NTNU
Norwegian University of
Science and Technology

Numerical Simulations of Flow around a Simplified Hull Form

Øyvind Andre Hagen Emblemsvåg

Marine Technology
Submission date: June 2016
Supervisor: Håvard Holm, IMT

Norwegian University of Science and Technology
Department of Marine Technology

NTNU
Norges teknisk-naturvitenskapelige universitet
Institutt for marin teknikk

MASTER'S THESIS IN MARINE HYDRODYNAMICS

Spring 2016

FOR

Stud.techn. Øyvind Andre Hagen Emblemsvåg

Numerical Simulations of Flow around a Simplified Hull Form

The candidate will familiarize himself with a CFD computer program, both the theory and the practicalities related to modelling different geometries of interest, simulating the flow, and post processing the data. The candidate should familiarize himself with the programs functionality, possibilities and limitations.

The candidate shall run simplified simulations of flow around a model of a simplified ship hull. The candidate should as far as possible control the results of the simulations with results from the literature.

Different turbulence models are to be tested. Differences, strengths and weaknesses of the models should be documented.

It is presumed that the Institute of Marine Technology, NTNU, can use the results freely in its research, with references to the students work.

The thesis shall be formatted as a research paper, with an abstract, conclusion, bibliography etc. The candidate should write as concise, clear and well written as possible.

The thesis is to be delivered in two copies within June 10, 2016.

Håvard Holm
Associate Professor

Preface

This thesis concludes my M.Sc. degree in Marine Technology at the Norwegian University of Science and Technology (NTNU). The topic is a result of suggestions from my supervisor and from DNVGL, and a continuous discussion along the way.

I would like to thank my supervisor Håvard Holm for his helpful guidance, and Tufan Arslan for his help with the software. Also, I would like to thank Olav Rognebakke of DNVGL for interesting discussions on the topic.

Trondheim, June 10, 2016

[illegible]

Øyvind Andre Hagen Emblemsvåg

Abstract

Hydrodynamic flow around a simplified hull form has been investigated using numerical simulations in OpenFOAM. The flow in question is three-dimensional and turbulent at a Reynolds number of 26,400. The base geometry is a rectangular cylinder (or box) with an aspect ratio of 5:1, and flow in longitudinal (x-)direction (cross-flow width is 1 and in-flow length is 5). The height of the body is 1 in the z-direction, hence it is quadratic in the y-z plane. Simulations have been run in parallel on the supercomputer Vilje at NTNU.

Three different cases have been investigated. Two of the cases are concerned with double-body flow. This means the geometry is "doubled" (mirrored across the x-y plane), and fully submerged with fluid on all sides. The difference between these two cases is the turbulence modelling; one uses Reynolds Averaged Navier-Stokes (RANS), while the other uses Large Eddy Simulation (LES). The third case is called the floating body case. This is not a double-body, but a "single" body next to a simple free-slip approximated free surface boundary. The floating body case uses RANS.

Both the RANS and the LES approaches are based on decomposing the field variables. In RANS, variables are decomposed into a mean and a fluctuating part. In LES, they are decomposed into a filtered and a residual part based on a filter width. In short, RANS decomposition is based on statistical averaging, while LES decomposition is based on spacial filtering. LES is generally much more detailed, and requires a much finer grid. RANS is relatively simple, and much faster to run. Increased accuracy from LES must be paid for in increased computational effort.

For both RANS cases, the grid used was the same (only doubled in the double-body case). Wall functions were applied to reduce cell count near the wall and hence simulation time. The LES grid was much finer, as is required when the whole boundary layer is to be resolved. Applied turbulence models were the realizable k-epsilon model (RANS) and the Smagorinsky sub-grid scale model (LES). These models proved to be well suited to describe the flow in question.

The results showed some differences between the cases. One difference was higher force coefficients in the floating body case than in the double-body case. This was likely caused by the restrictive free-slip free surface boundary in the floating body case. A grid sensitivity study for the double-body RANS case showed that the applied grid is fine enough for this application.

Due to limited time and computational resources, the LES case could not be run long enough to achieve satisfying statistical convergence. Therefore, the comparison between RANS and LES in this project has some shortcomings. It was, however possible to draw some conclusions from it. LES is clearly better at capturing the vortex shedding, and maintaining it further downstream. The RANS simulations also captured vortex shedding, but not as accurately. Another result of the comparison was that RANS seemed to under predict the drag coefficient.

Sammendrag

Hydrodynamisk strømning rundt en forenklet skrogform har blitt undersøkt ved hjelp av numeriske simuleringer i OpenFOAM. Strømningen er tredimensjonal og turbulent, med et Reynolds tall på 26.400. Basisgeometrien er en rektangulær sylinder (eller boks) med aspektforhold på 5:1 og strøm i langsgående (x-)retning (dimensjonene er 1 på tvers av strømmen og 5 på langs). Høyden på legemet er 1 i z-retningen, slik at det blir kvadratisk i y-z planet. Simuleringene har blitt kjørt i parallell på superdatamaskinen Vilje på NTNU.

Tre forskjellige caser har blitt undersøkt. To av casene omhandler såkalt *double-body* strømning. Dette betyr at basisgeometrien er "doblet" (speilet over x-y planet), og fullt neddykket med fluid på alle sider. Forskjellen mellom disse to casene er turbulensmodelleringen; én bruker *Raynolds Averaged Navier-Stokes* (RANS), mens den andre bruker *Large Eddy Simulation* (LES). Den tredje casen kalles flytende legeme. Dette er ikke et *double-body*, men et "enkelt" legeme inntil en fri-slipp forenklet fri overflate grensebetingelse. Flytende legeme-casen bruker RANS.

Både RANS og LES er basert på å dekomponere feltvariablene. I RANS blir variablene dekomponert til en midlere og en fluktuerende del. I LES blir de dekomponert til en filtrert del og en rest basert på en filterbredde. Kort sagt er dekomponeringen i RANS basert på å ta et statistisk gjennomsnitt, mens dekomponeringen i LES er basert på filtrering i rommet. LES er generelt mye mer detaljert, og behøver mye finere grid. RANS er relativt enkelt, og mye raskere å kjøre. Økt nøyaktighet fra LES må betales i økt antall prosessortimer i simuleringen.

Griddet var det samme for begge RANS-casene (bare doblet for *double-body* casen). Veggfunksjoner ble brukt for å redusere antall celler nær veggen og dermed redusere simuleringstiden. LES-griddet var mye finere. Det trengs når hele grensesjiktet skal løses. Turbulensmodellene som ble brukt var *realizable* k-epsilon modellen (RANS) og *Smagorinsky sub-grid scale* modellen (LES). Disse modellene viste seg å passe bra for å beskrive strømningen her.

Resultatene viste en del forskjeller mellom casene. Én forskjell var høyere kraftkoeffisienter for det flytende legemet enn for *double-body*. Dette ble sannsynligvis forårsaket av den restriktive fri-slipp fri overflate grensebetingelsen i flytende legeme-casen. En sensitivitetsstudie for *double-body* RANS-griddet viste at det var fint nok for disse simuleringene.

Grunnet begrenset tid og kapasitet på datamaskinen kunne ikke LES-casen bli kjørt lenge nok til å nå tilfredsstillende statistisk konvergens. Derfor har sammenlikningen mellom RANS og LES i dette prosjektet noen begrensninger. Det var likevel mulig å dra noen konklusjoner fra den. LES er klart bedre til å fange opp virvelavløsningen og følge den nedstrøms. RANS simuleringene fanget også opp virvelavløsningen, men ikke like nøyaktig. Et annet resultat av sammenlikningen var at RANS så ut til å underpredikere drag-koeffisienten.

Contents

List of Tables

List of Figures

List of Abbreviations

CFD: Computational Fluid Dynamics

RANS: Reynolds-Averaged Navier-Stokes

LES: Large Eddy Simulation

DNS: Direct Numerical Simulation

2-D: Two-dimensional

3-D: Three-dimensional

SST: Shear-Stress Transport

SGS: Sub-Grid Scale

FVM: Finite Volume Method

FDM: Finite Difference Method

FEM: Finite Element Method

CAD: Computer Aided Design

GAMG: Geometric Algebraic Multi-Grid

P(Bi)CG: Preconditioned (Bi-)Conjugate Gradient

CFL: Courant-Friedrich-Lewy number

Re: Reynolds number

List of Symbols

Item	Unit	Description
p	$[\frac{N}{m^2}]$	Pressure
t	[s]	Time
u	$[\frac{m}{s}]$	Velocity x-dir
v	$[\frac{m}{s}]$	Velocity y-dir
w	$[\frac{m}{s}]$	Velocity z-dir
i,j		Subscripts for direction (i=1,2 or 3 corresponds to x,y, or z direction) Example: u_i for velocity in all three directions
μ	$[\frac{N\cdot s}{m^2}]$	Dynamic viscosity
ν	$[\frac{m^2}{s}]$	Kinematic viscosity
ρ	$[\frac{kg}{m^3}]$	Density
σ	$[\frac{N}{m^2}]$	Stress tensor
s	$[\frac{1}{s}]$	Rate of strain tensor
δ_{ij}	[-]	Kronecker delta function
τ_{ij}	$[\frac{kg}{m\cdot s^2}]$	Reynolds stress
k	$[\frac{m^2}{s^2}]$	Turbulent kinetic energy
ν_T	$[\frac{m^2}{s}]$	Eddy viscosity/turbulent viscosity
P_k	$[\frac{m^2}{s^3}]$	Production of mean turbulent kinetic energy
D_k	$[\frac{m^2}{s^3}]$	Diffusive transport of mean turbulent kinetic energy
ε	$[\frac{m^2}{s^3}]$	Viscous dissipation of mean turbulent kinetic energy
C_μ	[-]	Constant in the mixing length model
σ_k	[-]	Constant
$C_{\varepsilon 1}$	[-]	Constant in the $k-\varepsilon$ model
$C_{\varepsilon 2}$	[-]	Constant in the $k-\varepsilon$ model
C_D	[-]	Constant in the $k-\varepsilon$ model
C'_μ	[-]	Constant in the $k-\varepsilon$ model
σ_ε	[-]	Constant in the $k-\varepsilon$ model
ω	[]	Specific turbulent dissipation rate/ turbulent vorticity/ turbulence frequency
β^*	[-]	Constant in the $k-\omega$ model
σ_{k1}	[-]	Constant in the $k-\omega$ model
γ_1	[-]	Constant in the $k-\omega$ model
β_1	[-]	Constant in the $k-\omega$ model
$\sigma_{\omega 1}$	[-]	Constant in the $k-\omega$ model
γ	[-]	Constant in the $k-\omega$ SST model
β	[-]	Constant in the $k-\omega$ SST model
σ_ω	[-]	Constant in the $k-\omega$ SST model
$\sigma_{\omega 2}$	[-]	Constant in the $k-\omega$ SST model
u_*	$[\frac{m}{s}]$	Friction velocity
τ	$[\frac{kg}{m\cdot s^2}]$	Shear stress
u^+	[-]	Non-dimensional velocity
y^+	[-]	Non-dimensional wall-distance
κ	[-]	Constant
$\overline{U}$	$[\frac{m}{s}]$	Filtered velocity

$u^{'}$	$[\frac{m}{s}]$	Residual velocity
G	[-]	Filter function
Δ	[m]	Filter width
$\overline{p}$	$[\frac{N}{m^2}]$	Filtered pressure
τ_{ij}^R	$[\frac{m^2}{s^2}]$	Residual stress tensor
k_r	$[\frac{m^2}{s^2}]$	Residual kinetic energy
τ_{ij}^r	$[\frac{m^2}{s^2}]$	Anisotropic residual stress tensor
$\overline{p}_m$	$[\frac{N}{m^2}]$	Modified filtered pressure
ν_r/ν_{SGS}	$[\frac{m^2}{s}]$	Residual eddy viscosity
l_S	[m]	Smagorinsky length scale
C_S	[-]	Constant in the Smagorinsky model
A^+	[-]	Constant in the Smagorinsky model
F		General field variable
$\vec{n}$		Normal vector
u_n	$[\frac{m}{s}]$	Normal velocity
S		Surface (FVM related)
V		Volume (FVM related)
Δt	[s]	Time step
Δx	[m]	Cell size
Δs	[m]	Cell size near the wall

1 Introduction

1.1 Background and Motivation

The role of Computational Fluid Dynamics (CFD) is becoming more important in both research and commercial activities. This is to a large extent due to the rapid evolution of more powerful computers in the last decades. As more computational power becomes available to researchers and engineers, the more attractive it becomes to take advantage of numerical methods in hydrodynamics. Today, CFD is being used for a wide variety of applications. In hydrodynamic research, numerical methods are often used to complement experiments for validation purposes. They are also used without experiments in cases where experiments are too difficult and/or expensive. In the industry, CFD is used for a wide variety of hydrodynamic applications, from estimation of ship resistance to detailed analysis of flow around various specialized structures, such as sub-sea modules. It should also be mentioned that CFD is being used in many different areas; not just in hydrodynamics.

Reynolds-Averaged Navier-Stokes (RANS) methods have for a long time been the most widely used approach for simulating turbulent flow. This is due to its low computational cost and reasonable accuracy for many applications. With improvements in computational power, other more advanced methods, such as Large Eddy Simulations (LES), are becoming increasingly attractive. It can be expected that this development will continue in the future. Therefore it is interesting to see how the different methods can be compared in terms of applicability, accuracy and computational cost for various hydrodynamic applications.

This master's thesis is designed to be an introduction to the world of CFD in hydrodynamics. It consists of a thorough introduction to the background theory of turbulent flows, a practical introduction to the computational methods used, and results of simulations of flow around simple geometries using different turbulence modelling techniques.

There were two main motivations behind this thesis. The first was to make efforts to be able to simulate flow around a ship hull. The second was to compare different turbulence modelling techniques, preferably both RANS and LES. The ship hull case is interesting because it is practically significant in the industry, and because it would give great experience in modelling flow around complicated geometries. The comparison of turbulence models is interesting because it would give a better understanding of both theory and practice regarding the many different models. This is something that is always of interest in CFD, since choice of turbulence model can affect results significantly.

As modelling flow around a ship hull proved to be too time consuming within the scope of this project, it was decided to use simplified geometries instead, and to focus on comparing different turbulence models and their applicability in this case. The simplified geometry used was a rectangular cylinder (or box), in both a fully submerged case and a floating case. The results of interest are the flow field in general and the forces on the body. All simulations were run on the supercomputer Vilje at NTNU.

1.2 Previous Work

Previous to the work presented in this master's thesis, the author has completed simulations of flow around even simpler geometries. This previous work served as a background for what will be presented later in this thesis. The simulations in question were concerned with flow around rectangular cylinders spanning the whole fluid domain, first in 2-D and then in 3-D. This was a simpler case, both in terms of flow complexity and pre-processing. These simulations were compared to results found in the literature. The 2-D results were compared to the study of Mannini et al. (2010), and the 3-D results were compared to the studies of Mannini et al. (2011) and Arslan et al. (2011). As the results corresponded quite well, the general approach was adopted into the simulations in this master's project, which have not been compared to results from the literature. This is because the author was unable to find suitable comparison studies. Hence, the previous comparison studies still have some relevance for validation of the results of this master's thesis. The work of Rodi (1997) has been used as a reference for the comparison between RANS and LES.

1.3 Outline of the Thesis

Chapter 2 is a review of the theory behind the applied computational methods. This includes the governing equations, a description of turbulence, and turbulence modelling in both RANS and LES. A short description of the finite volume method is also included. Chapter 3 is a description of the CFD software OpenFOAM. This includes both its background and practical use. Chapter 4 is an introduction to pre-processing in CFD. It is mostly concerned with grid generation. The remaining chapters contain the results of the numerical simulations, discussion of these results and finally some conclusions based on this.

2 Background Theory

2.1 Fundamental Equations of Viscous Flow

Numerical simulation of fluid flow is based on solving the fundamental equations of fluid mechanics. These are conservation of mass (the continuity equation), conservation of momentum (the Navier-Stokes equations), and conservation of energy. It is rarely necessary to solve the energy equation in marine hydrodynamics, since temperature can be assumed constant.

In fluid dynamics, the Eulerian form of motion is usually used to describe the flow field. This means that we look at the flow at every fixed point in the fluid as a function of time. In practice, the fluid domain is divided into control volumes, and the equations are solved in each of these. The alternative to Eulerian is Lagrangian. With the Lagrangian form of motion, one follows the trajectories of each individual particle in the flow. The common form of the conservation equations are Lagrangian in nature. This results in the so-called particle derivative when we want to describe the flow field using Eulerian formulation and the conservation equations (White, 2006).

2.1.1 The Continuity Equation

The continuity equation expresses conservation of mass in the entire fluid domain. In cartesian coordinates, the general compressible form is:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} + \frac{\partial (\rho w)}{\partial z} = 0 \tag{2.1}$$

In this project, only incompressible flow is considered. Then, the fluid density ρ can be omitted from the equation:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.2}$$

This can also be written as:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.3}$$

Here, i can be 1, 2 and 3, representing the x-, y- and z-directions respectively. When i is present more than once in a component of a formula, like in equation 2.3, all three components are added together. This means that $\frac{\partial u_i}{\partial x_i} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$, or that equation 2.2 is equivalent to equation 2.3. This notation, with both i and j, will be used throughout this report.

2.1.2 The Navier-Stokes Equations

The Navier-Stokes equations are derived from Newton's second law of motion for a fluid. Written on vector form:

$$\rho\frac{d\vec{u}}{dt} = -\nabla p + \mu\nabla^2\vec{u} \tag{2.4}$$

Here, $\frac{d}{dt}$ is the so-called particle derivative. It includes derivatives in both time and space; $\frac{d}{dt} = \frac{\partial}{\partial t} + (\vec{u} * \nabla)$. The background for this is that the conservation laws are Lagrangian, i.e they apply to fixed systems, or particles (White, 2006). Written on componential form, equation 2.4 becomes:

$$\rho(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}) = \frac{\partial}{\partial x_j}(\sigma_{ij}) \tag{2.5}$$

or:

$$\rho\frac{du_i}{dt} = -\frac{\partial p}{\partial x_i} + \mu\frac{\partial^2 u_i}{\partial x_i^2} \tag{2.6}$$

Here, $\sigma_{ij} = -p\delta_{ij} + 2\mu S_{ij}$ is the stress tensor, and $S_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$ is the rate of strain tensor. δ_{ij} is the Kronecker delta function which is equal to 1 for i=j and 0 otherwise. μ is the dynamic viscosity of the fluid. The assumptions behind these formulas are incompressibility, Newtonian fluid and constant fluid properties. Also, gravity forces have been neglected, which is reasonable when dealing with plane flows. Newtonian fluid means that the viscous stresses are linearly related to the strain-rate (White, 2006). The first term on the left side of equation 2.5 is the time derivative of velocity, which will go away if we assume steady flow. The second term on the left is the convection term, representing the spatial derivatives of velocity. The term on the right of equation 2.5 can best be seen by rewriting into the two terms seen on the right in equation 2.6. The first of these two terms is the pressure term, and the second is the viscous stress term. This last term is essential when dealing with turbulent flows.

2.1.3 Boundary Conditions

To solve the fundamental equations above, which are partial differential equations, boundary conditions are necessary. There are essentially two main categories of the most commonly used boundary conditions for a fluid, Dirichlet and Neumann conditions. A Dirichlet condition specifies the value of a variable at the boundary, while a Neumann condition specifies the derivative of the variable. Both are frequently used in CFD. Initial conditions for a CFD problem is set to specified values for the whole fluid domain. With the proper conservation equations, and boundary and initial conditions, the system is solvable. There is, however one phenomena that greatly complicates matters when it comes to fluid dynamics, and that is turbulence.

2.2 Principles of Turbulence

Turbulence is a phenomenon that is easy to recognize, but very difficult to define. Essentially, turbulence is the apparently random behaviour of the flow that causes it to deviate from its mean. The opposite of turbulent is laminar. Laminar flow is smooth and ordered, while turbulent flow is chaotic and random-looking.

When doing mathematical computations with turbulence, some characteristic properties are especially important to be aware of. First of all, turbulence occurs at high Reynolds numbers. At moderate Reynolds numbers, there is a transition between laminar and turbulent flow. The nature of the flow is very dependent on where the flow can be placed on the laminar-to-turbulent scale. At what Reynolds number a specific flow is fully turbulent depends on its nature. This paper deals with flow past a rectangular cylinder. For flow past a circular cylinder, fully turbulent flow occurs for $Re > 3.5 * 10^6$ (Pettersen, 2007).

