University of Twente

Internship at CIRA

Numerical Simulation over a backward facing step with OpenFOAM

Author:
Arjan Slaper

CIRA supervisor:
Dr. E. Iuliano

UT supervisor:
Prof. Dr. H.Hoeijmakers

August 24, 2014

Preface

From February to May 2014, I have done my internship at CIRA in Italy for my Mechanical Engineering master in Engineering Fluid Dynamics. Over these couple of months I have experienced an entirely different working and living environment. I enjoyed it very much and learned a lot.

I would like to thank professor Hoeijmakers for providing me this opportunity and the support for this internship from the Netherlands. From CIRA, I would specially like to thank dr. Emiliano Iuliano for all the help and guidance at CIRA and with the assignment but also for all the help and support outside of the working environment. Furthermore I would like to thank my colleagues from CIRA and friends for making this internship a wonderful experience.

Arjan Slaper

Contents

List of Figures

List of Tables

1 Introduction

1.1 CIRA

The Centro Italiano Ricerche Aerospaziali or CIRA is the largest aerospace research facility in Italy. It was founded in 1984 and is a corporation with private and public shareholders who are all interested in the same goal: improvement in aeronautical and space engineering. Together with the European Space Agency, they managed to the build many unique testing facilities as for example an icing wind tunnel and a plasma wind tunnel. There are currently about 350 people working for CIRA.
The icing wind tunnel of CIRA is the biggest icing wind tunnel in the world. There are different test sections which can be used to test either scale models or even complete parts of an airplane at altitudes up to 7000 meters, speeds of 800 km/h and temperatures up to -40 degrees Celsius.
The plasma wind tunnel is used to test the heat protection systems of re-entry vehicles. Objects with a diameter up to 2 meter can be put into the test section where temperatures up to 10.000 degrees Celsius can be reached. Because of the size of the objects and conditions the objects can be tested in, this facility is unique in the world.
At CIRA, there are also testing facilities for testing safety systems during air crashes as well as transonic and hypersonic wind tunnels. With all these facilities, it is possible for CIRA to do a wide variety of tests for all kinds of aerospace companies.
Besides all these testing facilities, CIRA also has a devision for computational fluid dynamics. All kinds of problems are examined here from icing problems to optimization problems as well as calculations on the re-entry vehicle which is currently under development.

1.2 Turbulence

When people are asked about turbulence, many will think about airplanes and the uncomfortable situations turbulence causes during flight. It will take a lot more thinking for most people to realize that turbulence is everywhere around us. The path of smoke coming from a cigarette or chimney, the flow of water from a fountain and the weather are all examples of situations where turbulence is encountered in our daily lives.

Figure 1.1: *Turbulence in a wind turbine park [20]*

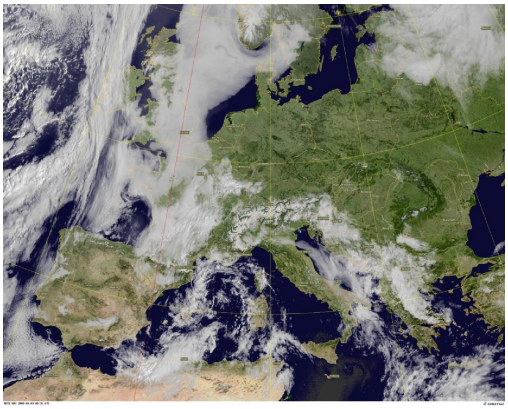


Figure 1.2: *Turbulence found on the weather chart [21]*

Because turbulence is everywhere around us, it can play an important role in design processes. Optimization is an important aspect of our daily lives since we want to go higher, faster and cheaper to reach our destinations and perform our processes. The understanding of turbulence is becoming a more important subject in these matters and it is Mother Nature who proves to be a though opponent of humans in their constant battle for improvement.
Turbulence is a subject widely studied by research facilities, universities and industry in order to get a better understanding of processes and phenomena and to ultimately develop models to predict the behaviour of turbulent flow. A turbulent flow is a very effective way to transport for example heat and with better understanding, it could be applied in all kinds heat transport applications. The precise cost of our inability to predict turbulence is hard to calculate, but if you think about the previous given examples, it can be imagined that the costs are very high in terms of efficiency for example.

The main predictor of turbulent behaviour is the Reynolds number. The Reynolds number (as defined in 1.1) is the relation between the viscous forces and the inertial forces. In low Reynolds number flows the flow is laminar and the viscosity plays an important role, while in high Reynolds number flows the influence of the viscous forces is very small in comparison to the inertial forces and the flow becomes turbulent.

$$Re = \frac{\rho U L}{\mu} \tag{1.1}$$

When a turbulent flow is studied, the structures observed are irregular both in space and time (see the structures in figures 1.1 and 1.2 as examples). This irregularity causes a lot of problems in the calculations of turbulent flow problems and the design processes. The turbulent structure at one time instant is very chaotic and this behaviour does not improve if the time is advanced. A turbulent flow is different at every spatial place and time instant. Because of this behaviour, it is very difficult to predict where a certain particle will end up. With experiments it was tried to find the flow path of a particle in a turbulent flow. It turned out that it was impossible to predict the exact location because the trajectory of the particle depends both on its position in space as well as in time. A small change in the initial condition will result in a totally different end position of the particle. The approach of trying to predict the path of each individual particle did not work out for a design process. It turned out, however, that it was possible to extract some statistics from the flow. These statistics, in the form of probability density functions, could be used to extract general behaviour of the flow particles and of the flow in general. When a turbulent flow field at a certain time instant is studied more carefully, it can be seen that it contains all kinds of eddies which contain eddies within itself. Lewis Richardson summarized the process of a turbulent flow in one of his papers written in 1922 as:

Big whorls have little whorls
That feed on their velocity,
And little whorls have lesser whorls
And so on to viscosity.

This poem describes the energy cascade through all the length scales up to a length scale where the viscosity becomes dominant again and the energy dissipates. The energy of each size of eddies can be plotted as function of the wave length as is done in figure 1.3. It shows that the big eddies contain the most energy and that the amount of energy of the flow decays through the flow regime. It was Kolmogorov who defined the length scale in the turbulent process at which the energy is dissipated through the viscosity. He stated that the bigger scales are not dependent of the viscosity and only dependent on the energy dissipation. The smallest eddies dependent on the dissipation and the viscosity. These statements were presented in 1941 so that is why this theory is called the K41 hypothesis. He also stated that the different eddy lengths all behave in a similar fashion and that scale similarity can be used in order to predict certain behaviours for certain eddy sizes. This similarity hypothesis describes that the energy is only dependent of the dissipation ratio and not of the viscosity for the larger sized eddies. The smallest scales are influenced by the dissipation as well as the viscosity.

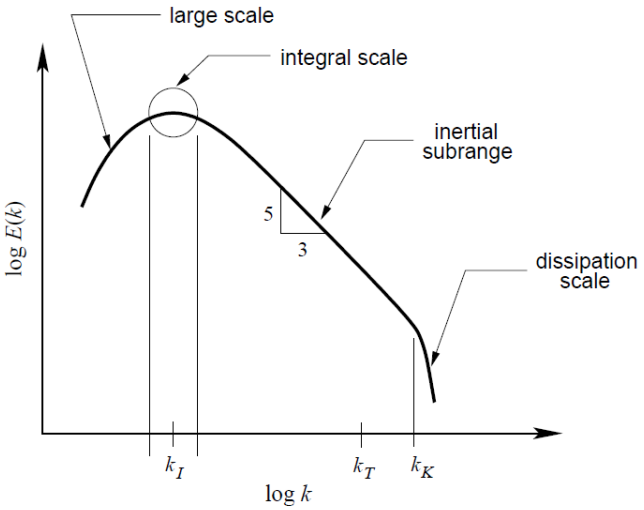


Figure 1.3: *Energy of a turbulent flow [4]*

In order to model the turbulent behaviour, an assumption has to be made on the behaviour of the stress terms in the equations used to model the flow problems (Navier Stokes equations, see also chapter 2). It turns out that the stresses are non linear and given that fact, it is difficult to model the stresses in a simple way. In 1877 Joseph Boussinesq was the first to say that the momentum transfer in a turbulent flow can be modeled by the use of an eddy viscosity. The Boussinesq hypothesis states that the stress tensor is proportional to the mean rate of strain. This hypothesis is given in 1.2 and forms the basis for many models.

$$\tau_{i,j} = 2\mu_t S^*_{i,j} - \frac{2}{3}\rho k \delta_{i,j}$$
$$-\overline{u'_i u'_j} = \nu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{2}{3}\frac{\partial U_k}{\partial x_k}\delta_{i,j} \right) - \frac{2}{3} k \delta_{i,j} \tag{1.2}$$

In the second half of the 20th century the computer power increased rapidly. This resulted in many new possibilities such as the moon landing but also the research in a lot of different fields including turbulence. The research in turbulence got a boost because the computer could be used to help the understanding of turbulence. Many different models and methods were developed in order to try and predict turbulent behaviour in all kind of different environments and designs. These models were tested using various different test cases and setups.

Before the computer was used in design processes, scale models and full-scale models were used to check whether or not design specifications were met. It took a long time to develop a model, test it, improve it and test it again until a satisfying result is found. In modern design processes, the computer is playing an increasingly important role, because the designs can be evaluated more quickly and more precise. A lot of tests can already be performed on a computer in order to obtain a good design. This greatly decreases the time and money needed to be spend on scale models because the models developed will be better so less improvement is necessary. With all the opportunities given by the computer, the demand for better models also increased. The limits of the design are pushed in order to get better results. Turbulence still proves to be a difficult process in order to predict and model. As long as there is not an uniform consistent and accurate model which can be solved within a reasonable time, the research and developments will go on.

