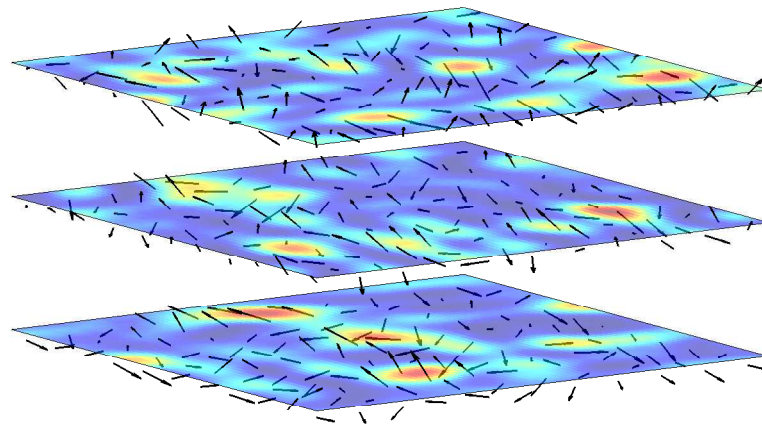




university of  
groningen

faculty of mathematics  
and natural sciences

# The optimal length scale for the Smagorinsky model



Master's thesis

November 2016

Student: R. B. Christoffers

Primary supervisor: Prof. R. W. C. P. Verstappen

Secondary supervisor: Prof. E. C. Wit

## Abstract

The simulation of turbulence is a computationally expensive task. That is why models are used to reduce the resolution of the computation, without losing too much accuracy. One of these models is the Smagorinsky model, which simulates the interaction between the large scales and the small scales that are lost by the reduced resolution. The Smagorinsky model contains the eddy viscosity, which is given by  $\nu_t = (C_S \Delta)^2 \|S\|$ . The value  $\Delta$  is equal to the mesh width on a uniform computational grid, but it is not clear what this value should be on a nonuniform grid. In this research different values for  $\Delta$  are tested on different nonuniform grids. On each grid is homogeneous isotropic turbulence simulated, for which the initial field is derived by a variation of the method in [3]. This variant seems to hold only for a specific set of nonuniform grids. The results of the simulations show that a large value for  $\Delta$  results in an excess of energy at fixed wavenumbers in the spectrum. That is why  $\Delta = \min dx dy dz$  is found to be the best value for  $\Delta$  in this research.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>General theory on computational fluid dynamics</b>	<b>4</b>
2.1	The Navier-Stokes equations . . . . .	4
2.2	Large-Eddy Simulation . . . . .	4
2.3	The Smagorinsky-Lilly model . . . . .	5
2.4	Homogeneous and isotropic turbulence . . . . .	6
2.5	The energy spectrum . . . . .	6
2.6	The scales in the energy spectrum . . . . .	8
2.6.1	The integral scales . . . . .	8
2.6.2	The inertial scales . . . . .	8
2.6.3	The dissipative scales . . . . .	10
<b>3</b>	<b>Constructing the initial velocity field</b>	<b>11</b>
3.1	The initial velocity field on a uniformly spaced grid . . . . .	11
3.1.1	Deriving the modified wavenumber . . . . .	11
3.1.2	Constructing the direction of the Fourier terms . . . . .	13
3.1.3	The amplitude of the Fourier modes . . . . .	14
3.1.4	Completing the velocity field . . . . .	15
3.2	The initial pressure field . . . . .	15
3.3	Problems for nonuniform grids . . . . .	16
3.4	Computing the initial field on a staggered grid . . . . .	18
3.4.1	Using the modified wavenumber on a staggered grid . . . . .	18
3.4.2	Notes on other possible methods . . . . .	19
<b>4</b>	<b>Research</b>	<b>20</b>
4.1	The research . . . . .	20
4.2	The filtered energy spectrum . . . . .	21
4.3	Choices of length scales . . . . .	22
4.4	The Reynolds number . . . . .	24
4.5	Finding the correct Smagorinsky constant . . . . .	24
4.6	Program specific choices . . . . .	25
<b>5</b>	<b>Results</b>	<b>27</b>
5.1	Results on the uniform domain . . . . .	27
5.1.1	The fine solutions . . . . .	27
5.1.2	The optimal solution . . . . .	30

5.2	Results on the nonuniform domain . . . . .	33
5.2.1	Case 1: changing one length . . . . .	34
5.2.2	Case 2: changing two lengths . . . . .	37
5.2.3	Case 3: changing one length between the minimum and maximum of the all the lengths of the domain . . . . .	40
5.2.4	Combined results . . . . .	43
<b>6</b>	<b>Conclusion</b>	<b>45</b>
6.1	Conclusion . . . . .	45
6.2	Discussion . . . . .	45
6.2.1	Choice of constants . . . . .	45
6.2.2	Accuracy . . . . .	46
6.2.3	The choice of LES filter . . . . .	46
<b>A</b>	<b>Implementation of the initial energy spectrum.</b>	<b>48</b>
A.1	Fitting the CBC data . . . . .	48
A.2	Algorithm for the implementation of the spectrum . . . . .	49
A.3	Algorithm for the LES . . . . .	49
<b>B</b>	<b>Derivation of a velocity field on a nonuniform grid</b>	<b>51</b>
<b>C</b>	<b>Source code</b>	<b>54</b>
<b>D</b>	<b>All results from the uniform domain with a regular grid</b>	<b>64</b>

# 1 | Introduction

An important aspect in the field of computational fluid dynamics is turbulence simulation. The problem with turbulence simulation is that the big difference between the length scales which produce turbulence and the scales of the turbulence itself. This means that even for simple problems high resolution grids are necessary to get accurate results. Models are derived to lower the resolution of the used grid and to decrease the computation time, while still keeping accurate results.

The Smagorinsky model is a well-known model which is used for the simulation of turbulence. It was derived by Joseph Smagorinsky in 1963 and was the first large-eddy simulation model. Since then, multiple different models have been derived from the it, like the dynamic Smagorinsky model. But despite the age of the model and the use in other models, not everything is known about it. The model simulates the eddy viscosity with,

$$\nu_t = (C_S \Delta)^2 \|S\|$$

The model and its components are explained later in this research. The main subject of this research is the length scale  $\Delta$ . This length scale is equal to the mesh width on a uniform grid but is not clear for a nonuniform grid. A common choice is the diameter of the grid cell, but this has not been tested thoroughly. Therefore, alternatives of  $\Delta$  are tested in this research on a collection of different nonuniform grids. For the testing of these grids initial velocity fields are necessary, for which a method is known for uniform grids. This method is extended in this research to a set of nonuniform grids and also investigated for general nonuniform grids.

There are better models than the Smagorinsky model for engineering purposes, so direct results may not seem relevant. However, since the Smagorinsky model is the basis for some other models it is still important to investigate the effects of the used grid on the different length scales of a model. This research also derives and investigates a method for the creation of an initial field for a nonuniform grid, which could be used for other models and simulations.

This research is divided into five parts. The first part explains the general terms in fluid dynamics, like the theory of the Smagorinsky model and the Kolmogorov's hypothesis. The second part is about the construction of an initial field for uniform and nonuniform grids. The third part describes the experiments and the choices made. The fourth part contains the results of the experiments. And the final part consists of the conclusions and the discussion of the results.

## 2 | General theory on computational fluid dynamics

### 2.1 The Navier-Stokes equations

The Navier-Stokes equations are used to describe the behavior of fluids. They are of great importance in the field of fluid dynamics. The equations consist of two parts, the momentum equation and the mass equation. Both equations are derived from their corresponding conservation laws. In this research we assume that the fluid is incompressible. Then the momentum equation reads

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla) \mathbf{u} - \nu \Delta \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega \times (0, T) \quad (2.1)$$

Where  $\mathbf{u}$  is the three dimensional velocity field,  $p$  is the pressure,  $\nu$  is the kinematic viscosity and  $\mathbf{f}$  represents all external forces acting on the fluid. For an incompressible fluid the law of conservation of mass is described by

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \times (0, T) \quad (2.2)$$

Equation (2.2) states that a solution of the incompressible Navier-Stokes equation is divergence free; a divergence free field is also called solenoidal. See [1] for a derivation of Equations (2.1) and (2.2).

There are just a few cases for which an exact solution to the Navier-Stokes equations is known, that is why fluid flow problems are approximated by numerical simulations. Numerical simulations of the Navier-Stokes equations with the discretised form of Equations (2.1) and (2.2) are referred to as direct numerical simulations or DNS.

The problem with these kind of simulations is that these simulations have a relative high computational cost. The main reason for the high computational cost is that a solution to a fluid computation problem consists of different scales of flows. The smaller scale flows are called small eddies and although they are small they still have a significant effect on the larger scales. The smallest eddies captured on a grid in a DNS are determined by the mesh width. This means that a fine grid is required for a sufficient solution to a DNS. There are multiple alternatives to DNS, like RANS, LES or DES, which is a hybrid of the previous two.

### 2.2 Large-Eddy Simulation

It was stated in Section 2.1 that simulating all the details of fluid flow requires a fine grid, which results in high computation costs. To avoid this high cost it is necessary to use a coarser grid and simulate the effect of the smaller scales. This is the main idea of the class of models

called the Large-Eddy Simulations (LES), these models are based on the larger scales. The theory of all the large-eddy simulation models is based on applying a filter on the Navier-Stokes equation. The most important properties of these filters is that they commute with space and time differentiation and that they are local in the spatial and frequency domain. Examples of filters as well as other properties and occurring problem are discussed in [1].

The velocity  $\mathbf{u}$  of a flow can be split into the part  $\bar{\mathbf{u}}$  which corresponds to the velocity of the larger scales and  $\mathbf{u}'$  which corresponds to that of the smaller scales. Applying a suitable filter  $G$  to this velocity field results in

$$G(\mathbf{u}) = G(\bar{\mathbf{u}} + \mathbf{u}') = G(\bar{\mathbf{u}}) + G(\mathbf{u}') = \bar{\mathbf{u}}$$

Where  $\bar{\mathbf{u}}$  is the residual of the filter. The theory of large-eddy simulations uses this filter on the Navier-Stokes Equation (2.1) to separate the largest scales from the smallest scales. This leads to

$$\begin{aligned} \bar{\mathbf{u}}_t + (\bar{\mathbf{u}} \cdot \nabla) \bar{\mathbf{u}} + \nabla \cdot (\bar{\mathbf{u}} \bar{\mathbf{u}} - \bar{\mathbf{u}} \bar{\mathbf{u}}) - \nu \Delta \bar{\mathbf{u}} + \nabla \bar{p} &= \bar{\mathbf{f}} \\ \nabla \cdot \bar{\mathbf{u}} &= 0 \end{aligned} \quad (2.3)$$

The additional terms which are not in Equation (2.1) appear because the filter does not commute with the nonlinearity of the convective term. This means that the filtered Navier-Stokes equation still contains terms which depend on the non-filtered velocity field. These terms are known as the subgrid-scale stresses and are modeled by using only the terms of the filtered velocity field. These stresses consist of three parts

$$L = \overline{\bar{\mathbf{u}} \bar{\mathbf{u}}} - \bar{\mathbf{u}} \bar{\mathbf{u}} \quad C = \overline{\bar{\mathbf{u}} \mathbf{u}'} \quad R = \overline{\mathbf{u}' \mathbf{u}'}$$

Called the Leonard stress, cross stress and Reynolds stress respectively. The sum of these stresses is denoted by  $\tau$  and large-eddy Simulations try to model this term. This makes the concerning equation as follows,

$$\begin{aligned} \bar{\mathbf{v}}_t + (\bar{\mathbf{v}} \cdot \nabla) \bar{\mathbf{v}} + -\nu \Delta \bar{\mathbf{v}} + \nabla \bar{p} &= \bar{\mathbf{f}} - \nabla \cdot \tau \\ \nabla \cdot \bar{\mathbf{v}} &= 0 \end{aligned} \quad (2.4)$$

where  $\mathbf{u}$  is replaced by  $\mathbf{v}$  because the equation is no longer solved for the exact solution  $\mathbf{u}$  but for the modeled situation  $\mathbf{v}$ .

## 2.3 The Smagorinsky-Lilly model

There is a wide range of different large-eddy simulation models. For this research the Smagorinsky-Lilly model is chosen because it is one of the first LES models and one of the most well known. Besides that, it is also one of the simpler models and the basis for a few more complex methods. The model is an eddy-viscosity model, which is a set of models based on the Boussinesq hypothesis.

**Boussinesq hypothesis:** *small-scale turbulent stress should be linearly proportional to the mean (large scale) strain rates.* [5]

The large scale strain rate tensor is a three dimensional matrix given by  $\bar{S}_{i,j} = \frac{1}{2}(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i})$ . The hypothesis is derived from the molecular degree of freedom in kinetic theory of gases, where there is a linear relation between the momentum fluxes and the strain rate of the large scales. It is shown in [10] that the hypothesis in general does not hold for turbulence modeling. This is

one reason why better LES models than the Smagorinsky-Lilly model exist, but it can still be accurate for some cases. From the Boussinesq hypothesis it follows that the small-scale turbulent stress is given by

$$\tau_{i,j}^{SGS} = -2\nu_{SGS}\bar{S}_{i,j}$$

Where  $\nu_{SGS}$  is a scalar called the eddy viscosity. Smagorinsky derived in [11] by studying fluid flows in the atmosphere that the eddy viscosity for his model is given by

$$\nu_{SGS} = \sqrt{2}(C_S\Delta)^2|\bar{S}|$$

Where  $\Delta$  is filter-width of the applied Large Eddy Simulation, which is directly proportional to the grid spacing of an uniform grid. The value  $C_S$  is the Smagorinsky constant which in early work was set to be somewhere between 0.1 and 0.2. Later studies and research refined this value and now it is usually set to 0.17. However other studies use 0.2 as can be seen in [8].

## 2.4 Homogeneous and isotropic turbulence

The experiments in this research are performed on homogeneous isotropic turbulence fields. This section explains the definition of both the terms homogeneous and isotropic and elaborates the mathematical consequences. The first part is the homogeneity of the turbulent flow and therefore it is useful to use statistics of a velocity field. A flow is homogeneous turbulent if the statistics of each velocity component are independent of their position in space and time. An example of a fluid flow with inhomogeneous turbulence is a wall flow since the flow close to the wall is more likely to have a turbulence character than the flow far from the wall, where it will behave more like a laminar flow. This is also described as statistically invariant under translation of the homogeneous turbulence field. Closely related to the definition of homogeneous turbulence is the isotropy property. An isotropic flow is a flow without any preference of direction. This is statistically equivalent to an uniform probability distribution for the direction at each location in the velocity field. The amplitude of the components in the velocity field is less of importance as long as there is not a skew relation between the directions of the velocity vector, as demonstrated in Figure 2.1. In this figure it can be seen that the directions northeast, southeast, southwest and northwest are all uniformly represented, but the field is not isotropic. This is because the amplitude of the northeast pointing vectors is dominant over the other directions, which results in a preferred direction for the general flow. If a flow has no preference for any direction than the flow is called statistically invariant under rotation and reflection. This explains why the flow in Figure 2.1 is non-isotropic since the behavior of the general flow changes after any translation or reflection operation on the field.

## 2.5 The energy spectrum

Energy conservation and energy flow are important concepts in fluid dynamics. The general formula for the kinetic energy  $E(\mathbf{x})$  of an unit of mass in a turbulent flow is given by

$$E(\mathbf{x}) = \frac{1}{2}\langle \mathbf{u} \cdot \mathbf{u} \rangle \quad (2.5)$$

It is more practical to look at the average of the energy in a certain number of experiments instead of the energy in the flow in one experiment due to the chaotic behavior of turbulence. The average is denoted in this research by angular brackets. The formula given in Equation (2.5)



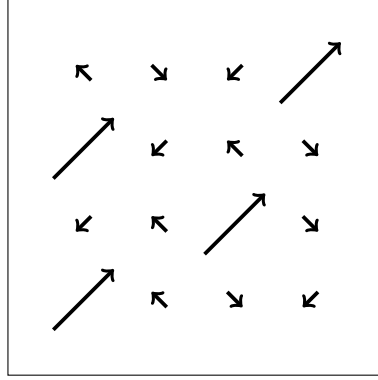


Figure 2.1: An example of an non-isotropic field

is used for the kinetic energy in the spatial domain. In a turbulent flow it is more suitable to look at the kinetic energy in the frequency domain since the energy of a single particle is of less significance than the energy at a certain length scale. There are multiple methods to implement kinetic energy in the frequency domain. The first method is to take the Fourier transform of the kinetic energy over the whole domain. The discretised Fourier transform of a function  $f$  and its inverse is given by

$$\hat{f} = \sum_{\mathbf{x} \in \Omega} f(\mathbf{x}) e^{-i2\pi \mathbf{x} \cdot \mathbf{k}} \quad f = \frac{1}{|\Omega|} \sum_{\mathbf{k} \in \hat{\Omega}} \hat{f}(\mathbf{k}) e^{i2\pi \mathbf{x} \cdot \mathbf{k}}$$

The Fourier transform is the operation which sends the nodes  $\mathbf{x}$  from the spatial domain  $\Omega$  to the nodes  $\mathbf{k}$  in the frequency domain  $\hat{\Omega}$ . The value  $|\Omega|$  is a scalar for the volume of the domain  $\Omega$ . The term  $\hat{f}$  is also known as a Fourier mode of  $f$ . As a result of these equations the Fourier transform of the energy equation is

$$\hat{E}(\mathbf{k}) = \sum_{\mathbf{x} \in \Omega} \langle \mathbf{u}(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x}) \rangle e^{-i2\pi \mathbf{x} \cdot \mathbf{k}}$$

While this formula does give the contribution of each wavenumber to the Fourier transform of the kinetic energy formula, it does not give any direct information about the kinetic energy of the spatial domain. Instead a similar formula to Equation (2.5) is used but in this case for the frequency domain, which reads

$$E(\mathbf{k}) = \langle \hat{\mathbf{u}}(\mathbf{k}) \cdot \hat{\mathbf{u}}^*(\mathbf{k}) \rangle \quad (2.6)$$

where the star denotes the complex conjugate of  $\hat{\mathbf{u}}$ , since the frequency domain allows complex numbers. The formula is similar to Equation (2.5) but is for the frequency domain instead of the spatial domain, which means that the contribution to the energy of frequency space is given for each wavenumber. The relation between Equation (2.5) and (2.6) becomes clear when the total energies over both respected spaces are computed. The following holds when the total energy is computed over a discrete spatial or frequency domain,

$$\sum_{\mathbf{x} \in \Omega} E(\mathbf{x}) = \frac{1}{2|\Omega|} \langle \sum_{\mathbf{x} \in \Omega} \mathbf{u} \cdot \sum_{\mathbf{k} \in \hat{\Omega}} \hat{\mathbf{u}} e^{i2\pi \mathbf{x} \cdot \mathbf{k}} \rangle = \frac{1}{2|\Omega|} \langle \sum_{\mathbf{k} \in \hat{\Omega}} \hat{\mathbf{u}} \cdot \sum_{\mathbf{x} \in \Omega} \mathbf{u} e^{i2\pi \mathbf{x} \cdot \mathbf{k}} \rangle = \frac{1}{2|\Omega|^2} \sum_{\mathbf{k} \in \hat{\Omega}} E(\mathbf{k})$$

where the second equality holds because the summation in the frequency domain commutes with the summation in the spatial domain and since  $\mathbf{u}$  is real it follows that  $\hat{\mathbf{u}}^*(\mathbf{k}) = \sum_{\mathbf{x} \in \Omega} \mathbf{u} e^{i2\pi \mathbf{x} \cdot \mathbf{k}}$ . The relation between the total energies of both spaces follows from the discrete version of the Parseval's theorem, which states that

$$\sum_{\mathbf{x} \in \Omega} |f(\mathbf{x})|^2 = \frac{1}{|\Omega|} \sum_{\mathbf{k} \in \Omega} |\hat{f}(\mathbf{k})|^2$$

There are some generalizations to the energy formula in Equation (2.6) for homogeneous isotropic turbulence flows. These kind of flows do not have a preference for direction, as was noted in Section 2.4. Therefore, when we look at the average over multiple experiments the general behavior is expected to be identical in every direction in the frequency space. This means that the energy spectrum can be expressed in a one dimensional variable  $\bar{k}$  instead of the three dimensional  $\mathbf{k} = [k_1 \ k_2 \ k_3]$ . This is done by integrating over a thin shell of a sphere with radius  $\bar{k} = \|\mathbf{k}\|_2$ . This makes the total energy for all wavenumbers with vector length  $k$

$$E(k) = \frac{2\pi k^2}{N} \sum_{k < \|\mathbf{k}\|_2 < k+\delta} \hat{\mathbf{u}}(\mathbf{k}) \cdot \hat{\mathbf{u}}(\mathbf{k})^* \quad (2.7)$$

where  $\delta$  is the thickness of the shell and  $N$  is the number of elements inside the shell.

## 2.6 The scales in the energy spectrum

This section provides an explanation of the energy spectrum of a homogeneous isotropic turbulence field. The energy spectrum can be divided into three parts, which will all be described in the following subsections:

1. The integral scales
2. The inertial scales
3. The dissipation scales

### 2.6.1 The integral scales

The integral scales are the domain of the largest scales in the spectrum. The three domains do not have exact boundaries, but commonly the length scales in the integral scale are approximately from the order  $\mathcal{O}(1)$  to  $\mathcal{O}(10^{-1})$  of the length of the domain. This means that the scales on this domain are defined by the type of problem. Therefore, this range could consist of scales larger than a kilometer, like in an atmospheric turbulence problem, or could range from a few centimeters for a small pipe flow problem. Generally the scales in integral range have the largest peaks in the energy spectrum. This is because the energy injection to the system is also of the order of these scales and that is why it referred to as the energy-containing range in some literature [7]. The eddies in the integral scales are usually highly anisotropic in contrary to the smaller scales.

### 2.6.2 The inertial scales

In 1941, Andrey Kolmogorov stated a few hypotheses for the behavior of turbulent flows. The first one stated here holds for the wave lengths in the inertial range and smaller.

**Kolmogorov's hypothesis of local isotropy:** *at sufficiently high Reynolds number, the small-scale turbulent motions are statistically isotropic.* [7]

The definition of isotropy is explained in Section 2.4. Note that the theorem does not imply that the whole flow is isotropic but only the small-scale turbulence motions. This is also known as local isotropy. This could seem to contradict popular statements like from Richardson [9], which stated that the big scales influence their smaller scales and the smaller scales influence their smaller scales. From this it may be assumed that since the larger scales may be anisotropic that the effect on their smaller scales can only be in the direction of the larger scales and so the smaller scales could also be anisotropic. Kolmogorov suggested in his hypothesis that the direction biases is lost in the interaction between the larger scales and their smaller scales due to the chaotic behavior of the scale-reduction process. The behavior of the small-scale motions is influenced by the energy transferred from the higher scales or also called the dissipation rate  $\epsilon$ . Furthermore, the viscosity  $\nu$  of any fluid could also effect the smaller scales. Kolmogorov stated that small-scale motions are only influenced by these two variable in his first similarity hypothesis.

**Kolmogorov's first similarity hypothesis:** *In every turbulent flow at sufficiently high Reynolds number, the statistics of the small-scale motions have a universal form that is uniquely determined by  $\nu$  and  $\epsilon$ .* [7]

This means that the small-scale motions are no longer directly influenced by the boundary condition and energy injection from outside the system. From the variables  $\epsilon$  and  $\nu$  it is possible to derive the Kolmogorov length scale  $\eta \equiv (\nu^3/\epsilon)^{1/4}$ . This derivation follows from dimensional analyses and this scale lays in the dissipation range, which will be explained in the Section 2.6.3. Again from dimensional analyses it follows that the dissipation rate is proportional to the length  $l_0$  and velocity  $u_0$  of the largest motions or  $\epsilon \sim u_0^3/l_0$ . Consequently, the ratio between the smallest-scale motions with length of the Kolmogorov length scale and largest motions is proportional to the Reynolds number with  $\eta/l_0 \sim Re^{3/4}$ . This means that for a flow with a high Reynolds number the largest length scales  $l_0$  is much larger than the smallest scales  $\eta$ . From this can be concluded that there is a range with length  $l$  with  $l_0 \gg l \gg \eta$  where the length scales are small enough for the first similarity hypothesis of Kolmogorov to hold, but the local Reynolds number  $lu(l)/\nu$  is still big enough to make the effect of the viscosity flow negligible small. This results in Kolmogorov's second similarity hypothesis.