Another important aspect of turbulence is its diffusivity. Turbulence causes increased mixing of the fluid and generates momentum, heat and mass transfer (Tennekes and Lumley, 1972). This is connected to the dissipative nature of turbulent flows. The turbulent kinetic energy is gradually transformed into heat. Hence, the turbulence will eventually fade out if there is not a supply of turbulent energy.

Turbulence is always a three-dimensional phenomenon. This is connected to its rotational nature. Random vorticity fluctuations are important characterizing aspects of turbulence, and cannot maintain themselves in two-dimensional flow since the important mechanism of vortex stretching is not present there (Tennekes and Lumley, 1972). This means that simulation of turbulent flow in 2-D has an important shortcoming, and is never fully correct physically.

2.3 Reynolds Averaging

In the context of numerical fluid dynamics, it is of course possible to solve the full Navier-Stokes equations for the entire flow field. This is called direct numerical simulation (DNS), and is a very demanding process. To do this, one needs very powerful computers and a lot of time and patience. In practice, modelling of turbulence is therefore an important field of study. This can save a lot of time and resources related to the simulations. Simple simulations of flow with modelled turbulence can be run even on home computers.

To begin modelling of turbulence, one can look at its most important characteristics; the flow variables are fluctuating around a mean. This means that one can separate the variables, like velocity, into a mean and a fluctuating component. The Reynolds Averaged Navier-Stokes (RANS) equations are deduced by two mathematical operations; Reynolds decomposition and averaging. A general reference for this section can be made to chapter 2 of Tennekes and Lumley (1972).

If absolute velocity is denoted $\hat{u_i}$, the Reynolds decomposition will be:

$$\hat{u_i} = U_i + u_i \tag{2.7}$$

Here, U_i is the mean component and u_i is the fluctuating component. We can introduce this in the incompressible continuity equation (equation 2.3), resulting in equation 2.8.

$$\frac{\partial U_i}{\partial x_i} + \frac{\partial u_i}{\partial x_i} = 0 \tag{2.8}$$

Looking at equation 2.3 we can say that the mean flow is incompressible:

$$\frac{\partial U_i}{\partial x_i} = 0 \tag{2.9}$$

And hence that the fluctuations must also be incompressible:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.10}$$

As for velocity, the corresponding decomposition for pressure will be: $\hat{p} = P_i + p_i$. Also, we can write $\hat{\sigma_{ij}} = \Sigma_{ij} + \sigma_{ij}$ and $\hat{S_{ij}} = S_{ij} + s_{ij}$, decomposing stress and strain tensor, respectively. If these decompositions are introduced in the Navier-Stokes equations without any further simplification, we get:

$$\frac{\partial U_i}{\partial t} + \frac{\partial u_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} + u_j \frac{\partial U_i}{\partial x_j} + U_j \frac{\partial u_i}{\partial x_j} + u_j \frac{\partial u_i}{\partial x_j} = \frac{1}{\rho}\left(\frac{\partial}{\partial x_j}\Sigma_{ij} + \frac{\partial}{\partial x_j}\sigma_{ij}\right) \tag{2.11}$$

The equation is then simplified by taking the time average of all terms. We then assume that all fluctuating components oscillate around a zero mean, meaning that the average of all fluctuating components are zero. The only fluctuating part of equation 2.11 that will be retained is the last term on the left side. Since this is the product of two fluctuating components, we cannot say that it averages to zero. Averaging all terms in equation 2.11 reduces it to:

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} + \overline{u_j \frac{\partial u_i}{\partial x_j}} = \frac{1}{\rho}\frac{\partial}{\partial x_j}(\Sigma_{ij}) \tag{2.12}$$

The overlining denotes a time average. Looking at term number three on the left side of equation 2.12, which is the fluctuating term, we can decompose it into: $u_j \frac{\partial u_i}{\partial x_j} = \frac{\partial}{\partial x_j}(u_i u_j) - u_i \frac{\partial u_j}{\partial x_j}$ where the second term is zero due to continuity (equation 2.10). Hence, equation 2.12 reduces to equation 2.13, the unsteady RANS equation.

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = \frac{1}{\rho}\frac{\partial}{\partial x_j}(\Sigma_{ij} - \rho\overline{u_i u_j}) \tag{2.13}$$

The last term on the right side of equation 2.13 is the so-called Reynolds stress term, and is often denoted τ_{ij}. This is the only term that is not present in the Navier-Stokes equations for non-turbulent flow. From this it is clear that turbulence seems to increase the stresses in the flow.

2.4 Turbulence Modelling in RANS

Solving the general RANS equation is not straight forward as there are too many unknowns. To solve it, we will need more equations and assumptions. A general reference for this section can be made to chapter 3 of Tennekes and Lumley (1972).

To begin, we need a relation between the mean flow and the turbulent stresses. Such a relation was proposed by J. Boussinesq in 1877 and reads:

$$-\rho\overline{u_i u_j} = 2\rho\nu_T S_{ij} - \frac{2}{3}\rho k \delta_{ij} \tag{2.14}$$

The mean flow is here represented by $S_{ij} = \frac{1}{2}(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i})$. The variable k is the mean turbulent kinetic energy:

$$k = \frac{1}{2}\overline{u_i u_i} \tag{2.15}$$

The relation in equation 2.14 implies that the turbulence has a dissipative effect and that this dissipation can be modelled by a turbulent viscosity term ν_T (Layton, 2014), also called eddy viscosity. It should be noted that this turbulent viscosity is not a physical property, but an imaginary variable to help understand the turbulence concept. It is also a flow property, not a fluid property, and will vary in space and time from flow to flow. The Boussinesq relation is not generally correct (Layton, 2014), but works well as an approximation as it captures important physics of turbulence.

If we expand the unsteady RANS equation using $\Sigma_{ij} = -P\delta_{ij} + 2\mu S_{ij}$ we get:

$$\frac{\partial U_i}{\partial t} + U_j\frac{\partial U_i}{\partial x_j} = \frac{1}{\rho}\frac{\partial}{\partial x_j}(-P\delta_{ij} + 2\mu S_{ij} - \rho\overline{u_i u_j}) \tag{2.16}$$

Introducing the Boussinesq relation into equation 2.16 we get:

$$\frac{\partial U_i}{\partial t} + U_j\frac{\partial U_i}{\partial x_j} = -\frac{\partial}{\partial x_i}(\frac{P}{\rho} + \frac{2}{3}k) + \frac{\partial}{\partial x_j}(2(\nu + \nu_T)S_{ij}) \tag{2.17}$$

The system of equations would now be solvable if we had knowledge of the turbulent viscosity and mean turbulent kinetic energy. To accomplish this, we can introduce new equations for them.

A general transport equation for mean turbulent kinetic energy can be written as:

$$\frac{dk}{dt} = D_k + P_k - \varepsilon \tag{2.18}$$

P_k is production of mean turbulent kinetic energy. It can be written as:

$$P_k = -\overline{u_i u_j}\frac{\partial U_i}{\partial x_j} \tag{2.19}$$

This implies that production of turbulent kinetic energy comes from interaction between turbulence and the mean flow. Introducing Boussinesq (equation 2.14) we can write this as:

$$P_k = (2\nu_T S_{ij} - \frac{2}{3}\delta_{ij}k)\frac{\partial U_i}{\partial x_j} = 2\nu_T S_{ij}\frac{\partial U_i}{\partial x_j} - \frac{2}{3}\delta_{ij}k\frac{\partial U_i}{\partial x_j} \tag{2.20}$$

The second term on the right of equation 2.20 becomes zero using continuity of the mean flow (equation 2.9) and the fact that the Kronecker delta function is zero when $i \neq j$. Then, the production term reduces to:

$$P_k = 2\nu_T S_{ij} \frac{\partial U_i}{\partial x_j} \tag{2.21}$$

D_k is diffusive transport of mean turbulent kinetic energy. It can be written as:

$$D_k = \frac{\partial}{\partial x_i} \overline{(u_i(\frac{P}{\rho} + \frac{1}{2} u_i u_i))} + \frac{\partial}{\partial x_i}(2\nu \overline{u_j \Delta_{ij}}) \tag{2.22}$$

The second term can be neglected since the molecular viscosity ν is small. Using the gradient diffusion hypothesis $-\overline{u_i\theta} = \gamma_T \frac{\partial \Theta}{\partial x_i}$ where $\gamma_T = \frac{\nu_T}{\sigma_k}$ and σ_k is the turbulent diffusion number or Prandtl number, and neglecting the pressure term, equation 2.22 reduces to:

$$D_k = \frac{\partial}{\partial x_i}(\frac{\nu_T}{\sigma_k} \frac{\partial k}{\partial x_i}) \tag{2.23}$$

ε is the viscous dissipation of turbulent kinetic energy. It can be written as:

$$\varepsilon = 2\nu \overline{s_{ij} s_{ij}} \tag{2.24}$$

$s_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$ is the turbulent strain tensor.

Inserting the expressions for production, diffusion and dissipation into the transport equation for turbulent kinetic energy (equation 2.18), it becomes:

$$\frac{dk}{dt} = 2\nu_T S_{ij} \frac{\partial U_i}{\partial x_j} + \frac{\partial}{\partial x_i}(\frac{\nu_T}{\sigma_k} \frac{\partial k}{\partial x_i}) - 2\nu \overline{s_{ij} s_{ij}} \tag{2.25}$$

Now, the system of equations is solvable if ν_T can be specified. To find a relation for ν_T, the mixing length model can be used. This was introduced by Ludwig Prandtl in 1925. The mixing length model states that the turbulent viscosity is proportional to a turbulent velocity scale U and a turbulent length scale L. The underlying assumption is that the large scale turbulence is characterized by velocity and length only, not viscosity. This is reasonable away from the wall where turbulent effects dominate over viscous effects. In mathematical terms, the mixing length model is $\nu_T = C_\mu U L$, where C_μ is an empirical constant that can be determined from experiments. Then, if the turbulent velocity scale is assumed to be equal to the square root of mean turbulent kinetic energy, we can write $\nu_T = C_\mu \sqrt{k} L$. Now, the system is solvable if we can specify L. This is not an easy task, as it is not straight forward to find a length scale for turbulence. This is where the famous $k - \varepsilon$ and $k - \omega$ models come into play.

2.4.1 K-Epsilon Model

In the $k - \varepsilon$ model, a transport equation for turbulent viscous dissipation $\varepsilon = 2\nu\overline{s_{ij}s_{ij}}$ is introduced. The general form is the same as for the turbulent kinetic energy (equation 2.18). The transport equation for ε can be written as:

$$\frac{d\varepsilon}{dt} = \frac{\partial}{\partial x_i}(\frac{\nu_T}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial x_i}) + \frac{\varepsilon}{k}(C_{\varepsilon 1} P_k - C_{\varepsilon 2} \varepsilon) \tag{2.26}$$

The first term on the right is diffusion, the second is production and the third is dissipation. P_k is production of turbulent kinetic energy, and $C_{\varepsilon 1}$ and $C_{\varepsilon 2}$ are empirical constants. These constants introduce a simplification. To make the system solvable, we now need a relation between ε and the turbulent length scale described previously. If we again use the assumption that the large scales of turbulence are characterized by a velocity scale and a length scale only, the viscous dissipation will relate to these two scales as $\varepsilon = C_D \frac{k^{3/2}}{L}$, where C_D is an empirical constant. Solving this for L and inserting it into the mixing length model will result in $\nu_T = C_\mu C_D \frac{k^2}{\varepsilon}$. If we also use $C_\mu C_D = C'_\mu$, we can write:

$$\nu_T = C'_\mu \frac{k^2}{\varepsilon} \tag{2.27}$$

Using the transport equations for k and ε along with this relation, the system is now solvable when the constants σ_k, C'_μ, σ_ε, $C_{\varepsilon 1}$ and $C_{\varepsilon 2}$ are specified. It is common to apply standard values found from experimental results.

2.4.2 K-Omega Model

The principle behind the $k - \omega$ model (Wilcox, 1988) is the same as for the $k - \varepsilon$ model, except that a transport equation for turbulent vorticity, ω, is introduced instead of ε. The quantity ω is also frequently called specific turbulent dissipation rate or turbulence frequency. With ω instead of ε, the expression for the turbulent viscosity (equation 2.27) must be rewritten in the form $\nu_T \sim \frac{k}{\omega}$ to give the same dimension. The incompressible form of the transport equations for k and ω in the original $k - \omega$ model can be written as in equations 2.28 and 2.29 below (formulations taken from Menter (1993) and converted to incompressible form).

$$\frac{Dk}{Dt} = \tau_{ij} \frac{1}{\rho} \frac{\partial u_i}{\partial x_j} - \beta^* \omega k + \frac{\partial}{\partial x_j} [(\nu + \sigma_{k1} \nu_t) \frac{\partial k}{\partial x_j}] \tag{2.28}$$

$$\frac{D\omega}{Dt} = \frac{\gamma_1}{\rho \nu_t} \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta_1 \omega^2 + \frac{\partial}{\partial x_j} [(\nu + \sigma_{\omega 1} \nu_t) \frac{\partial \omega}{\partial x_j}] \tag{2.29}$$

The constants in these equations are $\beta^*, \sigma_{k1}, \gamma_1, \beta_1$ and $\sigma_{\omega 1}$. These are referred to by Wilcox (1988) as closure coefficients, as they must be determined before the system of equations can be closed.

2.4.3 K-Omega SST Model

The $k - \omega$ *SST* model was proposed by Menter (1993). It uses a blending function to switch between the $k - \omega$ model in the boundary layer and the $k - \varepsilon$ model in the free stream. In addition, the $k - \omega$ *SST* model accounts for transport of turbulent shear stress in the boundary layer by reformulating the expression for ν_T to include the shear stress (Menter, 1993). The abbreviation SST comes from this shear stress transport.

The transport equations for the turbulent quantities in the SST model are derived by combining the original $k-\omega$ model (equations 2.28 and 2.29) with a transformed $k-\epsilon$ model. This transformed model is obtained by transforming the standard $k-\epsilon$ into a $k-\omega$ formulation. This formulation will have a cross-diffusion term that is not present in the original $k-\omega$ model (Menter, 1993). These two models are combined by using the blending function F_1. The purpose of this blending function is to effectively switch between the $k-\omega$ model in the near-wall region and the transformed $k-\epsilon$ model in the free stream. The incompressible formulation of Menter's model can be seen in equations 2.30 and 2.31 below.

$$\frac{Dk}{Dt} = \tau_{ij}\frac{1}{\rho}\frac{\partial u_i}{\partial x_j} - \beta^*\omega k + \frac{\partial}{\partial x_j}[(\nu + \sigma_k \nu_t)\frac{\partial k}{\partial x_j}] \tag{2.30}$$

$$\frac{D\omega}{Dt} = \frac{\gamma}{\rho\nu_t}\tau_{ij}\frac{\partial u_i}{\partial x_j} - \beta\omega^2 + \frac{\partial}{\partial x_j}[(\nu + \sigma_\omega \nu_t)\frac{\partial \omega}{\partial x_j}] + 2(1-F_1)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{2.31}$$

Only the transport equation for ω is different from the original $k-\omega$ model. It can be seen from equation 2.31 that the blending function should be equal to one near the wall, and go to zero in the free stream.

2.4.4 Realizable K-Epsilon Model

An improvement of the original $k-\varepsilon$ model was proposed by Shih et al. (1995). The idea behind this model is to reformulate the equation for the dissipation rate ε (equation 2.26), and the eddy viscosity ν_T (equation 2.27). The new transport equation for ε is based on the mean-square vorticity fluctuations at high Reynolds numbers. It is obtained by modelling a dynamic equation for the mean-square turbulent vorticity $\overline{\omega_i\omega_i}$, and using the relation $\varepsilon = \nu\overline{\omega_i\omega_i}$, which is valid at large Reynolds numbers (Shih et al., 1995). The expression for eddy viscosity is reformulated by making C'_μ a variable instead of a constant, hence making it "realizable".

2.4.5 Pros and Cons of the Models

All RANS turbulence models have their strong and weak sides. The $k-\varepsilon$ model is reported to have a lack of sensitivity to adverse pressure gradients, which can be a disturbing shortcoming. This makes the model overpredict the shear-stress levels near the wall, and can hence prevent or delay separation (Menter, 1993). Shih et al. (1995) claims that the original $k-\varepsilon$ model *"performs quite well for boundary layer flows but not for flows with a high mean shear rate or massive separation"*. This means that the $k-\varepsilon$ model may not be a good choice for all wall flows, and especially separated flows.

The $k-\omega$ model is designed to overcome the problems of the $k-\varepsilon$ model. It has a significantly better sensitivity to adverse pressure gradients, making it a better model for separated wall-flows. One problem with the $k-\omega$ model is that the results strongly depend on the chosen free-stream values of ω (Menter, 1993). This shortcoming is not present in the $k-\varepsilon$ model. The $k-\omega$ SST model will ideally overcome the shortcomings of both the original $k-\varepsilon$ and $k-\omega$ models, since it uses $k-\omega$ near the wall and $k-\varepsilon$ in the free-stream.

A general issue with all $k-\omega$ based models is the solid wall boundary condition of ω. In theory, the value of ω at the wall goes to infinity, which is of course impossible to impose in practice. This is discussed by e.g., Eça and Hoekstra (2004). Different empirical formulas are used to prescribe finite values of ω at wall boundaries.

The realizable $k-\varepsilon$ model performs better than the original $k-\varepsilon$ model in various flow types including boundary layer flows. However it is not intended to be applied very close to the wall (Shih et al., 1995). For separating boundary layer flows, the $k-\omega$ SST model might be a better choice. A way of avoiding the problems of $k-\varepsilon$ models in the near-wall region is to apply wall functions (will be explained later), as applied by Shih et al. (1995).

2.5 Turbulent Boundary Layers

Turbulent flows around solid bodies are very dependent on the behavior of the boundary layer. It is therefore important to have some insight into the physics of turbulent boundary layers when choosing how to model the turbulence. For a practical introduction to this concept, see e.g., Bredberg (2000).

The boundary layer can be divided into certain layers, where different physical effects are dominating. To understand why this is the case, we can start by defining the friction velocity:

$$u_* = \sqrt{\frac{\tau_w}{\rho}} \tag{2.32}$$

Here, τ_w is the shear stress at the wall. For a two-dimensional case with mean flow U in x-direction, the shear stress close to the wall can be approximated as:

$$\tau = \mu\frac{\partial U}{\partial y} - \rho\overline{uv} \tag{2.33}$$

The assumption behind this is that the y-dependence of the mean velocity profile dominates over its x-dependence.

To describe the velocity profile in the boundary layer, we can define dimensionless variables for velocity and length. Close to a wall, we can assume that the mean velocity is only dependent on the friction velocity u_*, the distance y from the wall perpendicular to the mean flow, and the kinematic viscosity ν. The non-dimensional variables are then:

$$u^+ = \frac{U}{u_*} \tag{2.34}$$

$$y^+ = y\frac{u_*}{\nu} \tag{2.35}$$

Very close to the wall, the velocity fluctuations u and v will be reduced to zero due to the no-slip condition. Then, the shear stress becomes $\tau = \mu\frac{\partial U}{\partial y}$. Using equation 2.32, we can then write $\mu\frac{\partial U}{\partial y} = \rho u_*^2$, integrate this to $U = \frac{u_*^2}{\nu}y$, and reformulate into $\frac{U}{u_*} = \frac{u_* y}{\nu}$. Using the non-dimensional variables, we can write this as:

$$u^+ = y^+ \tag{2.36}$$

Equation 2.36 is valid in the region of the boundary layer where viscous effects are dominating. It is commonly called the linear or the viscous sublayer.

Further away from the wall, the Reynolds stress $-\rho\overline{uv}$ dominates over the viscous stress. If we now assume that the mean velocity is only dependent on the friction velocity u_* and the distance y from the wall, and not on the viscosity, we can approximate the velocity gradient as $\frac{dU}{dy} = c\frac{u_*}{y}$. The constant c is commonly written as $\frac{1}{\kappa}$. If we integrate this expression in y and use the non-dimensional variables, we get:

$$u^+ = \frac{1}{\kappa}ln(y^+) + constant \tag{2.37}$$

Equation 2.37 is valid in the region of the boundary layer where Reynolds stresses are dominating. This region is called the logarithmic layer or the log-layer. Between the viscous layer and the log-layer, there will be a buffer layer where both viscous and turbulent effects are significant. References to these derivations can be made to chapter 5.2 of Tennekes and Lumley (1972) and chapter 3.18 of Newman (1977).

A graphical representation of the boundary layer can be seen in figure 2.1. As seen in the figure, the logarithmic layer is also frequently called the inertial sublayer. The transition between the viscous and the logarithmic layer can be seen to be somewhere around a y^+ range of 5 -30.