2 Fluid flow and modeling

2.1 The Navier-Stokes equations

The Navier-Stokes equations describe the motion of fluids and fluid-like substances. These equations form the basis for every analysis, design or production process of any kind of machine which has something to do with fluid flow. The Navier-Stokes equations (given in integral conservation form in 2.1) are the conservation laws for mass, momentum and energy. In order to apply these equations, it has to be assumed that the fluid is a continuum, uniform and homogeneous of composition.

$$\begin{aligned}
&\frac{\partial}{\partial t}\iiint_V \rho dV + \iint_{\partial V} \rho\left(\vec{u} - u_{\vec{\partial V}}\right)\cdot\vec{n}dS = 0 \\
&\frac{\partial}{\partial t}\iiint_V \rho\vec{u}dV + \iint_{\partial V} \rho\vec{u}\left[\left(\vec{u} - u_{\vec{\partial V}}\right)\cdot\vec{n}\right]dS = \iiint_V \rho\vec{f}dV - \iint_{\partial V} p\vec{n}dS + \iint_{\partial V} \bar{\bar{\tau}}\cdot\vec{n}dS \\
&\frac{\partial}{\partial t}\iiint_V \rho E dV + \iint_{\partial V} \rho E\left(\vec{u} - u_{\vec{\partial V}}\right)\cdot\vec{n}dS = \iiint_V \rho\vec{u}\cdot\vec{f}dV - \iint_{\partial V} p\vec{u}\cdot\vec{n}dS + \iint_{\partial V}\left[\bar{\bar{\tau}}\vec{u}\right]\cdot\vec{n}dS + \iiint_V \dot{Q}dV - \iint_{\partial V}\vec{u}\cdot\vec{n}dS
\end{aligned} \tag{2.1}$$

These equations can also be rewritten into the PDE (partial differential equation) non-conservation form resulting in 2.2.

$$\begin{aligned}
\frac{D\rho}{Dt} + \rho\vec{\nabla}\cdot\vec{u} &= 0 \\
\rho\frac{D\vec{u}}{Dt} &= \rho\vec{f} - \vec{\nabla}p + \vec{\nabla}\cdot\bar{\bar{\tau}} \\
\rho\frac{DE}{Dt} &= \rho\vec{u}\cdot\vec{f} - \vec{\nabla}\cdot(\bar{\bar{\tau}}\vec{u}) + \dot{Q} - \vec{\nabla}\cdot\vec{q}
\end{aligned} \tag{2.2}$$

For the problems described in this report, the main focus will be on the mass and momentum equation and the application of these two equations since the problems described here do not involve heat transport. When these two vector equations are put into the Einstein notation, the following can be obtained:

$$\begin{aligned}
\frac{\partial\rho}{\partial t} + u_i\frac{\partial\rho}{\partial x_i} + \rho\frac{\partial u_i}{\partial x_i} &= 0 \\
\frac{\partial\rho u_i}{\partial t} + \frac{\partial\rho u_j u_i}{\partial x_j} &= \rho f_i - \frac{\partial p}{\partial x_i} + \frac{\partial\tau_{j,i}}{\partial x_j}
\end{aligned} \tag{2.3}$$

In the momentum equation, the term $\frac{\partial\tau_{j,i}}{\partial x_j}$ represents the viscous stresses in the flow and the tensor $\tau_{i,j}$ is assumed to be symmetric. The fluids considered in this report are also considered Newtonian fluids. This assumptions means that the viscous stresses are assumed proportional to the rate of strain resulting in the expression given in 2.4 [1].

$$\tau_{i,j} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.4}$$

With μ being the viscosity

When the equations of 2.3 and 2.4 are combined, a set of equations is found which can be used to solve different types of flow problems.

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + u_i \frac{\partial \rho}{\partial x_i} + \rho \frac{\partial u_i}{\partial x_i} &= 0 \\
\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i}{\partial x_j} &= \rho f_i - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right)\right) \\
&= \rho f_i - \frac{\partial p}{\partial x_i} + \mu\left(\frac{\partial^2 u_j}{\partial x_i \partial x_j} + \frac{\partial^2 u_i}{\partial x_j \partial x_j}\right)
\end{aligned} \tag{2.5}$$

In case of an incompressible flow (which will be the case for the studied cases in this report), the equations reduce to 2.6.

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

And directly applied to the momentum equation:

$$\begin{aligned}
\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} &= f_i - \frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\mu}{\rho}\frac{\partial^2 u_i}{\partial x_j \partial x_j} \\
&= f_i - \frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j}
\end{aligned} \tag{2.6}$$

with ν being the kinematic viscosity

2.2 Modeling of the Navier Stokes equations

To use computer models as an analysis tool, a model is required to give a good approximation of the reality within a reasonable time frame. It is exactly this principle that causes the large variety in models and methods applied in CFD codes. For some purposes it can be good to have a quick solution and the accuracy is not so important while in other cases accuracy is more important. Because of this, there is a wide range of CFD codes, models and methods to solve for different flow simulations.

2.2.1 OpenFOAM

The program used to solve the different flow problems in this report is OpenFOAM. OpenFOAM is a free open source CFD package and can be downloaded from their website [15]. It is developed by OpenCFD Ltd and distributed by the OpenFOAM foundation. It is widely applicable because of the many different solvers implemented in the software. The software package can be used in both commercial and academic organizations to solve all types of problems (from complex flows with chemical reactions to turbulence and from electromagnetics to financial problems). The wide applicability of this package is caused by the ability of the software to solve partial differential equations very efficiently. The package is constantly developed not only by the people behind the code but also by users from all over the world.
The package includes software from the preprocessing up to the post processing processes. The package contains over 80 solver applications for different types of problems and over 170 utility applications that perform the pre- and post-processing tasks. [15].
One of the major advantages of OpenFOAM is the ability to run processes in parallel which greatly improves the speed of the calculations and processing of large amounts of data. Because there are so many different solvers and techniques implemented, it is very easy to switch in order to improve the quality of the results or to get a stable run for example.

2.2.2 Turbulence modeling techniques

An analytical solution for the Navier Stokes equation is not yet found. It is a difficult process to solve the Navier Stokes equations exact for all kinds of geometries because the equations are non linear and there are a lot of length scales involved. The behaviour of the flow depends on all of those different length scales as well as time scales. In order to get a good model for a turbulent flows, there are a different techniques which can be used to turn the Navier Stokes equations into equations which can be handled by a computer. In the next part a short description is given for some techniques which can be used to solve the Navier Stokes equations in a turbulent flow case.

Direct Numerical Simulation

Direct Numerical Simulation (DNS) solves the Naviers-Stokes equations for every scale that is present in the fluid motion. This results in a very accurate solution of the flow since only some numerical errors are introduced. In order to get a very accurate result, the grid used for DNS has to be fine enough to resolve the turbulent motion from the large motion scales on to the order of the Kolmogorov scale [3]. This means that in case of difficult geometries, the grid is very difficult to construct and extremely big. Because DNS is a very demanding process, it could only be applied from the 1970's because sufficient amount of computer power was available from that time on. It has been estimated that the computational cost scale with Re^3 [2] which makes this method not applicable in high Reynolds number flows because the time it takes to find a solution will be too long.
This method can be used in order to understand turbulence and to extract statistical data which can not easily be found in experiments. For example, it helped to quantify the influence of the tools used in measurements and helped to find better measurements methods [3].
DNS is an extremely interesting tool to analyze and understand turbulence better. In a design process however, this method is too demanding in order to be used effectively.

Reynolds-Averaged Navier-Stokes equations

Method A different technique to solve the Navier Stokes equation is the Reynolds-Averaged Navier-Stokes method or RANS method. This method is developed in order to find a suitable solution in a flow problem within a very reasonable time. This means that it will not solve the Navier Stokes equations exactly but will apply a model in order to come to a solution. A RANS solver uses Reynolds decomposition and the time-averaged Navier-Stokes equations. The main assumption in Reynolds decomposition is that the variables, which are solved, can be split into two parts, the first being the mean quantity and the second being a variation. This principle is shown for the velocity in figure 2.1 and in general in 2.7. It should be noted that $\bar{.}$ represents the mean value.

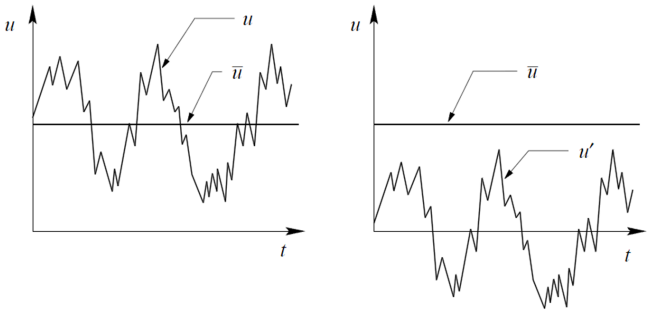


Figure 2.1: *Reynolds Decomposition [4]*

$$\vec{\phi} = \bar{\vec{\phi}} + \vec{\phi}' \tag{2.7}$$

With the use of the Reynolds decomposition of 2.7 and the Navier-Stokes equations in 2.6, a set of Reynolds decomposed Navier-Stokes equations can be defined as in 2.8.

$$\begin{aligned} \frac{\partial}{\partial x_i}(\overline{u}_i + u_i') &= \frac{\partial}{\partial x_i}\overline{u}_i + \frac{\partial}{\partial x_i}u_i' = 0 \\ \frac{\partial}{\partial t}(\overline{u}_i + u_i') + \frac{\partial}{\partial x_j}(\overline{u}_i + u_i')(\overline{u}_j + u_j') &= \frac{\partial}{\partial t}(\overline{u}_i + u_i') + \frac{\partial}{\partial x_j}(\overline{u}_i\overline{u}_j + u_i'\overline{u}_j + \overline{u}_i u_j' + u_i'u_j') \\ &= (\overline{f}_i + f_i') - \frac{1}{\rho}\frac{\partial}{\partial x_i}(\overline{p}_i + p_i') + \nu\frac{\partial^2}{\partial x_j \partial x_j}(\overline{u}_i + u_i') \end{aligned} \tag{2.8}$$

