Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Daniel Ørnes Halvorsen

Studies of turbulent diffusion through direct numerical simulation

Master's thesis in Applied Physics and Mathematics
Supervisor: Tor Nordam, Adjunct Associate Professor, IFY
June 2020

NTNU
Norwegian University of
Science and Technology

Daniel Ørnes Halvorsen

Studies of turbulent diffusion through direct numerical simulation

Master's thesis in Applied Physics and Mathematics
Supervisor: Tor Nordam, Adjunct Associate Professor, IFY
June 2020

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

NTNU

Kunnskap for en bedre verden

Abstract

In this thesis a three-dimensional solver for direct numerical simulation was implemented using a spectral method, and parallelized using a pencil decomposition, to study the superdiffusive nature of turbulent flow fields. Transport in turbulent flow fields were studied using two different methods; a stochastic particle model and the advection-diffusion equation. The particle model was parallelized directly by scattering the particles amongst computer cores, and the advection-diffusion equation was parallelized using a slab decomposition. The numerical implementations were run on the high-performance computers Idun and Vilje, both located at NTNU in Trondheim. Two simulations were run. The computed variance of the particle positions verifies the superdiffusive nature of turbulent flows in the first simulation as the variance scales with time according to $\sigma^2 \propto t^{2.54}$. In the second simulation the computed variance of the particle positions and the numerical solution to the advection-diffusion equation indicates a more linear scaling. The conclusion is that more simulations must be run to achieve a statistical confidence on the scaling of the turbulent diffusion.

Sammendrag

I denne mastergraden ble en tredimensjonal løser for direkte numerisk simulasjon implementert ved bruk av en spektralmetode, og parallellisert ved bruk av en penn-dekomposisjon, med hensikt å studere den superdiffusive oppførselen til turbulente strømningsfelt. Transport i turbulente strømningsfelt ble studert ved bruk av to ulike metoder; en stokastisk partikkel modell og adveksjons-diffusjons likningen. Partikkelmodellen ble parallellisert direkte ved å spre partiklene mellom prosessorer, og adveksjons-diffusjons likningen ble parallellisert ved bruk av en plan-dekomposisjon. De numeriske implementasjonene ble kjørt på superdatamaskinene Idun og Vilje, begge lokalisert ved NTNU i Trondheim. To simuleringer ble kjørt. Den kalkulerte variansen til partikkelposisjonene bekreftet den superdiffusive oppførselen til turbulente strømninger i den første kjøringen siden variansen skalerte med tiden som $\sigma^2 \propto t^{2.54}$. I den andre simuleringen indikerte den kalkulerte variansen av partikkelposisjonene og den numeriske løsningen til adveksjons-diffusjons likningen en mer lineær skalering. Konklusjonen er at flere simuleringer må kjøres for å danne statistisk konfidens på skaleringen av den turbulente diffusjonen.

Preface

This thesis is written as a part of the master program in Applied Physics and Mathematics at NTNU in Trondheim. The master's thesis yields 30 ETCS credits and has been conducted during the spring of 2020.

I would like to thank my supervisor, Tor Nordam, for his encouragement and advice throughout the last two semesters. He introduced me to the broad field of transport phenomena, which this thesis is devoted to study. The discussions on both the numerical methods involved and the results of this thesis have been invaluable and deeply appreciated.

Throughout my years in Trondheim I have accumulated a number of great friends. Even though we all will spread throughout the country to pursue individual careers, I know that the precious memories from our time in Trondheim will persist.

Finally, my sincere gratitude to my family for their continuous and invaluable support through all these years. Lastly, a special thanks to Silje and our dog Kelvin for filling my days with joy and love. I love you all.

- 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 system 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 scientists since the first transport models were designed. With the help of high-performance computers, the distribution can be studied 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 the chaotic and random motions of 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 dilution 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. Even though chaotic motion can happen in flows where two of the spatial directions dominate the extent of the system, such as macroscopic geophysical and astrophysical flows, and thin films in microscopic flows, turbulent flows are mainly associated with fully thee-dimensional systems. A lot of work has been conducted on the exciting phenomena and physical mechanisms associated with the reduction from three dimensions to two dimensions. The curious reader can consult, e.g., Batchelor (1969); Leith (1971); Kraichnan and Motgomery (1980); Boffetta and Ecke (2012); Maulik and San (2017), while the rest of the work in this thesis will be devoted to three-dimensional turbulence.

The current state-of-the-art research done in the field of fluid flow focuses on complex multi-scale flows including effects of combustion, shock-waves, acoustics etc. For instance, at the Department of Energy and Process Engineering at NTNU, within the Thermo fluids group, an extensive amount of work is conducted on understanding the instability of flames in combustion processes, as unstable flames consume more fuel, see, e.g., Buschmann et al. (2020); Orchini et al. (2019). Moreover, wind energy has received increasing attention in recent years due to improved availability of the technology. The deep understanding of fluid flow and how the wind turbines are impacted by the chaotic motion of wind, is crucial in order to develop optimized wind parks. An overview of recent technological advancements in this field is presented by, e.g., Willis et al. (2018).

1.1 Purpose and outline of the work done

The purpose of this thesis is to implement a fully parallelized direct numerical simulation of the equations of motion in three-dimensions in order to study the diffusive nature of turbulent flows. Advancing the fundamental theory and applied technologies of fluid flow has been and will continue to be an essential part of the ongoing process of ensuring a sustainable future for the world, and this thesis may serve as a brief introduction to parts of the technologies involved in current research.

In section 2.1 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, also referred to as the variance, is presented as a statistical metric to describe the distribution of a quantity affected by the turbulent flow field. Section 2.2 presents the equations of motion implemented in the direct numerical solver, followed by two different methods for modeling transportation of a physical quantity in a flow field. The first method is the advection-diffusion equation which is a partial differential equation, and the second method is a particle model involving a drift term due to a velocity field, as well as a random contribution to model diffusion. The particle model is a stochastic differential equation. Furthermore, an introduction to the concept of spectral methods is given which is used in the numerical implementation of the direct numerical simulation.

Section 3.1 and 3.2 presents the numerical implementation of the direct numerical simulation, the advection diffusion equation and the particle model, as well as some stability criteria and relevant numerical constants. Section 3.3 introduces the concept of a message passing interface to communicate between computer cores, which enables the numerical codes to run on high-performance computers. The high-performance computers utilized in the work done here were the computing clusters Idun and Vilje, both affiliated with the high-performance computing group at NTNU. Section 3.4 introduces a series of metrics or laws used to measure the performance of numerical codes executed on high-performance computers. Section 3.5 ends the chapter on numerical methods with a benchmark of the direct numerical simulation by investigating the properties of the Taylor-Green vortex.

Chapter 4 presents and discusses the result from the numerical simulations. Section 4.1 presents the performance of the direct numerical simulation run on the Vilje computing cluster, and the output from the direct numerical simulation is discussed in section 4.2. Section 4.3 presents and compares the result from the particle method and the advection-diffusion equation respectively, as well as challenges related to the numerical solutions of the different mathematical models introduced in section 2.2.

Lastly, it should be noted that a computational setup consisting of a direct numerical simulation, a particle model and the numerical solution of the advection-diffusion equation, all in three spatial dimensions, requires an immense amount of computational resources. The stochastic nature of turbulence calls for a large number of simulations in order to study its diffusive nature with statistical confidence. How-

ever, this was not practically possible with the computational resources available during the work on this thesis.

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

In this chapter the theoretical background for the thesis is presented. First, a brief introduction is given to the field of turbulence and its diffusive nature. Secondly, an overview of the model equations used to describe the transport phenomena which this thesis is devoted to study, e.g., the Navier-Stokes equation, the advection-diffusion equation and a particle model with a stochastic contribution. Moreover, an introduction is given to the concept of spectral methods which the numerical implementation of the Navier-Stokes equation relies on.

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

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

where u is the velocity of the fluid and D 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 fluid 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 flow 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 a 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 (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 one of the most important aspect of turbulent flows when considering possible applications. For example, 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 increases 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 phenomenon 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, 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 characteristic length scale, D, in the Reynolds number is commonly substituted with three different length scales of the flow. 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. All length scales between l and η are referred to as the *intermediate length scales*, denoted by λ. In three dimensional flows the non-linear term in the equations of motion break down large scale structures into smaller and smaller structures, or eddies. This is referred to as *vortex-stretching*. At the smallest possible length scale the viscous effects dominate 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.

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 the physical parameters viscosity and dissipation 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:

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

The collection of the largest scales in the flow, e.g., l, t and u, are often referred to as the integral scales, whereas the smallest scales in the flow, e.g., η, τ and v, are referred to as the *Kolmogorov microscales*. Throughout the thesis the terms *largest length scale*, *largest structures* or *transverse length scale* are used interchangeably. The same applies for the terms *smallest length scale*, *smallest structures* and *Kolmogorov length scale*. It is instructive to mention that $\frac{\eta v}{\nu} = \frac{ul}{\nu}\left(\frac{ul}{\nu}\right)^{-3/4}\left(\frac{ul}{\nu}\right)^{-1/4} = 1$ is the Reynolds number for the small scale flow, meaning that viscous effects are not negligible in these scales. An overview of how the energy in the flow is distributed amongst the different length scales is given in section 2.4

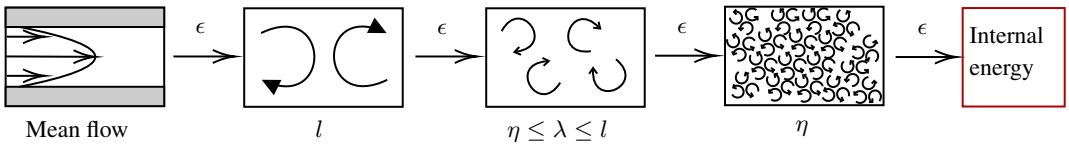


Figure 2.1: The energy cascade presented for three-dimensional flow. The energy is dissipated from the large scale structures and down to the smallest scales, indicated by the black arrow. l, λ and η indicates the transverse, the intermediate and the Kolmogorov length scales respectively. ϵ is the rate of energy dissipation.

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. the concentration of a physical quantity and c is the mean of the distribution. SCM are used to study the shape of the distribution $f(x)$ and are 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. Note that the conventional Fickian diffusion which occurs due to concentration gradients, scales linearly with time, i.e., $\sigma^2 \propto t$, see (Kloeden and Platen, 1992; Øksendal, 2003). The derivation of both scaling laws is presented by Richardson (1926), but is repeated here for instructional purposes.

Consider the displacement of an individual diffusing particle with velocity, u,

$$x(t) = \int_0^t u(t')dt', \tag{2.7}$$

where only the x-component of the displacement is considered for brevity. Taking the mean of both sides of equation (2.7) yields $\overline{x}(t) = 0$ if $\overline{u}(t) = 0$. By instead considering the rate of change of $x^2(t)$, equation (2.7) may be written as

$$\frac{dx^2(t)}{dt} = 2x\frac{dx}{dt} = 2\int_0^t u(t)u(t')dt'. \tag{2.8}$$

Where $\frac{dx}{dt} = u(t)$ has been moved inside of the integral since it is independent of t'. Taking the mean of both sides of equation (2.8) yields

$$\frac{d\overline{x^2}(t)}{dt} = 2\overline{u^2}\int_0^t R(\tau)d\tau, \tag{2.9}$$

where the *velocity autocorrelation function*, $R(\tau)$, is introduced, in addition to a change of integration variables from t' to $\tau = t' - t$. $R(\tau)$ is defined as

$$R(\tau) = \frac{\overline{u(t)u(t+\tau)}}{\overline{u^2}}, \tag{2.10}$$

and indicates how the velocity field u correlates with itself at different times, t. When τ approaches zero, the correlation is total, or 1. When τ approaches ∞, the correlation is zero. A plot over how the velocity autocorrelation depends on its argument, τ, is presented in figure 2.2.

Equation (2.9) is known in turbulent diffusion theory as Taylor's theorem, and is a very important result in the theory of diffusion by random motion, such as Brownian or turbulent motion. A mathematical description of Taylor's theorem is given by, e.g., Csanady (1973). It can further be shown that the left hand side of equation (2.9) is exactly two times the effective mixing parameter, or turbulent diffusion, K,

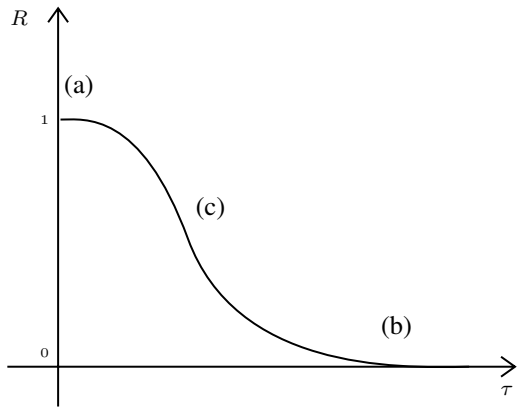


Figure 2.2: Qualitative sketch of the velocity autocorrelation function, $R(\tau)$. The points (a), (b) and (c) indicates the smallest, intermediate and largest *temporal* scales in the flow, corresponding to the largest, intermediate and smallest *spatial* scales in the flow.

$$\frac{1}{2}\frac{d\overline{x^2}(t)}{dt} = \overline{u^2}\int_0^t R(\tau)d\tau = K. \tag{2.11}$$

A derivation of this equality is presented by e.g. Ghandi (2004).

To achieve the result of Richardson, equation (2.6), Taylor's theorem is evaluated for certain limits, i.e. the autocorrelation, R, is computed for different length scales in a velocity field, u. Three cases will be considered: a) the smallest scales, b) the largest scales, c) the intermediate scales.

a) **Small scales:** In the smallest scales in the velocity field, where σ is of order $\mathcal{O}(\eta)$, i.e. the Kolmogorov length scale, the time scale can be assumed to approach zero. Since $R(0) = 1$, Taylor's theorem can be rewritten to

$$\frac{1}{2}\frac{d\sigma^2}{dt} = \overline{u^2}\int_0^t d\tau = \overline{u^2}t = K. \tag{2.12}$$

Since the mean square of the velocity field, $\overline{u^2}$ is considered to be constant, the effective mixing coefficient, K, is seen to increase linearly with time.

b) **Large scales:** In the largest scales in the velocity field, where σ is of order $\mathcal{O}(l)$, i.e. the integral length scale, the time scale can be assumed to be very large. Since $R(\tau) = 0$ for sufficiently large values of τ, Taylor's theorem can be rewritten to

$$\frac{1}{2}\frac{d\sigma^2}{dt} = \overline{u^2}\int_0^t R(\tau)d\tau = \overline{u^2}t_l = K. \tag{2.13}$$

Where t_l is an integration constant. The product of the mean square of the velocity field and this integration constant is still constant, and thus the effective mixing coefficient K is constant in time for the integral scales of the flow.

c) **Intermediate scales:** In the intermediate scales, i.e. where the Richardson scaling is expected to hold, $R(\tau)$ is unknown. A dimensional analysis of the effective mixing coefficient will reveal how it scales with time. To start such an analysis it is necessary to determine which parameters the coefficient is dependent upon.

Figure 2.3: Qualitative sketch of the transport of a particle pair with separation $D(t)$.

Consider the motion of the particle pair presented at two time steps, 0 and t, in figure 2.3. The distance between the two particles are denoted by D. For the description of the particle pair movement to hold in the intermediate length scales, the following assumptions must be valid:

- The time t is large enough for the particles at this time to forget about their initial state when $t = 0$.
- The distance, $D(t)$, is sufficiently larger than the Kolmogorov length scale η, such that the viscosity is negligible.
- The distance, $D(t)$ is sufficiently lower than the integral length scale l, such that the description of the particle pair movement is statistically isotropic, meaning it is not dependent on large scale production of energy or small scale viscous effects.

The distance is thus only dependent on the physical effects in the inertial sub-range, which is the energy dissipation rate, ϵ, which quantifies how much energy is transferred from the largest scales to the smallest scales, through the intermediate scales. The effective mixing coefficient is thus said to be equal to a function which depends only on this parameter, in addition to time,

$$\underbrace{K}_{[\frac{L^2}{T}]} = f(\underbrace{\epsilon}_{[\frac{L^2}{T^3}]}, \underbrace{t}_{[T]}) \tag{2.14}$$

Since the right hand side of equation (2.14) needs to have the same units as its left hand side, the dependent parameters, i.e. the viscous dissipation, ϵ, and the time, t is multiplied together in such a manner that this is achieved. The effective mixing coefficient is then seen to be proportional to

$$K \propto \epsilon t^2, \tag{2.15}$$

where a scaling factor, c, is included to make the proportionality an equality,

$$K = c\epsilon t^2. \tag{2.16}$$

By following Taylor's theorem, the variance of the particle distribution can be found,

$$\frac{1}{2}\frac{d\sigma^2}{dt} = K = c\epsilon t^2, \tag{2.17}$$

where integration in time yields,

$$\sigma^2 = \frac{2}{3} c\epsilon t^3, \tag{2.18}$$

or,

$$\sigma^2 \propto t^3. \tag{2.19}$$

This is often referred to as the Richardson-Obukov t^3 regime, see Malik (2018). To relate the mixing coefficient, K to the particle variance, σ^2, rewrite equation (2.16) to solve for t, and substitute this for t in equation (2.18), which yields

$$K = \left(\frac{9}{4} c\epsilon\right)^{\frac{1}{3}} \sigma^{\frac{4}{3}}, \tag{2.20}$$

or,

$$K \propto \sigma^{\frac{4}{3}}. \tag{2.21}$$

Hence, both versions of the scaling laws given by Richardson (1926) are found. The analysis of turbulent diffusion in this thesis relies on the version where $\sigma^2 \propto t^3$, as both the variance of a distribution and the time for which it occurs, are handily available when working numerically. The above results are in line with the assumption that the initial particle separation in figure 2.3 is forgotten for large values of t, meaning that $\sigma = 0$ for $t = 0$.

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 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, and it is not practical for industrial flows. For commercial software such as Ansys Fluent, a DNS solver is not directly implemented, although it can be accessed by choosing a laminar turbulence model and choosing a well resolved mesh. See the handbook on ANSYS, Incorporated (2013) for the present capabilities of commercial fluid flow software.

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 (Burattini et al., 2008; Moin and Mahesh, 1998). 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. For instance, the size of the smallest length scales in a flow with $Re = 1600$ is of order $\mathcal{O}(4 \times 10^{-3})$, which can be seen by substituting values into the relations in equation (2.2).

The numerical implementation of a DNS solver will vary with each task it is set to solve, and its performance will vary with problem size, method of parallelization etc. A three-dimensional DNS solver based on a spectral method implemented in Python is presented by, e.g., Mortensen and Langtangen (2016). A similar implementation is done for the DNS solver used in this thesis, and the following sections introduce the equations of motion and the numerical methods involved to implement the solver.

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,

$$\boldsymbol{u} = \boldsymbol{u}(t, x, y, z), \tag{2.22}$$

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:

$$\boldsymbol{F} = m\boldsymbol{a}. \tag{2.23}$$

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

$$\rho \frac{D\mathbf{u}}{Dt} = \boldsymbol{f}_{\text{body}} + \boldsymbol{f}_{\text{surface}}, \tag{2.24}$$

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

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0. \tag{2.25}$$

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

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \frac{1}{\rho}\boldsymbol{\nabla} \cdot \boldsymbol{\tau}, \tag{2.26}$$

where $\mathbf{u}$ 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

$$\boldsymbol{\tau} = \mu[\boldsymbol{\nabla}\mathbf{u} + (\boldsymbol{\nabla}\mathbf{u})^T] - \frac{2}{3}\mu\boldsymbol{\nabla} \cdot \mathbf{u}\mathbf{I}, \tag{2.27}$$

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,

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \nu\boldsymbol{\nabla}^2\mathbf{u}, \tag{2.28}$$

where $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity. The first term of (2.28) 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 Rotational form of the Navier-Stokes equations

The nonlinear convective term in equation (2.28) can be written in numerous ways, which are all equivalent when the equation is on differential form where all variables are continuous (additional complications due compressibility effects such as shock waves in the solution are not considered in this thesis). The different forms may differ in computational cost, conserved quantities, accuracy and stability when discretized, however this is not covered here. See e.g. Gresho and Sani (2000) or Gunzburger (1989) for an extensive overview of why the discretized forms are not equivalent. Some of the forms the convective term take are

$$\mathbf{u} \cdot \nabla\mathbf{u}, \qquad (\nabla \times \mathbf{u}) \times \mathbf{u} + \nabla\left(\frac{1}{2}|\mathbf{u}|^2\right), \qquad \frac{1}{2}\mathbf{u} \cdot \nabla\mathbf{u} + \frac{1}{2}\nabla(\mathbf{u}\mathbf{u}),$$

where the forms are called the divergence form, the rotation form and the skew-symmetric form respectively. Zang (1988) show that the rotational form is superior when it comes to conservation of quantities like helicity, enstrophy and vorticity, compared to the other forms, however when using spectral methods which induces aliasing errors, the rotation form is less accurate if not de-aliased. In this thesis the convective term is on rotation form, yielding the momentum equation on rotation form,

$$\frac{\partial \mathbf{u}}{\partial t} - \mathbf{u} \times \boldsymbol{\omega} = -\boldsymbol{\nabla} P + \nu\boldsymbol{\nabla}^2\mathbf{u}, \tag{2.29}$$

where the kinematic pressure is replaced by the total pressure, $P = p + \frac{1}{2}(\mathbf{u} \cdot \mathbf{u})$. The convective term in the conventional formulation is rewritten into the rotational form using the relation $(\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = (\boldsymbol{\nabla} \times \mathbf{u}) \times \mathbf{u} + \frac{1}{2}\boldsymbol{\nabla}(\mathbf{u} \cdot \mathbf{u})$, where the curl of the velocity field is per definition the vorticity field, $(\boldsymbol{\nabla} \times \mathbf{u}) = \boldsymbol{\omega}$.

2.2.3 The advection-diffusion equation

The advection-diffusion equation for a conserved quantity S is

$$\frac{\partial S}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u}S) = \boldsymbol{\nabla}(D\boldsymbol{\nabla}S), \tag{2.30}$$

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. The discretized form of equation (2.30) is presented in section 3.1.2, and its implementation on a parallel system is presented in section 3.3.2.

It is worth noting that for a real system, other physical effects may take place, such as reactions between different species, acting as sources or sinks in the solution. The diffusion parameter, D, may also be treated as a function of time and space. In this thesis the diffusion is treated as constant, and no reactions are considered.

2.2.4 Particle model

While the advection-diffusion equation has unique, true solutions (subject to certain conditions, which we assume to be satisified for the systems of interest here), these solutions are usually unknown. However, the solutions can be approximated by different methods, such as by numerical solution of the PDE, or by ensemble simulations with large numbers of particles. The latter method is introduced here.

Consider a selection of m particles at an initial position $\mathbf{x}(t = 0)$, whose future positions depend on the velocity field $\mathbf{u}(\mathbf{x}, t)$. The motion of these particles can be modeled by a rather simple ordinary differential equation (ODE), combining the rate of change of the particles' position with the velocity field through the relation

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}, t). \tag{2.31}$$

A random contribution is added to the particle position to model molecular diffusion. The equation is now a stochastic differential equation (SDE), on the form

$$d\mathbf{x} = \mathbf{u}dt + \sqrt{2nD}dW(t), \tag{2.32}$$

where $W(t)$ is a Wiener process, see, e.g., Kloeden and Platen (1992) or Øksendal (2003). When equation (2.32) is discretized with the Euler-Maruyama method, see, e.g., Higham (2001), it takes the form

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{u}\Delta t + \sqrt{2nD}\Delta W, \tag{2.33}$$

where Δt is the time step of discretized model, $\mathbf{x}_i = \mathbf{x}(t = i\Delta t)$, n is the number of spatial dimensions, i.e. $n = 3$ and D is the diffusion parameter. ΔW is a Gaussian random variable with expectation value $\langle \Delta W \rangle = 0$ and variance $\langle \Delta W^2 \rangle = \Delta t$. In practice the stochastic term adds a spatial contribution, $\mathbf{r}_i$, in each direction, scaled by the variance $\sqrt{6D\Delta t}$, as explained by Visser (2008).

It is important to note that the collection of particles modeled above are not reacting with the flow field $\mathbf{u}$, and are therefore referred to as *passive tracers*. More complicated transport models exist taking into account interaction between particles, buoyancy effects, inertia etc. More on particle methods can be

found in e.g. Rodean (1996), Yeoh and Tu (2009), Gnedenko and Kolmogorov (1954) and Montgomery and Runger (2013).

2.2.5 Equivalence of the particle method and the advection-diffusion equation

In the continuum limit, i.e., for an infinite amount of particles, the time evolution of a distribution of particles will be described by the Fokker-Planck equation,

$$\frac{\partial p}{\partial t} = -\boldsymbol{\nabla} \cdot (\mathbf{u}p) + \boldsymbol{\nabla}^2(Dp), \tag{2.34}$$

where $p(t, x, y, z)dxdydzdt$ is the probability of finding a particle within a small spatial volume $[x + dx, y + dy, z + dz]$ in the small time interval $[t, t + dt]$. If the diffusion parameter, D, is constant, equation (2.34) is the advection-diffusion equation for a distribution p. If D is not constant, a correction term must be added to the advection-part of (2.32) for it to be equal to equation (2.30). See e.g. Nordam et al. (2019) for a discussion of how this correction affects random walk schemes in oil spill modelling. To show that the stochastic differential equation, (2.32), is equal to the advection-diffusion equation, (2.30), in the continuum limit, a derivation following the notation used in the Appendix A in Nordam et al. (2019) is presented for the one-dimensional case.

