetilde{p} = p - p_0$. We use this to define the radial velocity by $\widetilde{v}_x = -\partial\widetilde{\phi}/\partial y$. Furthermore we define the radial convective heat flux profile by

$$\Gamma_0(x,t) = \frac{1}{L_y} \int_0^{L_y} \mathrm{d}y \, \widetilde{p}\,\widetilde{v}_x. \tag{6.3}$$

The energy integral of the mean and fluctuating motions are given by

$$\mathcal{U} = \int_0^{L_x} \mathrm{d}x \int_0^{L_y} \mathrm{d}y \frac{1}{2} v_0^2, \tag{6.4}$$

and

$$\mathcal{K} = \int_0^{L_x} \mathrm{d}x \int_0^{L_y} \mathrm{d}y \frac{1}{2} \left(\nabla_\perp \widetilde{\phi}\right)^2. \tag{6.5}$$

We further define the total thermal energy confined in the plasma layer by

$$\mathcal{E} = \int_0^{L_x} \mathrm{d}x \int_0^{L_y} \mathrm{d}y \, p. \tag{6.6}$$

In the following we use a quadratic domain with $L_x = L_y = 1$ and initialize the simulations with a convective cell for the electrostatic potential of the form

$$\phi\left(x, y, t = 0\right) = \phi_{12} \sin\left(\pi x\right) \sin\left(\frac{2\pi y}{L_y}\right) \tag{6.7}$$

with an amplitude of $\phi_{12} = 10^{-5}$. The initial diffusive pressure profile takes the form $p(x, y, t = 0) = 1 - x$.

In the following we will investigate the described system for different diffusion coefficients κ and μ and discuss the results.

6.1 Stationary convection

Firstly we investigate the model parameters slightly above the critical diffusion coefficient for the onset of convective motions, which is approximately $\kappa = \mu = 0.0183$. The initial convection cells are unstable and grow exponentially in time until they reach a steady state at a time of approximately $t \approx 750$. This can be seen in the evolution of the energy integral of the fluctuating motions, shown in figure 6.1.

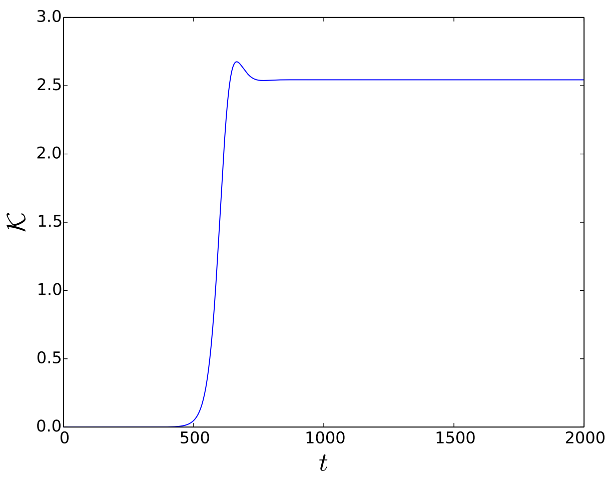


Figure 6.1: Evolution of the kinetic energy in fluctuating motions measured at (x,y) = (0.5,0.5) for the diffusion coefficients $\kappa = \mu = 0.0183$. The fluctuation amplitude saturates at $t \approx 750$.

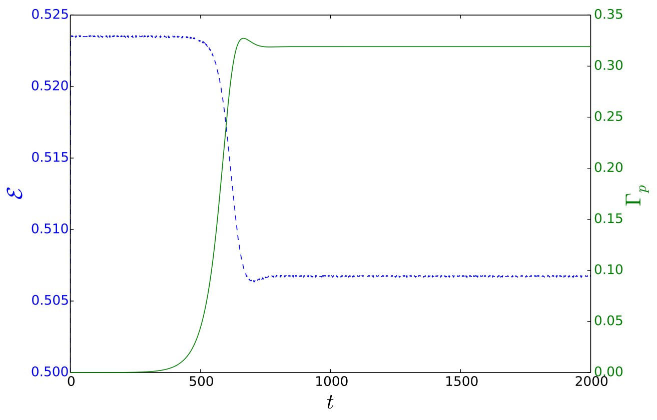


Figure 6.2: Evolution of the kinetic energy (blue line) and the radial heat flux (green line) measured at (x,y) = (0.5,0.5) for the diffusion coefficients $\kappa = \mu = 0.0183$.

The kinetic energy grows exponentially until the state of stationary convection is reached and the energy saturates. The spatial structure of pressure and electrostatic potential as well as the pressure fluctuations and the radial heat flux in the stationary state are shown in figure 6.3. The contour plots show the rising and descending of high and low density plasma advected by symmetric convection cells. We clearly observe the quasi-linear flattening of the pressure profile due to large convective transport in the central region. In addition we see that the structures of the plasma pressure fluctuations and the electrostatic potential are $\pi/2$ out of phase. This maximizes the radial convective heat transport which reduces the heat confinement and increases the heat flux as shown in figure 6.2. We compared our results to previous studies [44, 40] and find perfect agreement.

6.2 Convection with sheared flows

We now decrease the diffusion coefficients to $\kappa = \mu = 0.012$. The system reaches a quasi-stationary state at $t \approx 900$ with convection cells as in the state before, but with a seed azimuthal mean flow generated by numerical round-off errors that are unstable to the tilting mechanism. This tilted cells are a direct consequence of the azimuthal streaming motion in opposite directions at the poloidal boundaries.

The physical mechanism of the tilting instability is well understood, the first analytical work on the two-dimensional tilting instability was presented in [45].

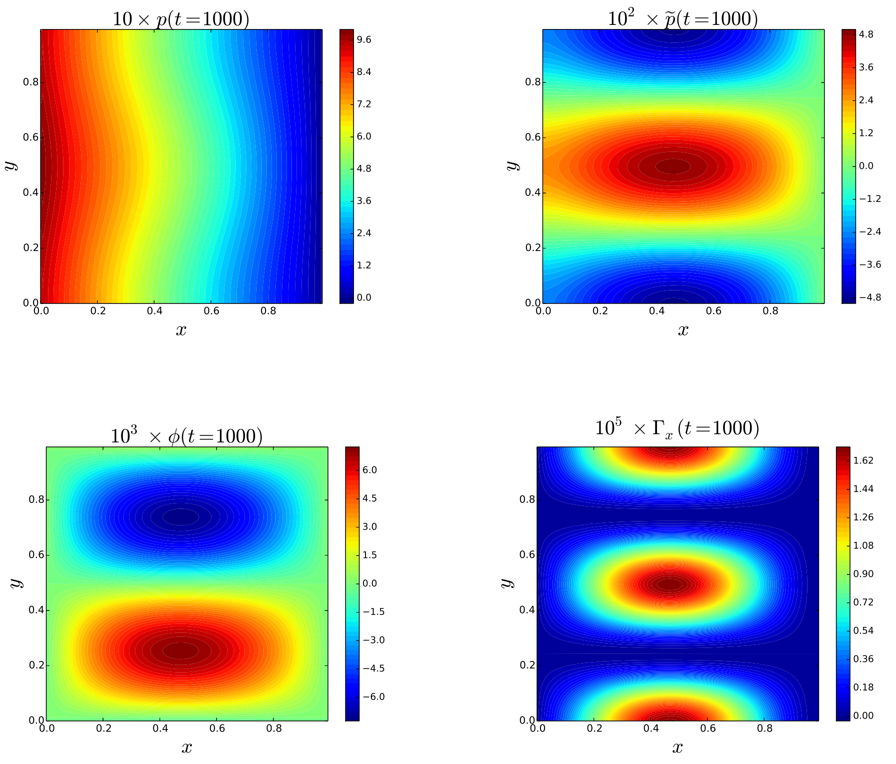


Figure 6.3: Spatial structure of the pressure and electrostatic potential, pressure fluctuations and radial heat flux in the stationary state at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.0183$.

The tilted cells are shown in figure 6.8 in addition to the pressure, pressure fluctuation and the radial heat flux. The profiles of the plasma pressure p_0, the potential ϕ_0 and the vorticity Ω_0 at $t = 1000$ are shown in figures 6.4 - 6.6. The appearance of the azimuthal streaming motion suppresses the convection cell with the opposite vorticity of the mean flow while the other cell is increased. In case of steady state this instability saturates through convective heat transport and shear flow generation by vertical momentum transport. The shear flow itself balances between the tilting instability and viscous dissipation.

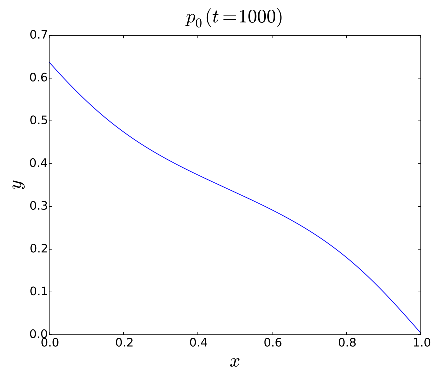


Figure 6.4: Pressure profile p_0 at $t = 1000$.

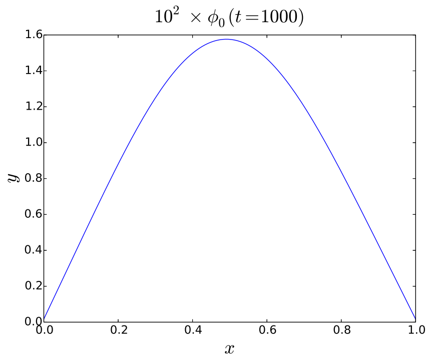


Figure 6.5: Potential profile ϕ_0 at $t = 1000$.