The time average of a variable will result in the mean value of that same variable. It can be shown that the time average of the fluctuations is equal to zero as is done in 2.9. The time average of the mass conservation equation (2.8) is taken. In 2.9, $\langle . \rangle$ is the time average.

$$\begin{gathered} \left\langle \frac{\partial}{\partial x_i}\overline{u}_i \right\rangle + \left\langle \frac{\partial}{\partial x_i}u_i' \right\rangle = \frac{\partial}{\partial x_i}\overline{u}_i = 0 \\ \text{and from 2.8} \\ \frac{\partial}{\partial x_i}u_i' = 0 \end{gathered} \tag{2.9}$$

The time averaging can also be applied to the momentum equation which will result in 2.10.

$$\left\langle \frac{\partial}{\partial t}(\overline{u}_i + u_i') \right\rangle + \left\langle \frac{\partial}{\partial x_j}(\overline{u}_i\overline{u}_j + u_i'\overline{u}_j + \overline{u}_i u_j' + u_i'u_j') \right\rangle = \left\langle (\overline{f}_i + f_i') \right\rangle - \left\langle \frac{1}{\rho}\frac{\partial}{\partial x_i}(\overline{p}_i + p_i') \right\rangle + \left\langle \nu \frac{\partial^2}{\partial x_j \partial x_j}(\overline{u}_i + u_i') \right\rangle$$

$$\frac{\partial}{\partial t}(\overline{u}_i) + \frac{\partial}{\partial x_j}(\overline{u}_i\overline{u}_j) = \overline{f}_i - \frac{1}{\rho}\frac{\partial \overline{p}_i}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} - \frac{\partial}{\partial x_j}\left\langle u_i' u_j' \right\rangle \tag{2.10}$$

The equations in 2.9 and 2.10 form the Unsteady Reynold-Averaged Navier-Stokes equations or uRANS equations. There are two basic arguments for keeping the time derivative in these equations. The first one has to do with the ergodicity of the equaions which states that the time averaging is equal to the ensemble averaging. The second has to do with the fact that there are multiple time scales where it is stated that there are two time scales, a large and a small for where the average is taken over the small time scale and the large time scale fluctuations are kept. There are some arguments for and opposed to these hypothesis and more information can be found in [4].

The last term in the second equation in 2.10 represent the so-called Reynold stresses (to make it a true stress, it should be multiplied with ρ, but this was already divided out since it was assumed to be constant). The Reynold Stress represents the transport of momentum through the turbulent nature of the flow.
The equation of momentum as given in 2.10 represents three independent equations. Together with either the conservation of mass (equation in 2.9) or the Poisson equation, the total of independent equations adds up to 4. However when the number of unknowns is counted, it can be seen that with the three components of velocity, pressure and the Reynold Stresses, it exceeds the number of independent equations (assuming the other forces are given). This problem is called the closure problem [2]. It is because of this closure problem that the uRANS-equations require some sort of model for the Reynold Stress. Many different models were developed in order to solve this part of the equations, however it is very difficult to find a suitable model. A few of these models will be explained later on.

Models There are many different models used to describe the Reynolds stress. The most simple model would be an algebraic solver which calculates the magnitude of the Reynolds stress directly from other flow variables. These models however do not take the history of for example diffusion or convection of turbulent energy into account and are not accurate in larger computations. These models are called zero-equation turbulence models and are sometimes used in the start of the computations to speed up the process. Because they are not used here, they will not be treated any further. In general only the one equation and two equations models are used. As the name already suggests, they differ in the amount of equations used to determine the Reynolds stress. A few common uRANS models are described below.

Turbulent kinetic energy model One of the easiest applicable models is the mixing length model. This model is applicable in two-dimensional boundary-layer flows and is based on the fact that the turbulent viscosity (ν_t) is based on the velocity gradient in perpendicular direction of the flow and the turbulent mixing length. This model however, is not valid in for example circular jet flows. From experiments it was determined that the turbulent viscosity was not zero in the center of the jet while from the mixing length theory it seemed like this was the case. This called for a new model and both Kolmogorov as well as Prandtl independently proposed a method based on the turbulent kinetic energy. This method is an one equation model solving the turbulent kinetic energy equation (k) and describes the kinematic eddy viscosity as $\nu_t = ck^{\frac{1}{2}}l_m$ with c as a constant and l_m as the mixing length. The transportation of the turbulent kinetic energy is modeled through equation 2.11. With the definitions in 2.12 and the remaining equations, the only thing needed to be specified is the mixing length l_m in order to give a complete description of the problem. This definition of the mixing length is the main disadvantage of this method, because it is not easy to define one uniform formula for the mixing length [5].

$$\frac{\partial k}{\partial t} + \bar{u}_j \frac{\partial k}{\partial x_j} = \tau_{i,j}\frac{\partial \bar{u}_i}{\partial x_i} - C_D \frac{k^{\frac{3}{2}}}{l_m} + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \tag{2.11}$$

$$
\begin{aligned}
\nu_t &= ck^{\frac{1}{2}}l_m \\
\mathcal{P} &= -\overline{u_i'u_j'}\frac{\partial \bar{u}_i}{\partial x_j} \\
\epsilon &= C_D\frac{k^{\frac{3}{2}}}{l_m} \\
\tau_{i,j} &= 2\nu_t S_{i,j} - \frac{2}{3}k\delta_{i,j} \\
C_D = 0.08 &\text{ and } \sigma_k = 1
\end{aligned}
\tag{2.12}
$$

The viscous strain tensor is defined as in 2.13.

$$
S_{i,j} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.13}
$$

Spalart-Allmaras model The Spalart-Allmaras model is an one equation turbulence model specially designed to simulate aerodynamic flows. The model directly solves the transport equation for the turbulent viscosity and is designed to be better than other one equation models but more simple than the two equation models. The idea is that the accuracy for aerodynamic flow problems is increased without increasing the calculation time too much. The equation solved by the Spalart-Allamaras model is given in 2.14. The other expressions for the constants and the original values are given in 2.15.

$$
\begin{aligned}
&\frac{\partial \tilde{\nu}}{\partial t} + \bar{u}_j\frac{\partial \tilde{\nu}}{\partial x_j} = C_{b1}(1-f_{t_2})\tilde{S}\tilde{\nu} + \frac{1}{\sigma}\left(\nabla\cdot[(\nu+\tilde{\nu})\nabla\nu] + C_{b2}\left|\nabla\nu\right|^2\right) - \left[C_{w1}f_w + \frac{C_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + f_{t1}\Delta\bar{u}^2 \\
&\qquad\qquad \text{with:} \\
&\nu_t = \tilde{\nu}f_{v1}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu} \\
&\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2}f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}
\end{aligned}
\tag{2.14}
$$

$$
\begin{aligned}
S = \sqrt{2\Omega_{ij}\Omega_{ij}}, \quad \Omega_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) \\
f_w = g\left[\frac{1+C_{w3}^6}{g^6+C_{w3}^6}\right]^{1/6}, \quad g &= r + C_{w2}(r^6 - r), \quad r = \frac{\tilde{n}u}{\tilde{S}\kappa^2 d^2} \\
f_{t1} = C_{t1}g_t exp\left(-C_{t2}\frac{\omega_t^2}{\Delta\bar{u}^2}[d^2 + g_t^2 d_t^2]\right), \quad f_{t2} &= C_{t3}exp\left(-C_{t4}\chi^2\right) \\
\sigma = \frac{2}{3}, \quad C_{b1} = 0.1355, \quad C_{b2} &= 0.622, \quad \kappa = 0.41, \quad C_{w1} = \frac{C_{b1}}{\kappa^2} + \frac{1+C_{b2}}{\sigma} \\
C_{w2} = 0.3, \quad C_{w3} = 2, \quad C_{v1} &= 7.1, \quad C_{t1} = 1, \quad C_{t2} = 2, \quad C_{t3} = 1.1, \quad C_{t4} = 2
\end{aligned}
\tag{2.15}
$$

***k* − *ϵ* model** The $k-\epsilon$ model is one of the most commonly applied turbulence models. This model is a two equation model and solves the equations for the turbulent kinetic energy (k) and for the dissipation (ϵ). The main advantage of this method is that the length scale can be determined from the equations and it is no longer needed to specify the turbulent mixing length.
The turbulent kinetic energy equation is the same as the one described above in 2.11 and the equation for the dissipation is given in 2.16. It is possible to obtain the analytical solution for the dissipation but since it needs to go all the way up to the dissipative ranges, this is not used in the modeling process because it is not a suitable starting point for the modeling process [2], instead the given equation in 2.16 is used which is more an empirical formula. The turbulent viscosity is calculated from $\nu_t = C_\mu k^2/\epsilon$ where C_μ is a constant equal to 0.09.

$$
\frac{\partial \epsilon}{\partial t} + \bar{u}_j\frac{\partial \epsilon}{\partial x_j} = \nabla\cdot\left(\frac{\nu_t}{\sigma_\epsilon}\nabla\epsilon\right) + C_{\epsilon 1}\frac{P\epsilon}{k} - C_{\epsilon 2}\frac{\epsilon^2}{k} \tag{2.16}
$$

The $k-\epsilon$ model is one of the simplest complete turbulence models and is applied in a lot of CFD codes. Because it is a relatively simple model, the errors made with complex structures can be very significant. Over the years many people have introduced variations to make this model more applicable in different situations and for more complex structures.

Large Eddy Simulations

Method The previous methods for solving a turbulent flow problem are either based on no modeling (DNS) or the use of the average and a model for the Reynolds Stress (uRANS). Large Eddy Simulation (LES) is a method of solving which combines a little of both. In a LES solver, the flow is also decomposed as is in RANS simulations, however the way of applying this decomposition is a little different.
The main idea of a LES solver is to exactly solve the range of scales which contain a lot of energy (the energy containing range contains approximately 80% of the energy) and model the scales which can not be resolved on the mesh (the so-called Subgrid Scales or SGS). By applying a SGS-model, the grid can be more coarse than that of a DNS calculation but the modeling is not as extensive as that of uRANS computations.