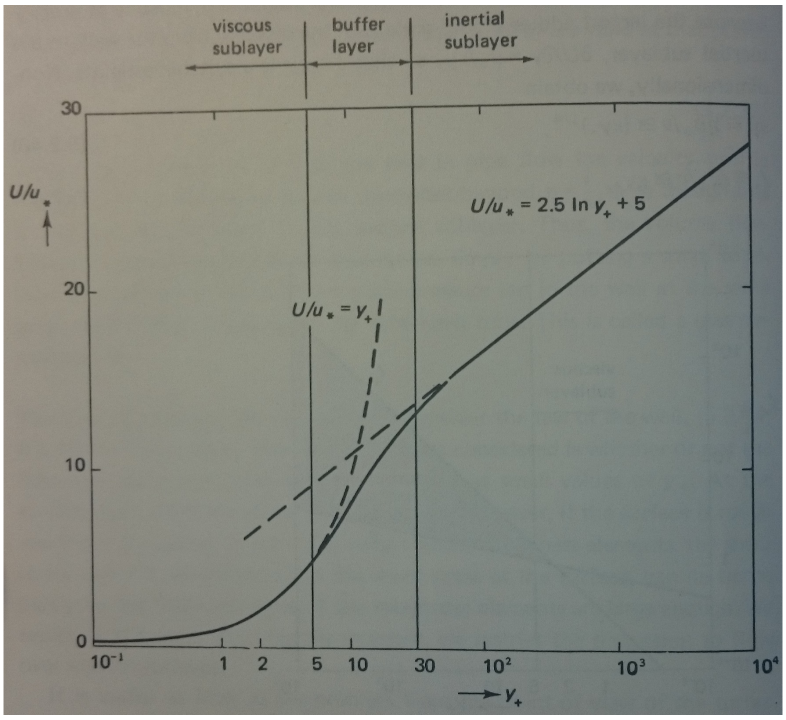

Figure 2.1: The law of the wall, turbulent boundary layer divided into sublayers, source: Tennekes and Lumley (1972)

2.6 Large Eddy Simulation

The gap between RANS methods and Direct Numerical Simulations (DNS) is potentially very large when it comes to accuracy, reliability and computational effort. The turbulence modelling in RANS methods are somewhat crude simplifications that can save a lot of time in the simulations. However, the simplifications are not always accurate, or reliable. There are many situations where RANS methods simply are unable to describe the flow accurately enough. Highly separated flow with large adverse pressure gradients are a documented problem for several RANS turbulence models, as reported by e.g., Wilcox (1988) and (Menter, 1993). In these cases, a DNS might be the best approach. However, DNS can be very demanding, since all of the flow, including the smallest scale motions, have to be resolved. Turbulent flow contains a large range of motion scales, resulting in a lot of computational effort in a DNS.

One way to close the gap between RANS and DNS is to use Large Eddy Simulation (LES). Essentially, LES is like DNS, but with less of the motion scales directly resolved. The idea behind LES is to use DNS for the large scales of motion (the large eddies), and model the small scale motion. This potentially saves a lot of unnecessary computational effort since the small scales are usually not as important as the large scales (Sagaut, 2006).

The mathematical background of LES and its turbulence modelling is described by e.g., Pope (2000), Sagaut (2006) and John (2004).

2.6.1 Filtering

To understand how LES works, one has to start with the concept of filtering. The scales of the flow has to be filtered, so that the simulation can know what is to be resolved and what is to be modelled. It is this filtering that allows LES to have a high degree of accuracy compared to RANS, while saving a considerable amount of computational effort compared to a full DNS. The concept of filtering in fluid mechanics is quite similar to the concept of Reynolds decomposition. Like in the RANS equations, the velocity is decomposed into two terms:

$$U(x,t) = \overline{U}(x,t) + u'(x,t) \tag{2.38}$$

Equation 2.38 is the notation for one dimension. The concept is the same in three dimensions. The first term $\overline{U}$ denotes the filtered velocity. This is the term that is resolved in LES. The second term u' is called the residual term. This term is not resolved, but modelled.

The filtering operation can in general be written (in one dimension) as seen in equation 2.39:

$$\overline{U}(x,t) = \int G(r,x)U(x-r,t)dr \tag{2.39}$$

The function G is the so-called filter function, and the integration is over the entire fluid domain. The filter function integrates to unity over the domain:

$$\int G(r,x)dr = 1 \tag{2.40}$$

There are many different types of filters that can be applied in LES. A graphic representation of the filtering of the velocity field can be seen in figure 2.2. The bold line is the filtered velocity. It is essentially a local average of the complete velocity field. The concept is the same for filtered pressure. All filters can be associated with a filter width Δ. Essentially, LES filtering will treat all eddies larger than Δ as large eddies, and all eddies smaller than Δ as small eddies (Ochoa and Fueyo, 2015).

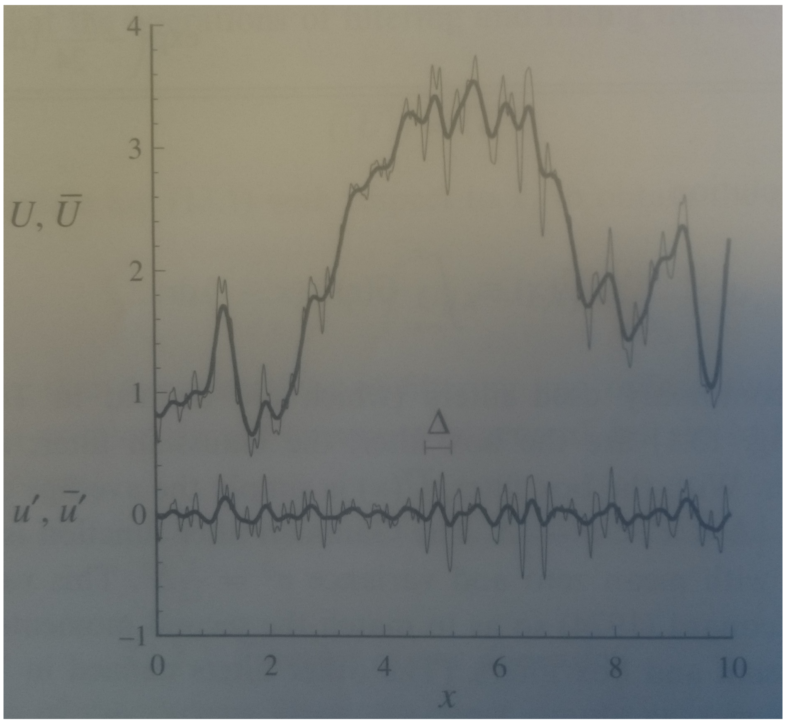

Figure 2.2: Graphic demonstration of filtering in LES. Bold line: filtered quantity. Source: Pope (2000)

It is common to use a filter width in the same order of magnitude as the cell size. This essentially means that the motion scales smaller than the cells will not be resolved. One simple way to apply this in practice is to take the cubic root of the cell volume as the Δ. Whether this method is good or not, depends on the nature of the flow and the grid size. According to Pope (2000), key mechanisms in the flow is present in the very close vicinity of a wall (y^+ values less than 20). This means that if the grid is too coarse, the filtering will not capture these mechanisms. The results is that LES grids have to be very fine close to walls (y^+ close to unity is a commonly used guideline). A way to improve the near wall modelling in LES is to use other types of Δ functions. One popular option is to use Van Driest damping, which reduces the filter width close to the wall to increase resolution. Both the cubic root of cell volume and the Van Driest approaches, along with several others, can be applied in OpenFOAM.

2.6.2 Filtered Conservation Equations

When the filtering described above is applied to velocity and pressure and introduced into the conservation equations, it results in the next three equations.

$$\frac{\partial \overline{U}_i}{\partial x_i} = 0 \tag{2.41}$$

$$\frac{\partial u_i'}{\partial x_i} = 0 \tag{2.42}$$

Equation 2.41 is the filtered continuity equation. Both the filtered field and the residual field are incompressible, seen from equations 2.41 and 2.42.

$$\frac{\partial \overline{U}_j}{\partial t} + \frac{\partial \overline{U_i U_j}}{\partial x_i} = \nu \frac{\partial^2 \overline{U}_j}{\partial x_i \partial x_i} - \frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_j} \tag{2.43}$$

Equation 2.43 is the filtered momentum equation. The difference between this and the RANS equation is the filtered product term $\overline{U_i U_j}$. This is not the same as the product of the filtered velocities $\overline{U}_i \overline{U}_j$ (Pope, 2000) in the RANS equation (equation 2.13). The difference between these terms is called the residual stress tensor τ_{ij}^R.

$$\tau_{ij}^R = \overline{U_i U_j} - \overline{U}_i \overline{U}_j \tag{2.44}$$

The following three definitions are used for rewriting the filtered momentum equation (Pope, 2000).

Residual Kinetic energy:

$$k_r \equiv \frac{1}{2}\tau_{ii}^R \tag{2.45}$$

Anisotropic residual stress tensor:

$$\tau_{ij}^r \equiv \tau_{ij}^R - \frac{2}{3}k_r \delta_{ij} \tag{2.46}$$

Modified filtered pressure:

$$\overline{p}_m \equiv \overline{p} + \frac{2}{3}k_r \tag{2.47}$$

Using the definitions of equations 2.45, 2.46 and 2.47, the momentum equation becomes:

$$\frac{\overline{D}\overline{U}_j}{\overline{D}t} = \nu \frac{\partial^2 \overline{U}_j}{\partial x_i \partial x_i} - \frac{\partial \tau_{ij}^r}{\partial x_i} - \frac{1}{\rho}\frac{\partial \overline{p}_m}{\partial x_j} \tag{2.48}$$

This equation is closed by modelling the anisotropic residual-stress tensor τ_{ij}^r, which is commonly called the subgrid scale (SGS) stress tensor. There are many ways of doing this in practice. The model proposed by Smagorinsky (1963) is the simplest one (Pope, 2000), (John, 2004).

2.6.3 Smagorinsky Subgrid Scale Model

The Smagorinsky SGS model is based on the simple eddy-viscosity model of Boussinesq described previously in chapter 2.4. In this case it can be written as:

$$\tau_{ij}^r = -2\nu_r \overline{S}_{ij} \tag{2.49}$$

Here, ν_r is the eddy-viscosity of the residual motions (Pope, 2000), also called the subgrid scale viscosity (sometimes denoted ν_{SGS}). This eddy viscosity is modelled by relating it to the so-called characteristic filtered rate of strain $\overline{S}$, through the Smagorinsky length scale l_s.

$$\overline{S} = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \tag{2.50}$$

$$l_s = C_S\Delta \tag{2.51}$$

$$\nu_r = l_s^2\overline{S} = (C_S\Delta)^2\overline{S} \tag{2.52}$$

Equation 2.52 closes the system of equations if the length scale can be specified. To specify the length scale, it is related to the filter width Δ and a constant C_S. The underlying assumption is that the filter width is proportional to the Smagorinsky length scale. The relation in equation 2.52 is essentially a mixing length model, with l_s analogous to the mixing length of Prandtl (Pope, 2000).

The formulation in equation 2.52 is for a simple type of delta function as described previously in this chapter (cubic root of volume). If one wants to use other more advanced functions, the equation will be different. Introducing Van Driest damping, for example, results in equation 2.53 for the Smagorinsky length scale (where A^+ is a constant).

$$l_s = C_S\Delta[1 - exp(-y^+/A^+)] \tag{2.53}$$

It can be seen from equation 2.53 that the Van Driest damping will reduce the effective filter width close to the wall (where y^+ is small), compared to the simple formulation in equation 2.51.

Although the Smagorinsky model is one of the most popular LES models, it has some drawbacks that should be kept in mind. The constant C_S in an a priori input. This is a simplification that is not necessarily appropriate in various turbulent flows. Other drawbacks are that the model prevents backscatter of energy and that it generally introduces too much diffusion in the flow (John, 2004). To avoid the simplification introduced by the constant C_S, a dynamic SGS model can be used. Such a model will treat C_S as a function of space and time, ideally describing different flow regions more appropriately. Many different dynamic models are in use today.

2.7 Finite Volume Method

As already mentioned, the fluid domain is divided into control volumes when solving the equations for the flow. This is the basic idea behind the Finite Volume Method (FVM), which is the most commonly used discretisation technique in CFD (Dejhalla and Prpic-Orsic, 2006). The Finite Volume Method is designed specifically for fluids. Other possible discretisation techniques are Finite Difference Method (FDM) and Finite Element Method (FEM). For references on general FVM, see e.g., Wendt (2009).

2.7.1 Discretisation

In FVM, the control volumes are usually called cells. The grid is the whole fluid domain divided into cells. The conservation equations are solved for each cell, giving all cells a specific value of all flow variables. Values at the boundaries can then be found from interpolation. There are many ways of doing this in practice. Nodes can be defined in the center of the elements, or somewhere on the sides. Cells can have many different shapes, and the grid can be of an ordered or disordered nature. In the following, the Finite Volume Method will be described as it is applied in OpenFOAM. OpenFOAM is an open source software package that was developed by CFD Direct, and is maintained by The OpenFOAM Foundation. For references, see the OpenFOAM user guide at cfd.direct (CFD Direct, 2015) and The OpenFOAM Foundation's description of the numerical methods of OpenFOAM at openfoam.org (The OpenFOAM Foundation, 2016).

In OpenFOAM, nodes are placed at the centre of each cell and the solution variables are defined here. The values at the cell faces, or boundaries, are interpolated from the values at the surrounding cell centres. OpenFOAM gives the user a large degree of freedom in what interpolation techniques to use. The simplest option is linear interpolation. Cells can have arbitrary polyhedral shapes in OpenFOAM. The simplest option is to use quadrilateral cells. This set up can be seen in figure 2.3.

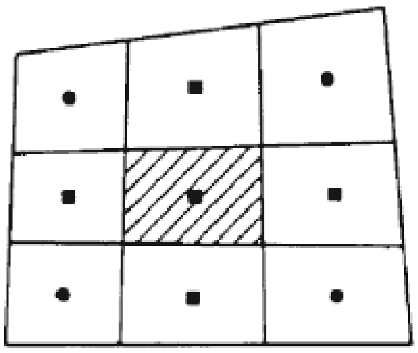

Figure 2.3: Example of orientation of cells, source: Wendt (2009)

2.7.2 Solution

The principle for solution in FVM is to balance the flux of each flow variable through the cell boundaries, and possibly the production of the variable inside the cell. Then the conservation equations will be satisfied within each cell. Gauss' divergence theorem gives the flux notation for the variables. The general form of the divergence theorem is:

$$\int\int\int_V (\nabla \cdot F) dV = \int\int_S (F \cdot \vec{n}) dS \tag{2.54}$$

The left side of equation 2.54 is an integral of the divergence of the general quantity F inside the whole volume V. In other words, it is the sum of sinks and sources of the variable in question inside the volume. The right side expresses the vector field integrated over the surface of the volume. This is the flux out of the volume. The vector $\vec{n}$ is the normal vector out of the surface S. Applying this approach to the conservation equations will result in the integral form of the equations, which is the starting point of the FVM. The integral form of the incompressible continuity equation (equation 2.3) becomes (in vector notation):

$$\int\int_S (\vec{u} \cdot \vec{n}) dS = 0 \tag{2.55}$$

For the incompressible Navier-Stokes equations (equation 2.5), the integral form becomes (also in vector notation):

$$\frac{\partial}{\partial t}\int\int\int_V \vec{u} dV + \vec{u}\int\int_S u_n dS = \int\int_S (\vec{\sigma} \cdot \vec{n}) dS \tag{2.56}$$

Here, V is the control volume (or cell volume), and S is its surface, while u_n denotes the velocity normal to the surface. Positive direction is out of the fluid, so that the flux represents transport out of the fluid.

Since each cell is assigned a constant value of each flow variable, the volume integral will simply be the volume times the variable. The area integral (or flux integral) over a surface will be the sum of the contribution from each face that the surface consists of. In this context, a face is one side, or boundary, of a three-dimensional polyhedral cell. The contribution from each boundary face will be its area times the normal component of the flow variable at that boundary. The number of faces is dependent on the type of cells and the structure of the grid. The values of the flow variables at the boundaries are as previously mentioned found from interpolation between the nodal values inside each cell. This is also the case for the gradients of the variables.

Establishing partial differential equations for each flow variable in this manner will result in an equation system that can be solved by the computer. How the equation system is solved numerically is dependent on the program used. OpenFOAM uses a segregated, iterative solution. This means that matrix equations are created for each partial differential equation, and then solved using an iterative sequence.

3 CFD Software

All numerical simulations in this project have been done using OpenFOAM. As mentioned previously, OpenFOAM is an open source CFD software package. For references, see the OpenFOAM user guide (CFD Direct, 2015).

3.1 Structure of OpenFOAM

OpenFOAM is a C++ toolbox without a graphical interface. This means that all operation of the program is done in the terminal, and all inputs are C++ text files. The standard structure of an OpenFOAM case is a specific set of folders and files. There are files describing the grid, the fluid properties, the initial and boundary conditions, discretisation and interpolation schemes, turbulence models and many more. The basic directory structure can be seen in figure 3.1.

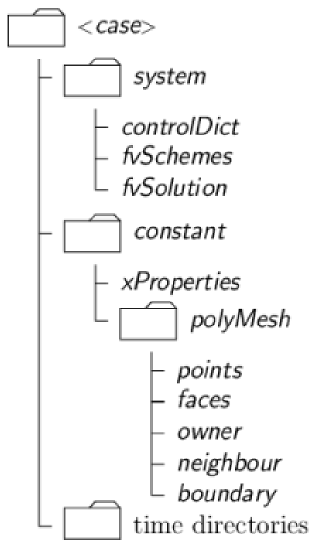

Figure 3.1: Basic directory structure of OpenFOAM, source: cfd.direct

The *polyMesh* directory under the *constant* directory contains all information about the mesh; cells, nodes, faces and boundary conditions. The mesh can be created in other programs, such as CAD software or other CFD codes, and transformed into OpenFOAM format. In this project, the mesh generator *mega*, which couples well with OpenFOAM, was used. This report will not go into detail about the files describing the structure of the grid. Only the *boundary* file will be described further in this report, since it contains the type of boundary conditions used. Files in OpenFOAM are also called dictionaries. Different ”sections” of the files are called sub-dictionaries.

Other files under the *constant* directory are the dictionaries describing the fluid and flow properties. They are denoted *xProperties* in figure 3.1. The most essential are *transportProperties* and *turbulenceProperties.* The *transportProperties* dictionary contains the kinematic viscosity of the fluid, while *turbulenceProperties* contains the type of flow model used. This can be laminar in the simplest case, or a turbulence model, like a RANS model or various other more advanced turbulence models like LES.