Consider the SDE,

$$dz = a(z, t)dt + b(z, t)dW, \tag{2.35}$$

which models a stochastic diffusion process referred to as an Itô diffusion. $a(z, t)$ and $b(z, t)$ are moderately smooth functions, and depending on their value, equation (2.35) resembles different diffusion processes, e.g., $a = 0$ and $b = 1$ equals the standard Wiener process, and $a = -x$ and $b = \sqrt{2}$ resembles the Ornstein-Uhlenbeck process, both important diffusion processes present in physics, biology and finance. The functions a and b are referred to as the drift and diffusion coefficient respectively. The Fokker-Planck equation for the rate of change of the probability density function, $p(t, x, y, z)$, is

$$\frac{\partial p}{\partial t} = \frac{1}{2}\frac{\partial^2}{\partial z^2}(b^2 p) - \frac{\partial}{\partial z}(ap). \tag{2.36}$$

When the Fokker-Planck equation is used to describe the distribution of particle positions, it is referred to as the Smoluchowski equation or the Smoluchowski convection-diffusion equation (Pavliotis, 2014) . Essentially, p is a probability density function of finding a particle under the influence of drag forces or advective-transport and random forces such as diffusion, at a certain position and time. Equation (2.36) can be rewritten to,

$$\frac{\partial p}{\partial t} = \frac{1}{2}\frac{\partial}{\partial z}\left(b^2\frac{\partial p}{\partial z}\right) - \frac{\partial}{\partial z}\left[\left(a - \frac{1}{2}\frac{\partial b^2}{\partial z}\right)p\right]. \tag{2.37}$$

By comparing (2.37) to the advection-diffusion equation, (2.30), where the advection speed is $w(z, t)$ and the diffusion parameter is $D(z)$,

$$\frac{\partial S}{\partial t} = \frac{\partial}{\partial z}\left(D\frac{\partial S}{\partial z}\right) - \frac{\partial}{\partial z}(wS), \tag{2.38}$$

it is possible to draw some similarities. If S is proportional to p, then $D(z) = \frac{b^2}{2}$, yielding $b = \sqrt{2D(z)}$, and $a = w + \frac{\partial D(z)}{\partial z}$. Inserting for a and b in the SDE (2.32) gives,

$$dz = \left(w + \frac{\partial D(z)}{\partial z} \right) dt + \sqrt{2D(z)} dW. \tag{2.39}$$

This is the corrected random walk, as discussed by Nordam et al. (2019). Since the diffusion parameter, D, is here considered to be constant, the SDE takes the form,

$$dz = w dt + \sqrt{2D(z)} dW, \tag{2.40}$$

which is the one dimensional case of equation (2.32). Hence, it is shown that the stochastic particle model presented in section 2.2.4 is equivalent with the advection-diffusion equation for a constant diffusion parameter.

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 inherits 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,

$$\text{error} \approx \mathcal{O}\left(\left(\frac{\mathrm{L}}{\mathrm{N}} \right)^{\mathrm{N}} \right), \tag{2.41}$$

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 defined as

$$\hat{u}_k = \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-ikx} dx \quad k = 0, \pm 1, \pm 2, \ldots. \tag{2.42}$$

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

$$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx}, \tag{2.43}$$

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 (2.43) takes the form

$$u = \sum_{k=-N/2}^{N/2-1} \hat{u}_k e^{ikx}, \tag{2.44}$$

which is named the truncated Fourier series. The function u involved in the continuous Fourier transform must 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

$$x_j = \frac{2\pi j}{N} \qquad j = 0, \ldots, N-1, \tag{2.45}$$

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,

$$\widetilde{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-ikx_j} \qquad -N/2 \leq k \leq N/2 - 1, \tag{2.46}$$

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

$$u(x_j) = \sum_{k=-N/2}^{N/2-1} \widetilde{u}_k e^{ikx_j} \qquad j = 0, \ldots, N-1. \tag{2.47}$$

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 (2.46) and (2.47) 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), see, e.g., Nussbaumer (1982).

2.3.3 Fast Fourier Transform

The FFT is a recursive algorithm for evaluating the transformations (2.46) and (2.47). 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 (2.46) and (2.47), the floating point operations involved are a sum over both the spatial index, j, and the wavenumber 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 (2.45), then the discrete Fourier coefficients can be expressed in terms of the exact Fourier coefficients of u through

$$\widetilde{u}_k = \hat{u}_k + \sum_{\substack{m=-\infty \\ m\neq 0}}^{+\infty} \hat{u}_{k+Nm} \qquad k = -N/2, \ldots, N/2 - 1 \tag{2.48}$$

where $\widetilde{u}_k$ are the discrete coefficients and $\hat{u}_k$ are the exact coefficients. Equation (2.48) reveals that the k-th mode of the discrete coefficient not only depends on the k-th mode of the exact coefficient but the mode $k + Nm$. The *Nyquist sampling theorem* states that k is the highest wave number that can be represented as a grid function f_j with $j = 1, \ldots, N = 2k$. Higher wave numbers $k_h \geq k$ are mapped back to available modes through the relation

$$k = k_h - nN, \quad n \in \mathbb{Z}. \tag{2.49}$$

Wave numbers higher than k, e.g., k_h, are referred to as the modes which *alias* the k-th mode. Rephrased, this means that in the DFT, aliasing occurs when more Fourier modes are present than the sampling rate allows to represent accurately. The approximated solution $\tilde{u}_k$ is essentially disturbed by spurious Fourier-modes due to poor sampling rate, see, e.g., Nussbaumer (1982).

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

$$W_j = U_j V_j \qquad j = 0, \ldots, N - 1 \tag{2.50}$$

where the linear factors are defined as

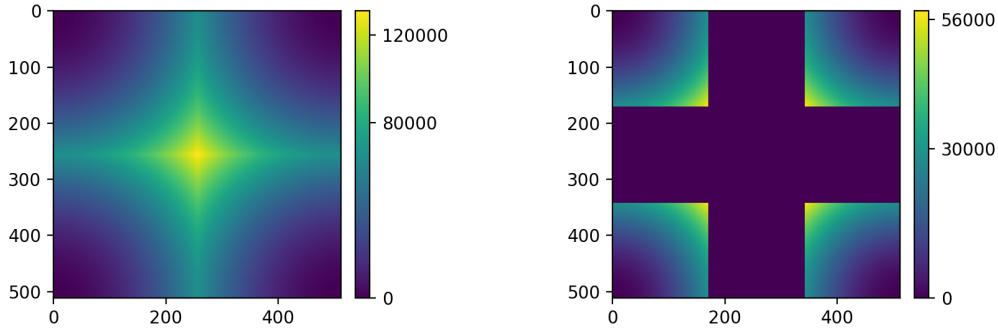
$$
\begin{aligned}
U_j &= \sum_{k=-N/2}^{N/2-1} \widetilde{u}_k e^{ikx_j} \\
V_j &= \sum_{k=-N/2}^{N/2-1} \widetilde{v}_k e^{ikx_j}.
\end{aligned}
\tag{2.51}
$$

The transformation of W_j is

$$
\widetilde{W}_k = \frac{1}{N} \sum_{j=0}^{N-1} W_j e^{-ikx_j} \qquad k = -N/2, \ldots, N/2 - 1, \tag{2.52}
$$

which by insertion of the linear factors in equation (2.51) give,

$$
\widetilde{W}_k = \sum_{m+n=k} \widetilde{u}_k \widetilde{v}_k + \sum_{m+n=k\pm N} \widetilde{u}_k \widetilde{v}_k \tag{2.53}
$$



(a) Cross section of the wavenumber mesh $\boldsymbol{k}^2$ before de-aliasing.

(b) Cross section of the wavenumber mesh $\boldsymbol{k}^2$ after de-aliasing.

Figure 2.4: Here the wavenumber 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. The presented cross section of the mesh is

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 the method referred to as truncation, or the 2/3- rule, see e.g. Orszag (1971). The idea is to choose the mesh size (or sampling rate to use the same phrase as before) to be of M points rather than N, where the two are related through $M \geq 3N/2$. It is then possible to choose M such that the extra terms in equation (2.48) and (2.53) 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 $2/3$ of the highest frequency $N/2+1$. The 2/3-rule is presented in figure 2.4 for the variable $\boldsymbol{k}^2 = k_x^2 + k_y^2 + k_z^2$, where k_x, k_y and k_z are wavenumber vectors presented in section 3.1.

2.4 Kinetic energy spectra

Using a spectral formulation to describe certain properties of a flow, such as energy distribution, can in some cases be convenient. In a spatial homogeneous flow, i.e. a flow where the statistical properties are independent of position, a spectral description is favorable. It allows for flow properties to be examined as a function of wavelength. The wavelengths in this context are the same as first presented in section 2.3, i.e. the largest eddies in the flow correspond to small wavenumbers, and the smallest eddies correspond to the highest wavenumbers. In this section a derivation of the kinetic energy spectrum for turbulent flows is presented, alongside some physical traits of the flow obtained from analyzing the given energy spectrum.

2.4.1 Velocity correlation tensor

A central part of the derivation of the energy spectrum is the velocity correlation tensor, defined as

$$R_{i,j}(\mathbf{r}) = \frac{1}{V} \iiint u_j(\mathbf{x}) u_i(\mathbf{x}+\mathbf{r}) d\mathbf{x}, \tag{2.54}$$

where V is the volume containing the velocity field, u, and the indices i and j indicate its direction. $R_{i,j}$ indicates how velocities at points separated by a vector $\mathbf{r}$ are correlated. Note that this is a spatial correlation, as opposed to the temporal correlation presented in equation (2.10). At very large separation distances, R approaches zero. For points that are closer in space, e.g. located on the same eddy, there will be some correlation.

2.4.2 Energy spectrum tensor

Taking the Fourier transformation of the velocity correlation tensor yields the energy spectrum tensor

$$\phi_{i,j}(\mathbf{k}) = \frac{1}{(2\pi)^3} \iiint R_{i,j}(\mathbf{r}) e^{-i\mathbf{kr}} d\mathbf{r}, \tag{2.55}$$

which contains information about the spectral distribution of the kinetic energy in the flow, i.e. how much of the kinetic energy is contained in the eddies whose sizes correspond to the wavenumber $\mathbf{k}$. This information is accessed when the tensor indices are equal, i.e. $i \neq j$, which yields the energy spectrum

$$E_{i,i}(\mathbf{k}) = \phi_{i,i}(\mathbf{k}). \tag{2.56}$$

The energy spectrum $E_{i,i}(\mathbf{k})$ contains information about the spectral direction, however in many cases, only the energy at a particular scale $k = |\mathbf{k}|$ is needed to compute certain quantities, such as the dissipation. To lose the directional dependence on the wavenumber, the energy spectrum is integrated over a sphere with radius k, yielding the isotropic energy spectrum,

$$E(k) = \oint k^2 E_{i,i}(\mathbf{k}) d\sigma, \tag{2.57}$$

where σ indicates the solid angle in wavenumber space, giving $d\sigma = \sin\theta d\theta d\Phi$.

The macroscopic description of the energy in physical space has been studied by many, and the energy cascade from the largest to the smallest spatial scales in the flow at which viscous friction dissipates it, as presented in section 2.1.1, is well understood. The work on the spectral description of the energy in isotropic turbulence is credited to the Russian mathematician Andrey Nikolaevich Kolmogorov, who found that the macroscopic energy cascade could be described using the wave number formulation, i.e. the direction of transfer of energy is from the low wavenumbers to the high wavenumbers. In the intermediate range of scales, the so-called inertial subrange, Kolmogorov's hypotheses led to the following universal form for the energy spectrum,

$$E(k) = K_0 \epsilon^{2/3} k^{-5/3}, \tag{2.58}$$

where K_0 is the Kolmogorov constant, ranging in values from 1.4 to 1.8 depending on the Reynolds number of the flow. Equation (2.58) is called *Kolmogorov's $k^{-5/3}$ law*, and is only applicable to the inertial sub-range, i.e. $\eta^{-1} \gg k \gg l^{-1}$, where η is the Kolmogorov length scale and l is the transverse length scale, introduced in section 2.1.1. The general outline of the energy spectrum is visualized in figure 2.5. General equations designed to fit both the inertial sub-range and the dissipative range have been formulated by e.g. Kovasznay (1948); Heisenberg (1948); Smith and Reynolds (1992); Martínez and Kraichnan (1996). These equations are on the form

$$E(k) = K_0 \epsilon^{2/3} k^{-5/3} f(k\lambda), \tag{2.59}$$

where the function $f(k\lambda)$ is designed such that the curve follows the energy drop in the dissipative range, as seen from figure 2.5.

From the turbulent energy spectrum, a range of different quantities can be computed, such as the dissipation rate for the turbulent kinetic energy,

$$\epsilon = 2\nu \int_0^\infty k^2 E(k) dk \tag{2.60}$$

It is clear from equation (2.60) that dissipation is mainly associated with high wavenumbers, i.e. the smallest eddies, even though kinetic energy is associated mainly with lower wavenumbers, i.e. the largest eddies. The integral here includes all wave numbers from 0 to ∞. In a numerical implementation the upper integration limit is $N/2$, where N is the number of nodal points in the discretized mesh.

2.4.3 Initial spectrum

There exist several methods for the initialization of the velocity field. A common factor is that most of them have a stochastic contribution in the method. One frequently method is to generate a field whose energy follows the Kolmogorov scaling from $t = 0\,\mathrm{s}$, and then let it propagate in time. This method would give an initial spectrum on the form

$$E(t=0,k) = \frac{9}{11} k_f^{-1} \times \begin{cases} (k/k_f)^2 & k \leq k_f \\ (k/k_f)^{-5/3} & k > k_f \end{cases} \tag{2.61}$$

where k_f is the highest forced wavenumber. Another method is to generate a flow with a specified amount of total energy, and let the Navier-Stokes equations handle the energy cascade by it self, eventually ending up with the Kolmogorov scaling. The latter method is used here.

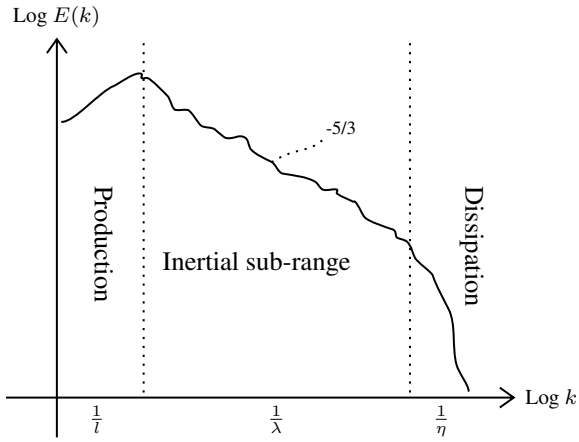


Figure 2.5: Distribution of energy in a turbulent flow field. l denotes the transverse length scale, where most of the energy exist and is produced through e.g. interaction between geometric objects and the flow. λ denotes the intermediate length scale, where neither production or dissipation occurs. η denotes the Kolmogorov length scale, where further breakdown of the eddies leads to energy dissipation as heat.

Following the procedure presented by Rogallo (1981) and Ketcheson et al. (2020), the initial spectrum is defined to be

$$E(t=0,k) = \frac{2C|k|N^3}{(2\pi)^3} e^{-\frac{|k|^2}{a^2}} \tag{2.62}$$

where C is a scaling parameter to increase or decrease the total energy in the initial flow. It is set to be $C = 10^4$, making the total energy in the turbulent flow field equal to the energy in the decaying Taylor-Green vortex which is introduced in section 3.5. The constant a, which represents the node with highest energy in the IC, is set to be equal to 9.5 in accordance with Ketcheson et al. (2020). The velocity field is computed in terms of the wavenumbers and the energy spectra with two stochastic scaling terms. These scaling terms are defined as

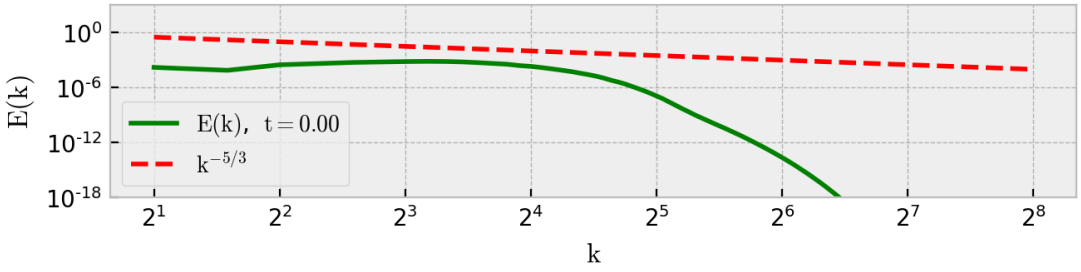
$$\alpha = \sqrt{\left(\frac{E(t=0,k)}{4\pi k^2}\right)} e^{i\theta_1} \cos(\phi) \tag{2.63}$$

$$\beta = \sqrt{\left(\frac{E(t=0,k)}{4\pi k^2}\right)} e^{i\theta_2} \cos(\phi) \tag{2.64}$$

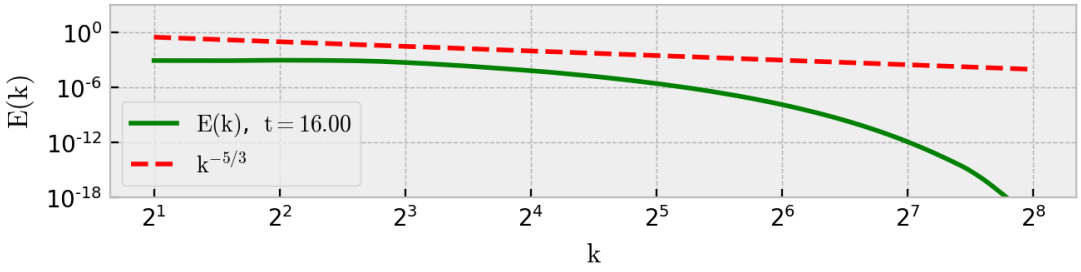
where ϕ, θ_1 and θ_2 are uniformly distributed random numbers on the interval $[0, 2\pi]$. The components of the velocity field are then defined as

$$
\begin{aligned}
u &= \left(\frac{\alpha k k_y + \beta k_x k_z}{k\sqrt{k_x^2 + k_y^2}} \right) \\
v &= \left(\frac{\beta k_y k_z - \alpha k k_1}{k\sqrt{k_x^2 + k_y^2}} \right) \\
w &= \left(\frac{\beta\sqrt{k_x^2 + k_y^2}}{k} \right)
\end{aligned}
\tag{2.65}
$$

Note that the different notations of wavenumbers are: $\boldsymbol{k}^2 = k_x^2 + k_y^2 + k_z^2$, and $k = \sqrt{|\boldsymbol{k}|^2}$. The energy spectrum in the isotropic turbulent flow field at $t = 0\,\mathrm{s}$ with the energy spectrum from equation (2.62) is presented in figure 2.6(a).



(a) $t = 0\,\mathrm{s}$.



(b) $t = 16\,\mathrm{s}$.

Figure 2.6: The initial energy spectrum in an isotropic turbulent flow field is seen in figure 2.6(a). The field is generated by the implementation of equation (2.62). The energy spectrum is also presented for the same flow 16 seconds later, where the energy is distributed more evenly and closer to the highest wavenumbers, corresponding to the smallest spatial scales.

2.4.4 Forcing in the low wavenumber band

A real three-dimensional flow is as explained in previous sections, continuously cascading its energy from the low wavenumbers to the high wavenumbers. Eventually, the dissipation of energy at the highest wavenumbers will make the flow deteriorate, and most likely return to a laminar state before stopping completely. Due to the stochastic nature of turbulent flow fields, when determining statistics of the flow it is advantageous to time- average to reduce statistical variability, which requires statistical stationarity of the flow. In natural occuring turbulent flow fields this stationarity exists due to the equilibrium of energy production at the largest scales, mainly due to geometric features of the flow or other processes acting as energy sources, and dissipation at the lowest scales due to viscous effects. Since there are no processes acting as source terms in the Navier-Stokes equation presented in section 2.2.1, and no geometric features are added to the flow, an energy forcing term must be added artificially in the numerical implementation to sustain the energy level at the largest scales. The assumption is that quantities of interest, such as energy transition in the intermediate sub-range, will not depend on the details of the large-scale dynamics, but only on its gross effects (e.g. the energy-addition rate).

Several forcing schemes have been developed for DNS solvers implemented using spectral methods. A naïve approach is to freeze the lower-wavenumber nodes responsible for the large energy containing scales of the flow at a desired value. Another approach is to maintain the energy in two wavenumber *shells* constant in time, making sure the ratio between the shells were consistent with the $k^{-5/3}$ scaling. Both these methods reach statistical stationarity, yet they don't maintain the anisotropy in the large scales, making it difficult to achieve the desired isotropic statistics. The forcing method implemented in the DNS solver in this thesis follows the procedure presented by Lamorgese et al. (2004); Pope (2000). The method is a deterministic scheme which forces the lower wave numbers with a constant forcing coefficient, $\hat{\boldsymbol{f}}$, scaled by kinetic energy of the same lower wave numbers. In spectral space the forcing coefficient is formulated as

$$\hat{\boldsymbol{f}}_{\boldsymbol{k}}(t) = \frac{P 1_{[|\boldsymbol{k}| \leq k_f]}}{2\kappa_f(t)} \hat{\boldsymbol{u}}_{\boldsymbol{k}}(t), \tag{2.66}$$

where P is the energy injection rate which is updated when the dissipation ϵ changes. $1_{[|\boldsymbol{k}| \leq k_f]}$ is a function which is equal to 1 in the wavenumber shell where $|\boldsymbol{k}| \leq k_f$, and zero otherwise. The cut-off wavenumber k_f is the highest wavenumber which is forced. In this thesis this number is set to be $k_f = 8$, which is similar in magnitude as the forced wavenumbers discussed by, e.g., Lamorgese et al. (2004); Eswaran and Pope (1988); Overholt and Pope (1988). The forcing is normalized by $\kappa_f(t)$ which is the kinetic energy in the same wavenumber shell. This forcing drives the flow toward a statistically stationary state such that $\epsilon = P$. In practice, the coefficients $\hat{\boldsymbol{f}}_{\boldsymbol{k}}(t)$ are computed, and then multiplied with the velocity field in spectral space in each time step.

It takes some time for the flow field to reach a statistically stationary state. By monitoring either the temporal variation in the spectrum or the computed dissipation, it is possible to establish at which time such a state is reached. Depending on the randomly initialized initial conditions, the statistical stationarity is reached at around $16\,\mathrm{s}$ for the system considered here, where the energy spectrum at this time level is presented in figure 2.6. Other forcing methods are presented by, e.g., Eswaran and Pope (1988); Overholt and Pope (1988); Sullivan et al. (1994).

Chapter 3

Numerical methods

In this chapter the numerical methods involved in the spatial discretization and the temporal discretization of the Navier-Stokes equation on rotational form and the advection-diffusion equation is presented. The numerical implementation of the particle method introduced in section 2.2.4 is also presented. An introduction is given to the message passing interface (MPI), enabling communication between computer cores. Three different methods for parallelization are discussed; the 2D pencil decomposition, the slab decomposition and the trivial case of an *embarrassingly parallelizable* task. A section is devoted to the performance of codes run on parallel systems, and the chapter is ended with a discussion of the verification and validation of the DNS solver using the Taylor-Green vortex as benchmarking.

3.1 Spatial discretization

3.1.1 Pseudo-Spectral Fourier-Galerkin method for the Navier-Stokes equation on rotational form

In this section the pseudo-spectral Fourier-Galerkin method, used to solve the spatial part of the Navier-Stokes equation on rotational form, is presented. The implementation of the numerical solver of this equation will in some cases be referred to as the DNS solver. The spectral Galerkin method is considered the workhorse for numerical simulations of homogeneous turbulence. The method has been refined by Rogallo (1981); Brachet et al. (1983). If symmetries exist in the flow, the numerical method can be tailored to take advantage of this in order to reduce the computational effort needed to solve the PDE. One flow with symmetries is the Taylor-Green vortex which is introduced in section 3.5.