The first step taken in order to describe a LES solver is to use a filtering process. The filtering process splits the flow domain in the resolved length scales and the modeled length scales. The general expression for the filter is given in 2.17. The filter has a similar function of the averaging as is done in the uRANS process. It splits the equations in an exactly solved part and a modeled part.

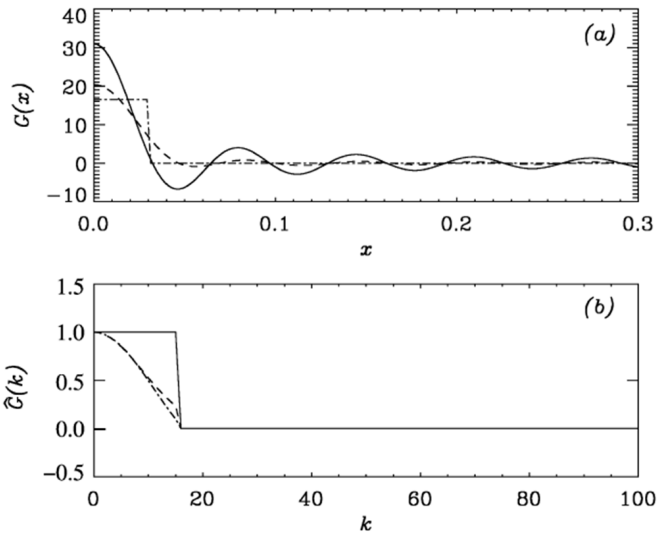


Figure 2.2: *Different filter types in the real space (a) and the Fourier space (b). —— sharp Fourier cutoff, - - - truncated Gaussian, - · - top-hat [6]*

$$\widetilde{f}(x) = \oint G(x, x'; \Delta) u(x') dx' \tag{2.17}$$

Equation 2.17 shows the definition of the filter where G is the filter applied to the function f. The most common used filters are the Gaussian filter (2.18a), the top-hat filter (2.18b) and the sharp Fourier cutoff filter (2.18c) [6]. In these equations, Δ is the filter width and will be equal to the spacing of the grid.

$$G(x, \Delta) = \sqrt{\frac{6}{\pi \Delta^2}} exp\left(-\frac{6x^2}{\Delta^2}\right) \tag{2.18a}$$

$$G(x, \Delta) = \begin{cases} \frac{1}{\Delta} & \text{if} |x'| = \frac{\Delta}{2} \\ 0 & \text{else} \end{cases} \tag{2.18b}$$

$$G(x, \Delta) = \begin{cases} \frac{1}{\Delta} & \text{if} k \leq \frac{\pi}{\Delta} \\ 0 & \text{else} \end{cases} \tag{2.18c}$$

These functions all fulfill the condition stated in 2.19.

$$\oint G(x, x'; \Delta) dx' = 1 \tag{2.19}$$

The functions are split in the filtered part and the residual part as given in 2.20 which is the requirement for the filter function.

$$f = \widetilde{f} + f' \tag{2.20}$$

The filtering process is then applied to the Navier-Stokes equations (2.6). Because of the choice of filter, $\widetilde{\frac{\partial u_i}{\partial x_i}}$ equals $\frac{\partial \widetilde{u}_i}{\partial x_i}$.

$$\begin{aligned} \frac{\partial \widetilde{u}_i}{\partial x_i} &= 0 \\ \frac{\partial}{\partial t}(\widetilde{u}_i) + \frac{\partial \widetilde{u_i u_j}}{\partial x_j} &= \widetilde{f}_i - \frac{1}{\rho}\frac{\partial \widetilde{p}_i}{\partial x_i} + \nu \frac{\partial^2 \widetilde{u}_i}{\partial x_j \partial x_j} \end{aligned} \tag{2.21}$$

The final result is almost the same as the LES equations and is displayed in 2.21. The equations show great similarity with the equations in 2.10 for uRANS solvers. Despite the great similarity, these equations are quite different from the equations of the uRANS approach. The difference is the meaning of the second (non linear) term of the momentum equation. This term consists of the filtered velocities, the subgrid velocities. The momentum equation displayed in 2.21 can not be used for calculation, because $\widetilde{\tilde{u}_i \tilde{u}_j}$ can not simply be replaced by $\tilde{u}_i \tilde{u}_j$. In order to be able to model these equations, the term $\tilde{u}_i \tilde{u}_j$ is subtracted and added to second term in 2.21 as shown in 2.22.

$$\begin{aligned} \overbrace{(\tilde{u}_i\tilde{u}_j + u_i u_j - \tilde{u}_i\tilde{u}_j)} &= \overbrace{\left(\tilde{u}_i\tilde{u}_j + (\widetilde{u}_i + u_i')\left(\widetilde{u_j} + u_j'\right) - \tilde{u}_i\tilde{u}_j\right)} \\ &= \widetilde{(\tilde{u}_i\tilde{u}_j)} + \left(\overbrace{(\widetilde{u}_i\widetilde{u_j} - \tilde{u}_i\tilde{u}_j)} \right) + \left(\widetilde{(u_i'\widetilde{u_j})} + \widetilde{(\widetilde{u}_i u_j')}\right) + \left(\widetilde{(u_i'u_j')}\right) \\ &= \widetilde{(\tilde{u}_i\tilde{u}_j)} + \tau_{SGS} \end{aligned} \tag{2.22}$$

The term τ_{SGS} represents the subgrid scale stresses and needs to be modeled. These stresses consist of the Leonard Stress (second term in the second line of 2.22), the cross stress (third term in the second line of 2.22) and the Reynold Stress (last term in the second line of 2.22). It has been tried to find models for each of the components of the stress, but without success. Because the Leonard Stress and the cross stress are not Galilean invariant (independent of the inertial frame), it has been not possible to model these individually. The combination of the SGS stresses is Galilean invariant and can therefore be modeled.

The computational cost for a LES simulation is less than the cost of a DNS simulation but it is higher than that of the RANS simulations. In order to get a good result, the simulated situation has to be simulated in three dimensions. This is more computational expensive than the simulation in RANS simulations because these can be done in just two dimensions.

Because the LES method models the small scale eddies, another drawback of LES is created in near-wall regions. The flow in near-wall regions is dominated by the small scale eddies and therefore not solved correctly by LES if the resolution is not lowered. In order to overcome this problem a lot of different wall models have been created which take over the calculation in the near wall region. Without a wall model the cell size at the wall should be around a typical length scale of $y^+ \approx 1$ (where $y^+ = \frac{u_* y}{\nu}$ and u_* is the friction velocity, y is the distance to the wall and ν the kinematic viscosity) and will result in a very fine mesh. This will result in a problem similar to a DNS simulation. To avoid a very fine mesh at the wall, wall models can be applied. Wall models describe the velocity profile and dissipation in the cell closest to the wall. This means that the different properties of the flow will be modeled through a preset model. The cell size can be taken much larger because the wall model will predict the velocity profile (for example) all the way up to the wall. However in some situations (with detaching flow for example) it is not possible to use a wall model. If the cell size is too small close to the wall, the wall model will predict a wrong result for the velocity. In reality it could be that the velocity is already much more slowed than the velocity predicted by the simulation. On the contrary is it also possible that, in case the wall size cells are to large, the velocity is predicted to slow. Furthermore it should be taken into consideration that a wall model does not predict a back flowing flow. A situation where it is not allowed to use wall models is shown in picture 2.3.

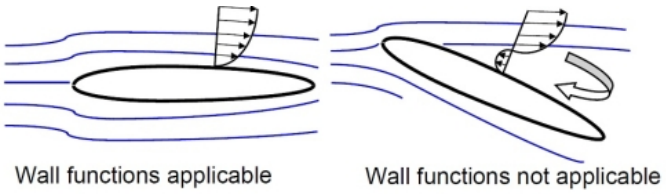


Figure 2.3: *When to use wall models [14]*

Models

Smagorinsky model The oldest method for LES is the Smagorinsky model and because of its simplicity, it is still widely used. The Smagorinsky model is based on the Boussinesq hypothesis and determines the subgrid viscosity from the relation in equation 2.23. The subgrid scale viscosity is based on the filter width and a constant which can be determined from the turbulence problem. Since the model depends on the filter width and the filter width is directly dependent on the grid size, the results will improve a lot with a very fine mesh. This model is also completely dissipative. This means that in the model, the energy from eddies with length scale k can only be transported to the smaller scales $k+1$. In reality however, some of the energy is also transported back up to the level $k-1$. This will not be a lot of energy but it happens. This model does not use this assumption so it can result in a system which is too dissipative.

$$\begin{aligned} \tau_{SGS,ij} &= -2\nu_{SGS}\tilde{S}_{ij}, \quad i,j=1,2,3, \quad i \neq j \\ \tilde{S}_{ij} &= \sqrt{2\Omega_{ij}\Omega_{ij}}, \quad \Omega_{ij} = \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} - \frac{\partial \tilde{u}_j}{\partial x_i}\right) \\ \nu_{SGS} &= (C_S\Delta)^2 \left|\tilde{S_{ij}}\right| \end{aligned} \tag{2.23}$$

The basic Smagorinsky model requires a fixed coefficient in order to determine the subgrid scale viscosity. In order to improve the results for the model, this coefficient can also be made dependent from the position and time and is the so-called dynamic Smagorinsky model. Germano et al [7] proposed the modification of the Smagorinsky model by applying a second filter. The filter width of the second filter is higher than that of the original filter (so it skips more of the information than the grid can supply). When the assumption is made that there is one coefficient that models all the stresses (scale similarity assumption) for both situations, two expressions for the stress can be found (one for each filter). In order to extract the coefficient, an error norm can be formed and by minimizing this error and applying an appropriate filter, the formulation for the filter can be completed as in 2.24. In this formulation, the α and β are the resolved fields of the original en second filter respectively , the $\tilde{.}$ is the data found by applying the second filter and $\langle . \rangle$ is the average taken [6].

