NTNU – Trondheim
Norwegian University of
Science and Technology

TFY4510 PHYSICS, SPECIALIZATION PROJECT - 2019

Studies of Turbulent Diffusion through Direct Numerical Simulation

Daniel Ørnes Halvorsen
Supervisor: Assoc. Prof. Tor Nordam

December 20, 2019

Preface

This report is written as a part of the course TFY4510 - Physics, Specialization Project, at NTNU in Trondheim. The specialization project yields 15 ETCS credits and has been conducted during the fall of 2019.

I would like to thank my supervisor, Associate Professor Tor Nordam, for his encouragement and advice throughout the semester. The discussions on both the numerical methods involved and the results of this thesis have been invaluable and much appreciated.

- Daniel Ørnes Halvorsen

Table of Contents

Chapter 1

Introduction

Most flows occurring in nature and in engineering applications are turbulent. The boundary layer of earth's atmosphere, the motion of cumulus clouds and currents in the water below the surface of the ocean are all turbulent flows. Many combustion processes are also turbulent. As well is the flow of water in rivers and canals, wakes of ships, cars, submarines and aircrafts. A turbulent flow is something that emerges in almost all fields of science involving fluids. Laminar flow in fluid mechanics is the exception, not the rule.

Examples of transportation of quantities in such flow fields are ash from volcanic eruption, advected with hot and turbulent air, oil-leakage from a rig in the North-Sea, advected with the ocean currents or bacteria transported in the ventilation of a hospital. The ability to predict the future distribution of a quantity transported in a flow field, either laminar or turbulent, is something that have interested scientist since the first transport models were designed. With the help of high performance computers, the distribution can be found numerically.

Turbulent diffusion is a term that appears frequently in this field of study. It is defined as the transport of a quantity such as mass, heat or momentum within a system, caused by chaotic and random motions, such as in a turbulent flow field. Turbulent diffusion occurs much more rapidly than molecular diffusion, and is consequently used frequently in industry and research. The turbulent diffusion acts as a process for quickly reducing concentration gradients, rapid contaminant reduction for safety or rapid mixing and homogenization of fluid mixtures to accelerate chemical reactions.

Understanding the structure in space and time of a turbulent flow, as well as its statistical properties, remains a challenge both for the experimentalist and the theoretician. The statistical properties of turbulent diffusion reveals plenty of information about the characteristics of a flow field and how it will transport affected physical quantities. The study of turbulent diffusion is most often conducted on three-dimensional flows, however, the work conducted here is on two-dimensional flows. In physical systems, a reduction in dimensionality often leads to exciting new phenomena. Full three-dimensional flow can hold two-dimensional characteristics if two spatial directions dominate the extent of the system, such as macroscopic geophysical and astrophysical flows, and thin films in microscopic flows. The physical mechanisms associated with the reduction from three dimensions to two dimensions is discussed in detail by e.g. Kraichnan and Motgomery (1980); Batchelor (1969); Leith (1971); Boffetta and Ecke (2012) and Maulik and San (2017).

The purpose of this thesis is to implement, parallelize and test a direct numerical solver of the equations of motion in two dimensions, as a preparation for the full three-dimensional case in the master thesis. In the following sections an introduction is given to the concept of turbulence to make the reader familiar with how it differs from a smooth and laminar flow field. The second-moment is presented as a statistical metric to describe the distribution of a quantity affected by the turbulent flow field. The equations of motion associated with fluids are presented, followed by methods used to solve them numerically. A section is devoted to the modification of a numerical solver to be run on a high-performance-computer using a message passing interface. Lastly, a set of two-dimensional flow fields, which has been constructed to study the transportation of a physical quantity, is presented, in company with the distributions of the quantity transported with the flows, and some statistical properties related to them. This section is ended by a famous quote, summarizing the core essence of this project:

> ”Big whirls have little whirls, that feed on their velocity
> And little whirls have lesser whirls and so on to viscosity.”
>
> - Lewis Fry Richardson

Chapter 2

Theory

2.1 Turbulence

In fluid dynamics, the motion of a fluid is said to be laminar if the particles in the fluid follows smooth paths in layers, with little to no lateral mixing between adjacent layers. The collective motion of the flow will have the same characteristics as a single particle trajectory following the flow. In fluid dynamics, the flows are described by several parameters were each portrays different aspects of the flow. One such critical parameter is the Reynolds number which outlines the ratio between the inertial and viscous forces of a fluid. It is defined as

$$Re = \frac{ul}{\nu}, \tag{2.1}$$

where u is the velocity of the fluid, l is a characteristic length scale of the flow, typically a geometric limitation or large scale structures in the flow. ν is the kinematic viscosity which is the ratio of a fluids viscosity μ and the density of the fluid ρ. Beyond a certain value for the critical parameter Re, a laminar flow will become unstable and evolve to a new flew regime. This new flow regime is not predicted by the extensive stability theory available for laminar flows, but is inevitable. For a thorough introduction to stability analysis of viscous flow, see White (2006) The flow transitions into an disorderly and unsteady motion in which transported quantities such as mass, momentum and pressure fluctuate in a chaotic manner. This disordered motion is called turbulence. Since the nature of turbulence is so complex, a complete analysis and quantification will most likely be absent for the foreseeable future. Richard Feynman has described turbulence as the most important unsolved problem in classical physics, see Feynman et al. (1963). Even though no general solutions of problems in turbulence exist, it is possible to describe some of the important flow properties:

1. Irregularity: All turbulent flows are irregular or maintain a level of randomness. A deterministic approach to problems with such flows is impossible, which is why statistical methods are extensively used.

2. Diffusivity: An important feature of all turbulent flows is the increased rate of transfer in momentum, heat and mass. The rapid mixing in the turbulent flow field causes an apparent increased

diffusivity of physical quantities. This is also the most important aspect of turbulent flows when considering possible applications. The diffusive nature prevents boundary-layer separation on airfoils at large angles of attack, heat transfer in machinery is increased, it is the main source of resistance of flow in pipelines and it increase the transportation of momentum between ocean currents and winds.

3. High Reynolds numbers: Turbulent flows always occur at high Reynolds number. The turbulent flow field originates from the instability of laminar flows which is related to viscous terms and non-linear inertia terms in the equations of motion presented in the following sections.

4. Three-dimensional: Turbulence is always three-dimensional and rotational. It is characterized by high levels of fluctuating vorticity. This random vorticity fluctuation would not maintain itself in a two-dimensional flow due to the missing vorticity-maintenance mechanism called vortex stretching. In two-dimensional flow fields with high Re, other phenomena occur which are not present in three-dimensional flows.

5. Dissipative: Work done by viscous shear stresses increases the internal energy, i.e. heat of the fluid, at the cost of kinetic energy of the turbulence, hence the turbulent flow field is dissipative. Due to this nature the flow needs a continuous supply of energy to uphold its kinetic energy.

6. Continuum: Turbulence is a continuum phenomena governed by the equations of motion. All length scales of a turbulent flow are far larger than the molecular scales of the flow. This makes it possible to use the Continuum Model which enables the modelling of the flow through partial differential equations (PDE).

Lastly it is important to mention that turbulence is not a feature of a fluid, but of the flow. The dynamics of turbulence is the same in all fluids regardless of it being a gas or a liquid. The major characteristics of the flow are not constrained by molecular properties of the fluid. For further insight into the characteristics of turbulence, see Tennekes and Lumley (1972), and for a summary of recent findings in the field, see Liu and Cai (2017).

2.1.1 Dimensional analysis and energy cascade

Together with statistical methods, a dimensional analysis is one of the most potent methods for studying flows with turbulent nature. In a flow field there exist several scales for time and space and combinations of the two such as velocity. The largest scale, denoted by l, is called the transverse length scale. It represents large scale structures in the flow such as the largest eddies or the width of the boundary layer. Its upper boundary is the geometric dimensions of the flow field. The smallest length scales are denoted by η, also called the Kolmogorov length scale. It represents the smallest eddies in the flow. In three dimensional flows the non-linear terms in the equations of motion breaks down large scale structures into smaller and smaller structures, or eddies. At the smallest possible length scale the viscous effects dominates over the non-linear effects and further energy dissipation is done through heat production. This energy conversion from large scale to small scale is called an energy cascade and is presented in figure 2.1. The inverse energy cascade is seen in two-dimensional flows where the three-dimensional non-linear vortex-stretching terms are absent. The energy is instead transferred from the smallest scales and up to the largest structures. We will return to this point when we discuss the equations of motion governing the flow.

The rate of which this energy is dissipated through the different length scales is called the energy dissipation rate, denoted by ϵ. The small scale structures for both length η, time τ and velocity v is related to physical parameters through the following relations:

$$\eta = \left(\nu^3/\epsilon\right)^{1/4}, \quad \tau = \left(\nu/\epsilon\right)^{1/2}, \quad v = \left(\nu\epsilon\right)^{1/4}, \tag{2.2}$$

and through rigorous dimensional analysis, presented by Tennekes and Lumley (1972); Canuto et al. (1987), we can relate the small scale structures to the large scale structures for both length, time and velocity as follows:

$$\eta/l = \left(\frac{ul}{\nu}\right)^{-3/4}, \quad \tau/t = \left(\frac{ul}{\nu}\right)^{-1/2}, \quad v/u = \left(\frac{ul}{\nu}\right)^{-1/4}. \tag{2.3}$$

l, t and u are the large scales in the flow and is called the integral scales. η, τ and v are the small scale structures and are called the Kolmogorov microscales. It is instructive to mention that $\frac{\eta v}{\nu} = 1$ is the Reynolds number for the small scale flow, meaning that viscous effects is not negligible in these scales.

2.1.2 Turbulent diffusion

According to Richardson (1926) the apparent diffusivity of a three dimensional turbulent flow field scales as

$$K \propto \sigma^{4/3}, \tag{2.4}$$

where K indicates the effective diffusive nature of the turbulent flow and σ is the square root of the second-Central-moment (SCM), defined as

$$\sigma^2 = \int_{-\infty}^{\infty} (x-c)^2 f(x) dx. \tag{2.5}$$

Here, $f(x)$ is a distribution on the x-axis, e.g. some distribution of a physical quantity and c is the mean of the distribution. SCMs are used to study the shape of the distribution $f(x)$ and is commonly known as the variance of the distribution. If the dependency on the mean is neglected, i.e. c is set to zero, the definition in equation (2.5) is referred to as the second-moment (SM). Since the determination of the true value of the turbulent diffusion, K, is hard, it is possible to check the scaling using the relation

$$\sigma^2 \propto t^3, \tag{2.6}$$

where t is the time scale of the problem. The derivation of both scaling laws is presented by Richardson (1926). It is instructive to note that the conventional Fickian diffusion which occurs due to concentration gradients, scales as $\sigma^2 \propto 1$.