Consider the Navier-Stokes equation on rotational form,

$$\frac{\partial \mathbf{u}}{\partial t} - \mathbf{u} \times \boldsymbol{\omega} = -\boldsymbol{\nabla} P + \nu \boldsymbol{\nabla}^2 \mathbf{u}, \tag{3.1}$$

where the variables involved are periodic in both x-, y- and z-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

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

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 wavenumber mesh is represented as

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

To transform between physical space $\boldsymbol{x}$ and spectral space $\boldsymbol{k}$, the discrete Fourier transformations presented in section 2.3.2 are applied to all the terms of equation (3.1). Effectively the method involves taking the inner product of the Navier-Stokes equations with the basis-functions $e^{i\boldsymbol{k}\boldsymbol{x}}$ which is the same as taking the Fourier transformation in all 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 ODEs for $\hat{\boldsymbol{u}}_{\boldsymbol{k}}$,

$$\frac{d\hat{\boldsymbol{u}}_{\boldsymbol{k}}}{dt} - \widehat{(\mathbf{u} \times \boldsymbol{\omega})}_{\boldsymbol{k}} = -\nu |\boldsymbol{k}|^2 \hat{\boldsymbol{u}}_{\boldsymbol{k}} - i\boldsymbol{k}\hat{P}_{\boldsymbol{k}}, \tag{3.4}$$

and the continuity equation reduces to the orthogonal inner product,

$$i\boldsymbol{k} \cdot \hat{\boldsymbol{u}}_{\boldsymbol{k}}. \tag{3.5}$$

The first term on the left hand side is the temporal evolution of the velocity field $\mathbf{u}$ in spectral space. The second term on the left hand side is the nonlinear convective term on rotational form, where the hat notation over the cross product indicates that the term is to be evaluated in spectral space which involves computing a nonlinear convolution sum. The first term on the right hand side is the viscous contribution. Note that the Laplace operator, $\boldsymbol{\nabla}^2$, in spectral space is just multiplication with the squared wavenumber mesh, $\boldsymbol{k}$. The last term on the right hand side is the total pressure in spectral space. For incompressible flows the pressure is no longer a thermodynamic variable, yet it acts as a mechanical force to ensure the enforcement of the continuity equation. By performing the dot product between the quantity $i\boldsymbol{k}$ and the transformed equation (3.4), the total spectral pressure, $\hat{P}_{\boldsymbol{k}}$ can be rewritten to

$$\hat{P}_{\boldsymbol{k}} = -\frac{i\boldsymbol{k} \cdot \widehat{(\mathbf{u} \times \boldsymbol{\omega})}_{\boldsymbol{k}}}{|\boldsymbol{k}|^2}. \tag{3.6}$$

Inserting this total spectral pressure written in terms of velocity, vorticity and the wavenumber mesh, into equation (3.4) yields

$$\frac{d\hat{\boldsymbol{u}}_{\boldsymbol{k}}}{dt} - \widehat{(\mathbf{u} \times \boldsymbol{\omega})}_{\boldsymbol{k}} = -\nu |\boldsymbol{k}|^2 \hat{\boldsymbol{u}}_{\boldsymbol{k}} - \boldsymbol{k} \frac{\boldsymbol{k} \cdot \widehat{(\mathbf{u} \times \boldsymbol{\omega})}_{\boldsymbol{k}}}{|\boldsymbol{k}|^2}. \tag{3.7}$$

The method is called the pseudo-spectral method as not all the terms are computed in spectral space. The nonlinear convective term is evaluated by first transforming the velocity and vorticity to physical space, performing the cross product, and then transforming the vector $(\mathbf{u} \times \boldsymbol{\omega})$ back to spectral space. This method requires two inverse transformations for the velocity and vorticity, and one forward transformation for the cross product. This is done for all three vector components, resulting in 9 DFTs in total to compute the nonlinear convective term, and is considered to be the most computationally extensive

part of the DNS implementation. This part of the numerical implementation is responsible for causing the aliasing errors introduced in section 2.3.4, and a method for removal of said errors is discussed there.

The use of periodic boundary conditions in all spatial dimensions eliminates the need for resolving boundary layers close to walls by assessing the so called y^+ values in the cells neighbouring the walls, which is necessary when using turbulence models such as e.g. the RANS model (Pletcher et al., 2013). The spatial resolution is thus only dependent on the Kolmogorov microscales as presented in section 2.1.3. See Pletcher et al. (2013) for details on treatment of walls and boundary layers for non-periodic problems. It is worth noting that for a two-dimensional implementation of periodic BCs, the computational domain of the PDE is essentially mapped onto a torus. A three-dimensional implementation of periodic BCs is rather hard to visualize, but is essentially a 3-torus.

One remark is that equation (3.7) is only semi-discretized and the temporal discretization remains to be presented. Since the problem now is an initial value problem, the solution of 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, (2.29), 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 (3.7) may be a poor match given the set of parameters involving time step, mesh size and viscosity. In section 3.2 a discussion is given on time integrators suitable to solve the initial value problem presented above.

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 share 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,

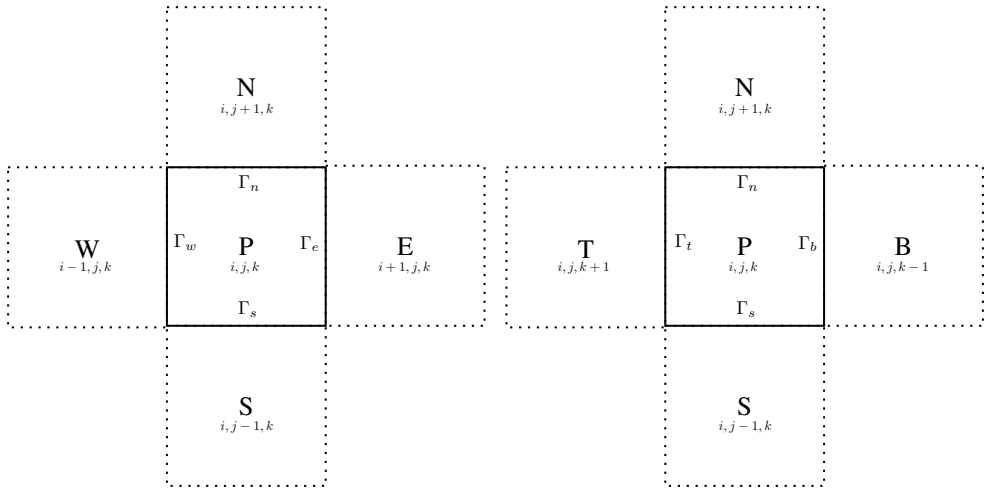
$$\int_V \frac{\partial S}{\partial t} dV = \int_{\partial V} \boldsymbol{f_d} \cdot \boldsymbol{n} dA - \int_{\partial V} \boldsymbol{f_a} \cdot \boldsymbol{n} dA, \tag{3.8}$$

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, Δy and Δy in the x, y and z-direction respectively. In each of these smaller volumes, the cell average is defined to be,

$$\hat{S}(t) = \frac{1}{\Delta x \Delta y \Delta z} \int_V S(t, x, y, z) dx dy dz. \tag{3.9}$$

which inserted into equation (3.8) yields,

$$\Delta x \Delta y \Delta z \frac{d\hat{S}_{i,j,k}(t)}{dt} = \sum_{l=e}^{b} \int_{\Gamma_l} \boldsymbol{f_d} \cdot \boldsymbol{n} dA - \sum_{l=e}^{b} \int_{\Gamma_l} \boldsymbol{f_a} \cdot \boldsymbol{n} dA, \tag{3.10}$$



(a) Two-dimensional slice of the numerical stencil showing the indices i and j.

(b) Two-dimensional slice of the numerical stencil showing the indices j and k.

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

The index notation i, j, k denotes the cell location on the Cartesian grid with equidistant spacing in all spatial directions, i.e., $\Delta x_{i,j,k} = \Delta y_{i,j,k} = \Delta z_{i,j,k} = \Delta x = \Delta y = \Delta z$ for all $i, j, k = 1, \ldots, N$. 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:

$$\int_{\Gamma_e} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j+1/2}}^{y_{j+1/2}} f(t, x_{i+\frac{1}{2}}, y, z) dy dz \approx f_{i+\frac{1}{2},j,k} \Delta y \Delta z \tag{3.11a}$$

$$\int_{\Gamma_w} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = -\int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} f(t, x_{i-\frac{1}{2}}, y, z) dy dz \approx -f_{i-\frac{1}{2},j,k} \Delta y \Delta z \tag{3.11b}$$

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

$$\int_{\Gamma_s} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = -\int_{z_{k-1/2}}^{z_{k+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} f(t, x, y_{j-\frac{1}{2}}, z) dx dz \approx -f_{i,j-\frac{1}{2},k} \Delta x \Delta z \tag{3.11d}$$

$$\int_{\Gamma_t} \boldsymbol{f_d} \cdot \boldsymbol{n} dA = \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} f(t, x, y, z_{k+\frac{1}{2}}) dx dy \approx f_{i,j,k+\frac{1}{2}} \Delta x \Delta y \tag{3.11e}$$

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

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

$$f_{i+\frac{1}{2},j,k} = f(S_{i+1,j,k}, S_{i,j,k}) = D\frac{S_{i+1,j,k} - S_{i,j,k}}{\Delta x} \tag{3.12a}$$

$$f_{i-\frac{1}{2},j,k} = f(S_{i,j,k}, S_{i-1,j,k}) = D\frac{S_{i,j,k} - S_{i-1,j,k}}{\Delta y} \tag{3.12b}$$

$$f_{i,j+\frac{1}{2},k} = f(S_{i,j+1,k}, S_{i,j,k}) = D\frac{S_{i,j+1,k} - S_{i,j,k}}{\Delta x} \tag{3.12c}$$

$$f_{i,j-\frac{1}{2},k} = f(S_{i,j,k}, S_{i,j-1,k}) = D\frac{S_{i,j,k} - S_{i,j-1,k}}{\Delta y} \tag{3.12d}$$

$$f_{i,j,k+\frac{1}{2}} = f(S_{i,j,k+1}, S_{i,j,k}) = D\frac{S_{i,j,k+1} - S_{i,j,k}}{\Delta z} \tag{3.12e}$$

$$f_{i,j,k-\frac{1}{2}} = f(S_{i,j,k}, S_{i,j,k-1}) = D\frac{S_{i,j,k} - S_{i,j,k-1}}{\Delta z}. \tag{3.12f}$$

The same control volume approach can be done on the advective terms in (3.10). By rearranging the terms we get the semi-discretized advection-diffusion equation,

$$\begin{aligned}\frac{dS_{i,j,k}(t)}{dt} = & \frac{D}{\Delta x^2}\left(S_{i+1,j,k} - 2S_{i,j,k} + S_{i-1,j,k}\right) \\ & + \frac{D}{\Delta y^2}\left(S_{i,j+1,k} - 2S_{i,j,k} + S_{i,j-1,k}\right) \\ & + \frac{D}{\Delta z^2}\left(S_{i,j,k+1} - 2S_{i,j,k} + S_{i,j,k-1}\right) \\ & - \frac{u_{i,j,k}}{2}(S_{i+1,j,k} - S_{i-1,j,k}) \\ & - \frac{v_{i,j,k}}{2}(S_{i,j+1,k} - S_{i,j-1,k}) \\ & - \frac{w_{i,j,k}}{2}(S_{i,j,k+1} - S_{i,j,k-1})\end{aligned} \tag{3.13}$$

Since the grid used here is equidistant in all these spatial dimensions the finite volume method and finite difference method are identical in implementation. The FDM approach is recognized in (3.13) 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, Δz, 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, Pe, and to achieve a solution free from oscillations and negative values we have the requirement,

$$Pe = \frac{max(\boldsymbol{u})}{ND} \leq 2, \tag{3.14}$$

where $\boldsymbol{u}$ is the same velocity vector as in (3.8), D is the diffusion constant and N is the number of cells in each spatial direction. For a grid with $N = 512$ cells and $D = 0.0005\,\frac{\mathrm{m}^2}{\mathrm{s}}$, the maximum speed in the velocity field must be lower than $0.512\,\mathrm{m/s}$ to prevent oscillations. The value of the diffusion parameter is about one order of magnitude larger than common values for gases; however, since the largest magnitudes of the velocities in the fields presented in section 4.2 are around $0.5\,\mathrm{m/s}$, the diffusion parameter is chosen to be this high to achieve stability. In order to evade this restriction on these parameters, oscillations in the solution can be reduced or eliminated by including flux-limiters in the FVM approach. An extensive discussion on non-oscillatory methods is given by Hundsdorfer and Verwer (2003). Other restrictions also apply depending on the choice of time integrator, which is discussed in section 3.2. An attempt on using a similar spectral method as for the DNS solver was done on the advection-diffusion equation. However, the method was not successful in achieving stability.

3.1.3 Interpolation to particle position

The particle method is dependent on the velocity field generated by the DNS solver, i.e. the inverse Fourier transform of the solution to equation (3.7). The particles can have any spatial value for the x,

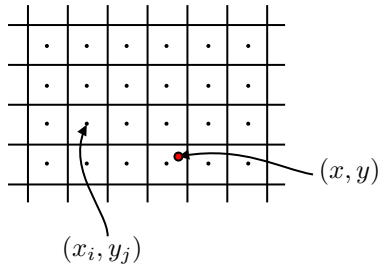


Figure 3.2: A two-dimensional grid with discrete values of a physical variable located in the cell centers, $(x_i, y_j$, and an interpolated value at the arbitrary point (x, y)

y and z positions inside the interval $[0, 2\pi]$, whereas the velocity field is only defined at discrete spatial points

$$\boldsymbol{x}_{ijk} = (x_i, y_j, z_k) = \left\{ \left(\frac{2\pi i}{N}, \frac{2\pi j}{N}, \frac{2\pi k}{N} \right) : \qquad i, j, k \in 0, \ldots, N-1 \right\}. \tag{3.15}$$

A two-dimensional illustration of the discretized grid and continuous particle position is given in figure 3.2. Hence, an interpolation method is required to evaluate the velocity field at arbitrary particle locations (x, y, z). The method implemented here is the linear version of *RegularGridInterpolator* from the Scipy library in Python. This method avoids extensive triangulation of input data by taking advantage of the regular grid structure. A linear interpolation is also called a second order spline interpolation. Note that a spline interpolator of order k is necessary to give continuous derivatives up to order $k-1$. Another available method in the same library is the nearest-neighbour interpolation which essentially locates the nearest discretized velocity to the particle position and uses this value. The nearest-neighbour method is an intuitive approach as the velocity in a given discretized cell is supposed to resemble the average velocity in that region. A thorough introduction of interpolation processes can be found in e.g. Mastroianni and Milovanovic (2000). For an overview of the documentation on the Scipy library, see Virtanen et al. (2020).

3.2 Temporal discretization

In this section the discretization of the temporal term for both the Navier-Stokes equation in spectral space 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 velocity $\mathbf{u}$ or the concentration distribution S is the dependent variable. The equations take the form

$$\frac{d\Phi(t, \mathbf{x})}{dt} = \Psi(t, \Phi) \tag{3.16}$$

where $\Phi(t, \mathbf{x})$ is either $\boldsymbol{u}$ or S, and $\Psi(t, \Phi)$ is the spatially discretized right hand side of either equation (3.7) or equation (3.13). When discretizing the ODE, an explicit method is chosen over an implicit method. Mathematically an explicit approximation to the time derivative in equation (3.16) is

$$\Phi_{n+1} = \Phi_n + \Psi(t_n, \Phi_n), \tag{3.17}$$

whereas the implicit method is defined as

$$\Phi_{n+1} = \Phi_n + \Psi(t_{n+1}, \Phi_{n+1}), \tag{3.18}$$

where the index notation n denotes the time level $n\Delta t$. Implicit methods essentially involves solving a system of equations in each time step, which is of order $\mathcal{O}(N^3)$, and the fine grid needed to resolve the smallest eddies in the DNS solver requires a substantial amount of computational resources. 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 Navier-Stokes equation in spectral space and the advection-diffusion equation is presented in the following sections.

3.2.1 Fourth order Runge-Kutta method

The Navier-Stokes equation on rotaional form in spectral space, equation (3.7), is discretized in time using a Runge-Kutta method referred to as the *classic fourth-order method*. To show how this method differs from other Runge-Kutta methods, an breif introduction is given on the general scheme.

Consider the ODE given in equation (3.16). An explicit Runge-Kutta method takes the form

$$\Phi_{n+1} = \Phi_n + \Delta t \sum_{i=1}^{s} b_i k_i, \tag{3.19}$$

where Δt is a small time step, the index notation n refers to the time propagation of the solution, $\Phi_n = \Phi(t = n\Delta t)$, and coefficients b_i are specific to certain schemes. k_i are evaulations of the spatial part of the ODE, where

$$\begin{aligned}
k_1 &= \Psi(t_n, \Phi_n) \\
k_2 &= \Psi(t_n + c_2\Delta t, \Phi_n + \Delta t(a_{21}k_1)) \\
k_3 &= \Psi(t_n + c_3\Delta t, \Phi_n + \Delta t(a_{31}k_1 + a_{32}k_2)) \\
&\vdots \\
k_i &= \Psi\left(t_n + c_i\Delta t, \Phi_n + \Delta t \sum_{j=1}^{i-1} a_{ij}k_j\right).
\end{aligned} \tag{3.20}$$

The coefficients a_{ij}, b_i and c_i are listed in the so called Butcher tableau presented in table 3.1.

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

Table 3.1: The general Butcher tableau. The entries a_{ij}, b_i and c_i are the coefficients in equation (3.20).

The butcher tableau for the classic fourth-order method presented in table 3.2 and the scheme essentially involves computing the next value Φ_{n+1} using the previous value Φ_n plus a weighted average of four increments in time. k_1 is based on the slope at the beginning of the interval, using Φ_n as argument to the right hand side of equation (3.16). k_2 is based on the slope at the midpoint of the interval, using $\Phi_n + \Delta t k_1/2$. k_3 is also based on the slope at the midpoint. However, it uses the values $\Phi_n + \Delta t k_2/2$. The last increment, k_4 is based on the slope at the end of the interval, and utilizes $\Phi_n + \Delta t k_3$. As revealed by the name, this scheme is fourth-order accurate.

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

Table 3.2: The Butcher tableau of the fourth order Runge-Kutta method.

One remark worth making on the time discretization for the DNS solver is that the diffusion term in equation (3.7) can be treated implicitly with no extra computational cost. To reduce complexity of the numerical implementation, and since the extra stability the aforesaid implicit approach would yield was not needed, this term was treated explicitly in time. A discussion on implicit time integrators in spectral methods can be found in, e.g., (Canuto et al., 1987, p. 209).

3.2.2 Explicit Euler method

The time integrator used to discretize the temporal term in the semi-discretized advection-diffusion equation (3.13) 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

$$\begin{aligned}\frac{S_{i,j,k}^{n+1} - S_{i,j,k}^{n}}{\Delta t} = \quad & \frac{D}{\Delta x^2}\left(S_{i+1,j,k} - 2S_{i,j,k} + S_{i-1,j,k}\right) \\ & + \frac{D}{\Delta y^2}\left(S_{i,j+1,k} - 2S_{i,j,k} + S_{i,j-1,k}\right) \\ & + \frac{D}{\Delta z^2}\left(S_{i,j,k+1} - 2S_{i,j,k} + S_{i,j,k-1}\right) \\ & - \frac{u_{i,j,k}}{2}(S_{i+1,j,k} - S_{i-1,j,k}) \\ & - \frac{v_{i,j,k}}{2}(S_{i,j+1,k} - S_{i,j-1,k}) \\ & - \frac{w_{i,j,k}}{2}(S_{i,j,k+1} - S_{i,j,k-1})\end{aligned} \tag{3.21}$$

This scheme is first order accurate in time and second order accurate in space. Note that the explicit Euler method is a first order Runge-Kutta method with the Butcher tableau presented in table 3.3.

0	0
	1

Table 3.3: The Buthcer tableau of the explicit Euler method.

As explained in section 3.1.2, some restrictions on the parameters in the discretization apply. In addition to the upper limit on relative contribution of advection to diffusion, i.e. the Péclet number, the time step, Δt must be chosen relative to the spatial discretization, $\Delta x = \Delta y = \Delta z$, and the diffusion parameter, D. Following a Von Neumann stability analysis as presented in e.g. Pletcher et al. (2013), it can be shown that this constraint is

$$D\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right) \leq 0.5. \tag{3.22}$$

For a mesh with $N = 512$, i.e. $\Delta x = \Delta y = \Delta z = \frac{2\pi}{512}\,\mathrm{m} = 1.23 \times 10^{-2}\,\mathrm{m}$, a diffusion parameter of $D = 5 \times 10^{-4}\,\frac{\mathrm{m}^2}{\mathrm{s}}$, the time step must be $\Delta t \leq 1.27 \times 10^{-3}\,\mathrm{s}$. In this thesis, a time step of $\Delta t = 1 \times 10^{-3}\,\mathrm{s}$ is chosen for simplicity. In order to use a larger time step for a relatively rapid simulation, either a reduction in diffusion parameter, D, or a larger spatial discretization, $\Delta x = \Delta y = \Delta z$, must be chosen. This further impacts the restriction on the velocity.

A Lax-Friedrichs integrator was also implemented for the advection-diffusion equation. This method adds a numerical viscosity, smearing out unwanted discontinuities, and effectively making the method more stable compared to that of an explicit Euler method. Adding a numerical viscosity is a common method to achieve stability for advection dominated problems, however, it was deemed too diffusive as there already is a diffusive element in the advection-diffusion equation.

As seen in the discretized version of the SDE (2.33), used in the particle method, an explicit Euler method was used to integrate the temporal term. A higher order method would increase the accuracy, however, the increased computational cost needed to interpolate the velocity field at different increments in time, was found to be too expensive, mainly due to the naïve implementation of the parallelized particle method, which will be discussed in section 3.3.5.

3.2.3 Initial conditions for the advection-diffusion equation and the particle model

The IC for the advection-diffusion equation is a multivariate normal distribution with variance $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$ and mean $\mu = \pi\,\mathrm{m}$. It is defined in the following way,

$$S(t=0,\boldsymbol{x}) = \exp\left(-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2} + \frac{(y-\mu)^2}{\sigma^2} + \frac{(z-\mu)^2}{\sigma^2}\right)\right). \tag{3.23}$$

After the initialization of the distribution S, it is normalized with respect to the total sum of the values in all cells in the mesh, such that the total computational "mass" is equal to one. Figure 3.3 presents the IC visually for the xy-plane with $z = \pi$.

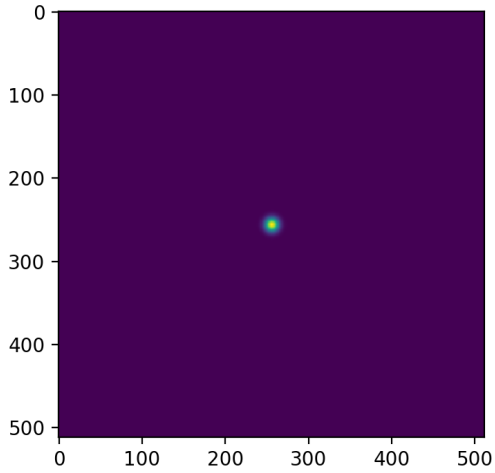
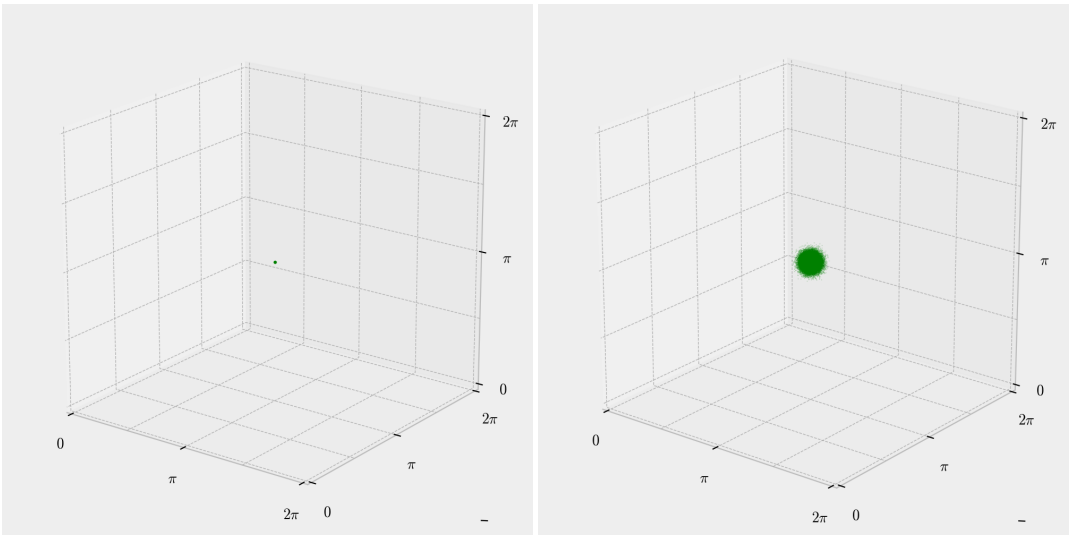


Figure 3.3: Two-dimensional slice of the initial condition for the advection-diffusion equation. The slice is in the xy-plane with $z = \pi$.

The particle has two ICs. One where all the particles are located in the center of the computational domain, i.e., $(x, y, z) = (\pi, \pi, \pi)$. This IC is presented visually in figure 3.4(a). The second IC is a particle distribution where the initial coordinates are random numbers drawn from a multivariate distribution with variance $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$ and mean $\mu = \pi\,\mathrm{m}$. This IC is presented visually in figure 3.4(b). Results from the numerical simulation using both IC are presented in section 4.3; however, only the simulation using the second IC will be compared to the advection-diffusion equation due to the limited computational resources available as discussed in section 4.3.

3.3 Implementation on parallel system

In this section an overview is given on the parallel implementation of the discretized Navier-Stokes equations in spectral space, the discretized advection-diffusion equation, and the particle method. The reason a considerable portion of the thesis is devoted to the parallelization of scientific code is twofold.



(a) Initial condition where all particles are located in the point $(x, y, z) = (\pi, \pi, \pi)$

(b) Initial condition where the position of the particles are random numbers drawn from a multivariate distribution with variance $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$ and mean $\mu = \pi\,\mathrm{m}$.