$$\begin{aligned} C_v &= -\frac{1}{2}\frac{\langle \mathbf{L} : \mathbf{M} \rangle}{\langle \mathbf{M} : \mathbf{M} \rangle} \\ \mathbf{L} &= \widetilde{\tilde{u}\tilde{u}} - \tilde{\tilde{u}}\tilde{\tilde{u}} \\ \mathbf{M} &= \beta - \tilde{\alpha} \end{aligned} \tag{2.24}$$

Detached Eddy Simulations

The last method for solving the Navier-Stokes equations discussed in this report is the Detached Eddy Simulation (DES) and some additions to this model. DES is a combination of RANS modeling and LES modeling and is therefore sometimes called a 'hybrid' solver. The main focus of the DES solver is to combine the speed of a RANS model with the accuracy of the LES model. The RANS model calculates the regions close to a boundary so the grid does not have to be refined as much as would be required by a LES simulation. A distance parameter (2.25) is formulated which indicates when the switch is made between the RANS model and the LES model. This method is well applicable to massive separating flows because the difference between the RANS and LES areas is clear. The separation needs to be large in order for the model to get a good switch between the RANS and LES regions.

$$\widetilde{d} \equiv min(d, C_{DES}\Delta), \quad \text{with } \Delta \equiv max(\Delta x, \Delta y, \Delta z) \tag{2.25}$$

The distance parameter ($\widetilde{d}$) depends on the distance d to the closest wall, the grid spacing Δ and a coefficient C_{DES}. In the region close to the wall ($d \ll \Delta$) the model uses a RANS model while in the region where the maximum cell width is smaller than the distance to the wall ($d < C_{DES}\Delta$) the LES model is used. The influence of the coefficient C_{DES} was studied by Shur *et al* [11] where it was found that the results for $C_{DES} = 0.65$ gave good results for the test cases. The influence of this coefficient was also studied and it was found that the solutions were not very sensitive to the value of C_{DES}. The value of the coefficient can be a little higher when higher-order differencing schemes are applied while lower values can be found when lower-order upwind schemes are used [12].
As with all the models, there is a constant update on modeling techniques in order to get the most optimal result in the least amount of time. One of the problems with the DES simulations occurred when thick boundary layer flows were simulated. LES approximation is not good in the boundary layer so when a thick boundary layer is present, the region for RANS should be larger.

Models DES uses only one type of turbulence model throughout the entire domain.

S-A DES The first DES model was designed with the Spalart-Allmaras model. This model applies the Spalart-Allmaras model for RANS computations and the Smagorinsky based model for the LES computations. The switch is made with the help of the distance parameter described in 2.25. Because the switch is suddenly made, the gradient over the interface is discontinues. It was shown that in practice this does not effect the solution, but should be kept in mind. The idea is that the boundary layer is resolved through the RANS computations while the rest of the flow is resolved with the LES method. When the grid is too small, the switch can be triggered within the boundary layer which will result in too dissipative results. It is therefor better to over predict the boundary layer size than to under predict it.

S-A IDDES The Improved Delayed Detached Eddy Simulations method is a method which intends to improve the normal DES formulation. The idea is to make the switch between LES and RANS no longer dependent on the overall grid spacing but on the turbulence present and the local grid spacing. This method intends to prevent the switch of RANS and LES in the boundary layer.

2.2.3 Numerical solutions

After the choice of the way in which the problem is going to be solved, the correct type of discretization has to be chosen. The type of discretization influences for example the accuracy and the stability of the solution. It is important to chose the right type of discretization for the simulation because the results found can be very wrong if a solution is found at all.

Spatial discretization

In order to do a CFD analysis on a geometry, a discretization of the domain is needed. This discretization in the spacial domain is called meshing. Creating a mesh is often not very hard, while creating a good mesh is a profession on its own. There are a lot of programs able to make a mesh, but in order to use these meshes and extract good results from the CFD calculations, experience is a helpful factor. There is not one mesh applicable in all situations. The mesh depends on the geometry, the flow properties and the method of solving. In LES calculations for example, the mesh defines the size of the eddies solved and the size of the eddies modeled. As said before, it is important to have a good mesh and in order to create one there are a couple of things which are important to keep in mind.

- **Cell type**: The type of cell determines the structure of the mesh and the way the solution is created. Some cells (for example triangles) are easier to use but the accuracy of the solution is less (than for example quadrilateral elements).
- **Cell size**: The size of the cell decides the resolution of the solution and also the magnitude of the numerical errors introduced. Numerical errors are introduced because a continuous situation is approximated by a discrete solution. Compare it with the approximation of a parabola, the more points used to model the parabola, the more it will look like a parabola and the smaller the error will be.
- **Cell distribution**: The way in which the cells are distributed determines the accuracy of the solution on the points of interest. It is important to have a lot of cells on the points of interest and there where the solution on the points of interest are greatly influenced. There can be less cells in place where the solution is not much of an interest or of influence on the final solution in order to save calculation time.
- **Aspect Ratio**: The aspect ratio of a cell is defined as the length of the longest cell side divided by the length of the shortest cell side. If this value is very large, the cells are very flat in one direction. This means that in one direction there will be a very coarse resolution and in another very fine. The best cell shapes are squares or cubes, with an aspect ratio equal to one, because the interpolation errors introduced through the difference in size are kept to a minimum.
- **Skewness**: The skewness determines the difference between the largest angle and the smallest angle within a cell (figure 2.4). If a large skewness is present, large errors in the interpolation can occur which will result in bad results.
- **Orthogonality**: The orthogonality of a cell determines the angle between the direct line from two cell centers and the face normal between those cells (figure 2.5). A non-orthogonal mesh requires extra correctors during the actual calculations in order to solve the problem correctly.
- **Stretching**: Stretching is the ratio between the cell lengths of two adjacent cells. This ratio should not be to high in order to avoid large steps in resolution in the cell. These steps in resolution introduce some interpolation errors and should be avoided.

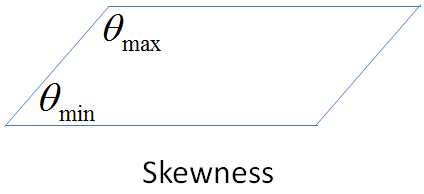


Figure 2.4: *Skewness of a cell [19]*

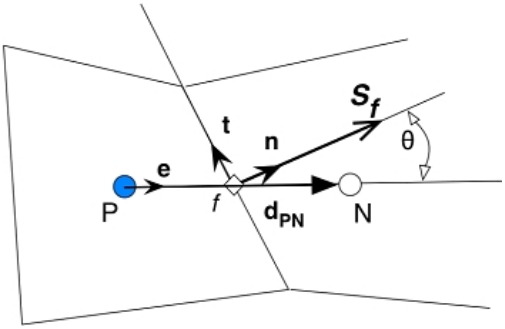


Figure 2.5: *A non orthogonal cell [18]*

Many programs can be used to do an automatic meshing of a given geometry. These meshing programs use all kinds of different algorithms in order to get a mesh. However, these programs can not create the perfect mesh so it should always be checked to see if the mesh is indeed correctly set up. OpenFOAM uses the blockMesh utility in order to create meshes. This is done by supplying some points which are used to make blocks. For each block, the number of cells in every direction can be set and the stretching within the blocks can be influenced. The user should make sure that the cell size on the boundaries of the block are of the right magnitude in order to avoid too much stretching between blocks and to have the correct cell size at the boundaries. This can be checked by the formulas in appendix B.

Equation discretization

The Navier-Stokes equations are in a continuous form written and also need to be discretized. The discretization can be done with the finite difference method, finite volume method or finite element method.

Finite difference method The first method is the finite difference method. When this method is applied, the solution is approximated by using the difference between the discrete points. It is based on the Taylor series expansion and can be applied in all types of problems. An example of how the finite difference method works is given in appendix A.

Finite volume method The second method is the finite volume method. Around the discretized points, small volumes are defined which must obey the equations to be solved. In order to fulfill the equations, the properties inside the cell can only change by the flux through the cell faces or a change within the cell (in case of heating for example a source inside the cell). For every cell, the flux through the boundary can be defined and because of the system being conservative, these fluxes can be linked together so the conservation laws are satisfied. Because this method can deal with unstructured meshes very easily, it is applied in a lot of CFD packages such as OpenFOAM.

Finite element method The finite element method is a method which solves boundary value problems by applying the minimization of an error function. This method uses simple solutions and shapes in order to create the solution for a very complex problem and/or shape. This method is often applied in structural mechanics because it can handle complex geometries a lot better than finite difference methods. For CFD types of problems however, this method is not applied often because the amount of calculations and stored variables is higher than with the finite volume method. Since CFD problems are often very complex and have a very high number of cells, the operations per cell will be kept as low as possible.

OpenFOAM uses the finite volume method in order to discretize the equations that are solved. OpenFOAM is used in the test cases later on so the discretization applied in OpenFOAM is discussed here. Because the finite volume method is used, all the parts of equations need to be specified as an integral over the volume of the cell. By using Gauss theorem, these volume integrals can be converted to surface integrals. These surface integrals represent the fluxes through each cell. In order to solve the equations, the basic terms are linearized. These basic parts make up the set of equations needed to be solved. With the help of an implicit method (iteration is needed because the next value depends also on itself: $y^{n+1} = f(y^n, y^{n+1})$) or an explicit method (in which the next values is determined from the previous values: $y^{n+1} = f(y^n)$), the values for the next step will be calculated.
In order to discretize all the different parts of the equations, OpenFOAM gives the possibility to specify the discretization of every part of the conservation equations. With the careful study of the conservation equations and the derived equations in order to solve them, different types of basic equations can be found. By defining the way in which these basic equations are discretized, all equations can be discretized by combining these basic parts. OpenFOAM uses a standard definition for every basic equation so that different variables can be filled in later on [16]. In the basic

equations, ϕ represents any transported variable such as velocity or momentum and the Γ represents a diffusivity coefficient such as the kinematic viscosity ν. $\mathbf{S}$ will be the surface of the cell and V the volume of the cell.