**Kolmogorov's second similarity hypothesis:** *In every turbulent flow at sufficient high Reynolds number, the statistics of the motions of scale  $l$  in the range  $l_0 \gg l \gg \eta$  have a universal form that is uniquely determined by  $\epsilon$  independent of  $\nu$ .* [7]

With this theorem it is possible to determine a formula for the energy spectrum in the frequency space. This follows once again from dimension analyses. From Equation (2.7) it can be seen that the dimensions of the total energy contained in all Fourier modes with length  $k$  is given by  $[E(k)] = [u]^2[l]$ , the dimension of the rate of energy dissipation is given by  $[\epsilon] = [u]^3/[l]$  and the dimension of the wavelength is given by  $[k] = 1/[l]$ . This results in the general form of the energy spectrum in the inertial domain

$$E(k) = C\epsilon^{2/3}k^{-5/3}$$

where  $C$  is a universal constant, which is found by experiments to be approximately 1.5.

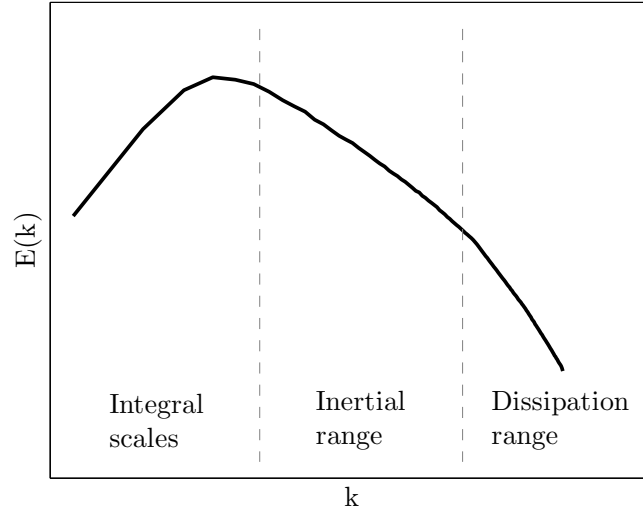


Figure 2.2: The energy spectrum

### 2.6.3 The dissipative scales

The motions in the dissipation range have the smallest length scales. They no longer satisfy Kolmogorov's second similarity hypothesis but do still satisfy the first. Just like the Kolmogorov length scale there exists the Kolmogorov velocity scale, which is given by  $u_\eta = (\epsilon\nu)^{1/4}$ . From this and the Kolmogorov length scale can be seen that the Reynolds number corresponding to the smallest scale in this range equals one, since  $Re = lu/\nu = \eta u_\eta/\nu = (\nu^3/\epsilon)^{1/4}(\epsilon\nu)^{1/4}/\nu = 1$ . Therefore, the viscosity has a dominant effect in this range which results in the energy being dissipated into heat. The general form of energy spectrum of a homogeneous isotropic field can be seen in Figure 2.2 as seen in [4].

## 3 | Constructing the initial velocity field

### 3.1 The initial velocity field on a uniformly spaced grid

The method presented for the construction of an initial velocity field was proposed in [3]. The aim was to derive a random velocity field which was divergence free under its discretisation. The energy spectrum must be adjustable without interfering with the isotropy or divergence condition. This method was used in [3] to construct an initial field for a three-dimensional turbulence simulation. The domain has periodic boundary conditions. The method to derive a suitable initial field consists of four steps, which will be explained in the following sections.

1. We take the Fourier transform of the solenoidal property to construct the modified wavenumber.
2. We choose an arbitrary vector from the space orthogonal to the modified wavenumber.
3. The corresponding energy spectrum is implemented.
4. The field is modified without loss of the desired properties such that the field is real after the inverse Fourier transformation.

#### 3.1.1 Deriving the modified wavenumber

The method in [3] starts with an undefined three-dimensional discrete velocity field  $\mathbf{u} = [u \ v \ w]^T$  on a uniform grid with mesh width  $\Delta$ . The grid has  $N_x$ ,  $N_y$  and  $N_z$  points in the  $x, y$  and  $z$  direction respectively. The velocity in each point  $\mathbf{u}(\mathbf{x})$  can be represented by Fourier modes as can be seen in Equation (3.1)

$$\mathbf{u}(\mathbf{x}) = \frac{1}{N_x N_y N_z} \sum_{\mathbf{k} \in \hat{\Omega}} \hat{\mathbf{u}}(\mathbf{k}) e^{i2\pi \mathbf{k} \cdot \mathbf{x}} \quad (3.1)$$

where  $\hat{\Omega}$  is a multi-index set consisting of all the wavenumbers. Since the computational domain is three-dimensional, the frequency space is also three-dimensional. It is not necessary to have equidistant frequencies, but it is assumed it is for the ease of the derivation. The wavenumbers  $k$  are scaled by the length of the computational domain, which is  $\Delta_i N_i$ . The wavenumbers are shifted to be almost symmetrical around 0, which is necessary to get a real solution in the spatial domain. This makes the multi-index set of the wavenumbers

$$\widehat{\Omega} = \left\{ \mathbf{k} = [k_1 \ k_2 \ k_3]^T \mid \frac{-1}{2\Delta_i} \leq \frac{k_i}{\Delta_i N_i} \leq \frac{1}{2\Delta_i} - \frac{1}{\Delta_i N_i} \text{ for } k_i \in \mathbb{Z} \right\}$$

The Fourier modes of  $\mathbf{u}$  are computed in a similar way by taking the sum over the nodes instead of the frequencies.

$$\widehat{\mathbf{u}}(\mathbf{k}) = \sum_{\mathbf{x} \in \Omega} \mathbf{u}(\mathbf{x}) e^{-i2\pi \mathbf{k} \cdot \mathbf{x}} \quad (3.2)$$

The Fourier transform of the solenoidal property in Equation (2.2) results in  $\mathbf{k} \cdot \widehat{\mathbf{u}}(\mathbf{k}) = 0$ . This means that the Fourier transform of the velocity  $\mathbf{u}$  will always be perpendicular to the wavenumbers  $\mathbf{k}$  in a continuous solenoidal field. However, since Equation (2.2) is approximated by a discretisation, the relation differs. The derivatives can be discretised in different ways, which leads to different relations between the wavenumbers and the Fourier transform of velocity field. In the research in [3] a second order central discretisation is used as demonstrated in Equation (3.3). This method will also be used for the further explanation of the construction of the initial field. Higher and lower order discretisations are also possible as long as similar derivation are possible for them.

$$\frac{\partial u}{\partial x} \simeq \frac{u_{i-2} - 8u_{i-1} + 8u_{i+1} - u_{i+2}}{12\Delta_x} \quad (3.3)$$

The Fourier transform of this discretisation results in

$$\widehat{\frac{\partial u}{\partial x}} \simeq \frac{\widehat{u_{i-2}} - 8\widehat{u_{i-1}} + 8\widehat{u_{i+1}} - \widehat{u_{i+2}}}{12\Delta_x} \quad (3.4)$$

Note that since the simulations in [3] are done on a uniform grid so  $\Delta_x$  is constant over the whole computational domain and in every direction. Hence, the grid width commutes with the Fourier transform. This property however does not hold on a non-uniform grid, which will be studied later in this research. The benefit of evaluating the discretisation in the frequency space is that it is possible to express Fourier modes in terms of their neighbouring points. From Equation (3.2) results

$$\begin{aligned} \widehat{u_{i+1}}(\mathbf{k}) &= \sum_{\mathbf{x} \in \Omega} u_{i+1}(\mathbf{x}) e^{-i2\pi \mathbf{k} \cdot \mathbf{x}} \\ &= \sum_{\mathbf{x} \in \Omega} u_i(\mathbf{x}) e^{-i2\pi(\mathbf{k} \cdot \mathbf{x} - \frac{k_1}{\Delta N_x} \Delta)} \\ &\quad + \sum_{\mathbf{x} \in \Gamma_1} u_i(\mathbf{x}) e^{-i2\pi(\mathbf{k} \cdot \mathbf{x} - \frac{k_1}{\Delta N_x} \Delta)} - \sum_{\mathbf{x} \in \Gamma_2} u_i(\mathbf{x}) e^{-i2\pi(\mathbf{k} \cdot \mathbf{x} - \frac{k_1}{\Delta N_x} \Delta)} \end{aligned} \quad (3.5)$$

where  $\Gamma_1$  is the set of nodes left from the computational domain  $\Omega$  and  $\Gamma_2$  are the nodes on the right side in the domain, see Figure 3.1. Since the boundary conditions are periodic it holds that  $\mathbf{u}(\mathbf{x}_0) = \mathbf{u}(\mathbf{x}_0 + \Delta[N_x \ 0 \ 0]^T) = \mathbf{u}(\mathbf{x}_{N_x})$  for  $\mathbf{x}_0 \in \Gamma_1$  and  $\mathbf{x}_{N_x} \in \Gamma_2$ . From this it can be seen that any node in  $\Gamma_1$  corresponds uniquely to a node in  $\Gamma_2$ , with the same  $y$  and  $z$  coordinate and with the same velocity. The only difference between the summations over the sets  $\Gamma_1$  and  $\Gamma_2$  could come from the  $x$  coordinate in the exponent. However, the following equation shows that these two exponents are equal.

$$e^{-i2\pi \mathbf{k} \cdot \mathbf{x}_{N_x}} = e^{-i2\pi(\mathbf{k} \cdot \mathbf{x}_0 + \frac{k_1}{\Delta N_x} \Delta N_x)} = e^{-i2\pi(\mathbf{k} \cdot \mathbf{x}_0)} e^{-i\pi k_1} = e^{-i2\pi(\mathbf{k} \cdot \mathbf{x}_0)}$$

Consequently, Equation (3.5) can be simplified to

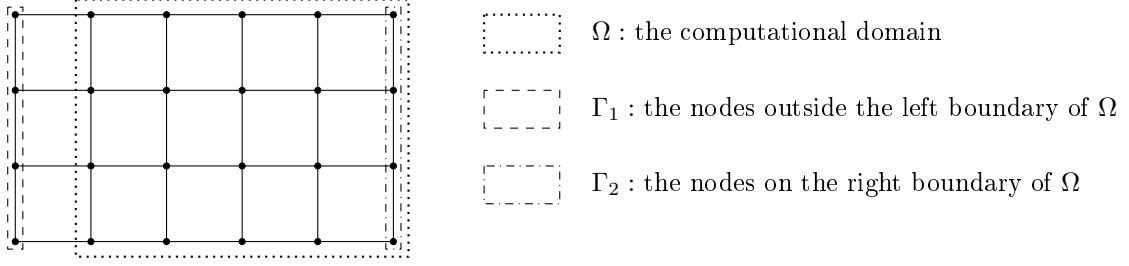


Figure 3.1: The domains  $\Omega, \Gamma_1$  and  $\Gamma_2$

$$\begin{aligned}
\widehat{u_{i+1}}(\mathbf{k}) &= \sum_{\mathbf{x} \in \Omega} u_i(\mathbf{x}) e^{-i2\pi(\mathbf{k} \cdot \mathbf{x} - \frac{k_1}{\Delta N_x} \Delta)} \\
&\quad + \sum_{\mathbf{x} \in \Gamma_1} u_i(\mathbf{x}) e^{-i2\pi(\mathbf{k} \cdot \mathbf{x} - \frac{k_1}{\Delta N_x} \Delta)} - \sum_{\mathbf{x} \in \Gamma_2} u_i(\mathbf{x}) e^{-i2\pi(\mathbf{k} \cdot \mathbf{x} - \frac{k_1}{\Delta N_x} \Delta)} \\
&= \left( \sum_{\mathbf{x} \in \Omega} u_i(\mathbf{x}) e^{-i2\pi \mathbf{k} \cdot \mathbf{x}} \right) e^{i2\pi k_1 / N_x} \\
&= \widehat{u_i}(\mathbf{k}) e^{i2\pi k_1 / N_x}
\end{aligned} \tag{3.6}$$

Similar relations can be derived in directions of  $v$  and  $w$  and also between  $\widehat{u_i}(\mathbf{k})$  and  $\widehat{u_{i+2}}(\mathbf{k})$ . Using these relations on the earlier given Fourier transform of the discretised first order derivative results in

$$\frac{\widehat{u_{i-2}} - 8\widehat{u_{i-1}} + 8\widehat{u_{i+1}} - \widehat{u_{i+2}}}{12\Delta_x} = i \frac{8 \sin \frac{2\pi k_1}{N_x} - \sin \frac{4\pi k_1}{N_x}}{6\Delta_x} \widehat{u}(\mathbf{k}) = K_1 \widehat{u}(\mathbf{k}) \tag{3.7}$$

where the element  $K_1$  is the first element of the modified wavenumber. The other elements follow from the discretisations of the derivatives of  $v$  and  $w$ . It can be concluded that in a discrete velocity field the Fourier modes are orthogonal to the modified wavenumber vector  $K = [K_1(k_1)K_2(k_2)K_3(k_3)]$ , which depends on the real wavenumber  $k = [k_1, k_2, k_3]$ .

### 3.1.2 Constructing the direction of the Fourier terms

The second step in the construction of a divergence free isotropic velocity field is to choose at each wavenumber a vector orthogonal to the modified modified wavenumber, which was defined in the previous section. The set of all vectors orthogonal to the modified wavenumber vector form a plane in the three-dimensional real frequency space. It is possible to derive all divergence free vectors from the Fourier modes in span of the vectors in this plane and its complex variant. Taking any such vector at random results in an isotropic velocity field if the field contains enough nodes. An efficient method to take a random Fourier mode is to solve the real and imaginary part separately. These vectors are taken such that they have unit length because the amplitude will not influence the orthogonality and by using this length could the real and complex components be combined in the end. The functions in Equation (3.8) are used as a method to find a fixed unit vector  $\widehat{\mathbf{u}}(\mathbf{k})$  in the orthogonal real plane for each wavenumber  $k$ . The only exception is  $k = 0$  for which we get  $\widehat{\mathbf{u}}(0) = 0$ .

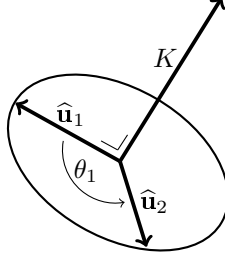


Figure 3.2: Constructing an arbitrary orthogonal vector

$$\begin{aligned}
\hat{u} &= \frac{1}{\sqrt{2K_1^2 + K_2^2 + 2K_2K_3 + K_3^2}} \begin{bmatrix} K_2 + K_3 \\ -K_1 \\ -K_1 \end{bmatrix} \\
\hat{v} &= \frac{1}{\sqrt{2K_2^2 + K_1^2 + 2K_1K_3 + K_3^2}} \begin{bmatrix} -K_2 \\ K_1 + K_3 \\ -K_2 \end{bmatrix} \\
\hat{w} &= \frac{1}{\sqrt{2K_3^2 + K_1^2 + 2K_1K_2 + K_2^2}} \begin{bmatrix} -K_3 \\ -K_3 \\ K_1 + K_2 \end{bmatrix}
\end{aligned} \tag{3.8}$$

The obtained vector  $\hat{\mathbf{u}}_1$  is then rotated around the axis  $K$  with an arbitrary angle  $\theta_1$ . This results in the vector  $\hat{\mathbf{u}}_2$  which represents the real part of the random unit vector. The same method is then repeated with the same  $\hat{\mathbf{u}}_1$  but with a different angle  $\theta_2$  to get vector  $\hat{\mathbf{u}}_3$  which represents the imaginary part. Both parts are combined to one vector with unit length by a third random angle  $\theta_3$  such that  $\hat{\mathbf{u}} = \sin \theta_3 \hat{\mathbf{u}}_2 + i \cos \theta_3 \hat{\mathbf{u}}_3$ . This method makes it possible to get the set of Fourier modes with unit length which corresponds to a divergence free velocity field. By changing the amplitude of the Fourier mode it is possible to extent this set to the whole space of Fourier modes which lead to a divergence free velocity field.

### 3.1.3 The amplitude of the Fourier modes

In the previous section was found the whole set of all unitary Fourier modes which lead to a divergence free velocity field. The Fourier modes can be altered such that their spectrum corresponds to that of a homogeneous isotropic turbulence flow. This is done by changing the amplitude of the Fourier modes, which does not change their solenoidal property. The amplitudes can be derived from the relation between the Fourier modes and the energy stated in Equation (2.7). For this an energy spectrum is required, which is similar to Figure 2.2, to be relevant for a homogeneous isotropic flow. The limitation of the obtained relation is that it is one dimensional. To find a three-dimensional field we can divide the energy spectrum in the maximum number of intervals such that each interval contains at least one wavenumber corresponding to the discrete frequency space. The following assumption is made in this research:

**Assumption for the energy distribution:** *The energy in an interval in the spectrum as defined above should be divided equally over all relevant wavenumbers corresponding to*



*this range to ensure that the field is homogeneous isotropic.*

This assumption is clear for a grid with the same number of nodes in each direction, since taking any other distribution of the energy will result in a dominant direction in the frequency space, which will have an impact on the spatial domain. This assumption makes it possible to fix the amplitude of the Fourier modes if an energy spectrum function is available. See Appendix A for the choice of the energy spectrum function for this research. The aim is to derive a homogeneous isotropic velocity field which is a velocity field with no preference for direction in the spatial domain. This means that the frequency domain is homogeneous isotropic as well. Equation (2.7) then reads for a homogeneous isotropic initial velocity field:

$$E(k) = 2\pi k^2 \|\hat{\mathbf{u}}(\mathbf{k})\|_2^2$$

for any vector  $\mathbf{k}$  in the frequency domain with length  $k$ . With this function it is possible to derive the correct Fourier modes for each wavenumber such that the corresponding velocity field is homogeneous isotropic and has the correct energy spectrum.

### 3.1.4 Completing the velocity field

The last step of the construction of the initial velocity field is to ensure that the velocity field will be real. This can be achieved by computing halve the Fourier modes and derive the complete Fourier space by using the complex conjugate with  $\hat{\mathbf{u}}(\mathbf{k}) = -\hat{\mathbf{u}}^*(\mathbf{k})$ . Any complex component which could appear in Equation (3.1) at a wavenumber  $\mathbf{k}$  is canceled out by the complex term of the Fourier modes at wavenumber  $-\mathbf{k}$ . The only modes in the frequency domain which do not have a negative counter part are the modes with  $k_i = -N_i/2$  for any  $i$ . The contribution of these modes to the total energy is small and reduces further for finer grids. That is why these modes are set to zero and they are not included in the second step of the method described above for the construction of the initial field. Using Equation 3.1 will complete the construction of the real homogeneous isotropic velocity which also has the desired spectrum.

## 3.2 The initial pressure field

Section 3.1 explained how the initial velocity field  $\mathbf{u}$  can be constructed. Other variables like the viscosity, density and external forces of the fluid are assumed to be known for the simulation. The pressure is not known beforehand and has to be derived at every time iteration including for the initial step. Deriving the pressure at any point in the initial velocity field can be done by using the Navier-Stokes equations and the velocity field derived in the current time step. The problem for the initial time step is that the Navier Stokes equation includes a time derivative as can be seen in Equation (2.1). Since the discretisation of this time derivative requires at least two complete velocity fields and in the initial step only one is given it follows that the initial pressure can not be derived the same way as the pressure at other iterations. This problem can be solved by taking the divergence of Equation (2.1). Because the time derivative and divergence operator commutes it follows through the solenoidal property that the time derivative vanishes. For the Navier-Stokes equation without any model and external forces this results in

$$\Delta p = \nabla \cdot \left( \Delta \mathbf{u} - \nu(\mathbf{u} \cdot \nabla) \mathbf{u} \right) \quad (3.9)$$

Solving this Poisson equation can be done by using the Fourier transform. Section 3.1 showed that the Fourier transform of an exact derivative is equivalent to an inner product between the

wavenumber vector  $\mathbf{k}$  in the frequency domain, while the Fourier transform of the discretised derivative results in an inner product with a modified wavenumber  $K(\mathbf{k})$ . Since the Laplacian is the same as a double derivative there are multiple methods for the approximation of the right-hand side. The first method uses the exact representation of the derivatives of the Laplacian. This works and is sufficient for an exact solution, but does not hold for the discrete solution. An alternative method is to use the discretisation from Equation 3.3 twice to derive a second order derivative discretisation. Using this method will give a discrete solution for the pressure field, without any additional error [3]. The discretisation in a single direction is the following,

$$\frac{1}{144\Delta^2} \left( p_{i-4} - 16p_{i-3} + 64p_{i-2} + 16p_{i-1} - 130p_i + 16p_{i+1} + 64p_{i+2} - 16p_{i+3} - 16p_{i+4} + p_{i+4} \right)$$

By taking the Fourier transform of this discretisation and using Equation (3.6) follows,

$$\frac{1}{72\Delta^2} \left( -65 + 16 \cos 2\pi\Delta k_1/N_x + 64 \cos 4\pi\Delta k_1/N_x - 16 \cos 6\pi\Delta k_1/N_x + \cos 8\pi\Delta k_1/N_x \right) \hat{p} = K' \hat{p}$$

where  $K'$  has similarities to the modified wavenumber in Section 3.1. Since the right-hand side of Equation (3.9) is known for the initial field from the method described in Section 3.1 it follows that the Fourier modes of  $p$  can be computed and therefore the initial spatial pressure field.

### 3.3 Problems for nonuniform grids

The method described in Section 3.1 was based on the discretised version of the solenoidal property in Equation (2.2). For a uniform grid this property is global and so holds on the whole domain. This explained why the grid width commutes with the Fourier transform in Equation (3.4). For the nonuniform grid the discretized solenoidal property is different between at least two points in the domain. This means that the grid width in general no longer commutes with the Fourier transform, which makes it harder to separate the velocity components from the components which strictly depend on the wavenumber and mesh width, similarly to Equation (3.4). To explain other problems for the nonuniform grid we look at a two dimensional problem with a second order discretisation. The discretised solenoidal free velocity is then given by

$$\frac{u(\xi_{i+1,j}) - u(\xi_{i-1,j})}{\xi_{i+1,j} - \xi_{i-1,j}} + \frac{v(\xi_{i,j+1}) - v(\xi_{i,j-1})}{\xi_{i,j+1} - \xi_{i,j-1}} = 0 \quad (3.10)$$

where  $\xi$  corresponds to any nonuniform grid. The aim is to create a homogeneous isotropic velocity field so the spectrum needs to satisfy the properties stated in Section 2.5. That is why the Fourier transform is needed to translate Equation (3.10) to the frequency domain. There are some nonuniform grids for which the grid width does commute with the Fourier inverse, namely when  $\xi_{i,j} - \xi_{i-1,j} = \xi_{i+2n,j} - \xi_{i-1+2n,j}$  and  $\xi_{i,j} - \xi_{i,j-1} = \xi_{i,j+2n} - \xi_{i,j-1+2n}$  holds for  $n \in \mathbb{N}$  and any node in the grid. The previous equalities contain more terms for higher order discretisations and since these equalities hold for any node it follows that the only cases for which the grid width term commutes with the Fourier transform are the cases when the grid is uniform. In general it holds in the nonuniform Fourier transform that the terms for the grid and for the velocity are hard to separate. This can be solved by taking the transform over a uniform grid  $\mathbf{x}$  with  $\xi_{i,j} = \phi(\mathbf{x}_{i,j})$ , where  $\phi$  is a transformation between the uniform and nonuniform grid. The Fourier transform of Equation (3.10) over the uniform grid will become

$$\begin{aligned}
0 = & \mathcal{F} \left( \frac{1}{\phi(\mathbf{x}_{i+1,j}) - \phi(\mathbf{x}_{i-1,j})} (u(\phi(\mathbf{x}_{i+1,j})) - u(\phi(\mathbf{x}_{i-1,j}))) \right) \\
& + \mathcal{F} \left( \frac{1}{\phi(\mathbf{x}_{i,j+1}) - \phi(\mathbf{x}_{i,j-1})} (v(\phi(\mathbf{x}_{i,j+1})) - v(\phi(\mathbf{x}_{i,j-1}))) \right)
\end{aligned} \tag{3.11}$$

The benefit of a Fourier transform over this uniform grid is that the convolution theorem holds. The discrete convolution between two functions  $f$  and  $g$  in a two dimensional domain is given by

$$(f * g)[n, m] = \sum_{s=1}^N \sum_{t=1}^M f[n-s, m-t] g[s, t]$$

The convolution operation also exists for other dimensions. The convolution theorem states that  $\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$  and  $\mathcal{F}(fg) = \mathcal{F}(f) * \mathcal{F}(g)$ . Applying the last statement on Equation (3.11) separates the grid and velocity terms. If  $\xi_{i,j}$  already corresponded to a uniform grid then one of the parts separated by the convolution theorem is independent of  $\mathbf{x}$ , which means that its Fourier transform equals the Kronecker Delta function times a constant. From this results the earlier established relation  $K \cdot \hat{\mathbf{u}}(\mathbf{k}) = 0$ . The modified wavenumber also needs to be derived in the case that  $\xi_{i,j}$  corresponds to a nonuniform grid. This follows from the term containing the velocity in Equation (3.11) which is split from the grid term by the convolution theorem. The neighbouring principle derived in Equation (3.6) does in general not hold on a nonuniform grid since the distance between  $\xi_{i,j}$  and its neighbours is not constant, but since the Fourier transform was taken over  $\mathbf{x}_{i,j}$  it follows that  $\mathcal{F}(u(\xi(\mathbf{x}_{i+1,j}))) (\mathbf{k}) = \mathcal{F}(u(\xi(\mathbf{x}_{i,j}))) (\mathbf{k}) e^{2\pi\Delta k_1}$ , where  $\Delta$  is the distance between nodes in the uniform grid corresponding to  $\mathbf{x}$ . From this we can rewrite (3.11) as

$$\mathcal{F} \left( \frac{1}{\phi(\mathbf{x}_{i+1,j}) - \phi(\mathbf{x}_{i-1,j})} \right) * (\sin 2\pi\Delta k_1 \hat{u}^*) + \mathcal{F} \left( \frac{1}{\phi(\mathbf{x}_{i,j+1}) - \phi(\mathbf{x}_{i,j-1})} \right) * (\sin 2\pi\Delta k_2 \hat{v}^*) = 0 \tag{3.12}$$

where  $\hat{u}^*$  and  $\hat{v}^*$  are the result of the Fourier transform of  $u$  and  $v$  taken on the uniform grid. One of the problems for the nonuniform grid is that the relation between the spectrum and the Fourier modes is not known. When  $\hat{u}$  and  $\hat{v}$  are obtained from the nonuniform Fourier transform on the grid corresponding to  $\xi_{i,j}$ , then the relation stated in Equation (2.7) holds. However, in general it does not hold that  $\hat{u} = \hat{u}^*$  and  $\hat{v} = \hat{v}^*$  because  $\hat{u} = \sum_{\xi} u(\xi) e^{-2\pi i \mathbf{k} \xi / L_{\xi}} \neq \sum_{\mathbf{x}} u(\phi(\mathbf{x})) e^{-2\pi i \mathbf{k} \mathbf{x} / L_{\mathbf{x}}} = \hat{u}^*$ . An additional problem is that the derivation on the uniform grid contained an inner product between the velocity field and the modified wavenumber, while for the nonuniform grid it is necessary to derive the complete kernel of the sum of two convolution operations. It is possible to find a solution, but it is significantly harder to find the entire set which is needed to pick an element at random to get the final field homogeneous isotropic. This means that the method above is not suitable for a general nonuniform grid but could still be used to derive one nonuniform grid for which the method does work. This is when the mapping  $\phi$  is  $\phi(\mathbf{x}_i) = \Delta_i \mathbf{x}_i$  for any real  $\Delta_i$ . The nonuniform grid which follows from this operation consists of equally shaped rectangles and is also called a regular grid. On a regular grid the convolution in Equation (3.12) is simplified to

$$\left[ \frac{\sin 2\pi\Delta k_1}{2\Delta_1} \quad \frac{\sin 2\pi\Delta k_2}{2\Delta_2} \right] \cdot \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = 0$$

which is similar to the equation on the uniform grid. In Appendix B is an example of derivation of an initial field on a nonuniform grid but which is too inefficient for practical use.