2.1.3 Direct Numerical Simulation

The most common approach when working with turbulent flows is to divide the velocity field into a mean velocity U and its fluctuating part u. The exact value of the velocity is $\tilde{u} = U + u$. This is called a Reynolds decomposition. Inserting this decomposed value into the equations of motion and taking the time avarage over all the values will yield a new set of equations where the mean flow and

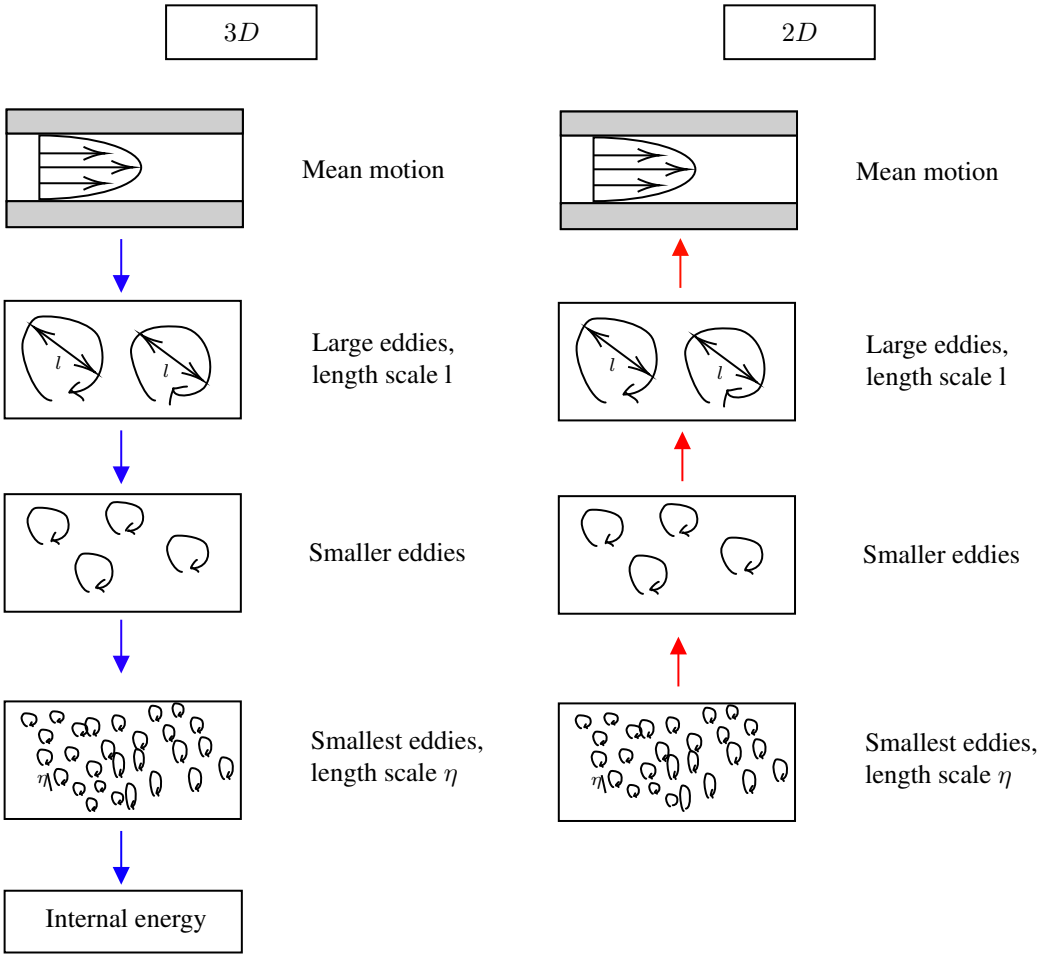


Figure 2.1: The energy cascade presented for both three-dimensional and two-dimensional flows. For three-dimensional flows the energy is dissipated from the large scale structures and down to the smallest scales, indicated by the blue arrow. For two-dimensional flows the energy is transfered from the smallest scales up to the largest scales, indicated by the red arrow.

turbulent fluctuations are separated. The turbulent part can then be modeled using several methods. This approach is called a Reynolds-Averaged Navier-Stokes (RANS) model. Here all turbulent length scales are modeled using e.g. algebraic equations, and a lot of the information in the flow field is lost. Another method is to solve a spatial average of the equations of motion were the largest structures are resolved accurately, but the smallest eddies are modeled as in the RANS model. This method is referred to as the large eddy simulation (LES). The last method is to resolve all possible scales by numerically solving the equations of motion directly. This method is called a direct numerical simulation (DNS). No parameterized modelling of the turbulent parts is required, but the computational cost is high. It is not practical for industrial flow and is not even implemented in commercial software such as Ansys Fluent, see reference Ansys (November, 2019).

The term DNS is restricted to computational fluid dynamics (CFD) simulations where all the length scales ranging from the Kolmogorov microscales to the integral scales, as well as the time scales, are fully resolved. The numerical methods involved need to be of such high order and accuracy that errors related to numerical diffusion and dispersion are negligible compared to the actual physical diffusion and dispersion. DNS solvers are an invaluable research tool in fluid dynamics, and their usage is considered to be comparable to carefully conducted experiments. Since none of the small scales are modeled by e.g. algebraic equations, the amount of computational resources needed to efficiently run a DNS solver is high. To illustrate this, the size smallest scales in a flow with $Re = 1600$ is of order $\mathcal{O}(4 \times 10^{-3})$. The performance of a three-dimensional DNS solver implemented in Python is presented by Mortensen and Langtangen (2016). In the following sections the equations of motion solved in the DNS and the numerical methods used are presented.

2.2 Model equations

2.2.1 Navier-Stokes equations

The equations of motion governing the transportation of mass, momentum and heat in viscous flows are called the Navier-Stokes equations and are named after the scientists Claude-Louis Navier and George Gabriel Stokes. The equations are conservation laws stating that certain physical properties, such as mass, momentum and heat, do not change over time in an isolated system, essentially meaning that the change in one of the terms in the mathematical equations must be accounted for through change in any of the other terms. A complete description of a fluid is available if the velocity field as a function of space and time is known,

$$\boldsymbol{u} = \boldsymbol{u}(t, x, y, z), \tag{2.7}$$

as well as the thermodynamic variables like the density ρ, temperature T and pressure p. The conservation of momentum is presented in this section. For a thorough introduction to the other conservation laws, see e.g. White (2011).

The relation generally known as Newton's second law, states a proportionality between a force $\boldsymbol{F}$ and the resulting acceleration $\boldsymbol{a}$ of a particle with mass m:

$$\boldsymbol{F} = m\boldsymbol{a}. \tag{2.8}$$

For fluid particles the applied forces are acting either on the surface of the particle or its body. Rewriting Newton's second law in terms of these forces, and relating the acceleration to the velocity field **u**, yields

$$\rho \frac{D\mathbf{u}}{Dt} = \boldsymbol{f}_{\text{body}} + \boldsymbol{f}_{\text{surface}}, \tag{2.9}$$

where ρ is the density of the fluid particle and the force terms have units force per unit volume. The operator $\frac{D}{Dt}$ is referred to as the total derivative, or material derivative. It describes the rate of change of a physical quantity that is subject to change in both time and space. If the fluid is assumed to be of constant density, or incompressible, it can be shown that the velocity field is divergence free

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0. \tag{2.10}$$

The equation stating the divergence free velocity field is named the continuity equation and is describing the transport of some quantity, which in this case is mass. In addition, if the body forces are considered to be only due to pressure forces acting on the entire body volume, the Newtons second law takes the form

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \frac{1}{\rho}\boldsymbol{\nabla} \cdot \boldsymbol{\tau}, \tag{2.11}$$

where **u** is the velocity field, ρ is the density, p is the pressure and $\boldsymbol{\tau}$ is the viscous stress tensor. The viscous stress tensor of a Newtonian fluid is defined as

$$\boldsymbol{\tau} = \mu[\boldsymbol{\nabla}\mathbf{u} + (\boldsymbol{\nabla}\mathbf{u})^T] - \frac{2}{3}\mu\boldsymbol{\nabla} \cdot \mathbf{u}\mathbf{I}, \tag{2.12}$$

where μ is the dynamic viscosity and $\mathbf{I}$ is the identity matrix. Assuming the fluid is Newtonian such that the surface forces are viscous stresses, the conservation of momentum of a fluid particle takes the form of the momentum equation,

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \nu \boldsymbol{\nabla}^2 \mathbf{u}, \tag{2.13}$$

where $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity. The first term of (2.13) denotes the temporal change of the solution. The second term is the convection of the flow. The third term is the forces due to pressure, and the fourth term is internal forces due to viscosity.

2.2.2 Vorticity-Stream function formulation

The conventional formulation of the two-dimensional Navier-Stokes equations are two transport equations for the velocity field $\mathbf{u} = [u, v]$, and a Poisson equation for the pressure p. For two-dimensional incompressible flows it is possible to model the fluid flow using only one transport equation and a Poisson equation. This method is called the vorticity-stream function formulation. Due to the reduced complexity this method is commonly used for computations in two-dimensions. However, it is less popular for three-dimensional flows as then the method requires solving six PDEs, as opposed to the four equations needed for the velocity field u, v and w and the pressure p, in the conventional formulation.

The vorticity vector at a point in space is defined as

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \times \mathbf{u} \tag{2.14}$$

which for a two-dimensional flow in the x-y plane reduces to

$$\omega_z = \boldsymbol{\omega} \cdot \hat{k} = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}. \tag{2.15}$$

This vorticity field describes the local spinning motion of a continuum or a fluid in the regions close to the point. Vorticity often originates from shear stresses in the flow caused by solid wall boundaries or local obstructions in the flow. To make the mathematical formulation independent of the velocity fields u and v, it is possible to rewrite them in terms of the derivatives of a scalar function ψ which we name the stream function. This stream function is chosen such that the continuity equation (2.10) is satisfied,

$$\boldsymbol{\nabla} \times \psi \hat{k} = \mathbf{u}. \tag{2.16}$$

The velocity field can be seen satisfying the incompressible flow assumption by substituting the constructed velocity field $\mathbf{u}$ in (2.16) into (2.10), since taking the divergence of the curl of a vector field is always zero. Inserting the same constructed velocity field into equation (2.15) results in a Poisson equation for ψ

$$\boldsymbol{\nabla}^2 \psi = -\boldsymbol{\omega}, \tag{2.17}$$

where $\boldsymbol{\omega}$ acts as a source term. This is an equation connecting the stream function to the vorticity in an incompressible flow. A transport equation for the vorticity can be derived by taking the curl of the velocity component in equation (2.11)

$$\boldsymbol{\nabla} \times \frac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\nabla} \times (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = -\frac{1}{\rho}\boldsymbol{\nabla} \times \boldsymbol{\nabla} p + \nu \boldsymbol{\nabla} \times \boldsymbol{\nabla}^2 \mathbf{u}, \tag{2.18}$$

where using the follow vector identities,

$$(\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} = \frac{1}{2}\boldsymbol{\nabla}(\mathbf{u}\cdot\mathbf{u}) - \mathbf{u}\times(\boldsymbol{\nabla}\times\mathbf{u}) \tag{2.19}$$

$$\boldsymbol{\nabla}\times(\boldsymbol{\nabla}p) = 0, \tag{2.20}$$

reduces equation (2.18) to

$$\frac{\partial\boldsymbol{\omega}}{\partial t} + (\mathbf{u}\cdot\boldsymbol{\nabla})\boldsymbol{\omega} = (\boldsymbol{\omega}\cdot\boldsymbol{\nabla})\mathbf{u} + \nu\boldsymbol{\nabla}^2\boldsymbol{\omega}. \tag{2.21}$$

For two-dimensional flow the first term on the right hand side of equation (2.21), which represents vortex stretching, is zero. See Tennekes and Lumley (1972) for a detailed discussion of this mechanism. Equation (2.21) reduces to

$$\frac{\partial\omega}{\partial t} = -u\frac{\partial\omega}{\partial x} - v\frac{\partial\omega}{\partial y} + \nu\left(\frac{\partial^2\omega}{\partial x^2} + \frac{\partial^2\omega}{\partial y^2}\right), \tag{2.22}$$

which we name the two-dimensional vorticity-transport equation. Solving equation (2.22) using the methods described in chapter 3 yields a vorticity field, which is then used in (2.17) to solve for the stream function ψ, followed by solving (2.16) for the final velocity field. An alternative derivation of the vorticity-transport equation can be done by taking the derivative of the x-component of the NS equations with respect to y, and the derivative of the y-component of NS equations with respect to x. By subtracting the x-component from the y-component and using the continuity equation (2.10) and (2.15) the vorticity-transport equations unveils.

Using this approach the pressure has been eliminated explicitly from the transport equation. The conventional formulation of NS requires solving a Poisson equation for the pressure to make sure the continuity equation is satisfied and the handling of such explicit pressure forces is the most challenging part of the method. In the vorticity-stream function formulation this is taken care of directly by defining the velocity fields in a clever way through the stream function.