Figure 3.4: Initial condition for the particle method for two different simulation setups.

Reason number one is that large, complicated computational tasks often take up a great bulk of the memory available in a single processor. The ability to distribute this memory opens up the possiblity to solve even more compliacted and demanding tasks. The second reason is speed. A single computing core is physically limited and the amount of floating point operations (FLOP) can only be as high as the theoretical maximum output of the processor. To increase the speed, either a change of algorithm to reduce the number of operations, or an increase in computer core count is necessary. In many cases the algorithms have been improved on for years, and little to no enhancements can be made, making parallelization of the task the next trivial step. For a detailed discussion on parallel computing and the relevant hardware of high-performance-computers, see Pacheco (2011).

3.3.1 Idun cluster and Vilje

Most of the simulations presented in this thesis was run on the NTNU IDUN computing cluster. Själander et al. (2019) presents an extensive overview of this systems architecture and some of the research outcome. Only a brief summary is given here. The cluster has more than 70 nodes and 90 General-purpose computing graphics processing units (GPGPUs). Each node contains two Intel Xeon cores, at least 128 GB of main memory, and is connected to an Infiniband network. Half of the nodes are equipped with two or more Nvidia Tesla P100 or V100 GPGPUs. Idun's storage is provided by two storage arrays and a Lustre parallel distributed file system.

The supercomputer Vilje was used for larger simulations utilizing more computer cores. Vilje is a SGI Altix ICE X system which NTNU, UNINETT Sigma2 and The Norwegian Meteorological Institute (MET) acquired in 2012, used mainly for numerical weather prediction in operational forecasting by

MET, however it also serves researchers at NTNU, other Norwegian universities and research institutes. The system holds 19.5 racks with a total of 1404 nodes, where each node has 16 physical cores and 32 logical cores through hyperthreading. This adds up to 22464 physical cores working at a theoretical peak performance of 467 Teraflop/s. The systems interconnect is a Mellanox FDT infiniband, Enhanced Hypercube Topology. PBS is used as job scheduler. An overview of Vilje's specifications is listed in NOTUR (2011).

An important distinction between the two systems is that Vilje has a memory of $32\,\mathrm{GB}$ per node, whereas Idun has at least $128\,\mathrm{GB}$ per node. As the implementation of the different numerical solver required access to a large amount of memory, the capacity of Vilje was in some cases insufficient. For the pencil decomposed velocity field in the DNS-solver to be broadcasted to a slab decomposed field in the advection-diffusion equation solver, the field had to be gathered and redistributed. Additionally, the implementation of the particle method required the velocity field to be fully available to every computer core. It is possible to schedule a job using several nodes and their memory capacity without booking all of the cores, however, this in practice results in longer queue times.

3.3.2 Message Passing Interface

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 distribution of the workload amongst the computer cores is done 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).

3.3.3 2D Pencil decomposition

The costly computational tasks in the DNS solver are the FFTs performed along the different axes of the mesh. These transformations are often the bottleneck of a scientific code, and substantial contributions are done in the field each year to improve both the serial FFT algorithm and its parallel implementation, see e.g. Aznag et al. (2020). The FFT on multidimensional data can be performed as a sequence of one-dimensional transformations along each of the spatial dimensions. For instance, a 3D array with shape $N_x \times N_y \times N_z$ can be Fourier-transformed by first performing $N_x \times N_y$ serial transformations of length N_z along the z-axis, preceded by $N_x \times N_z$ transformations of length N_y along the y-axis, and lastly $N_y \times N_z$ transformations of length N_x along the x-axis.

For problems with large computational domains such as fully resolved turbulent flows, the data is often too large to fit in the local memory of a single computer core. Consequently, the domain must be decomposed such that only a small section of the 3D array is available on each core. To take advantage of the large number of cores on both the Idun and Vilje computing clusters, the mesh used in the DNS solver is split into smaller domains using a 2D pencil decompositio. These smaller domains are often

referred to as *pencils*, giving the method its name, and they open up for utilization of N^2 cores for a mesh of size N^3. Note that this is the theoretical maximum number of cores, and the performance of the code is notably reduced in this limit (Pacheco, 2011).

In a 2D pencil decomposition, the data along two of the axes in the multidimensional array are distributed amongst the cores in the computer, e.g. P_1 cores along the first axis and $P_2 = \frac{\#\text{Cores}}{P_1}$ cores along the second axis, resulting in a local computational domain in physical space with dimensions $N_1 \times N_2 \times N$, where $N_1 = \frac{\#\text{Cores}}{P_1}$, and $N_2 = \frac{\#\text{Cores}}{P_2}$. The cores distributed along the first axis and second axis will sometimes be referred to as processor group P_1 and P_2 respectively. The next step is to perform FFTs of the data along the different axes while making sure all the necessary information is available to each core. There exist several packages on the market that take care of both the transformation and communication between the different cores, e.g. P3DFFT and 2DECOMP&FFT, see Pekurovsky (2012); Li and Laizet (2010) for documentation. These two are the most common packages for FFTs on a 2D pencil decomposed mesh. For higher-dimensional arrays than a three-dimensional mesh, the PFFT package can be used, see Pippig (2013). In this thesis, the 2D pencil decomposition is implemented for the DNS solver following a procedure presented in Dalcin et al. (2019). The physical mesh, the fully transformed complex mesh and its intermediate states are shown in figure 3.5.

Consider the computational domain in physical space, u_{r_1,r_2,r_3}, where the indexes r_1, r_2 and r_3 indicates the axis in x-, y- and z- direction with real data points. This computational domain is divided into smaller local computational domains where the r_1 axis is distributed amongst processor group P_1, and the r_2 axis is distributed amongst the processor group P_2. The distributed computational domain has the form $u_{r_1/P_1,r_2/P_2,r_3}$. To transfer this domain from real physical space to a complex domain in three dimensions in terms of Fourier transformations, the following procedure is applied:

1. The first step is to transform the real physical mesh along the last axis, i.e. the r_3 axis, since the physical mesh is continuous here, as seen in figure 3.5(a).

$$\tilde{u}_{r_1/P_1,r_2/P_2,c_3} = \mathcal{F}_3(u_{r_1/P_1,r_2/P_2,r_3}), \tag{3.24}$$

   where $\mathcal{F}_i$ indicates the forward Fourier transformation along the i'th axis. Since the transformation is along the r_3 axis, the new third axis is complex, denoted by c_3. The output from the transformation is the partially real and complex mesh seen in figure 3.5(b). The third axis has length $\frac{N}{2}$ since the transformation is done on purely real values. Actually the last axis should be of length $N/2+1$. The data stored in this part of the mesh is sometimes referred to as the *Nyquist frequency*, and this frequency is neglected to maintain the capability of distributing the information amongst an even number of processors. The neglection of the *Nyquist frequency* is not uncommon in turbulence simulations, as it is a Fourier mode that is not carried in the Fourier representation of the solution, as explained by Lee et al. (2014).

2. The next step is to perform a global redistribution in order to align the mesh such that it is continuous along the second axis. Essentially the information each computer core holds is redistributed along the first and last axis, making the second axis continuous. The layout of the processor distribution along the mesh is seen visually when going form figure 3.5(b) to figure 3.5(c).

$$\tilde{u}_{r_1/P_1,r_2,c_3/P_2} \xleftarrow[P_2]{3\to 2} \tilde{u}_{r_1/P_1,r_2/P_2,c_3} \tag{3.25}$$

3. Once the computational domain is aligned along the second axis, a second Fourier transformation is performed on the new global array, resulting in a complex second axis.

$$\tilde{u}_{r_1/P_1,c_2,c_3/P_2} = \mathcal{F}_2(\tilde{u}_{r_1/P_1,r_2,c_3/P_2}). \tag{3.26}$$

4. Again, the information must be redistributed to prepare for the transformation along the first axis. Hence, the P_1 processor group is distributed along the second axis, making the first axis continuous.

$$\tilde{u}_{r_1,c_2/P_1,c_3/P_2} \xleftarrow[P_1]{2\to 1} \tilde{u}_{r_1/P_1,c_2,c_3/P_2} \tag{3.27}$$

5. Finally, the last transformation is performed along the x-axis, making the initially real valued computational domain complex with processor groups P_1 and P_2 distributed along the second and third axis, denoted by c_2 and c_3.

$$\tilde{u}_{c_1,c_2/P_1,c_3/P_2} = \mathcal{F}_1(\tilde{u}_{r_1,c_2/P_1,c_3/P_2}) \tag{3.28}$$

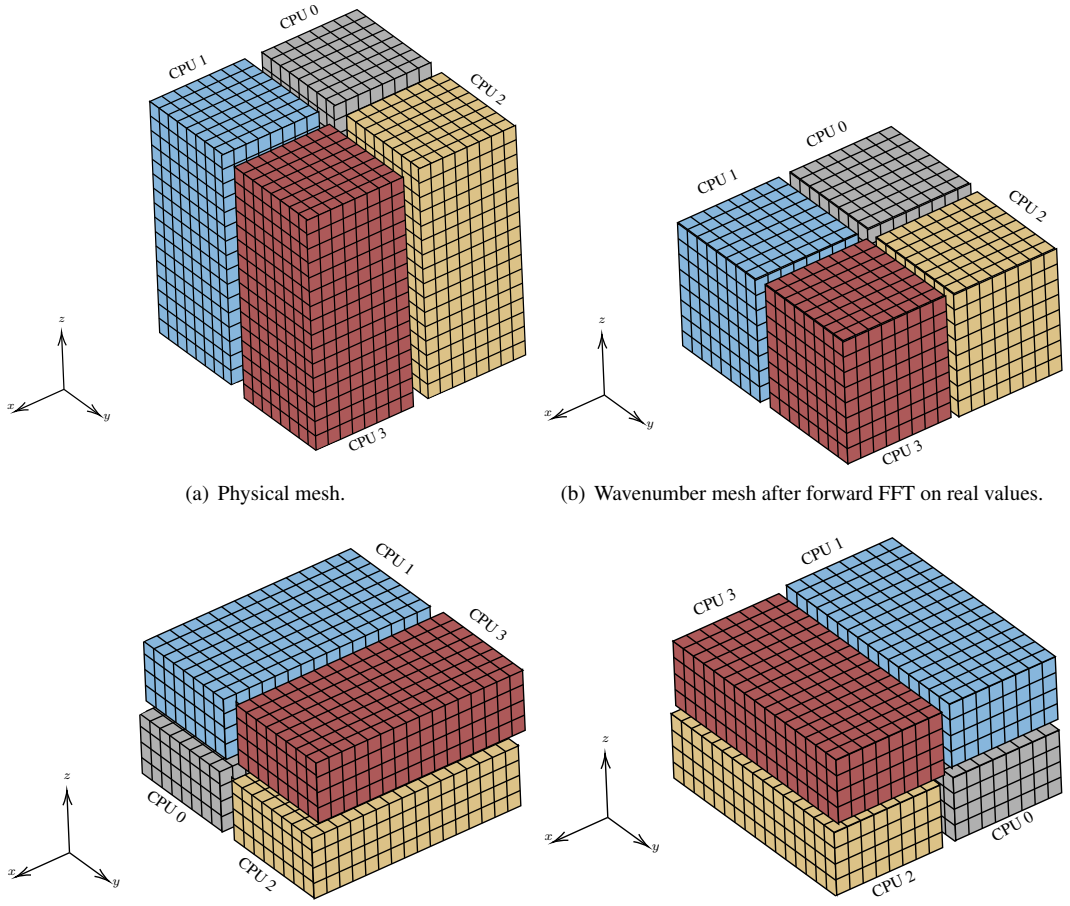
The final transformed mesh is seen in figure 3.5(d). The global redistribution performed in step 2. and 4. combine the function *Rollaxis*() from the Numpy package, see Oliphant (2006), and *alltoall*() from the mpi4py package. A complete and thorough description of the procedure, as well as an extensive performance analysis, can be found in Dalcin et al. (2019). The performance of the 2D pencil decomposition on the Vilje computing cluster is investigated and discussed in section 4.1.

3.3.4 1D Slab decomposition

The grid used in the advection-diffusion equation is divided into equally large portions between the processes as seen in figure 3.6, which means that the number of grid points N must be divisible by the number of assigned cores. Each portion is named a local field. The local fields have an extra layer of data located in the first and last plane aligned with the z-axis, which is used to store the information received from the neighbouring local domains. These layers are referred to as *ghost boundaries* or *boundary halos*, see, e.g., Pletcher et al. (2013); Versteeg and Malalasekera (2007). 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 $(N \times N \times 1)$. If the stencil were to use both the first and second neighbouring points in all directions, e.g. from a higher-order spatial discretization, the ghost boundary must include a second plane, $(N \times N \times 2)$, to store the extra information needed. The distribution amongst the different ranks is presented in figure 3.6. The ghost boundaries of each local domain are presented in different colors. Planes shown in the same color send information to the same rank, i.e. the blue plane containing information from rank 1 and 3, both send their information to rank 2. The slab decomposition is proven to be more efficient than the pencil decomposition, discussed in Dalcin et al. (2019). However, due to the strict limitation on the maxiumum of N cores in a $N \times N \times N$ slab decomposed mesh, the need for a 2D decomposition such as the pencil decomposition is present for computationally extensive problems.

3.3.5 Embarassingly parallelizable particle method

To parallelize the particle method presented in section 2.2.4, the array containing information on the particle position is arranged in the way shown in table 3.4,
which is an array of shape $(m, 3)$, where m is the number of particles in the simulation and 3 denotes the three spatial dimensions. The initial position of the particles is here chosen to be $x_{t=0} = z_{t=0} = z_{t=0} = \pi$ for simplicity. However, in the actual simulations the particles are chosen randomly such that the collected position of the particles fit a normal distribution as presented in section 2.2.4. This array is used as input to an ODE solver. By assigning a portion of the first axis, i.e. a portion of the m particles



(a) Physical mesh.

(b) Wavenumber mesh after forward FFT on real values.

(c) Intermediate wavenumber mesh.

(d) Final wavenumber mesh.

Figure 3.5: 2D pencil decomposition of a physical mesh with real values is presented in 3.5(a), and the intermediate stages and the final wavenumber mesh is presented in 3.5(b), 3.5(c) and 3.5(d). For illustration purposes the sketched 2D pencil decomposition uses 4 computer cores. They are distributed equally amongst two of the axes. These axes are always normal to the direction of the FFT; hence, a redistribution of the computer cores is necessary when performing the FFT along another axis.

to each computer core, the particle method can be run in parallel with no other MPI communication than at the very start, to scatter the array amongst the computer cores, and in the end, to gather the data for post-processing. Methods that involve little to no MPI communication is often referred to as an *embarrassingly parallelizable* method. The drawback when using this method is that each computer core must store the entire velocity field from the DNS solver in its local memory. This is due to the velocity field being distributed amongst computer cores in two spatial dimensions, whereas in the particle method individual particles move around the computational domain with no spatial restrictions. Since the propagation of the particle position in time is dependent on an interpolated value of the velocity field at

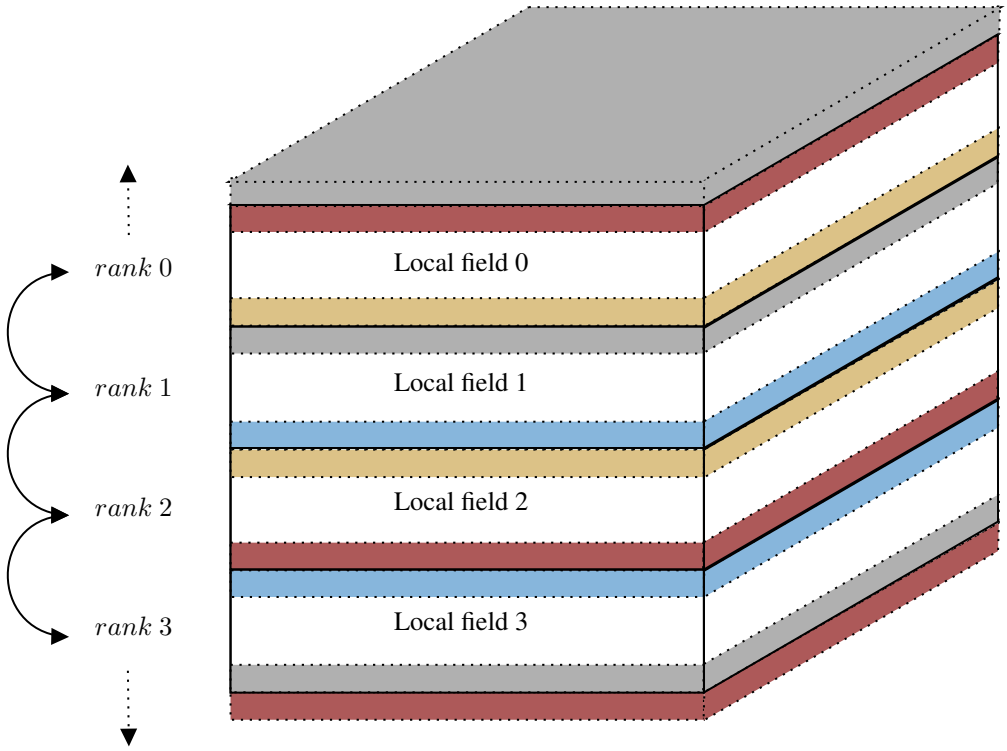


Figure 3.6: A computational domain for the advection-diffusion equation divided amongst four cores. The local field is the partial computational domain were each process computes a part of the solution. The information needed at the boundaries of each local field is sent through MPI communication using ghost boundaries indicated by color. This computational domain is periodic in all spatial directions, which means that there is communication between rank 0 and rank 3 as well.

the particles previous position, the entire velocity field must be stored. By implementing a more extensive communication routine such that the particles move back and forth between computer cores depending on their position, the storage of the total velocity field could be avoided. However, this method is more computationally demanding due to increased communication, in addition to being more complicated to implement.

3.4 Performance of parallel implementation

In section 3.3 the challenges with memory and speed were introduced to be the main purposes of utilizing the parallel architecture of a system. Overcoming these challenges will in many cases increase the overall performance of the code, and the following section introduces several laws or metrics used to quantify how well a parallel program runs on multiple computer cores.

The best possible performance happens when the work is equally divided amongst the cores while at the same time introducing no additional work for each core due to communication. This extra work

Particle #	x-position	y-position	z-position
1	π	π	π
2	π	π	π
3	π	π	π
$\vdots$	$\vdots$	$\vdots$	$\vdots$
m	π	π	π

Table 3.4: Initial position of the m particles. All particles are located in the point $(x, y, z) = (\pi, \pi, \pi)$.

will be referred to as *parallel overhead*. In this scenario, a program run with p cores will run p times faster than the serial program which corresponds to 1 computer core. If the serial run-time is T_{serial} and the parallel run-time is $T_{parallel}$, the best execution time will be $T_{parallel} = T_{serial}/p$. This is called a *linear speedup*. Practically, a linear speedup is unlikely due to the invariable introduction of parallel overhead. Distributed-memory programs will in most cases need to send and receive data between cores, which is usually much slower than accessing the local cache of each core (Pacheco, 2011). Furthermore, it's likely that the overhead increases as the number of cores, p, increases. The communication between cores is redundant in serial programs as they are run using only one computer core.

The general speedup of a parallelized program is the ratio of the serial run-time and parallel run-time,

$$S = \frac{T_{serial}}{T_{parallel}}, \tag{3.29}$$

where a linear speedup corresponds to $S = p$. The speedup, S, will in most cases be measured as a smaller and smaller portion of the ideal speedup as the number of cores increases. The ratio of the measured speedup and the ideal speedup is referred to as the *efficiency*, defined as

$$E = \frac{S}{p} = \frac{\left(\frac{T_{serial}}{T_{parallel}}\right)}{p} = \frac{T_{serial}}{p \cdot T_{parallel}}. \tag{3.30}$$

It is clear from the relations above that S, E and $T_{parallel}$ all depend on the core count, p. What is less obvious is the dependency on the size of the problem which the program is intended to solve. For instance, a DNS solver with a mesh of size $N = 64$ requires less FLOP than a mesh of size $N = 2048$, and hence have a faster run-time. However, the larger mesh will benefit from a high number of computer cores.

A connection between the speedup, efficiency and parallel overhead can be seen from the relation

$$T_{parallel} = T_{serial}/p + T_{overhead}, \tag{3.31}$$

where the extra time a program uses due to parallel overhead is denoted by $T_{overhead}$. From equation (3.31) it is clear that if $T_{overhead}$ grows slower than T_{serial} when the problem size is increased, the efficiency and speedup will increase. There is more work for the processes to do which makes the relative amount of time spent coordinating the work of the processes less significant.

In the next couple of subsections we use a notation where T_p is the execution time of the parallel part of a program, and $T(p)$ is the total execution time of a code run with p processors. Hence, $T(1)$ is the total execution time of a code run with $p = 1$.

3.4.1 Amdahl's law

Estimating the limitation of the speedup when modifying a code to run in parallel is an enlightening exercise, and was once done by Gene Amdahl in the 1960s. His result, now known as Amdahl's law, states that if a fraction $(1 - r)$ of the serial version of the program remains unparallelized, then the speedup will not increase beyond $1/(1-r)$. This does not take into account e.g. the problem size, which in many cases will reduce the serial fraction of the program. Generally, the law gives a theoretical speedup of the execution of a program where the amount of work is constant, and the number of processors increases. The total runtime, T, of a program can be divided into a serial part and a parallel part,

$$T = T_p + T_{serial}. \tag{3.32}$$

If r denotes how large fraction of the program is parallelized, and $(1-r)$ denotes the unparallelized part, equation (3.32) can be rewritten to

$$T = rT + (1 - r)T, \tag{3.33}$$

which after the allocation of p additional cores yields the following relation for the total runtime

$$T(p) = \frac{r}{p}T + (1 - r)T. \tag{3.34}$$

As stated in equation (3.29), the resulting speedup is the proportion of the serial runtime relative to the parallel runtime,

$$S_{\text{Amdahl}}(p) = \frac{T}{T_p} = \frac{1}{(1 - r) + \frac{r}{p}}. \tag{3.35}$$

From equation (3.35) it is clear that $\lim_{p\to\infty} S_{\text{Amdahl}} = \frac{1}{(1-r)}$, which is exactly what Amdahl's law states. Alternatively, if the parallelized fraction, r, approaches 1, then $S_{\text{Amdahl}}(p) = p$, which is the ideal speedup.

3.4.2 Gustafson's law

A drawback in Amdahl's law is its presumption of a fixed problem size, i.e the workload in the program does not change when the amount of resources available increase. John L. Gustafson and Edwin H. Barsis proposed a law for speedup in a parallel program where the execution time is kept constant, and the workload increased as the number of processes increases. The idea is that as more resources become available, larger and more complicated problems are solved in order to fully exploit the computing power. Their theory, known as Gustafson's Law, gives the factor by which the resources need to be scaled if the problem size is increased in order to maintain a constant execution time. If the serial and parallel part of the program run on a parallel system are s_t and p_t respectively, the speedup from Gustafson's Law is

$$S_{\text{Gustafson}}(p, s_t) = p + (1 - p)s_t. \tag{3.36}$$

Since the serial and parallel fractions sum up to $s_t + p_t = 1$, (3.36) can also be written in terms of the parallel time fraction,

$$S_{\text{Gustafson}}(p, s_t) = p + (1 - p)(1 - p_t). \tag{3.37}$$

Note that $S_{\text{Gustafson}}(p, p_t) \approx p$ when the problem size increases to the point where $(1 - p_t) \approx 0$, i.e the parallel time fraction is close to 1.

3.4.3 Karp-Flatt metric

The Karp-Flatt metric is a measure of how well a code is parallelized. It relates the speedup, S, of the code when the program is run using p processors to find out the experimental serial fraction, s_t. By deriving the metric in terms of equation (3.34), it is demonstrated that it is consistent with Amdahl's law,

$$T(p) = \frac{T_{parallel}}{p} + T_{serial}, \tag{3.38}$$

and by substituting in the serial time fraction $s_t = \frac{T_s}{T(1)}$, where $T(1)$ represents the total execution time of a program when the core count $p = 1$, yields,

$$T(p) = T(1)s_t + \frac{T(1)(1 - s_t)}{p}. \tag{3.39}$$

By also substituting in for the speedup introduced in equation (3.29),

$$\frac{1}{S} = s_t + \frac{1 - s_t}{p}, \tag{3.40}$$

and solving for the serial fraction, s_t, gives the final metric

$$s_t = \frac{\frac{1}{S} - \frac{1}{p}}{1 - \frac{1}{p}}. \tag{3.41}$$

By letting the speedup S approach the ideal speedup, p, equation (3.41) approaches zero. This indicates that as the parallel fraction of a code increases, s_t decreases, making it a good metric to measure this ratio.

3.5 Verification and validation

In the preceding chapters an overview of the modeling of fluid flow through the Navier-Stokes equations and the numerical solution of them obtained through discretization in spectral space was presented. The combination of a mathematical model and its discretization introduce errors in the numerical solution. The influence such errors have on the flow field must be evaluated. Ideally the mathematical model should describe all the features of the system it is meant to mirror, and the discretization of the relevant equations should be of such a fine scale that it eventually resembles the continuous model. The procedure in which the credibility in the mathematical model and its discretization is established is called verification and validation. The American Institute of Aeronautics and Astronautics (AIAA) (Computational Fluid Dynamics Committee, AIAA, 2002) defines the two constituents of this procedure in the following way:

> **Verification**: the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.