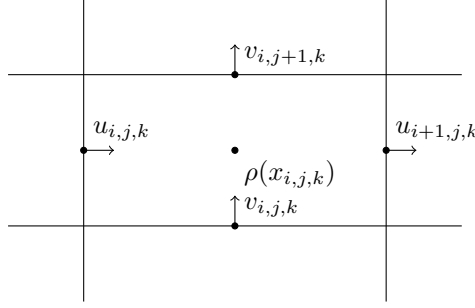


Figure 3.3: Example of a cell in a staggered grid

### 3.4 Computing the initial field on a staggered grid

The method described in Section 3.1 is specific for a collocated grid, which means that the velocity component  $u, v$  and  $w$  scalar variables, like the pressure, are all located on the intersections of the grid lines. A problem which could arise from using a collocated grid is odd-even decoupling, which is a discretisation error where the odd and even numbered elements act significant differently from each other. A solution to this problem is to use a staggered grid. A staggered grid divides the domain in cells with at the center of each cell the scalar variables, like pressure and density, and the velocity components each in the centers of their corresponding faces of the cell, as can be seen in Figure 3.3. There are multiple ways to use the method explained in Section 3.1 to construct an initial field for a staggered grid. Most of these methods use a variant of the modified wavenumber. This section describes one method which is similar to the method in Section 3.1 and preserves the same properties.

#### 3.4.1 Using the modified wavenumber on a staggered grid

The discretisation used in Equation (3.3) is chosen such that it was intuitive for the solenoidal property to hold at the grid nodes of a collocated grid. Using the same discretisation on a staggered grid gives the first order derivatives which are located at the faces of the cell where they are derived, so the solenoidal property is defined as the sum over velocity components which are not located at the same location. A better discretisation for the first order derivatives is one which uses the direct faces of the cell and its neighbouring cells for a higher order discretisations. This yields,

$$q_i = \frac{-u_{i+1} + 27u_i - 27u_{i-1} + u_{i-2}}{24\Delta_x} \quad (3.13)$$

which is also a fourth order discretisation. With this discretisation  $\partial_x u$ ,  $\partial_y v$  and  $\partial_z w$  are all located in the center of the cells in the staggered grid. The solenoidal property corresponds in this case to the centers of each cell, which makes it possible to take the Fourier transform of the sum of Equation (3.13) and the other corresponding derivatives. From this Fourier transform does not directly follow a similar relation for  $\hat{u}$ ,  $\hat{v}$  and  $\hat{w}$  as in Equation (3.4), since those terms are found by transforming over their corresponding face instead of the cell center. This can be solved on a uniform grid by stating that

$$\sum_{\mathbf{x} \in \Omega_1} q_i e^{-2i\pi \mathbf{k} \cdot \mathbf{x}} = e^{i\pi k_1 \Delta_x} \sum_{\mathbf{x} \in \Omega_2} q_i e^{-2i\pi \mathbf{k} \cdot \mathbf{x}} \quad (3.14)$$

where  $\Omega_1$  is the set of all the nodes in the center of the cells and  $\Omega_2$  the set of all points in the center of the left face of each cell. The neighbouring principle of Equation (3.6) still holds on a staggered grid. Using this equation with the previous statement for the solenoidal equation gives the following definition for modified wavenumber on a staggered grid.

$$K_i = \left[ \frac{27 \sin(\pi k_i \Delta_{x_i}) - \sin(3\pi k_i \Delta_{x_i})}{12\Delta_{x_i}} \right]$$

Different discretisation will also result in different modified wavenumbers. The fourth order discretisation is chosen in this research for its accuracy and also for the similarities with the research in [3]. Besides the change of the modified wavenumber it is also necessary to take a Fourier transform to the corresponding staggered grid. This can be done by first using the inverse of the shift in Equation (3.14) and then apply the Fourier transform.

### 3.4.2 Notes on other possible methods

The method described above creates a velocity field with the same properties as the one in Section 3.1. Other methods could be used to construct a velocity field, but may result in a loss of one or more of these properties.

*Halving the mesh width:* It is possible to derive a staggered velocity field if a grid is given with double the amount of grid points in each direction. A subset of the solution given by the velocity field forms a isotropic field and by using a suitable modified wavenumber it is possible to make the field solenoidal. The downside to this method is that the corresponding spectrum is no longer certain and doubling the number of nodes increases the computation time.

*Shifting a collocated grid:* Another option is to change the modified wavenumber such that the solenoidal property lies between two grid nodes on a otherwise collocated grid. Using a shift transformation on the computed velocity field could result in a velocity field for a staggered grid. The problem is that this method only works for lower order discretisations or could result in a complex modified wavenumber.

## 4 | Research

### 4.1 The research

The aim of this research is to find the best length scale for the Smagorinsky model on different kind of grids. The different length scales are chosen based on the known length scale for the uniform case. An exact solution is needed to test the results of the LES done with the Smagorinsky model. This exact solution is approximated by a fine simulation without any model, which is also referred to as a direct numerical simulation or DNS. As stated in Section 3.3, it is hard to compute a velocity field for an arbitrary anisotropic grid, so we test the different length scales on regular grids. The tests on the regular grids can be divided in two main categories:

1. *Constant domain:* The simulations in this set are done on a uniform domain with a varying number of grid points. The lengths of the domain for all simulations are  $l_x = l_y = l_z = 1$ . The number of grid points for the fine simulation is  $N_x = N_y = N_z = 256$  and the uniform LES computations are done with  $N_x = N_y = N_z = 64$ . Nonuniform grids are created by dividing the number of grid nodes by two or four. The number of nodes has to be a power of 2 to ensure that the fast Fourier transform works. The benefit to the simulations in this set is that every LES on any grid approximates the fine simulation on the grid with  $N_x = N_y = N_z = 256$ . The disadvantage is that the grid can only be coarsened a few times without getting a too high numerical error.
2. *Constant number of nodes:* The other set of tests are done with a varying domain. The uniform computations in this set are the same as in the other set with  $N_x = N_y = N_z = 256$  for the fine simulation,  $N_x = N_y = N_z = 64$  for the LES and a uniform domain with  $l_x = l_y = l_z = 1$ . The other simulations keep the same number of nodes for the fine simulation and LES and vary in lengths of the domain. This set of tests can be categorized in three cases and all of them have  $l_y = 1$ . The first case only changes  $l_x$  with  $l_x = \{1, 2, \dots, 10\}$  and has  $l_z = l_y$ , the second case also changes  $l_z$  with the same rate as  $l_x$  and the last case has  $l_x = 10$  and  $l_z$  varies the same as  $l_x$  did in the other cases. The benefit to the simulations in this set are that more simulations are possible compared to the number of simulations in the other set because there is no restriction on the length of the domain. The disadvantage is that the simulations on different domains do not correspond to each other. The simulations with the constant domain and different choices of grid all simulated the same flow, while the flows in this set have different length scales which lead to different characteristics.

The initial spectrum for all simulations are as described in Appendix A.

## 4.2 The filtered energy spectrum

The initial field of a large-eddy simulations differs from the initial field of the fine simulation. If the method for the initial field of the fine solution is also used for the initial field of the large-eddy simulation then the resulting spectrum has less energy at the highest obtainable wavenumbers than the fine solution has at the same wavenumbers. The velocity field of a LES  $\bar{\mathbf{u}}$  is the result of filtering a velocity field  $\mathbf{u}$  with a filter  $G$  such that the effect of the higher wavenumbers is negated, while the lower wavenumbers are preserved. The choice of this filter affects the filtered energy spectrum and the relation with it to the non filtered spectrum. An usual choice for a filter is the following variant of a three-dimensional Gaussian filter

$$G(x) = \left( \sqrt[3]{\frac{\gamma}{\pi}} \frac{1}{\Delta_A} \right)^3 \exp \left( -\frac{\gamma x^2}{\Delta_A^2} \right)$$

Where  $\gamma$  is a constant which is often set to 6 and  $\Delta_A$  is the filter length scale. Filters are applied by computing the convolution between the velocity field and the filter. From the convolution theorem it then follows that for the Fourier transform of the filtered velocity field is

$$\mathcal{F}(\bar{u}) = \mathcal{F}(G * u) = \mathcal{F}(G) \cdot \mathcal{F}(u) = \exp \left( -\frac{\Delta_A^2}{4\gamma} k^2 \right) \cdot \hat{u}$$

From this relation between the filtered and unfiltered velocity field and Equation 2.6 results the filtered energy spectrum  $\bar{E}(\mathbf{k})$  is

$$\bar{E}(\mathbf{k}) = \exp \left( -\frac{(k_1 \Delta_{A,1})^2 + (k_2 \Delta_{A,2})^2 + (k_3 \Delta_{A,3})^2}{2\gamma} \right) E(\mathbf{k})$$

With this definition for the filtered energy spectrum it is possible to construct the amplitude of the filtered velocities with a similar relation as in Equation 2.7. The benefit to the Gaussian filter is that the filtering process is revertible. The function  $G(x)$  and its Fourier transform  $\hat{G}(k)$  tend to go to zero for large positive or negative values for  $x$  and  $k$ . However, they never reach zero so a cut-off is necessary for numerical implementation. This method uses an additional step in the post processing to translate  $\bar{E}$  tot  $E$ . In addition to that may the Gaussian filter need different values for  $\gamma$  for regular grids with different lengths.

For this research not a Gaussian filter is used but the sharp cut-off spectral filter. This filter gives an identical spectrum for the lower wavenumbers until the cut-off length  $\bar{k}$  after which the energy is zero for all wavenumbers. The benefit to this filter is that it has local support in the frequency domain where it has the same spectrum as the fine simulation. The only parameters of the sharp cut-off filter are the longest wavenumbers in each direction. The downside to this filter is that the homogeneous isotropic turbulence property might be influenced. This follows from the fact that some wavenumbers are only present in some directions, for example the wavenumbers corresponding to the longest diagonal. These wavenumbers get energy distributed by the method for the construction of the initial field, while other wavenumbers with the same length do not get any energy. This means that the overall behavior of the flow might be dominant in the direction of the diagonals. Another disadvantage to this filter is that it is not local in the spatial domain.

All the energy spectra discussed in this section are represented in Figure 4.1. The filtered spectrum is given for  $\bar{E}(k)$  and not  $E(k)$ .

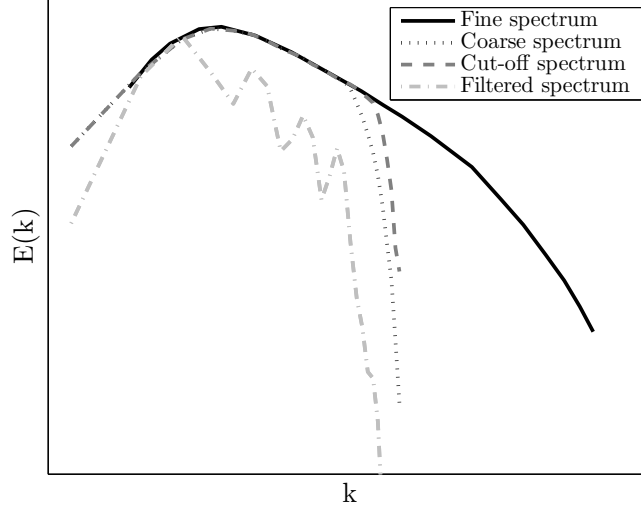


Figure 4.1: Different spectra for the LES

### 4.3 Choices of length scales

In Section 2.3 it was stated that the Smagorinsky model has the variable  $\Delta$  which depends on the dimensions of the grid cells. For a uniform grid this variable is equal to the length of a grid cell. In this research different functions for  $\Delta$  are tested for different values of  $dx$ ,  $dy$  and  $dz$ . All of these functions should satisfy the following conditions. First, the function preserves value when all the variables are the same. This follows from the uniform case for which it is known that  $dx = dy = dz = \Delta$ . The second condition is that the function is invariant under exchange, i.e. changing the order of the variables  $dx, dy$  and  $dz$  does not affect  $\Delta$ . This property needs to hold because the flow is isotropic, so rotation of the computational domain cannot have any influence on  $\Delta$ . The last condition is that the function for  $\Delta$  has to be homogeneous of the first order. This means that if  $dx$ ,  $dy$  and  $dz$  are scaled by the same scaler than  $\Delta$  also scales by the value. The following six functions satisfy these conditions and are tested in this research.

#### Minimum

The Smagorinsky model is an eddy-viscosity model which simulates the smaller scales in the spectrum by adding additional dissipation to the coarse simulation. If the contribution from the model is not enough the result will be a simulation where the energy is transported to the higher wavenumbers at a larger rate than it could dissipate, which can be seen in the spectrum as an accumulation of energy at the higher wavenumbers. Associated to this length scale is the following function for  $\Delta$

$$\Delta_{\min} = \min(dx, dy, dz)$$

This function will be the lower bound for the search space of functions tested in this research. It is assumed that when the ratio between the dimensions of the cells varies significantly the value for  $\Delta$  will result in energy excess at the higher wavenumbers.



## Maximum

The maximum function will be the upper bound to the search space, similar to the previous function for the lower bound. The result of taking a function for  $\Delta$  which is too large will be that the eddy-viscosity model will add too much dissipation. This can be seen in the spectrum as a function which decays too fast in respect to the fine simulation. The maximum function will be the upper bound to the search space, similar to the previous function for the lower bound. The result of taking a function for  $\Delta$  which is too large will be that the eddy-viscosity model will add too much dissipation. This can be seen in the spectrum as a function which decays too fast in respect to the fine simulation.

$$\Delta_{\max} = \max(dx, dy, dz)$$

## Arithmetic mean

Commonly known as mean is a common choice for any kind of research in any field. The benefit of this and the following function over the previous two is that they depend on all three dimensions. The variables  $dx$ ,  $dy$  and  $dz$  have a linear relation in this function for  $\Delta$ .

$$\Delta_{\text{AM}} = \frac{dx + dy + dz}{3}$$

## Geometric mean

The geometric mean is related to the volume of a grid cell. When the volume is enlarged by a value then  $\Delta$  is enlarged by the cube root of that value.

$$\Delta_{\text{GM}} = \sqrt[3]{dx \, dy \, dz}$$

## Harmonic mean

With the previous two means, the harmonic mean forms the three classical Pythagoras means. It is commonly used when the ratios between values are of interest.

$$\Delta_{\text{HM}} = \frac{3}{\frac{1}{dx} + \frac{1}{dy} + \frac{1}{dz}}$$

## Root mean square

The root mean square is also used for the reference velocity. Beside that it is also directly proportional to the longest diagonal of the grid cell.

$$\Delta_{\text{RMS}} = \sqrt{\frac{dx^2 + dy^2 + dz^2}{3}}$$

All of these functions fall under the generalized mean function given by  $M_p = \left( \frac{1}{n} \sum_{i=1}^n x_i^p \right)^{\frac{1}{p}}$ . The minimum and maximum are given by  $p = -\infty$  and  $p = \infty$  respectively and the geometric mean follows when  $p \rightarrow 0$ . All of these functions satisfy the mean inequality which states that  $M_p \leq M_q$  if and only if  $p \leq q$  and the equality only holds when the means are taken over a uniform set. It yields that that  $\Delta_{\text{Min}} \leq \Delta_{\text{HM}} \leq \Delta_{\text{GM}} \leq \Delta_{\text{AM}} \leq \Delta_{\text{RMS}} \leq \Delta_{\text{max}}$ . From these inequalities and the data a value for  $p$  can be derived if the optimal function for  $\Delta$  is in the family of the generalized mean function.

## 4.4 The Reynolds number

The Reynolds number is the dimensionless characteristic of a flow which indicates the effect of the inertial and viscous forces. Different flows with a similar Reynolds number will have similar characteristic. The Reynolds number can be computed by

$$Re = \frac{U_{ref} L_{ref}}{\nu} \quad (4.1)$$

where  $L_{ref}$  is the reference length scale,  $U_{ref}$  is the reference velocity and  $\nu$  the kinematic viscosity. The kinematic viscosity depends on the kind of fluid, the temperature and its density. In this research the temperature fluctuation is negated and it is assumed that the flow is incompressible, which means that the kinematic viscosity is constant over the whole domain. The kinematic viscosity is set to  $\nu = 1.5 \cdot 10^{-5}$ , which is based on the settings in the research in [12]. The reference length  $L_{ref}$  depends on the kind of domain and simulation. For this research  $L_{ref} = 1$ , which corresponds to the lengths of the uniform domain. The simulations with varying domain lengths use the same reference length to not lose the effect of the scales corresponding to the smallest domain length by the force of the scales of the largest domain lengths.

A different Reynolds number could be used for the simulations on the fine grid and the grid of the LES. This is because the reference velocity is different. The reference velocity is equal to the root mean square velocity.

$$U_{ref} = \sqrt{\sum_{x \in \Omega} \frac{u(x)^2 + v(x)^2 + w(x)^2}{3N_x N_y N_z}}$$

From the earlier mentioned Parseval theorem it follows that the reference velocity is equal to the area under the spectrum function. The LES are constructed such that they have the same spectrum as the fine solution for the lower wavenumbers, but is cut-off in the frequency space. The reference velocity of the fine solution is at least higher than that of the LES. But it is decided to use the same Reynolds number for the fine simulation and for the LES. This is because the missing part of the spectrum is modeled by the subgrid-scale model. The simulations are done with  $Re = 10129.18133$ .

## 4.5 Finding the correct Smagorinsky constant

Section 2.3 stated that the Smagorinsky constant is theoretical derived to be around the  $C_S = 0.17$  for homogeneous isotropic turbulence. The simulations in this research aim to simulate homogeneous isotropic turbulence, but since numerical errors and simulation errors could be made it is decided to test different values for  $C_S$ . Another reason for a different choice of the Smagorinsky constant follows from the DNS data, which is used in this research as an exact solution. The resolution for the fine simulation in this research could be too low to be an exact solution. Taking a suitable  $C_S$  could change the LES to better fit the fine simulation instead of an real exact solution. This way the value for  $\Delta$  is not influenced by the error between the fine and a exact solution. The value for  $C_S$  is derived from the uniform grid on the uniform domain, because the optimal value for  $\Delta$  is known for LES on this grid. The domain used for the investigation of  $C_S$  is  $\Omega = [0, 1]^3$ , with  $N_x = N_y = N_z = 256$  for the fine simulation and  $N_x = N_y = N_z = 64$  for the LES. The results can be seen in Figure 4.2. Testing the Smagorinsky model with  $C_S = 0.5$  resulted in a spectrum with too much dissipation in comparison to the DNS data. For  $C_S = 0$  the LES is equivalent to a DNS with  $N_x = N_y = N_z = 16$ . When the choice of optimal  $C_S$  is based on the total error than the best optimum would be a  $C_S$

around  $C_S = 0.16$ . The problem with this choice is that the error made is only compensated at the higher wavenumbers by adding a small spike in the energy spectrum. The spectrum for  $C_S = 0.17$  does have less energy than the DNS and a higher total error than some other choice for  $C_S$ . However, it represents the shape of the spectrum from Section 2.6.2 more accurate. This is an characteristic property for homogeneous isotropic turbulence simulation so it is decided to keep  $C_S = 0.17$  for the Smagorinsky constant.

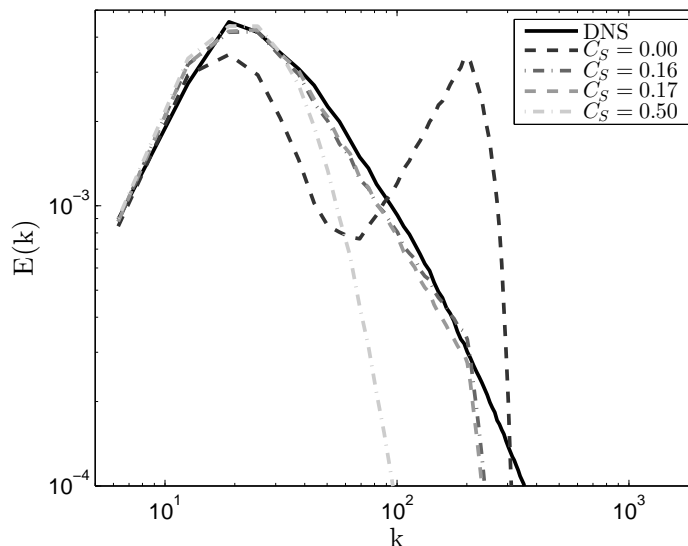


Figure 4.2: Results of simulations with different values of  $C_s$  at  $t = 1.59432 \cdot 10^{-1}$

## 4.6 Program specific choices

The program for the computation of the initial field on a regular grid is written specific for this research. The source code can be found in Appendix C. Most algorithms used are described in Section 3.1 except the fast Fourier inverse for a staggered grid. The relevant input parameters of this code are the dimensions of the spatial domain, the number of grid cells in each direction and a function for the shape of the spectrum. The spectrum used in this research is given in Appendix A. The output of the program is a data file with a three-dimensional solenoidal velocity field with the desired spectrum. In addition, the pressure is computed from the velocity field based on a fourth-order discretisation. The data file is compatible with the in-house code of the Computational Mechanics and Numerical Mathematics department of the University of Groningen. This code offers different methods to compute the evolution of a velocity field over time. One of these methods is using the Smagorinsky model, but with  $\Delta$  fixed to the mesh width  $dx$ . Multiple versions of this program were made, each with one of the functions for  $\Delta$  as described in Section 4.3 and one without any model.