3.2 Solvers and Numerical Schemes

The *system* directory contains the files that control the solution of the problem. In *fvSchemes*, the numerical schemes for different terms in the equations are set. There are schemes for interpolation between cell centres and faces, and for derivatives in time and space. An example of an *fvSchemes* dictionary as used in this project can be seen below:

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,epsilon) Gauss limitedLinear 1;
    div(phi,omega)  Gauss limitedLinear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}
```

As it can be seen above, there are several choices of numerical schemes for the different terms. In this project, Gauss integration and linear interpolation was used. The sub-dictionary *ddtSchemes* specifies the discretisation of first order time derivatives. It can be set to *steadyState* if the user does not wish to solve for time derivatives. The *gradSchemes* sub-dictionary specifies the discretisation scheme for first order derivatives, and *laplacianSchemes* for second order. *Gauss linear* means that Gauss integration is used with a linear interpolation scheme. Discretisation of convective terms (or divergence terms) are specified by the *divSchemes* sub-dictionary. Here, a discretisation and interpolation scheme is needed for each term. Gauss integration is the only choice for discretisation of divergence terms, but there are several choices for interpolation scheme. The *snGradSchemes* sub-dictionary specifies the discretisation scheme for surface normal gradients. The surface normal gradient is the component normal to a cell face, of the gradient in each cell that is connected at that face. In the *fluxRequired* sub-dictionary, the fields that require a calculation of flux are listed. In this case, this is just pressure, p.

The *fvSolution* dictionary contains the choice of solvers for each equation, and some customization such as their tolerance. Most important is the *solvers* sub-dictionary. An example from the files used in this project is shown below:

```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|omega|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother       GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}
```

The solver for each equation is specified in separate sub-dictionaries as seen above. For the RANS simulations in this project, the GAMG solver was used for pressure, and the smooth solver was used for all other equations. The GAMG solver is a geometric-algebraic multi-grid solver. Its main principle is to make the original grid coarser, find a solution for this grid, and then refine the grid in stages, mapping the solution for coarser grids onto the finer grids. In other words, the GAMG solver iterates from solutions of coarser grids in steps to find a solution for the final grid. The *agglomerator*, *nCellsInCoarsestLevel* and *mergeLevels* keywords specify how this iteration is done.

PCG and PBiCG (Preconditioned (Bi-)Conjugate Gradient) are other OpenFOAM solvers. These were used for the LES case in this project, although the GAMG solver can also be used for LES (Arslan et al., 2011). What solver is the best is dependent on the case. Generally, the GAMG solver is reported to be the fastest when the number of processors are relatively low (Rivera and Furlinger, 2011). Doing a sensitivity study with respect to the choice of solvers was considered to be outside the scope of this project.

Controlling the convergence of the solution is done using the *tolerance* and *relTol* keywords. As mentioned previously, the OpenFOAM solvers are iterative. For each iteration, a residual is produced for each equation. The residual is the difference between the left and right side of the equation, so a lower residual means the solution is "more correct". The *tolerance* keyword specifies the value that the residual must be reduced to in order for the iteration to be finished. Hence, a lower tolerance will result in more iterations and hopefully a more accurate solution. The *relTol* keyword specifies the tolerance for ratio between current to initial residual. If the ratio drops below this value, the iteration will be finished. If the relative tolerance is set to zero, only the absolute tolerance will control the solution. The maximum number of iterations can also be specified using the *maxIter* keyword. The default value is 1000, but that many iterations should rarely be necessary.

3.3 Boundary Conditions

3.3.1 Prescribing Boundary Conditions

The boundary conditions are specified in the *boundary* dictionary under the *constant/polyMesh* directory, and in the dictionaries under the *0* directory. Looking at figure 3.1, the *0* directory will be located in the *time directories* group. The *boundary* dictionary is a part of the mesh files and specifies the basic type of boundary for each edge of the fluid domain. An example from this project can be seen in the following:

```
cylinder
    {
        type            wall;
        inGroups        1(wall);
        nFaces          8800;
        startFace       9748950;
    }
    inlet
    {
        type            patch;
        nFaces          13000;
        startFace       9757750;
    }
    outlet
    {
        type            patch;
        nFaces          13000;
        startFace       9770750;
    }
    top
    {
        type            cyclic;
        inGroups        1(cyclic);
        nFaces          25500;
        startFace       9783750;
        matchTolerance  0.0001;
        neighbourPatch  bottom;
    }
    bottom
    {
        type            cyclic;
        inGroups        1(cyclic);
        nFaces          25500;
        startFace       9809250;
        matchTolerance  0.0001;
        neighbourPatch  top;
```

The boundary types used in this project are *wall*, *patch* and *cyclic*. Which faces belong to what boundary is specified by the *nFaces* and *startFace* keywords. As the name suggests, the *wall* type is suitable for solid walls, when a no-slip condition is wanted. The *patch* type is a more general boundary without any geometrical constraints. It can be used for many different specific types, such as free-slip or fixed value boundaries. The *cyclic* type specifies periodic boundary conditions. Use of this condition requires two boundary patches to be connected. Looking at the example above, the *top* and *bottom* boundaries are connected through the *neighbourPatch* keyword.

In OpenFOAM, all grids are seen as three-dimensional, as it does not have a 2-D engine. It is, however possible to solve in 2-D by using the *empty* type boundary. This tells OpenFOAM that no solution is required for this boundary, resulting in a 2-D solution.

While the *boundary* dictionary sets the basic type of boundaries, dictionaries for each variable under the *0* directory sets the more specific boundary and initial conditions. An example of a *U* dictionary, describing conditions for velocity, can be seen on the next page. The *dimensions* keyword sets SI dimensions for the variable. The dimension for velocity is here $m \cdot s^{-1}$.

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (1 0 0);

boundaryField
{
    cylinder
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    inlet
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    top
    {
        type            cyclic;
    }

    bottom
    {
        type            cyclic;
    }
```

As seen above, the *boundaryField* sub-dictionary has one further sub-dictionary for each boundary that was specified in the *boundary* dictionary. Each sub-dictionary sets the condition for one boundary. The boundary conditions used here are *fixed-Value*, *zeroGradient* and *cyclic*. Initial conditions for the whole field are set by the *internalField* keyword. For the cylinder boundary, which is a solid wall, *fixedValue* is used to specify a no-slip condition by setting velocity to zero in all directions. At the inlet, *fixedValue* is used to specify the desired far-field velocity, U_∞. The *zeroGradient* type is used to model free-slip conditions. In mathematical terms, a *zeroGradient* condition simply means that the gradient, or spatial derivative, of the variable in question is set to zero. Again, the *cyclic* type is used to specify periodic conditions, and the *empty* type is used to get a 2-D solution, just as in the *boundary* dictionary.

3.3.2 Wall Functions

When running simulations with turbulence modelling, the variables for turbulence must also have boundary and initial conditions. As mentioned previously, wall functions can be used to specify boundary conditions at solid walls for these variables. An example of the dictionary k, that sets boundary and initial conditions for mean turbulent kinetic energy, is seen below:

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.000006;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0.000006;
    }
    outlet
    {
        type            zeroGradient;
    }
    cylinder
    {
        type            kqRWallFunction;
        value           uniform 0.000006;
    }
    frontAndBack
    {
        type            empty;
    }

    top
    {
        type            cyclic;
    }

    bottom
    {
        type            cyclic;
    }
}
```

A *kqRWallFunction* type boundary specifies a wall function for k at the cylinder solid wall. A wall function is a way of simplifying the modelling of the near-wall region. When doing CFD simulations of turbulent flows, a very fine grid in the near-wall region is needed to capture important mechanisms in the flow such as separation. With high Reynolds numbers, the boundary layer will be very thin. This means that the cells near the wall must be very small for the solution to be able to accurately describe the distribution of the flow variables here. This can result in a large number of cells and hence long simulation time. The wall function approach avoids this problem by assigning a distribution of the variables to the first cell near the wall. This distribution is taken from experience and experimental results. There are many kinds of wall functions that can be used in CFD.

When using a wall function, the value in the first cell near the wall will ideally describe the entire viscous part of the boundary layer, meaning that only one cell is needed in this region. This greatly simplifies the meshing in the near-wall region, and can save a lot of computation time. The wall function approach is however not always valid. As discussed in chapter 2.5, the boundary layer consists of a viscous and a logarithmic sublayer, with a buffer region between them (see figure 2.1 or Tennekes and Lumley (1972)). Since the wall function approach is meant to simplify the viscous part of the boundary layer (which is the most complicated to describe), the height of the first cell near the wall should be specified so that it is somewhere inside the log-layer. This way, one ensures that the whole viscous sublayer is contained in the first cell. The log-layer is generally between y^+ of 30 to 300 as seen in figure 2.1. By choosing a suitable y^+, the height of the first cell near the body can be calculated from equation 2.35. Note that y^+ should not be higher than the height of the log-layer, as the first cell will then span also outside the boundary layer.

3.4 Running OpenFOAM

3.4.1 Time Step Control

Controlling an OpenFOAM simulation is done in the *controlDict* dictionary under the *system* directory. The most important controlling feature is the time step. To ensure stability and convergence of the solution, the time step must be set sufficiently small. However, the smaller the time step, the longer simulation time, hence the time step should preferably be set as high as possible while ensuring stability. If Δt is the time step and Δx is the size of a cell, the Courant-Friedrich-Lewy (CFL) number is defined as:

$$CFL = \frac{u\Delta t}{\Delta x} \tag{3.1}$$

The CFL number is a measure of how far a fluid particle might travel relative to the cell size. A CFL number of 1.0 means that the particle travels the whole length of the cell during one time step. Too large CFL will be a problem for the stability of the solution, as the flow will be "too fast" to be described correctly by the grid and the solver. In practice, CFL should be set significantly lower than unity to ensure stability of the solution. In the author's experience from this project, CFL$\leq$ 0.5 is usually a safe choice.

Other important controls in the *controlDict* dictionary are start and end times, when to log the field data and the precision of time and the writing. Also, the user can add various functions to *controlDict* to make OpenFOAM calculate various extra results such as forces and average fields. This is a very useful feature. In this project, calculation of force coefficients simplified the presentation of results, and calculation of average velocity and pressure fields was useful as a way to check that the vortex shedding was stable.

When running OpenFOAM, the program writes some important output data to the terminal for the user. An example can be seen below:

```
Time = 0.01

Courant Number mean: 0.0164731 max: 0.449916

smoothSolver: Solving for Ux, Initial residual = 0.014553, Final residual = 5.45318e-07, No
Iterations 2

smoothSolver: Solving for Uy, Initial residual = 0.00104905, Final residual = 1.69027e-06, No
Iterations 2

GAMG: Solving for p, Initial residual = 0.0967255, Final residual = 0.00604559, No Iterations 2

time step continuity errors : sum local = 3.53756e-08, global = -1.00971e-11, cumulative =
2.05232e-09

GAMG: Solving for p, Initial residual = 0.00680829, Final residual = 9.79532e-07, No
Iterations 76

time step continuity errors : sum local = 5.23358e-12, global = -7.31602e-15, cumulative =
2.05231e-09

smoothSolver: Solving for epsilon, Initial residual = 0.000471161, Final residual = 9.15529e-06,
No Iterations 1

smoothSolver: Solving for k, Initial residual = 0.00274485, Final residual = 9.24964e-06, No
Iterations 2

ExecutionTime = 1.55 s ClockTime = 5 s
```

As it can be seen above, OpenFOAM gives the user the CFL number, and the residuals and number of iterations for each equation. In other words, this output tells the user about how the solution is doing. The user can from this see if the CFL number is too high, or if more or less precision in the residuals is needed. When something goes wrong, the user can go back in the log to see what happened, and then try to fix the problem. If the CFL number is too high, the user will have to either reduce the time step, or make adjustments to the grid. If the number of iterations is very high, the user might consider adjusting the tolerance of the residuals.

3.4.2 Choosing Numerical Schemes

The choice of numerical schemes for the solution is also a very important consideration. This affects both stability and accuracy of the results. In general, a higher order scheme will be more accurate. However, first order schemes can be used for convective terms and time derivative terms to achieve increased stability if this is an issue. Higher order schemes are generally more oscillatory and unstable, which may be a problem in flows that are not. In OpenFOAM, usual time derivative discretisation schemes are *Euler* (first order) and *backward* (second order). For convective terms, usual interpolation schemes are *linear* (second order) and *upwind* (first order). It is a common guideline to use first order schemes in the beginning of the simulation to preserve stability, and then switch to higher order schemes for increased accuracy (Guerrero, 2015). This may be less important for unsteady flows, as it was not an issue in this project. Rodi (1997) even argues that first order upwind schemes should not be used for any vortex shedding flows because they may damp out the periodic motion. When running LES, first order accuracy will generally not be sufficient (also argued by Rodi (1997)). That's why the *backward* time scheme was used for the LES case in this project. This was the only difference from the schemes used in the RANS cases, as second order interpolation schemes were used in all cases.

4 Pre-Processing

4.1 Basics of Grid Generation

In the context of CFD, knowledge about good grid generation techniques are very valuable. A good grid should enable the solution to be as accurate as necessary, while ensuring that the amount of computational effort can be handled by the available hardware.

The basic idea behind fluid domain grid generation is to have many small cells in the areas with the dominating flow mechanisms, and fewer and bigger cells further away. This is to be able to accurately describe the flow physics in areas of special interest, like near a solid body, while keeping the number of cells from being unnecessarily high. The shape of the cells can also be important. As mentioned previously, it is possible to have arbitrary polyhedral shaped cells in numerical flow simulations. An ordered quadrilateral structure is the simplest, and has been used in this project. Such cells do not have to be quadratic, or even rectangular. The cell sides do not even have to be straight.

Even though the user has a large degree of freedom when it comes to generating a grid, there are some basic pitfalls to be aware of. It is always important to be aware of the length of the cells in different directions relative to what flow mechanisms are present near that cell. For example, if a cell is much longer in one direction than in the other, the cell might not be able to describe the flow accurately in both directions. Keeping in mind the CFL number (equation 3.1), the length of the cell is especially important in the direction of the flow. Having long cells in an area where the flow is fast, like near the body, may compromise the stability of the solution, and hence force a smaller time step. Another point to be aware of is that very different shaped cells next to each other may cause problems. The reason for this is that they may have different abilities to describe the flow accurately, resulting in a rapid change in the flow variables across the cells. This may result in a flow field that is not physically correct. A smoother change in cell geometry across an area will enable the flow variables to also have a smooth distribution across the cells. In general, the safest approach is probably to have simple, straight, quadratic cells. This may be difficult to achieve in practice, but it is a good starting point for a grid generation. Important learning from this project is that the more distorted shape of the cells, the more difficulties arise, and that this is mainly important in the region near the body, where complex mechanisms like separation are going on.

It is not only the cell structure of the grid that is important in grid generation. The size of the whole fluid domain can also have significant impacts on the results if it is not carefully selected. Domains that are too small will be an issue. Domains that are too large will not be a problem for the results, but might result in unnecessary computational effort. The reason why too small domains are a problem is that the far-field boundary conditions should not influence the solution. This principle is not only related to choice of boundary conditions, but also to the domain size.

Far-field boundary conditions will always introduce a restriction on the flow. It is rarely the case that such boundary conditions are 100% correct physically (unless they are solid walls). For example, a zero-gradient far-field boundary condition will not be a free-slip condition exactly, but an approximation that imposes additional restrictions. Therefore, if the domain is too small, the restrictions of the boundary conditions might be so close that they influence the flow in the region of main interest, like the cylinder in this case. A sensitivity study with respect to domain size is a good idea for avoiding this problem.

4.2 Grid Generation Method

4.2.1 General approach

The procedure of generating a grid for CFD should contain some important points. Planning and refining are keywords here. In order to end up with a good grid, it is important to take all the important aspects of the flow into account; What flow mechanisms are present? Are there solid bodies in the way? What is the Reynolds number? What turbulence model is appropriate? What computational capacities are available? These are all questions that should be answered as well as possible before making the grid.

Since this project deals with flow past solid bodies, this will be the focus here. Other types of flow may require other considerations. When dealing with flow past bluff bodies, the key is to be able to capture the separation and vortex generation properly. This will always require a relatively fine grid close to the body. How fine it needs to be is dependent on many aspects. Most important are Reynolds number, turbulence model, and near-wall modelling approach.

4.2.2 Planning

A key parameter to be determined before generating the grid is the y^+ value (equation 2.35). As mentioned previously in chapter 2.6, a LES grid needs y^+ values around 1.0 to capture the mechanisms near the wall. In RANS simulations, the required y^+ is dependent on the near-wall modelling. Without wall functions, the grid needs to be as fine as the LES grid close to the wall, for the same reasoning. As described previously in chapter 3.3.2, wall functions can be used as a simple near-wall modelling. With this approach, y^+ can be increased significantly to make the first cell extend into the log-layer (see chapter 2.5 about turbulent boundary layers).

The required size of the domain should also be estimated before making the grid, as argued in chapter 4.1. This can be done by reviewing previous work by others.

The acceptable number of cells related to the computational capacity should be taken into account. This is to avoid simulations that are too time consuming.

4.2.3 Refining

The first estimate of y^+ may not be correct after the first simulation. For example, the flow will be faster over the body than in the free stream, due to blocking effects. This will increase the y^+ value in this region (see equation 2.35). This means that it might be necessary to refine the grid by changing the y^+ value at the body after the first simulation. OpenFOAM has built-in functions to check the value of y^+ during and after the simulations.

As already mentioned, it is a good idea to do a domain size sensitivity study to determine if the domain size is affecting the solution. This can be done by making the domain smaller and larger, and see if the results change. If a larger domain changes the results significantly, the first domain was probably too small. If a smaller domain does not change the results significantly, the first domain was probably unnecessarily large.

Capturing the vortex street behind the body requires the grid in the wake to be fine enough. Looking at the flow after an initial simulation can tell the user if this is achieved or not. If vortices disappear much quicker than expected from experiments, the grid may need more cells in the wake, if this is of interest. Also, if separation is not at all present, even though it is expected, the grid close to the body may be too coarse to capture this.

4.2.4 OpenFOAM Meshing Utilities

OpenFOAM has built-in meshing utilities that allow the user to generate a grid by writing simple text files with a specific syntax. The simplest utility is *blockMesh*, which generates a grid based on blocks with prescribed size, number of cells and grading. The other, more advanced option is *snappyHexMesh*, which is used to make grids around complex geometries. As *snappyHexMesh* is not used in this project, it will not be discussed further here.

The *blockMesh* approach is based on the following items:

- Defining nodes (x,y,z coordinates).
- Defining 3-D blocks from nodes (8 nodes per block).
- Prescribing each block a number of cells and grading in each direction.
- Defining boundaries from nodes (4 nodes per boundary patch).
- Prescribing boundary conditions to each boundary (like patch, wall or cyclic).

All of the above is written in a file called *blockMeshDict*, and the mesh is generated by running the OpenFOAM function *blockMesh*. Examples from a simple *blockMeshDict* file can be seen on the next page. There are three different subdictionaries defining nodes(vertices), blocks and boundaries, respectively. Since OpenFOAM has no graphic user interface, visualization of the grid has to be done in third-party software like paraFoam.

```
vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);

boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

OpenFOAMs meshing utilities have both advantages and disadvantages. The big advantage is that when a *blockMeshDict* has been written, mesh generation is quick and small adjustments are easily done. Disadvantages are the lack of a graphic interface and the amount of time required to write a *blockMeshDict* for a complex grid.

4.2.5 Mega Mesh Generator

Mega is an in-house meshing program for Linux developed at the Institute of Marine Technology at NTNU. Its graphical interface allows the user to visualize the grid while it is under construction. This is a huge advantage compared to OpenFOAMs own utilities. Mega uses the *gmsh* format, which can be converted to OpenFOAM format using the built-in OpenFOAM function *gmshToFoam*. The combination of a graphic meshing program like Mega and a simple format conversion is a very attractive approach. However, one downside is that the conversion may take a long time if the grid is very fine, which can make small grid adjustments annoyingly time consuming.

4.3 Grid Types

4.3.1 Grid Classification

There are three different ways to classify a grid based on cell connectivity. The classification is described by e.g., CFD Online (2012). The grid can be structured, unstructured, or hybrid. Hybrid just means it's a mix of structured and unstructured. A structured grid has an ordered nature, where the connectivity can be expressed as a simple array. An unstructured grid has irregular connectivity that can not be fully described by a simple array. This means that it requires more storage space than a structured mesh because the connectivity must be described explicitly for each cell. An advantage of using unstructured grids is that all kinds of cell geometries can be used, while a structured grid is restricted to using one specific type. Unstructured grids will not be discussed further as only structured grids have been applied in this project.

Grids can also be classified based on type of cell geometry and dimension (2-D/3-D). The most common choices are quadrilaterals (rectangles), triangles, hexahedrals (cubes), tetrahedrals or pyramids.

4.3.2 Grid Layout

Some basic types of grid design will be briefly discussed here. The layout that is easiest to make for structured grids is the so-called H-mesh. A H-mesh basically consists of boxes with straight edges. If the edges are not straight, the grid is often called C-mesh or O-mesh, depending on how many of the edges are curved. Examples of this can be seen in figure 4.1. Only H-type grids have been used in this project.

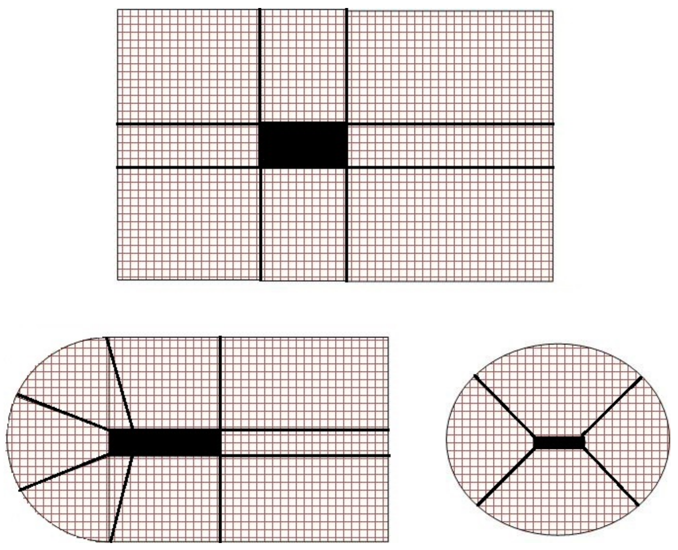

Figure 4.1: Different grid layouts: H-mesh (upper), C-mesh (lower left), O-mesh (lower right).