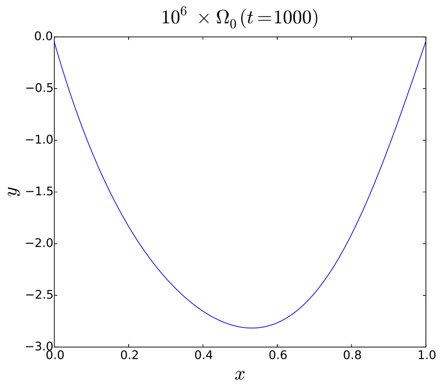


Figure 6.6: Vorticity profile Ω_0 at $t = 1000$.

As the sheared mean flow appears the amplitudes of the fluctuating motions are reduced, as we see in figure 6.7. This leads to a decrease of the convective heat flux which causes a reduction in the drive of the pressure motion. Compared to the convective state discussed before the total heat flux is reduced. This phenomenon is similar to the low to high confinement transition in magnetic confinement experiments and therefore of great interest to the plasma physics community [44].

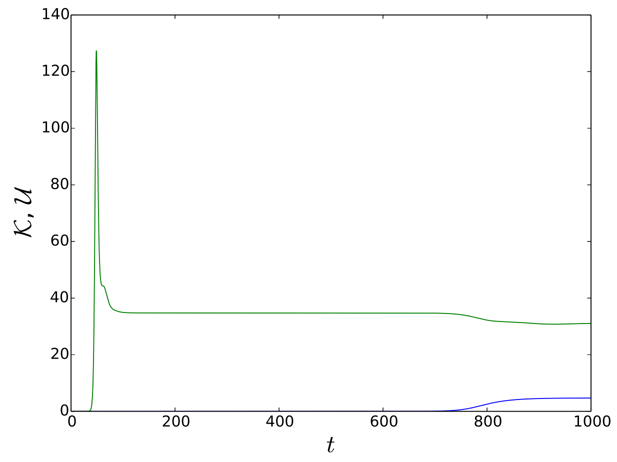


Figure 6.7: Evolution of the kinetic energy integrals $\mathcal{K}$ (green) and $\mathcal{U}$ (blue) for the diffusion coefficients $\kappa = \mu = 0.012$. The energy integral of the fluctuating motion decreases after the appearance of the sheared mean flow at $t \approx 750$.

6.3 Oscillatory motion with shared flows

For further decrease of the diffusion coefficients to $\kappa = \mu = 0.011$ we observe that the system enters a state of oscillatory convection. The direction in which the tilt initially develops depends on the numerical round-off errors and therefore can vary for different simulations with slightly different parameters. The evolution of the energy integrals is shown in figure 6.9. We clearly see that the system undergoes a stable limit cycle starting at $t \approx 750$. Figure 6.10 shows one period of oscillation. An explanation for this motion might be explained as the following.

We start with the state at $t = 873$ where the fluctuations are driving the mean flow through the tilting mechanism. The convective cells deposit almost all their energy into the mean flow, which reduces the convective transport, shown at $t = 888$. Now the low fluctuation level of the convection cells however is not able to sustain the shared mean flow which leads to a decrease of this flow due to viscous dissipation. Now the convective instability leads to a state dominated by the spatial fluctuations again, shown at $t = 900$, and the cycle

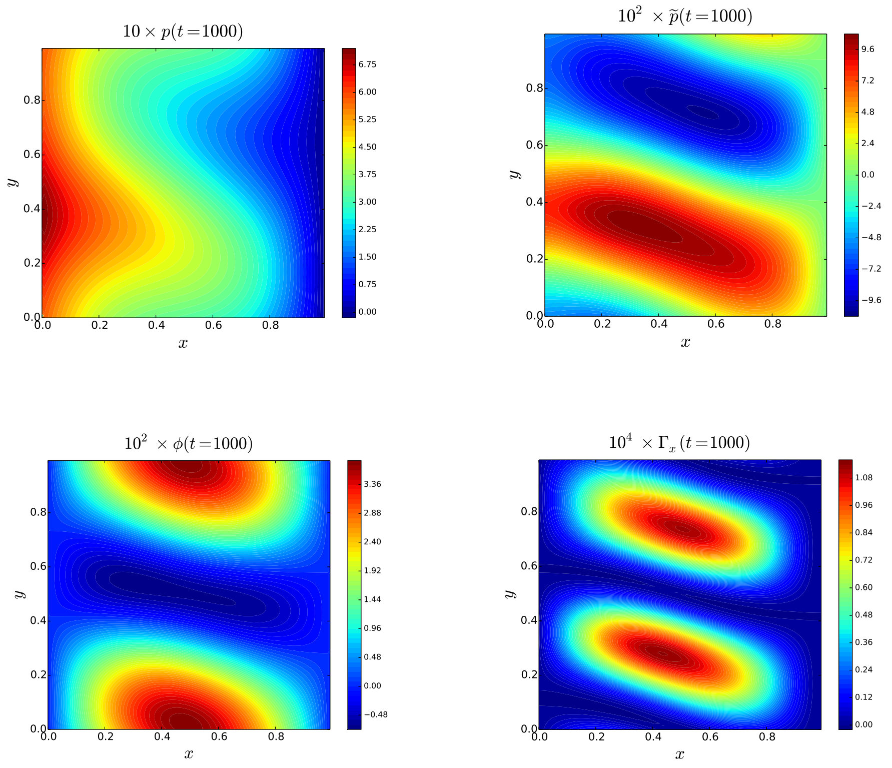


Figure 6.8: Spatial tilted structure of the pressure and electrostatic potential, pressure fluctuations and radial heat flux in the stationary state at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.012$.

repeats. The time evolution of the spatial structure shows that the convection cells swing forth and back with the oscillating mean flow without any net drift.

The integrated convective heat flux Γ_p and the integrated plasma pressure $\mathcal{E}$, are shown in figure 6.11. We observe that the integrated pressure that represents the plasma confinement, shown by the blue line, oscillates at the same time scale as the kinetic energy, whereas the convective heat flux, represented by the green line, shows an additional oscillation on a bigger time scale. A possible explanation for this effect might be the instability mechanism described in [46].

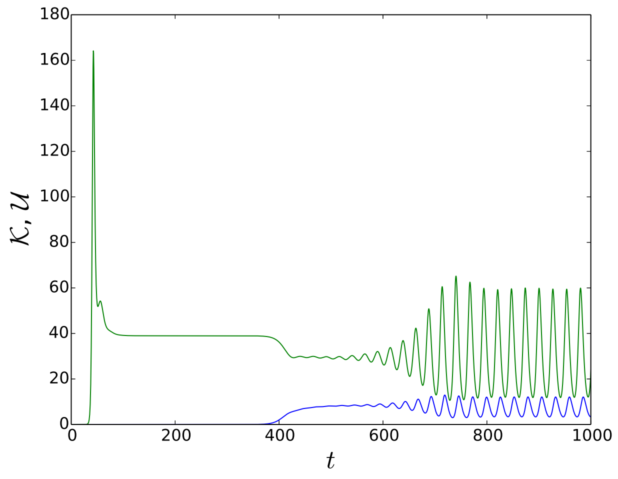


Figure 6.9: Evolution of the kinetic energy integrals $\mathcal{K}$ (green line) and $\mathcal{U}$ (blue line) for the diffusion coefficients $\kappa = \mu = 0.011$.

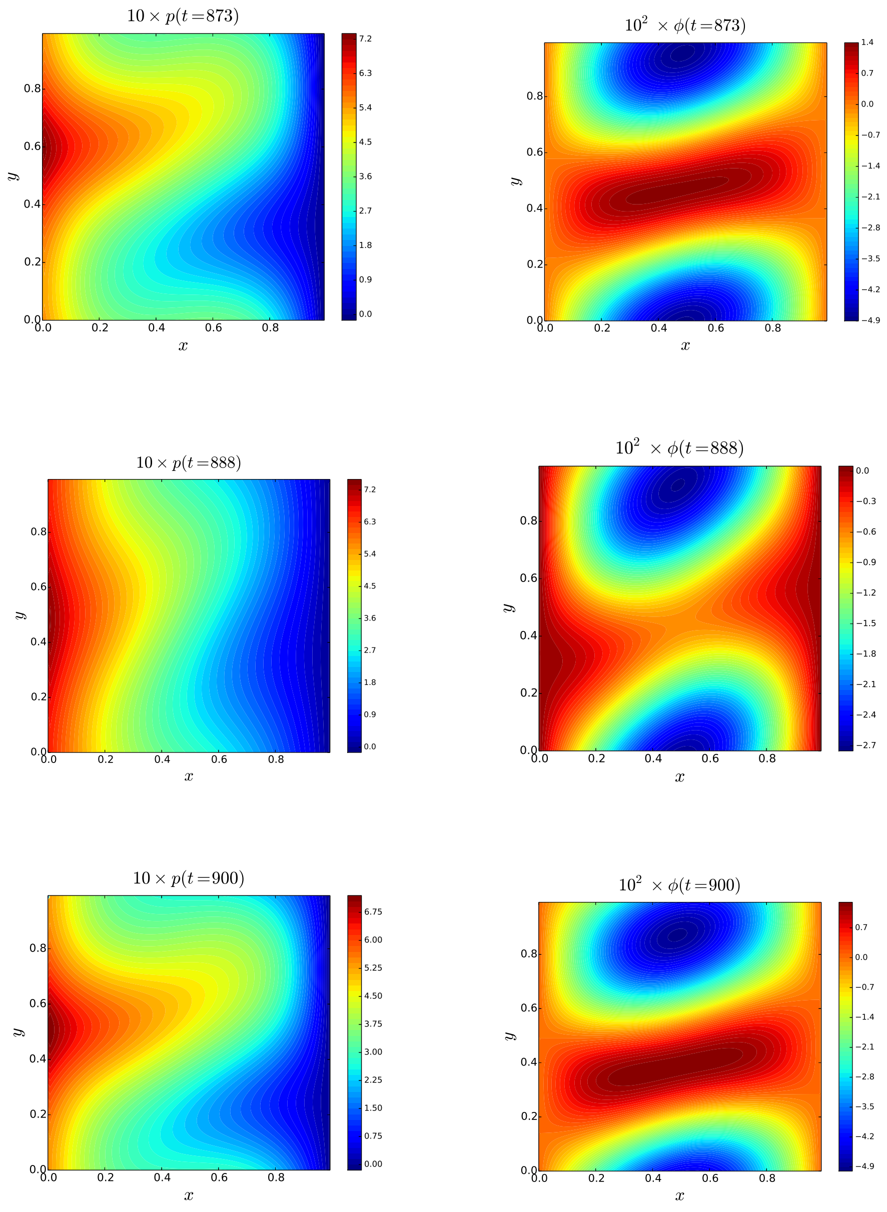


Figure 6.10: Evolution of the plasma pressure and the electrostatic potential over one period of oscillation for the diffusion coefficients $\kappa = \mu = 0.011$.