The eliminated pressure can be calculated if needed through the vorticity field and the stream function using the relation

$$\boldsymbol{\nabla}^2 p = 2\rho\left[\frac{\partial^2\psi}{\partial x^2}\frac{\partial^2\psi}{\partial y^2} - \left(\frac{\partial^2\psi}{\partial x\partial y}\right)^2\right] \tag{2.23}$$

It is clear that for incompressible flows the pressure has an elliptic behaviour. The spatial and temporal discretization involved in the numerical solution of the two-dimensional vorticity equation is covered in chapter 3.

2.2.3 The Advection-Diffusion equation

The advection-diffusion equation for a conserved quantity S is

$$\frac{\partial S}{\partial t} + \boldsymbol{\nabla}\cdot(\mathbf{u}S) = \boldsymbol{\nabla}(D\boldsymbol{\nabla}S), \tag{2.24}$$

where D is the diffusivity and $\mathbf{u}$ is the advection velocity vector. The advection-diffusion equation models the physical transportation of a conserved quantity inside a system by the effects of diffusion and

advection. The term $\boldsymbol{\nabla} \cdot (\boldsymbol{u}S)$ models advection of S in a flow field $\boldsymbol{u}$, and the term $\boldsymbol{\nabla}(D\boldsymbol{\nabla}S)$ models the diffusion, which is the gradual movement of S due to concentration gradients.

2.3 Spectral methods

In this section the background theory for the spectral method used in the spatial discretization of the vorticity-transport equation is presented. A spectral method can be considered as a part of the discretization scheme known as the method of weighted residuals (MWR). The MWR is a method where the solution of the differential equation is assumed to be well approximated by a finite sum of test functions, ϕ_k. The challenging task is to find the coefficient value, $\hat{u}_k$ of each corresponding test function, such that the sum, $\tilde{u} = \sum_{k=-\infty}^{\infty} \hat{u}_k \phi_k$, approximates the exact solution of the differential equation, u, accurately. The coefficients, $\hat{u}_k$, are referred to as the expansion coefficients or trial functions.

In a conventional MWR the expansion functions are locally defined and are well suited for approximating solutions in complex geometries. The expansion functions in spectral methods are infinitely differentiable and globally defined, convenient for problems with plain geometry and periodic boundaries.

One well known expansion function which inherit the characteristics of a spectral method is the Fourier series. Since the k-th coefficient of the expansion decays faster than any inverse power of k, as long as there are enough coefficients to represent all essential structures of the function, a Fourier expansion will yield spectral accuracy, meaning that its convergence rate is exponential,

$$\text{error} \approx \mathcal{O}\left(\left(\frac{\mathrm{L}}{\mathrm{N}}\right)^{\mathrm{N}}\right), \tag{2.25}$$

where L is the length scale of the computational domain and N is the number of coefficients in the truncated expansion of the approximated solution to the differential equation. For a deeper insight into the MWR, see Canuto et al. (1987). In the following sections the Fourier expansion used in the spectral method is presented.

2.3.1 The continuous Fourier expansion

For a complex valued function u defined in physical space on the interval $[0, 2\pi]$, the Fourier coefficients of u are introduced as

$$\hat{u}_k = \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-ikx} dx \quad k = 0, \pm 1, \pm 2, \ldots. \tag{2.26}$$

This transformation lets us associate a sequence of complex numbers to the function u

$$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx}, \tag{2.27}$$

where the coefficients in this series are recognized as the Fourier transform of u evaluated at integer steps. This expansion of u is named the Fourier series of u. For a finite number of elements in the series, equation (2.27) takes the form

$$u = \sum_{k=-N/2}^{N/2-1} \hat{u}_k e^{ikx}, \tag{2.28}$$

which is named the truncated Fourier series. The function u involved in the continuous Fourier transform can only be defined in a way such that the integral and summations converge. This is ensured if u is a Riemann integrable function, for which two of the features are that u is bounded and piecewise continuous in the interval $[0, 2\pi]$. An overview of relevant mathematical concepts, such as the Riemann integral, is given in the appendix of Canuto et al. (1987).

2.3.2 The discrete Fourier expansion

For practical applications involving numerical methods based on Fourier series there are some difficulties associated with the continuous transformation. The function of interest, u may not always be on a closed form as required for the convergence and hence it must be approximated. There must also be a way to convert the information calculated in transformed or Fourier space over to physical space. Lastly the occurence of nonlinearities will cause complications when expanding such functions using a continuous Fourier expansion. A solution to overcome these obstacles is to use the discrete Fourier transform (DFT) and the discrete Fourier series.

Consider the set of points or nodes

$$x_j = \frac{2\pi j}{N} \qquad j = 0, \ldots, N-1, \tag{2.29}$$

where j is a positive integer. The discrete Fourier coefficients of a complex valued function u in the interval $[0, 2\pi]$ at the points x_j are,

$$\widetilde{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-ikx_j} \qquad -N/2 \leq k \leq N/2 - 1, \tag{2.30}$$

where $u(x_j)$ and $\widetilde{u}_k$ is inversely related through

$$u(x_j) = \sum_{k=-N/2}^{N/2-1} \widetilde{u}_k e^{ikx_j} \qquad j = 0, \ldots, N-1. \tag{2.31}$$

Hence, the DFT is the mapping between the N complex numbers $u(x_j)$, $j = 0, \ldots, N-1$ and the N complex numbers $\widetilde{u}_k$, $k = -N/2, \ldots, N/2 - 1$. Equations (2.30) and (2.31) are referred to as the DFT and the inverse DFT respectively and they show that the transformation is orthogonal on the complex plane $\mathbb{C}$. The transformations can be numerically accomplished by the Fast Fourier Transform (FFT).

2.3.3 Fast Fourier Transform

The FFT is a recursive algorithm for evaluating the transformations (2.30) and (2.31). Its purpose is to achieve computational speedup in the transformations and it is involved in many practical applications e.g. signal processing, by transforming the signal from its original domain to its frequency domain. The DFT is obtained by decomposing a sequence of values into components of different frequencies, which has proven useful in many fields of science, but computing the sum directly is generally too slow for

practical applications. As seen from equation (2.30) and (2.31), the floating point operations involved are a sum over both the spatial index, j, and the wave-number k, resulting in the number of floating point operations scaling as $\mathcal{O}(N^2)$. The FFT algorithm recursively breaks down a DFT of any composite size $N = N_1 N_2$ into several smaller DFTs of size N_1 and N_2, reducing the complexity, and hence the number of floating point operations. A simple FFT algorithm involves $\mathcal{O}(N \log(N))$ floating point operations. A rigorous introduction to the FFT algorithm, as well as an implementation of the algorithm in FORTRAN, can be found in appendix B of Canuto et al. (1987).

2.3.4 Aliasing & De-aliasing

One concern emerging when expanding the function u in terms of discrete coefficients rather than the exact, continuous coefficients of u, is how large the series should be to represent the function correctly. If the continuous expansion converges to the function u at every node, see equation (2.29), then the discrete Fourier coefficients can be expressed in terms of the exact Fourier coefficients of u through

$$\widetilde{u}_k = \hat{u}_k + \sum_{\substack{m=-\infty \\ m\neq 0}}^{+\infty} \hat{u}_k + Nm \qquad k = -N/2, \ldots, N/2 - 1 \tag{2.32}$$

where $\widetilde{u}_k$ are the discrete coefficients and $\hat{u}_k$ are the exact coefficients. Equation (2.32) reveals that the k-th mode of the discrete coefficient not only depends on the k-th mode of the exact coefficient but the modes $(k + Nm)^{th}$ which is referred to as the modes which *alias* the k-th mode. Rephrased, this means that in the DFT, aliasing occur when the size N of the sampled Fourier modes is less than the actual size of the Fourier modes, resulting in a Fourier mode with wave number out of the size range being aliased to a wave number inside the domain, i.e wave numbers $k \geq N/2$ are aliased to wave numbers $k - N$. The approximated solution $\tilde{u}_k$ is essentially disturbed by bad Fourier-modes due to poor sampling rate.

The aliasing error also presents itself when dealing with convolution sums for nonlinear problems, such as in pseudo-spectral methods. Consider solving for the nonlinear variable

$$W_j = U_j V_j \qquad j = 0, \ldots, N - 1 \tag{2.33}$$

where the linear factors are defined as

$$\begin{aligned} U_j &= \sum_{k=-N/2}^{N/2-1} \widetilde{u}_k e^{ikx_j} \\ V_j &= \sum_{k=-N/2}^{N/2-1} \widetilde{v}_k e^{ikx_j}. \end{aligned} \tag{2.34}$$

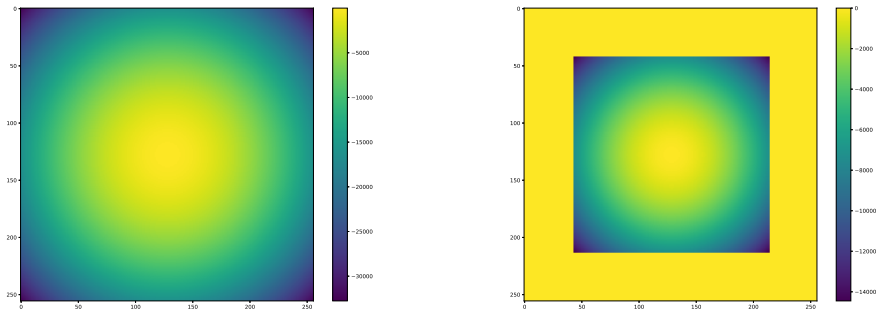
The transformation of W_j is

$$\widetilde{W}_k = \frac{1}{N} \sum_{j=0}^{N-1} W_j e^{-ikx_j} \qquad k = -N/2, \ldots, N/2 - 1, \tag{2.35}$$

which by insertion of the linear factors in equation (2.34) give,

$$\widetilde{W}_k = \sum_{m+n=k} \widetilde{u}_k \widetilde{v}_k + \sum_{m+n=k\pm N} \widetilde{u}_k \widetilde{v}_k \tag{2.36}$$

where the first term is the truncated Fourier expansion of W_j and the latter term is the alias error. The error due to this alias-dependence can be shown to be larger than the error due to the truncation of the Fourier series and consequently must be dealt with. It is shown by Canuto et al. (1987) that the error due to aliasing can be removed by by the method referred to as truncation, or the 2/3- rule. The idea is to choose the mesh size (or sampling rate to use the same phraseology as before) to be of M points rather than N, where the they are related as $M \geq 3N/2$. It is then possible to choose M such that the extra terms in equation (2.32) and (2.36) related to the aliasing-error, is neglected. In practice, this is done by choosing a mesh size N of a specific size, and remove frequencies in the spectral mesh k that are larger than $2/3$s of the highest frequency $N/2+1$. The 2/3-rule is presented in figure 2.2 for the variable $\boldsymbol{k}^2 = k_x^2 + k_y^2$, where k_x and k_y are wave-number vectors presented in the next section.



(a) Wave-number mesh $\boldsymbol{k}^2$ before de-aliasing.

(b) Wave-number mesh $\boldsymbol{k}^2$ after de-aliasing.

Figure 2.2: Here the wave-number mesh, $\boldsymbol{k}^2$, has its highest frequencies truncated by an outline of zeros as a part of the 2/3-rule used to de-alias the approximated solution to the differential equation being studied.

Chapter 3

Numerical methods

In this chapter the numerical methods involved in the spatial discretization and the temporal discretization of the vorticity-transport equation and the advection-diffusion equation is presented. A procedural overview of the numerical method is given as well as an introduction to the message passing interface, enabling communication between computer cores.