There are multiple parameters which are the same for all versions of the program. The first one is the Reynolds number which is chosen as in Section 4.4. The next three are  $nt$ ,  $dt$  and  $nse$ , which are the number of time steps, the time step width and the number of steps after which the velocity field is stored respectively. The choice of the Reynolds number and  $dt$  are important for the stability of the simulation. After testing it was found that the simulation is

stable for  $dt = 1.59432 \cdot 10^{-4}$ . The number of time steps is  $nt = 4000$ . For the computation of the spectrum it is necessary to compute the Fourier transform of the velocity field, which is often a costly operation. That is why it was chosen to implement the Fast Fourier transform (FFT), which uses a recursive algorithm. The in-house code is written in Fortran and since recursive programs are not supported in older versions of Fortran, the velocity field is exported to Matlab. Here the derivation of the spectrum was done as well as other post processing. The storage of the velocity field is a time and memory consuming operation so the number of iterations after which the velocity is stored is set to  $nse = 1000$ .

The simulation computes from the previous velocity field the next velocity field by using it corresponding models at every iteration. From the new velocity the pressure is computed by using a Poisson solver as described in Section 3.2. The tolerance for the Poisson solver is kept at its original value at  $tol = 10^{-14}$ . The next step is to check if the field is still divergence free, which could have changed due to numerical precision and rounding errors. If this is not the case then the code automatically tries to fix this or terminates if the error is too big. After it is checked whether the results are still stable the process is repeated.

All the spatial discretisation used are of fourth-order and especially designed for staggered grids. The boundary condition are all set to be periodic so  $\mathbf{u}(x_1, y, z) = \mathbf{u}(x_n, y, z)$ ,  $\mathbf{u}(x, y_1, z) = \mathbf{u}(x, y_n, z)$  and  $\mathbf{u}(x, y, z_1) = \mathbf{u}(x, y, z_n)$  for all  $x, y$  and  $z$ .

## 5 | Results

Different kind of test cases were conducted for the investigation of the optimal value for  $\Delta$ . The first tests were done on an uniform domain but with a different number of grid cells to create a nonuniform grid.

### 5.1 Results on the uniform domain

#### 5.1.1 The fine solutions

The choice of  $N_x, N_y$  and  $N_z$  does not effect the kind of fluid which the simulations approximates, but only the accuracy of the approximation. This means that taking a different grid gives approximately the same results as long as the grid is fine enough. This is tested for the fine grid with  $N_x = N_y = N_z = 256$ . Dividing the number of cells in some direction by two or four gives a spectrum which still simulates the same fluid. The resulting spectra from the simulations on these grids can be seen in Figure 5.1. Similar results between Figures 5.1a and 5.1b can be seen, especially for the lower wavenumbers. The difference between both figures is noticeable in the bump in the spectrum of Figure 5.1b where an accumulation of energy happens for the earliest timesteps. The same happens at the same  $k$  in Figure 5.1d, where more energy is accumulated and the effect is also present at the later timesteps. It seems that the origin of this phenomenon lies at  $k = 128\pi$ , which is the largest wavenumber contained in the  $x$  direction. This is equivalent to the smallest length captured in the most coarse direction. In general energy is transported from the larger scales to the smaller scales where the energy dissipates. The smallest scales in the  $x$  direction are too large to dissipate the energy fast enough and there are less options to transport the energy to than for other wavenumbers with the same length but in the  $y$  and  $z$  direction. The result is that the energy is accumulated until it dissipates or can be transported to the higher wavenumbers. The energy in Figure 5.1d can not be dissipated fast enough in both the  $x$  and  $z$  directions, which explains why the excess of is larger than in Figure 5.1b. The excess of energy at a certain wavenumber also happens at  $k = 64\pi$  in the cases where  $N_x = 64$ .

In contrast to the energy excess at the largest wavenumber in a specific direction are the smaller values for  $k$ . The smaller values for  $k$  lose their energy faster on a nonuniform grid than on the uniform grid. This effect can also be seen in DNS with a low resolution, as can be seen in Figure 4.2 for  $C_S = 0$ . This effect follows from the coarsening of the grid, where the lower wavenumbers produce extra dissipation and the higher wavenumbers need more. This last effect can clearly be seen in Figure 5.1f where the energy increases over time at the higher wavenumbers.

The energy from the highest wavenumbers on the uniform grid can not be captured on the nonuniform grid because the largest wavenumber possible on a nonuniform grid scales with the

resolution with  $\sqrt{\frac{N_x^2}{2l_x} + \frac{N_y^2}{2l_y} + \frac{N_z^2}{2l_z}}$ . The energy lost due to the moving cut-off boundary is negligible for most simulations since most of the energy is contained at the lower wavenumbers.

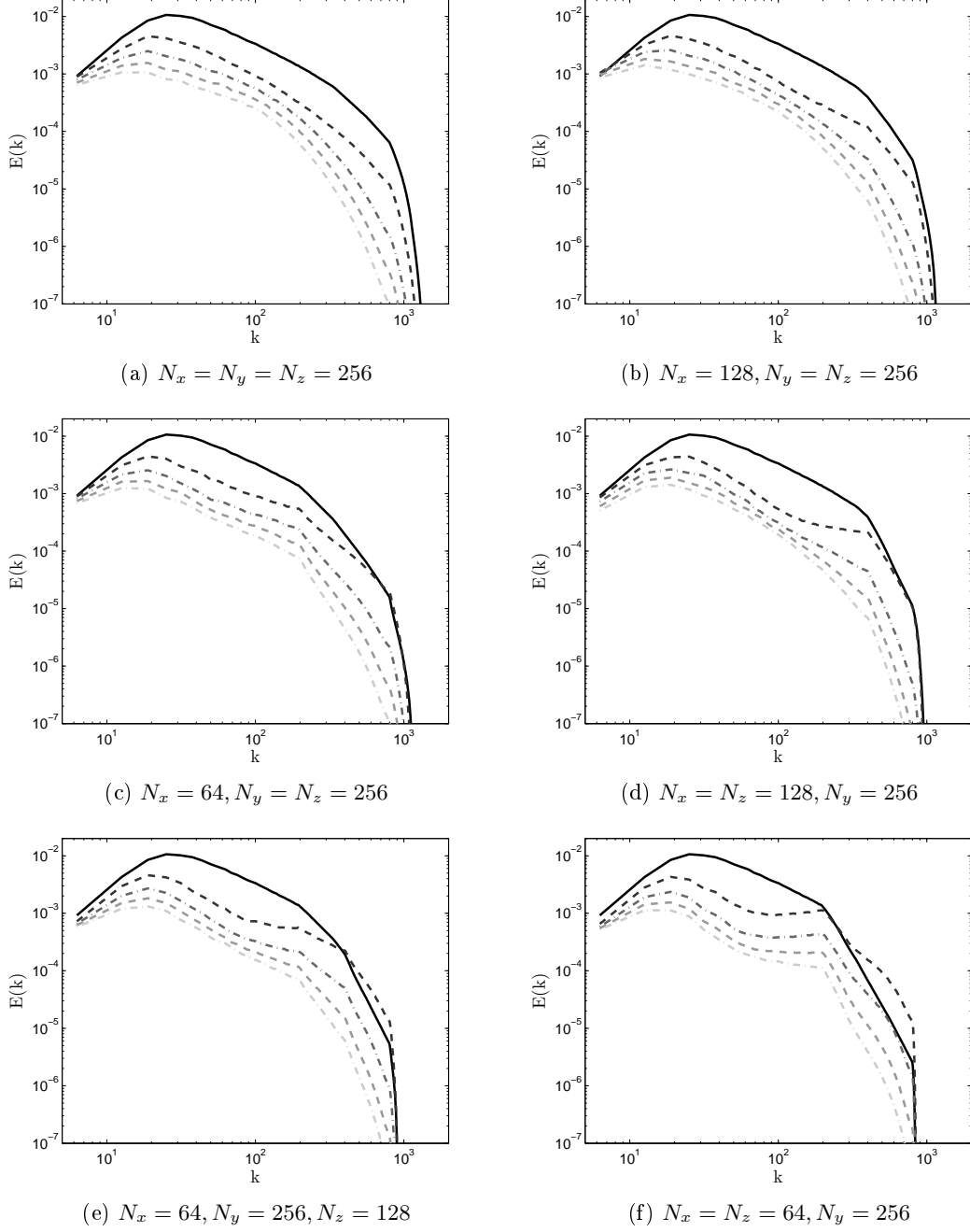


Figure 5.1: The spectrum at different times: initial spectrum (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (---),  $t = 0.63772$  (---) and  $t = 0.79716$  (---)

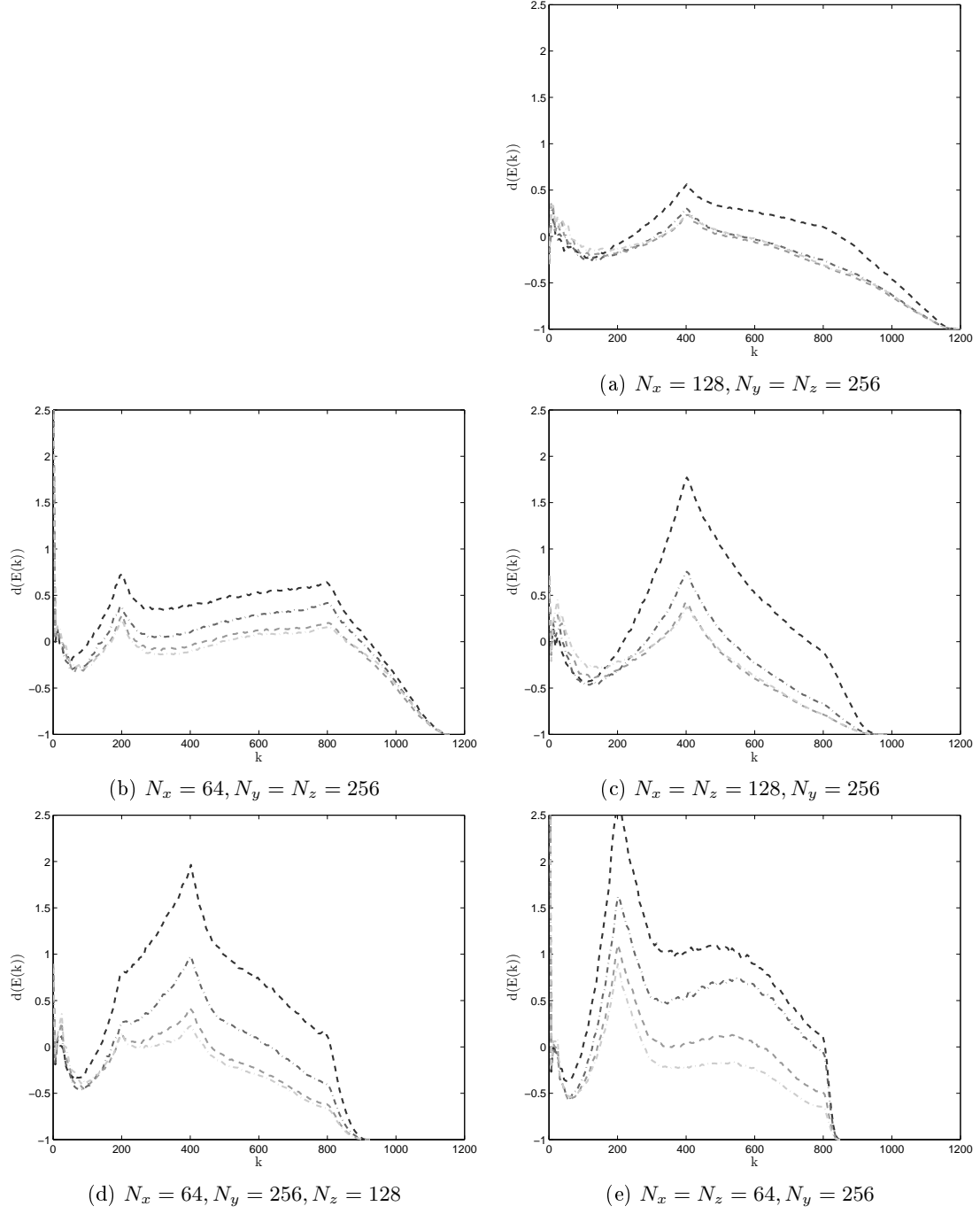


Figure 5.2: The numerical relative differences  $d$  of the spectra compared to the spectrum of the uniform grid:  $t = 0.159432$  (---),  $t = 0.318864$  (-.-),  $t = 0.478296$  (···),  $t = 0.63772$  (- - -) and  $t = 0.79716$  (-·-·-)

### 5.1.2 The optimal solution

The best results of a simulation on a nonuniform grid for the uniform domain tested in this research are from the grid with  $N_x = 32$  and  $N_y = N_z = 64$ . This is because it has the highest resolution of all the tested nonuniform grids. The results of the tested choices for  $\Delta$  on this grid are given in Figure 5.7. From these results it is possible to define an optimal function for  $\Delta$ . However, the choice of the optimal depends on multiple factors. The optimal  $\Delta$  could be the value for which the total error over time and all  $k$  is the smallest, but since all the models are known to have bad results for  $k$  around the cut-off value could it be better to exclude the end of the spectrum. The choice of the largest  $k$  included in the error computation could be smaller if the wavenumbers in the spectrum beyond the energy excess are excluded. This excess was discussed in Section 5.1.1. Besides the choice of  $k$  included in the error computation it is also possible to exclude certain time results. Different time results can be included or excluded if an optimal  $\Delta$  is needed for shorter or longer simulation.

In Figure 5.8 the cumulative relative error for each  $k$  at each  $t$  is determined. From these results can be concluded that the  $\Delta = \max dx, dy, dz$  works the best for short simulations where only the first few wavenumbers are important. In other situations this choice of  $\Delta$  is not optimal and it is suggested to pick one of the other choices. The minimum seems one of the best choices for  $k > 150$ , since it has the lowest total error at any time after this wavenumber. The problem with the choice  $\Delta = \min dx, dy, dz$  is that it has the largest error for the lowest wavenumbers. The other four models for  $\Delta$  are very close to each other in the value for  $\Delta$  and their error. They all perform better than  $\Delta = \min dx, dy, dz$  for the smallest wavenumbers but worse at the higher ones. The opposite holds for the comparison with  $\Delta = \max dx, dy, dz$ .

In this research are all the errors given at  $t = 0.159432$  and  $t = 0.79716$  to show the effects of the models at a instance after a short and long time. This error is taken over all wavenumbers contained on the grid of the LES, which results in Tables 5.3 and 5.4. Besides that also the total error of the energy dissipation rate is given, which is derived with

$$\varepsilon = \sum_{k < \bar{k}} k^2 (E_{fine}(k) - E_{LES}(k))$$

for the cut-off length  $\bar{k}$ . These results can be seen in Tables 5.5 and 5.6. Two things can be concluded from these results. The first thing is that the minimum is the optimal choice for any resolution, under the definition of the earlier stated error computation. This means that only a upper-bound for  $\Delta$  is found, since  $\Delta = \min dx, dy, dz$  was the lowest tested value. A lower value for  $\Delta$  could also have the same effect as lowering the Smagorinsky constant  $C_S$ . The result of lowering the Smagorinsky is that the decrease of energy at the higher wavenumbers weakens, which resulted in a less constant spectrum. That is why it is advised to not use a value for  $\Delta$  which is lower than the minimum, to preserve a smooth spectrum. The second conclusion from the tables is that a higher resolution in one direction does not means that the simulation will be better. The results for grids which are coarse in two directions and fine in one are noticeably better than the results with one additional resolution refinement in one direction.



Function for $\Delta$	Grid resolution ( $[N_x, N_y, N_z]$ )					
	[64,64,64]	[32,64,64]	[16,64,64]	[32,64,32]	[16,64,32]	[16,64,16]
Minimum	0.0077	0.0255	0.0314	0.0155	0.0197	0.0108
Harmonic mean	0.0077	0.0275	0.0353	0.0178	0.0281	0.0226
Geometric mean	0.0077	0.0280	0.0371	0.0187	0.0302	0.0260
Mean	0.0077	0.0286	0.0387	0.0194	0.0319	0.0281
RMS	0.0077	0.0291	0.0398	0.0199	0.0331	0.0292
Maximum	0.0077	0.0313	0.0415	0.0216	0.0362	0.0311

Figure 5.3:  $Err_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Grid resolution ( $[N_x, N_y, N_z]$ )					
	[64,64,64]	[32,64,64]	[16,64,64]	[32,64,32]	[16,64,32]	[16,64,16]
Minimum	0.0037	0.0048	0.0072	0.0029	0.0037	0.0040
Harmonic mean	0.0037	0.0049	0.0075	0.0032	0.0055	0.0047
Geometric mean	0.0037	0.0049	0.0076	0.0035	0.0059	0.0047
Mean	0.0037	0.0049	0.0078	0.0037	0.0063	0.0049
RMS	0.0037	0.0049	0.0081	0.0038	0.0065	0.0052
Maximum	0.0037	0.0055	0.0087	0.0044	0.0074	0.0059

Figure 5.4:  $Err_{t=0.79716}$  of each model at each grid

Function for $\Delta$	Grid resolution ( $[N_x, N_y, N_z]$ )					
	[64,64,64]	[32,64,64]	[16,64,64]	[32,64,32]	[16,64,32]	[16,64,16]
Minimum	2.6580	7.5438	8.8746	3.8086	4.4471	2.2370
Harmonic mean	2.6580	8.1316	9.9896	4.3665	6.3525	4.6864
Geometric mean	2.6580	8.2803	10.4790	4.5749	6.8293	5.3822
Mean	2.6580	8.4387	10.9532	4.7429	7.2232	5.8207
RMS	2.6580	8.5885	11.2531	4.8690	7.4963	6.0492
Maximum	2.6580	9.2533	11.7262	5.3019	8.1856	6.4420

Figure 5.5:  $\varepsilon_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Grid resolution ( $[N_x, N_y, N_z]$ )					
	[64,64,64]	[32,64,64]	[16,64,64]	[32,64,32]	[16,64,32]	[16,64,16]
Minimum	1.2933	1.4256	2.0299	0.7146	0.8375	0.8347
Harmonic mean	1.2933	1.4344	2.1137	0.7941	1.2535	0.9758
Geometric mean	1.2933	1.4353	2.1575	0.8488	1.3375	0.9804
Mean	1.2933	1.4380	2.2187	0.8970	1.4163	1.0100
RMS	1.2933	1.4438	2.2799	0.9353	1.4776	1.0739
Maximum	1.2933	1.6389	2.4505	1.0809	1.6654	1.2284

Figure 5.6:  $\varepsilon_{t=0.79716}$  of each model at each grid

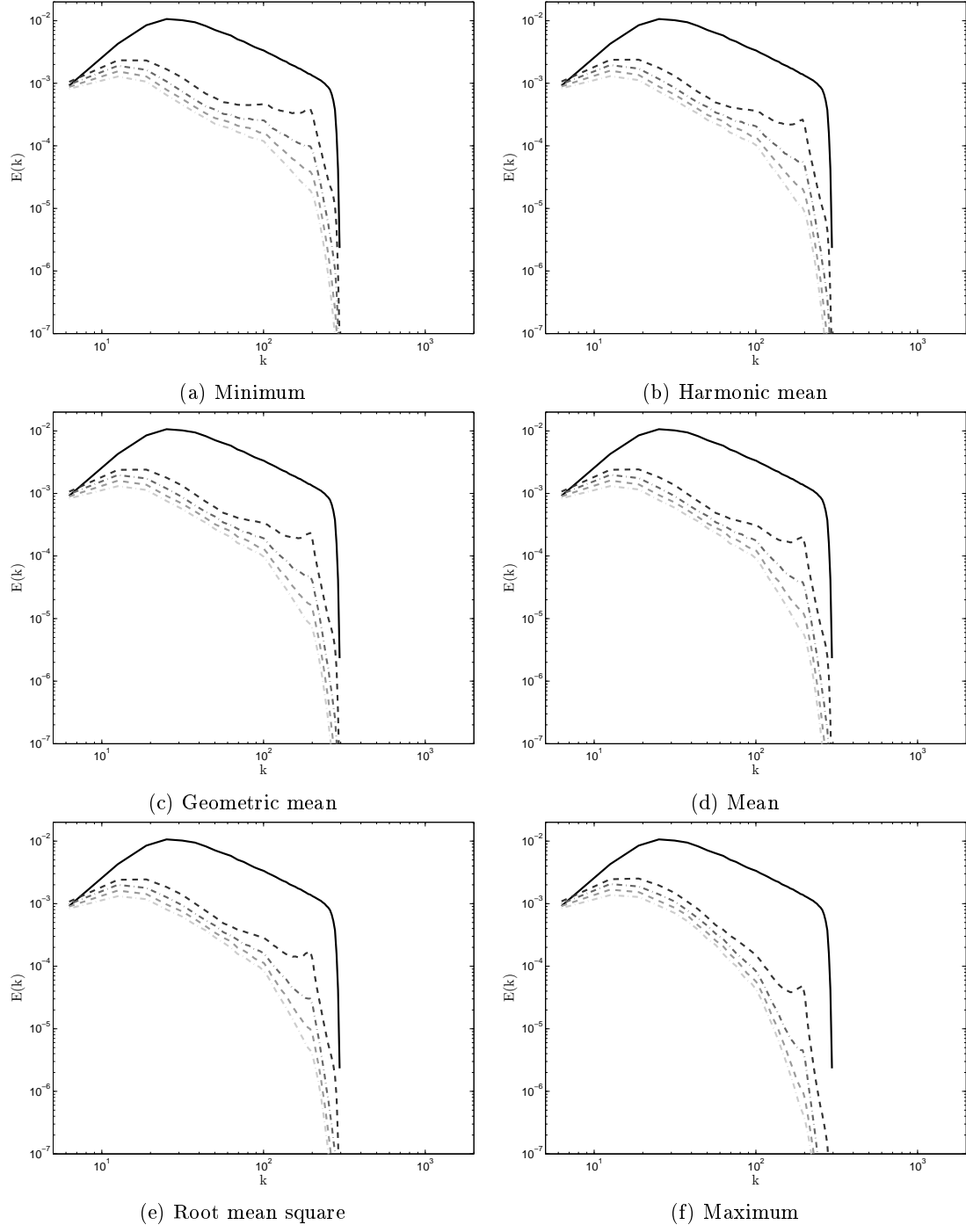


Figure 5.7: The spectrum at different times: initial spectrum (—),  $t = 0.159432$  (---),  $t = 0.318864$  (- - -),  $t = 0.478296$  (- · - ·),  $t = 0.63772$  (- - -) and  $t = 0.79716$  (- · - ·)

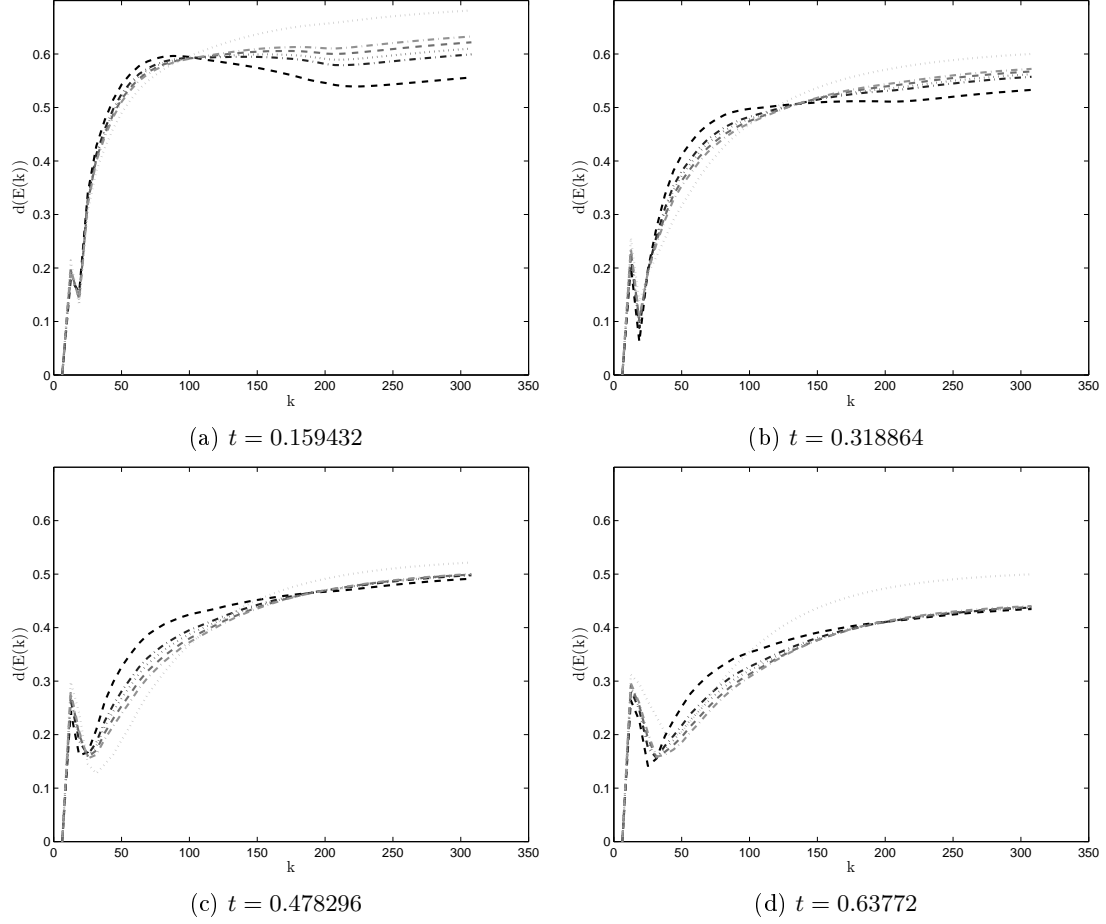


Figure 5.8: The total relative error of all models with a right bound  $k$  for the error computation domain: Minimum (---), Harmonic mean (-.-), Geometric mean (.....), Mean (---), Root mean square (-.-) and Maximum (.....)