> **Validation**: the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

The objective of the verification process is to verify that the equations in the mathematical model are solved correctly. This is done by checking convergence of certain attributes of the solution while refining the mesh, i.e. include more nodal points in the discretized equations. The aforementioned attributes are e.g. residuals, number of iterations or a result like the dissipation of the flow field. Since the spectral implementation of the Navier-Stokes equations is not found through iteration of the solution, the convergence must be checked through the other attributes.

The validation process is as described in the definition by the AIAA, a test to make sure that the right equations are solved, i.e. the mathematical model should correctly mimic the natural system it aims to model. This is checked by comparing the numerical solution with experimental data.

To verify and validate the numerical solution from the DNS-code, a well known initial condition used for benchmarking similar codes is utilized, the Taylor-Green vortex. The Taylor-Green vortex is an unsteady, decaying vortex which for the incompressible Navier-Stokes equations has an exact closed form solution. Its initial velocity field $\mathbf{v}(t=0) = \mathbf{v}(t=0, u, v, w)$ is given by

$$u = sin(x)cos(y)cos(z) \tag{3.42}$$

$$v = -cos(x)sin(y)cos(z) \tag{3.43}$$

$$w = 0. \tag{3.44}$$

The Taylor-Green vortex has been thoroughly studied, and many of its features are well documented, e.g. its kinetic energy spectrum and dissipation of kinetic energy. A simulation is run with parameters matching those presented in Koen and Cagnone (2017), i.e. $Re = 1600$ and $N = 512$. A two-dimensional cross section whose normal vector aligns with the z-axis is used to generate a snapshot of the flow field in the given cross section for four different time steps. The images are presented in figure 3.8. The velocity fields initially show large-scale structures, corresponding to low wave number energy. The large-scale structures break down to form smaller eddies through vortex-stretching as explained in section 2.1

The kinetic energy spectra, computed from the velocity fields at the same time steps using equation (2.57) are presented in figure 3.9. The initial spectrum shown in figure 3.9 verify that the energy is mainly in the low wavenumber region, i.e. the large-scale structures in the flow. As the numerical solution is propagated in time the energy is seen to distribute throughout the rest of the wavenumbers, following Kolmogorov's scaling in equation (2.58) as presented in section 2.4.2, illustrated by the dashed red line. The region which follow this theory is commonly labeled as the *inertial sub-range*. The high wavenumber region corresponding to the smallest structures in the flow is losing energy faster than the $-5/3$ scaling, and is labeled the *dissipation range*.

For the smallest eddies in the flow, the vortex-stretching term in the Navier-Stokes equations (2.29), responsible for creating smaller and smaller eddies, yields to the viscous term. The energy in these eddies is instead dissipated as heat to the system. The energy dissipation is computed at several time steps for the Taylor-Green vortex and is presented in figure 3.7. A dataset from the annual Workshop on High-Order CFD methods, see Koen and Cagnone (2017), is added for validation of the dissipation values obtained from the DNS code used in this project. The values overlap well in the interval $t \in [0\,\mathrm{s}, 20\,\mathrm{s}]$, where the reference data is available. The peak value of dissipation occurs at $t = 9\,\mathrm{s}$ for both datasets. However, the dissipation data used in this project is sampled at a lower rate, and the true peak of dissipation is not fully captured, thus resulting in a truncated maximum. A discussion on the energy spectrum and energy

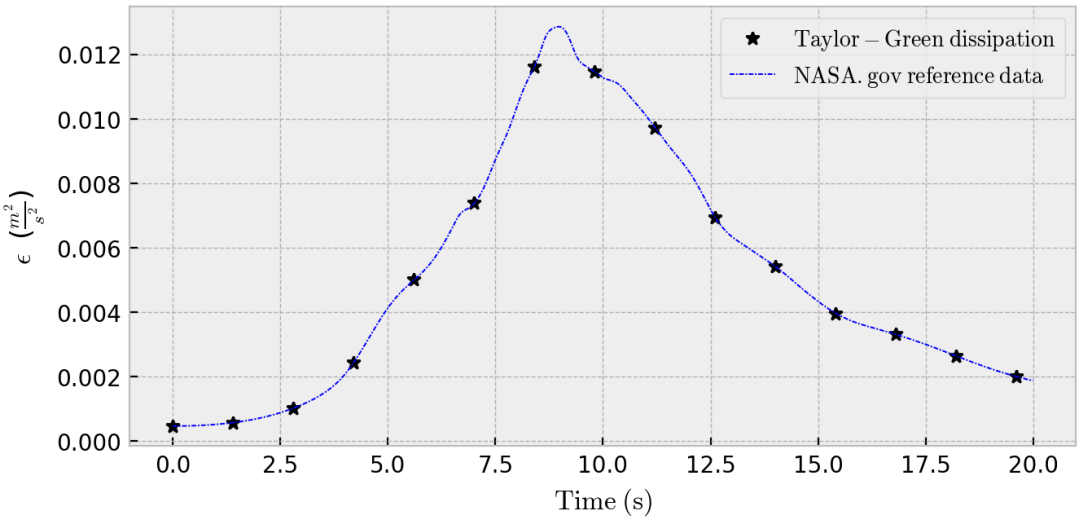
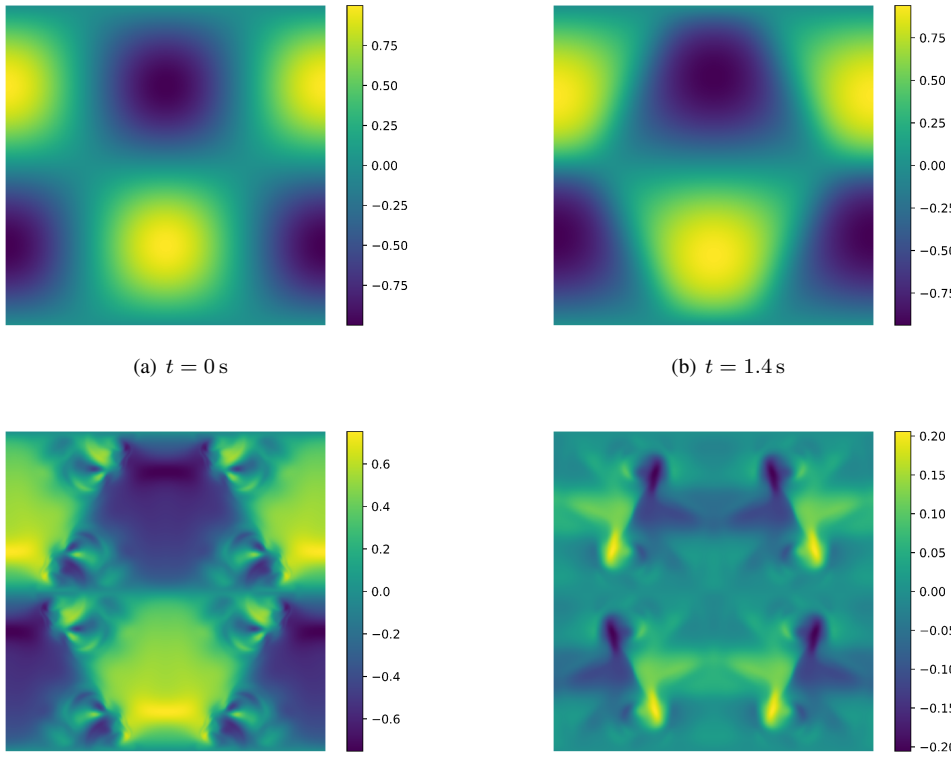


Figure 3.7: Dissipation, ϵ, in the decaying Taylor-Green vortex at a range of different time steps. Numerical data from Koen and Cagnone (2017) is added for comparison.

dissipation in the Taylor-Green vortex with $Re = 1600$ is given by, e.g., Brachet (1991); Brachet, Marc E. and Meiron, Daniel I. and Orszag, Steven A. and Nickel, B. G. and Morf, Rudolf H. and Frisch, Uriel.



(a) $t = 0\,\mathrm{s}$ (b) $t = 1.4\,\mathrm{s}$

(c) $t = 9.8\,\mathrm{s}$ (d) $t = 42\,\mathrm{s}$

Figure 3.8: Four snapshots of the temporal evolution of the velocity field in a decaying Taylor-Green vortex. The velocity values are indicated with yellow for positive values and dark blue for negative values.

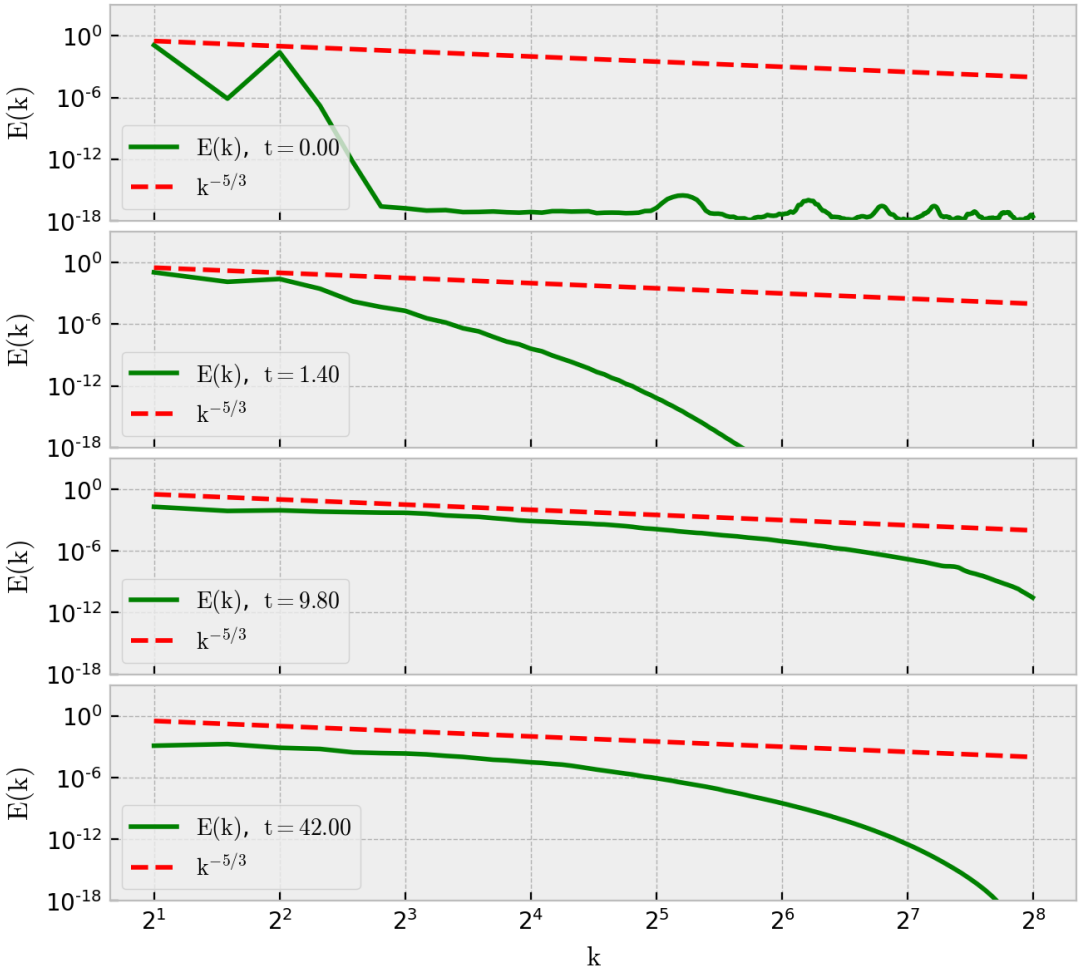


Figure 3.9: Energy spectrum, $E(k)$, in the same flow and time steps presented by visual snapshots in figure 3.8, i.e. $t = 0\,\mathrm{s}$, $t = 1.4\,\mathrm{s}$, $t = 9.8\,\mathrm{s}$ and $t = 42\,\mathrm{s}$. It is clear that the majority of initial energy in the Taylor-Green vortex is located in the lower wavenumbers, i.e. the largest spatial scales. Eventually, the energy is distributed amongst all of the spatial scales, as can be seen at the time level $t = 9.8\,\mathrm{s}$.

Chapter 4

Results and Discussion

This chapter is devoted to the presentation and discussion of the results related to the numerical solution of the particle model, the Navier-Stokes equation in spectral space and the advection-diffusion equation, i.e., equation (2.32), (3.7) and (3.21). The first section concerns the performance of the parallel implementation of the DNS solver, presenting the metrics for performance introduced in section 3.4. The second section presents various statistics and visualizations of an isotropic turbulent flow field outputted from the DNS solver at various time steps. The third section presents the numerical solution of the particle model and the variance statistics of the particle positions. Visualizations of the complete ensemble of particles as well as a trajectory-plot of a selection of four individual particles is presented. The fourth section presents the numerical solution of the advection-diffusion equation and the variance statistics of the solution field, S. Recall that the variance of both the particle positions and the distribution S from the advection-diffusion equation are directly connected to the effective mixing parameter, K through Richardson's law, equation (2.4). Note that the maximum simulation time is varying in the individual simulations from $t \approx 20\,\mathrm{s}$ up to $t \approx 80\,\mathrm{s}$. The simulation time is stated in each section.

The code in which the numerical solvers are implemented is presented on the author's GitHub profile[1]. A selection of results are animated for a more visual presentation and displayed on the same web page.

4.1 Parallel performance

The performance of the DNS solver is quantified using the metrics introduced in section 3.4, i.e., the speedup, S, Efficiency, E and the Karp-Flatt metric, all dependent on the measured walltime when computing the spatial terms of equation (3.7). Note that the performance is measured on a simplified version of the DNS solver. For instance, the pencils containing the local velocity field in each computer core are not gathered to make the full velocity field. This omits a large portion of communication that is present when computing the velocity field, and the numerical solutions to the particle model and advection-diffusion equation simultaneously. Moreover, there is no post-processing or storing of data to disk. The most complicated tasks are the computation of the convective terms, which involve both the

[1] `https://github.com/danielhalvorsen/Project_Turbulence_Modelling`

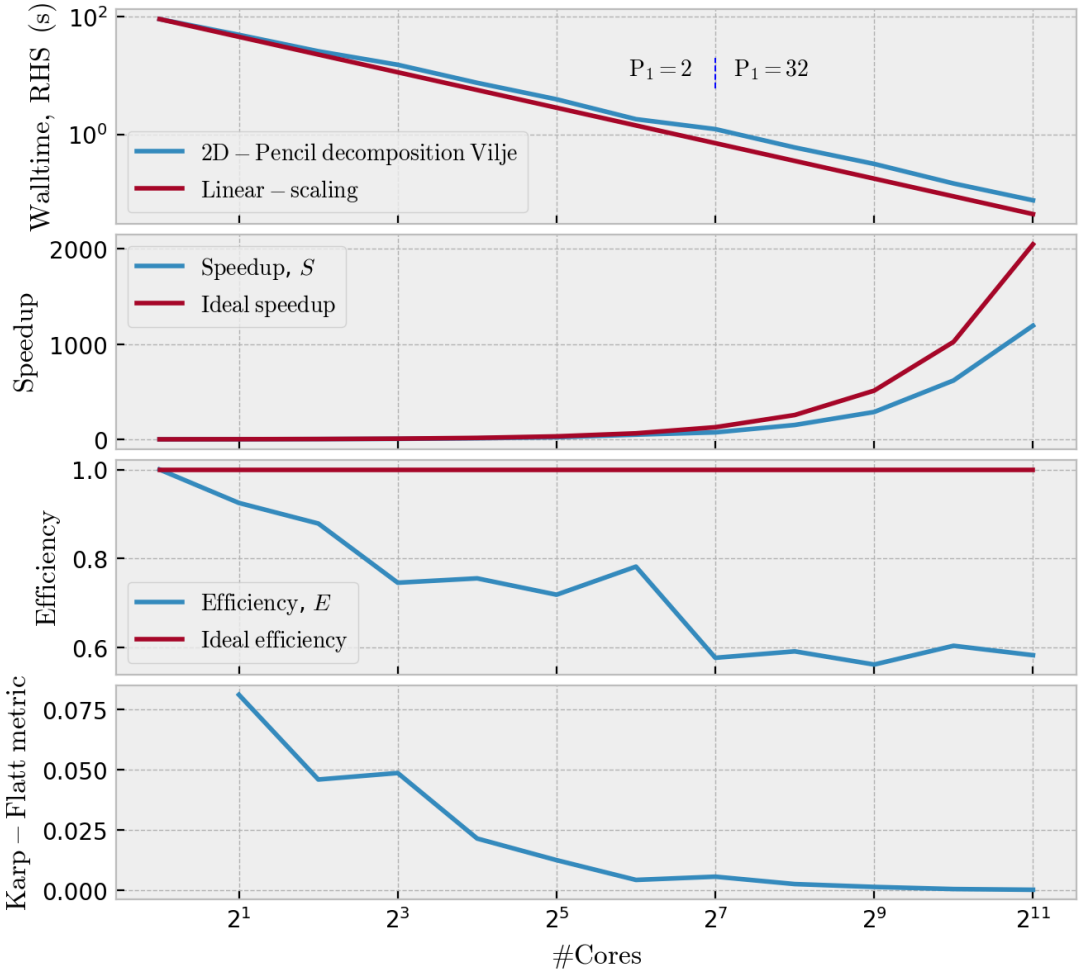


Figure 4.1: Four indicators for quantifying the parallelized portion of the DNS solver are presented. The first metric from the top presents the averaged measured walltime when computing the spatial terms of equation (3.7) on the Vilje computing cluster. Ideal linear scaling plotted for comparison. The second metric show the speedup relative to the serial execution due to parallelization. Ideal speedup plotted for comparison. The third metric shows the efficiency of parallelization, i.e., measured speedup relative to ideal speedup. Ideal efficiency, i.e., $S/p = 1$ is plotted for comparison. At the bottom the Karp-Flatt metric is presented, indicating the serial fraction of the code. $s_t = 0$ corresponds to a fully parallelized code.

computation of the curl of the velocity field, forward and backward FFTs and communication between computer cores. The performance metrics are measured on an amount of computer cores ranging from $\#\text{Cores} = p \in [1, 2048]$ and are presented in figure 4.1. The notation for number of cores, p, is the one introduced in section 3.4; however, the notation $\#\text{Cores}$ is used in the plots to make them more readable without having to read the theory and method sections of the thesis.

The top graph in figure 4.1 shows the average measured walltime time the code spends inside the function dedicated to compute the spatial terms of equation (3.7), which corresponds to computing the RHS of an ODE. The averaging is done over several hundred calls to the aforementioned function. A linear-scaling plot is added in the same graph to compare the measured walltime with the ideal walltime. The walltime follows a linear-scaling relatively close, up until $p = 2^7 = 128$, where the pencil decomposition is changed from $P_1 = 2$ to $P_1 = 32$, i.e., the physical mesh along the x axis is distributed amongst 32 cores instead of 2. At this point, the walltime is seen to decrease by a smaller factor compared to the prior measurements. One reason for this effect is that more computational effort is required to communicate along several axis, compared to the case where the data is mostly distributed along the y axis in physical space.

The second graph in figure 4.1 shows the speedup, which in section 3.4 was defined as the relative measurement of the serial case and the parallel case, i.e., $S = T_{serial}/T_{parallel}$. This graph also shows that the measured walltime experiences close to linear-scaling, up until the point $p = 2^7 = 128$. Here, the measured speedup diverges from the ideal speedup, and continues to diverge up until the maximum core count of $p = 2^{11} = 2048$. The efficiency, E, is plotted in the third graph alongside the ideal efficiency. This is essentially the speedup normalized with the ideal speedup. The efficiency is seen to decrease as the number of cores increases; however, it levels out around $p = 2^3 = 8$ cores at the value $E = 0.75$, until the core distribution is altered at $p = 2^7 = 128$. Here, the efficiency drops again to the value $E \approx 0.6$, but stays fairly constant. The sudden drop in efficiency and the diverging speedup are also due to the increased communication when distributing cores along several axis.

The fourth and last graph in figure 4.1 shows the Karp-Flatt metric which essentially measures the fraction of the code that is not parallelized, s_t. This metric is used to determined whether the speedup barrier is due to a large serial fraction of the code or due to parallel overhead from increased MPI communication. It also detects inefficiencies due to process startup time and imbalanced workload, which cannot be easily interpreted from the measured walltime, speedup or efficiency plots. The value of s_t starts at approximately 0.075 and drops towards zero as the number of computer cores increase. The low value of s_t verifies that the code is highly parallelizable. The serial fraction measured at $p = 2^7$ is slightly higher than the one measured at $p = 2^6$, which again confirms the increased parallel overhead at this point. Moreover, the constant value which s_t approaches for $p > 2^8 = 256$ indicates that the parallel overhead is not increasing after this point. If the code had been tested for $p > 2^{11} = 2048$, the workload per computer core would eventually decrease beyond a point where parallel overhead starts to dominate, resulting in an increasing s_t.

A slab decomposition of the DNS solver has been measured to scale better than the 2D pencil decomposition by, e.g., Mortensen and Langtangen (2016). This method is not presented in section 3.1 since the increased computational capacity which a 2D pencil decomposition allows was determined to be superior to the slab decomposition's better scaling. Mortensen and Langtangen (2016) also measured the performance of a DNS solver using the 2D pencil decomposition, and their result follows the linear scaling slightly closer than the measurements done in this thesis. It is worth noting that their solver computed the convective terms, i.e., the most computationally demanding terms, using a Cython implementation of the Python functions computing relevant cross-product of vectors. They measure a

20 % better performance when optimizing the code using Cython. This optimization method essentially produces compiled functions written with a Python-like syntax, see Behnel et al. (2011). Optimization through Cython was not done when running the DNS solver in this thesis on Vilje; however, the Python library Numba was tested on the IDUN computing cluster, and an performance increase comparable to the one measured by Mortensen and Langtangen (2016) using Cython was measured. The Numba library is presented by Lam et al. (2015).

The measured walltime presented in the top graph of figure 4.1 was investigated more closely for the point $p = 2^5 = 32$. The measurements are presented in figure 4.2 and consist of the non-averaged walltime spent inside the function that computes the spatial terms of equation (3.7). The figure shows that the walltime fall into one of two distributions; first peaking at $t \approx 3\,\mathrm{s}$ and another peaking at $t \approx 4.25\,\mathrm{s}$. Exactly $1/4$ of the measurements fall into the first distribution, leaving $3/4$ measurements in the second distribution. This coincides with the four calls to the function responsible for computing the spatial terms of equation (3.7), necessary to compute the next time step using the fourth order Runge-Kutta method presented in section 3.2.1. A closer look revealed that the quickest measurements which fell into the first distribution where also the first function call of each time step. The three remaining function calls per time step all fell into the second distribution. The higher walltime for the three last function calls are directly caused by backward FFTs which are necessary to evaluate the velocity field at the different time increments, as explained in section 3.2. This backward FFT is not necessary in the first function call.

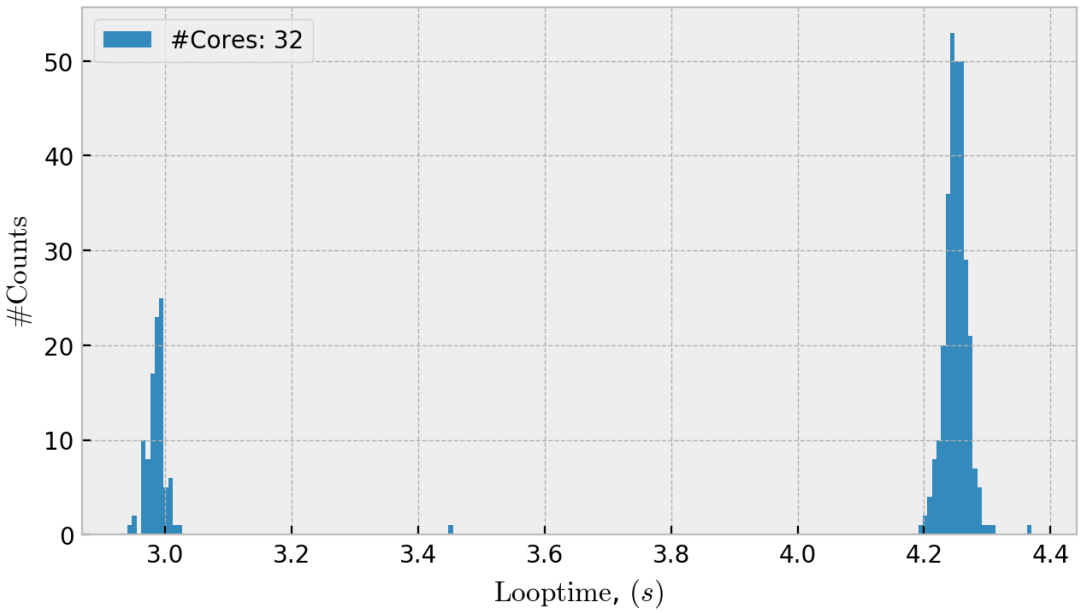


Figure 4.2: Measured walltime when computing the spatial terms of equation (3.7) using $p = 32$ computer cores. $1/4$ of the measurements are seen to have a shorter walltime than the remaining $3/4$.