Laplacian term The Laplacian term (or diffusion term) is defined as the left hand side of 2.26, linearized and discretized as shown in 2.27.

$$\iiint_V \nabla \cdot (\Gamma \nabla \phi) dV = \iint_S d\mathbf{S} \cdot (\Gamma \nabla \phi) = \sum_f \Gamma_f \mathbf{S}_f \cdot (\nabla \phi)_f \tag{2.26}$$

$$\mathbf{S}_f \cdot (\nabla \phi)_f = |S_f| \frac{\phi_N - \phi_P}{|\mathbf{d}|} \tag{2.27}$$

with: $\mathbf{d} =$ distance between the cell center (P) of the current cell and center of the neighbouring cell (N)

The discretization shown in 2.27 is applied when the distance $\mathbf{d}$ is orthogonal on the face plane. This method is an implicit method. When the mesh is non-orthogonal, a correction is made which uses interpolated cell center gradients in order to adapt the flux. An example of a non-orthogonal mesh is shown in figure 2.5.

Convective term The convective term in the finite volume method is defined as the left hand side in 2.28.

$$\iiint_V \nabla \cdot (\rho \mathbf{U} \phi) dV = \iint_S d\mathbf{S} \cdot (\rho \mathbf{U} \phi) = \sum_f \mathbf{S}_f \cdot (\rho \mathbf{U})_f \phi_f = \sum_f F \phi_f \tag{2.28}$$

This term can be discretized as shown, where ϕ_f is the value for the transported variable determined from the cells. There are different methods which can be applied in order to find ϕ_f which will be explained now.

- **Central Differencing**: Central differencing is, just as in appendix A, a second order accurate method. The value searched for is the value at the boundary between two cells. In order to find this value, the function f_x is applied. This function is the length from the surface f to the center of the cell N divided by the length between the cell centers P and N as shown in 2.29.

$$\begin{gathered} \phi_f = f_x \phi_P + (1 - f_x) \phi_N \\ \text{with: } f_x = \frac{\overline{fN}}{\overline{PN}} \end{gathered} \tag{2.29}$$

  Disadvantages of using central differencing is that it can introduce unphysical oscillations. This due to the fact that the equation is not bounded. When the convective term takes the overhand in an equation, the results can become very bad because of these oscillations.

- **Upward Differencing**: Upward differencing is, also just as in appendix A, a first order accurate method. In OpenFOAM it is implemented as shown in 2.30. The advantage is that this solution is bounded but at the cost of accuracy.

$$\phi_f = \begin{cases} \phi_P & \text{for: } F \geq 0 \\ \phi_N & \text{for: } F < 0 \end{cases} \tag{2.30}$$

- **Blended Differencing**: Blended differencing is a combination of the upwind and the central differencing methods. The idea is to get an bounded solution but also to keep some more accuracy.

Time derivative term The time is discretized with the use of the time step. The first time derivative is integrated over the volume and approximated by the values of the previous time step or time steps. There are several methods which can be used in OpenFOAM (note: i in the following items represents the i^{th} element in time at a certain position, $i + 1$ is then at the same position but for the next time step).

- **Euler Implicit**: The Euler implicit scheme is a simple way of defining the time derivative. This method only depends on the previous time step and therefore will only be first order accurate in time. The definition is shown in 2.31.

$$\frac{\partial}{\partial t}\iiint_V \rho\phi dV = \frac{(\rho_P\phi_P V)_i - (\rho_P\phi_P V)_{i-1}}{\Delta t} \tag{2.31}$$

- **Backward Differencing**: The backward differencing implemented in OpenFOAM is a second order accurate time scheme which depends on the previous two time steps and is shown in 2.32.

$$\frac{\partial}{\partial t}\iiint_V \rho\phi dV = \frac{3(\rho_P\phi_P V)_i - 4(\rho_P\phi_P V)_{i-1} + (\rho_P\phi_P V)_{i-2}}{2\Delta t} \tag{2.32}$$

- **Crank-Nicolson**: This is based on the Crank Nicolson time discretization scheme.This method is a second order accurate time discretization and is based on the formula in 2.33. In OpenFOAM it is also possible to blend this scheme with the Euler scheme in order to get a stable result if Crank-Nicolson alone is not stable.

$$\frac{\partial\phi}{\partial t} = \frac{\phi_{i+1} - \phi_i}{\Delta t} = \frac{1}{2}\left[F^{i+1}(\phi, x, t, \frac{\partial\phi}{\partial t}, \frac{\partial^2\phi}{\partial t^2}) + F^{i}(\phi, x, t, \frac{\partial\phi}{\partial t}, \frac{\partial^2\phi}{\partial t^2})\right] \tag{2.33}$$

- **Steady State**: By defining steady state in OpenFOAM, the solver will skip the time derivative. This can only be done in a steady state case obviously and is used in order to save time.

Solution algorithm

After the equations are discretized, the most important part of the simulation needs to be done namely the solving. The type of solver also influences the accuracy of the solution and the stability of the run. There are many different solvers implemented in OpenFOAM. A few of these solvers will be discussed here.

- **PCG**: The PCG solver is the preconditioned conjugate gradient solver for symmetric matrices in OpenFOAM. The conjugate gradient method uses an iterative approach in order to find a solution. This method consists of calculating the residuals and changing the solution until the residual is small enough. In order to speed up the process, preconditioning is applied. The preconditioned conjugate gradient solving method is described in 2.34. The system to be solved here is $Ax - b = 0$ with preconditioning matrix C [17].

$$\begin{aligned}
r_o &= b - Ax_o \\
z_o &= C^{-1}r_o \\
d_o &= z_o \\
\text{for } k &= 0, 1, 2, ... \\
\alpha_k &= \frac{z_k^T r_k}{d_k^T A d_k} \\
x_{k+1} &= x_k + \alpha_k d_k \\
r_{k+1} &= r_k - \alpha_k A d_k \\
z_{k+1} &= C^{-1}r_{k+1} \\
\beta k + 1 &= \frac{z_{k+1}^T r_{k+1}}{d_k^T A d_k} \\
d_{k+1} &= z_{k+1} + \beta_{k+1} d_k \\
&\text{stop if } r \text{ is below a prescribed value}
\end{aligned} \tag{2.34}$$

There are several preconditioning methods which can be applied in OpenFOAM. These include diagonal incomplete-Cholesky (DIC), diagonal incomplete LU (DILU) and GAMG.

In case the matrix is not symmetric, the PBiCG method can be used which stands for preconditioned bi-conjugate gradient solver.

- **GAMG**: GAMG is the generalised geometric-algebraic multi-grid solver build in OpenFOAM. The idea of a multi-grid solver is to solve the problem on a coarse grid and use this solution to find the solution on the fine grid. High frequency errors can be reduced very effectively. By solving the problem on a coarse grid, the low frequency errors of the fine grid will appear as high frequency errors on the coarse grid and are therefore easily reduced. With the solution then brought back to the fine grid, the final high frequency errors on the fine grid can be reduced. The coarse grid is made from the original grid in order to save memory, because it is not necessary to create an entirely new grid. With OpenFOAM, it is also possible to choose the smoother as well as the number of pre- and post sweeps. In figure 2.6, the procedure for a multi grid solver is shown. This multi grid algorithm shows a multi grid for 5 levels and in combination with Gauss Seidel smoother, has ν_1 pre-sweeps and ν_2 post sweeps. On the coarsest grid, the number of sweeps done is ν_0. This depends on the set tolerance for the problem.

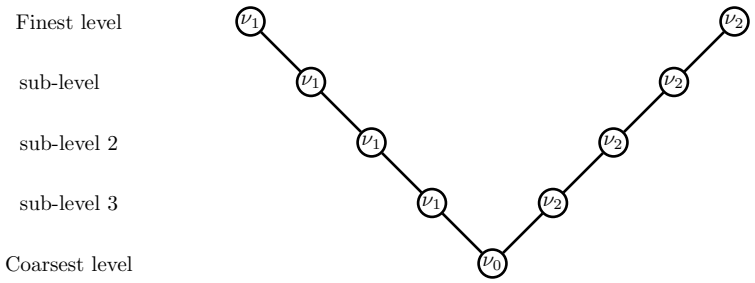


Figure 2.6: *Multi-grid solver with 5 levels*

Numerical algorithm

When the discretization of the entire problem is specified, the problem needs to be solved. In order to solve a problem in OpenFOAM many different solver algorithms can be used. There are many solver algorithms implemented in OpenFOAM for all kinds of different problems (ranging from CFD solvers to financial solvers). Two algorithms implemented in OpenFOAM will be discussed now.

SIMPLE algorithm The simpleFoam solver is a steady state solver for incompressible, turbulent flow based on the SIMPLE algorithm. SIMPLE stand for Semi-Implicit Method for Pressure Linked Equations. The idea behind this algorithm is to calculate the steady state for the stated problem from the different differential equations [10].

Step 1: The pressure is assumed to be equal to p.

Step 2: The momentum equation is solved with pressure p for the different components of u.

Step 3: The pressure equation is solved with the help of u from the previous step in order to find a new value for p.

Step 4: In case of more variables which influence p or u, these variables are solved using the new u and p. If the other variables do not influence the pressure or the velocity, it is better to solve these when convergence is reached to save time.

Step 5: Repeat this process until the specified convergence is reached.

The solution of the SIMPLE algorithm could be used if a steady problem is studied, but it is also a quick way to obtain a initial field for an unsteady problem.