## 5.2 Results on the nonuniform domain

The following results are derived on nonuniform grids on nonuniform domains. The number of grid cells for the fine simulations are  $[N_x, N_y, N_z] = [256, 256, 256]$  and for the LES  $[64, 64, 64]$ . The varying of the length of the domain can be divided in three cases. The first case starts on a uniform domain and increases the length of one side of the domain for each new simulation, the second case starts on the same uniform domain but increases two sides by one for each new simulation and the last case starts from the last domain from the first case and changes one side by one for each simulation such that it will become the same domain as the last simulation in the second case. All cases contain ten different domains.

### 5.2.1 Case 1: changing one length

The simulations on the uniform domain all approximated the same flow, which is not the case for the nonuniform domain. One of the reasons why each flow on each domain is different is because the reference length could vary. This means that each flow has a different Reynolds number to have the same behavior, but this is not done for this research. This is because there are methods to derive the best value for  $L_{ref}$ , but the best value is found a posteriori and that is not possible since the optimal value for  $\Delta$  is not known either.

Before the spectra of the LES are researched should the spectra of the fine simulation be understood. Two of the spectra of the fine solution from case 1 can be seen in Figure 5.9. There are two clear differences between the two figures and the uniform case in Figure 5.1a. These differences are that when one length of the domain increases the energy at the lower wavenumbers decreases slower and the energy at the higher wavenumbers decreases faster. This was not the case for the simulations on the uniform domain and could be because of two different reasons. The first is that for the large stretched domains there are more wavenumbers captured by the grid in the range of the smaller wavenumbers. In these additional wavenumbers more energy could be stored which explains the slower decrease. Similarly are there less wavenumbers captured at the higher wavenumbers so the energy has to dissipate faster. The other reason could be because of the use of the same Reynolds number on every domain, which results in a flow being simulated that has more dissipation than the flow on the uniform domain.

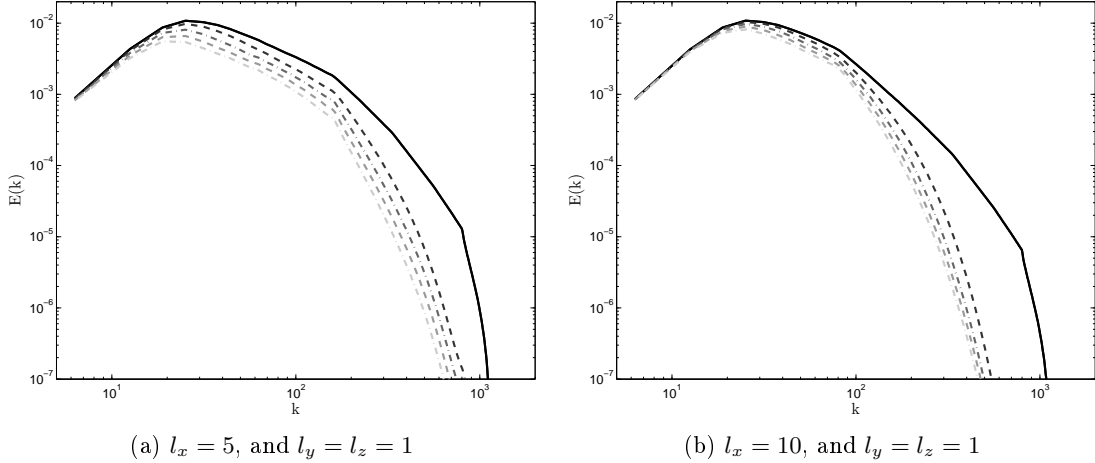


Figure 5.9: The energy spectra of the fine simulations where only one direction is changed. Initial spectrum (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (---),  $t = 0.63772$  (---) and  $t = 0.79716$  (---)

In Tables 5.10 and 5.11 are all the LES results from the first case. Similarly to the conclusion of the uniform grid can be seen that the minimum is the optimal choice for  $\Delta$ . The overall trend seems that the error increases when the size of the domain increases, which partially follows from the larger discretisation error. There are some exceptions like the domain with  $l_x = 8$ , where the total error is significantly less for all models in comparison to the domains with  $l_x = 7$  and  $l_x = 9$ . The spectrum of the optimal solution can be seen in Figure 5.14.

Function for $\Delta$	Length of $l_x$				
	1	2	3	4	5
Minimum	0.0074	0.0079	0.0109	0.0102	0.0141
Harmonic mean	0.0074	0.0116	0.0158	0.0174	0.0200
Geometric mean	0.0074	0.0131	0.0192	0.0217	0.0256
Mean	0.0074	0.0142	0.0231	0.0291	0.0348
RMS	0.0074	0.0155	0.0272	0.0358	0.0424
Maximum	0.0074	0.0253	0.0410	0.0513	0.0578

Function for $\Delta$	Length of $l_x$				
	6	7	8	9	10
Minimum	0.0127	0.0241	0.0156	0.0308	0.0263
Harmonic mean	0.0186	0.0276	0.0199	0.0332	0.0288
Geometric mean	0.0252	0.0330	0.0266	0.0376	0.0336
Mean	0.0367	0.0430	0.0400	0.0477	0.0454
RMS	0.0458	0.0511	0.0500	0.0556	0.0542
Maximum	0.0616	0.0648	0.0651	0.0677	0.0670

Figure 5.10:  $Err_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Length of $l_x$				
	1	2	3	4	5
Minimum	0.0035	0.0044	0.0046	0.0083	0.0112
Harmonic mean	0.0035	0.0049	0.0067	0.0114	0.0148
Geometric mean	0.0035	0.0054	0.0083	0.0129	0.0167
Mean	0.0035	0.0056	0.0085	0.0149	0.0190
RMS	0.0035	0.0060	0.0096	0.0154	0.0213
Maximum	0.0035	0.0090	0.0151	0.0223	0.0298

Function for $\Delta$	Length of $l_x$				
	6	7	8	9	10
Minimum	0.0122	0.0152	0.0129	0.0181	0.0162
Harmonic mean	0.0157	0.0177	0.0157	0.0206	0.0188
Geometric mean	0.0185	0.0212	0.0196	0.0243	0.0240
Mean	0.0224	0.0262	0.0278	0.0328	0.0342
RMS	0.0261	0.0314	0.0350	0.0393	0.0413
Maximum	0.0363	0.0413	0.0452	0.0484	0.0506

Figure 5.11:  $Err_{t=0.79716}$  of each model at each grid

Function for $\Delta$	Length of $l_x$				
	1	2	3	4	5
Minimum	1.2067	1.4245	1.9300	1.6860	1.9635
Harmonic mean	1.2067	2.0152	2.7170	2.6614	2.7044
Geometric mean	1.2067	2.2113	3.1202	3.2079	3.3203
Mean	1.2067	2.3945	3.5847	3.9547	4.1634
RMS	1.2067	2.5986	4.0010	4.5309	4.7625
Maximum	1.2067	3.6034	5.0756	5.5782	5.6926

Function for $\Delta$	Length of $l_x$				
	6	7	8	9	10
Minimum	1.6035	2.4102	1.5722	2.5279	2.0892
Harmonic mean	2.3066	2.8165	2.0247	2.7720	2.3264
Geometric mean	3.0111	3.3401	2.6703	3.1772	2.7649
Mean	4.0282	4.1876	3.7628	3.9728	3.6686
RMS	4.6957	4.7442	4.4220	4.4716	4.2090
Maximum	5.5922	5.4666	5.1977	5.0653	4.8174

Figure 5.12:  $\varepsilon_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Length of $l_x$				
	1	2	3	4	5
Minimum	0.3532	0.4178	0.5651	0.8635	1.0599
Harmonic mean	0.3532	0.5596	0.8600	1.2445	1.4732
Geometric mean	0.3532	0.6077	0.9940	1.4407	1.7173
Mean	0.3532	0.6595	1.0881	1.6458	1.9783
RMS	0.3532	0.7043	1.2041	1.7602	2.1717
Maximum	0.3532	0.9661	1.5838	2.1627	2.5779

Function for $\Delta$	Length of $l_x$				
	6	7	8	9	10
Minimum	1.0953	1.3272	1.0928	1.4462	1.2624
Harmonic mean	1.4985	1.6226	1.4103	1.6670	1.4825
Geometric mean	1.8128	1.9343	1.7818	1.9642	1.8421
Mean	2.1758	2.3317	2.3272	2.4537	2.4076
RMS	2.4158	2.6000	2.6529	2.7301	2.7024
Maximum	2.8301	2.9693	3.0235	3.0411	3.0124

Figure 5.13:  $\varepsilon_{t=0.79716}$  of each model at each grid

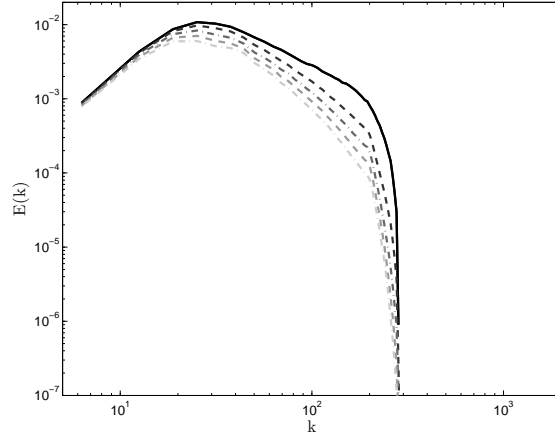


Figure 5.14: Spectrum of LES with  $\Delta = \min(dx, dy, dz)$

### 5.2.2 Case 2: changing two lengths

Changing two sides at the same rate enhances the effect seen in case 1 where the decrease of energy is slower for the lower wavenumbers than for the higher wavenumbers. Besides that the cut-off wavenumber is lower for higher values of  $l_x$  and  $l_z$ . This means that even the DNS will have problems with simulating the actual flow and so is a higher resolution necessary. The results can be seen in Tables 5.16 and 5.17. While a bigger domain in general leads to a bigger discretisation error, does it not mean that the total error increases for each increase in  $l_x$  and  $l_y$ . This follows from the fact that a larger mesh width increases the accuracy of the higher wavenumbers and since these wavenumbers have the most energy, it follows that stretching the domain gives a more accurate simulation. Stretching the domain too much could mean that the error at the lower wavenumbers affects the total accuracy which was gained by adding more wavenumbers to the lower frequency range.

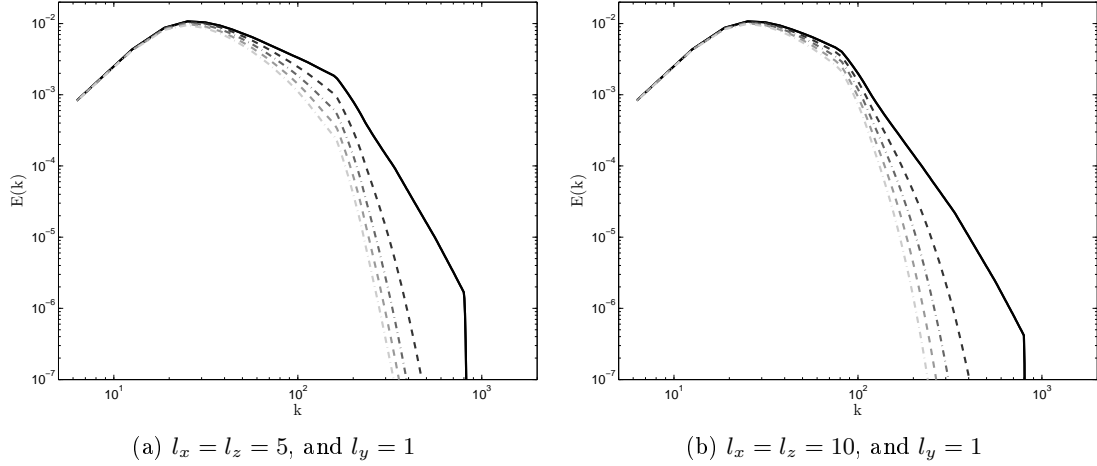


Figure 5.15: The energy spectra of the fine simulations where two direction are changed. Initial spectrum (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (---),  $t = 0.63772$  (---) and  $t = 0.79716$  (---)

Function for $\Delta$	Lengths of $l_x$ and $l_z$				
	1	2	3	4	5
Minimum	0.0074	0.0085	0.0064	0.0046	0.0169
Harmonic mean	0.0074	0.0134	0.0137	0.0116	0.0205
Geometric mean	0.0074	0.0153	0.0176	0.0168	0.0242
Mean	0.0074	0.0171	0.0211	0.0215	0.0280
RMS	0.0074	0.0183	0.0234	0.0246	0.0304
Maximum	0.0074	0.0234	0.0293	0.0308	0.0350

Function for $\Delta$	Lengths of $l_x$ and $l_z$				
	6	7	8	9	10
Minimum	0.0161	0.0279	0.0212	0.0357	0.0383
Harmonic mean	0.0187	0.0291	0.0223	0.0362	0.0386
Geometric mean	0.0223	0.0312	0.0245	0.0373	0.0394
Mean	0.0264	0.0339	0.0277	0.0390	0.0408
RMS	0.0289	0.0356	0.0296	0.0401	0.0416
Maximum	0.0334	0.0385	0.0328	0.0419	0.0430

Figure 5.16:  $Err_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Lengths of $l_x$ and $l_z$				
	1	2	3	4	5
Minimum	0.0033	0.0070	0.0083	0.0066	0.0083
Harmonic mean	0.0033	0.0081	0.0084	0.0090	0.0115
Geometric mean	0.0033	0.0089	0.0107	0.0128	0.0158
Mean	0.0033	0.0100	0.0135	0.0169	0.0202
RMS	0.0033	0.0103	0.0153	0.0194	0.0228
Maximum	0.0033	0.0128	0.0196	0.0242	0.0273

Function for $\Delta$	Lengths of $l_x$ and $l_z$				
	6	7	8	9	10
Minimum	0.0084	0.0156	0.0126	0.0237	0.0268
Harmonic mean	0.0116	0.0177	0.0146	0.0248	0.0275
Geometric mean	0.0164	0.0212	0.0186	0.0271	0.0294
Mean	0.0215	0.0254	0.0236	0.0303	0.0322
RMS	0.0242	0.0276	0.0262	0.0321	0.0338
Maximum	0.0287	0.0313	0.0302	0.0348	0.0361

Figure 5.17:  $Err_{t=0.79716}$  of each model at each grid



Function for $\Delta$	Length of $l_x$				
	1	2	3	4	5
Minimum	1.2088	1.2582	0.8651	0.6399	1.9114
Harmonic mean	1.2088	1.9557	1.7856	1.4465	2.2659
Geometric mean	1.2088	2.1630	2.1714	1.9581	2.5993
Mean	1.2088	2.3343	2.4973	2.4150	2.9353
RMS	1.2088	2.4684	2.7178	2.6968	3.1389
Maximum	1.2088	2.9586	3.2384	3.2414	3.5086

Function for $\Delta$	Length of $l_x$				
	6	7	8	9	10
Minimum	1.6621	2.4472	1.8259	2.5706	2.5398
Harmonic mean	1.9121	2.5487	1.9127	2.6035	2.5596
Geometric mean	2.2278	2.7149	2.0860	2.6817	2.6146
Mean	2.5768	2.9236	2.3210	2.7997	2.7041
RMS	2.7817	3.0481	2.4581	2.8694	2.7570
Maximum	3.1317	3.2582	2.6828	2.9858	2.8439

Figure 5.18:  $\varepsilon_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Length of $l_x$				
	1	2	3	4	5
Minimum	0.3444	0.6234	0.6840	0.5173	0.7581
Harmonic mean	0.3444	0.8343	0.9338	0.9283	1.0878
Geometric mean	0.3444	0.9232	1.1063	1.2130	1.3823
Mean	0.3444	1.0058	1.2951	1.4756	1.6454
RMS	0.3444	1.0571	1.4109	1.6249	1.7882
Maximum	0.3444	1.2729	1.6805	1.8939	2.0229

Function for $\Delta$	Length of $l_x$				
	6	7	8	9	10
Minimum	0.7298	1.1960	0.9487	1.5033	1.5771
Harmonic mean	1.0172	1.3499	1.0909	1.5657	1.6176
Geometric mean	1.3442	1.5722	1.3432	1.6999	1.7211
Mean	1.6448	1.8062	1.6253	1.8725	1.8659
RMS	1.7985	1.9273	1.7651	1.9615	1.9409
Maximum	2.0327	2.1093	1.9669	2.0933	2.0511

Figure 5.19:  $\varepsilon_{t=0.79716}$  of each model at each grid

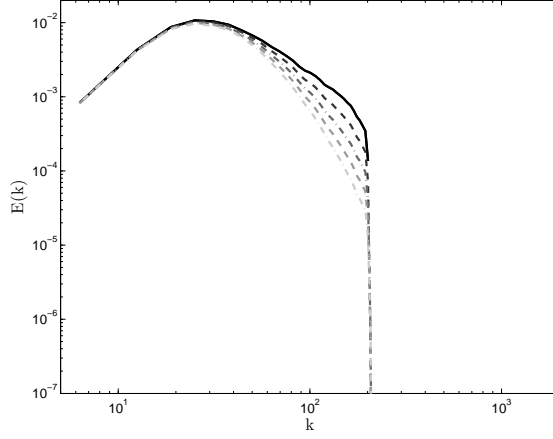


Figure 5.20: Spectrum of LES with  $\Delta = \min(dx, dy, dz)$

### 5.2.3 Case 3: changing one length between the minimum and maximum of the all the lengths of the domain

The domains in this case have  $l_y = 1$  for the shortest length, which is the same as the other cases, and the longest length on every domain is  $l_x = 10$ . The third length  $l_z$  varies between the values of  $l_x$  and  $l_y$ , such that the first and last domain have similarities to the last domains of case 1 and 2 respectively. The results of all simulations are given in Tables 5.22 and 5.23.

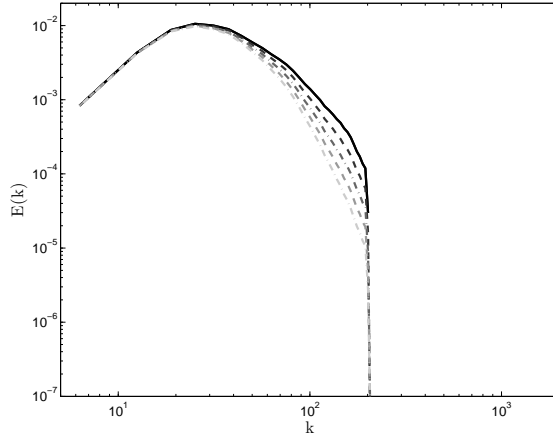


Figure 5.21: The energy spectrum of the fine simulations with  $l_x = 10, l_z = 5$ , and  $l_y = 1$ . Initial spectrum (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (---),  $t = 0.63772$  (---) and  $t = 0.79716$  (---)

Function for $\Delta$	Length of $l_z$				
	1	2	3	4	5
Minimum	0.0264	0.0165	0.0298	0.0255	0.0192
Harmonic mean	0.0288	0.0196	0.0314	0.0270	0.0207
Geometric mean	0.0336	0.0237	0.0335	0.0293	0.0232
Mean	0.0454	0.0324	0.0377	0.0331	0.0271
RMS	0.0543	0.0403	0.0416	0.0365	0.0302
Maximum	0.0670	0.0553	0.0518	0.0460	0.0394

Function for $\Delta$	Length of $l_z$				
	6	7	8	9	10
Minimum	0.0354	0.0510	0.0720	0.0547	0.0383
Harmonic mean	0.0361	0.0513	0.0724	0.0548	0.0386
Geometric mean	0.0374	0.0518	0.0733	0.0552	0.0394
Mean	0.0395	0.0527	0.0748	0.0558	0.0408
RMS	0.0411	0.0533	0.0757	0.0562	0.0416
Maximum	0.0459	0.0552	0.0782	0.0570	0.0430

Figure 5.22:  $Err_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Length of $l_z$				
	1	2	3	4	5
Minimum	0.0163	0.0099	0.0174	0.0142	0.0101
Harmonic mean	0.0190	0.0147	0.0202	0.0170	0.0129
Geometric mean	0.0238	0.0201	0.0238	0.0209	0.0173
Mean	0.0343	0.0295	0.0296	0.0264	0.0229
RMS	0.0413	0.0366	0.0343	0.0304	0.0267
Maximum	0.0506	0.0482	0.0445	0.0404	0.0364

Function for $\Delta$	Length of $l_z$				
	6	7	8	9	10
Minimum	0.0219	0.0341	0.0596	0.0389	0.0268
Harmonic mean	0.0234	0.0347	0.0605	0.0392	0.0275
Geometric mean	0.0261	0.0360	0.0625	0.0401	0.0294
Mean	0.0297	0.0378	0.0653	0.0415	0.0322
RMS	0.0320	0.0390	0.0669	0.0422	0.0338
Maximum	0.0382	0.0422	0.0706	0.0439	0.0361

Figure 5.23:  $Err_{t=0.79716}$  of each model at each grid

Function for $\Delta$	Length of $l_z$				
	1	2	3	4	5
Minimum	2.0897	1.2511	2.3340	2.1199	1.7196
Harmonic mean	2.3268	1.5448	2.4636	2.2442	1.8410
Geometric mean	2.7654	1.9062	2.6389	2.4331	2.0466
Mean	3.6681	2.6100	2.9566	2.7338	2.3519
RMS	4.2096	3.1743	3.2371	2.9778	2.5751
Maximum	4.8170	4.0664	3.8688	3.6077	3.1973

Function for $\Delta$	Length of $l_z$				
	6	7	8	9	10
Minimum	2.7043	3.5357	4.3671	3.4847	2.5396
Harmonic mean	2.7566	3.5525	4.3945	3.4936	2.5596
Geometric mean	2.8585	3.5897	4.4593	3.5169	2.6147
Mean	3.0108	3.6473	4.5587	3.5546	2.7040
RMS	3.1165	3.6855	4.6202	3.5778	2.7567
Maximum	3.4344	3.8007	4.7773	3.6286	2.8438

Figure 5.24:  $\varepsilon_{t=0.159432}$  of each model at each grid

Function for $\Delta$	Length of $l_z$				
	1	2	3	4	5
Minimum	1.2666	0.6481	1.1220	0.9765	0.7902
Harmonic mean	1.4900	1.0026	1.3117	1.1620	0.9760
Geometric mean	1.8382	1.3553	1.5292	1.4003	1.2452
Mean	2.4078	1.8912	1.8472	1.7074	1.5678
RMS	2.7031	2.2401	2.0785	1.9143	1.7651
Maximum	3.0118	2.7099	2.5144	2.3554	2.2137

Function for $\Delta$	Length of $l_z$				
	6	7	8	9	10
Minimum	1.4463	2.0586	3.2230	2.1845	1.5769
Harmonic mean	1.5369	2.0915	3.2758	2.2029	1.6176
Geometric mean	1.6929	2.1584	3.3887	2.2484	1.7212
Mean	1.8880	2.2492	3.5367	2.3140	1.8658
RMS	2.0040	2.3030	3.6166	2.3505	1.9404
Maximum	2.2927	2.4433	3.7919	2.4224	2.0511

Figure 5.25:  $\varepsilon_{t=0.79716}$  of each model at each grid

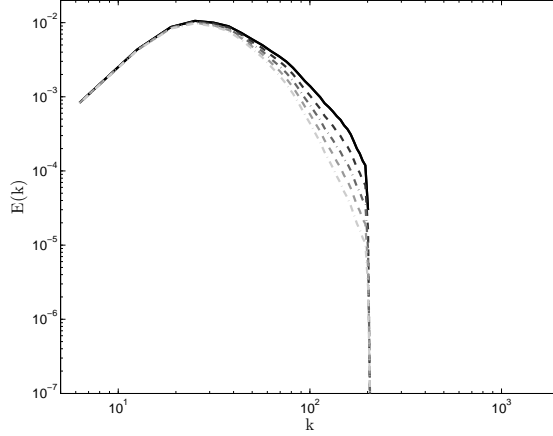


Figure 5.26: Spectrum of LES with  $\Delta = \min(dx, dy, dz)$

#### 5.2.4 Combined results

The total error of all models with the minimum are given in Figure 5.27. The most notable effect can be seen in all cases for  $l = 8$ , here cases 1 and 2 have a local minimum and case 3 has a local maximum. For cases 1 and 2 the result can be explained by the shape of the used data from the CBC experiment, which has a peak at the wavenumber corresponding to the length  $l = 8$ . The difference between the error for  $l = 7$  and  $l = 8$  follows from a better representation of this peak. The domains in Case 3 captures this peak from the CBC data also for  $l = 7$ , which explains why there is no local minimum for Case 3 at  $l = 8$ . The local maximum at  $l = 8$  is the result of the similar effect of the energy excess at certain wavenumbers in the uniform domain. For case 3 and  $l = 8$  is there one direction which contains the energy of the peak in Figure 5.27 but cannot transport it to the lower wavenumbers. This explains the large error for all choices of  $\Delta$ .