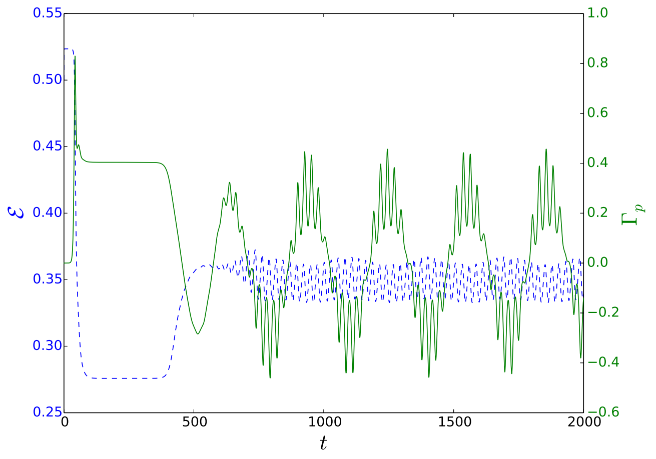


Figure 6.11: Evolution of the integrated convective heat flux Γ_p (green line) and the integrated pressure $\mathcal{E}$ (blue broken line) for the diffusion coefficients $\kappa = \mu = 0.011$, showing the oscillatory convection from $t \approx 750$ on.

6.4 Onset of turbulent convection

At $\kappa = \mu = 0.0016$ the system enters a state of turbulent convection and chaotic oscillations. The energy integrals shown in figure 6.12 show that the energy of the mean flow U exceeds the fluctuating motion energy $\mathcal{K}$ at $t \approx 75$ and that $\mathcal{U}$ lags $\mathcal{K}$ which is consistent with the energy transfer due to tilting as described above. The integrated pressure and the integrated convective heat flux are shown in figure 6.13. The convective heat flux, being the product of two strongly correlated fields, shows always strong spikes in the turbulent state. Note that the integrated pressure shows that the plasma confinement is lower than in the state of oscillatory convection. The spatial structures of the pressure and the electrostatic potential, pressure fluctuations and radial heat flux are shown in figure 6.14 exemplary at $t = 1000$. An explanation for this bursting process and a stochastic analysis of this turbulent convection is presented in the chapter **Stochastic analysis of bursty transport**.

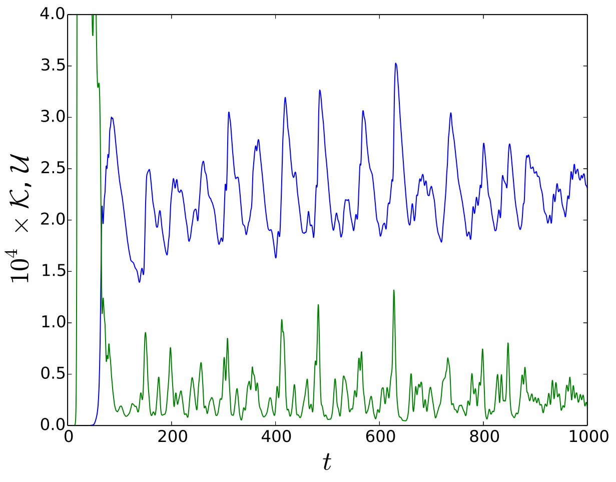


Figure 6.12: Evolution of the kinetic energy integrals $\mathcal{K}$ (green line) and $\mathcal{U}$ (blue line) for the diffusion coefficients $\kappa = \mu = 0.0016$.

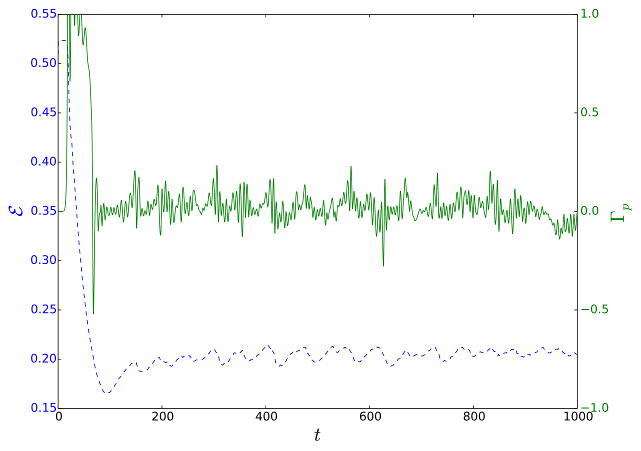


Figure 6.13: Evolution of the integrated convective heat flux Γ_p (green line) and the integrated pressure $\mathcal{E}$ (blue broken line) for the diffusion coefficients $\kappa = \mu = 0.0016$. The system shows no clear periodicity after $t \approx 100$.

6.5 Intermittent convection

We further decrease the diffusion coefficients to $\kappa = \mu = 0.0007$. The evolution of the kinetic energies is shown in figure 6.15. We observe fewer but bigger

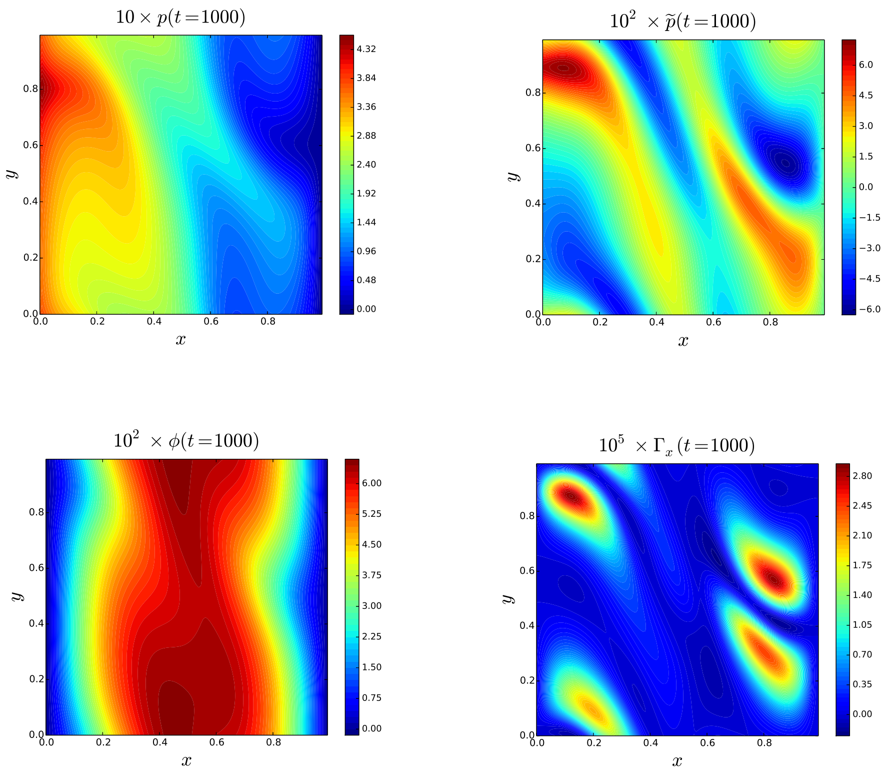


Figure 6.14: Spatial structures of the pressure and electrostatic potential, pressure fluctuations and radial heat flux at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.0016$.

peaks compared to the previous state. The evolution of the integrated convective heat flux and the integrated pressure show the same picture, presented in figure 6.16. The spatial structures of the pressure and the electrostatic potential, pressure fluctuations and radial heat flux are shown in figure 6.17 exemplary shown at $t = 1000$. We observe that the spatial structures descending from the initial convection cells are more elongated than in the state with higher doffusion coefficients. Further investigations including a stochastic analysis of this turbulent convection is presented in chapter **Stochastic analysis of bursty transport**.

The observed bursty behaviour seems to be typical for strongly driven turbulent convection systems with differential rotation [47, 48].

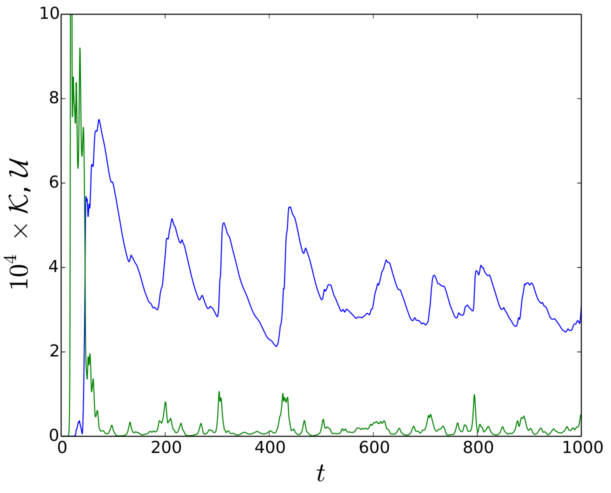


Figure 6.15: Evolution of the kinetic energy integrals $\mathcal{K}$ (green line) and $\mathcal{U}$ (blue line) for the diffusion coefficients $\kappa = \mu = 0.0007$.