PISO algorithm The PISO algorithm is used to solve transient, incompressible flow problems. PISO is short for Pressure Implicit with Splitting Operator solver. The idea behind the PISO algorithm is to solve the set of equations at every time step. It is possible to apply a number of correctors in order to achieve better convergence.

Step 1: The pressure is assumed to be equal to p.

Step 2: The momentum equation is solved with pressure p for the different components of u.

Step 3: The pressure equation is solved with the help of u from the previous step in order to find a new value for p.

Step 4: Update the velocity based on the new pressure.

Step 5: Determine the rest of the variables which influence p or u, these variables are solved using the new u and p

Step 6: Repeat this process from step 3 if this is specified (through the corrector steps).

Step 7: Advance 1 step in time and repeat the process until the final time is reached.

Boundary conditions

The boundary conditions are very important for the computations. With the wrong boundary condition, the entire simulation can be wrong. When boundary conditions are specified, it is important to keep in mind what the grid looks like, because the boundary condition is not only dependent on the physical problem but also on the grid size. One of the examples is the use of wall models in the simulation. If the grid size near the wall is too large to create a good approximation of the boundary layer, the use of wall models can be justified in order to predict the behaviour of the boundary layer. However if the grid size is small enough, the use of wall models will decrease the accuracy of the solution. In those cases it can be much more convenient to use the fixed value or zero gradient boundary conditions.

Results

When the results of a simulation are found and extracted, it is very important to check them with experimental data and other references. With the setup of the case an estimation of the results has to be made in order to determine the correct setup for the case. When the results are found it is important to check if these results are also similar to that what was expected. It is important to check the differences and try to explain them.

2.2.4 Running a case

Many steps are to be taken to get from a problem to a solution. Because flow type problems are often very demanding from a computational point of view, the set up of a case is very important. As mentioned before, the set up of a case can be crucial for finding good results in a reasonable time. Before a case is run, the following steps need to be taken in order to find a valid result.

Step 1: **Define the problem clearly.** Describe the flow situation, boundary conditions and domain. Study the theory and similar types of problems. Also check if someone already did a similar calculation and try to find their setup.

Step 2: **Create the spatial mesh.** Make sure the grid is refined enough on the points of interest.

Step 3: **Choose the method of solving and the turbulence modeling models.** Keep in mind what the main idea is behind the simulation, is it to get an accurate description of reality or to get a global view of the situation.

Step 4: **Define the numerical boundary and initial conditions.** Define all the boundary conditions and the initial conditions for the created model. Make sure that the mesh created is accurate for your boundary conditions (for example for the use of wall models).

Step 5: **Define the numerical schemes.** Make sure that the accuracy of each type of discretization is high enough for its purpose. For example, do not use first order methods in case of a LES simulation.

Step 6: **Choose the accuracy of the numerical solvers.** Choose the tolerance for the case (is it better to use only the absolute tolerance or will a relative tolerance do the job as well).

Step 7: **Set the rest of the parameters.** Choose the time step, the time domain etc.

Step 8: **Set up some control points.** If additional information (such as lift coefficients for example) can be calculated during the simulation, make sure that the set up for these parameters is in order as well.

Step 9: **Run tests.** Run tests to see if the case set up is done correctly. Make sure that the case will not diverge directly and that it will not take to long. Try to optimize the settings by comparing different setups, but also try if it is possible to run the case on multiple processors and what the ideal amount will be.

Step 10: **Run the case.** During the simulation, keep an eye on the parameters like the residuals or coefficients in order to make sure that the case converges.

Step 11: **Check and extract the results.** Check the results such as residuals, the coefficients and the flow patterns to see if something strange happens.

Step 12: **Interpret the results.** Check the results carefully to make sure the results are valid. Compare the results to the expectations, experimental data and data from others who simulated a similar case.

Step 13: **Explain the results.** Try to explain possible differences and show why the results are valid or not.

3 Simulated Cases

3.1 Test Cases

In this report two different test cases are modeled in OpenFOAM. The first one is the flow around a circular cylinder, the second is the flow over the backward facing step. These two problems are two problems which are commonly studied and that is why there is also a lot of experimental data available. The first case, flow around the circular cylinder, is mainly used to get used to OpenFOAM and setting up of cases. The second case is the main investigation of this report and will be used to compare with other solvers as well as with experimental results. The influence of different LES and DES models is to be investigated in this report. Both cases are main test cases in fluid flow simulations and are therefore suitable for the research here.

3.2 Circular cylinder

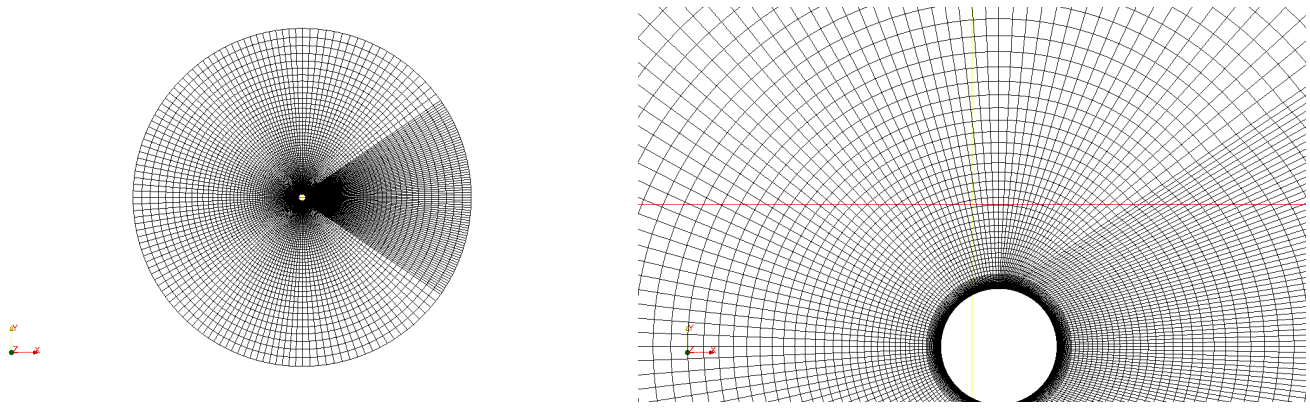
3.2.1 Problem

The first flow problem is the flow around a circular cylinder. The flow is studied at two different Reynolds numbers, namely Re 40 and Re 10^6. In the first case the flow will be laminar while in the latter case the flow is turbulent.

3.2.2 Laminar simulation at *Re* 40

The first case is the laminar flow around a circular cylinder.

Set up In this case the flow solver used is the laminar solver. This solver can be used since Reynolds 40 is a laminar case. In case an uRANS model (for example) would be chosen, the result will be the same, but it will take longer to find the result.
The mesh used in this case is a *O* mesh and is shown in figure 3.1a along with a close up of the mesh at the cylinder wall in figure 3.1b.



(a) *O mesh* (b) *Zoom*

Figure 3.1: *Mesh used in the circular cylinder case of Re* 40

As can be seen in figure 3.1, there is a refinement in the wake of the mesh as well as very close to the wall. The cylinder has a diameter of $2m$. The first ring of blocks has a height of $0.25m$ measured from the cylinder wall. There are 200 cells in the circumferential direction and 100 in the direction normal to the wall. These are divided with 30 cells on the inner part of the mesh and 70 in the outer part. The domain is 50D in x and y direction and $0.1m$ thick. Only one cell is used in this direction since it is a two dimensional case. OpenFOAM is a 3D solver so in order to run a 2D case, it is necessary to specify a third direction.
The mesh has some steps in size, but in this case, it does not seem to be a problem as will be shown later. This mesh has to be refined and redesigned for a turbulent case.

The velocity is $1\frac{m}{s}$ so that the kinematic viscosity becomes $\nu = 0.05$ in order to get a Re_D of 40. The boundary condition on the cylinder surface is the zero gradient boundary condition for pressure and a fixed value of $(0; 0; 0)$

for the velocity. At the outer boundary of the domain, a zero gradient for the pressure is used and the so-called inletOutlet for the velocity. This inletOutlet boundary condition uses a fixed value if the velocity vector is pointed into the domain and a zero gradient if the velocity vector is pointed outwards. The simulation ran for the first $10sec$ with a $\Delta T = 0.0002sec$ and after that for an additional $90sec$ with a $\Delta T = 0.002sec$.

Results The convergence of this case is shown in figure 3.2. It can be seen that the residuals for the velocity components and the pressure are well converged. The jump in the residuals can be explained from the jump in time step as well as the restarting point of the computations. A similar explanation is true for the jump around 40 seconds since the computation was stopped and started at this point as well. The results for the velocity and the pressure distributions are given in figure 3.3.