4.2 DNS solver

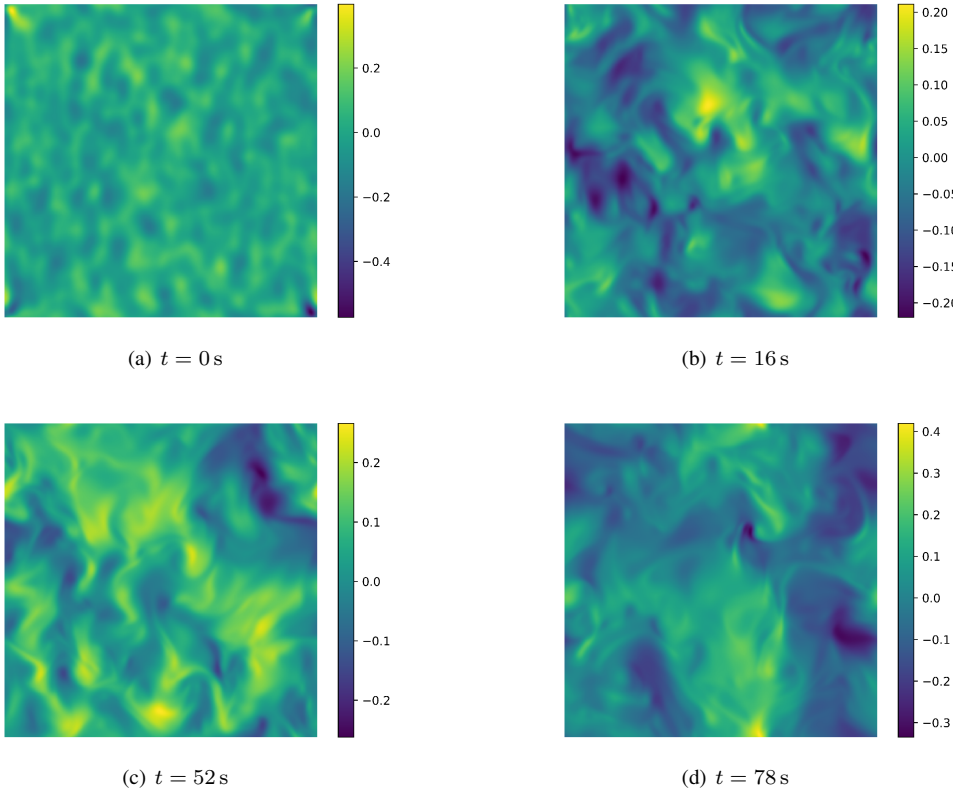
This section presents a velocity field generated by the DNS solver. The maximum simulation time was $t = 80\,\mathrm{s}$. Visual snapshots of the velocity field in physical space at four different time steps are presented. Additionally, the kinetic energy distribution evaluated in spectral space at the same time steps are presented, as well as four alternative ways of visualizing the velocity field.

4.2.1 Velocity field

Four two-dimensional snapshots are taken of a velocity field at the time steps $t = 0\,\mathrm{s}$, $t = 16\,\mathrm{s}$, $t = 52\,\mathrm{s}$ and $t = 78\,\mathrm{s}$, and are presented in figure 4.3. The snapshots are taken in the xy plane at $z = \pi$. The first plot of the four, i.e., figure 4.3(a), is the initial velocity distribution, computed from equation (2.65). As the flow develops over time, the different spatial scales start to be visible. All spatial scales are expected to be present as the non-linear terms break down the largest structures to smaller structures through vortex-stretching, and the forcing of the low wavenumbers, i.e., the largest spatial scales, maintains the energy in the largest eddies. The value of the velocity is presented by color, where a light green color indicates positive values, and dark blue indicates negative values. The highest absolute value of velocity is approximately $0.3\,\frac{\mathrm{m}}{\mathrm{s}}$. The range of different values are indicated by a colorbar to the right of each plot. It is instructive to repeat that the initial velocity field has a random contribution included, as presented in section 2.4.3, which makes each simulation of the velocity field unique.

Most techniques used to visualize 3D vector fields suffer from visual clutter, and the task of effectively conveying both structural and directional information of the vector fields remains a challenge. Some of these challenges are discussed by, e.g., Chen et al. (2011), which also presents a framework for visualizing 3D vector fields. The Python library *Mayavi* is another framework for interactive scientific data visualization and 3D plotting. See Ramachandran and Varoquaux (2011) for an introduction to the documentation, underlying technologies and capabilities. This library is used to produce four alternative methods of visualizing the velocity field at the time step $t = 16\,\mathrm{s}$, first presented in figure 4.3(b). The alternative visualizations are presented in figure 4.4.

Figure 4.4(a) presents a three-dimensional box plot which essentially captures the edges of the computational domain. The directional information in the velocity field is not easily interpreted from this plot; however, the spatial extent and the layout of the computational domain is directly available. Moreover, the box plot hides information on everything that is inside of the edges of the computational domain. Figure 4.4(b) presents three cross-sectional plots placed in the xy-, xz- and yz-planes of the velocity field u at the time $t = 16\,\mathrm{s}$. The planes are positioned in the middle of the computational domain. This plot still captures the general layout of the computational domain, in addition to containing some interior information of the velocity field. Figure 4.4(c) presents a 3D quiver plot where the direction and magnitude of the velocity field at certain locations are visible. The immense amount of information presented at the same time in such quiver plots makes it nearly impossible to make qualitative interpretations; however,the chaotic nature of turbulent flows is clearly portrayed through the apparent randomness in direction and magnitude. The last visualization is presented in figure 4.4(d) which presents a three-dimensional isosurface of the areas in the velocity field with a value of $u = 0.2\,\mathrm{m/s}$. This type of visualization really shows the inner structures of the flow and can be used to find both small and large spatial scales. Another common method is to visualize the vorticity field, i.e., the curl of the velocity field, using an isosurface presentation. However, this is not done here.



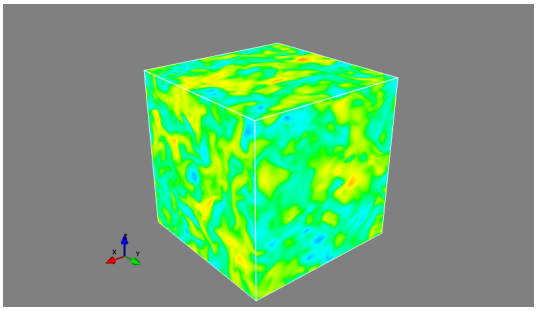
(a) $t = 0\,\mathrm{s}$ (b) $t = 16\,\mathrm{s}$

(c) $t = 52\,\mathrm{s}$ (d) $t = 78\,\mathrm{s}$

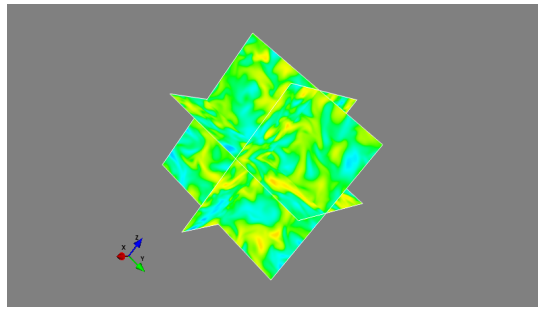
Figure 4.3: Two dimensional snapshot of a computed velocity field, u, at four different time levels. The snapshot is taken in the xy plane at $z = \pi$.

4.2.2 Energy spectrum of the velocity field

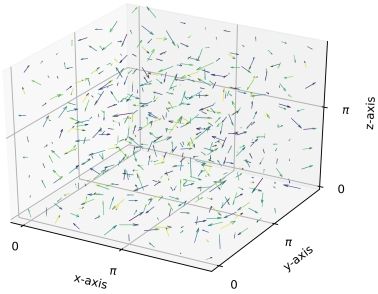
The kinetic energy spectrum of the velocity field is presented in figure 4.3 for the time steps $t = 0\,\mathrm{s}$, $t = 16\,\mathrm{s}$, $t = 52\,\mathrm{s}$ and $t = 78\,\mathrm{s}$. The spectrum is computed by the evaluation of equation (2.57) presented in section 2.4.2. *Kolmogorov's $k^{-5/3}$ law*, also introduced in section 2.4.2, is included for comparison. The top graph in figure 4.5 presents the initial kinetic energy spectrum used to compute the velocity field from equation (2.65). The initial spectrum is seen to have most of its energy located in the lower wavenumbers, i.e., the largest spatial scales. The highest energy is located at the wavenumber $k = 9.5$, coinciding with the variable a introduced in section 2.4.3. The remaining three graphs in figure 4.5 show that as the flow is developing over time, the energy is distributed over all wavenumbers, i.e., over all spatial scales. The cascading of energy from low wavenumbers to high wavenumbers is seen to follow Kolmogorov's theory closely up to the point $k = 2^5 = 32$, where the energy starts to drop. The region in the kinetic energy spectrum stops following the scaling of $k^{-5/3}$, i.e., the dissipation range, follows the scaling presented in equation (2.59) presented in section 2.4.2. Further investigation of the dissipative range is not conducted here; however, numerical results for turbulent flows with similar



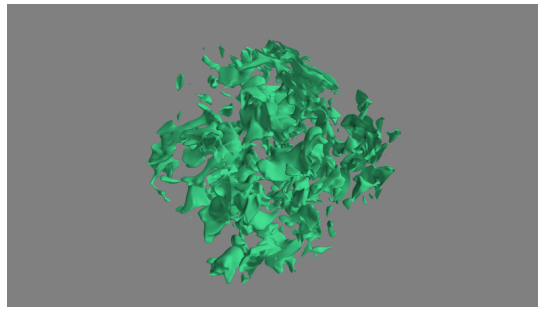
(a) Three-dimensional box-plot of the velocity field u at the time $t = 16\,\mathrm{s}$. Light-yellow colors indicate velocities with positive values. A dark blue color indicates velocities with negative values.



(b) Cross-sectional plots in the xy-, xz- and yz-plane of the velocity field u at the time $t = 16\,\mathrm{s}$. The value along the last axis parallel to the normal vector of each plane is π for all cross sections.



(c) Three-dimensional quiver plot of the velocity field u at the time $t = 16\,\mathrm{s}$.



(d) Three-dimensional isosurface plot for the velocity value $u = 0.2\,\mathrm{m/s}$

Figure 4.4: Four visualization methods for the velocity field u at the time $t = 16\,\mathrm{s}$. Each method presents a portion of the same data. However, the data perceived by the reader varies in each method. For instance, the box plot and cross-sectional plot both show the smooth transitions happening between eddies in the flow, whereas the quiver plot reveals the directional chaos present in a turbulent flow field.

parameters to the ones used in this thesis, are presented by, e.g., Kovasznay (1948); Heisenberg (1948); Smith and Reynolds (1992); Martínez and Kraichnan (1996).

Furthermore, it is clear that the forcing term presented in section 2.4.4 maintains the energy levels at the forced wavenumbers, i.e., $k \leq 8$. After the kinetic energy is distributed amongst all wavenumbers, which occurs at approximately $t = 16\,\mathrm{s}$, the energy is relatively constant. A longer simulation would reveal if the kinetic energy distribution actually remains this way, verifying that the forcing scheme works properly. The statistical stationarity in the timescale $t \in [16\,\mathrm{s}, 80\,\mathrm{s}]$ due to the implementation of this particular forcing scheme was concluded to be sufficient.

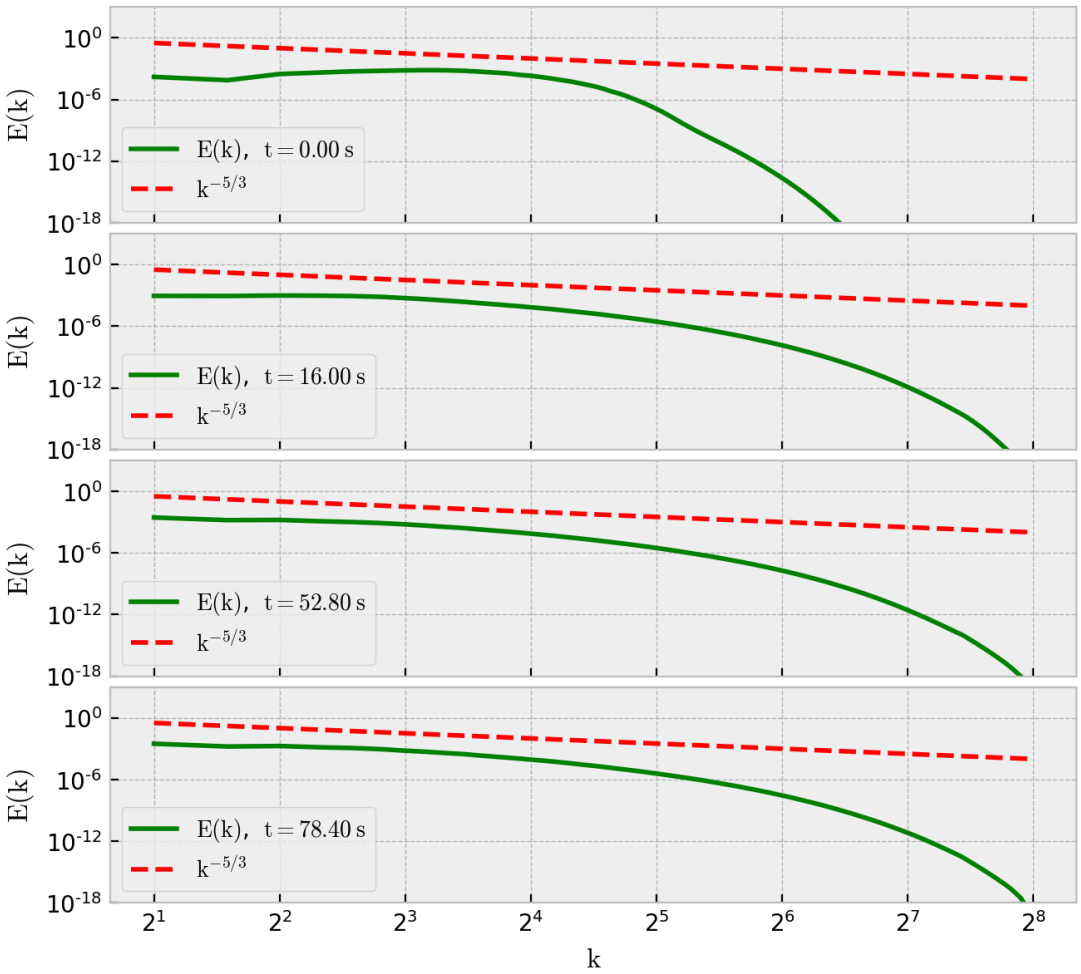


Figure 4.5: Computed kinetic energy spectrum of the velocity field, u, at the time levels $t = 0\,\mathrm{s}$, $t = 16\,\mathrm{s}$, $t = 52\,\mathrm{s}$ and $t = 78\,\mathrm{s}$. The two upper graphs have been presented previously in section 2.4, and is repeated here for the purpose of illustrating the temporal evolution of the energy in the flow.

4.3 Particle model and the advection-diffusion equation

This section presents the particle distributions obtained by numerically solving the particle model introduced in section 2.2.4. Some visualizations of the distributions are given, as well as a presentation of the variance of the particle positions, which was introduced in section 2.1.2 to be directly linked to the effective mixing parameter, K, also referred to as the turbulent diffusion.

Note that the particle model is solved numerically using two separate setups. The first setup is a simulation where the position of all $m = 80000$ particles start at the center of the computational domain, i.e., $(x, y, z) = (\pi, \pi, \pi)$. This simulation is run from $t = 0\,\mathrm{s}$ up to $t = 80\,\mathrm{s}$. The other setup is a

simulation where the position of the particles are initialized randomly from a normal distribution whose mean is $\mu = \pi$ and variance is $\sigma^2 = 0.01$ for all three dimensions. The simulation time runs from $t = 0\,\mathrm{s}$ up to $t \approx 16\,\mathrm{s}$ The second simulation setup is the one which is compared to the numerical solution of the advection-diffusion equation, presented in section 4.3.2. The combination of the DNS solver, the numerical solution of the advection-diffusion equation and the particle model are only simulated up to $t = 16\,\mathrm{s}$ due to the immense amount of computational resources required. To put things in perspective; the DNS solver was run alongside the particle model using a time step of $dt = 0.01\,\mathrm{s}$ and a maximum simulation time of $t = 80\,\mathrm{s}$. This took well over 10000 CPU hours on the IDUN computing cluster. As explained in section 3.2.2, to achieve a stable numerical solution to the advection-diffusion equation, a time step of $dt = 0.001\,27\,\mathrm{s}$ is necessary. In theory, this roughly leads to a 8 times increase of computational resources compared to a simulation using $dt = 0.01\,\mathrm{s}$.

4.3.1 First setup with particle model

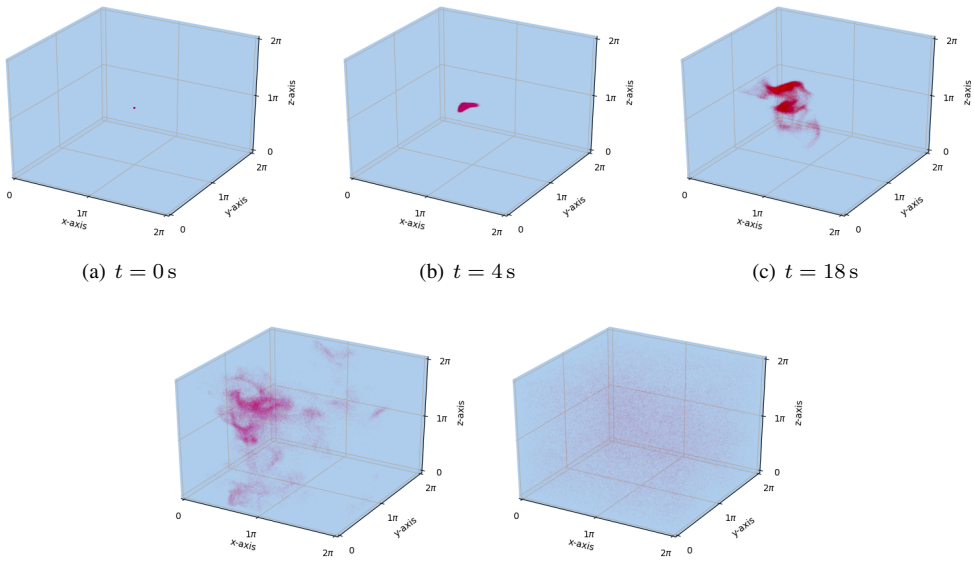
Particle trajectory

The particle distributions from the first particle model setup at the time steps $t = 0\,\mathrm{s}$, $t = 4\,\mathrm{s}$, $t = 18\,\mathrm{s}$, $t = 38\,\mathrm{s}$ and $t = 80\,\mathrm{s}$ are presented in figure 4.6. Figure 4.6(a) shows the initial particle distribution where all particles are located at the center of the computational domain, i.e., $(x, y, z) = (\pi, \pi, \pi)$. Figure 4.6(b) shows the particle distribution four seconds later. At this stage, the diffusion term in the particle model transport the particles randomly from their initial position. The distribution after $t = 18\,\mathrm{s}$, presented in figure 4.6(c), shows that transportation in the z-direction is dominating, and continues to dominate even after $t \geq 38\,\mathrm{s}$. Eventually, the positions of the particles follow a close to uniform distribution, as seen in figure 4.6(e). Note that this is only one simulation of particle transport in one velocity field, and the dominating transport along the z-axis is a statistical event that is not likely to occur more regularly than a dominating transport in the x- or y- direction if the number of simulations increase.

In figure 4.7 the trajectory of four selected particles are presented. Figure 4.7(a) shows the trajectories for $t \in [0\,\mathrm{s}, 80\,\mathrm{s}]$ with $dt = 0.01\,\mathrm{s}$, and figure 4.7(b) shows the trajectories for $t \in [0\,\mathrm{s}, 2\,\mathrm{s}]$ with $dt = 0.01\,\mathrm{s}$. It is clear that the four particles are initially transported mainly due to advection, as they transport along the negative x-axis. Eventually, the particles enter separate eddies and follow seemingly random trajectories. Recall that with periodic BCs, if a particle transports out of one of the planes bounding the computational domain, it will enter through the opposite plane.

Variance of distribution

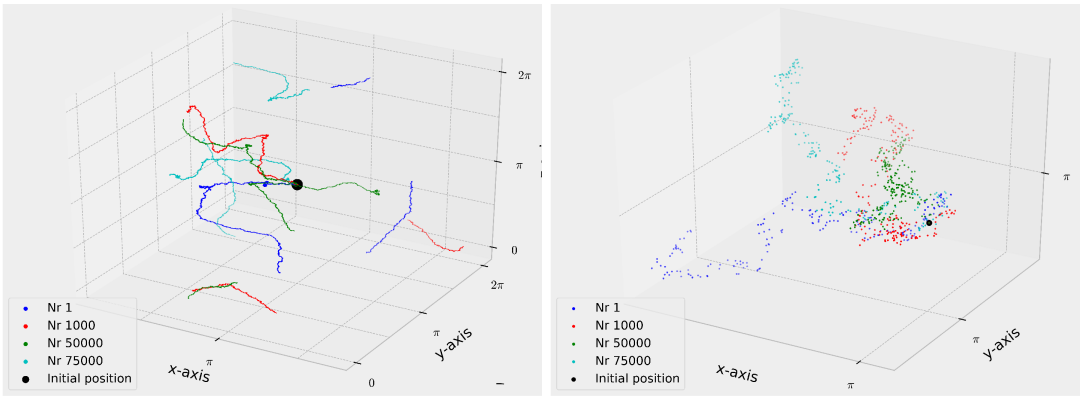
To analyse the turbulent diffusion causing the mixing of the particles in the turbulent flow field, the variance of the particle positions are being computed. One variance component is being computed for each particle coordinate, e.g., the variance in x-direction, σ_x^2, is computed from the particle positions along the x-axis. The three variance components are presented in figure 4.8. Additionally, the geometric mean of the variance is computed as a measure for the collected distribution along all axes. The geometric mean for a set of numbers $x_1 x_2 x_1 \dots x_n$ is defined as $\sqrt[n]{x_1 x_2 x_1 \dots x_n}$, which gives the geometric mean for the variance components, $\sigma^2_{\text{geometric}} = \sqrt[3]{\sigma_x^2 \sigma_y^2 \sigma_z^2}$. Moreover, a first order polynomial curve fit is computed in two regions for each variance component, as well as the geometric mean of the variances. The first curve fitted region is at the very start of the simulation, where the particles are clustered together



(a) $t = 0\,\mathrm{s}$ (b) $t = 4\,\mathrm{s}$ (c) $t = 18\,\mathrm{s}$

(d) $t = 38\,\mathrm{s}$ (e) $t = 80\,\mathrm{s}$

Figure 4.6: Three-dimensional scatter plot of the position to all $m = 80000$ particles at five different time steps. The initial distribution is located at $(x, y, z) = (\pi, \pi, \pi)$. At the time $t = 80\,\mathrm{s}$ the particle positions appear to be uniformly distributed.



(a) 8000 time steps. (b) 200 time steps.

Figure 4.7: The temporal evolution of the position to four individual particles are presented for 8000 time steps, i.e. $t = 80\,\mathrm{s}$ with $dt = 0.01$, in figure 4.7(a). Figure 4.7(b) presented the same four particles temporally truncated to 200 time steps, where the spatial axis are heavily zoomed in.

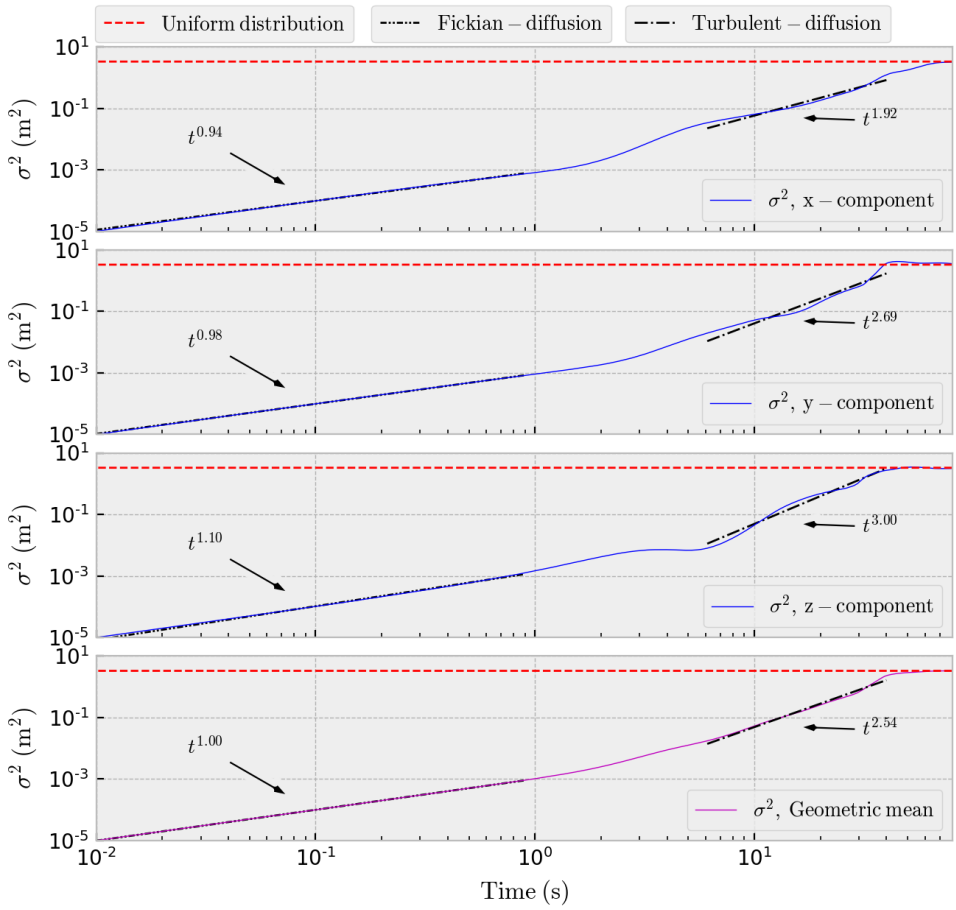


Figure 4.8: Presentation of the variance in the particle positions along the x-, y- and z-direction for the time interval $t \in [0\,\mathrm{s}, 80\,\mathrm{s}]$. The geometric mean of the variances is computed and presented as well. A first order polynomial curve fit is computed in two regions for each variance component. The first region is at the start of the simulation where the particles are clustered and thus heavily spatially correlated. The variance curve is seen to coincide with the Fickian theory of diffusion, i.e. $K \propto t^1$. The second region is after the flow has had some time to develop and the particles have started to follow individual trajectories, resulting in a turbulent mixing, i.e. $K \propto t^3$. The uniform distribution is also included to present the theoretical upper limit for mixing.

and heavily spatially correlated. The second region is around $t \geq 10\,\mathrm{s}$, where the flow has had some time to develop.