3.1 Spatial discretization

3.1.1 Pseudo-Spectral Fourier-Galerkin method for the Vorticity transport equation

In this section the pseudo-spectral Fourier-Galerkin method, used to solve the spatial part of the Navier-Stokes equation on vorticity form, is presented. The implementation of the numerical solver of this equation will sometimes be referred to as the DNS solver. The vorticity-transport equation is as presented in equation (2.22),

$$\frac{\partial \omega}{\partial t} = -u\frac{\partial \omega}{\partial x} - v\frac{\partial \omega}{\partial y} + \nu\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right), \tag{3.1}$$

where the variables involved are periodic in both x- and y-direction. They are discretized on an equidistant and structured computational mesh with N points in both directions such that the physical node points can be represented as

$$\boldsymbol{x} = (x, y) = \left\{\left(\frac{2\pi i}{N}, \frac{2\pi j}{N}\right) : \qquad i, j \in 0, \ldots, N-1\right\}. \tag{3.2}$$

When discretizing the spatial part of the differential equation using the pseudo-spectal Fourier-Galerkin method, all of the variables will be transformed from physical space $\boldsymbol{x}$ to a wave number mesh in Fourier space or spectral space. This wave-number mesh is represented as

$$\boldsymbol{k} = (k_x, k_y) = \{(l, m) : l, m \in -N/2 + 1, \ldots, N/2\} \tag{3.3}$$

To transform between physical space $\boldsymbol{u}$ and spectral space $\boldsymbol{k}$, the discrete Fourier transformations presented in section 2.3.2 is used on both the vorticity ω and the velocity u, and v in x- and y-direction respectively. Effectively the method involves taking the inner product of the Navier-Stokes equations on vorticity form with the basis-functions $e^{i\boldsymbol{kx}}$ which is the same as taking the Fourier transformation in both spatial dimensions. The use of exactly this basis function takes care of the periodic boundary condition with no extra attention needed. The PDE now takes the form of a system of ordinary differential equations for ω_k

$$\frac{d\hat{\omega}_k}{dt} = -\widehat{\left(u\frac{\partial\omega}{\partial x}\right)} - \widehat{\left(v\frac{\partial\omega}{\partial y}\right)} - \nu\left[k_x^2 + k_y^2\right]\hat{\omega}_k, \tag{3.4}$$

where $\hat{\omega}_k$ is the vorticity in Fourier space and k_x and k_y are the wave number meshes. The two first terms on the right side of equation (3.4) are the nonlinear convective terms in Fourier space. The hat notation indicates the the term is to be evaluated in Fourier space which involves computing a nonlinear convolution sum. In the pseudo-spectral method, this term is evaluated in physical space by transforming only the convective terms through the inverse Fourier transformation. After succesfully computing the vorticity in Fourier space, the velocity in Fourier space is reconstructed through the relations (2.17) and (2.16) which yields

$$\hat{u}_k = \frac{ik_y}{|k_x^2 + k_y^2|}\hat{\omega}_k \tag{3.5}$$

$$\hat{v}_k = \frac{ik_x}{|k_x^2 + k_y^2|}\hat{\omega}_k, \tag{3.6}$$

which gives the velocity field $\boldsymbol{u}$ in real space through the inverse transformation of $\hat{u}_k$ and $\hat{v}_k$. The use of periodic boundary conditions in both spatial dimensions essentially maps the solution of the PDE onto a torus. This eliminates the need for resolving boundary layers close to walls by assessing the y^+ values in the cells neighbouring the walls, which is necessary when using turbulence models such as the RANS model. See Pletcher et al. (2013) details. for The spatial resolution is thus only dependent on the Kolmogorov microscales as presented in section 2.1.3.

It is important to note that equation (3.4) is only semi-discretized and the temporal discretization remains to be presented in section 3.2. Since the problem now is an initial value problem, the solution of the vorticity ω, and hence the velocity field $\mathbf{u}$, is highly dependent on the initial conditions (IC). The choice of IC is not completely arbitrary. The initial velocity field must be divergence free as the original form of the Navier-Stokes equation in equation in (2.11) was defined to be incompressible. Otherwise the continuous initial value problem itself fails to have a classical solution. Even though the IC is divergence free, a solution on a given time t is not guaranteed as the time integrator chosen to solve the transient term in (3.4) may be a poor match given the set of parameters involving time step, mesh size and viscosity.

3.1.2 Finite Volume Method for the Advection-Diffusion equation

In this section the finite volume method (FVM) for a Cartesian grid will be applied to discretize the advection-diffusion equation. Since the 1970s the FVM approach has become the industrial standard for spatial discretization in computational fluid dynamics (CFD). The method's success is mainly a consequence of its properties:

- discrete conservation property,
- integral formulation in physical space,
- shock capturing capability,
- local grid adaptation capability,
- geometric flexibility,

where especially the conservation property is important for discretized transport equations like the the advection-diffusion equation. This property is ensured by letting a quantity's flux out from a cell be the flux into the neighbouring cell. Compared to the finite difference method (FDM) and the finite element method (FEM) the numerical analysis for the FVM concerning stability and accuracy is less advanced, and the extension to higher orders of accuracy is not as trivial as for other methods. However, schemes of lower order and regular mesh size shares the same stability features and ease of implementation on a computer as the FDM. A range of different FVM methods for different PDEs, as well as their stability analysis, is presented by Pletcher et al. (2013).

In the finite volume method the integral form of the advection-diffusion equation is discretized on a volume V with boundary ∂V,

$$\int_V \frac{\partial S}{\partial t} dV = \int_{\partial V} \boldsymbol{f_d} \cdot \boldsymbol{n} dA - \int_{\partial V} \boldsymbol{f_a} \cdot \boldsymbol{n} dA, \tag{3.7}$$

where $\boldsymbol{f_d} = (D\boldsymbol{\nabla} S)$ is the diffusive flux and $\boldsymbol{f_a} = (\boldsymbol{u}S)$ is the advective flux over the boundary ∂V with normal vector $\boldsymbol{n}$. Moreover, the volume V is divided into finite volumes, sometimes referred to as control volumes or cells, with a spatial extension of Δx and Δy in the x- and y-direction respectively. In each of these smaller volumes, the cell average is defined to be,

$$\hat{S}(t) = \frac{1}{\Delta x \Delta y} \int_V S(x, y, t) dx dy. \tag{3.8}$$

which inserted into equation (3.7) yields,

$$\Delta x \Delta y \frac{d\hat{S}_{i,j}(t)}{dt} = \sum_{l=e}^{s} \int_{\Gamma_l} \boldsymbol{f_d} \cdot \boldsymbol{n} dA - \sum_{l=e}^{s} \int_{\Gamma_l} \boldsymbol{f_a} \cdot \boldsymbol{n} dA, \tag{3.9}$$

The index notation i, j denotes the cell location on the Cartesian grid with equidistant spacing in both x- and y-direction, i.e. $\Delta x_i = \Delta y_i = \Delta x = \Delta y$ for all $i, j = 1, \ldots, N$. The cell boundaries Γ are indicated in figure 3.1. The hat notation on the conserved quantity $\hat{S}$ is omitted in the flux terms and in the rest of this thesis. The flux integrals are evaluated over the cell boundaries by the following relations:

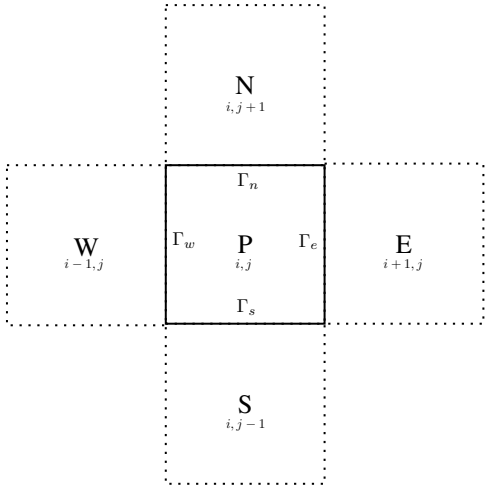


Figure 3.1: Equidistant cell in the finite volume method. Γ_l where $l = w, n, e, s$ denotes the faces of cell P neighbouring the cells W, N, E and S.

$$\int_{\Gamma_w} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = -\int_{y_{j-1/2}}^{y_{j+1/2}} f_1(x_{i-\frac{1}{2}}, y, t) dy \approx -f_{1,i-\frac{1}{2},j} \Delta y \tag{3.10}$$

$$\int_{\Gamma_n} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = \int_{x_{j-1/2}}^{x_{j+1/2}} f_2(x, y_{j+\frac{1}{2}}, t) dx \approx f_{2,i,j+\frac{1}{2}} \Delta x \tag{3.11}$$

$$\int_{\Gamma_e} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = \int_{y_{j+1/2}}^{y_{j+1/2}} f_1(x_{i+\frac{1}{2}}, y, t) dy \approx f_{1,i+\frac{1}{2},j} \Delta y \tag{3.12}$$

$$\int_{\Gamma_s} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = -\int_{x_{j-1/2}}^{x_{j+1/2}} f_2(x, y_{j-\frac{1}{2}}, t) dx \approx -f_{2,i,j-\frac{1}{2}} \Delta x \tag{3.13}$$

where the fluxes in the middle term of all equations are approximated by numerical flux functions, $f_{i\pm\frac{1}{2},j}$ and $f_{i,j\pm\frac{1}{2}}$. These numerical flux functions depend on the approximations $S_{i,j}$ of adjacent cell averages, and can thus be approximated by e.g. a forward difference scheme resulting in the following discretized flux functions:

$$f_{1,i+\frac{1}{2},j} = f_1(S_{i+1,j}, S_{i,j}) = D\frac{S_{i+1,j} - S_{i,j}}{\Delta x} \tag{3.14}$$

$$f_{2,i-\frac{1}{2},j} = f_2(S_{i,j}, S_{i-1,j}) = D\frac{S_{i,j} - S_{i-1,j}}{\Delta y} \tag{3.15}$$

$$f_{1,i,j+\frac{1}{2}} = f_1(S_{i,j+1}, S_{i,j}) = D\frac{S_{i,j+1} - S_{i,j}}{\Delta x} \tag{3.16}$$

$$f_{2,i,j-\frac{1}{2}} = f_2(S_{i,j}, S_{i,j-1}) = D\frac{S_{i,j} - S_{i,j-1}}{\Delta y}. \tag{3.17}$$

The same control volume approach can be done on the advective terms in (3.9). By rearranging the terms we get the semi-discretized advection-diffusion equation,

$$\begin{aligned}\frac{dS_{i,j}(t)}{dt} = & \frac{D}{\Delta x^2}\left(S_{i+1,j} - 2S_{i,j} + S_{i-1,j}\right) \\ & + \frac{D}{\Delta y^2}\left(S_{i,j+1} - 2S_{i,j} + S_{i,j-1}\right) \\ & - \frac{u_{i,j}}{2}(S_{i+1,j} - S_{i-1,j}) \\ & - \frac{v_{i,j}}{2}(S_{i,j+1} - S_{i,j-1}).\end{aligned} \tag{3.18}$$

Since the grid used here is equidistant in both spatial dimensions the finite volume method and finite difference method are identical in implementation. The FDM approach is recognized in (3.18) by approximating both the diffusive and advective term by central difference schemes which are $\mathcal{O}(\Delta^2)$ accurate in space. Schemes of higher order accuracy will cause the solution to oscillate unless special precaution is taken on the numerical parameters such as $\boldsymbol{u}$, Δx, Δy, D and Δt, where Δt is presented in the next section. In a transport equation like the advection-diffusion equation, the diffusion process affects the distribution of a transported quantity along its gradients in all directions, whereas advection spreads influence only in the flow direction. This crucial difference manifests itself in a strict upper limit to the cell size, that is dependent on the relative contribution of advection and diffusion, for stable advection-diffusion calculations with central differencing. This relative contribution is known as the Péclet number and to achieve a solution free from oscillations and negative values we have the requirement,

$$Pe = \frac{\boldsymbol{u}}{ND} \leq 2, \tag{3.19}$$

where $\boldsymbol{u}$ is the same velocity vector as in (3.7), D is the diffusion constant and N is the number of cells in each spatial direction. The oscillations in the solution can be managed by including flux-limiters in the FVM approach. An extensive discussion on non-oscillatory methods is given by Hundsdorfer and Verwer (2003)

3.2 Temporal discretization

In this section the discretization of the temporal term for both the vorticity-transport equation and the advection-diffusion equation is presented. The two equations have been semi-discretized and presented as a system of ordinary differential equations (ODE) where time is the independent variable and either the vorticity ω or the concentration distribution S is the dependent variable. The equations take the form

$$\frac{d\Phi(t)}{dt} = \Psi(t, \boldsymbol{x}) \tag{3.20}$$

where $\Phi(t)$ is either ω or $\boldsymbol{u}$. $\Psi(\boldsymbol{x})$ is the spatially discretized right hand side of either equation (3.4) or equation (3.18). When discretizing the ODE, an explicit method is chosen over an implicit method. Implicit methods involves solving a system of equations in each time step, and the fine grid needed to resolve the smallest eddies in the DNS solver causes the size of the system of equation to be of order $\mathcal{O}(N)$. Since iterative methods are used to solve such problems, instead of direct methods, the computational

effort is higher. For an introduction to iterative methods, see Pletcher et al. (2013). The unconditional stability that follows an implicit method is considered not to be worth the extra computations involved for this specific problem.