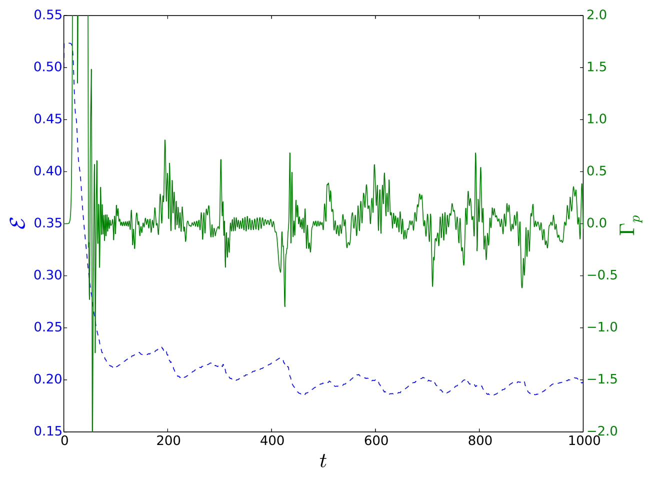


Figure 6.16: Evolution of the integrated convective heat flux Γ_p (green line) and the integrated pressure $\mathcal{E}$ (blue broken line) for the diffusion coefficients $\kappa = \mu = 0.0007$.

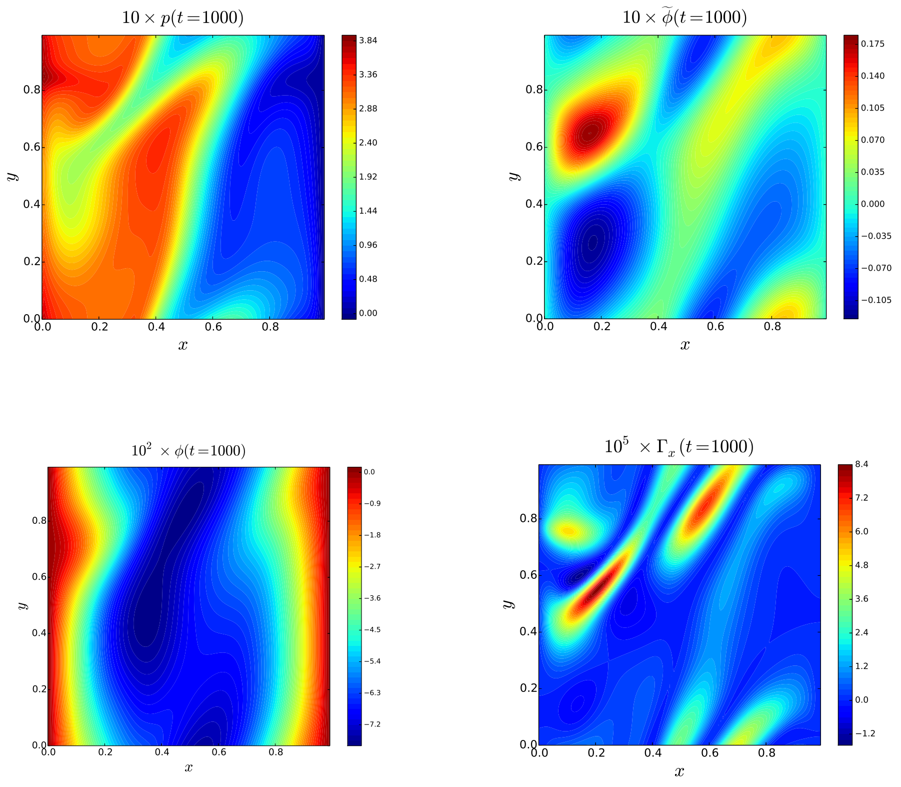


Figure 6.17: Spatial structures of the pressure and electrostatic potential, pressure fluctuations and radial heat flux at $t = 1000$ for the diffusion coefficients $\kappa = \mu = 0.0007$.

7 Statistical analysis of fluctuations

In this chapter we present a statistical analysis of simulation data from turbulent convection for two sets of diffusion coefficients discussed in chapters 6.4 and 6.5.

7.1 Statistical concepts

Before we present the results we need to define several statistical quantities and concepts.

In the following we use besides the absolute values of the investigated physical quantities normalized values as well. For a given time series of the quantity X we define the normalized variable $\widetilde{X}$ as

$$\widetilde{X} = \frac{X - \langle X \rangle}{X_{\mathrm{rms}}} \tag{7.1}$$

where $\langle X \rangle$ is the mean value of the series and X_{rms} its root mean square given by

$$X_{\mathrm{rms}} = \langle (X - \langle X \rangle)^2 \rangle^{1/2} . \tag{7.2}$$

Note that the expectation value of the new variable $\widetilde{X}$ is zero, while the standard deviation is unity.

In order to investigate the properties of large-amplitude events we define for each quantity a threshold value. If the investigated quantity exceeds this value we search for its local maximum and note its position and amplitude. In figure 7.1 and figure 7.2 we see this method at the example of the normalized radial flux for the tubulent and intermittent case with a time window $t \in (100, 1000)$. The threshold of $\widetilde{\Gamma}_x = 1.5$ is represented by the red broken line and the local maxima are displayed by the black circles. This allows us to define the waiting time τ_k between two successive bursts at t_{k-1} and t_k as $\tau_k = t_k - t_{k-1}$. In order to avoid counting two closely consecutive maxima as two different events we chose a burst length of 20 in which all values exceeding the threshold are counted at one burst.

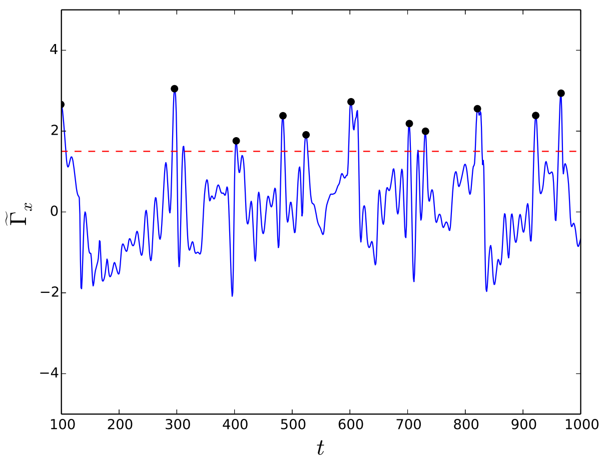


Figure 7.1: Time series of the normalized radial flux for the diffusion coefficients $\kappa = \mu = 0.0016$. The threshold at $\widetilde{\Gamma}_x = 1.5$ is represented by the reed broken line. The local maxima are displayed by the black circles.

We define the cumulative distribution function (CDF) of a random variable X as

$$\mathrm{CDF}_X(x) = \mathcal{P}[X \leq x], \tag{7.3}$$

here $\mathcal{P}[X \leq x]$ is the probability of $X \leq x$. Note that $\mathrm{CDF}_X(\infty) = 1$, $\mathrm{CDF}_X(-\infty) = 0$ and that $\mathrm{CDF}_X(x)$ is a non-decreasing function. We now use the CDF to define the probability density function (PDF) as

$$\mathrm{PDF}_X(x) = \frac{\mathrm{dCDF}_X(x)}{\mathrm{d}x}. \tag{7.4}$$

This function has the following properties:

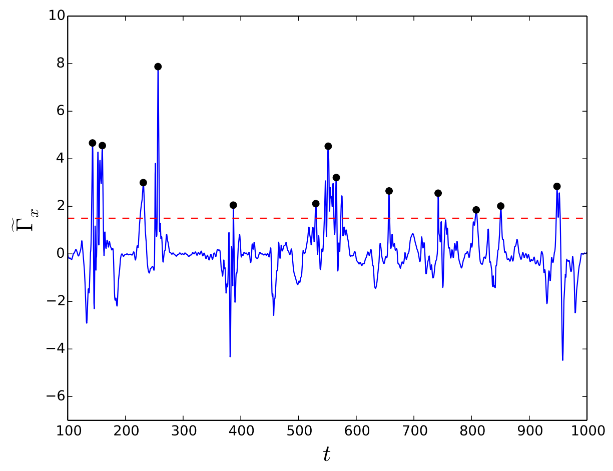


Figure 7.2: Time series of the normalized radial flux for the diffusion coefficients $\kappa = \mu = 0.0007$. The threshold at $\widetilde{\Gamma}_x = 1.5$ is represented by the reed broken line. The local maxima are displayed by the black circles.