The top graph in figure 4.8 shows the variance in the x-position of the particles. Initially the distribution scales as a Fickian diffusion, i.e., the variance of the distribution scales as $\sigma^2 \propto t^1$. This is

expected as the particles are closer together and are transported mainly through advection by the same eddy in the flow. The spreading of the distribution is thus due to the random diffusion component in the particle model, which is expected to scale as a Fickian diffusion. After some time, the rate of change of the variance starts to increase, and at $t \geq 10\,\mathrm{s}$ the variance is seen to scale as $\sigma^2 \propto t^{1.92}$, up to the point of uniform distribution which happens at around $t \approx 50\,\mathrm{s}$. This scaling is not as high as predicted by Richardson (1926) for mixing in turbulent flow fields, which in section 2.1.2 was introduced to scale as $\sigma^2 \propto t^3$. However, the x-component of the variance still experiences a higher scaling at some time steps, e.g., for $t \in [2\,\mathrm{s}, 7\,\mathrm{s}]$. The second and third graphs show the variance scaling for the y- and z-components of the particle positions. Initially they follow a Fickian scaling, but soon transitions into turbulent- or close to turbulent scaling, i.e., $\sigma^2 \propto t^{2.69}$ for the y-component and $\sigma^2 \propto t^3$ for the z-component. The geometric mean of the three variance components are presented in the last graph in figure 4.8, and it shows that the initial scaling is Fickian, and eventually a close to turbulent-scaling of $\sigma^2 \propto t^{2.54}$.

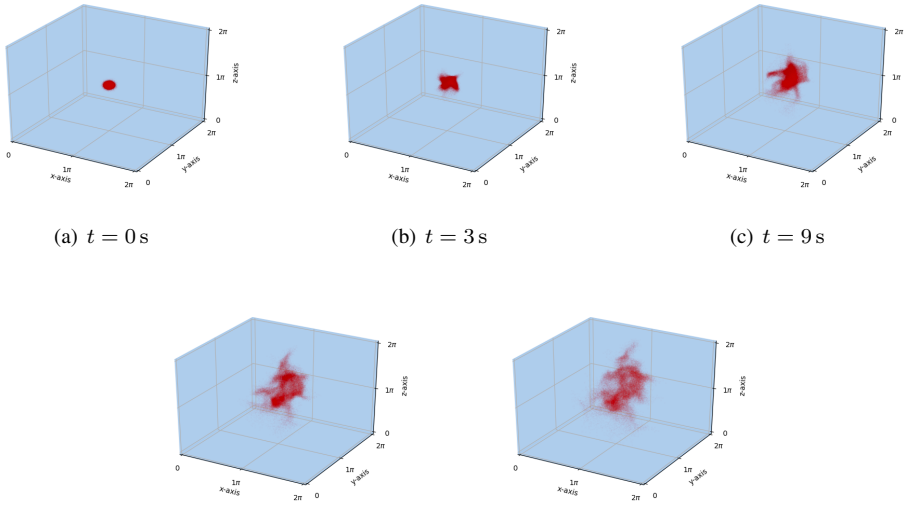
As stated above, this is data from only one simulation, and the statistical background for stating that turbulent-diffusion is present in either spatial direction for a general flow, is insufficient. However, the statistics available is enough to state that this particular flow has a higher effective mixing in the z-direction, compared to the two other spatial directions. Recall that in section 2.1.2 a proof was given for the scaling of the effective mixing coefficient, K, introduced by Richardson (1926). It was found that K scales as $\sigma^{4/3}$. The result from the geometric mean presented in figure 4.8 can be connected with the effective mixing coefficient through some simple algebraic manipulations of the theory presented in section 2.1.2, yielding $K \propto \sigma^{4/2.54} = \sigma^{1.574}$. A revision of the results from Richardson (1926) have been presented in Malik (2018), and they find that the effective mixing coefficient in fact does scale as $K \propto \sigma^{1.564}$, and not like $K \propto \sigma^{4/3} = \sigma^{1.333}$ as stated by Richardson (1926). Instead of Richardson's law which is said to be valid in the inertial-subrange, they propose two scaling laws which are dependent on the scale of energy cascade from the low wave numbers to high wave numbers, see Malik (2018) for details. Nonetheless, their analysis seems to align with the data presented in this thesis for the first particle setup.

As discussed in section 2.4.4, a statistically stationary flow may be seen after $t \approx 16\,\mathrm{s}$. The start of the simulation of the particle model is set to coincide with the start of the DNS solver, i.e., $t = 0\,\mathrm{s}$. At this stage the flow is not yet fully developed. It is likely that the variance measurements presented in figure 4.8 would look differently, possibly with higher values, if the simulation had started at a later time step.

4.3.2 Second setup with particle model and the advection-diffusion equation

The particle distributions from the second particle model setup at the time steps $t = 0\,\mathrm{s}$, $t = 3\,\mathrm{s}$, $t = 9\,\mathrm{s}$, $t = 13\,\mathrm{s}$ and $t = 16\,\mathrm{s}$ are presented in figure 4.9. Figure 4.9(a) shows the initial particle distribution where the position of each particle is randomly selected from a multivariate distribution with variance $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$ and mean $\mu = \pi\,\mathrm{m}$ in all three dimensions. The figures 4.9(b), 4.9(c), 4.9(d) and 4.9(e) show the particle distribution steadily spread around in the computational domain in the time steps $t = 3\,\mathrm{s}$ through $t = 16\,\mathrm{s}$. No obvious dominant direction is observed as opposed to the first setup where the z-direction was dominating.

Figure 4.10 presents the particle distribution from the second setup in two-dimensional projections. The columns present the distributions in the xy, xz and yz- plane respectively, and the rows presents the different time steps, $t = 0\,\mathrm{s}$, $t = 5\,\mathrm{s}$ and $t = 16\,\mathrm{s}$. As in figure 4.9, no obvious dominant direction of transport is observed; however, a small drift along the y-axis can be seen in, e.g., figure 4.10(i).



(a) $t = 0\,\mathrm{s}$ (b) $t = 3\,\mathrm{s}$ (c) $t = 9\,\mathrm{s}$

(d) $t = 13\,\mathrm{s}$ (e) $t = 16\,\mathrm{s}$

Figure 4.9: Three-dimensional scatter plot of the position to all $m = 80000$ particles at five different time steps.

The numerical solution to the advection-diffusion equation is presented in figure 4.11 at three different time levels for two-dimensional slices in the xy, xz and the yz-plane in the middle of the computational domain. The columns present the distributions in the xy, xz and yz- plane respectively, and the rows presents the different time steps, $t = 0\,\mathrm{s}$, $t = 5\,\mathrm{s}$ and $t = 16\,\mathrm{s}$. The projection of the three-dimensional data are plotted along the projected axes to better present the evolution of the distribution, S. Note that the colors in the plots are normalized such that the highest value in the available data is assigned the brightest color. Keeping the color scaling to a fixed value relative to the data in the first time step would eventually lead to plots too dark to be interpreted.

Compared to the particle distributions seen in figure 4.10, the dominant drift in the y-direction is more prominent in the numerical solution to the advection-diffusion equation. This can be seen from, e.g., figure 4.11(f) and 4.11(i). Moreover, by directly comparing the images from, e.g., $t = 5\,\mathrm{s}$, it is clear that the distribution has spread more in the numerical solution to the advection-diffusion equation, compared to the distribution from the simulation of the particle model. At $t = 5\,\mathrm{s}$ the particles in figure 4.10(d), 4.10(e) and 4.10(f) are clustered together close the the center of the computational domain, and the distribution still looks similar to the initial multivariate distribution at $t = 0\,\mathrm{s}$. The distribution from the advection-diffusion equation is already approaching the boundaries of the computational domain as the time approaches $t = 5\,\mathrm{s}$.

As done for the first setup with the particle model, the statistics of the distributions from the simulation of the particle model and the numerical solution to the advection-diffusion equation, S is computed and presented in figure 4.12. The figure contains four graphs where the first graph holds information of the measured variance of the particle distribution and S along the x-axis, the second figure holds information of the variance of the distributions along the y-axis, and the third graph is for the z-axis. The

fourth and last graph is a combination of the three spatial directions using a geometric mean of all the variance components. The data in each graph is fitted to a first order polynomial curve to investigate how the variance scales with time.

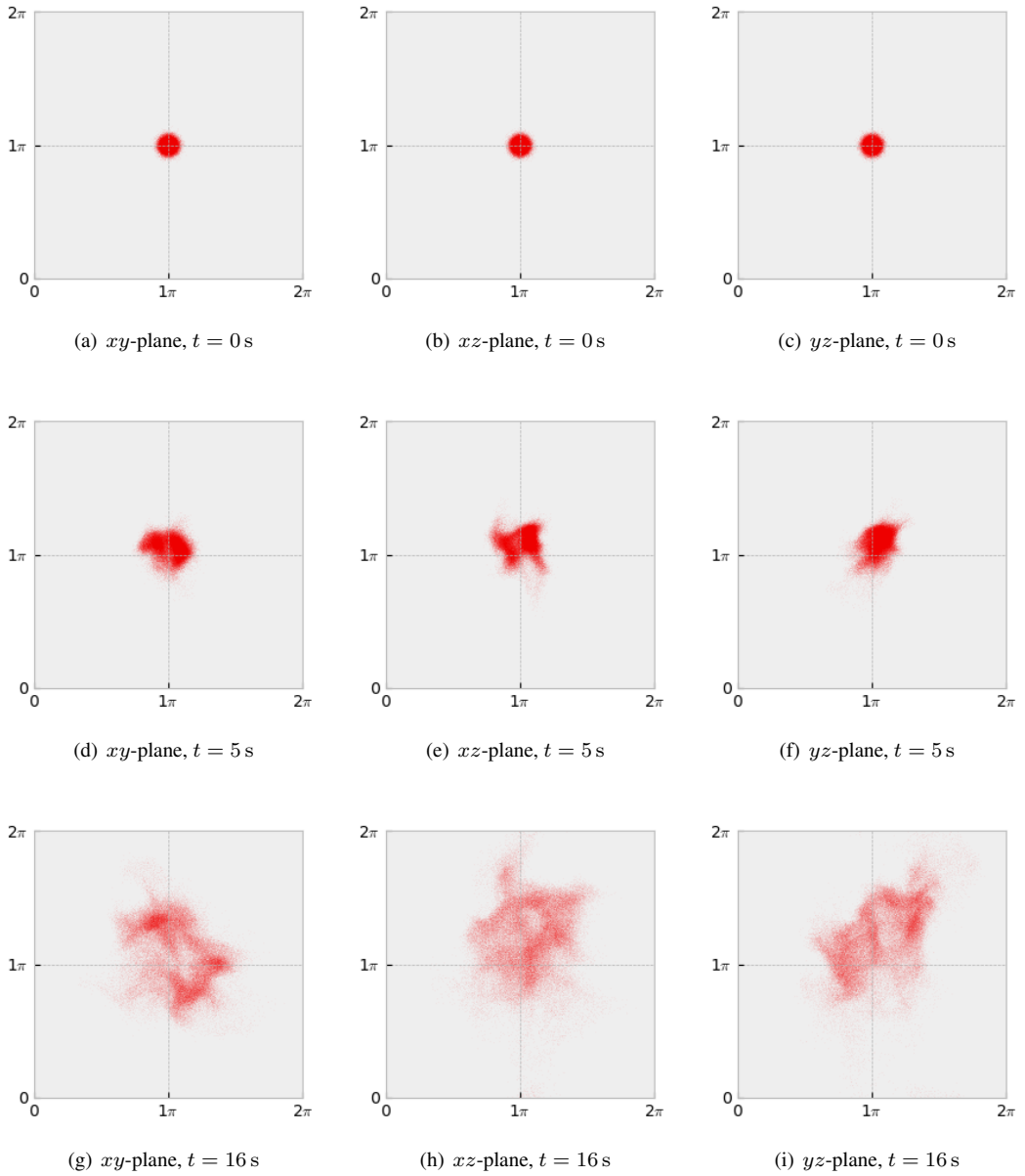
From figure 4.12 it is immediately clear that the computed variances differ remarkably from the variances computed from the particle model using the first setup. All components of the variance of the position of the particle distributions, including the geometric mean, are seen to be close to constant in the first second of the simulation, i.e. $\sigma^2 \propto 1$. After the first second the variance is seen to grow steadily. The variance of the particle distribution is seen to scale as $\sigma^2 \propto t^{1.44}$ for the x-component, $\sigma^2 \propto t^{2.18}$ for the y-component and $\sigma^2 \propto t^{2.11}$ for the z-component. The geometric mean scales as $\sigma^2 \propto t^{1.91}$. The y-direction experience the highest increase in variance, which fits the observations done in figure 4.10 and 4.11. Moreover, the computed variance of the distribution S in each direction, indicated by light blue colored squares in figure 4.12, are seen to be higher in value than the variance of the particle distribution; however, the scaling with time is linear rather than turbulent. Two important observations is made here; $1)$ The numerical solution of the advection-diffusion equation deviates to some extent from particle model, even though the theory in section 2.2.5 states that the two models are equal in the continuum limit, i.e., the number of particles, m is infinite, and the grid size in the spatial discretization of the advection-diffusion equation and the step size in time, Δt, is infinitesimal small. In practice this is not possible; however, it is not likely to be the cause of the mismatch between the transport models. $2)$ The scaling of the variance with time does not follow either the scaling proposed by Richardson (1926), nor does it follow the corrected scaling introduced and discussed by Malik (2018).

Firstly, the overall higher variance value in the distribution S is presumably due to the effects of an non-physical diffusion contribution, referred to as *numerical viscosity*. This extra contribution is an inevitable effect of discretized PDEs and essentially indicate that the difference of the discretized model and the PDE is nonzero. A numerical viscosity is usually related to error terms of a discretized PDE that are of first order, e.g., terms that are proportional to Δx, Δy or Δz. The errors in the spatial discretization of the advection-diffusion equation is presented in section 3.1 to be of second order, i.e., $\mathcal{O}(\Delta^2)$. Since the values in the distribution S are very small due to the normalization of the initialized distribution, as introduced in section 3.2.3, the second order error term may have some impact on the values in S. If the numerical viscosity changes the distribution S even slightly, the distribution may be transported around on different eddies of the flow, and the result will look different. An increased amount of particles and even finer mesh for the advection-diffusion equation would most likely reduce the deviation between the two transport models. Note that the initial variance and the final variance of both models are roughly the same, e.g., for the geometric mean the initial variance is $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$, as expected, and the final variance is about $\sigma^2 = 0.6\,\mathrm{m}^2$. This indicates that the mixing is essentially the same for this time interval, even though the details of the mixing are different. If the simulation were run for longer than $t = 16\,\mathrm{s}$, it is probable that the final variance of the distributions from the two models would deviate.

The fact that the overall scaling of variance with time follows neither Richardson (1926) nor Malik (2018) for the second simulation setup is assumed to be due to the statistical nature of the simulation. It could either be due to a poorly initialized velocity field where few eddies are located in the region where the initial particle distribution and S are located, or, as discussed in the results for the first particle setup, the velocity field isn't fully developed from $t = 0\,\mathrm{s}$, and needs some seconds to cascade the energy from the largest scales to the smallest scales. To initiate the particle distribution and S at times much later than $0\,\mathrm{s}$ has proven to be practically impossible. Due to the immensely complicated computational setup, a single simulation with the DNS solver, the simulation of the particle model and the numerical solution of the advection-diffusion equation, including MPI communication and storing of data, takes

about 12500 CPU hours for a simulation to run from $t = 0\,\mathrm{s}$ up to $t = 16\,\mathrm{s}$. By making a compromise between a large number of computer cores and a short queue time on the high-performance computer, a single simulation may in practice take about 8 days to complete. Recall that the second setup running from $t = 0\,\mathrm{s}$ to $t = 16\,\mathrm{s}$ uses a time step of $\Delta t = 1 \times 10^{-3}\,\mathrm{s}$, whereas the first simulation is stable for $\Delta t = 1 \times 10^{-2}\,\mathrm{s}$. Subsequently, it can be discussed whether more time should be spent on implementing a non-oscillatory spatial discretization for the advection-diffusion equation such that the time step could be one order higher, or if the process of implementation would take too much time. Nonetheless, the limit on computational resources has made it a challenge to simulate more than one long simulation with the DNS solver and the particle model, and one short simulation with the particle model and the advection-diffusion equation. Ideally, an ensemble of a large amount of simulations should be run to essentially capture the turbulent nature of several different initial conditions, in addition to study the turbulent diffusion first after the flow is fully developed.

Even though the scaling of variance is far from close to that of a turbulent diffusion in the second setup, some parallels can be drawn to the first setup. The main statistical difference between the first and second simulation setup is the initial variance of the distributions. In the first setup the variance is zero, whereas the variance is $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$ in the second setup. By looking at the geometric mean of the variances in the first simulation setup in figure 4.8, it takes roughly $9\,\mathrm{s}$ to increase the variance one order, e.g., from $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$ to $\sigma^2 = 1 \times 10^{-1}\,\mathrm{m}^2$. The second simulation needs approximately $8\,\mathrm{s}$ to achieve the same increase in variance. This indicates that even if the scaling of variance with time is linear in the second setup and turbulent in the first setup, the mixing is equal in the region $\sigma^2 \in [1 \times 10^{-2}\,\mathrm{m}^2, 1 \times 10^{-1}\,\mathrm{m}^2]$.

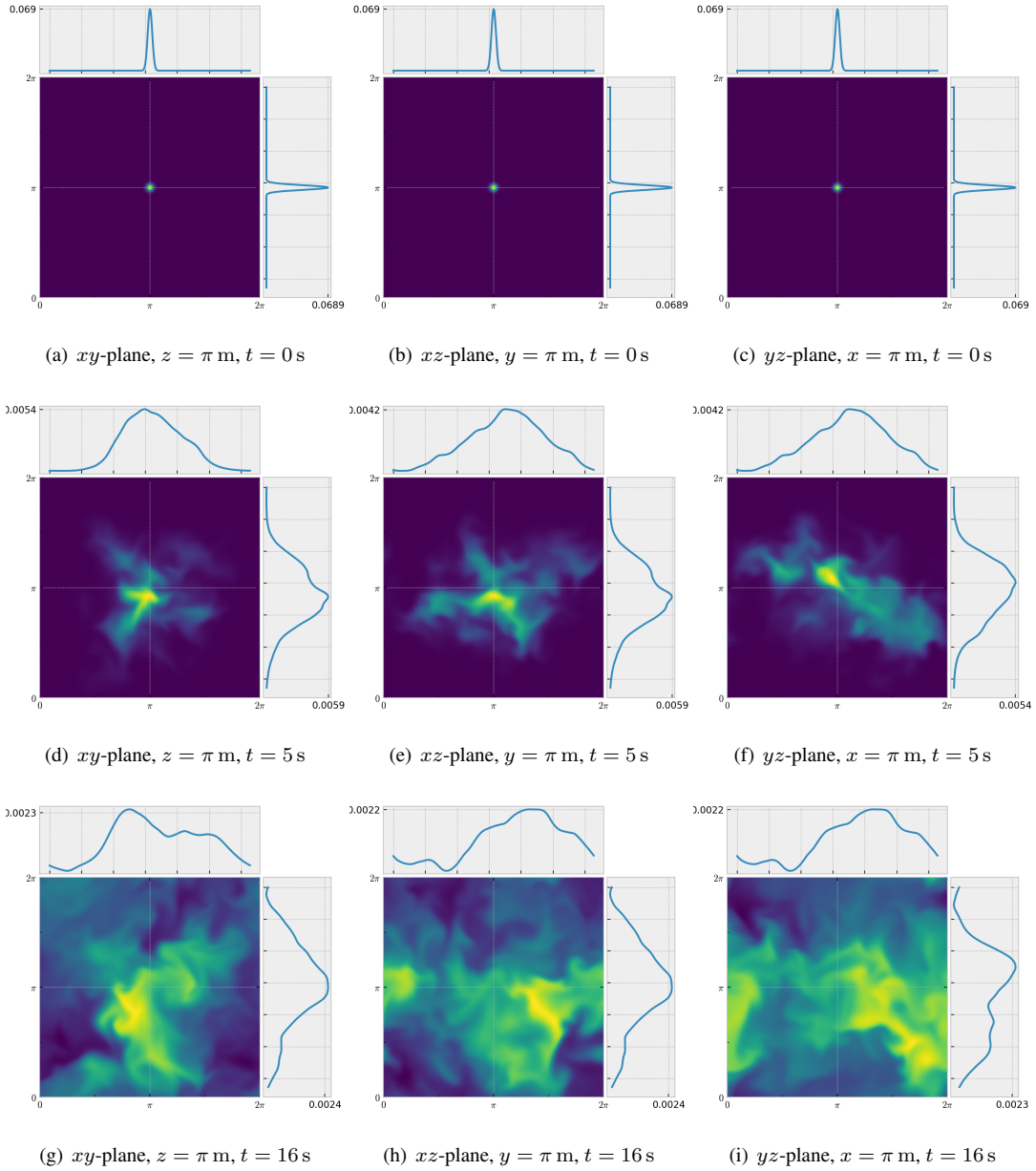


(a) xy-plane, $t = 0\,\mathrm{s}$ (b) xz-plane, $t = 0\,\mathrm{s}$ (c) yz-plane, $t = 0\,\mathrm{s}$

(d) xy-plane, $t = 5\,\mathrm{s}$ (e) xz-plane, $t = 5\,\mathrm{s}$ (f) yz-plane, $t = 5\,\mathrm{s}$

(g) xy-plane, $t = 16\,\mathrm{s}$ (h) xz-plane, $t = 16\,\mathrm{s}$ (i) yz-plane, $t = 16\,\mathrm{s}$

Figure 4.10: Scatter plot of the particle distribution using only two of the spatial components. The first column presents the distribution in the xy-plane, the second column presents the distribution in the xz-plane and the last column presents the distributions in the yz-plane. The different time steps are plotted in each row, where row 1, 2, and 3 shows the time steps $t = 0\,\mathrm{s}$, $t = 5\,\mathrm{s}$ and $t = 16\,\mathrm{s}$ respectively. In this particular figure the ab-plane yields the a-axis along the second axis, and the b-axis along the first axis. For example, in figure 4.10(i), the z-axis is horizontal and the y-axis is vertical.



(a) xy-plane, $z = \pi$ m, $t = 0$ s (b) xz-plane, $y = \pi$ m, $t = 0$ s (c) yz-plane, $x = \pi$ m, $t = 0$ s

(d) xy-plane, $z = \pi$ m, $t = 5$ s (e) xz-plane, $y = \pi$ m, $t = 5$ s (f) yz-plane, $x = \pi$ m, $t = 5$ s

(g) xy-plane, $z = \pi$ m, $t = 16$ s (h) xz-plane, $y = \pi$ m, $t = 16$ s (i) yz-plane, $x = \pi$ m, $t = 16$ s

Figure 4.11: A collection of two-dimensional snapshots of the numerical solution to the advection-diffusion equation. The distributions in column 1, 2 and 3 are in the xy, xz and yz-plane respectively. Row 1, 2 and 3 are at the time levels $t = 0$ s, $t = 5$ s and $t = 16$ s respectively. The projection of the three-dimensional data are plotted along the projected axes. In this particular figure the ab-plane yields the a-axis along the first axis, and the b-axis along the second axis. For example, in figure 4.10(i), the z-axis is vertical and the y-axis is horizontal, opposite from the system in figure 4.10.