5 Simulation Set-up

The goal of this project was to simulate flow around a rectangular cylinder that is conceptually a simplified ship hull model, hence floating in the free surface. As the scope of the project did not include advanced free surface modelling techniques, other solutions had to be found. The two methods that were within reach are using a simple free-slip free surface boundary condition, and using a double-body.

In all cases, the body of interest was a 5:1 rectangular cylinder with flow in longitudinal direction. The Reynolds number is 26,400 in all cases, with respect to the cross-flow width. Turbulence model for all RANS cases is the realizable k-epsilon model of Shih et al. (1995). The k-omega SST model of Menter (1993) was also applied in the early stages of the project, but this model did not produce satisfactory results. This will be discussed further in the results section. The Smagorinsky SGS model (Smagorinsky, 1963) was used in the LES case.

5.1 Double-Body Grid

The concept of a double-body means that the geometry in question is mirrored over the free surface to create a body that is twice the size, with fluid on all sides. Hence, the double-body case is essentially a fully submerged box of double height. The idea behind this is to avoid the issue of the free surface modelling, and simulate a more realistic flow. Of course, the total resistance will have to be divided by two to compensate for the double size of the body. Disadvantages of this method is that it requires more cells, and that it may introduce 3-D effects that are not necessarily relevant in the real case.

Several grids were created for this case in order to do a grid size sensitivity study. Doing such a study is vital for validation purposes, and a very common procedure in CFD. A grid size sensitivity study should include a study of the cell count and general fineness of the grid, the domain size, and possibly the near wall cell size if this is of interest.

A short domain size sensitivity study was done by using a grid from previous work, reducing the lengths of the outer boundaries in steps and seeing if the solution changed. The resulting domain can be seen in figure 5.1. Incoming flow direction is positive x-direction. Comparing with the domain of Ong (2012), this domain should be large enough to avoid far-field effects on the body. Although the case of Ong (2012) is 2-D, the geometry is similar and it is assumed that with a small increase in the size (as seen in figure 5.1), it will be sufficient also in 3-D. It should be noted that the vertical length of the domain (z-direction) was not subject to a sensitivity study. This was simply due to the need to limit the number of cells in this direction, as computational capacity was limited. The vertical distance from body to boundary was chosen to be as large as possible while keeping the number of cells within reasonable limits.

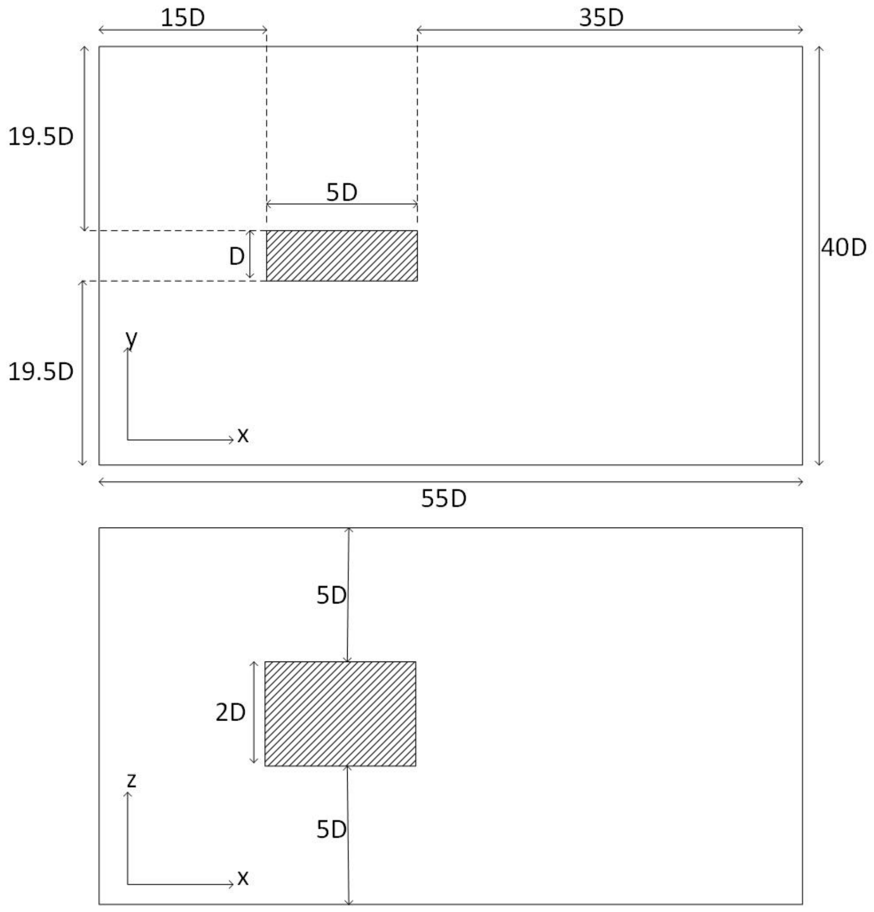

Figure 5.1: Double-body fluid domain

The characteristics of the different grids used can be seen in table 5.1 below. All the grids have the same near-wall cell size and should hence have the same values of y^+ around the body. The theoretical value of y^+ is based on free-stream velocity. The actual value will vary over the body surface because the fluid velocity changes (see equation 2.35).

Table 5.1: Grids used in sensitivity study

Grid	**Cell count**	$\boldsymbol{\Delta s}$	**cell count on body**	**theoretical $\boldsymbol{y^+}$**
Coarse	3,275,000	0.06 m	100x20	43.62
Medium	3,740,000	0.06 m	100x20	43.62
Fine	4,160,000	0.06 m	100x20	43.62

5.2 Floating Body Grid

The floating body case is the simplest one used in this project. The grid simply ends where the free surface is thought to be, and the body is placed next to this boundary (see figure 5.2). This is thought to be a more simplified approach than the double-body case, so less focus was put onto it. Only one grid was used for this case, and no grid sensitivity study was performed. This was considered reasonable because the grid used is the same as the medium grid of the double-body case, except that it is half the size. It is assumed that this grid would work well also in this case since it was proven to be sufficient for the double-body case. Grid properties are summarized in table 5.2. The domain seen in figure 5.2 is the same as for the double-body in the x-y plane, hence far-field boundary spacing is the same.

Table 5.2: Floating body grid properties

Cell count	$\boldsymbol{\Delta s}$	**cell count on body**	**theoretical $\boldsymbol{y^+}$**
1,870,000	0.06 m	100x20	43.62

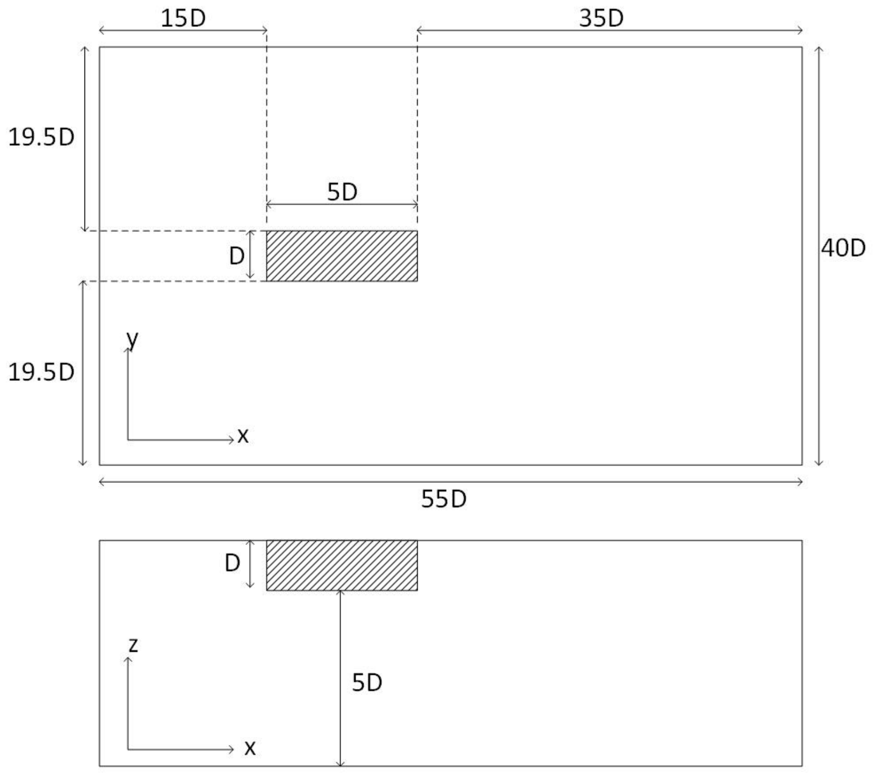

Figure 5.2: Floating body fluid domain

5.3 LES Grid

The LES case is a double-body case. No floating body LES was performed, as the double-body case was considered the main focus. The difference between the LES and the RANS double-body grid is essentially a much higher cell count and a much smaller near-wall cell size in the LES grid. No thorough grid sensitivity study was performed for this case, simply because the time scope did not allow it. Running one LES grid took long enough. Instead, the gridding approach was adopted from Arslan et al. (2011), which case is somewhat similar. It was assumed that making the grid using the same approach would ensure that the grid is fine enough, as the results of Arslan et al. (2011) were considered highly trustworthy.

The grid of Arslan et al. (2011) is a hybrid grid, only structured near the wall. Since only purely structured grids was used in this project, the resulting cell count of the LES grid was higher than for Arslan et al. (2011). A summary of the key grid properties can be seen in table 5.3. Domain size is the same as in the RANS case.

Table 5.3: LES grid (double-body)

Cell count	Δs	**cell count on body**	**theoretical** y^+
15,620,000	0.001 m	250x100	0.7269

5.4 Boundary Conditions

Boundaries for the double-body and floating body cases can be seen in figure 5.3 and 5.4, respectively. A more detailed summary for each variable is seen in tables 5.4 and 5.5. When possible, periodic boundary conditions were used for the far field boundaries. This was considered to be the least restrictive of the possible choices. Periodic boundary conditions are widely used in similar studies in the literature, such as Mannini et al. (2010) and Arslan et al. (2011).

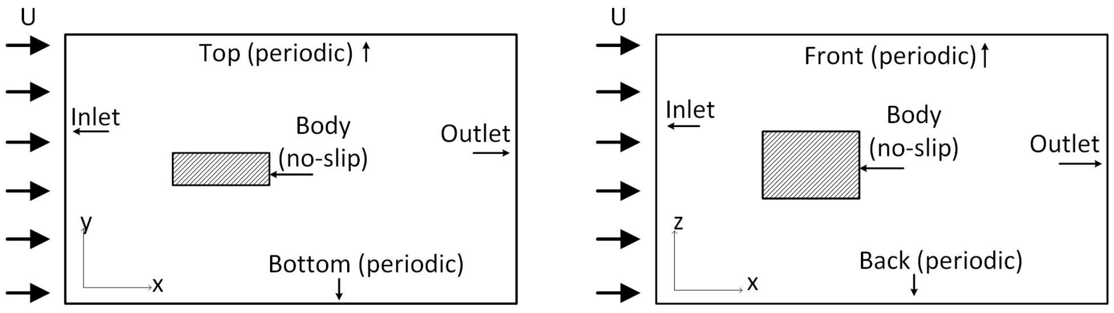

Figure 5.3: Double body boundaries

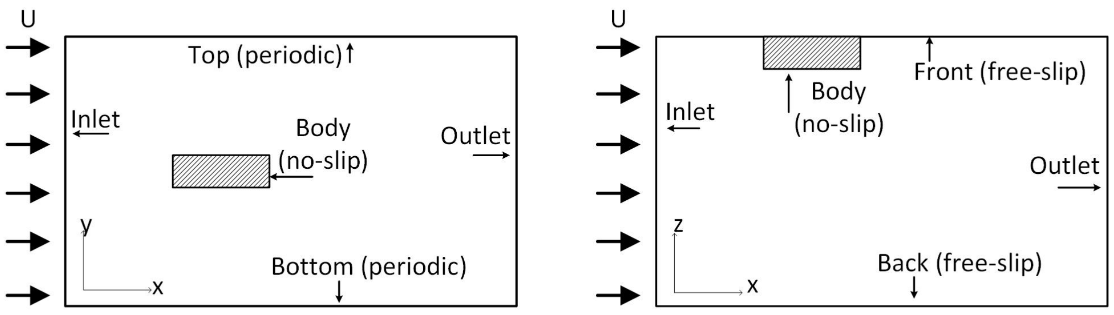

Figure 5.4: Floating body boundaries

Inlet conditions are prescribed velocity and zero-gradient pressure. Outlet conditions are zero-gradient velocity and prescribed zero pressure. Wall conditions are prescribed zero velocity and zero-gradient pressure.

Table 5.4: Boundary Conditions: Double-Body Case

	Pressure	**Velocity**	**k** (RANS only)	**epsilon** (RANS only)	ν_{SGS} (LES only)
Inlet	zero-gradient	constant	constant	constant	zero-gradient
Outlet	constant=0	zero-gradient	zero-gradient	zero-gradient	zero-gradient
Body	zero-gradient	constant=0	wall function	wall function	zero-gradient
Top/ Bottom	periodic	periodic	periodic	periodic	periodic
Front/ Back	periodic	periodic	periodic	periodic	periodic

Table 5.5: Boundary Conditions: Floating Body Case

	Pressure	**Velocity**	**k**	**epsilon**
Inlet	zero-gradient	constant	constant	constant
Outlet	constant=0	zero-gradient	zero-gradient	zero-gradient
Body	zero-gradient	constant=0	wall function	wall function
Top/Bottom	periodic	periodic	periodic	periodic
Front/Back	zero-gradient	zero-gradient	zero-gradient	zero-gradient

5.5 Numerical Set-up

A summary of applied numerical schemes and solvers can be seen in table 5.6. See chapter 3.2 for an explanation of the terms. There is a difference between RANS and LES cases in the time scheme used. This is due to increased need for accuracy in LES. The first order Euler scheme is sufficiently accurate for RANS simulations, while the more accurate second order *backward* scheme is better suited for LES.

Table 5.6: Numerical Schemes and Solvers

	RANS	**LES**
Solver for Pressure	GAMG	PCG
Solver for Other Variables	smoothSolver	PBiCG
Time Scheme	Euler	backward
Gradient Scheme	Gauss linear	Gauss linear
Convective Schemes	Gauss linear/ limitedLinear	Gauss linear/ limitedLinear
Laplacian Scheme	Gauss linear corrected	Gauss linear corrected
Interpolation Scheme	linear	linear
SN Gradient Scheme	corrected	corrected

No sensitivity study with respect to the numerical schemes has been carried out. For the RANS cases, choice of schemes were taken from OpenFOAM tutorials. For the LES case, choice of a second order time scheme was made on recommendations of Rodi (1997) and Mannini et al. (2010). Other than the time scheme, the numerical schemes are the same in RANS and LES.

When dealing with numerical schemes, it is important to note the nature of the flow. In this case, the flow is unsteady in nature. This may make choice of numerical schemes somewhat easier since one does not have to worry about unwanted instability. For simulation of steady flows, the use of more stable convective terms such as first order upwind can be used to ensure stability (Guerrero, 2015). It is assumed that in this unsteady flow case, the choices made for numerical schemes are well suited.

6 Results

In the following chapters, the results of each case will be presented individually. Only the RANS double-body case includes a grid sensitivity study. There was a large difference between the RANS and LES cases in the amount of computational time required. The double-body RANS (medium grid) case was run for about 27 hours on 16 processors. The LES case was run on 64 processors for about 320 hours, and even then it would be preferred to run it a lot longer.

Simulations with the $k - \omega$ SST model were unable to capture vortex shedding for the double-body case. The simulations simply resulted in a steady, symmetric flow around the body. This is why the realizable $k - \varepsilon$ model, which captured the vortex shedding clearly, was chosen instead.

6.1 RANS Double-Body

6.1.1 Grid Sensitivity Study

A summary of the results of the grid sensitivity study can be seen in table 6.1. The average value of y^+ is taken as a sample at the latest time step, which is at 400 seconds.

Table 6.1: Results of grid sensitivity study

Grid	**mean Cd**	**stdev Cl**	**St**	**average y^+**
Coarse	0.8732	0.0815	0.1310	26.8930
Medium	0.8751	0.0799	0.1430	27.0964
Fine	0.8795	0.0837	0.1370	27.1296

The difference from the medium grid to the coarse and fine grids in mean drag coefficient are 0.2% and 0.5%, respectively. This is a very small difference. For lift coefficient, the difference is larger. It is 2.0% difference to the coarse and 4.8% to the fine grid in standard deviation of lift coefficient. The differences in Strouhal number are also larger than for drag coefficient. It is 8.4% difference to the coarse grid and 4.2% to the fine grid. The values of average y^+ are very close for all grids. It should, however be noted that the average value does not tell us very much about the distribution of y^+ around the body. Since y^+ is dependent on fluid velocity, it will vary drastically across the wall. A plot of the y^+ distribution around the body in an x-y cutting plane can be seen in figure 6.1. The variation here is large, with high values of y^+ before the separation point, and then much lower values behind the separation point, until reattachment at the back gives higher values again. Ideally, the cell sizes at the wall should compensate for this. It is best to have a more or less uniform distribution of y^+ at the wall. This may be especially important when using wall functions, as this approach is very sensitive to the boundary layer thickness, as argued earlier in chapter 3.3.2. However, making variable cell heights on the wall to compensate for this would be too time consuming in the scope of this project. The results here are assumed to be accurate as long as the average value of y^+ is reasonable.

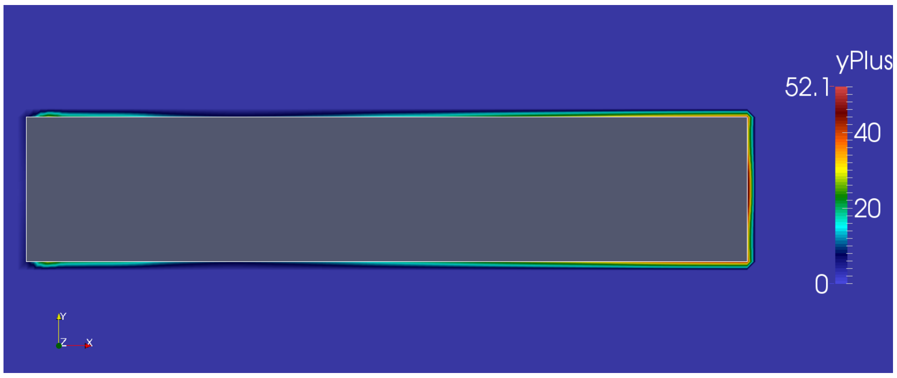

Figure 6.1: Distribution of y^+ around the body for RANS double-body case

6.1.2 Force Coefficients and Frequency

Plots of drag and lift coefficients, and spectrum of lift fluctuations, can be seen in figures 6.2, 6.3 and 6.4, respectively. All figures give indications of highly unsteady and non-harmonic flow. The spectral plot shows that there is a very clear peak in frequency of oscillations. Some other oscillation frequencies are present, but not as distinct.