- $\mathrm{PDF}_X(x) \geq 0$
- $\int_{-\infty}^{\infty} \mathrm{d}x\, \mathrm{PDF}_X(x) = 1$
- $\mathrm{CDF}_X(x) = \int_{-\infty}^{x} \mathrm{d}\xi\, \mathrm{PDF}_X(\xi)$
- $\mathcal{P}[a \leq X \leq b] = \int_a^b \mathrm{d}x\, \mathrm{PDF}_X(x)$

We further define the moments $\langle X^n \rangle$ where n is an integer. The angular brackets donate an average of a random variable over all its values. The raw moments are given by

$$\langle X^n \rangle = \int_{-\infty}^{\infty} \mathrm{d}x\, x^n P_X(x), \tag{7.5}$$

while the central moments are defined as

$$\mu_n = \langle (X - \langle X \rangle)^n \rangle, \quad n > 1. \tag{7.6}$$

The next property we need to define is the skewness, which is a measure of the asymmetry of the distribution function. If this property is negative, the left tail of the distribution is longer or fatter than the right tail, and reverse for positive skewness. The skewness of a symmetric distribution is zero. The definition of the skewness S_X is given by

$$S_X = \frac{\mu_3}{\mu_2^3/2}. \tag{7.7}$$

The kurtosis or flatness of a distribution measures how peaked a distribution is. In the case of a large kurtosis the distribution is sharp or has fatter tails, while a distribution with a rounded peak and thin tails has a low kurtosis. The flatness F_X of X is defined by

$$F_X = \frac{\mu_4}{\mu_2}. \tag{7.8}$$

We further use the conditional average method to investigate the properties of large-amplitude fluctuations in noisy signals, as used in previous studies on fusion plasma devices such as [49, 50]. For a given time series $\Phi(t)$ we require a condition C to pick out certain sub-intervals of $\Phi(t)$. An example for such a condition would be the exceedance of a threshold as discussed before. The conditionally averaged signal is then given by the ensemble average of all sub-intervals

$$\Phi_C = \langle \Phi | C \rangle \,. \tag{7.9}$$

For each maxima of Φ above the threshold we pick a sub-interval $(t_m - \Delta, t_m + \Delta)$ where t_m stands for the time of the maxima and 2Δ is the length of the subinterval. We further require that those sub-intervals are not allowed to overlap. The conditionally averaged signal is then given by

$$\Phi_C = \frac{1}{M} \sum_{m=1}^{M} \Phi(t - t_m) \, \text{for} \, t - t_m \in (-\Delta, \Delta), \tag{7.10}$$

where M represents the total number on subintervals [51, 52].

7.2 Probability density functions

The first statistical analysis we perform is to investigate the distribution of the radial velocity fluctuations $\widetilde{v}_x$, the radial heat flux $\widetilde{\Gamma}_x$ as well as the fluctuations of the electrostatic potential $\widetilde{\phi}$, the vorticity $\widetilde{\Omega}$ and the plasma pressure $\widetilde{p}$. We therefore measure these quantities at one point in the centre of the simulation domain at $(x, y) = (0.5, 0.5)$. These quantities are recorded with a sampling time of $\Delta_t = 0.05$. In figure 7.3 we present the normalized probability distribution function for the radial velocity fluctuation $\widetilde{v}_x$ for the diffusion coefficients $\kappa = \mu = 0.0016$. The distribution function has a nearly Gaussian form with vanishing skewness and kurtosis of $F_{\widehat{v}} = 3.08$. We compare this distribution to the high turbulence state with the diffusion coefficients of $\kappa = \mu = 0.0007$, shown in figure 7.4. The local fluctuations become increasingly non-Gaussian and develop exponential tails. Again there is essentially no skewness of the distribution and the kurtosis becomes $F_{\widehat{v}} = 4.87$.

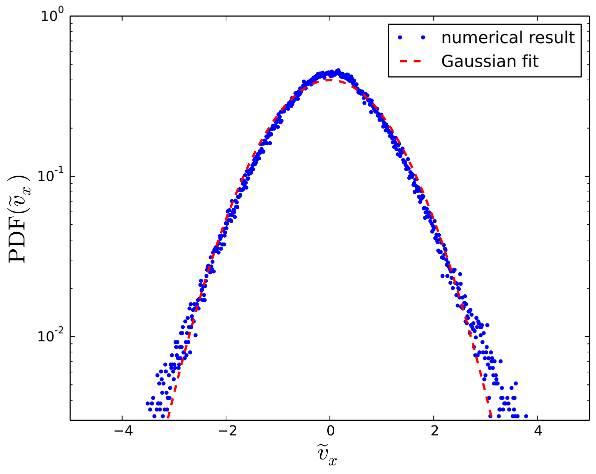


Figure 7.3: Normalized PDF for the radial velocity fluctuation $\widetilde{v}_x$ measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$, showing nearly Gaussian statistics.

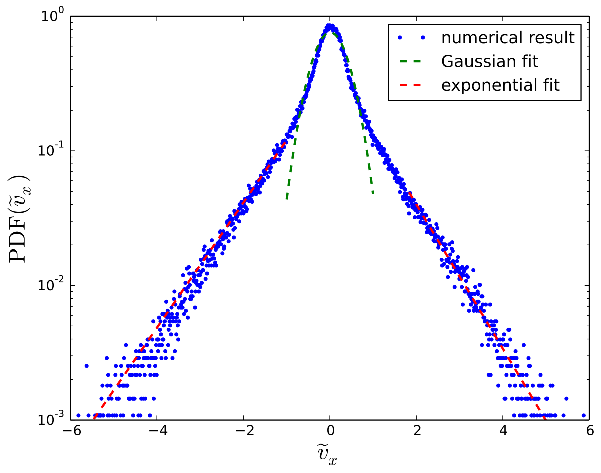


Figure 7.4: Normalized PDF for the radial velocity fluctuation $\widetilde{v}_x$ measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$, revealing exponential tails for large fluctuation amplitudes.

In order to understand the non-Gaussian probability distribution for small diffusion coefficients, we show the evolution of the normalized radial heat flux in figure 7.2 for $\kappa = \mu = 0.0007$. We observe that the plasma system displays quasi-periodic bursting of the convective transport. Large amplitude events follow after relatively quiescent periods with suppressed convective transport.

An explanation of this bursting process may be given as follows. Without any significant mean flow energy the spatial structures grow exponentially in amplitude until the convection cells give most of their energy to the sheared mean flows due to the tilting instability. This leads to a differential advection that suppresses the remaining convective motion. This causes a temporal decay of the mean flow energy due to viscous dissipation and increases the confined heat. Now the gradient of the mean pressure becomes strong enough again to drive the primary instability mechanism which leads to a new burst of fluctuations and transport and the cycle repeats. The observed relatively long periods of suppressed fluctuation levels broken by strong bursts lead to the quasi-periodic occurrence of large amplitude events. This results in the non-Gaussian shape of the PDFs [40].

The probability density functions of the normalized pressure $\widetilde{p}$, electrostatic potential $\widetilde{\phi}$, vorticity $\widetilde{\Omega}$ and radial heat flux $\widetilde{\Gamma}$ are shown in figure 7.5 for the diffusion coefficients $\kappa = \mu = 0.0016$. The distributions are compared to a fitted normal distribution and are measured in the centre of the domain at (x,y) =(0.5,0.5) with a sampling time of $t = 0.05$. We observe that the distribution functions of the normalized potential and vorticity show nearly Gaussian statistics. The normalized pressure and the normalized radial heat flux clearly deviate from the normal distribution. An explanation for this might be the different boundary conditions used for the pressure field. Since we use a Neumann condition at the inner radial boundary and a Dirichlet condition at the outer boundary we do not expect the distribution functions to be symmetric. The radial heat flux, which depends on the pressure and the radial velocity, thus deviates from the normal distribution as well. For the electrostatic potential and vorticity we use Dirichlet conditions on both boundaries, which is consistent the symmetric distribution.

We investigate the same properties for the intermittent convection with the diffusion coefficients $\kappa = \mu = 0.0007$, shown in figure 7.6. We observe that the distributions become increasingly non-Gaussian and develop exponential tails as for the radial velocity fluctuations shown in figure 7.4. The explanation for this is the same as for the distribution function of the radial velocity explained above.

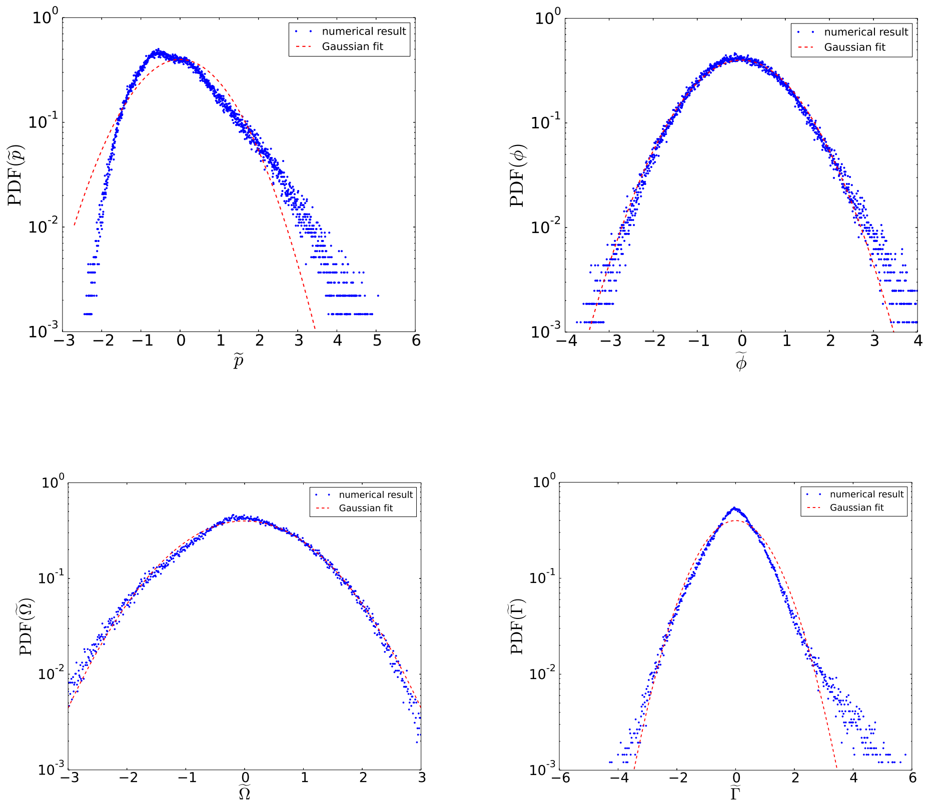


Figure 7.5: Probability distribution functions of the normalized pressure $\widetilde{p}$, potential $\widetilde{\phi}$, vorticity $\widetilde{\Omega}$ and the radial heat flux $\widetilde{\Gamma}$, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$.