There is a range of different methods to solve an ODE, such as Runge-Kutta methods or linear multistep methods. Some of them hold properties like stiffness detection or conservation of the positivity of the solution. An overview of methods used to discretize time-dependent terms in ODEs in spectral space is given by Canuto et al. (1987), and for advection-diffusion-reaction equations in Hundsdorfer and Verwer (2003) The methods used to discretize the temporal part of the vorticity-transport equation and the advection-diffusion equation is presented in the following sections.

3.2.1 Semi-Implicit Crank-Nicolson method

The time integrator used to discretize the temporal term in equation (3.4) is the semi-implicit Crank-Nicolson scheme. It combines the averaged contribution from a forward Euler and backward Euler scheme into an implicit scheme for the linear terms and explicit for the nonlinear terms. The term semi-implicit is commonly given to numerical schemes whose convective terms are discretized explicitly in time and the diffusive terms are discretized implicitly in time. This combination increases the stability of the method without increasing the computational cost as much as a fully implicit scheme would do. The scheme takes the form

$$\frac{\hat{\omega}_k^{n+1} - \hat{\omega}_k^n}{\Delta t} = \frac{1}{2}\left[L^{n+1}(\hat{\omega}_k^{n+1}) + L^n(\hat{\omega}_k^n)\right] + NL^n(\hat{\omega}_k^n) \tag{3.21}$$

where Δt is the time step between the values t^n and t^{n+1}, $L^{n+1}(\hat{\omega}_k^{n+1})$ denotes the linear term, which is the viscous term in the vorticity transport equation, evaluated at the time level $n+1$. $L^n(\hat{\omega}_k^n)$ is the linear term evaluated at time level n, and $NL^n(\hat{\omega}_k^n)$ are the nonlinear convective terms evaluated at time level n. After some rigorous algebra the semi-implicit method takes the explicit form

$$\hat{\omega}_k^{n+1} = \frac{1}{\frac{1}{\Delta t} - \frac{1}{2}\nu(k_x^2 + k_y^2)}\left(\left(\frac{1}{\Delta t} + \frac{1}{2}\nu(k_x^2 + k_y^2)\right)\hat{\omega}_k^n - \left(\widehat{u^n \frac{\partial \omega^n}{\partial x}}\right) - \left(\widehat{v^n \frac{\partial \omega^n}{\partial y}}\right)\right). \tag{3.22}$$

While the fully implicit Crank-Nicolson scheme is second order accurate in time, the semi-implicit is only first order accurate.

3.2.2 Explicit Euler Method

The time integrator used to discretize the temporal term in the semi-discretized advection-diffusion equation (3.18) is the explicit Euler method. This method is the simplest explicit numerical integrator for an ordinary differential equation, and it often serves as a basis to construct methods with higher complexity and accuracy. It involves evaluating the time gradient using the current time value and the value which is a time step Δt in the future. The fully discretized scheme for the advection-diffusion equation takes the form

$$\begin{aligned}\frac{S_{i,j}^{n+1}-S_{i,j}^{n}}{\Delta t} = & \frac{D}{\Delta x^2}\left(S_{i+1,j}^{n}-2S_{i,j}^{n}+S_{i-1,j}^{n}\right)\\ & +\frac{D}{\Delta y^2}\left(S_{i,j+1}^{n}-2S_{i,j}^{n}+S_{i,j-1}^{n}\right)\\ & -\frac{u_{i,j}^{n}}{2}(S_{i+1,j}^{n}-S_{i-1,j}^{n})\\ & -\frac{v_{i,j}^{n}}{2}(S_{i,j+1}^{n}-S_{i,j-1}^{n}).\end{aligned} \tag{3.23}$$

This scheme is first order accurate in time and second order accurate in space.

3.3 Procedural implementation

In this section the procedure to achieve a solution for Navier-Stokes equation on vorticity form is presented in sequential order. It will serve as a summary for the numerical part of this chapter and as a small overview of the Python code used to implement the numerical solvers.

3.3.1 DNS solver

1. Declare systems constants:
   - $\mathrm{t_{final}}$=100 s
   - Mesh size, $N = 256$,
   - Length scale of computational domain, $L = 2\pi$.
   - Cell size, $dx = L/N = 0.0245$.
   - Kinematic viscosity, $\nu = 5 \times 10^{-3}$
2. Set initial conditions of the vorticity, ω, in physical space $\boldsymbol{x}$ as presented in figure 4.1(a).
3. Divide physical mesh $\boldsymbol{x}$ and spectral mesh, $\boldsymbol{k}$, amongst the available cores. Communication between cores takes place in the Fourier transformations.
4. Pre-allocate a range of empty arrays used in the computation to speed up the code and optimize memory usage.
5. Start computing the first time step:
   (a) Read in the vorticity, ω in physical space, and convert to spectral space through the FFT algorithms available in the Numpy package in Python.
   (b) Compute the velocity vectors in spectral space, $\hat{u}_k$ and $\hat{v}_k$, using equations (3.5) and (3.6).
   (c) Transform the spectral velocities back to physical space to get u and v.
   (d) compute the derivative of vorticity with respect to each spatial direction in spectral space using $\widehat{\frac{\partial \omega}{\partial x}} = ik_x\hat{\omega}_k$ and $\widehat{\frac{\partial \omega}{\partial y}} = ik_y\hat{\omega}_k$.

(e) Transform the derivative of the vorticity from Fourier space back to physical space. Since the non-linear terms are computed in physical space, the method is referred to as a pseudo-spectral method.

(f) Multiply the computed velocity with the computed derivative of vorticity with respect to space as done in the non-linear terms of equation (3.4). Transform the computed non-linear term back to Fourier space.

(g) Compute the viscous term directly in Fourier space as done in equation (3.4).

(h) Combine the computed non-linear terms and the viscous terms in spectral space to the right hand side of an ODE, and solve it using the semi-implicit Crank-Nicolson method as presented in section 3.2.1.

(i) Update the old time step. Store velocity fields if needed for future computations.

(j) Start new time step.

3.3.2 Advection-diffusion equation

1. Declare system constants:
   - $\mathrm{t_{final}}$=100 s
   - Mesh size, $N = 256$,
   - Length scale of computational domain, $L = 2\pi$.
   - Cell size, $dx = L/N = 0.0245$.
   - Diffusion constant, $D = 0.0008$
2. Load in the first chunk of the velocity field, e.g. the first 1/100 of the stored time steps.
3. Set initial conditions of the advection-diffusion equation as presented in figure 4.1(d).
4. Distribute the physical mesh $\boldsymbol{x}$ and the velocity fields u and v amongst the cores as presented in figure 3.2.
5. Start computing the first time step:
   (a) Exchange information at field boundaries using MPI communication.
   (b) Do one iteration of the central-space discretized algorithm as presented in equation (3.23).
   (c) Update the old time step. Store the solution of the advection-diffusion equation if needed for future computations.
   (d) Start new time step and exchange information across local field boundaries using MPI communication.
6. If the end of the loaded chunk of the velocity field is reached, load in the next chunk and distribute to the available processors.

3.3.3 Initial conditions

The DNS solver is executed two times using two different sets of IC. The first set of ICs used for the vorticity-transport equations is a collection of Gaussian distributions located in a non-orderly pattern in the computational domain. They hold values of different size and sign and is expected to evolve in a chaotic and random manner, as expected from a turbulent flow. The mathematical expression is

$$\begin{aligned}\omega(x,y,0) = \quad & e^{(-((2*x-\pi+\pi/5)**2+(4*y-\pi+\pi/5)**2)/0.3)} \\ & - e^{(-((2*x-\pi-\pi/5)**2+(3*y-\pi+\pi/5)**2)/0.2)} \\ & + e^{(-((x+\pi-\pi/5)**2+(2*y-\pi-\pi/5)**2)/0.4)} \\ & + e^{(-((2*x-\pi+\pi/5)**2+(y-\pi+\pi/5)**2)/0.3)} \\ & - e^{(-((x-\pi-\pi/5)**2+(y-\pi+\pi/5)**2)/0.2)} \\ & + e^{(-((x-\pi-\pi/5)**2+(3*y-\pi-\pi/5)**2)/0.4)} \\ & + e^{(-((x-\pi+\pi/5)**2+(y-\pi+\pi/5)**2)/0.3)} \\ & + e^{(-((x-\pi-\pi/5)**2+(3*y-\pi+\pi/5)**2)/0.3)} \\ & + e^{(-((x+\pi-\pi/5)**2+(y+\pi-\pi/5)**2)/0.4)} \\ & - e^{(-((x-\pi+\pi/5)**2+(y-\pi+\pi/5)**2)/0.3)} \\ & + e^{(-((2*x-\pi-\pi/5)**2+(3*y-\pi+\pi/5)**2)/0.2)} \\ & + e^{(-((x-\pi-\pi/5)**2+(y-\pi-\pi/5)**2)/0.4)}.\end{aligned} \tag{3.24}$$

The second set of initial conditions for the DNS solver is relatively similar except from some small changes to the numerical values in equation (3.24). The mathematical expression for the second set of ICs for the DNS solver is thus omitted for brevity. The resulting collection of vortices can be seen in figure 4.1(a) for the first set of IC and in figure 4.2(a) for the second set of IC.

The initial condition, $S(x,y,0)$, for the advection-diffusion equation is similar, but considerably shorter. It is the combination of a Gaussian distribution projected on the x-axis with a mean of $\mu = 2$ and a Gaussian distribution projected along the y-axis with a mean $\mu = 3$. The covariance matrix relating the two distributions is

$$\begin{bmatrix} 0.05 & 0.10 \\ 0.01 & 0.05 \end{bmatrix} \tag{3.25}$$

The resulting multivariate distribution can be seen in figure 4.1(d).