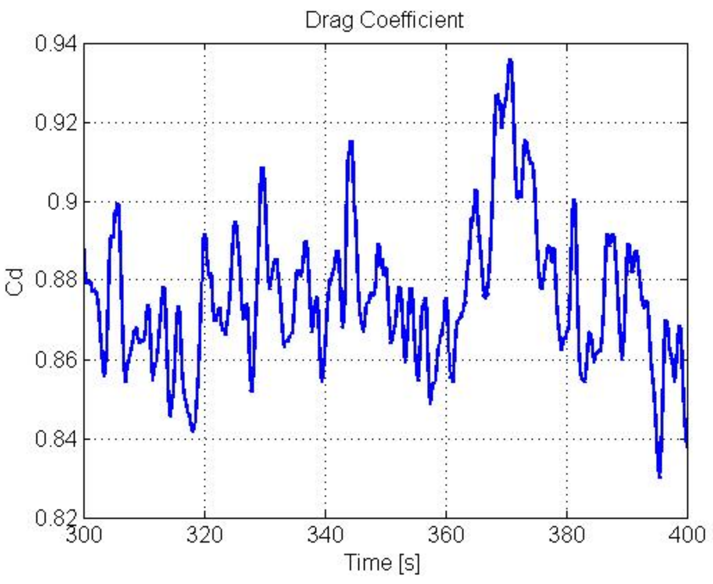

Figure 6.2: Time-history plot of drag coefficient for RANS double-body case

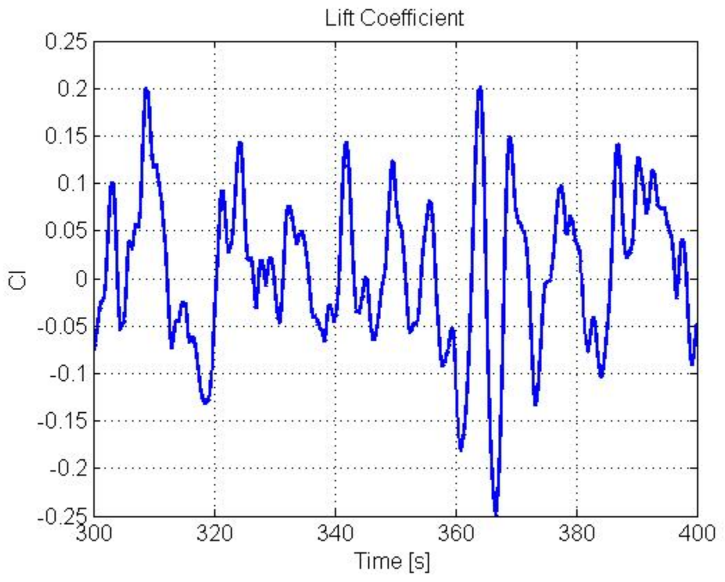

Figure 6.3: Time-history plot of lift coefficient for RANS double-body case

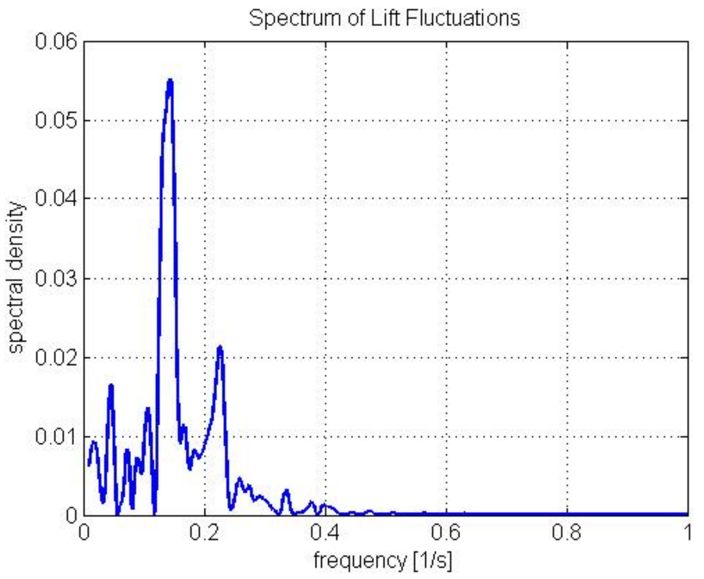

Figure 6.4: Spectrum of lift fluctuations for RANS double-body case

6.1.3 Flow Variable Plots

Figures 6.5 and 6.6 show snapshots of pressure and velocity, respectively, both in the x-y plane. The pressure is relative, so the free-stream value is zero. Figure 6.7 shows vorticity in both x-y and x-z planes. This is to illustrate clearly the three-dimensionality of the flow. The figures show that there is distinct vortex shedding and reattachment. There looks to be vortex shedding in both planes. Figure 6.8 shows streamlines of the mean flow. It looks to be reasonably symmetrical.

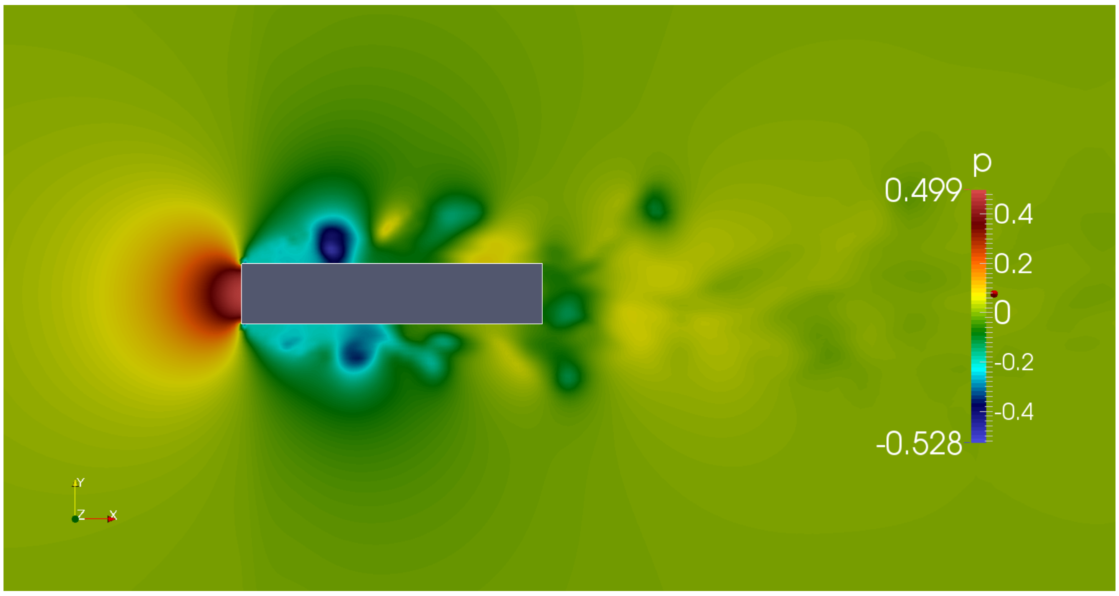

Figure 6.5: Pressure for RANS double-body (x-y plane)

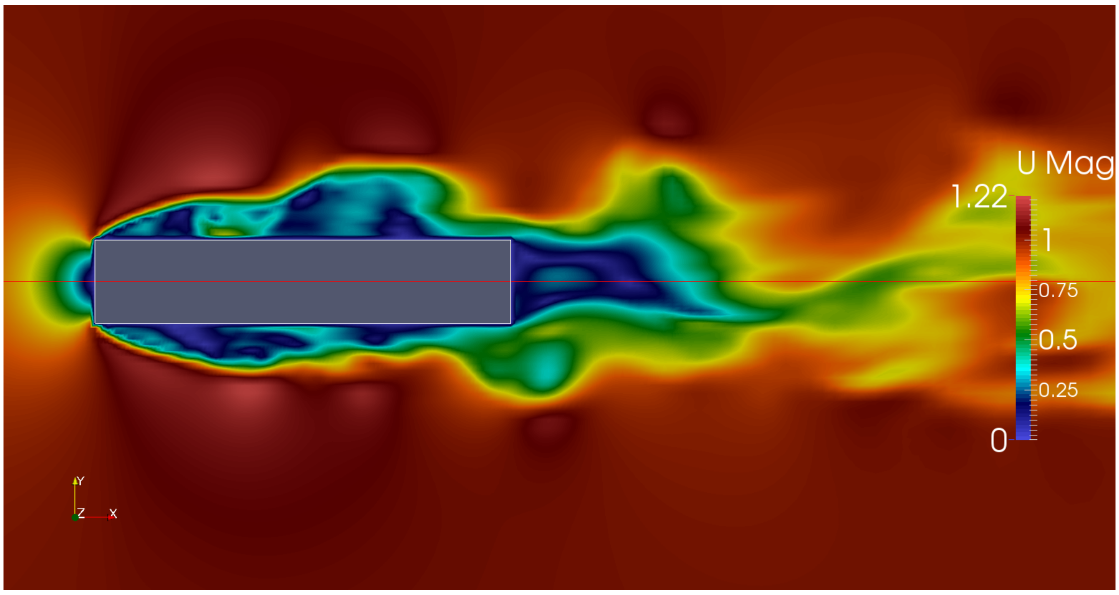

Figure 6.6: Velocity for RANS double-body (x-y plane)

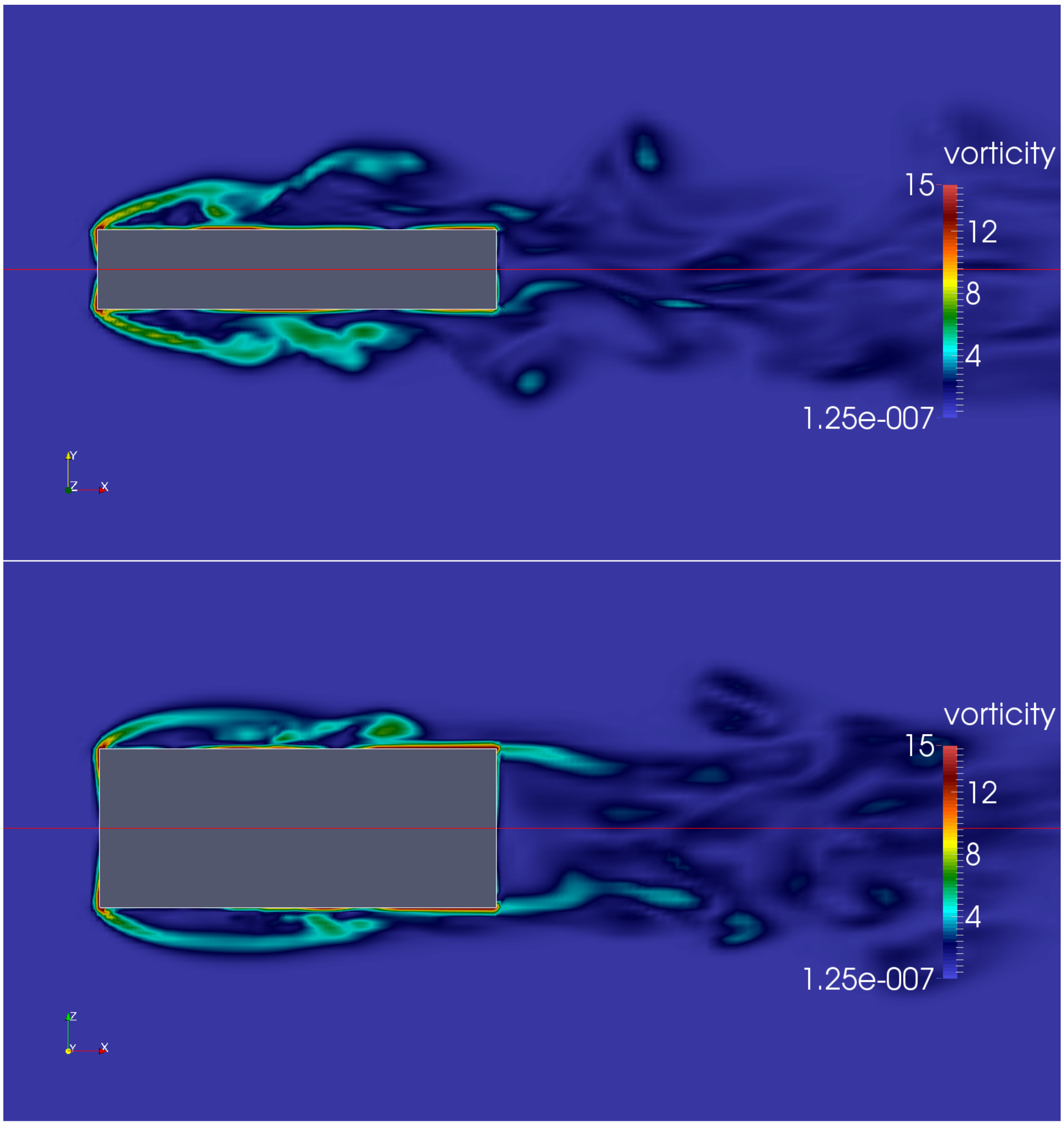

Figure 6.7: Vorticity for RANS double-body: x-y (upper) and x-z (lower)

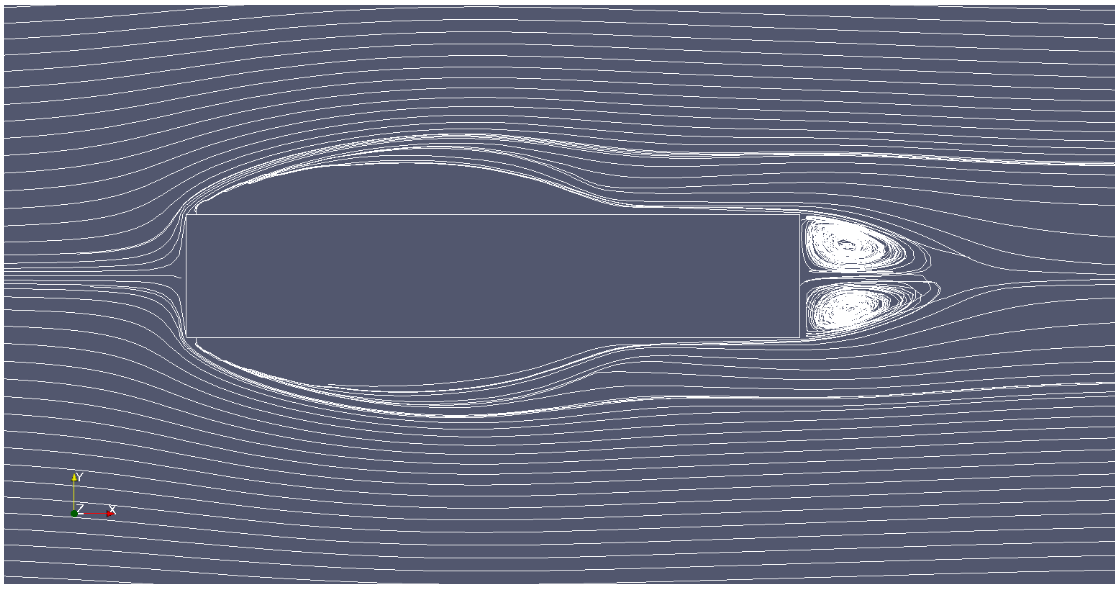

Figure 6.8: Streamlines of mean flow around RANS double-body (x-y plane)

6.2 RANS Floating Body

6.2.1 Force Coefficients and Frequency

Figures 6.9 and 6.10 give plots of drag and lift coefficients, respectively. Figure 6.11 gives the spectrum of lift fluctuations. It shows several distinct peaks, meaning that there are several oscillation frequencies present in the data set.

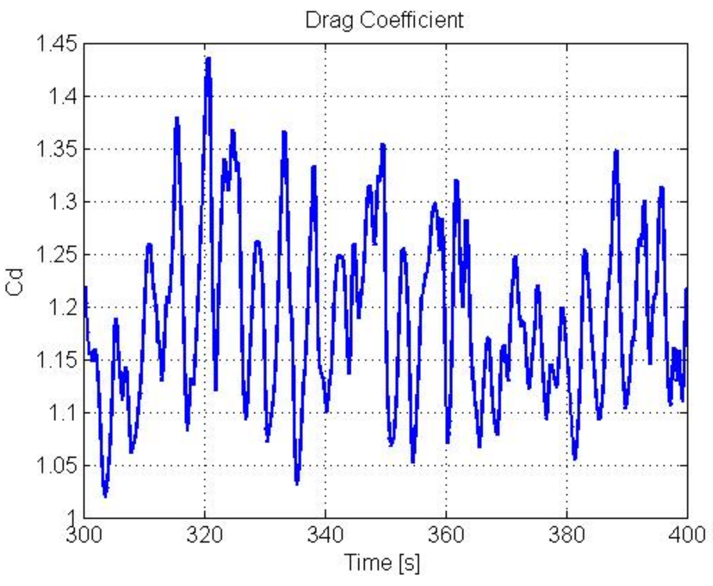

Figure 6.9: Time-history plot of drag coefficient for floating body case

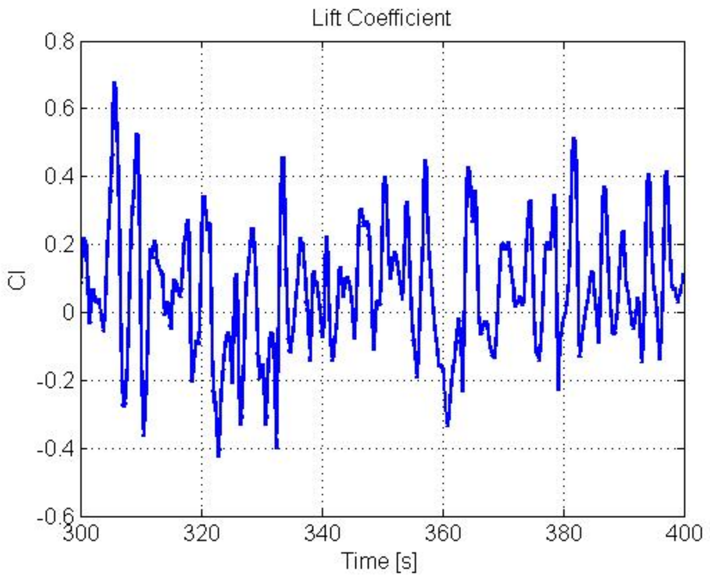

Figure 6.10: Time-history plot of lift coefficient for floating body case

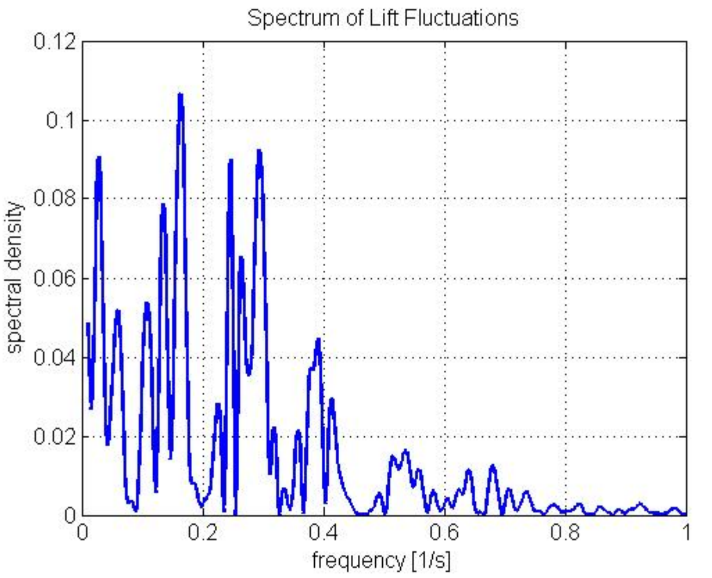

Figure 6.11: Spectrum of lift fluctuations for floating body case

6.2.2 Flow Variable Plots

In figures 6.12, 6.13 and 6.14, snapshots of the flow field around the floating body is shown. Clear vortex shedding can be seen also in this case. It can be seen from figure 6.14, which shows vorticity in both the horisontal and the vertical plane, that the free-slip boundary next to the body greatly influences the tree-dimensionality of the flow. Figure 6.15 shows streamlines of the mean flow. This looks similar to the streamlines from the double-body case.