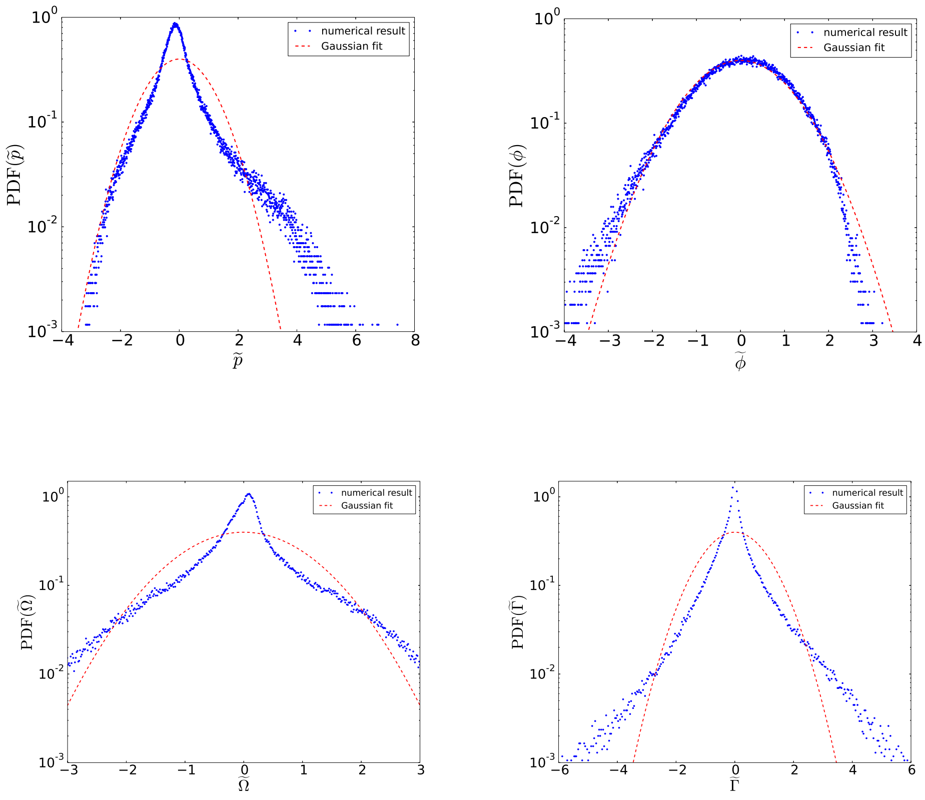


Figure 7.6: Probability distribution functions of the normalized pressure $\widetilde{p}$, potential $\widetilde{\phi}$, vorticity $\widetilde{\Omega}$ and the radial heat flux $\widetilde{\Gamma}$, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$.

7.3 Waiting time distribution

Next, we investigate the distribution of the waiting time between consecutive large-amplitude bursts. The probability density function of the waiting time of the normalized pressure for the diffusion coefficients $\kappa = \mu = 0.0016$ is presented in figure 7.7. The measurements are performed in the middle of the domain at (x,y) = (0.5,0.5). The distributions are compared to a fitted exponential distribution. The scale parameter of the exponential distribution is $\langle\tau\rangle = 46.30$.

We perform the same measurements for the intermittent convection case with the diffusion coefficients $\kappa = \mu = 0.0007$. The scale parameter becomes $\langle\tau\rangle = 74.78$. This agrees with the results of chapter 6.5 where we observe fewer but stronger bursts compared to the states with $\kappa = \mu = 0.0016$. In both cases the waiting time distribution is well described by an exponential distribution. This is in full agreement with previous studies [53] and experimental observations [54, 55].

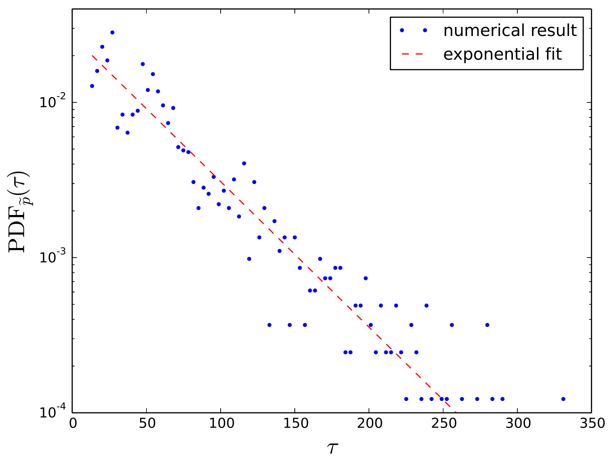


Figure 7.7: Waiting time distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$.

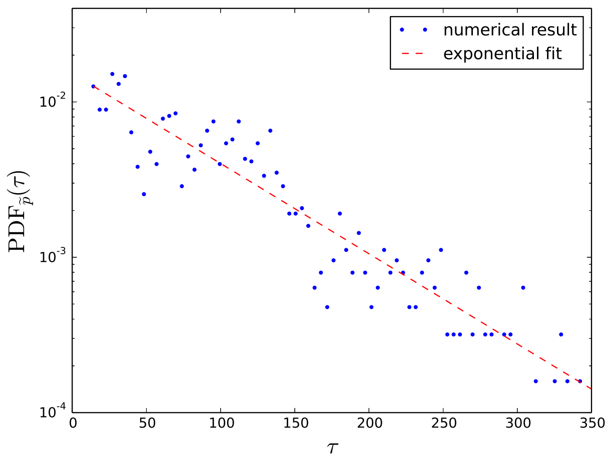


Figure 7.8: Waiting time distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$.

7.4 Amplitude distribution

Each time the normalized pressure exceeds the threshold value we measure in addition to the waiting time also the peak amplitude of the event. The amplitude distribution of the normalized pressure for the diffusion coefficients $\kappa = \mu = 0.0016$ is shown in figure 7.9. We perform the measurement again in the middle of the domain at (x,y) = (0.5,0.5). The results are compared to a fitted exponential distribution giving a mean amplitude of $\langle A \rangle = 2.27$.

We compare the results to the highly turbulent state with $\kappa = \mu = 0.0007$ shown in figure 7.10. The distribution is again fitted with an exponential distribution giving a mean amplitude of $\langle A \rangle = 2.80$. The result agrees with the observations of chapter 6.5. In the intermittent state we observe that the large-amplitude bursts are rarer but have higher amplitudes.

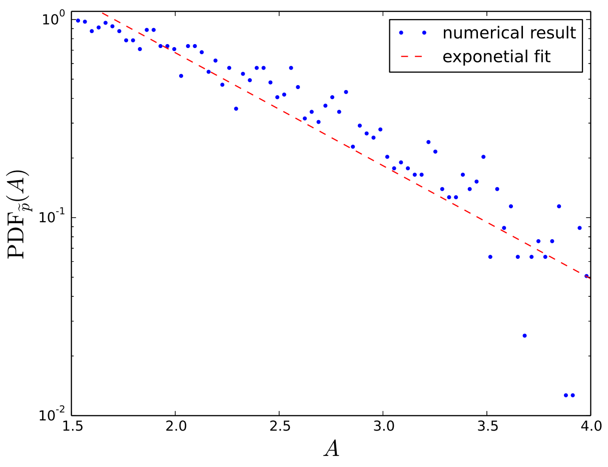


Figure 7.9: Amplitude distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$.

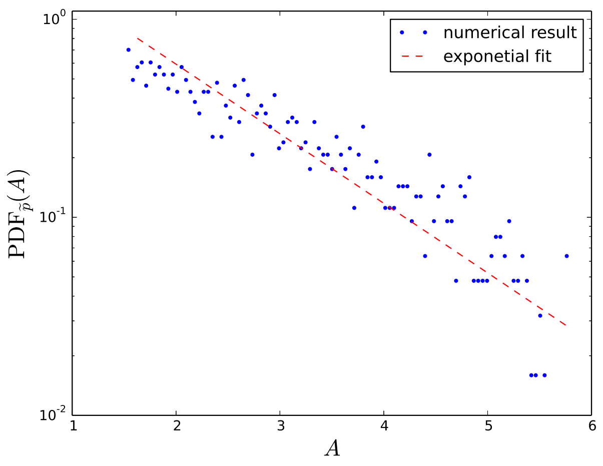


Figure 7.10: Amplitude distribution of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$. An event is registered if the normalized pressure exceeds the threshold value of $\widetilde{p} = 1.5$.

7.5 Conditionally averaged waveform

We calculate the conditionally averaged waveform for the normalized pressure $\widetilde{p}$ of 2000 events for the state with $\kappa = \mu = 0.0016$, shown in figure 7.11. As we see, shortly before and after each event that exceeds the threshold density, a smaller peak emerges. An explanation for this might be the poloidal propagation of the spatial structures. Since we use periodic boundary conditions in the poloidal direction the spatial structures, propagating in both poloidal and radial directions, may pass the point where we perform the measurements more than once during the time window we chose for one event. As we see in figure 7.11 the spatial structures have a transit time in poloidal direction of $t \approx 13$ which is consistent with the values of v_0.

We perform the same measurement for the intermittent state with $\kappa = \mu = 0.0007$, shown in figure 7.12. We compare each tail separately to an exponential function of the form $f(\tau) = \exp(\tau/\tau_r)$ for $\tau \le 0$ which results in a characteristic rise time $\tau_r = 4.76$ and $f(\tau) = \exp(-\tau/\tau_r)$ for $\tau \ge 0$ which results in a characteristic fall time $\tau_f = 7.14$, giving the estimated duration time $\tau_d = \tau_r + \tau_f = 11.9$. Here, we do not observe any additional peak and the distribution shows a nearly exponential shape. This indicates that the radial velocity of the spatial structures is high enough for the entire structure to traverse the point where the measurements are performed before the structure traverses the whole length of the domain in poloidal direction.