3.4 Message Passing Interface

There is a large number of floating point operations involved in the numerical procedures presented in the earlier chapters as well as a huge amount of data involved if every time step is to be stored for future calculations. For instance, the relatively low resolved velocity fields with a cell count of $N = 256$ and a time step of $dt = 0.001\,\mathrm{s}$ and a time scale of $t = 100\,\mathrm{s}$ yields $256 \times 256 \times 100 \times \frac{1}{0.001} = 6.5536 * 10^9$ different variables. Since all the different variables are stored in a Numpy array with datatype *float64* object giving double precision, the number of gigabytes (GB) required to store one of these velocity fields

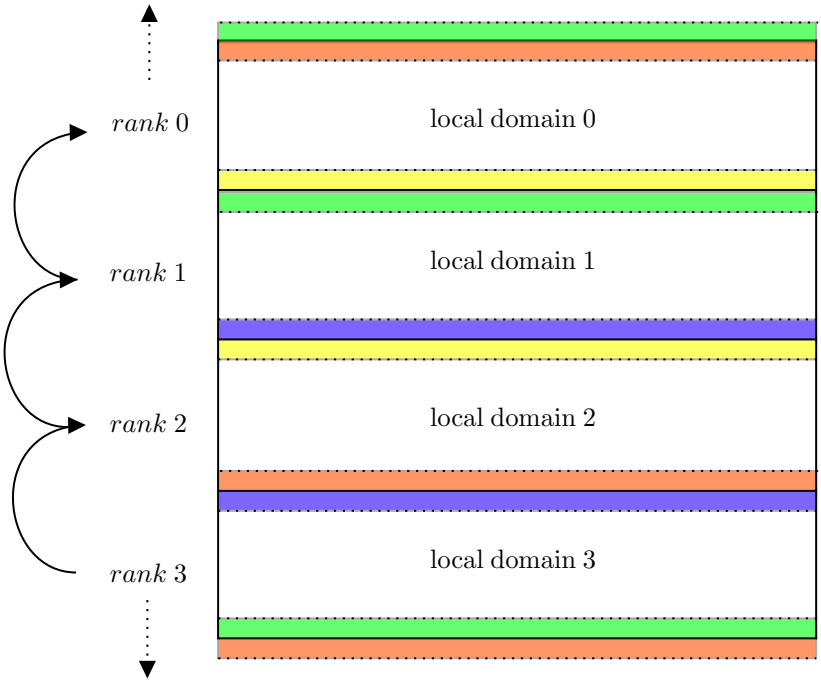


Figure 3.2: A computational domain for the advection-diffusion equation divided amongst four cores. The local domain is the partial computational domain were each process computes a part of the solution. The information needed at the boundaries of each local domain is sent through MPI communication using ghost boundaries indicated by color. This computational domain is periodic in both spatial directions, which means that there is communication between rank 0 and rank 3 as well.

is $6.5536 \times 10^9 \times 64 \times 1.25 \times 10^{-10} = 52\,\mathrm{GB}$. And since the velocity field in both x- and y-direction are needed, a total of $105\,\mathrm{GB}$ is required to store all the variables given the presented parameters. This, together with the number of floating point operations, suggest that the time spent optimizing the code and make it eligible to run on a super computer utilizing multiple cores will be beneficial.

The idea behind parallelization or running a program on multiple cores, commonly named ranks, is to break down a large problem, usually solved by one core in a serial manner, to pieces of smaller problems that can be independently solved at the same time by distributing the workload on the different cores of a computer. The main advantage of a parallel program is that it scales with problem size, meaning that with more computational resources, larger problems can be solved. This is especially convenient for DNS solvers with the heavy number crunching involved.

The method used to distribute the workload amongst the computers core the through a Message Passing Interface (MPI) library. This method is applicable to systems with a distributed memory, which are characterized by a collection of core-memory pairs connected by a network. The memory associated with one core is only directly accessible by that core, and to share the data located in each core's memory with other cores, communication through MPI is required. The package *mpi4py* in Python contains all the tools necessary to implement MPI communication for all cores. For a overview of the documentation of *mpi4py*, see Dalcin et al. (2005, 2008, 2011).

The grid used in the advection-diffusion equation is divided into equally large portions between the processes (as seen in figure 3.2), which means that the number of grid points N must be divisible by the number of assigned cores. Each portion is named a local domain. The local domains have a ghost boundary located in in the first and last row, which is used to store the information received from the neighbouring local domains. Since the numerical stencil used in the advection-diffusion equation only depends on the first neighbouring grid points in each direction, the ghost boundary needs to have the dimensions (1xN). If the stencil were to use both the first and second neighbouring points in both directions, e.g. from a higher-order spatial discretization, the ghost boundary must include a second row, (2xN), to store the extra information needed. The distribution amongst the different ranks is presented in figure 3.2. The ghost boundaries of each local domain are presented in different colors. Rows shown in the same color send information to the same rank, i.e. the blue rows containing information from rank 1 and 3, both send their information to rank 2.

The distribution of data in the DNS solver is done in a similar manner, however there is no need for ghost boundaries. The physical grid is distributed in the same direction as for the advection-diffusion equation, i.e. continuous in the column-direction and distributed in the row-direction. The distribution of the spectral wave-number mesh is oriented $90\,\mathrm{deg}$ with respect to the physical mesh, i.e. continuous distribution in the row-direction and distributed in the column-direction. The need for ghost boundaries is neglected as whenever communications takes place between the cores, all information is sent from all to all cores through the function *Alltoall* in the *mpi4py* package. The communication is only needed when doing the Fourier transformations using the FFT algorithm. The specific method to distribute the two-dimensional array of data described here is called the 1D slab-decomposition. This is easy to implement, but one drawback is that the number of cores must be equal to or smaller than the number of grid points in each direction, i.e nr. of cores $\leq N$. The limitations due to this choice of mesh decomposition is considered to be small when working in two dimensions. For a three dimensional problem the overhead related to communication between cores would be relatively smaller than for the two-dimensional problem since the amount of floating point operations increase drastically. Then a 2D slab-decomposition would be convenient. For a detailed discussion on parallel computing and the relevant hardware of high-performance-computers, see Pacheco (2011).

Chapter 4

Results

In the following sections the numerical solution from the DNS solver with two different IC's is presented alongside the concentration distribution S of an unspecified physical property using the two resolved velocity fields in both x- and y-direction. The solutions are presented at different time steps to show the temporal evolution of the solution. The SM of the function S about both the middle of the computational domain as well as the location of the mean-value of the distribution is presented. Lastly the performance of the Python code of the DNS solver on the IDUN cluster is presented.

4.1 Vorticity field and concentration distribution

The simulation using the first set of initial conditions for the vorticity-transport equation is presented in figure 4.1. The first two rows present snapshot pairs of both the vorticity field ω and the concentration distribution S at the times $t = 0, 9$ and 27 seconds, and the last two rows present snapshot pairs at the times $t = 54, 63$ and 100 seconds. The time $t = 0\,\mathrm{s}$ indicates the initial condition for both fields. The highest value of both solutions at all time steps is set to be the highest level on the colorbar, meaning that the colorbar is scaling each time step to fit with the new data. The simulation using the second initial condition for the vorticity-transport equation is presented in figure 4.2. The time steps are the same as for the first simulation.

A series of animated solutions are presented on the writers GitHub profile, available on `https://github.com/danielhalvorsen/Project_Turbulence_Modelling`. The animations presents the rapid changes and the scales of velocity better than static figures can, and will serve as a supplement to the results.

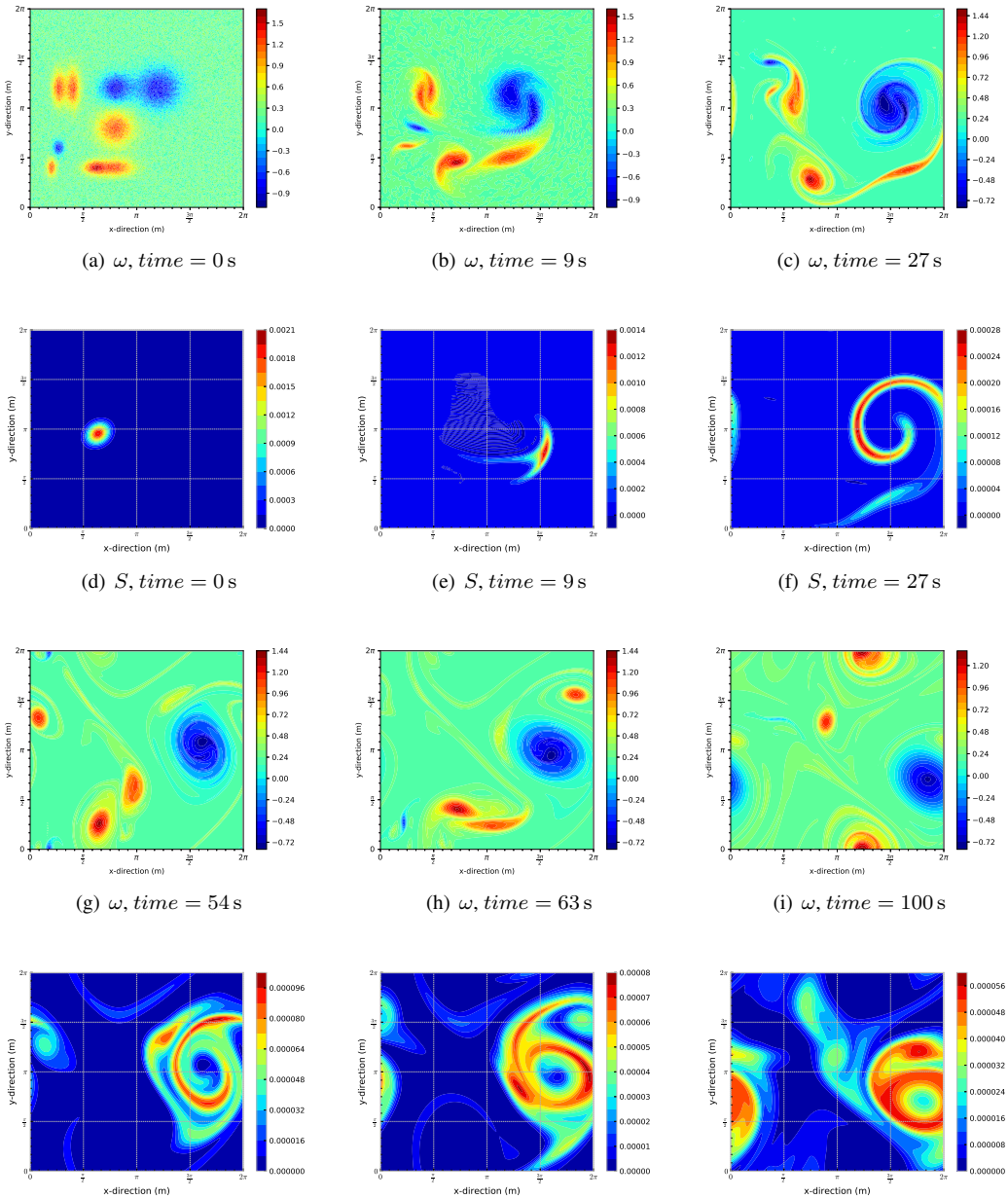
4.2 Second-moments

The scaling of the SMs and SCMs with respect to time are presented in figure 4.3 for the first simulation and figure 4.4 for the second simulation. This scaling with time will in the remainder of this thesis be referred to as the SM or SCM respectively. In both figures the first row presents the statistics for the concentration distribution projected along the x-axis, and the second row presents the statistics of the

distribution projected along the y-axis. The first column presents the SM data, and second column the presents the SCM data.

4.3 Scaling of MPI communication

The performance of the DNS code is tested on the IDUN cluster managed by the High Performance Computing (HPC) group at NTNU. The code has been run multiple times on up to 64 cores, and the wall-time or the elapsed time spent executing the program, is measured. The data is presented in figure 4.5. The parameters used in the DNS solver are the same as for the other simulations, i.e. $N = 256$, $dt =$, $\nu = 5 \times 10^{-4}$ and $t = 100\,\mathrm{s}$. The measured wall-time is decreasing as the core count increases, and through the *powerlaw* package in Python the scaling factor is found to be 1.71, meaning that a doubling in core count reduces the wall-time by a factor of 1.71.