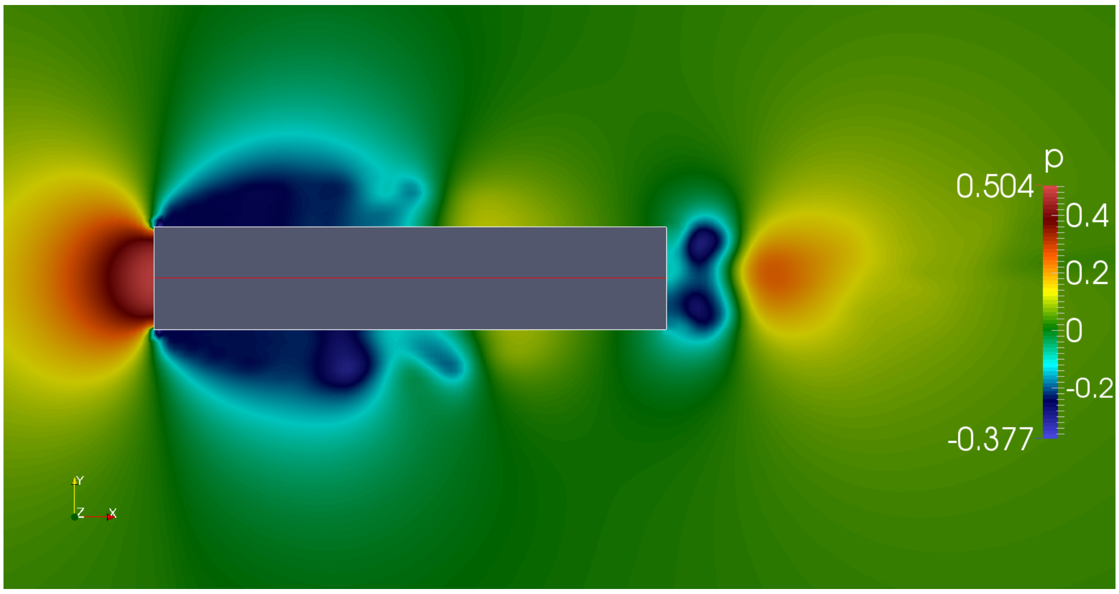

Figure 6.12: Pressure for floating body case (x-y plane)

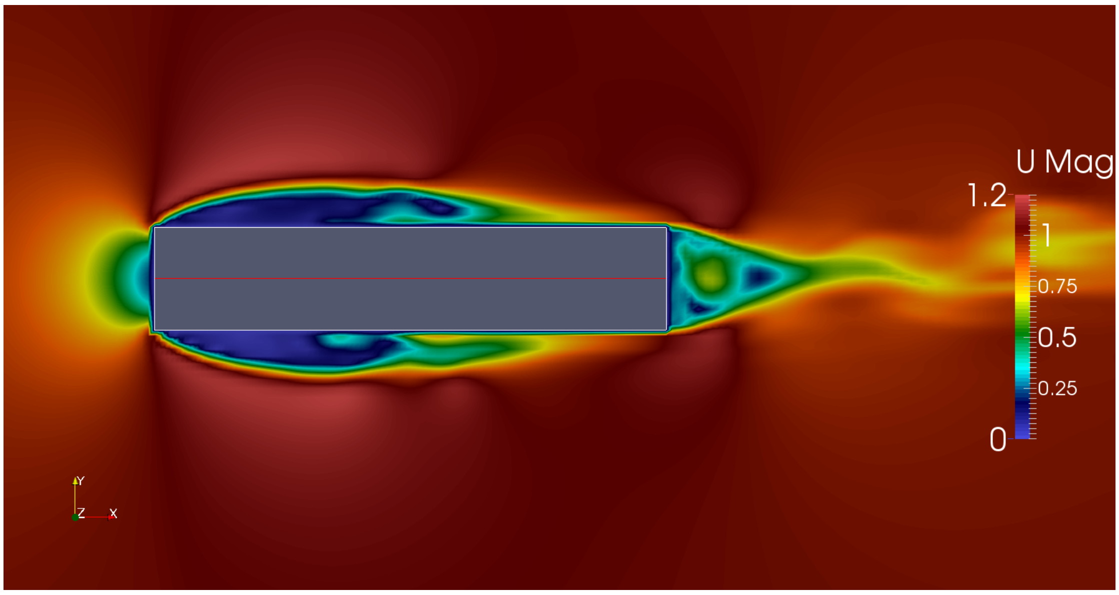

Figure 6.13: Velocity for floating body case (x-y plane)

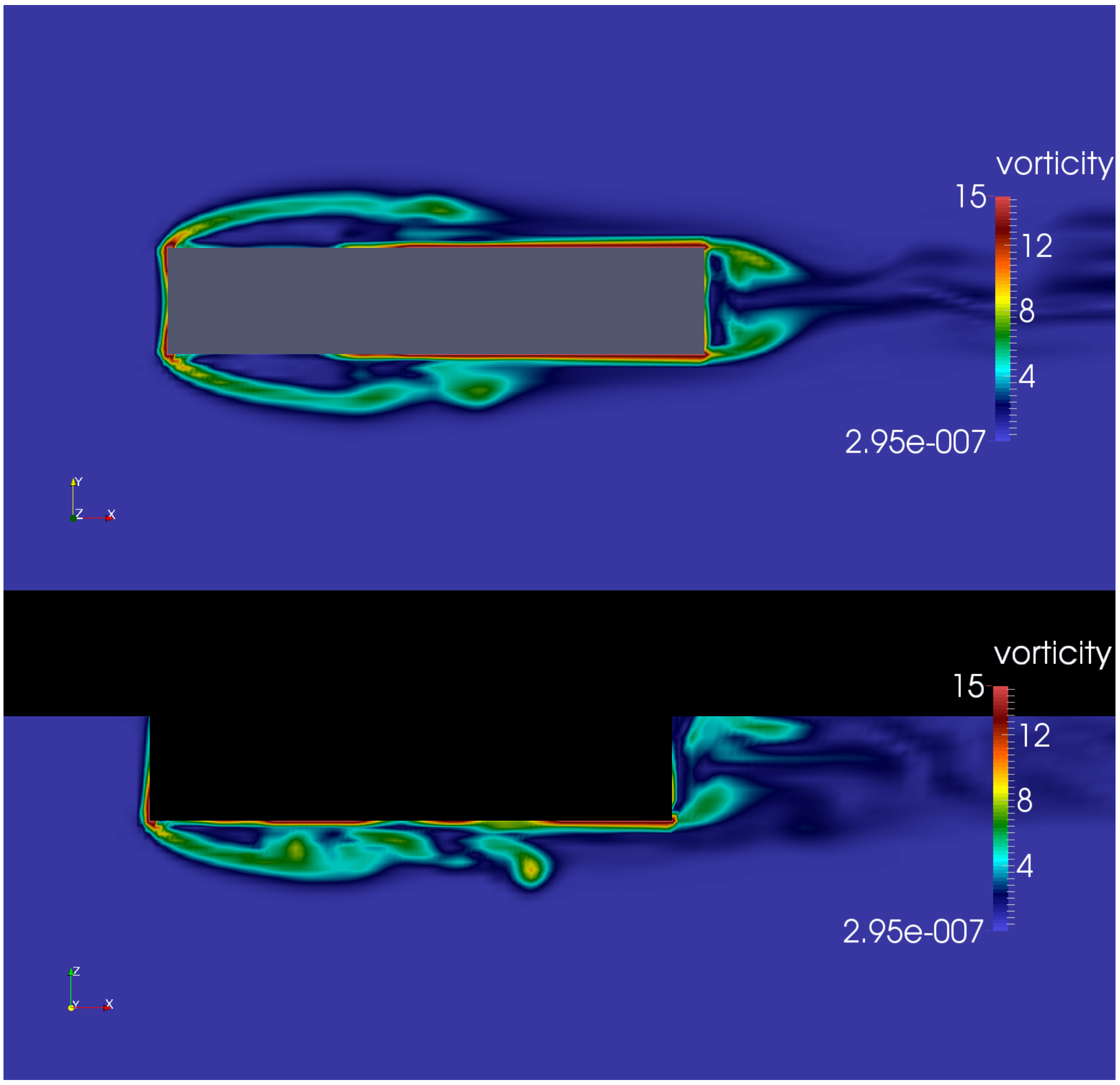

Figure 6.14: Vorticity for floating body case: x-y (upper) and x-z (lower)

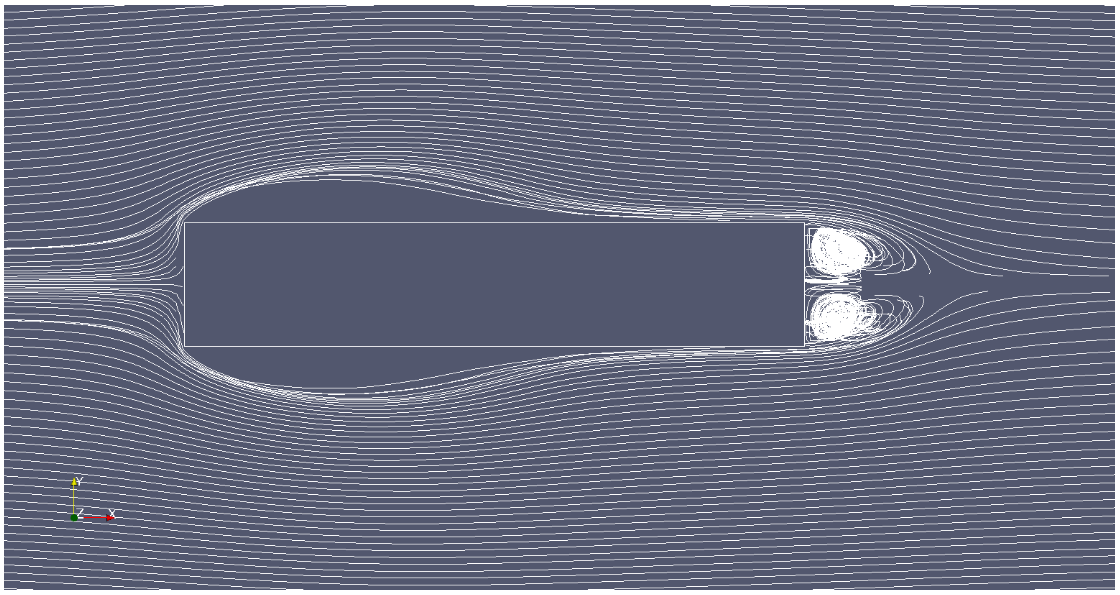

Figure 6.15: Streamlines of mean flow around floating body (x-y plane)

6.3 LES Double-Body

6.3.1 Force Coefficients and Frequency

Lift and drag coefficients and spectral analysis of the LES case can be seen in figures 6.16, 6.17 and 6.18, respectively. It can be seen from the force curves that the solution does not have full statistical convergence. There was unfortunately not enough time to continue the LES simulation further. The oscillations in the drag curve seems to enter a somewhat stabilizing trend around 80-100 seconds. However, the oscillations of the lift curve does not seem to have stabilized at all, at least not with respect to the extreme values. The results before 100 seconds have been omitted when calculating mean drag and standard deviation of lift.

The spectral analysis plot has one dominant peak, meaning there is one dominant oscillation frequency. This can also be argued from the lift coefficient plot. Comparing the spectral analysis of the LES case with the spectral analysis of the RANS cases (figure 6.4 and 6.11), it should be noted that the value of the peak frequency is quite different in the LES case. This is due to that the velocity of the flow was lowered in the LES case (although the Reynolds number is the same) to allow larger time steps and hence shorten simulation time.

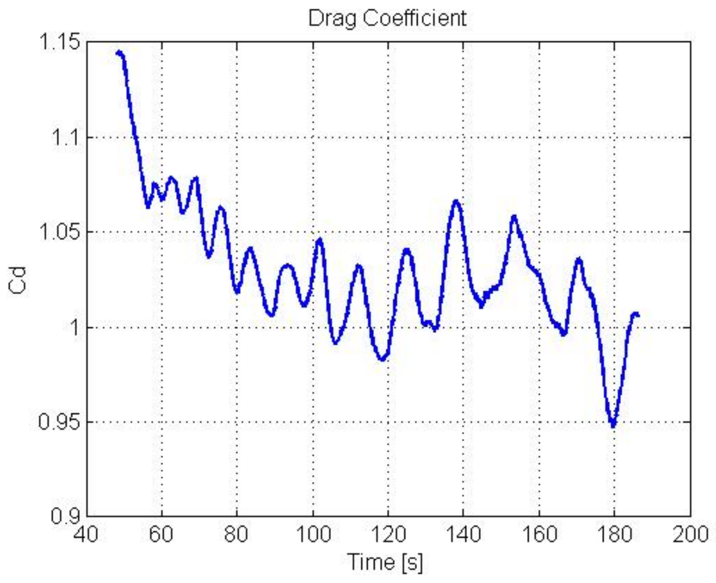

Figure 6.16: Time-history plot of drag coefficient for LES double-body case

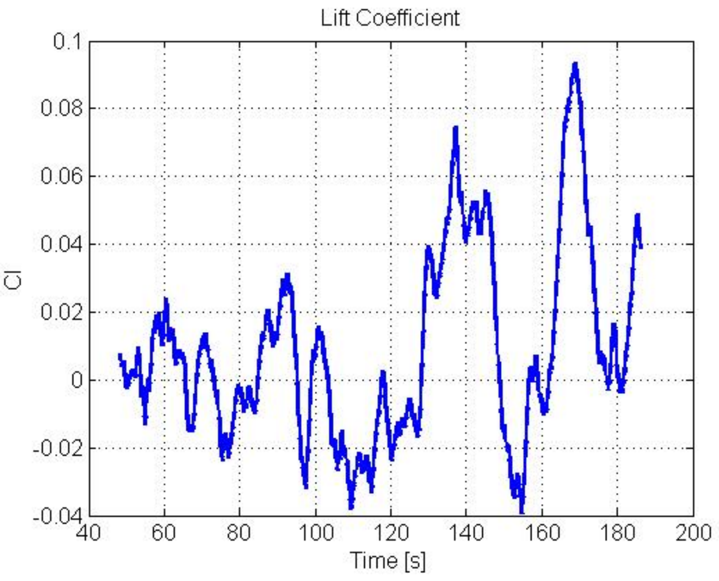

Figure 6.17: Time-history plot of lift coefficient for LES double-body case

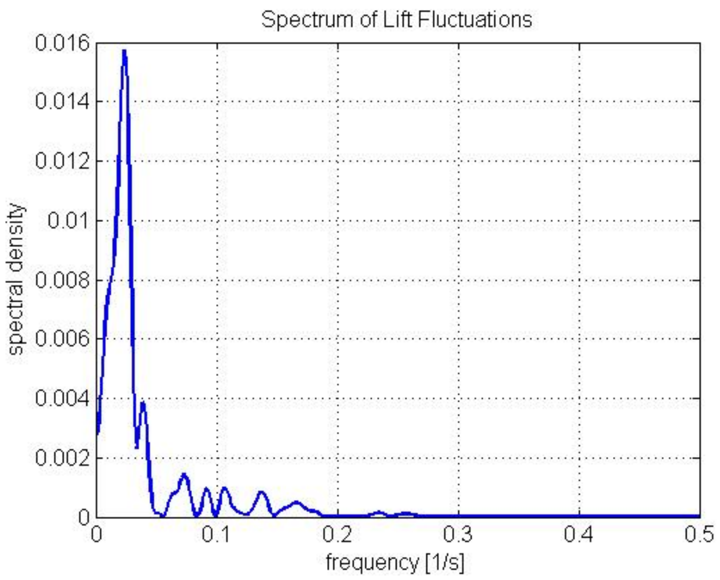

Figure 6.18: Spectrum of lift fluctuations for LES double-body case

6.3.2 Flow Variable Plots

The figures on the next three pages show the flow field around the double-body in the LES case. Figure 6.19, 6.20 and 6.21 show snapshots of pressure, velocity and vorticity, respectively. It is clear from the figures that vortex shedding is captured by the simulation, and that the flow field is very turbulent and irregular. Figure 6.22 is a plot of the subgrid-scale viscosity ν_{SGS}. This was included due to interest in the LES turbulence model. The subgrid-scale viscosity is the viscosity of the small scale motions (see chapter 2.6.3). Hence, the field in figure 6.22 tells us where the small scale motions are present. It can be seen that they are maintained far behind the body. No streamline plot has been included for the LES case since the solution had not converged well enough, and the mean flow field would not be entirely reliable.

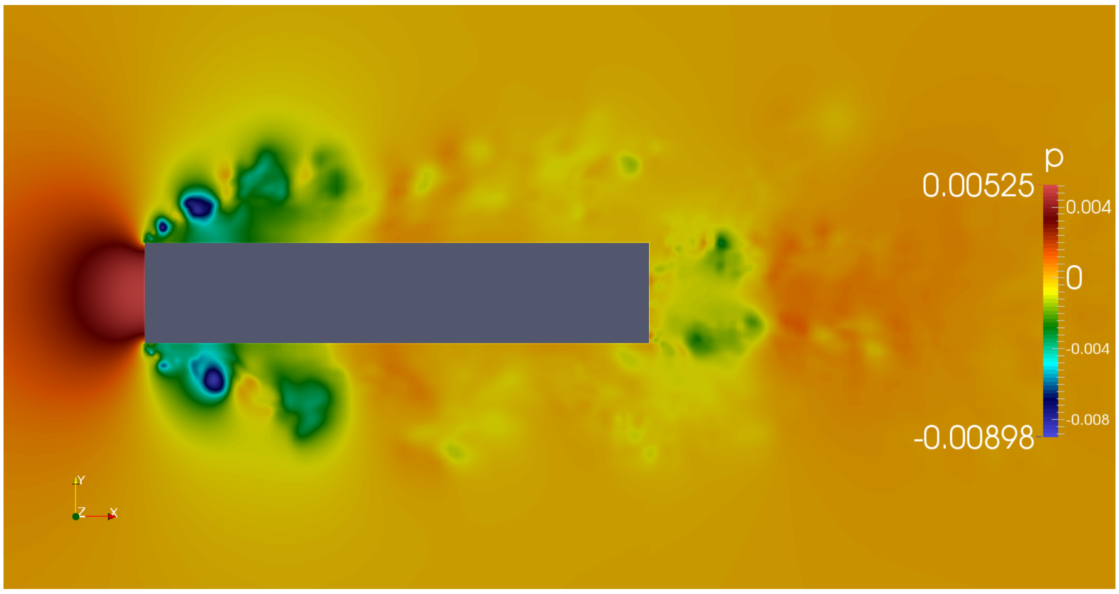

Figure 6.19: Pressure for LES double-body case (x-y plane)

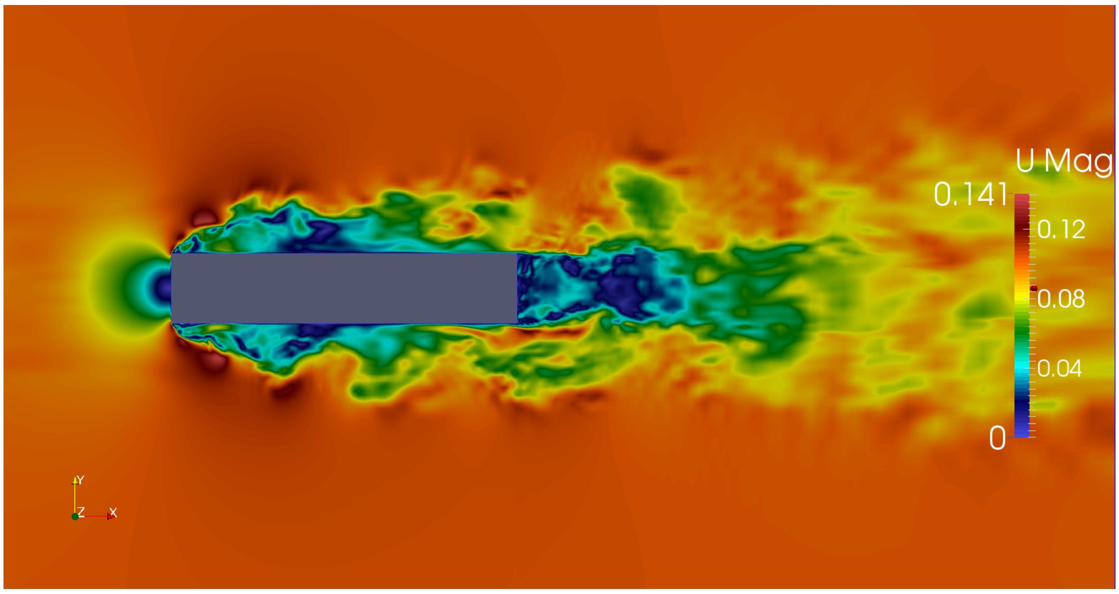

Figure 6.20: Velocity for LES double-body case (x-y plane)