Most of the results of all the cases have that the simulations with an even value for  $l$  perform better than the simulations with  $l - 1$ , while the discretisation error is bigger. This could follow from a better approximation of the peak in the CBC data but this is not clear.

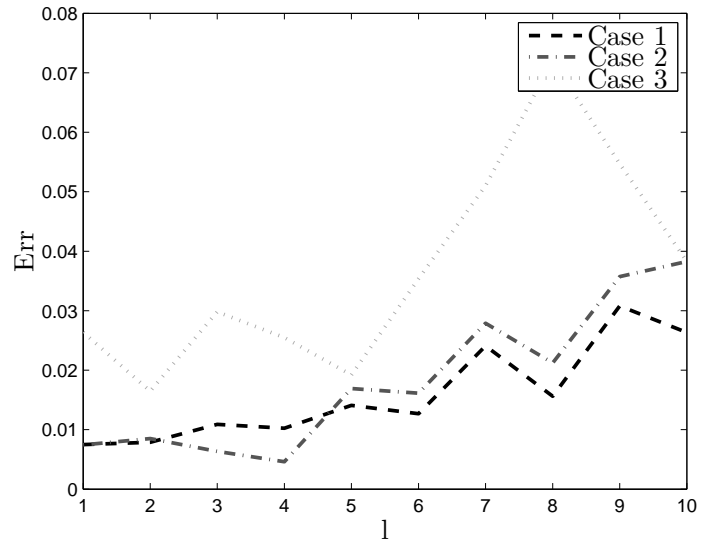


Figure 5.27: The error on each domain with the minimum function for  $\Delta$

## 6 | Conclusion

### 6.1 Conclusion

The results of every simulation in this research showed that the optimal value for the length scale in the Smagorinsky model is given by  $\Delta = \min dx, dy, dz$ . This was tested on two sets of regular grids, one set which consists of regular grids on an uniform domain and the other set of grids with a constant resolution of different stretched domains. Lower values for  $\Delta$  were tested by lowering the Smagorinsky constant, this resulted in an irregular spectrum around the cut-off length.

The error of the tested models is influenced by multiple factors. The first factor was noted in the simulations on the uniform domain which showed that the lowest component of the resolution of the grid resulted in energy excess at fixed wavenumbers. This excess was inflated by models which added too much dissipation, which explained the good results of the minimum function. If the effect of the resolution on the spectrum only occurs at high wavenumbers close to the cut-off length than higher order average models could be used, like the harmonic mean or the geometric mean functions. However, the minimum function is advised for all other cases. Another effect on the error of the models was the used spectrum. In this research the spectra are based on the CBC data. While this influenced the results on the nonuniform domain, this has no further influence on the conclusion of this research.

The Smagorinsky model does have problems for simulations on nonuniform grids. This mainly follows from solving a three dimensional problem with a one dimensional parameter. Problems can occur in the direction, which needed a change of  $\Delta$ , but this also influences the other directions.

The conclusion of this research is that the minimum function is the best choice for the Smagorinsky model on a nonuniform grids. However, the Smagorinsky model with this  $\Delta$  is not the best method for simulating a flow on a nonuniform grid. There are better models available which are more suitable to use on a nonuniform grid. A good alternative is the Dynamic Smagorinsky, explained in [6]. This model automatically searches the correct value product of  $C_S$  and  $\Delta$ .

### 6.2 Discussion

#### 6.2.1 Choice of constants

Some assumptions were made for the simulations on the nonuniform domains. The first one is that the reference length was the same for all simulation. This was justified by taking the reference length in the  $y$  direction such that  $L_{ref} = l_y = 1$  for all simulations. Similarly was the Smagorinsky constant kept at  $C_S = 0.17$  for all LES on every grid and domain. These choices

were based on the aim to make the resulting spectra for every domain similar, but this did not work. The spectrum of the fine simulation for  $l_x = 2$  and  $l_y = l_z$  did still have similarities to the uniform domain, although this was no longer the case when the domain was stretched much further. It can be discussed that the simulation no longer simulates homogeneous isotropic turbulence for these stretched domains. Therefore a different Smagorinsky constant can give better results since  $C_S = 0.17$  was specific derived for homogeneous isotropic turbulence. This can be tested by using the found optimal value for  $\Delta$ . The new value for  $C_S$  can then be used to test the optimal value for  $\Delta$ . Repeating this process could lead to a better optimal value for both  $\Delta$  and  $C_S$ .

### 6.2.2 Accuracy

For this research  $N = 256$  was used for the DNS results, while other current researches use  $N = 512$  or  $N = 1024$ . This resolution was not feasible with the available resources. The result of a higher accuracy could be a steeper spectrum in the inertial range, because this effect was observed when the accuracy for the DNS in this research was set from  $N = 128$  to  $N = 256$ . The optimal Smagorinsky model needs to give extra dissipation to account for this steeper spectrum, which means that  $C_S$  or the choice of  $\Delta$  should be increased. This might explain why some people prefer to use the geometric mean for  $\Delta$  instead of the minimum. Conclusions in this research are generally for coarse grids with  $N = 64$  and other choices for  $\Delta$  could follow from a finer grid. It is not advised to use a coarser grid for similar simulations, since those gave in accurate results in this research.

### 6.2.3 The choice of LES filter

A different result might occur when a different filter is used for the separation of the higher and lower wavenumbers. The sharp cut-off filter in the frequency domain is used because of its local support and direct relation to the energy spectrum of the non filtered velocity field. Since other filters tend to amplify the spectrum at the lower wavenumbers and decrease the effect of the higher wavenumbers could it be possible that a higher value for  $\Delta$  is needed to approximate the filtered energy spectrum better.



# Bibliography

- [1] R.B. Christoffers. Report on different large eddy simulation models, 2013.
- [2] Geneviève Comte-Bellot and Stanley Corrsin. Simple eulerian time correlation of full-and narrow-band velocity signals in grid-generated, ‘isotropic’ turbulence. *Journal of Fluid Mechanics*, 48(2):273–337, July 1971.
- [3] W.C. Reynolds D. Kwak and J. H. Ferziger. *Three-Dimensional Time Dependent Computation of Turbulent Flow*. Stanford University, 1974.
- [4] T.Iliescu L.C. Berselli and W.J.Layton. *Mathematics of Large Eddy simulation of Turbulent Flow*. Springer, 2006.
- [5] J.M. McDonough. *Introductory lectures on turbulence*. University of Kentucky, 2004.
- [6] Robert Peston. The digital economy bill, 9 April 2010. Accessed 8 February 2012.
- [7] S.B. Pope. *Turbulent Flows*. Cambridge university press, 2000.
- [8] J.H. Ferziger R.A.Clark and W.C. Reynolds. Evaluation of subgrid-scale models using an accurately simulated turbulent flow. *Journal of Fluid Mechanics*, 91(1):1–16, March 1979.
- [9] L.F. Richardson. *Weather Prediction by Numerical Process*. Cambridge university press, 1922.
- [10] F.G. Schmit. *About Boussinesq’s turbulent viscosity hypothesis: historical remarks and a direct evaluation of its validity*. Elsevier Masson, 2007.
- [11] J. Smagorinsky. General circulation experiment with the primitive equations. *Monthly Weather Review*, 91(3):99–164, March 1963.
- [12] W.Rozema. *Low-dissipation methods and models for the simulation of turbulent subsonic flow*. PhD thesis, University of Groningen, 2015.

# A | Implementation of the initial energy spectrum.

## A.1 Fitting the CBC data

Different energy spectra could be used for the construction of the initial velocity field. That is why the created software also allows different kind of input for the energy spectrum, which could be a set of data or a function. For the simulation of homogeneous isotropic turbulence it is necessary that chosen spectrum represents the scales defined in Section 2.6. The spectrum used in the research is based on the experimental results from Geneviève Comte-Bellot and Stanley Corrsin who wrote about it in [2]. This dataset is referred to as the CBC data in reference to the creators. Another research as well as software using these results is found in [12]. The data fits the Kolmogorov theorem in the inertial range, since it has the  $-5/3$  factor which can be seen in Figure (A.1). The data is only available for the measured locations, which are denoted by a square, so it is necessary to interpolate the intermediate points and extrapolate for the values of  $k$  which are smaller than the first measurements. The interpolation is linear and computed on the logarithmic values of  $k$  and  $E$ , because of the linearity after this transformation in the inertial range. This means that the energy  $E$  at a point  $k$  between measure points  $k_1$  and  $k_2$  is given by

$$E(k) = \exp \left( \frac{\log(k/k_1)}{\log(k_2/k_1)} \log(E(k_2)/E(k_1)) + \log(E(k_1)) \right)$$

for  $k_1 < k_2$ . The extrapolation is similar but takes the two smallest measure points and uses these to derive any smaller values. Additional to this it is ensured that the energy is positive and else the energy is set to 0. The CBC spectrum is scaled such that the largest values of  $k$  can not be simulated on the fine grid with  $N_x = N_y = N_z = 256$ , so an extrapolation for the larger wavenumbers is not needed.

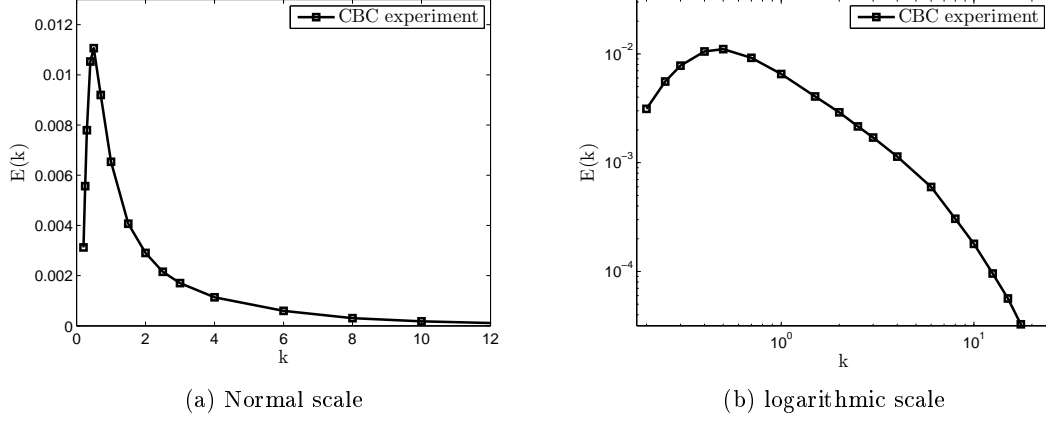


Figure A.1: The initial spectrum domain based on the CBC data

## A.2 Algorithm for the implementation of the spectrum

For the fine simulation is Equation (2.7) used. In addition to that is the energy first integrate over the grid cell in the spatial domain to ensure a smooth spectrum where much of the energy is preserved. This energy is then divided over the grid in the frequency domain, which results in:

$$\|\hat{\mathbf{u}}\|_2^2 = \sqrt{\frac{E(k)}{2\pi((k/(dx\,dy\,dz))^2)} \frac{dk_1\,dk_2\,dk_3}{dx\,dy\,dz}}$$

## A.3 Algorithm for the LES

If the previous algorithm is used for the coarse grid of the LES then the spectrum will have a similar shape around the cut-off wavenumber  $\bar{k} = \sqrt{\max(k_1)^2 + \max(k_2)^2 + \max(k_3)^2}$  as the fine simulation has in its dissipation range. This results in a spectrum which seems to be dissipating too fast by itself even without the effect of any model. A better spectrum for the LES is a spectrum which is the same as the spectrum of the fine simulation up until the cut-off length after which it is set to zero for all wavenumbers. This can be implemented with the following algorithm

$$\|\hat{\mathbf{u}}\|_2^2(\mathbf{k}) = \sqrt{\frac{E(k)}{2\pi((k/(dx\,dy\,dz))^2)} \frac{dk_1\,dk_2\,dk_3}{dx\,dy\,dz}} \sqrt{\frac{\#N_{256}}{\#N_{64}}}$$

where  $\#N_{256}$  is the number of wavenumber of length  $k$  captured in the fine grid of the fine simulation and  $\#N_{64}$  the same for the coarse grid.

The CBC spectrum used for this research and two derived spectra can be seen in Figure A.2. Three things discussed earlier can be seen in this figure. The first is that CBC data reaches a bit further than the spectrum with  $N_x = N_y = N_z$ . The second is that the total energy for the initial spectrum of the LES is less than that of the fine solution.

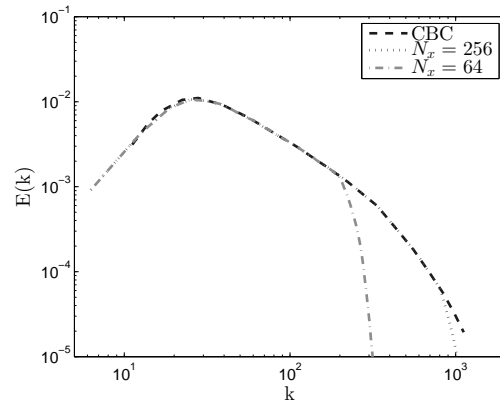


Figure A.2: The used CBC spectrum and two derived initial spectra

## B Derivation of a velocity field on a nonuniform grid

In this section is a derivation of a homogeneous isotropic turbulence velocity field on a nonuniform grid to demonstrate the problems regarding efficiency. The example consists of a two dimensional four by four collocated grid as given in Figure B.1.

The discretisation for the derivatives is the same as in Equation (3.10). The boundary conditions are periodic which means that 18 unknown variables must be solved. In Section 3.3 it is stated that taking the Fourier transform of the divergence free condition resulted in solving the kernel of a convolution. This is in general a costly operation and that is why this example uses another method. The divergence free condition can be written as  $DU = 0$  where  $U = [u_{1,1}, \dots, u_{3,3}, v_{1,1}, \dots, v_{3,3}]^T$  and the discretisation matrix  $D$  is given by

$$D = \begin{bmatrix} & & & & & & & & & \\ & 6 & -6 & & & & & & & \\ -4 & & 4 & & & & & & & \\ 4 & 4 & & & & & & & & \\ & & & 2 & -2 & & & & & \\ & & & & 4 & & -2 & & & \\ & & -4 & & & & -2 & & & \\ & & 4 & -4 & & & & -6 & & \\ & & & & 6 & -6 & 4 & & & \\ & & & & & 4 & & & -4 & \\ & & & & -4 & & 4 & & & \\ & & & & & & & 4 & & \\ & & & & & & & & -4 & -4 \\ & & & & & & & & & 6 \end{bmatrix}$$

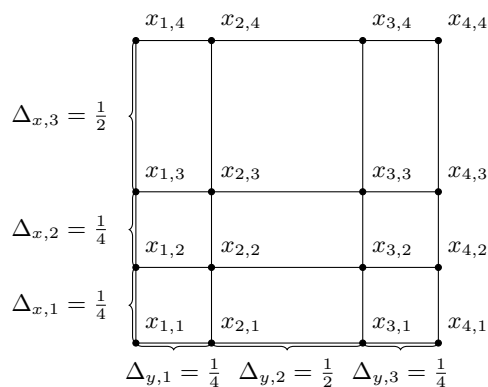


Figure B.1: An example of a nonuniform grid

This means that each solenoidal field is in the nullspace of  $D$ . A few vectors in this space can be directly found because of the sparsity of the matrix. By reducing the matrix with these found elements and by reducing the problem to row echelon form it follows that the following set can be found as a basis for the nullspace of  $D$ .

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \\ -3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 3 \\ -2 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ -3 \\ 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 3 \\ -2 \\ 3 \\ -2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

These vectors will be denoted with  $l_1, l_2, \dots, l_9$ . Any vector in the span of this set of vectors is divergence free. The next step is to limit this set to a smaller set which has the correct energy spectrum. For this it is needed to compute the Fourier modes corresponding to the velocity field. This can be done with the product of each basis of the null space and the following matrix

$$Q(k_1, k_2) = \begin{bmatrix} 1 & \dots & e^{-2\pi i(k_1 x_i + k_2 y_j)} & \dots & e^{-2\pi i(\frac{3k_1}{4} + \frac{k_2}{2})} & 0 & \dots & 0 \\ 0 & \dots & \dots & 0 & 1 & \dots & e^{-2\pi i(k_1 x_i + k_2 y_j)} & \dots & e^{-2\pi i(\frac{3k_1}{4} + \frac{k_2}{2})} \end{bmatrix}$$

The Fourier transform of the velocity field will be of the form

$$\mathcal{F}(\mathbf{u})(k_1, k_2) = Q(k_1, k_2)[q_1 l_1 + q_2 l_2 + q_3 l_3 + q_4 l_4 + q_5 l_5 + q_6 l_6 + q_7 l_7 + q_8 l_8 + q_9 l_9] = Q(k_1, k_2)L$$

Using these Fourier modes allows us to adjust the velocity field such that the energy spectrum is as in Section 2.5. An example of such a restriction is that the mean flow velocity is 0. This is equivalent to stating that the largest flow on the grid has no energy or by using Equation (2.7)  $E(0) = \|\mathcal{F}(\mathbf{u})(0, 0)\|_2^2$ . This results in solving the nonlinear system

$$(3q_1 + 3q_2 + 3q_3 - q_4 - q_5 + q_6 + q_7)^2 + (3q_4 - q_5 + q_6 - q_7 + 3q_8 + 3q_9)^2 = 0$$

For the other restriction to the final velocity field it is necessary to define the energy spectrum and a grid for the frequency domain. The frequency domain grid can be chosen to be uniform while the spatial grid is nonuniform. This is suggested to ensure that the  $q_i \in \mathbb{R}$  for all  $i$  without introducing new restrictions. If the same number of nodes for the grid is chosen as in the spatial domain then it follows that the energy spectrum is only defined for  $\|k\|_2 \in \{0, 1, \sqrt{2}\}$ . A possible

choice for the spectrum could be

$$f = \begin{cases} x^{-5/3} & \text{if } x > 0 \\ 0 & \text{elsewhere} \end{cases}$$

This results in solving the following nonlinear system for all  $q_i$

$$\begin{aligned} 0 &= \|Q(0,0)L\|_2 \\ 1 &= \|Q(1,0)L\|_2 + \|Q(-1,0)L\|_2 + \|Q(0,1)L\|_2 + \|Q(0,-1)L\|_2 \\ 2^{-5/6} &= \|Q(1,1)L\|_2 + \|Q(-1,1)L\|_2 + \|Q(1,-1)L\|_2 + \|Q(-1,-1)L\|_2 \end{aligned}$$

Solving such a system is hard but can be done by a computer. However it is not possible to find the whole solution set. It is possible to derive multiple solutions and to pick one from that set at random to approach a homogeneous isotropic field. However this means that a lot of solutions are necessary to represent the solution space well enough, which will take too long for any real problem.

# C | Source code

```

1 function [] = init_field_main(Nx,Ny,Nz,lx,ly,lz,flag_spectrum,folder,variant,note)
2
3 % This programm is used to create an initial velocity field for a staggered
4 % grid. Created by R. B. Christoffers for his master thesis research project
5
6 % Flags
7     finite_difference_scheme = ...
8         'init_field_central_fourth_order_uniform_staggered';
9         '%init_field_central_second_order_uniform_staggered';
10    used_spectrum = flag_spectrum;
11    % = 1 Spectrum based on CBC results
12    % = 2 Spectrum based on CBC results with sharp cutoff
13    compute_Reynolds = 0;
14    compute_velocity_field = 0;
15    compute_pressure = 0;
16    test_the_results = 1;
17    create_fld00 = 0;
18
19
20 % Counter
21     counter = 1;
22
23 % Dimensions of the problem
24     tic
25
26 %% Resolution
27 %     Nx = 32;
28 %     Ny = 32;
29 %     Nz = 32;
30
31     pressure = zeros(Nx,Ny,Nz);
32
33 %% Dimensions domain
34 %     lx = 1;
35 %     ly = 1;
36 %     lz = 1;
37
38 % Step size
39     dx = lx/Nx;
40     dy = ly/Ny;
41     dz = lz/Nz;
42
43 % Grid
44     x = [dx/2:dx:lx - dx/2];
45     y = [dy/2:dy:ly - dy/2];
46     z = [dz/2:dz:lz - dz/2];
47
48     viscosity = 1.5*10^-1;
49
50     %folder = fullfile('/data/s1892487/',folder);
51     %mkdir(folder);
52
53
54     T = toc;
55     fprintf(['Phase %i: Initiation compleet\n\t\t within %f4 seconds\n'],...
56         counter,T)

```



```

57     counter = counter + 1;
58
59
60 % Constructing the amplitudes of the Fourier components
61 tic
62 if (used_spectrum)
63     quantity = init_field_wavenumbers_length_k(...
64         Nx,Ny,Nz,lx,ly,lz,rounded_spectrum_to_ints);
65 if (used_spectrum == 8)
66     quantity_fine = init_field_wavenumbers_length_k(...
67         256,256,256,lx,ly,lz,rounded_spectrum_to_ints);
68 else
69     quantity_fine = [];
70 end
71 amplitudes = init_field_amplitudes(...
72     quantity,Nx,Ny,Nz,lx,ly,lz,used_spectrum,quantity_fine);
73 fprintf(['Phase %i: Completed the construction of the ',...
74     'correct amplitudes\n\t\t within %f4 seconds\n'],counter,T)
75 counter = counter + 1;
76
77 % Computing the Fourier Modes
78 tic
79 [uk,vk,wk] = init_field_gen_Fourier_nodes(Nx,Ny,Nz,lx,ly,lz,...,
80     dx,dy,dz,finite_difference_scheme,amplitudes,rounded_spectrum_to_ints);
81 T = toc;
82 fprintf(['Phase %i: Completed the spectrum space\n\t\t',...
83     ' within %f4 seconds\n'],counter,T)
84 counter = counter + 1;
85
86 % Compute the Reynolds number
87 if (compute_Reynolds == 1)
88     tic
89     Re = init_field_compute_Reynolds_freq(uk,vk,wk,viscosity,Nx,Ny,Nz);
90     T = toc;
91     fprintf(['Phase %i: The Reynolds number is %f4 \n\t\t',...
92         ' within %f4 seconds\n'],counter,Re,T)
93     counter = counter + 1;
94 end
95
96 % Constructing the velocity field
97 if (compute_velocity_field == 1)
98     tic
99     [u,v,w]= init_field_inverse_fourier(uk,vk,wk,Nx,Ny,Nz,lx,ly,lz);
100    T = toc;
101    fprintf(['Phase %i: Constructed the velocityfield\n\t\t',...
102        ' within %f4 seconds\n'],counter,T)
103    counter = counter + 1;
104 end
105
106 % Compute the pressure
107 if (compute_pressure == 1)
108     tic
109     [pressure] = init_field_pressure(u,v,w,viscosity,Nx,Ny,Nz,lx,ly,lz);
110     T = toc;
111     fprintf(['Phase %i: Computed the pressure\n\t\t',...
112         ' within %f4 seconds\n'],counter,T)
113     counter = counter + 1;
114 end
115
116 % Create fld00.dat file
117 if (create_fld00 == 1)
118     tic
119     init_field_writefield(u,v,w,pressure,Nx,Ny,Nz,folder)
120     T = toc;
121     fprintf(['Phase %i: Created fld00.dat file\n\t\t',...
122         ' within %f4 seconds\n'],counter,T)
123     counter = counter + 1;
124 end
125
126 % Test the results
127 if (test_the_results == 1)
128     fprintf('\nTest results:\n')
129     if (compute_velocity_field == 1)

```

```

130         test_div_free_spatial
131     end
132     test_div_free_Fourier
133     test_spectrum
134     fprintf('\tThe spectrum is displayed in the figure\n')
135     if (compute_velocity_field == 0)
136         fprintf(...
137             '\tset the compute_velocity_field flag to true\n')
138     end
139 end
140 end