Again, the shape of the waveform for the highly turbulent state agrees with experimental observations [54, 55].

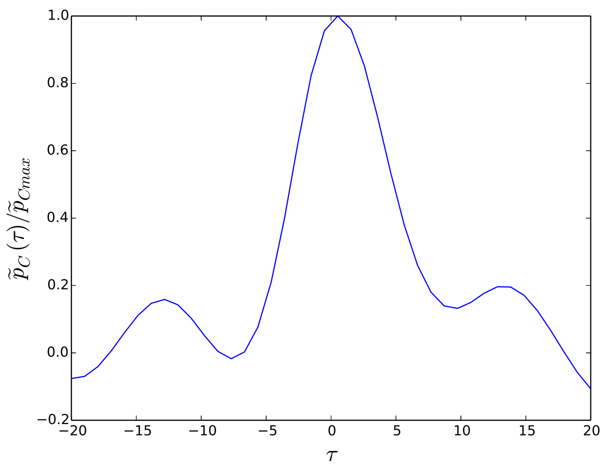


Figure 7.11: Conditionally averaged waveform of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$.

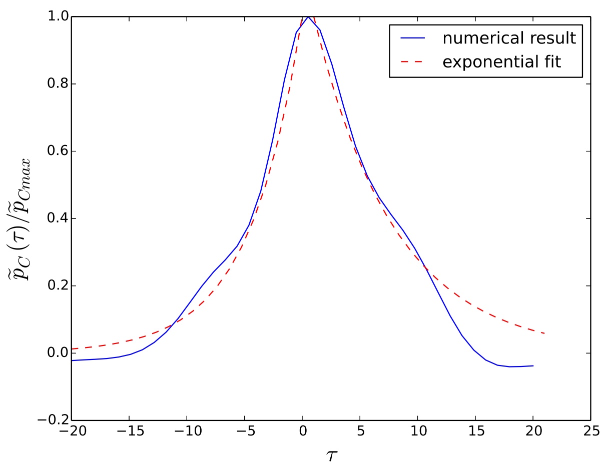


Figure 7.12: Conditionally averaged waveform of the normalized pressure, measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$.

7.6 Autocorrelation function

In figure 7.13 the autocorrelation function is shown for the normalized pressure for the state with $\kappa = \mu = 0.0016$. We observe that the exponential decay of the autocorrelation function is superimposed by an oscillation with the frequency $\omega \approx 7$. An explanation for this might be the mechanism discussed in chapter 7.5.

For the intermittent state with $\kappa = \mu = 0.0007$ we observe that the autocorrelation function of the normalized pressure takes an exponential form as we would expect from the previous results for conditional averaging. We obtain a time scale of $\tau_C = 2.7$ which deviated clearly from the estimated duration time of $\tau_d = 11.9$. An explanation might be the fact, that the autocorrelation function weights all amplitudes equally whereas the conditionally averaged waveform counts only big amplitudes. This would indicate that small amplitudes with small correlation times dominate the autocorrelation function.

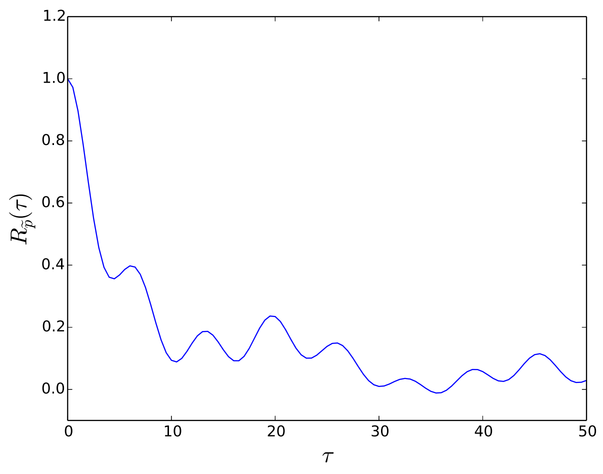


Figure 7.13: Autocorrelation function of the normalized pressure measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0016$.

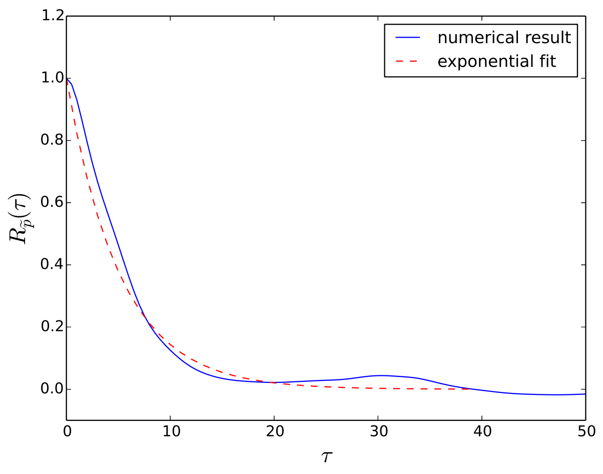


Figure 7.14: Autocorrelation function of the normalized radial velocity measured at $(x, y) = (0.5, 0.5)$ for the diffusion coefficients $\kappa = \mu = 0.0007$.

8 Conclusion and Outlook

In this thesis we investigate convective motions in magnetized plasmas by means of two-dimensional numerical simulations. In particular, our investigation focuses on intermittent fluctuations and turbulence-induced transport in magnetically confined fusion plasmas. We derive a set of reduced fluid equations describing the evolution of plasma pressure and electric drift vorticity for a two-dimensional plane describing the plasma motions perpendicular to the magnetic field. We present a numerical simulation code implemented and parallelized on graphical processing units on order to perform long time simulations and compute time series of unprecedented duration. We observe significant speedup compared to sequential Fortran implementations. In terms of numerical results we perform a parameter scan for the diffusion coefficients and identify different transport and confinement states. We observe states of chaotic oscillations and intermittent convection which we investigate by performing signal analysis techniques. We observe that the probability density functions for the normalized radial velocity, heat flux and pressure fluctuations starting from nearly Gaussian form at the onset of turbulent convection become increasingly non-Gaussian and develop exponential tails with increasing heat flux drive. In addition we observe that the waiting time and amplitude distribution of quasi-periodic bursts take a nearly exponential form. We further compare those results to experimental measurements and predictions from stochastic modelling and find very good agreement.

A more detailed overview of the different chapters looks as the following:

In chapter two we derive the model equations describing collective motions of non-uniformly magnetized plasmas. These describe the evolution of plasma pressure and electric drift vorticity for a two-dimensional plane perpendicular to the magnetic field. The model equations are derived from the momentum equation and the resulting drifts in toroidal geometry applying local slab coordinates. We introduce dimensionless variables in order to reduce the parameter scale to two free parameters, the pressure diffusion coefficient κ and the vorticity equivalent μ.

Chapter three is dedicated to the numerical methods that are implemented in the simulation code. We describe finite differences and spectral schemes and Neumann, Dirichlet and periodic boundary conditions. We use an Arakawa scheme for the two-dimensional advection and present the implementation of a stiffly stable semi-implicit time integration scheme. In addition we discuss the computational complexity of the different methods.

Code parallelization and graphical processing units are described in chapter four. We present a short overview of CUDA programming and compare our implementation to previous Fortran simulations in terms of performance. In chapter five we evaluate the code with simplified but topic related models and observe perfect agreement with theoretical predictions and previous studies.

In chapter six we finally present the simulation results for different transport and confinement states in flux driven thermal convection. We identify states of stationary convection, convection with sheared flows, oscillatory motion and finally turbulent and intermittent convection.

In the last chapter we perform statistical analysis of long time series from single-point recordings. We compare the results to experimental measurements and predictions of stochastic modelling.

Outlook

The developed GPU code is fully flexible in its treatment of initial and boundary conditions and therefore suitable for numerous different models. Because of its increasing speedup for an increasing number of grid points compared to sequential implementations, the code is highly suitable for more complex models that require higher resolution. An example would be the implementation of sheath losses on one side of the simulation domain corresponding to transport along field lines in the SOL of a tokamak plasma. The simplest approach would describe the sheath losses by linear damping terms. For that we would divide the simulation domain in two parts: the SOL region and the wall shadow region.

The border between those regions is given by L_{SOL}. The model equations would take the form

$$\begin{aligned} \frac{\partial \ln n}{\partial t} + \{\phi, \ln n\} &= \kappa \nabla_\perp^2 \ln n + \kappa \left(\nabla_\perp \ln n\right)^2 - \sigma_n\left(x\right) \\ \frac{\partial \Omega}{\partial t} + \{\phi, \Omega\} + \frac{\partial \ln n}{\partial y} &= \mu \nabla_\perp^2 \Omega - \sigma_\Omega\left(x\right)\Omega \end{aligned} \tag{8.1}$$

where $\sigma_{n/\Omega}\left(x\right)$ is a linear damping term defined by

$$\sigma_{n/\Omega}\left(x\right) = \begin{cases} 0 & \text{for } x < L_{SOL} \\ \hat{\sigma}_{n/\Omega} & \text{for } x > L_{SOL} \end{cases} \tag{8.2}$$

and where we use $\ln n$ instead of n, allowing order unity variations of the particle density [56, 57].

The code will be merged with a GPU code using a spectral Fourier Galerkin method written by Ph.D. Ralph Kube. This enables to chose between Neumann, Dirichlet or periodic boundary conditions in the radial direction. The code will therefore be used for further research.

In terms of statistical analysis, a study of the time series measured at different positions in the simulation domain would be recommended. All measurements in this thesis have been done at the middle of the domain, measurements near the domain boundary would be relevant in order to investigate the effects of the boundaries. In addition, further stochastic analysis of the time series is recommended.

Bibliography

[1] Anthony J Webster. Fusion: Power for the future. *Physics education*, 38(2):135, 2003.