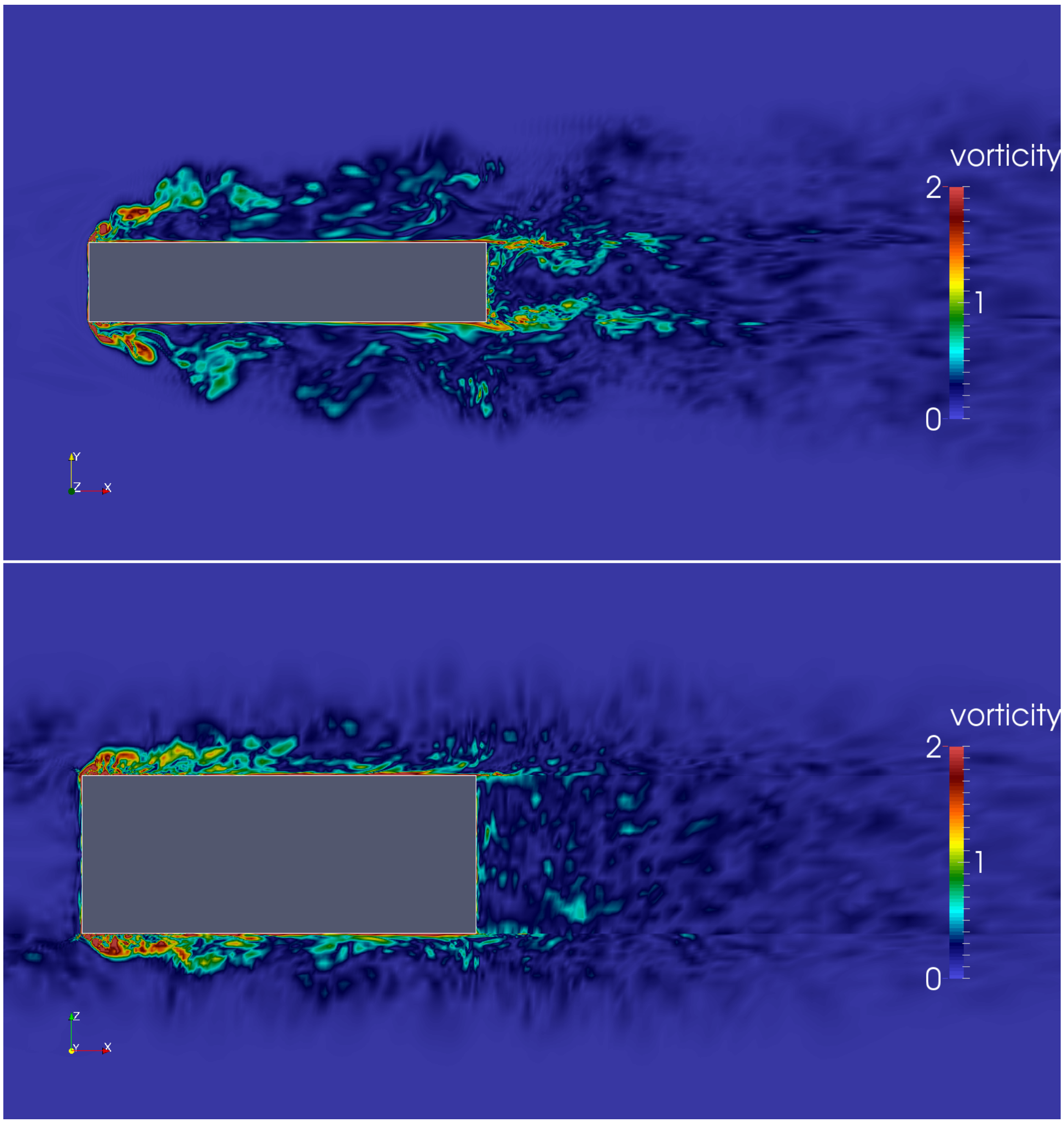

Figure 6.21: Vorticity for LES double-body case: x-y (upper) and x-z (lower)

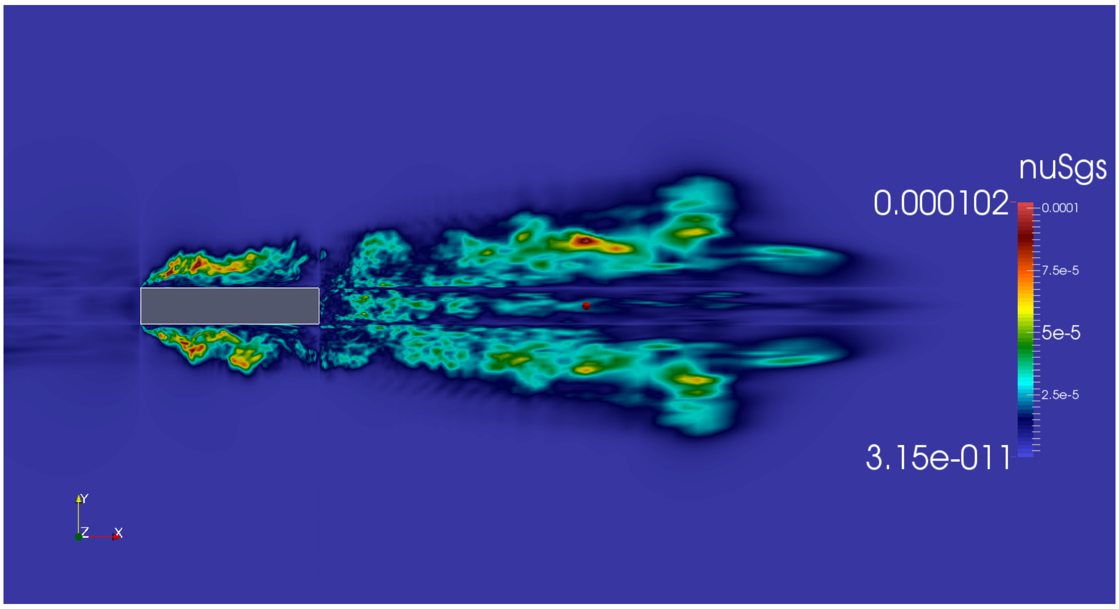

Figure 6.22: Subgrid-scale viscosity for LES double-body case (x-y plane)

7 Discussion and Comparison of Results

A summary of the results from all three cases can be seen in table 7.1 below. When comparing the cases, there are two main focuses. The first is to compare the double-body case to the floating body case. This is to see what kind of domain discretisation is best in this case. Only RANS results are of interest in this comparison, since no floating body LES was performed.

The second focus is to compare the two double-body cases, RANS and LES. This is to see the differences from using these two turbulence models. The RANS model is relatively fast, while the LES model is considerably more demanding. It is interesting to see the differences in results compared to the differences in computational effort. It would be valuable to be able to draw conclusions on what turbulence model is best suited for this case.

Table 7.1: Summary of results of each case

Case	mean Cd	stdev Cl	St	average y^+
Double-Body (RANS)	0.8751	0.0799	0.1430	27.0964
Floating Body (RANS)	1.1841	0.1828	0.1640	27.9287
Double-Body (LES)	1.0149	0.0331	0.2400	1.08698

7.1 Grid Convergence

There is a relatively large difference in cell count between the three grids used in the sensitivity study. This probably means that if the results do not deviate too much, the medium grid is fine enough. How much difference in results can be tolerated is dependent on the difference between grids and the variable in question. It is assumed that the differences in force coefficients and shedding frequency here are acceptable considering the significant difference in fineness of the grids. The medium grid is therefore used for all final results. It should be noted that this is a subjective observation. The conclusion depends on what accuracy is needed. If very high accuracy is needed in finding the force coefficients, then one might consider expanding the grid sensitivity study, to be even more certain in what grid to use.

7.2 Double-Body vs. Floating Body

When comparing the two RANS cases, there is one especially distinct difference; the force coefficients. The standard deviation of lift is increased by almost 130% in the floating body case compared to the double-body case. Also the mean drag coefficient is increased, although by a more modest 35%. Arslan et al. (2011) also reports higher drag for a similar floating body case compared to a long cylinder. The main difference between the two cases lies in the distance to the free-slip boundary, hence this must be used to explain the difference in results. In the floating body case, the free surface (free-slip) boundary is adjacent to the body. This is more restrictive on the flow around the body compared to the double-body case, where the boundaries are further away. It is reasonable to think that these boundary induced restrictions will force the flow towards the body, since it cannot move through the boundary. From this, it is reasonable that the close boundary in the floating body case increases the forces on the body in general, giving the difference in results seen in table 7.1.

Looking at the Strouhal number, the difference from double-body to floating body is an increase of 14.7%. This is a notable difference, however it is not critically large (considerably lower than the differences in force coefficient). This indicates that the flow field and the vortex shedding in general should not have changed too much between the two cases. Comparing figures 6.5 and 6.12 for pressure, and 6.6 and 6.13 for velocity, is is obvious that there are some differences. It should, however be noted that these figure are merely snapshots of the instantaneous flow field, and it is questionable to use them in a general comparison. The streamline plots of figure 6.8 and 6.15 are more applicable for comparison since they represent the mean flow. There are no very distinct differences to spot between these two figures. This again implies that the general nature of the flow fields in the two cases may be quite similar. The most notable difference lies in the trailing edge vortices, although it should be noted that there is some uncertainty related to the creation of these figures, since the applied software paraFoam required some manual effort to produce the streamlines. The size of the leading edge vortices on the streamline figures seem to be very similar in size. The vortices themselves are not in the figures because they proved too difficult to reproduce in paraFoam.

Comparing the two plots of pressure (figures 6.5 and 6.12) and the streamline plots (figures 6.8 and 6.15), some other observations can be used to explain the higher lift coefficient in the floating body case. It may seem that the flow in the double-body case is "more separated" than in the floating body case. In the double-body case, the vortices seem to move further away from the body than in the floating body case and there is a larger low-pressure region around the vortex shown for the floating body. Also, the vortices of the floating body case seem slightly larger when comparing the pressure plots. This is not seen in the streamline plots though. Looking at the streamline plot of the double-body case, many of the streamlines from the leading edge never reattach to the body. This is not the case for the floating body streamlines. All this indicates that the increased lift forces in the floating body case is due to a "more attached" flow, that is caused by the restrictive close boundary which forces the fluid towards the body.

There are also some notable differences between the two cases in the instantaneous velocity plots (figures 6.6 and 6.13). The double-body field seems more disordered than the floating body field. This observation may also be seen from the vorticity plots (figures 6.7 and 6.14), although not as distinct. The more disordered nature of the flow around the double-body may cause the extreme values of lift coefficient to be lowered, because there may be significant flow effects pulling in both directions. It should again be underlined that the plots of instantaneous flow variables should be compared with caution, since they don't represent the mean flow. It is, however assumed that the their general nature is representative of the flow. Instantaneous field plots at different time steps have been checked to ensure that the differences shown here are not random.

Looking at the two spectral analysis plots (figures 6.4 and 6.11), the difference is quite significant. The double-body flow has one dominant peak in frequency, while the floating body flow has several peaks of similar magnitude. It is not straight forward to understand why this is the case. The sample interval is the same in both cases. Although they don't explain the phenomenon, a comparison of the pressure plots may show the same as the spectral analyses. The pressure plot of the floating body case (figure 6.12) has a trailing edge vortex with similar magnitude as the leading edge vortex. Both vortices oscillate, but not necessarily with the exact same frequency. This may explain the additional peaks in the spectral analysis of the floating body flow. In the double-body flow, the leading edge vortex seems to be much more significant than the trailing edge vortex (see figure 6.5), explaining why only one clear peak is seen in the spectral analysis.

7.3 RANS vs. LES

Comparing the results of the two double-body simulations, it is clear that there are great differences between RANS and LES methods. The pressure and vorticity plots of the LES (figure 6.19 and 6.21) show that the vortices and turbulent structures are better retained downstream than with RANS (figure 6.5 and 6.7). It can also be seen that the LES flow field is generally more turbulent than the RANS flow field, which is more ordered. This is seen especially when comparing figures 6.6 and 6.20.

Since the LES case has not achieved full statistical convergence, it is not straight forward to compare the force coefficients and spectral analyses. However, since the drag coefficient in figure 6.16 seems to be relatively close to statistical convergence, the mean value obtained here is assumed to be accurate enough for a somewhat reliable comparison. The drag is about 16% higher with LES than with RANS. Higher drag with LES than with RANS was also reported by Rodi (1997), which case is similar, although in 2-D. This implies that RANS methods with wall functions are not always very accurate in predicting the values of integrated results like force coefficients. Since no RANS simulation without wall functions was performed in this project, it is somewhat uncertain if the differences in results are caused mostly by the turbulence models or the near-wall modelling. The assumptions behind wall functions are questionable in separated flow regions (Rodi, 1997), implying that the wall functions may be causing inaccuracies in this case.

The standard deviation of lift is much lower with LES than with RANS. This contradicts the results of Rodi (1997). Looking at figure 6.17, it is clear that the standard deviation of lift coefficient in the LES case is increasing with time. It can be assumed that the LES lift oscillations will approach the RANS results if the simulation is continued. It is unfortunate that there was not enough time to run it as long as the RANS case.

Strouhal number was 68% higher with LES than with RANS. From the results of Rodi (1997), it was expected that the two results would be closer. Rodi (1997) also states that the vortex shedding in the LES case is generally less regular than in the RANS case. This may be backed up comparing the plots of velocity and vorticity (figures 6.6 and 6.20, and 6.7 and 6.21). However, comparison of the spectral analyses (figures 6.4 and 6.18) shows that the peak frequency is more dominant in the LES case, implying that the shedding may be more regular for LES. These contradictory results again show how the lift and spectral analysis results of the LES are unsuitable for a realistic comparison to the RANS results.

It was mentioned in chapter 6.3.1 that the velocity in the LES case was lowered to attempt to reduce simulation time. In hindsight, this was probably not a very good approach. Lower velocity may allow larger time steps, but the larger the time step, the more iterations the solver will need at each time step. Also, the flow will be slower, and hence the total needed simulation time will be longer. These two effects probably cancel out the effect of larger time step, making the approach more or less ineffective. It would be easier to just have the same velocity in all cases, especially for post-processing.

8 Conclusions

Comparing the two geometric cases, double-body and floating body, it is clear that there are some noteworthy differences in the resulting flow. The simulations have shown that the restrictive free surface boundary in the floating body case causes increased forces on the body. If the double-body case is assumed to be the more realistic flow, since it is less influenced by restrictive boundaries, the floating body case may be too inaccurate for calculating forces. When it comes to computational effort, the floating body approach is of course faster, since the grid is half the size. Hence, this approach could be preferred for cases where computational capacity is very limited. The restrictions imposed by the inaccurate free surface modelling should then be taken into account, or the results will probably be inaccurate to some degree.

To make a definite conclusion as to which geometric case is the best, a more accurate floating body case with actual free surface modelling would be required. It is the impression of the author that such a simulation will be closer to the double-body case in results, although this cannot be said for sure based on the results of this project. There are many different methods for more realistic free surface modelling. Multiphase flow with both air and water is possible in OpenFOAM, and this would probably be a good idea to implement in further work.

The results of the simulations show that LES is generally better suited for describing fully turbulent and massively separated flows like the one in question here. This conclusion is mainly based on the visualization of the flow fields, which show that LES captures the vortices and turbulent structures better, and is able to follow them further downstream. Unfortunately, the force coefficient and spectral analysis results of the LES in this project are too inaccurate to compare to the RANS results with certainty. Therefore, no certain conclusions can be drawn from this, other than that the RANS approach with wall functions seems to under predict the mean drag coefficient. It would be interesting to run a RANS simulation with a finer grid without wall functions to see if this will resemble the LES results more. This is also recommended as further work.

When choosing between LES and RANS, computational effort has to be taken into account. It was found from this project that LES is very expensive compared to RANS, with respect to both computational time and preprocessing. More advanced gridding techniques like unstructured grids and automatic gridding could help reduce the effort of LES. The lesson learned here is that for practical engineering purposes, LES should only be chosen over RANS if it is strictly necessary for accuracy of results. RANS is simple and fast, and probably accurate enough for most practical applications.

It was interesting to note that the $k-\omega$ SST model was not well suited for this application. This contradicted the authors previous experience with simpler flows. The realizable $k-\varepsilon$ model turned out to be working much better for this problem. It might seem that the combination of $k-\omega$ SST and wall functions is not always a good choice, although this cannot be concluded purely based on the results of this project.

OpenFOAM has proven to be well suited for the simulations in this project. Different turbulence models are relatively easy to implement, and documentation and support are widely available on the web. The structure and interface of the program may be a bit hard to grasp for new users, but it opens up for a thorough understanding of both inputs and outputs.

This master's project has laid the ground works for being able to simulate flow around more complex geometries in the future. The author is now many steps closer to the ship hull case, which was one of the initial motivations behind this project. Even though no real ship hull geometry was investigated here, knowledge about turbulence models and gridding techniques will be very valuable in future endeavours on this topic.

References

Tufan Arslan, Bjornar Pettersen, and Helge I Andersson. Calculations of the flow around rectangular shaped floating structures. In *ICWE 2011, 13th International Conference on Wind Engineering, Amsterdam*. Norwegian University of Science and Technology, 2011.

Jonas Bredberg. On the wall boundary condition for turbulence models. Report, Department of Thermo and Fluid Dynamics, Chalmers University of Technology, 2000.

. CFD Direct. *OpenFOAM User Guide*. 2015. URL `http://cfd.direct/openfoam/user-guide/`. Accessed: 26.05.2016.

. CFD Online. *Meshing*. 2012. URL `http://www.cfd-online.com/Wiki/Meshing`. Accessed: 27.05.2016.

Rojko Dejhalla and Jasna Prpic-Orsic. A review of the state-of-the-art in marine hydrodynamics. *Brodogradnja*, 57(1):13–22, 2006. ISSN 0007215X.

L. Eça and M. Hoekstra. On the grid sensitivity of the wall boundary condition of the k-omega turbulence model. *Journal of Fluids Engineering, Transactions of the ASME*, 126(6):900–910, 2004. ISSN 00982202. doi: 10.1115/1.1845492.

Joel Guerrero. *Tips and tricks in OpenFOAM*. University of Genoa, DICCA, URL http://www.dicat.unige.it/guerrero/oftraining/9tipsandtricks.pdf, 2015.

Volker John. *Large eddy simulation of turbulent incompressible flows : analytical and numerical results for a class of LES models*, volume 34 of *Lecture notes in computational science and engineering*. Springer, Berlin, 2004. ISBN 3540406433.

William Layton. The 1877 boussinesq conjecture: Turbulent fluctuations are dissipative on the mean flow. Report, Department of Mathematics, University of Pittsburgh, 2014.

Claudio Mannini, Ante Soda, and Günter Schewe. Unsteady rans modelling of flow past a rectangular cylinder: Investigation of reynolds number effects. *Computers and Fluids*, 39(9):1609–1624, 2010.

Claudio Mannini, Ante Soda, and Günter Schewe. Numerical investigation on the three-dimensional unsteady flow past a 5:1 rectangular cylinder. *Journal of Wind Engineering and Industrial Aerodynamics*, 99(4):469–482, 2011.

Florian R. Menter. Zonal two equation kappa-omega turbulence models for aerodynamic flows - nasa-tm-111629. In *24th Fluid Dynamics Conference, Orlando, Florida*. Sponsoring Organization: NASA Ames Research Center, 1993.

J. N. Newman. *Marine hydrodynamics*. MIT Press, Cambridge, Massachusetts, 1977. ISBN 0262140268.

J.S. Ochoa and N. Fueyo. Large eddy simulation of the flow past a square cylinder. Report, Fluid Mechanics Group, University of Zaragoza, Spain, 2015.

Muk Chen Ong. Unsteady rans simulations of flow around a 5:1 rectangular cylinder at high reynolds numbers. In *Proceedings of the ASME 2012 31st International Conference on Ocean, Offshore and Arctic Engineering*. Department of Marine Technology, Norwegian University of Science and Technology, and MARINTEK, 2012.

Bjornar Pettersen. *Marin Teknikk 3 Hydrodynamikk*. Department of Marine Technology, NTNU, Trondheim, Norway, 2007.

Stephen B. Pope. *Turbulent flows*. Cambridge University Press, Cambridge, 2000. ISBN 0521591252.

Orlando Rivera and Karl Furlinger. Parallel aspects of openfoam with large eddy simulations. In *2011 IEEE International Conference on High Performance Computing and Communications*, pages 389–396. Leibniz Supercomputing Centre (LRZ) and Ludwig-Maximilians-Universitat (LMU), 2011. ISBN 978-1-4577-1564-8. doi: 10.1109/HPCC.2011.57.

W. Rodi. Comparison of les and rans calculations of the flow around bluff bodies. *J. Wind Eng. Ind. Aerodyn.*, 71:55–75, 1997. ISSN 0167-6105.

Pierre Sagaut. *Large eddy simulation for incompressible flows : an introduction*. Introduction a la simulation des grandes echelles pour les ecoulements de fluide incompressible, mathematique et applications. Springer, Berlin, 3rd ed. with a foreword by charles meneveau. edition, 2006. ISBN 3540263446.

Tsan-Hsing Shih, William W. Liou, Aamir Shabbir, Zhigang Yang, and Jiang Zhu. A new k-epsilon eddy viscosity model for high reynolds number turbulent flows. *Computers and Fluids*, 24(3):227–238, 1995. ISSN 00457930. doi: 10.1016/0045-7930(94)00032-T.

J. Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment*. *Monthly Weather Review*, 91(3):99–164, 1963. ISSN 0027-0644. doi: 10.1175/1520-0493(1963)091¡0099:GCEWTP¿2.3.CO;2.

H. Tennekes and John L. Lumley. *A first course in turbulence*. MIT Press, Cambridge, Mass, 1972. ISBN 0262310902.

. The OpenFOAM Foundation. *Numerical Methods*. 2016. URL `www.openfoam.org/features/numerical-method.php`. Accessed: 19.02.2016.

John F. Wendt. *Computational Fluid Dynamics*. Springer Berlin Heidelberg, 2009. ISBN 9783540850564.

Frank M. White. *Viscous fluid flow*. McGraw-Hill series in mechanical engineering. McGraw-Hill Higher Education, Boston, 3rd ed. edition, 2006. ISBN 9780072402315.

David C. Wilcox. Reassessment of the scale-determining equation for advanced turbulence models. *AIAA journal*, 26(11):1299–1310, 1988. ISSN 00011452.