```

Listing C.1: Main file

```

1 function [quantity_at_wavenumber] = init_field_wavenumbers_length_k...
2     (Nx,Ny,Nz,lx,ly,lz,rounded_spectrum_to_ints)
3
4 % Description:
5 % This code gives for any wavenumber k the number of vectors which have
6 % the same length. This number is number is given by element
7 % quantity_at_wavenumber(||k||^2). Note that quantity_at_wavenumber(0) is
8 % not included since it is trivial that quantity_at_wavenumber(0) = 1.
9 %
10 % Input:
11 %     Nx          -> number of Fourier node in k1-direction
12 %     Ny          -> number of Fourier node in k2-direction
13 %     Nz          -> number of Fourier node in k3-direction
14 %     lx          -> length of the domain in the x-direction
15 %     ly          -> length of the domain in the y-direction
16 %     lz          -> length of the domain in the z-direction
17 %
18 % Output:
19 %     quantity_at_wavenumber -> number of wavenumbers with same length
20
21 counter = 0;
22 for ii = -Nx/2 + 1:Nx/2 - 1
23     for jj = -Ny/2 + 1:Ny/2 - 1
24         for kk = -Nz/2 + 1:Nz/2 - 1
25             if (ii == 0 && jj == 0 && kk == 0)
26                 else
27                     counter = counter + 1;
28                     occuring_wavenumber(counter) = 2*pi*sqrt((ii/lx)^2 + (jj/ly)^2 + (kk/lz
29 )^2);
30                     if (rounded_spectrum_to_ints == 1)
31                         occuring_wavenumber(counter) = round(occuring_wavenumber(counter));
32                     end
33                 end
34             end
35         end
36     end
37     occuring_wavenumber = sort(occuring_wavenumber);
38     counter = 1;
39     quantity_at_wavenumber(counter,2) = 0;
40
41     for ii = 1:length(occuring_wavenumber) - 1
42         quantity_at_wavenumber(counter,1) = occuring_wavenumber(ii);
43         quantity_at_wavenumber(counter,2) = quantity_at_wavenumber(counter,2) + 1;
44         if (quantity_at_wavenumber(counter,1) == occuring_wavenumber(ii + 1))
45             else
46                 counter = counter + 1;
47                 quantity_at_wavenumber(counter,2) = 0;
48             end
49         end
50     quantity_at_wavenumber(counter,2) = quantity_at_wavenumber(counter,2) + 1;

```

Listing C.2: Code for counting the number of wavenumbers with a certain length

```

1 function [amplitude] = init_field_amplitudes(quantity_at_wavenumber,...
2     Nx,Ny,Nz,lx,ly,lz,used_spectrum,quantity_at_wavenumber_fine_mesh)
3

```

```

4 % Description:
5 %   This program constructs a vector AMPLITUDE with at AMPLITUDE(i) the
6 %   amplitude of an Fourier nodes with the length of the wavenumber as sqrt(i).
7 %   (example K = [1 2 -1] -> i = 6)
8 %
9 % Input:
10 %   Nx          -> number of Fourier node in k1-direction
11 %   Ny          -> number of Fourier node in k2-direction
12 %   Nz          -> number of Fourier node in k3-direction
13 %   quantity_at_wavenumber -> number of wavenumber with this length
14 %
15 % Output:
16 %   amplitude    -> the amplitude of each wavenumber
17
18 [n,m] = size(quantity_at_wavenumber);
19 amplitude = [quantity_at_wavenumber(:,1), zeros(n,1)];
20
21 elseif (used_spectrum == 1)
22     for ii = 1:n
23         if(amplitude(ii,1) == 0)
24             amplitude(ii,2) = 0;
25         else
26             dx = lx/Nx;
27             dy = ly/Ny;
28             dz = lz/Nz;
29             amplitude(ii,2) = sqrt( init_field_energyspectrum_CBC(
30                 quantity_at_wavenumber...
31                 (ii,1))/(2*pi*(quantity_at_wavenumber(ii,1))^2))/(dx*dy*dz)*(2
32                 *pi)^1.5/(lx*ly*lz)^0.5;
33             end
34         end
35     end
36 elseif (used_spectrum == 2)
37     for ii = 1:n
38         if(amplitude(ii,1) == 0)
39             amplitude(ii,2) = 0;
40         else
41             dx = lx/Nx;
42             dy = ly/Ny;
43             dz = lz/Nz;
44
45             jj = find(quantity_at_wavenumber(ii,1) ==
46                 quantity_at_wavenumber_fine_mesh(:,1));
47
48             amplitude(ii,2) = sqrt( init_field_energyspectrum_CBC(
49                 quantity_at_wavenumber...
50                 (ii,1))/(2*pi*(quantity_at_wavenumber(ii,1))^2))/(dx*dy*dz)*(2
51                 *pi)^1.5/(lx*ly*lz)^0.5*...
52                 (quantity_at_wavenumber_fine_mesh(jj,2)/quantity_at_wavenumber
53                 (ii,2))^0.5;
54             end
55         end
56     end
57 end

```

Listing C.3: Code for computing the amplitude of each Fourier term

```

1 function [energy] = init_field_energyspectrum_CBC(k)
2
3 % Description:
4 %   This code is used to compute the energy at k. The shape of the spectrum is
5 %   given by f,g and h.
6 %
7 % Input:
8 %   k          -> the current square of the wavenumber length
9 %
10 % Output:
11 %   energy     -> the energy at all wavenumber with length sqrt(k)
12
13
14 %E = [ 129    230    322    435    457    380    270    ...
15 %     168    120    89     70.3  47     24.7  12.6    ...
16 %     7.42   3.9   2.33   1.35   0.8];
17 %K = [ 0.2    0.25   0.3    0.4    0.5    0.7    1.0    ...

```

```

18 %          1.5      2.0      2.5      3.0      4.0      6.0      8.0 ...
19 %          10.0     12.5     15       17.5     20.0];
20
21 E =      [    0.003123      0.005568      0.007795      0.010530 ...
22            0.011063      0.009199      0.006536      0.004067 ...
23            0.002905      0.002154      0.001702      0.001138 ...
24            0.000598      0.000305      0.000180      0.000096 ...
25            0.000056      0.000033      0.000019];
26
27 K =      [   11.176000      13.970000      16.764000      22.352000 ...
28            27.940000      39.116000      55.880000      83.820000 ...
29            111.760000      139.700000      167.640000      223.52000 ...
30            335.280000      447.040000      558.800000      698.50000 ...
31            838.200000      977.900000      1117.600000];
32
33 if (k == 0)
34     energy = 0;
35 else
36     if any(k == K)
37         energy = (E(k==K));
38     else
39         if (k > K(end))
40             k1 = K(end - 1);
41             k2 = K(end);
42             E1 = E(end - 1);
43             E2 = E(end);
44         elseif (k < K(1))
45             k1 = K(1);
46             k2 = K(2);
47             E1 = E(1);
48             E2 = E(2);
49         else
50             index = find(K>k,1);
51             k1 = K(index - 1);
52             k2 = K(index);
53             E1 = E(index - 1);
54             E2 = E(index);
55         end
56         energy = max(0,exp(log(k/k1)/log(k2/k1)*log(E2/E1) + log(E1)));
57     end
58 end

```

Listing C.4: Code for implementing the right shape of the spectrum

```

1 function [K] = init_field_central_fourth_order_uniform_staggered(k,L,N)
2
3 % Description:
4 %   This code derives the modified wavenumber corresponding to wavenumber k and
5 %   The fourth order central derivative
6 %
7 % Input:
8 %   k          -> current wavenumber
9 %   L          -> Length of the domain
10 %   N          -> number of node in corresponding direction
11 %
12 % Output:
13 %   amplitude  -> the modified wavenumber
14
15 K = (27*sin(pi*k/N) - sin(3*pi*k/N)) / (12*L/N);
16 end

```

Listing C.5: Code for the used discretisation and the modified wavenumber

```

1 function [uk,vk,wk] = init_field_gen_Fourier_nodes(...
2     Nx,Ny,Nz,lx,ly,lz,dx,dy,dz,finite_difference_scheme,amplitudes,...
3     rounded_spectrum_to_ints)
4
5 % Description:
6 %   This subroutine construct the general Fourier nodes.
7 %
8 % Input:

```

```

9  % Nx -> number of Fourier node in k1-direction
10 % Ny -> number of Fourier node in k2-direction
11 % Nz -> number of Fourier node in k3-direction
12 % dx -> gridwidth x-direction
13 % dy -> gridwidth y-direction
14 % dz -> gridwidth z-direction
15 % finite_difference_scheme -> the scheme corresponding to the problem
16 % amplitudes -> the amplitude of each wavenumber
17 %
18 % Output:
19 % uk -> Fourier terms corresponding to u
20 % vk -> Fourier terms corresponding to v
21 % wk -> Fourier terms corresponding to w
22 %
23 % Initialisation
24 uk = zeros(Nx,Ny,Nz);
25 vk = zeros(Nx,Ny,Nz);
26 wk = zeros(Nx,Ny,Nz);
27 Random_degree = 100000;
28 rng('shuffle')
29 %
30 % Calling the right function or the modified wavenumber
31 scheme = str2func(finite_difference_scheme);
32 %
33 % Constructing the Fourier nodes
34 for k3 = -Nz/2 + 1 : 1 : Nz/2 - 1
35 for k2 = -Ny/2 + 1 : 1 : Ny/2 - 1
36 for k1 = -Nx/2 + 1 : 1 : 0 % only need halve the k's since u is real
37
38 % Square of the length of the wavenumbervector
39 tempk = 2*pi*sqrt((k1/lx)^2 + (k2/ly)^2 + (k3/lz)^2);
40
41 % The restrictions are to make sure that we ddo not compute unnecessary modes
42 if (k1 < 0 || (k1 == 0 && (k2 < 0 || (k2 == 0 && k3 < 0))))
43
44 % Deriving the Modified wavenumbers for each direction
45 K1 = scheme(k1,lx,Nx);
46 K2 = scheme(k2,ly,Ny);
47 K3 = scheme(k3,lz,Nz);
48
49 % Constructing the orthogonal unit vector
50 if (K1 ~= 0)
51 L = 1/sqrt(2*K1^2 + K2^2 + 2*K2*K3 + K3^2);
52 u_temp = L*(K2 + K3);
53 v_temp = L*-K1;
54 w_temp = L*-K1;
55 elseif (K2 ~= 0)
56 L = 1/sqrt(2*K2^2 + K1^2 + 2*K1*K3 + K3^2);
57 u_temp = L*-K2;
58 v_temp = L*(K1 + K3);
59 w_temp = L*-K2;
60 elseif (K3 ~= 0)
61 L = 1/sqrt(2*K3^2 + K1^2 + 2*K1*K2 + K2^2);
62 u_temp = L*-K3;
63 v_temp = L*-K3;
64 w_temp = L*(K1 + K2);
65 end
66
67 L = sqrt(K1^2 + K2^2 + K3^2);
68 K1 = K1/L;
69 K2 = K2/L;
70 K3 = K3/L;
71
72 % Constructing the real part
73 angle = 2*pi/Random_degree*randi(Random_degree,1);
74
75 u_real = (K1*K1*u_temp + K2*K1*v_temp + K3*K1*w_temp)*...
76 (1 - cos(angle)) + u_temp*cos(angle) + (K2*w_temp - K3*v_temp)*...
77 sin(angle);
78 v_real = (K1*K2*u_temp + K2*K2*v_temp + K3*K2*w_temp)*...
79 (1 - cos(angle)) + v_temp*cos(angle) + (K3*u_temp - K1*w_temp)*...
80 sin(angle);
81 w_real = (K1*K3*u_temp + K2*K3*v_temp + K3*K3*w_temp)*...

```

```

82         (1 - cos(angle)) + w_temp*cos(angle) + (K1*v_temp - K2*u_temp)*...
83         sin(angle);
84
85     % Constructing the imaginary part
86     angle = 2*pi/Random_degree*randi(Random_degree,1);
87
88     u_imag = (K1*K1*u_temp + K2*K1*v_temp + K3*K1*w_temp)*...
89             (1 - cos(angle)) + u_temp*cos(angle) + (K2*w_temp - K3*v_temp)*...
90             sin(angle);
91     v_imag = (K1*K2*u_temp + K2*K2*v_temp + K3*K2*w_temp)*...
92             (1 - cos(angle)) + v_temp*cos(angle) + (K3*u_temp - K1*w_temp)*...
93             sin(angle);
94     w_imag = (K1*K3*u_temp + K2*K3*v_temp + K3*K3*w_temp)*...
95             (1 - cos(angle)) + w_temp*cos(angle) + (K1*v_temp - K2*u_temp)*...
96             sin(angle);
97
98     % Putting both parts together
99     angle = 2*pi/Random_degree*randi(Random_degree,1);
100
101     u_temp = sin(angle)*u_real + i*cos(angle)*u_imag;
102     v_temp = sin(angle)*v_real + i*cos(angle)*v_imag;
103     w_temp = sin(angle)*w_real + i*cos(angle)*w_imag;
104
105     % Fixing the amplitude
106     if (rounded_spectrum_to_ints == 1)
107         temp = (amplitudes(:,1) == round(tempk));
108     else
109         temp = (amplitudes(:,1) == tempk);
110     end
111     amp = amplitudes(temp,2);
112     uk(k1 + Nx/2 + 1, k2 + Ny/2 + 1, k3 + Nz/2 + 1) = u_temp*amp;
113     vk(k1 + Nx/2 + 1, k2 + Ny/2 + 1, k3 + Nz/2 + 1) = v_temp*amp;
114     wk(k1 + Nx/2 + 1, k2 + Ny/2 + 1, k3 + Nz/2 + 1) = w_temp*amp;
115 end
116 end
117 end
118 end
119
120 for kk = 2:Nz
121     for jj = 2:Ny
122         for ii = Nx/2 + 2 : Nx
123             uk(ii,jj,kk) = conj(uk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
124             vk(ii,jj,kk) = conj(vk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
125             wk(ii,jj,kk) = conj(wk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
126         end
127     end
128 end
129
130 for kk = 2:Nz
131     for jj = Ny/2 + 2:Ny
132         for ii = Nx/2 + 1
133             uk(ii,jj,kk) = conj(uk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
134             vk(ii,jj,kk) = conj(vk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
135             wk(ii,jj,kk) = conj(wk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
136         end
137     end
138 end
139
140 for kk = 2:Nz
141     for jj = Ny/2 + 1
142         for ii = Nx/2 + 1
143             uk(ii,jj,kk) = conj(uk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
144             vk(ii,jj,kk) = conj(vk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
145             wk(ii,jj,kk) = conj(wk(Nx + 2 - ii,Ny + 2 - jj, Nz + 2 - kk));
146         end
147     end
148 end

```

Listing C.6: Code for the construction of the Fourier terms

```

1 function [Re] = init_field_compute_Reynolds_freq(uk,vk,wk,viscosity,Nx,Ny,Nz)
2

```

```

3
4 u_ref = sqrt(sum(sum(sum(uk.*conj(uk) + vk.*conj(vk) + wk.*conj(wk)))/(3*(Nx*Ny*
  Nz)^2)));
5
6 l_ref = 2*pi;
7
8 Re = u_ref*l_ref/viscosity;

```

Listing C.7: Code for the derivation of the Reynolds number

```

1 function [u,v,w] = init_field_inverse_fourier(uk,vk,wk,Nx,Ny,Nz,lx,ly,lz)
2
3 % Transform the modes to the location corresponding to the staggered grid
4 kx = [-Nx/2:Nx/2 - 1];
5 ky = [-Ny/2:Ny/2 - 1];
6 kz = [-Nz/2:Nz/2 - 1];
7
8 trans_fac_x = exp(1i*pi*kx/Nx);
9 trans_fac_y = exp(1i*pi*ky/Ny);
10 trans_fac_z = exp(1i*pi*kz/Nz);
11
12 for kk = 1:Nz
13     for jj = 1:Ny
14         for ii = 1:Nx
15             uk(ii,jj,kk) = trans_fac_x(ii)*uk(ii,jj,kk);
16             vk(ii,jj,kk) = trans_fac_y(jj)*vk(ii,jj,kk);
17             wk(ii,jj,kk) = trans_fac_x(kk)*wk(ii,jj,kk);
18         end
19     end
20 end
21
22 % Matlab routine works with shifted values
23 uk_shift = ifftshift(uk);
24 vk_shift = ifftshift(vk);
25 wk_shift = ifftshift(wk);
26
27
28 % perform the Fourier transform
29 u = ifftn(uk_shift);
30 v = ifftn(vk_shift);
31 w = ifftn(wk_shift);
32
33 end

```

Listing C.8: The Matlab implementation of the Fourier Inverse

```

1 function [pressure] = init_field_pressure(u,v,w,viscosity,Nx,Ny,Nz,lx,ly,lz)
2
3
4 Delta_x = lx/Nx;
5 Delta_y = ly/Ny;
6 Delta_z = lz/Nz;
7
8 % Derivative of the velocity
9
10 dudx = (u([end,1:end - 1],:,:) - 27*u(:,:,:) + 27*u([2:end,1],:,:)) ...
11         - u([3:end,1:2],:,:))/(24*Delta_x);
12 dudy = (u(:,[end,1:end - 1],:) - 27*u(:,:,:) + 27*u(:,[2:end,1],:)) ...
13         - u(:,[3:end,1:2],:))/(24*Delta_y);
14 dudz = (u(:,:[end,1:end - 1]) - 27*u(:,:,:) + 27*u(:,:[2:end,1]) ...
15         - u(:,:[3:end,1:2]))/(24*Delta_z);
16
17 dvdx = (v([end,1:end - 1],:,:) - 27*v(:,:,:) + 27*v([2:end,1],:,:)) ...
18         - v([3:end,1:2],:,:))/(24*Delta_x);
19 dvdy = (v(:,[end,1:end - 1],:) - 27*v(:,:,:) + 27*v(:,[2:end,1],:)) ...
20         - v(:,[3:end,1:2],:))/(24*Delta_y);
21 dvdz = (v(:,:[end,1:end - 1]) - 27*v(:,:,:) + 27*v(:,:[2:end,1]) ...
22         - v(:,:[3:end,1:2]))/(24*Delta_z);
23
24 dwdx = (w([end,1:end - 1],:,:) - 27*w(:,:,:) + 27*w([2:end,1],:,:)) ...
25         - w([3:end,1:2],:,:))/(24*Delta_x);

```

```

26 | dwdy = (w(:,[end,1:end-1],:,:) - 27*w(:,:,:) + 27*w(:,[2:end,1],:)) ...
27 | - w(:,[3:end,1,2],:,:))/(24*Delta_y);
28 | dwdz = (w(:,:,[end,1:end-1]) - 27*w(:,:,:) + 27*w(:,:,[2:end,1]) ...
29 | - w(:,:,[3:end,1,2]))/(24*Delta_z);
30
31
32 | du2dx2 = ( ...
33 |     dudx([end-1,end,1:end-2],:,:) - 8*dudx([end,1:end-1],:,:) + ...
34 |     8*dudx([2:end,1],:,:) - dudx([3:end,1,2],:,:))/(12*Delta_x);
35 | du2dy2 = ( ...
36 |     dudy(:,[end-1,end,1:end-2],:) - 8*dudy(:,[end,1:end-1],:) + ...
37 |     8*dudy(:,[2:end,1],:) - dudy(:,[3:end,1,2],:))/(12*Delta_y);
38 | du2dz2 = ( ...
39 |     dudz(:,,[end-1,end,1:end-2]) - 8*dudz(:,,[end,1:end-1]) + ...
40 |     8*dudz(:,,[2:end,1]) - dudz(:,,[3:end,1,2]))/(12*Delta_z);
41
42 | dv2dx2 = ( ...
43 |     dvdx([end-1,end,1:end-2],:,:) - 8*dvdx([end,1:end-1],:,:) + ...
44 |     8*dvdx([2:end,1],:,:) - dvdx([3:end,1,2],:,:))/(12*Delta_x);
45 | dv2dy2 = ( ...
46 |     dvdy(:,[end-1,end,1:end-2],:) - 8*dvdy(:,[end,1:end-1],:) + ...
47 |     8*dvdy(:,[2:end,1],:) - dvdy(:,[3:end,1,2],:))/(12*Delta_y);
48 | dv2dz2 = ( ...
49 |     dvdz(:,,[end-1,end,1:end-2]) - 8*dvdz(:,,[end,1:end-1]) + ...
50 |     8*dvdz(:,,[2:end,1]) - dvdz(:,,[3:end,1,2]))/(12*Delta_z);
51
52 | dw2dx2 = ( ...
53 |     dwdx([end-1,end,1:end-2],:,:) - 8*dwdx([end,1:end-1],:,:) + ...
54 |     8*dwdx([2:end,1],:,:) - dwdx([3:end,1,2],:,:))/(12*Delta_x);
55 | dw2dy2 = ( ...
56 |     dwdy(:,[end-1,end,1:end-2],:) - 8*dwdy(:,[end,1:end-1],:) + ...
57 |     8*dwdy(:,[2:end,1],:) - dwdy(:,[3:end,1,2],:))/(12*Delta_y);
58 | dw2dz2 = ( ...
59 |     dwdz(:,,[end-1,end,1:end-2]) - 8*dwdz(:,,[end,1:end-1]) + ...
60 |     8*dwdz(:,,[2:end,1]) - dwdz(:,,[3:end,1,2]))/(12*Delta_z);
61
62
63 | u_aver = (-u([end,1:end-1],:,:) + 9*u(:,:,:) + ...
64 |     9*u([2:end,1],:,:) - u([3:end,1,2],:,:))/16;
65 | v_aver = (-v(:,[end,1:end-1],:) + 9*v(:,:,:) + ...
66 |     9*v(:,[2:end,1],:) - v(:,[3:end,1,2],:))/16;
67 | w_aver = (-w(:,:,[end,1:end-1]) + 9*w(:,:,:) + ...
68 |     9*w(:,:,[2:end,1]) - w(:,:,[3:end,1,2]))/16;
69
70 | RighthandSideU = viscosity.*(du2dx2 + du2dy2 + du2dz2) - ...
71 |     (u_aver.*dudx + v_aver.*dudy + w_aver.*dudz);
72 | RighthandSideV = viscosity.*(dv2dx2 + dv2dy2 + dv2dz2) - ...
73 |     (u_aver.*dvdx + v_aver.*dvdy + w_aver.*dvdz);
74 | RighthandSideW = viscosity.*(dw2dx2 + dw2dy2 + dw2dz2) - ...
75 |     (u_aver.*dwdx + v_aver.*dwdy + w_aver.*dwdz);
76
77 | GradRHS = ...
78 |     ( RighthandSideU([end-1,end,1:end-2],:,:) ...
79 |     - 8*RighthandSideU([end,1:end-1],:,:) ...
80 |     + 8*RighthandSideU([2:end,1],:,:)) ...
81 |     - RighthandSideU([3:end,1,2],:,:))/(12*Delta_x) + ...
82 |     ( RighthandSideV(:,[end-1,end,1:end-2],:) ...
83 |     - 8*RighthandSideV(:,[end,1:end-1],:) ...
84 |     + 8*RighthandSideV(:,[2:end,1],:)) ...
85 |     - RighthandSideV(:,[3:end,1,2],:))/(12*Delta_y) + ...
86 |     ( RighthandSideW(:,,[end-1,end,1:end-2]) ...
87 |     - 8*RighthandSideW(:,,[end,1:end-1]) ...
88 |     + 8*RighthandSideW(:,,[2:end,1]) ...
89 |     - RighthandSideW(:,,[3:end,1,2]))/(12*Delta_z);
90
91 |
92 | FourierRHS = fftn(GradRHS);
93
94 | % Orientation of [k1,k2,k3] is based on the fft from Matlab
95 | for k3 = 0:Nz - 1
96 | for k2 = 0:Ny - 1
97 | for k1 = 0:Nx - 1
98 |

```



```

99     K_pressure(k1 + 1,k2 + 1,k3 + 1) = ...
100     (cos(8*pi*k1/Nx) - 16*cos(6*pi*k1/Nx) + ...
101     64*cos(4*pi*k1/Nx) + 16*cos(2*pi*k1/Nx) - 65) ...
102     / (72*Delta_x^2) + ...
103     (cos(8*pi*k2/Ny) - 16*cos(6*pi*k2/Ny) + ...
104     64*cos(4*pi*k2/Ny) + 16*cos(2*pi*k2/Ny) - 65) ...
105     / (72*Delta_y^2) + ...
106     (cos(8*pi*k3/Nz) - 16*cos(6*pi*k3/Nz) + ...
107     64*cos(4*pi*k3/Nz) + 16*cos(2*pi*k3/Nz) - 65) ...
108     / (72*Delta_z^2);
109
110 end
111 end
112 end
113
114 Fourier_pressure = zeros(Nx,Ny,Nz);
115 I = (K_pressure ~= 0);
116 Fourier_pressure(I) = FourierRHS(I)./K_pressure(I);
117
118 pressure = ifftn(Fourier_pressure);
119
120 end