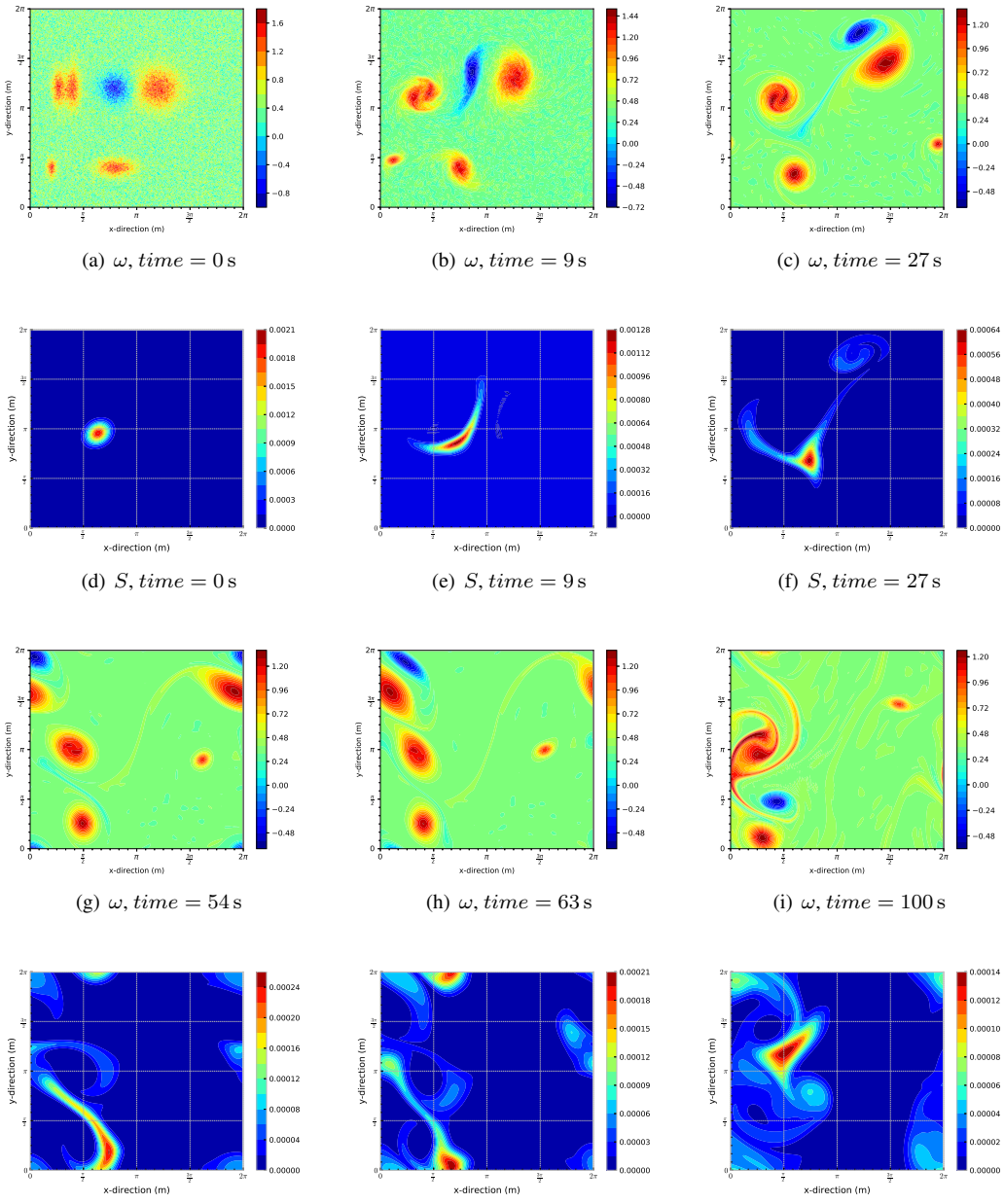
(a) $\omega, time = 0\,\mathrm{s}$ (b) $\omega, time = 9\,\mathrm{s}$ (c) $\omega, time = 27\,\mathrm{s}$

(d) $S, time = 0\,\mathrm{s}$ (e) $S, time = 9\,\mathrm{s}$ (f) $S, time = 27\,\mathrm{s}$

(g) $\omega, time = 54\,\mathrm{s}$ (h) $\omega, time = 63\,\mathrm{s}$ (i) $\omega, time = 100\,\mathrm{s}$

(j) $S, time = 54\,\mathrm{s}$ (k) $S, time = 63\,\mathrm{s}$ (l) $S, time = 100\,\mathrm{s}$

Figure 4.1: The vorticity ω and the concentration distribution of the function S is presented at the time steps $t = 0, 9, 27, 54, 63, 100$ seconds for the first IC of the vorticity field.



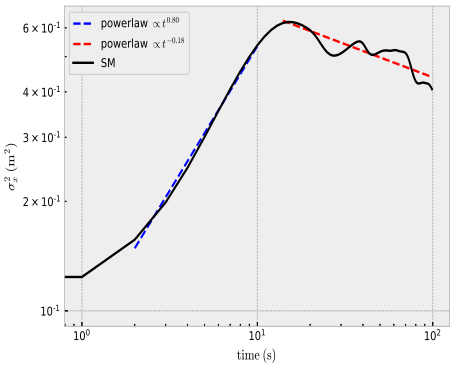
(a) $\omega, time = 0\,\mathrm{s}$ (b) $\omega, time = 9\,\mathrm{s}$ (c) $\omega, time = 27\,\mathrm{s}$

(d) $S, time = 0\,\mathrm{s}$ (e) $S, time = 9\,\mathrm{s}$ (f) $S, time = 27\,\mathrm{s}$

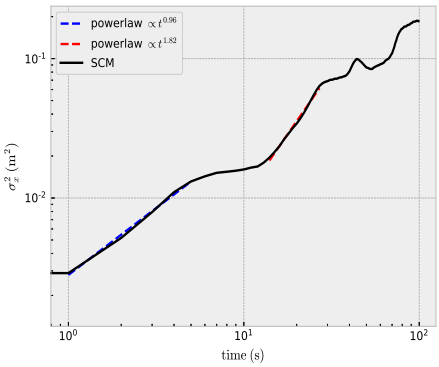
(g) $\omega, time = 54\,\mathrm{s}$ (h) $\omega, time = 63\,\mathrm{s}$ (i) $\omega, time = 100\,\mathrm{s}$

(j) $S, time = 54\,\mathrm{s}$ (k) $S, time = 63\,\mathrm{s}$ (l) $S, time = 100\,\mathrm{s}$

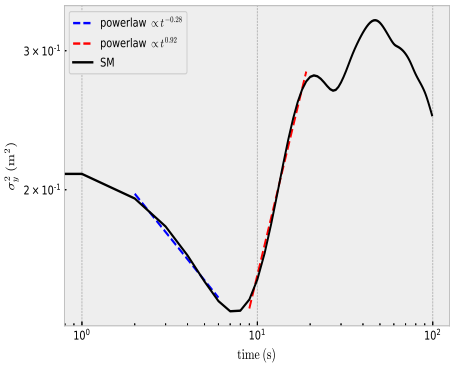
Figure 4.2: The vorticity ω and the concentration distribution of the function S is presented at the time steps $t = 0, 9, 27, 54, 63, 100$ seconds for the second IC of the vorticity field.



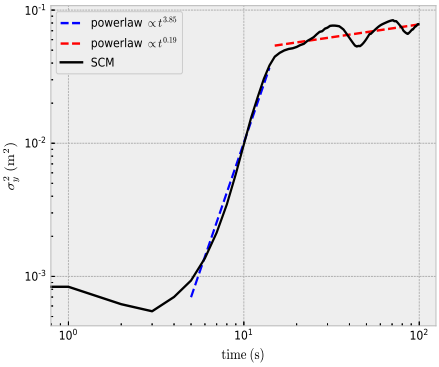
(a) SM for x-projection



(b) SCM for x-projection

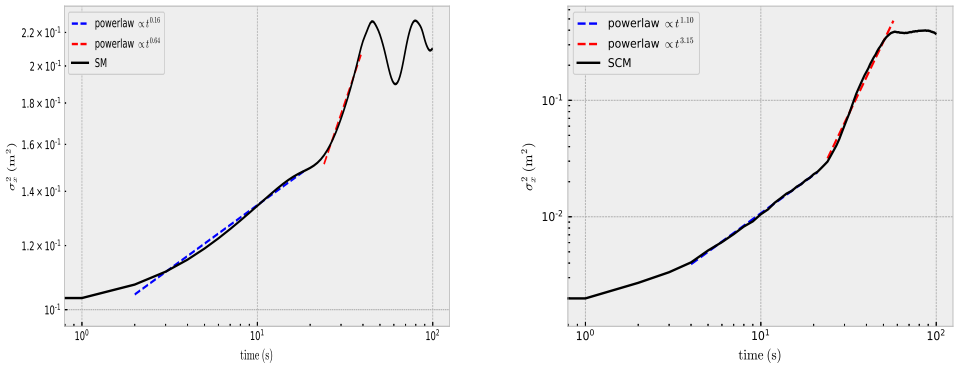


(c) SM for y-projection



(d) SCM for y-projection

Figure 4.3: SM and SCM presented for the x- and y-projected distributions of the solution to the advection-diffusion equation for simulation 1.



(a) SM for x-projection

(b) SCM for x-projection

(c) SM for y-projection

(d) SCM for y-projection

Figure 4.4: SM and SCM presented for the x- and y-projected distributions of the solution to the advection-diffusion equation for simulation 2

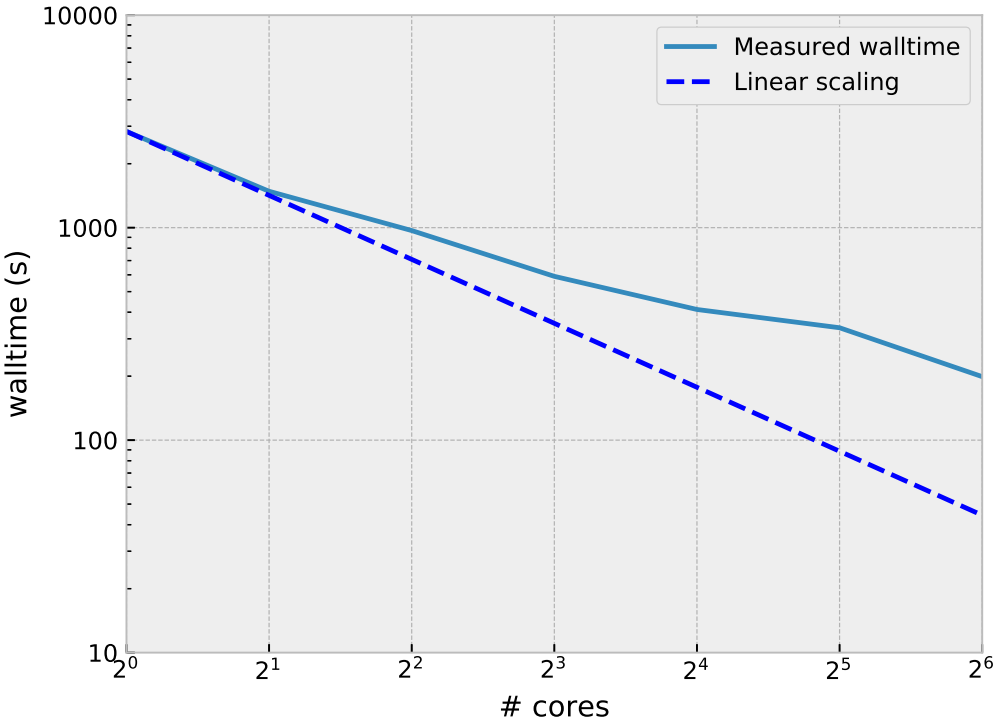


Figure 4.5: Measured wall-time of the DNS solver on the IDUN cluster.

Chapter 5

Discussion

5.1 Turbulent mixing

The solution of the vorticity-transport equation which is seen in the odd-rows of figure 4.1 and 4.2 is as one would expect. The initial distribution of vortices of varying strength and direction is transported around in the computational domain, each influencing each other. Smaller vortices merge to form larger structures, as discussed in the section 2.1. As larger structures start to form, the transportation of them slow down, and they tend to remain at one fixed location. As the smaller vortices have a lesser influence on the future position of a large vortex, the first large vortex in the flow field dominates the flow. For the first simulation, this vortex is forming at $t \leq 9\,\mathrm{s}$ and is positioned in the middle of the field with a value of $\omega = -0.9$, where the units are rotations per second. The sign of the value indicates if the flow is rotating clockwise with positive values, or anti-clockwise with negative values. The strength of the dominant vortex is smaller in magnitude than for the second-largest vortex forming somewhere in the interval $t \in [63\,\mathrm{s} - 100\,\mathrm{s}]$. Since the computational time is here limited to $t = 100\,\mathrm{s}$, it is likely that this second vortex whose value is larger in magnitude, would dominate the flow in the future. For the second simulation, the dominant vortex is seen forming in the interval $t \in [63\,\mathrm{s} - 100\,\mathrm{s}]$.