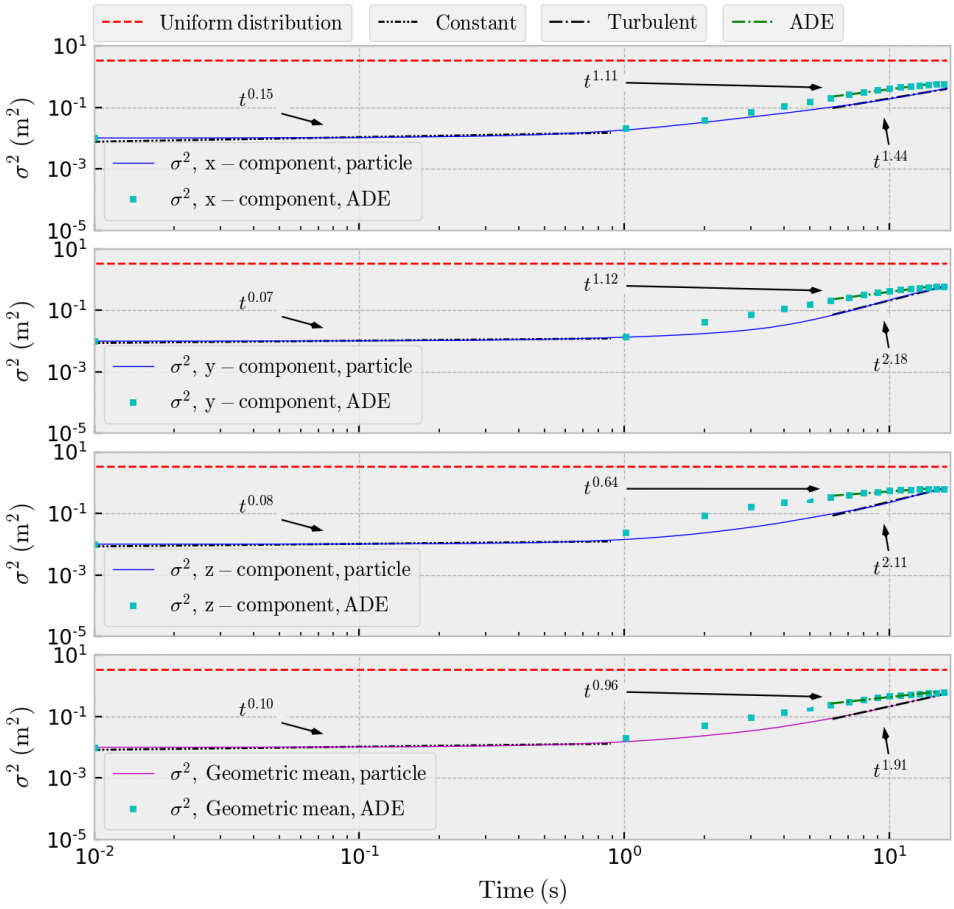


Figure 4.12: Presentation of the variance in the particle positions along the x-, y- and z-direction, as well as the computed variance for the numerical solution of the advection-diffusion equation, with the abbreviation ADE, are presented for the time interval $t \in [0\,\mathrm{s}, 16\,\mathrm{s}]$. The geometric mean of the variances is computed and presented as well. A first order polynomial curve fit is computed in two regions for each variance component. The uniform distribution is also included to present the theoretical upper limit for mixing.

Chapter 5

Conclusion

The work in this thesis has involved the implementation of a 3D solver for direct numerical simulation of the Navier-Stokes equation using a spectral method, fully parallelized using a pencil decomposition. The performance of the direct numerical simulation was investigated and found to scale well up to a $p = 2048$ cores with an efficiency of approximately $60\,\%$. The diffusive nature of a turbulent flow field with $Re = 1600$, generated from the direct numerical simulation, was studied using two different methods; a stochastic particle method, and the advection-diffusion equation. The particle method was parallelized by directly scattering relevant work amongst available computer cores, and the advection-diffusion equation was parallelized using a slab decomposition. All numerical implementations were run on the computing clusters Idun and Vilje, both located at NTNU in Trondheim.

Due to the computationally extensive problem, only two simulations were run. The first run included the direct numerical simulation and the particle model. The simulation time was run from $t = 0\,\mathrm{s}$ up to $t = 80\,\mathrm{s}$. The variance in the position of the particles indicated a Fickian diffusion in the initial stages of the simulation followed by a turbulent diffusion. The geometric mean of the variance scaled initially as $\sigma^2 \propto t^1$, and increased to $\sigma^2 \propto t^{2.54}$. This corresponds to an effective mixing parameter, K, which scales as $K \propto \sigma^{1.57}$, which is close to the theoretical scaling presented by Richardson (1926) and Malik (2018).

The second run included the direct numerical simulation, the particle model and the advection-diffusion equation, and was run from $t = 0\,\mathrm{s}$ up to $t = 16\,\mathrm{s}$. The variance of the particle position and the distribution from the numerical solution of the advection-diffusion equation both indicate a velocity field that is less turbulent than the field from the first simulation. This is likely due to a combination of a poorly initialized velocity field, an underdeveloped flow and random variations. The minor deviation between the particle model and the advection-diffusion equation is probably due to numerical viscosities that are present in discretized partial differential equations.

5.1 Further work

To improve the work done in this thesis the following could be done:

- Implement a non-oscillatory scheme for the advection-diffusion equation to increase temporal step

size, and thus reduce the required computational resources.

- Increase amount of simulations run to achieve statistical confidence in the data.
- Implement MPI communication of the velocity field in the particle model to eliminate the need for storage of the entire velocity field in the local memory of every computer core.

The author has experienced both the benefits and limitations of working with a three-dimensional solver for direct numerical simulation of turbulence. It is a powerful tool used in fundamental research in turbulence, and it allows for extraction of information at points in the flow that are inaccessible in an experimental setup. The amount of computational resources required to resolve all the relevant length and time scales are enormous, and turbulent flows encountered in most industrial applications requires a computational capacity that exceeds the limits of the most powerful computers of modern society. A large ensemble of simulations should be run to improve the statistical foundation of the data; however, this has proven to be practically impossible due to the limitations on both hardware and time.

Bibliography

ANSYS, Incorporated, 2013. ANSYS Fluent Theory Guide.

Aznag, K., Datsi, T., El Oirrak, A., El Bachari, E., 2020. Towards an improvement of fourier transform. International Journal of Advanced Computer Sciene and Applications 11. doi:10.14569/IJACSA.2020.0110188.

Batchelor, G.K., 1969. Computation of the energy spectrum in homogeneous two-dimensional turbulence. Physics of Fluids 12, 233–239.

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D.S., Smith, K., 2011. Cython: The best of both worlds. Computing in Science & Engineering 13, 31–39.

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

Brachet, M.E., 1991. Direct simulation of three-dimensional turbulence in the taylor-green vortex. Fluid Dynamics Research 8, 1–8. doi:10.1016/0169-5983(91)90026-F.

Brachet, M.E., Meiron, D.I., Orszag, S.A., Nickel, B.G., Morf, R.H., Frisch, U., 1983. Small-scale structure of the taylor-green vortex. Journal of Fluid Mechanics 130, 411–452. doi:10.1017/S0022112083001159.

Brachet, Marc E. and Meiron, Daniel I. and Orszag, Steven A. and Nickel, B. G. and Morf, Rudolf H. and Frisch, Uriel, journal=Journal of Statistical Physics, v.n.y., . The taylor-green vortex and fully developed turbulence .

Burattini, P., Leonardi, S., Orlandi, P., Antonia, R.A., 2008. Comparison between experiments and direct numerical simulations in a channel flow with roughness on one wall. Journal of Fluid Mechanics 600, 403–426. doi:10.1017/S0022112008000657.

Buschmann, P.E., Mensah, G.A., Moeck, J.P., 2020. Intrinsic thermoacoustic modes in an annular combustion chamber. Combustion and Flame 214, 251–262. doi:10.1016/j.combustflame.2019.11.006.

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

Chen, C.K., Yan, S., Yu, H., Max, N., Ma, K.L., 2011. An illustrative visualization framework for 3d vector fields. Computer Graphics Forum 30, 1941–1951.

Computational Fluid Dynamics Committee, AIAA, 2002. Guide: Guide for the Verification and Validation of Computational Fluid Dynamics Simulations. The American Institute of Aeronautics and Astronautics. doi:10.2514/4.472855.001.

Csanady, G.T., 1973. Turbulent Diffusion in the Environment. D. Reidel Publishing Company.

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., Mortensen, M., Keyes, D.E., 2019. Fast parallel multidimensional fft using advanced mpi. Journal of Parallel and Distributed Computing 128. doi:10.1016/j.jpdc.2019.02.006.

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.

Eswaran, V., Pope, S.B., 1988. An examination of forcing in direct numerical simulations of turbulence. Computers & Fluids 16. doi:10.1016/0045-7930(88)90013-8.

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

Ghandi, K.S., 2004. Turbulence and dispersion. Resonance 9, 48–61. doi:10.1007/BF02834869.

Gnedenko, B.V., Kolmogorov, A.N., 1954. Limit Distributions for Sums of Independent Random Variables. Addison-Wesley.

Gresho, P.M., Sani, R.L., 2000. Incompressible Flow and the Finite Element Method, Volume 1: Advection-Diffusion and Isothermal Laminar Flow. Wiley.

Gunzburger, M., 1989. Finite Element Methods for Viscous Incompressible Flows. Academic Press.

Heisenberg, W., 1948. The statistical theory on turbulence. Zeitschrift für Physik A Hadrons and nuclei 124, 628–657. doi:10.1007/BF01668899.

Higham, D.J., 2001. An algorithmic introcution to numerical simulation of stochastic differential equations. SIAM REVIEW 43, 525–546.

Hundsdorfer, W., Verwer, J.G., 2003. Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations. Springer-Verlag Berlin Heidelberg.

Ketcheson, D.I., Mortensen, M., Parsani, M., Schilling, N., 2020. More efficient time integration for fourier pseudospectral dns of incompressible turbulence. International Journal for Numerical Methods in Fluids 92, 79–93. doi:10.1002/fld.4773.

Kloeden, P.E., Platen, E., 1992. Numerical Solution of Stochastic Differential Equations. Springer-Verlag Berlin Heidelberg.

Koen, H., Cagnone, J.S., 2017. Dns of the taylor-green vortext at $re = 1600$, in: 5th International Workshop on High-Order CFD Methods.

Kovasznay, L.S.G., 1948. Spectrum of locally isotropic turbulence. Journal of Aeronautical Sciences 15. doi:10.1016/j.jmarsys.2006.07.007.

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.

Lam, S.K., Pitrou, A., Seibert, S., 2015. Numba: A llvm-based python jit compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/2833157.2833162, doi:10.1145/2833157.2833162.

Lamorgese, A.G., Caughey, D.A., Pope, S.B., 2004. Direct numerical simulation of homogeneous turbulence with hyperviscosity. Physics of Fluids 17. doi:10.1063/1.1833415.

Lee, M., Malaya, N., Moser, R.D., 2014. Petascale direct numerical simulation of turbulent channel flow on up to $786k$ cores, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, IEEE. doi:10.1145/2503210.2503298.

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

Li, N., Laizet, S., 2010. Decomp&fft - a highly scalable 2d decomposition library and fft interface. Cray User Group 2010 conference .

Liu, C., Cai, X., 2017. New theory on turbulence generation and structure - dns and experiment. Science China, Physics, Mechanics & Astronomy 60. doi:10.1007/s11433-017-9047-2.

Malik, N.A., 2018. Turbulent particle pair diffusion: A theory based on local and non-local diffusional processes. PLoS One 13.

Martínez, D.O., Kraichnan, R.H., 1996. Energy Spectrum in the Dissipation Range of Fluid Turbulence. Technical Report.

Mastroianni, G., Milovanovic, G., 2000. Interpolation Processes, Basic Theory and Applications. Springer-Verlag Berlin Heidelberg.

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 doi:10.1080/10618562.2017.1287902.

Moin, P., Mahesh, K., 1998. Direct numerical simulation: A tool in turbulence research. Annu. Rev. Fluid Mech. 30, 539–578. doi:10.1146/annurev.fluid.30.1.539.

Montgomery, D.C., Runger, G.C., 2013. Applied Statistics and Probability for Engineers. Wiley.

Mortensen, M., Langtangen, H.P., 2016. High performance python for direct numerical simulation of turbulent flows. Computer Physics Communications 203, 53–65. doi:10.1016/j.cpc.2016.02.005.

Nordam, T., Nepstad, R., Litzler, E., Johannes, R., 2019. On the use of random walk schemes in oil spill modelling. Marine Pollution Bulletin 146, 631–638. doi:10.1016/j.marpolbul.2019.07.002.

NOTUR, 2011. VILJE - The new supercomputer at NTNU. https://www.sigma2.no/sites/default/files/meta_magazine_4-11.pdf.

Nussbaumer, H.J., 1982. Fast Fourier Transform and Convolution Algorithms. Springer-Verlag Berlin Heidelberg.

Oliphant, T.E., 2006. A guide to NumPy. volume 1. Trelgol Publishing USA.

Orchini, A., Mensah, G.A., Moeck, J.P., 2019. Effects of nonlinear modal interactions on the thermoacoustic stability of annular combustors. The Journal of Engineering for Gas Turbines and Power 141. doi:10.1115/1.4040768.

Orszag, S.A., 1971. On the elimination of aliasing in finite-difference schemes by filtering high-wavenumber components. Journal of the Atmospheric Sciences doi:10.1175/1520-0469(1971)028<1074:OTEOAI>2.0.CO;2.

Overholt, M.R., Pope, S.B., 1988. A deterministic forcing scheme for direct numerical simulations of turbulence. Computers & Fluids 27, 11–28. doi:10.1016/S0045-7930(97)00019-4.

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

Pavliotis, G.A., 2014. Stochastic Processes and Applications - Diffusion Processes, the Fokker-Planck and Langevin Equations. Springer, New York, NY.

Pekurovsky, D., 2012. P3dfft: a framework for parallel computations of fourier transforms in three dimensions. SIAM Journal on Scientific Computing 34, 192–209. doi:10.5281/zenodo.2634590.

Pippig, M., 2013. Pfft: An extension of fftw to massively parallel architectures. SIAM Journal on Scientific Computing 35, 213–236. doi:10.1137/120885887.

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.

Pope, S.B., 2000. Turbulent Flows. Cambridge University Press.

Ramachandran, P., Varoquaux, G., 2011. Mayavi: 3D Visualization of Scientific Data. Computing in Science & Engineering 13, 40–51.

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.

Rodean, H., 1996. Stochastic Lagrangian Models of Turbulent Diffusion. Springer-Verlag Berlin Heidelberg.

Rogallo, R.S., 1981. Numerical Experiments in Homogeneous Turbulence. Technical Report.

Själander, M., Jahre, M., Tufte, G., Reissmann, N., 2019. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure. arXiv:1912.05848.

Smith, L.M., Reynolds, W.C., 1992. The dissipation-range spectrum and the velocity-derivative skewness in turbulent flows. Physics of FLuids A: Fluid Dynamics 3. doi:10.1063/1.857979.

Sullivan, N.P., Mahalingam, S., Kerr, R.M., 1994. Deterministic forcing of homogeneous, isotropic turbulence. Physics of Fluids 6. doi:10.1063/1.868274.

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

Versteeg, H., Malalasekera, W., 2007. Introduction to Computational Fluid dynamics - The Finite VOlume Method. Prentice Hall.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E.W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Contributors, S..., 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 17, 261–272. doi:https://doi.org/10.1038/s41592-019-0686-2.

Visser, A.W., 2008. Lagrangian modelling of plankton motion: From deceptively simple random walks to fokker-planck and back again. Journal of Marine Systems 70, 287–299. doi:10.1016/j.jmarsys.2006.07.007.

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

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

Willis, D.J., Niezrecki, C., Kuchma, D., Hines, E., Arwade, S.R., Barthelmie, R.J., DiPaola, M., Drane, P.J., Hansen, C.J., Inalpolat, M., Mack, J.H., Myers, A.T., Rotea, M., 2018. Wind energy research: State-of-the-art and future research directions. Renewable Energy 125, 133–154. doi:0.1016/j.renene.2018.02.049.

Yeoh, G.H., Tu, J., 2009. Computational Techniques for Multiphase Flows. Elsevier Science & Technology.

Zang, T.A., 1988. On the rotation and skew-symmetric forms for incompressible flow simulations. Applied Numerical Mathematics 7, 27–40. doi:10.1016/0168-9274(91)90102-6.

Øksendal, B., 2003. Stochastic Differential Equations. Springer-Verlag Berlin Heidelberg.

Appendices

Appendix A

Procedural Implementation

This section presents a step-by-step procedure on how the velocity field from the Navier-Stokes equation, the distribution S from the advection diffusion equation, and the particle distribution, are achieved. It will serve as a summary for the numerical part of chapter 3 and as a small overview of the Python code used to implement the numerical solvers. The numerical parameters are chosen to have metric units; however, they could be treated as dimensionless without implications for numerical implementation.

DNS solver routine

1. Declare system constants
   - Mesh size, $N = 2^9 = 512$
   - Length in each dimension, $L = 2\pi\,\mathrm{m}$
   - Viscosity, $\nu = 1/1600\,\mathrm{m}^2/\mathrm{s}$
   - Reynolds number, $Re = 1600$
   - Time step, $dt = 1 \times 10^{-3}\,\mathrm{s}$
   - Simulation time, $T_{\text{max}} \in [20\,\mathrm{s}, 80\,\mathrm{s}]$
   - Highest forced wavenumber, $k_f = 8$
2. Pre-allocate empty arrays used in the different routines for optimized memory management and speed-up of code.
3. Distribute the physical mesh, $\mathbf{x}$ and spectral mesh, $\boldsymbol{k}$ amongst the available processors according to the 2D pencil decomposition presented in section 3.3.3. Note that the MPI communication takes place during the forward and backward Fourier transformations.
4. Set initial energy spectrum as presented in section 2.4.3 and compute the initial velocity field from the energy spectrum.
5. Do the first forward Fourier transform of the velocity field.

6. Start computing the first time step using the fourth order Runge-Kutta method. The model equation is given by (3.7).
   (a) The convective term is computed by transforming the velocity field back to physical space, computing the curl of the velocity field, computing the cross product of the velocity and the curl, all in physical space. The resulting field from the cross product is forward transformed to spectral space.
   (b) The solution is dealiased after the convective term is added, following the procedure presented in section 2.3.4.
   (c) The pressure term is computed in spectral space after the convective term is available.
   (d) The viscous term is computed in spectral space.
   (e) Add a forcing contribution to certain wavenumbers following the procedure presented in section 2.4.4.
   (f) Once the velocity field is computed at the new time level, perform the backward Fourier transform to physical space and gather the array to one single core.
   (g) Send the full velocity field into the particle routine.
   (h) Send the full velocity field into the advection-diffusion routine.
   (i) At certain time steps do other post-processing routines such as computation of dissipation, computation of energy spectrum, storage of velocity fields to disk etc.
   (j) Iterate until T_{max}.

Advection-diffusion equation routine

1. Declare system constants
   - Mesh size, $N = 2^9 = 512$
   - Length in each dimension, $L = 2\pi\,\text{m}$
   - Diffusion parameter, $D = 5 \times 10^{-4}\,\text{m}^2/\text{s}$
   - Time step, $dt = 1 \times 10^{-3}\,\text{s}$
   - Simulation time, $T_{\text{max}} \in [20\,\text{s}, 80\,\text{s}]$
2. Pre-allocate empty arrays used in the different routines for optimized memory management and speed-up of code.
3. Set the initial conditions for the distribution S to be a three-dimensional multivariate distribution with variance $\sigma^2 = 0.01$ and mean $\mu = \pi$, in each spatial direction.
4. Distribute the field, S, amongst the available processors according to the slab decomposition presented in section 3.3.4. The distributed fields are called local domains. The local domains store boundary information from the neighbouring local domains in ghost points.
5. Start computing the first time step using the explicit Euler method.
   (a) Receive the velocity field from the DNS solver on one of the computer cores. Distribute the velocity field amongst the available processors according to the slab decomposition.

(b) Communicate the ghost points between the different local domains following the procedure presented in section 3.3.4. Ghost point communication is done both for the field, S, and for the distributed velocity field.

(c) Compute the inner points of the local domain using vectorized operations for optimized performance.

(d) compute the boundary points of the local domain using vectorized operations.

(e) At certain time steps do post-processing such as storing the field to disk.

(f) Iterate until T_{max}.

Particle method routine

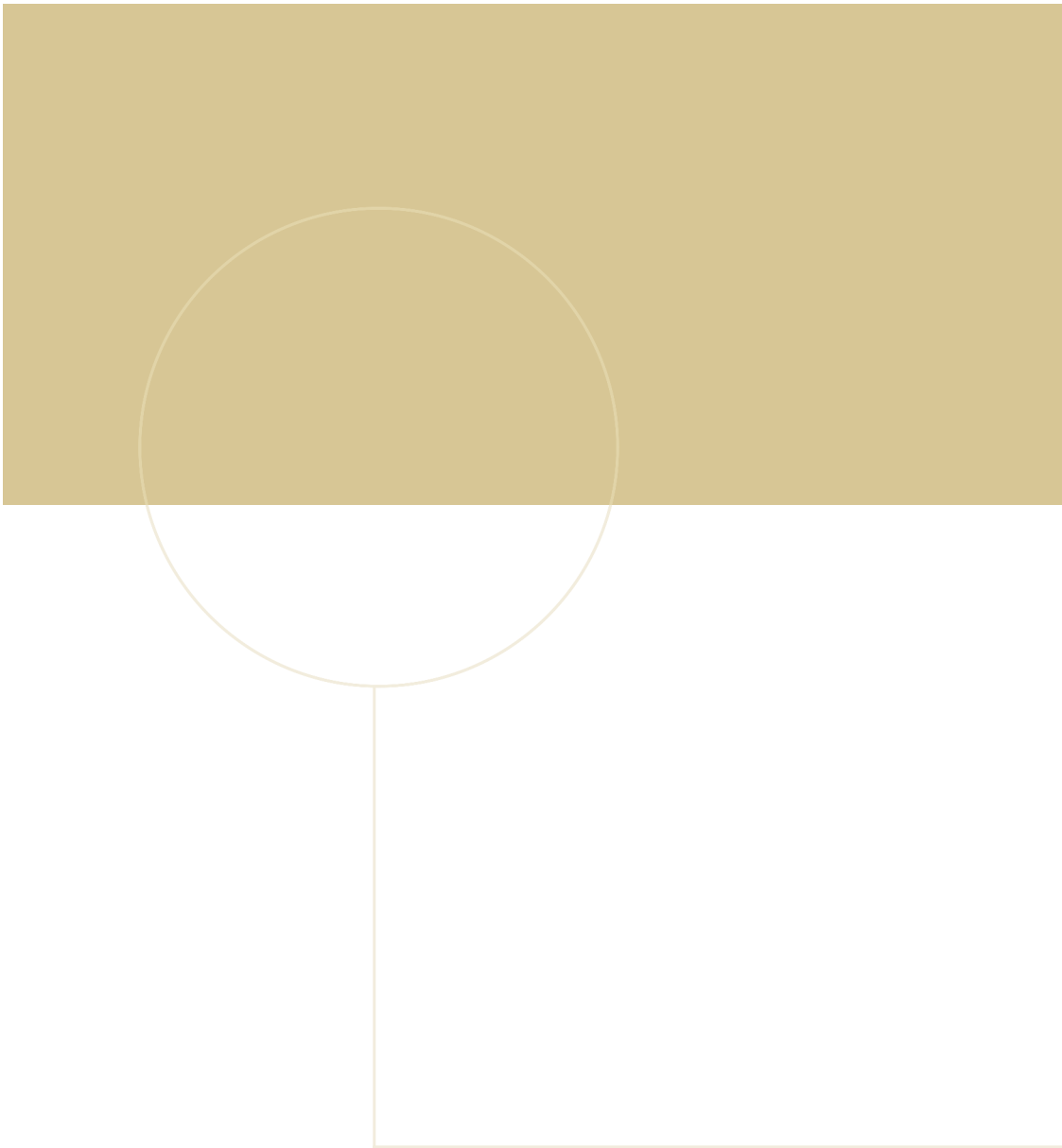
1. Declare system constants
   - Number of particles, $m = 80000$
   - Length in each dimension, $L = 2\pi\,\mathrm{m}$
   - Diffusion parameter, $D = 5 \times 10^{-4}\,\mathrm{m}^2/\mathrm{s}$
   - Time step, $dt = 1 \times 10^{-3}\,\mathrm{s}$
   - Simulation time, $T_{\text{max}} \in [20\,\mathrm{s}, 80\,\mathrm{s}]$
2. The initial positions of each particle are random numbers drawn from a multivariate distribution with variance, $\sigma^2 = 1 \times 10^{-2}\,\mathrm{m}^2$, and mean, $\mu = \pi\,\mathrm{m}$.
3. A collection of particles are scattered to each computer core. E.g. 80000 particles distributed amongst 128 cores results in 625 particles per core. No MPI communication related to the particles is needed except from the initial scattering and the eventual gathering of the particles.
4. Start computing the first time step using the explicit Euler method.
   (a) Receive the velocity field from the DNS solver on one of the computer cores. Broadcast the velocity field to each computer core as explained in section 3.3.5.

   (b) Interpolate velocity values to each particle position from the discrete velocity field.

   (c) Compute the next particle position from the previous position, velocity drift $\mathbf{u}$ and diffusion, D.

   (d) At certain time steps do post-processing such as storing the particle positions to disk.

   (e) Iterate until T_{max}.



NTNU

Norwegian University of
Science and Technology