```

Listing C.9: The Matlab implementation for the computation of the pressure

```

1 function [] = init_field_writefield(u,v,w,p,Nx,Ny,Nz,folder)
2
3 counter = 1;
4
5 for kk = 1:Nz
6 for jj = 1:Ny
7     big([(counter - 1)*Nx + 1:counter*Nx],:) = real([u(:,jj,kk),v(:,jj,kk),w(:,jj,
8     kk),p(:,jj,kk)]);
9     counter = counter + 1;
10 end
11 end
12
13 if (exist(fullfile(folder,'fld00.dat'))
14     delete(fullfile(folder,'fld00.dat'))
15 end
16
17 str = ' 0.000000';
18 dlmwrite(fullfile(folder,'fld00.dat'),str,'-append','delimiter','', 'newline', 'pc
19 %26.17f', 'newline', 'pc')
20 end

```

Listing C.10: The Matlab implementation for the creation of fld00.dat

D | All results from the uniform domain with a regular grid

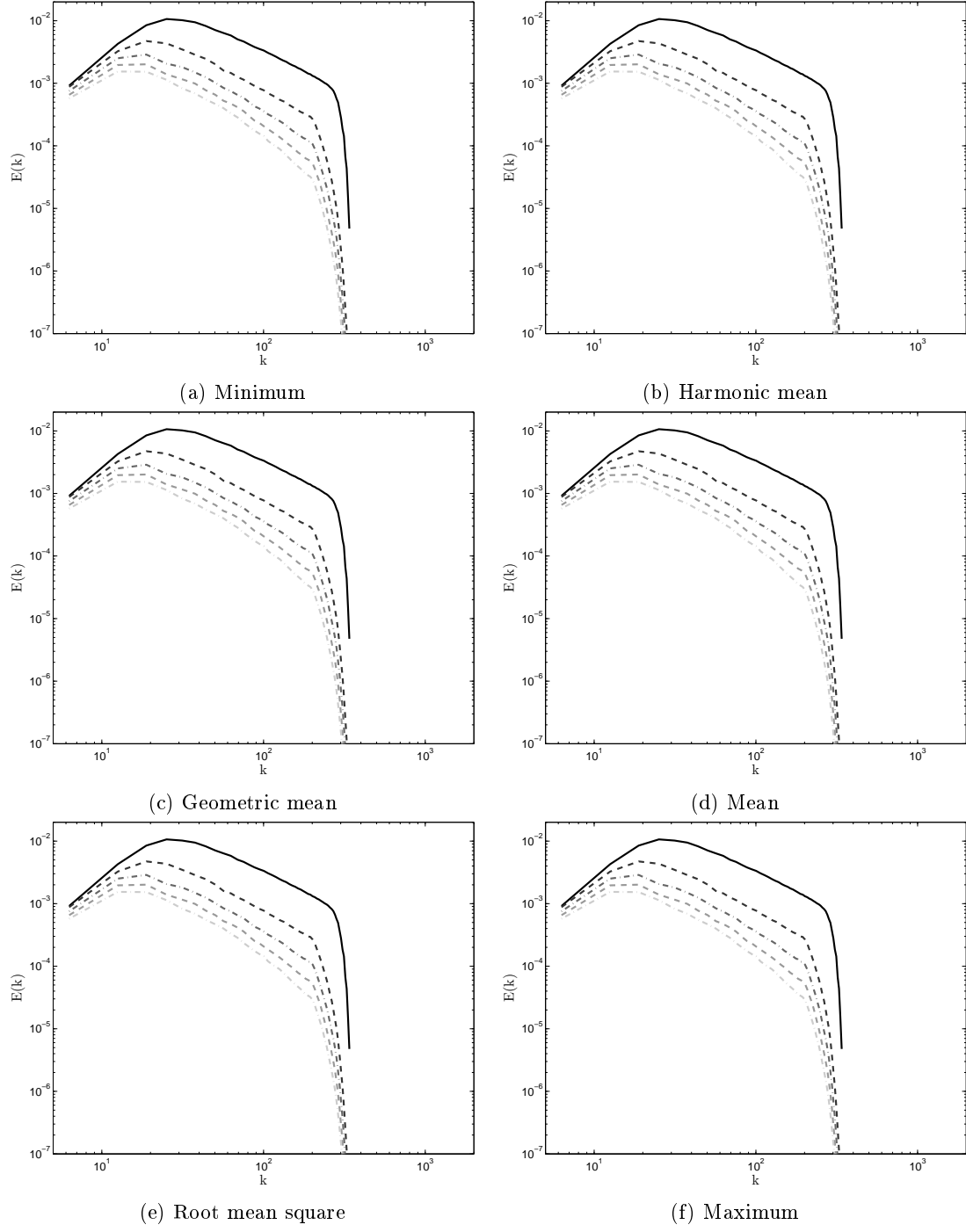


Figure D.1:  $N_x = N_y = N_z = 64$ : (—),  $t = 0.159432$  (---),  $t = 0.318864$  (- - -),  $t = 0.478296$  (· · ·),  $t = 0.63772$  (---) and  $t = 0.79716$  (- - -)

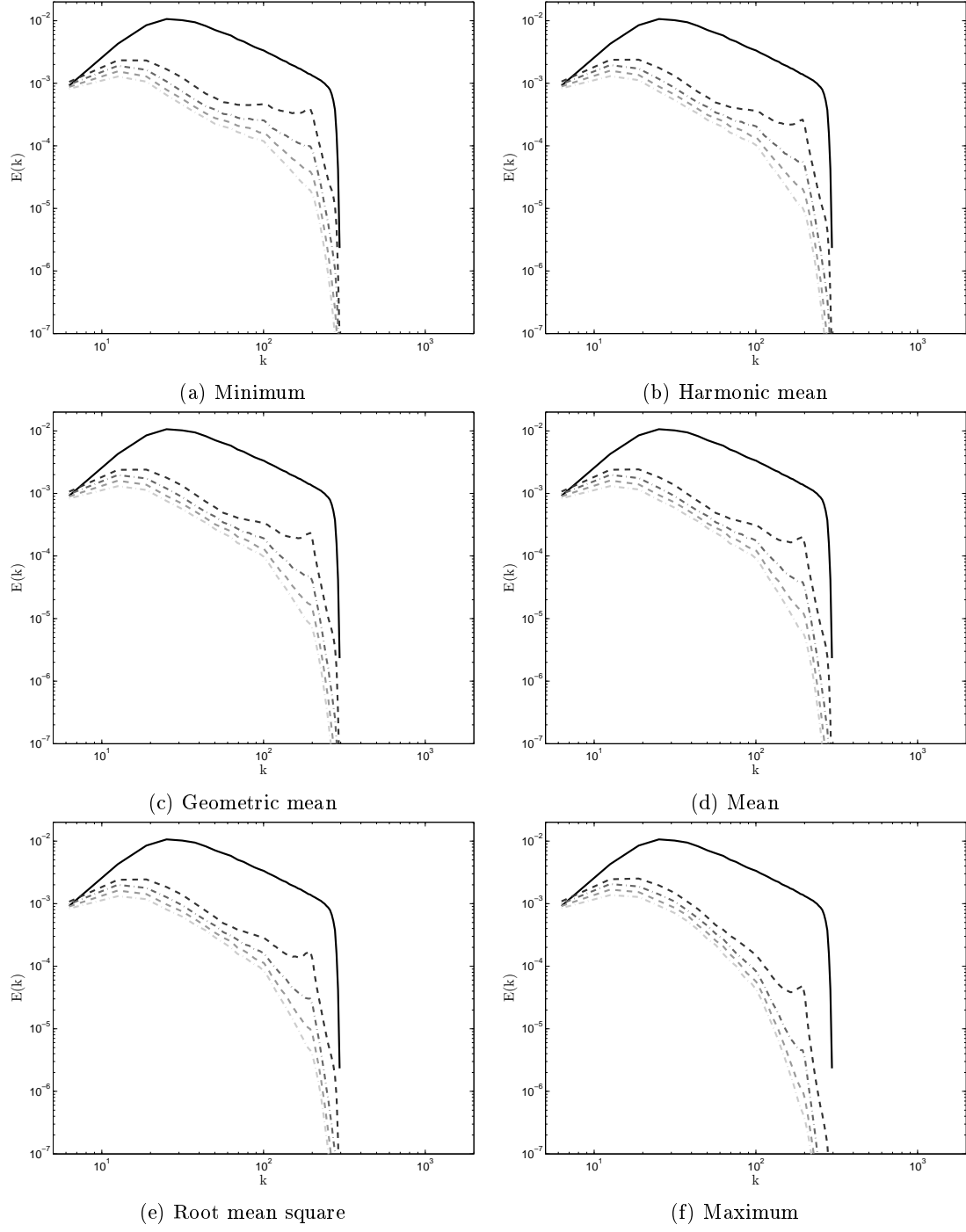


Figure D.2:  $N_x = 32$  and  $N_y = N_z = 64$ : (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (-·-·-),  $t = 0.63772$  (···) and  $t = 0.79716$  (- - - -)

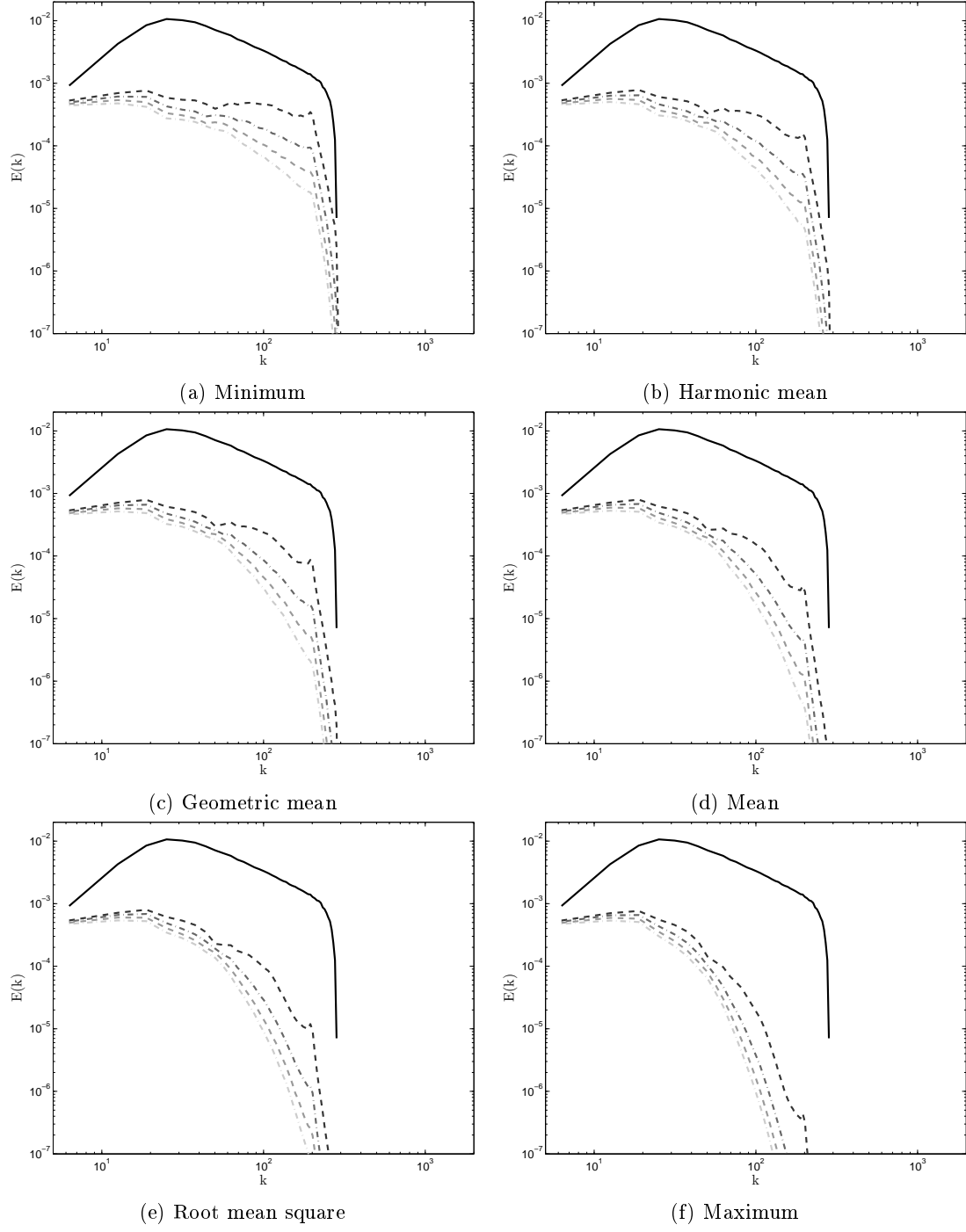


Figure D.3:  $N_x = 16$  and  $N_y = N_z = 64$ : (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (---),  $t = 0.63772$  (---) and  $t = 0.79716$  (---)

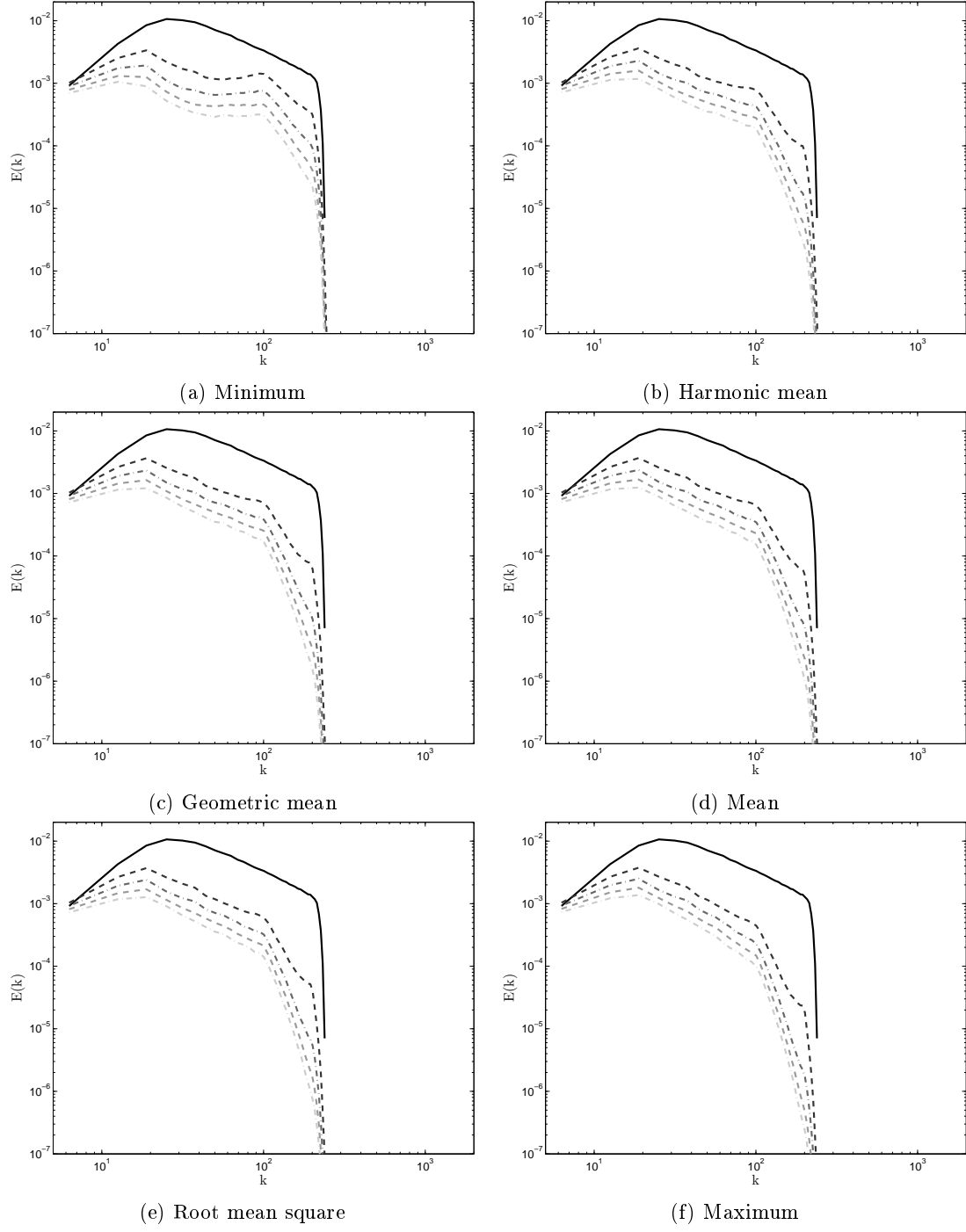


Figure D.4:  $N_x = N_z = 32$  and  $N_y = 64$ : (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (-·-·-),  $t = 0.63772$  (···) and  $t = 0.79716$  (- - - -)

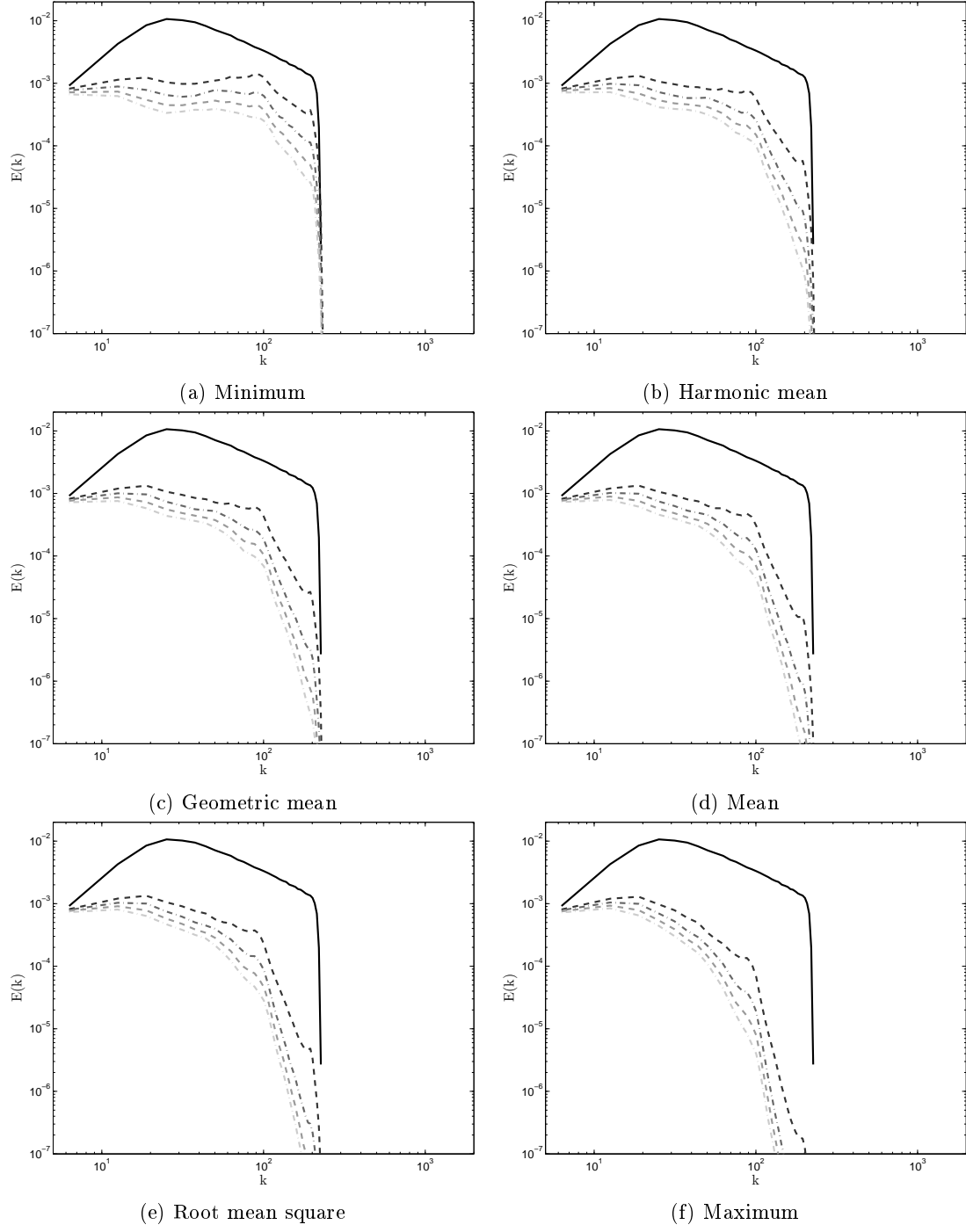


Figure D.5:  $N_x = 16$ ,  $N_y = 64$  and  $N_z = 32$ : (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (-·-·-),  $t = 0.63772$  (···) and  $t = 0.79716$  (- - - -)

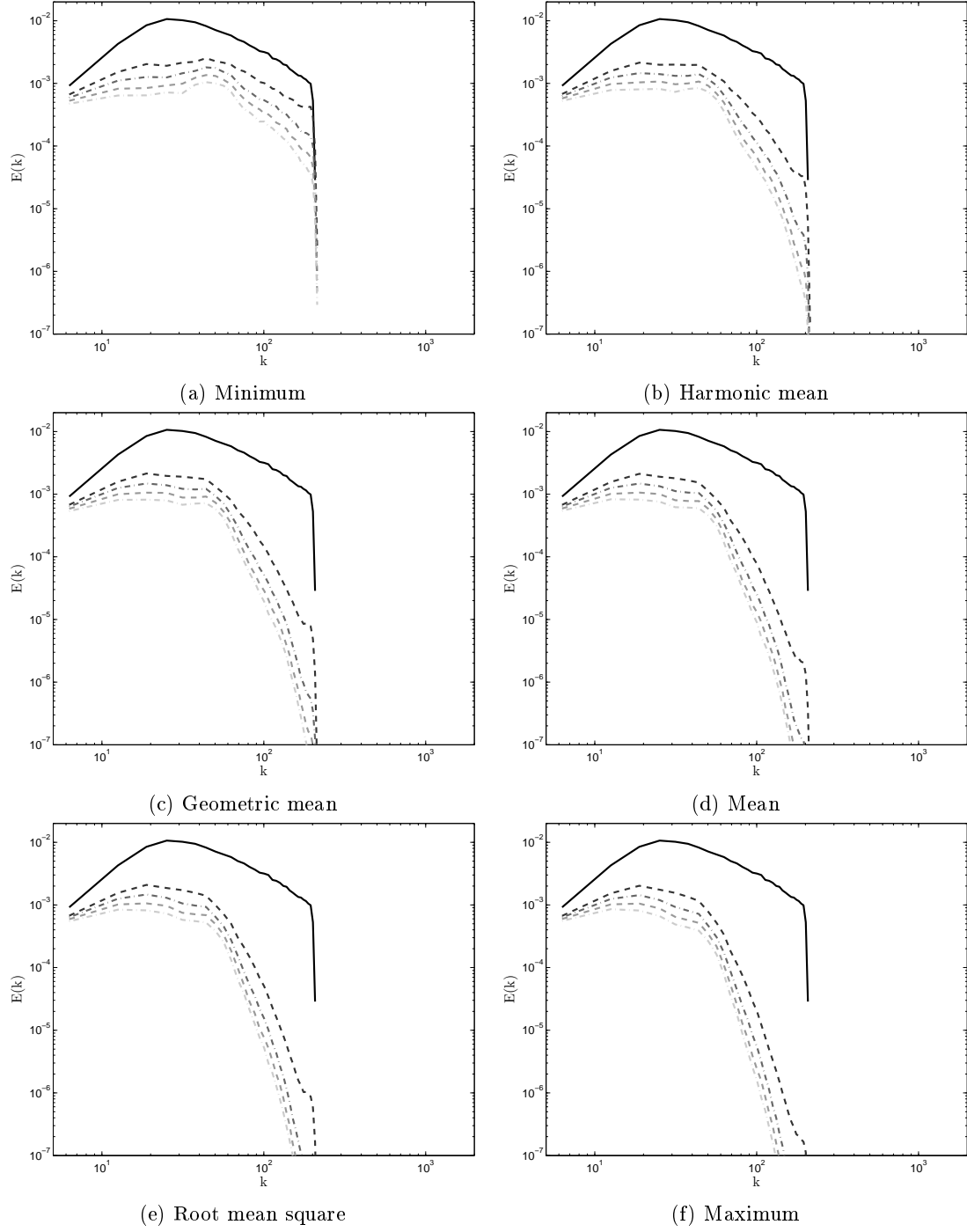


Figure D.6:  $N_x = N_z = 16$  and  $N_y = 64$ : (—),  $t = 0.159432$  (---),  $t = 0.318864$  (---),  $t = 0.478296$  (---),  $t = 0.63772$  (---) and  $t = 0.79716$  (---)