The solution of the advection-diffusion equation for the quantity S presented in the even-numbered rows of figure 4.1 and 4.2 is seen to correlate well with the flow field which the solution depends upon. The concentration distribution is transported around with high velocity when in regions of high vorticity. As large-structure vortices form, the concentration distribution gets trapped inside such regions. In the first simulation this is seen already at $t \leq 27\,\mathrm{s}$, where the distribution starts to spiral due to the high vorticity. The low influence of the other vortices slow down the mixing once this trapping occurs. A similar trapping mechanism is seen in the second simulation at the end of the computation, i.e. $t = 100\,\mathrm{s}$, where the distribution of S is densest in the same region as the largest vortex. The spiraling nature of the mixing is not as prominent as for the first simulation, which is due to the late formation of the dominant vortex. It is important to emphasize that for a three-dimensional flow, the largest structures would break down into smaller eddies as presented in section 2.1, and there would be no particular dominant vortex trapping the advected quantity S.

For the first simulation, there is a small numerical instability emerging around $t = 9\,\mathrm{s}$, seen as misplaced contour lines in figure 4.1(e), which is due to the transportation being dominated by large

convective forces from the vortices. The Péclet number is thus large, and instabilities are expected for numerical schemes involving central-differences in the convective terms. The instabilities present themselves as oscillations in the solution, they are however damped fairly quickly by the physical diffusion present in the advection-diffusion equation, which is defined in section 3.3.2. The initial noise which is added to the vorticity field is damped by the physical viscosity present in the vorticity-transport equation, defined in 3.3.1. The distribution S is always positive, which is a common metric to look at for advection-diffusion problems. A concentration of a physical quantity should never be negative.

As presented in 2.1 the turbulent diffusion for a three-dimensional flow should scale as the $4/3$ power of the SCM, or the SCM should scale as t^3. This scaling is seen in some small time intervals in figure 4.3(b), 4.3(d), 4.4(b) and 4.4(d) which presents the SCMs for the concentration distribution S in a two-dimensional flow, projected along the x-axis and y-axis. In regions with a clear increase or decrease of either SM or SCM, the data is fitted with a power law to show the scaling with time. For the first simulation, the SM in figure 4.3(a) and 4.3(c) is fitted with lines that scales closely to $\propto t^1$. In regions where the turbulent diffusion is highest, the SCM is seen fitted with power laws that scale much closer to the values presented for three-dimensional flows in Richardson (1926), and since the scaling laws presented by Richardson (1926) is modelled after the SCM and not the SM, it is worth mentioning that this data will resemble the physical diffusion best.

The effects of the dimensional reduction from 3D to 2D is seen in several regions, and they are likely to be the cause of the limited scaling to a full turbulent diffusion. For the first simulation the spreading or mixing of the concentration is close to linear in the $x$direction for the first 15 seconds as seen in figure 4.3(b). After 15 seconds a small region with increased diffusion is seen as the SCM scales as $t^{1.82}$. After this region, the SCM starts to flatten out. This is related to the concentration being trapped in a spiral motion of the dominating vortex in the flow field, as seen in figure 4.1(f) and hence the transportation of S will be heavily influenced on the location of the vortex, whose position is fairly stationary.

The SCM for the distribution in the $y$direction, presented in figure 4.3(d), is dropping the first 5-10 seconds. This is due to the concentration being advected rapidly towards the right at the start from conveniently placed vortices of opposing direction, see figure 4.1(a). The rapid motion flattens the concentration in the $y$direction, and hence its SCM, and SM, decreases. After 10 seconds it increases in the same manner as its counterpart in the $x$direction, however with a scaling similar to that of a three-dimensional flow, $t^{3.85}$. The flattening due to the concentration of S being trapped in a spiraling motion is also seen in the end of the simulation in figure 4.3(d).

A similar scaling in both x- and y-direction is seen for the second simulation using other ICs. The start of the simulation scales linearly in the x-direction as seen in 4.4(b), and closer to turbulent in 4.4(d). In the end the direction switches, and the scaling in the x-direction scales as a three-dimensional flow with $\propto t^{3.15}$, whereas the y-direction scales as $\propto t^{1.25}$. Common for both simulations and directions is that there exist regions with linear scaling of the diffusion, and regions with turbulent scaling of the diffusion. The turbulent scaling is typically seen in the middle of the simulation where there still exist more than a couple of eddies, each with roughly the same size and strength. As the eddies start to combine into the large-scale eddies, which is seen when the time approaches $t = 100\,\mathrm{s}$, the scaling of the diffusion drops to that of a linear scaling. Since the flow is largely dominated by a scaling of diffusion close to that of a Fickian diffusion, i.e. $\propto t^1$, it is clear that the flow fields do not inherit the true diffusive nature of turbulent flows. This is mainly due to the flow field being two-dimensional, and that the inverse energy-cascade creates larger and larger vortices which traps flow and slows down the mixing. Other initial conditions could also contribute to increased mixing. Solving the vorticity-transport equation and the advection-diffusion equation for a large selection of ICs, and look at an averaged SCM of the

ensemble, would rule out bad ICs and improve the statistical basis for concluding that two-dimensional flows do not maintain turbulent diffusion. Setting the initial conditions to those presented in Kempf et al. (2012), to better control the energy of the vorticity, could also be beneficial

Some oscillations can be seen in either the SM or the SCM for both simulations. They are especially prominent in figure 4.3(a) and 4.3(d) near the end of the simulation. The oscillations may be due to the periodic implementation of the boundary conditions. For the first simulation, a large portion of S is seen moving across the right boundary and in through the left boundary. Since the largest vortex is located close to this boundary, the shape of the distribution S will change considerably. Since the SM and SCM are statistical metrics used to describe the shape of this distribution, an oscillating distribution can cause oscillations in the SM and SCM.

5.2 Model performance

The wall-time spent on the IDUN-cluster at NTNU is seen to scale fairly well with the core count, as presented in figure 4.5. By comparing the measured wall-time and the logarithmic line indicating a perfect scaling, it is clear that even though a good reduction in wall-time is seen as the core count increases, some overhead connected to MPI communication is present. When attempting a simulation using 128 cores, the measured wall time was of the same order as for 1 core. As the physical mesh of the DNS solver was discretized using $N = 256$ points, and a slab-decomposition was used to distribute data amongst the cores, it would mean that each core had two rows of data with 256 variables in each row to account for. If overhead related to MPI communication was the problem, it should be somewhat noticeable in the data for 64 cores as well. It is possible that the job was hanging due to some problems related to the node on IDUN. As the allocation of node number and core number is out of control for an external user of IDUN, badly allocated resources could lead to unnecessary overhead. A large ensemble of executions of the code on several cores should be run to average out this statistical effect.

Chapter 6

Conclusion

In this project the two-dimensional mixing of a quantity S is studied in a flow field by solving the Navier-Stokes equations on vorticity-form and the advection-diffusion equation. The solution to the vorticity-transport equation is found numerically by a DNS solver implemented in Python and parallelized with MPI. The advection-diffusion equation is solved using the finite volume method, also parallelized with MPI. Both solvers ran on the IDUN-cluster managed by the HPC group at NTNU, and the measured wall-time for the DNS solver showed good scaling with the number of cores used. The mixing of S was seen to follow a set of power laws relating the second-central-moment to the time by $\sigma^2 \propto t^\alpha$, where α ranged from 0.19 to 3.85, depending on the initial conditions and how the eddies in the flow field develops over time. The Fickian scaling, i.e. $\sigma^2 \propto t^1$, is seen for a larger time portion in the simulation compared to the three-dimensional turbulent diffusion model proposed by Richardson (1926), $\sigma^2 \propto t^3$. This suggests that two-dimensional flows experience three-dimensional scaling as long as the eddies in the flow are of a certain size. When the two-dimensional effects such as the inverse-energy-cascade starts to dominate, the scaling of diffusion drops down to that of a Fickian model.

6.1 Further work

In this section some bullet-points of suggestions for future studies are mentioned. These points may serve as a basis for the extension of this project to a master thesis.

- Run a large ensemble of simulations to achieve a better statistical basis to conclude if two-dimensional flows holds three-dimensional scaling of diffusion.
- Optimize the DNS solver to fit the linear scaling in seen in figure 4.5.
- Do a thorough energy analysis of the flow field. Is the energy conserved? Is the inverse-energy-cascade similar to that of the literature on two-dimensional flows?
- Store less time steps of the flow field from the DNS solver and implement an interpolation method to reduce the need for storage. How will the solution of the advection-diffusion equation, and thus the turbulent diffusion react to such an interpolation?

- Study the performance of the code using metrics like the Karp-Flatt metric, Amdahl's law and Gustafson's law.
- Extend the equations of motion from 2D to 3D.
- Compare the results from a three-dimensional study to the two-dimensional study.

Bibliography

Ansys, I., November, 2019. Ansys fluent, commercial cfd software. https://www.ansys.com/products/fluids.

Batchelor, G.K., 1969. Computation of the energy spectrum in homogeneous two-dimensional turbulence. Physics of Fluids 12.

Boffetta, G., Ecke, R.E., 2012. Two-dimensional turbulence. Annu. Rev. Fluid Mech. 44, –51. doi:10.1146/annurev-fluid-120710-101240.

Canuto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A., 1987. Spectral Methods in Fluid Dynamics. Springer Series in Computational Physics.

Dalcin, L., Kler, P., Paz, R., Cosimo, A., 2011. Parallel distributed computing using python. Advances in Water Resources 34, 1124–1139. doi:10.1016/j.advwatres.2011.04.013.

Dalcin, L., Paz, R., Storti, M., 2005. Mpi for python. Journal of Parallel and Distributed Computing 65, 1108–1115. doi:10.1016/j.jpdc.2005.03.010.

Dalcin, L., Paz, R., Storti, M., D'Elia, J., 2008. Mpi for python: performance improvements and mpi-2 extensions. Journal of Parallel and Distributed Computing 68, 655–662. doi:10.1016/j.jpdc.2007.09.005.

Feynman, R., Leighton, R., Sands, M., 1963. Feynman lectures on physics. Addison-Wesley, London.

Hundsdorfer, W., Verwer, J.G., 2003. Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations. Springer Series in Computational Physics.

Kempf, A.M., Wysocki, S., Pettit, M., 2012. An efficient, parallel low-storage implementation of klein's turbulence generator for les and dns. Computers Fluid .

Kraichnan, R.H., Motgomery, D., 1980. Two-dimensional turbulence. Reports on Progress in Physics 43, 547–619. doi:10.1088/0034-4885/43/5/001.

Leith, C.E., 1971. Atmospheric predictability and two-dimensional turbulence. Pjournal of the Atmospheric Sciences 28, 145–161.

Liu, C., Cai, X., 2017. New theory on turbulence generation and structure - dns and experiment. Science China, Physics, Mechanics Astronomy 60.

Maulik, R., San, O., 2017. A dynamic framework for functional parameterizations of the eddy viscosity coefficient in two-dimensional turbulence. International Journal of Computational Fluid Dynamics .

Mortensen, M., Langtangen, H.P., 2016. High performance python for direct numerical simulation of turbulent flows. Computer Physics Communications .

Pacheco, P., 2011. An Introdution to Parallel Programming. Morgan Kaufmann.

Pletcher, R.H., Tennehill, J.C., Anderson, D.A., 2013. Computational Fluid Mechanics and Heat Transfer. CRC Press, series in computational and physical processes in mechanics and thermal sciences.

Richardson, L.F., 1926. Atmospheric diffusion shown on a distance-neighbour graph. Royal Society of London. Series A, Containing Papers of a Matheatical and Physical Character 110, 709–737.

Tennekes, H., Lumley, J.L., 1972. A First Course in Turbulence. The MIT Press.

White, F.M., 2006. Viscous Fluid Flow. McGraw Hill Education.

White, F.M., 2011. Fluid Mechanics. McGraw Hill Education.