[2] T Kenneth Fowler. Nuclear power—fusion. *Reviews of Modern Physics*, 71(2):S456, 1999.

[3] Jeffrey P Freidberg. *Plasma physics and fusion energy*. Cambridge university press, 2008.

[4] Richard Pitts, Richard Buttery, and Simon Pinches. Fusion: the way ahead. *Physics World*, 19(3):20, 2006.

[5] European consortium for the development of fusion energy. `https://www.euro-fusion.org/`. Accessed: 29.4.2016.

[6] Volker Naulin. Turbulent transport and the plasma edge. *Journal of nuclear materials*, 363:24–31, 2007.

[7] OE Garcia. Blob transport in the plasma edge: a review. *Plasma and Fusion Research*, 4:019–019, 2009.

[8] RJ Maqueda, DP Stotler, SJ Zweben, and NSTX The. Intermittency in the scrape-off layer of the national spherical torus experiment during h-mode confinement. *Journal of nuclear materials*, 415(1):S459–S462, 2011.

[9] SI Krasheninnikov, DA D'ippolito, and JR Myra. Recent theoretical progress in understanding coherent structures in edge and sol turbulence. *Journal of Plasma Physics*, 74(05):679–717, 2008.

[10] OE Garcia, NH Bian, V Naulin, AH Nielsen, and J Juul Rasmussen. Mechanism and scaling for convection of isolated structures in nonuniformly magnetized plasmas. *Physics of Plasmas (1994-present)*, 12(9):090701, 2005.

[11] Sergei I Krasheninnikov. On scrape off layer plasma transport. *Physics Letters A*, 283(5):368–370, 2001.

[12] OE Garcia, NH Bian, and W Fundamenski. Radial interchange motions of plasma filaments. *Physics of Plasmas (1994-present)*, 13(8):082309, 2006.

[13] Andreas Schmid, Albrecht Herrmann, HW Müller, et al. Experimental observation of the radial propagation of elm induced filaments on asdex upgrade. *Plasma Physics and Controlled Fusion*, 50(4):045007, 2008.

[14] OE Garcia, J Horacek, and RA Pitts. Intermittent fluctuations in the tcv scrape-off layer. *Nuclear Fusion*, 55(6):062002, 2015.

[15] Harnessing the power of the sun: fusion reactors. `http://www.scienceinschool.org/2012/issue22/fusion`. Accessed: 29.3.2016.

[16] RJ Maqueda, GA Wurden, DP Stotler, SJ Zweben, B LaBombard, JL Terry, JL Lowrance, VJ Mastrocola, GF Renda, DA D'Ippolito, et al. Gas puff imaging of edge turbulence. *Review of Scientific Instruments*, 74(3):2020–2026, 2003.

[17] Ralph Kube. Numerical studies of radial filament motion in toroidally confined plasmas. Master's thesis, University of Tromsø, Faculty of Science and Technology Department of Physics and Technology, 2010.

[18] SI Braginskii. Transport processes in a plasma. *Reviews of plasma physics*, 1:205, 1965.

[19] OE Garcia, NH Bian, V Naulin, AH Nielsen, and J Juul Rasmussen. Two-dimensional convection and interchange motions in fluids and magnetized plasmas. *Physica Scripta*, 2006(T122):104, 2006.

[20] OE Garcia. 2dads documentation. *unpublished*, 2011.

[21] cufft | nvidia developer. `https://developer.nvidia.com/cufft`. Accessed: 29.4.2016.

[22] cusolver :: Cuda toolkit documentation. `http://docs.nvidia.com/cuda/cusolver/`. Accessed: 29.3.2016.

[23] Volker Naulin and Anders H Nielsen. Accuracy of spectral and finite difference schemes in 2d advection problems. *SIAM Journal on Scientific Computing*, 25(1):104–126, 2003.

[24] S Lennart Johnsson and Robert L Krawitz. Cooley-tukey fft on the connection machine. *Parallel Computing*, 18(11):1201–1221, 1992.

[25] Steven G Johnson. Notes on fft-based differentiation, 2011.

[26] George Em Karniadakis, Moshe Israeli, and Steven A Orszag. High-order splitting methods for the incompressible navier-stokes equations. *Journal of computational physics*, 97(2):414–443, 1991.

[27] Akio Arakawa. Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. part i. *Journal of Computational Physics*, 1(1):119–143, 1966.

[28] William H Press. *Numerical recipes in C/C++*. Cambridge University Press, 2002.

[29] Nvidia profiler user's guide. `http://docs.nvidia.com/cuda/profiler-users-guide/#axzz46G2kPmNk`. Accessed: 19.4.2016.

[30] David B Kirk and W Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Newnes, 2012.

[31] Herb Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's journal*, 30(3):202–210, 2005.

[32] Barry Wilkinson and Michael Allen. *Parallel programming*, volume 999. Prentice hall Upper Saddle River, NJ, 1999.

[33] Shane Ryoo, Christopher I Rodrigues, Sara S Baghsorkhi, Sam S Stone, David B Kirk, and Wen-mei W Hwu. Optimization principles and application performance evaluation of a multithreaded gpu using cuda. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 73–82. ACM, 2008.

[34] Victor W Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, et al. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 451–460. ACM, 2010.

[35] Cuda c programming guide. `http://docs.nvidia.com/cuda/cuda-c-programming-guide/`. Accessed: 5.4.2016.

[36] L Genovese. Graphic processing units: A possible answer to hpc. In *4th ABINIT Developer Workshop*, 2009.

[37] Naga K Govindaraju, Brandon Lloyd, Yuri Dotsenko, Burton Smith, and John Manferdelli. High performance discrete fourier transforms on graphics processors. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 2. IEEE Press, 2008.

[38] Mark Silberstein, Assaf Schuster, Dan Geiger, Anjul Patney, and John D Owens. Efficient computation of sum-products on gpus through software-managed cache. In *Proceedings of the 22nd annual international conference on Supercomputing*, pages 309–318. ACM, 2008.

[39] Two-dimensional advection-diffusion solver. `https://github.com/gregordecristoforo/2dads`. Accessed: 12.5.2016.

[40] OE Garcia, NH Bian, JV Paulsen, S Benkadda, and K Rypdal. Confinement and bursty transport in a flux-driven convection model with sheared flows. *Plasma physics and controlled fusion*, 45(6):919, 2003.

[41] Harold P Furth, John Killeen, and Marshall N Rosenbluth. Finite-resistivity instabilities of a sheet pinch. *Physics of Fluids (1958-1988)*, 6(4):459–484, 1963.

[42] M Berning and KH Spatschek. Bifurcations and transport barriers in the resistive-g paradigm. *Physical Review E*, 62(1):1162, 2000.

[43] OE Garcia. Two-field transport models for magnetized plasmas. *Journal of plasma physics*, 65(02):81–96, 2001.

[44] OE Garcia. *Convective transport and sheared flows in fluids and magnetized plasmas*. PhD thesis, PhD thesis, University of Tromsø, Norway, 2002.

[45] LN Howard and R Krishnamurti. Large-scale flow in turbulent convection: a mathematical model. *Journal of fluid mechanics*, 170:385–410, 1986.

[46] JM Finn, JF Drake, and PN Guzdar. Instability of fluid vortices and generation of sheared flow. *Physics of Fluids B: Plasma Physics (1989-1993)*, 4(9):2758–2768, 1992.

[47] V Naulin, J Juul Rasmussen, and J Nycander. Transport barriers and edge localized modes-like bursts in a plasma model with turbulent equipartition profiles. *Physics of Plasmas (1994-present)*, 10(4):1075–1082, 2003.

[48] Zhihong Lin, TS Hahm, WW Lee, WM Tang, and RB White. Gyrokinetic simulations in general geometry and applications to collisional damping of zonal flows. *Physics of Plasmas (1994-present)*, 7(5):1857–1862, 2000.

[49] GY Antar, G Counsell, and J-W Ahn. On the scaling of avaloids and turbulence with the average density approaching the density limit. *Physics of Plasmas (1994-present)*, 12(8):082503, 2005.

[50] DL Rudakov, JA Boedo, RA Moyer, S Krasheninnikov, AW Leonard, MA Mahdavi, GR McKee, GD Porter, PC Stangeby, JG Watkins, et al. Fluctuation-driven transport in the diii-d boundary. *Plasma physics and controlled fusion*, 44(6):717, 2002.

[51] Audun Theodorsen. Stochastic modelling of intermittent scrape-off layer plasma fluctuations. 2015.

[52] HL Pécseli and J Trulsen. A statistical analysis of numerically simulated plasma turbulence. *Physics of Fluids B: Plasma Physics (1989-1993)*, 1(8):1616–1636, 1989.

[53] OE Garcia. Stochastic modeling of intermittent scrape-off layer plasma fluctuations. *Physical review letters*, 108(26):265001, 2012.

[54] A Theodorsen, OE Garcia, J Horacek, R Kube, and RA Pitts. Scrape-off layer turbulence in tcv: evidence in support of stochastic modelling. *Plasma Physics and Controlled Fusion*, 58(4):044006, 2016.

[55] OE Garcia, I Cziegler, R Kube, B LaBombard, and JL Terry. Burst statistics in alcator c-mod sol turbulence. *Journal of Nuclear Materials*, 438:S180–S183, 2013.

[56] OE Garcia, J Horacek, RA Pitts, AH Nielsen, W Fundamenski, JP Graves, V Naulin, and J Juul Rasmussen. Interchange turbulence in the tcv scrape-off layer. *Plasma physics and controlled fusion*, 48(1):L1, 2005.

[57] DA Russell, JR Myra, and DA D'Ippolito. Collisionality and magnetic geometry effects on tokamak edge turbulent transport. ii. many-blob turbulence in the two-region model. *Physics of Plasmas (1994-present)*, 14(10):102307, 2007.