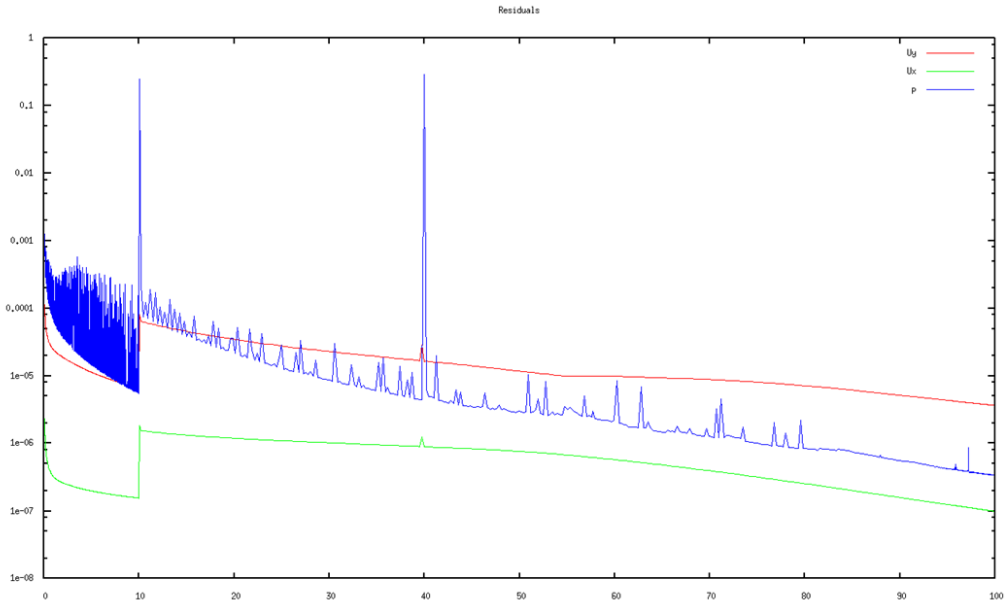
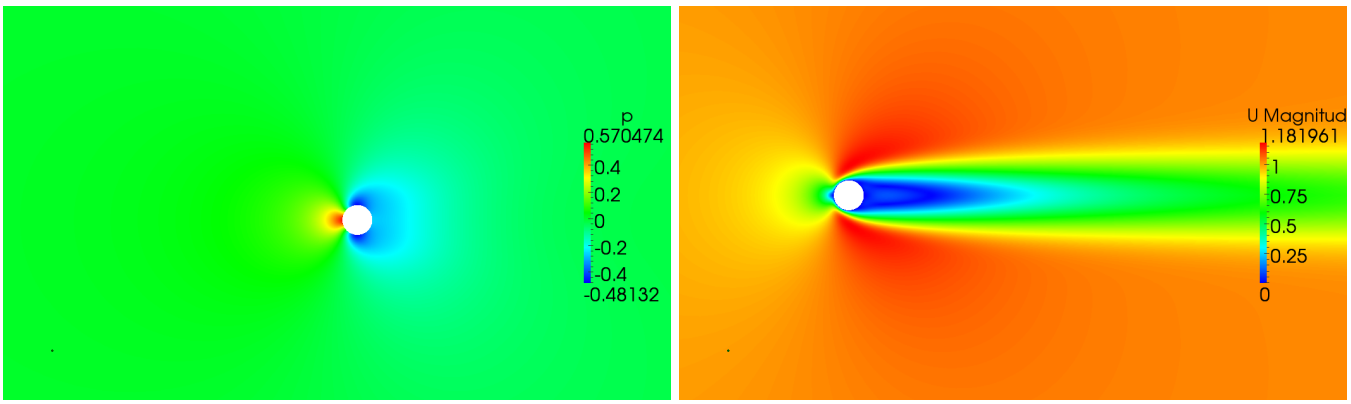


Figure 3.2: *Convergence for the circular cylinder case with Re* 40



(a) *Pressure distribution* (b) *Velocity distribution*

Figure 3.3: *Results for pressure and velocity in the circular cylinder case with Re* 40

The pressure distribution around the cylinder is calculated as well and is given in figure 3.4. The separation angle and the length of the bubble are also measured (table 3.1) as well as the drag coefficient. It should be noted that the bubble length is the length from the cylinder wall up to the end of the bubble. In [8] a model is developed to simulate the flow around a circular cylinder at $Re40$. A comparison with experimental data is also made as well as with other models. In this report, the OpenFOAM simulation is compared to the experimental data found in [8] as well as with the model used in [8]. The pressure coefficient distribution can also be found in [8] (figure 3.4) and compared to that found with OpenFOAM. It can be seen that the maximum and minimum pressure coefficients are very similar so the results seem quite good. It can also be seen that the separation angle and bubble length are very similar to the ones of the simulation in [8]. A spread in experimental data can be seen, but the found values in the OpenFOAM simulation seem to be in a reasonable range.

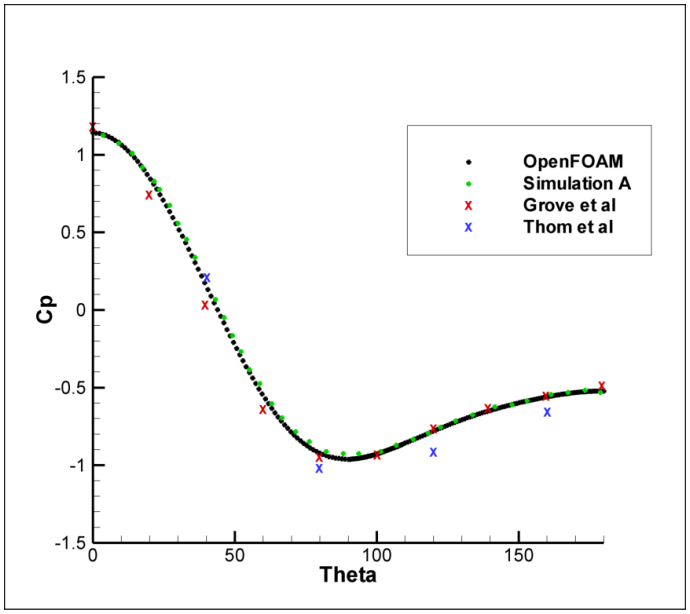


Figure 3.4: *Results for pressure distribution at the circular cylinder wall with Re* 40

Variable	OpenFOAM simulation	Simulation A	Grove et al	Coutencau et al
Separation Angle θ (in degrees)	126.67	126.67	137.2	126.5
Bubble Length L (in diameters)	2.24	2.22	1.54	2.13
C_D	1.53	1.58	-	-

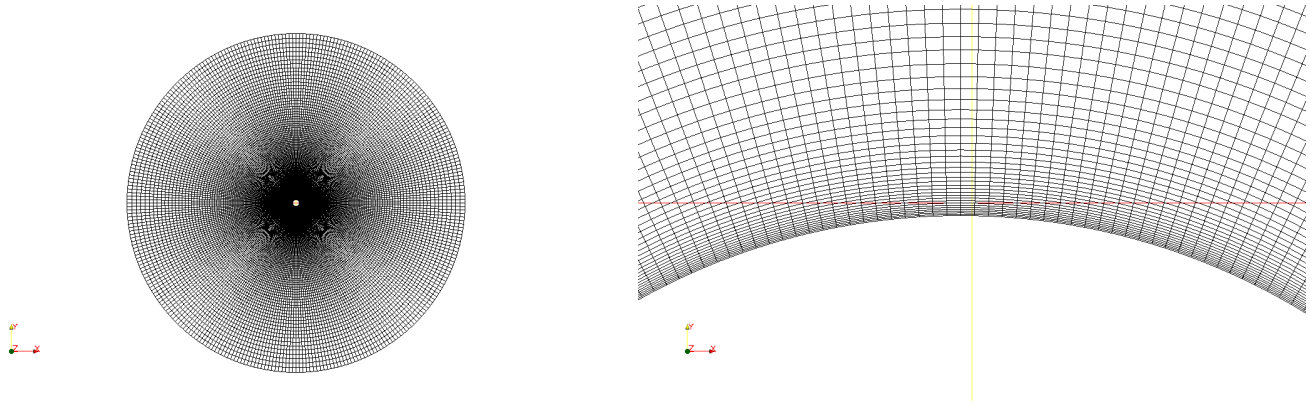
Table 3.1: *Summary results of Re* 40 [8]

Conclusion From this test, it can be concluded that the laminar solver of OpenFOAM is accurate for the flow around the circular cylinder as modeled here. The test results are in good agreement with test results as well as with other models.

3.2.3 uRANS simulation at *Re* 10^6

The next case studied is a turbulent case. The main point is to compare the results of this simulation with the results found in literature [13] to determine if the set up of the case is correct and to get to know OpenFOAM for turbulent cases.

Set up This case requires a better mesh. If the mesh of the Reynolds 40 case would be used, the results would have been bad if the solution would have converged at all. The mesh has been revised and can be seen in figure 3.5a in complete and zoomed at the cylinder wall in figure 3.5b. The domain size is the same as in the Reynolds 40 case.



(a) *O mesh* (b) *Zoom*

Figure 3.5: *Mesh used for the circular cylinder case with Re* 10^6

Again, the mesh has been divided in several blocks. Around the circular cylinder 300 points are used. In the wall normal direction, 30 points are used for the first $0.25m$ and 150 points in the rest of the domain.

The kinematic viscosity ν is set to $1.7e-6\frac{m^2}{s}$ and the velocity U to $8.5\frac{m}{s}$ in order to get a Reynolds number Re_d of $1e6$. The boundary conditions for this case can be found in table 3.2. It should be noted that the velocity is strictly in x_1-direction and that the inletOutlet boundary condition prescribes a fixed value if the velocity vector at that cell points into the domain and a zero gradient boundary condition when the velocity vector is pointed outwards. The 'calculated' boundary condition tells the program to calculate the value from the other solved values. The use of wall models can be justified by the fact that the y^+ distribution is between 10 and 40.

Variable	Internal field	Edge of domain	Wall
U	uniform 8.5	inletOutlet 8.5	0
p	uniform 0	zero gradient	zero gradient
ϵ	uniform 0.000765	inletOutlet 0.000765	Epsilon wall function
k	uniform 0.00375	inletOutlet 0.00375	kqR wall function
ν_t	calculated	calculated	nutk wall function

Table 3.2: *Boundary conditions for the circular cylinder case at* $Re_d = 1e6$

The time step in this case is $0.00023529s$, chosen based on the calculations in the reference paper [13]. The end time was $70.855s$ and since there will be turbulence, the mean values were taken from $35.2935s$ until the end time.

Results The residuals are given in figure 3.6. As can be seen in this figure, the residuals do not go down as in the case of Re 40. Instead they move to an equilibrium point and oscillate. These oscillations are caused by the vortex shedding in the unsteady case. Because of the vortex shedding, it is useless to measure the bubble length at a certain time. Instead, all the computations and plots shown here are time averages and averaged from 35 seconds until the end time. The mean pressure and velocity distributions are displayed in figure 3.7 and the pressure coefficient in figure 3.8.

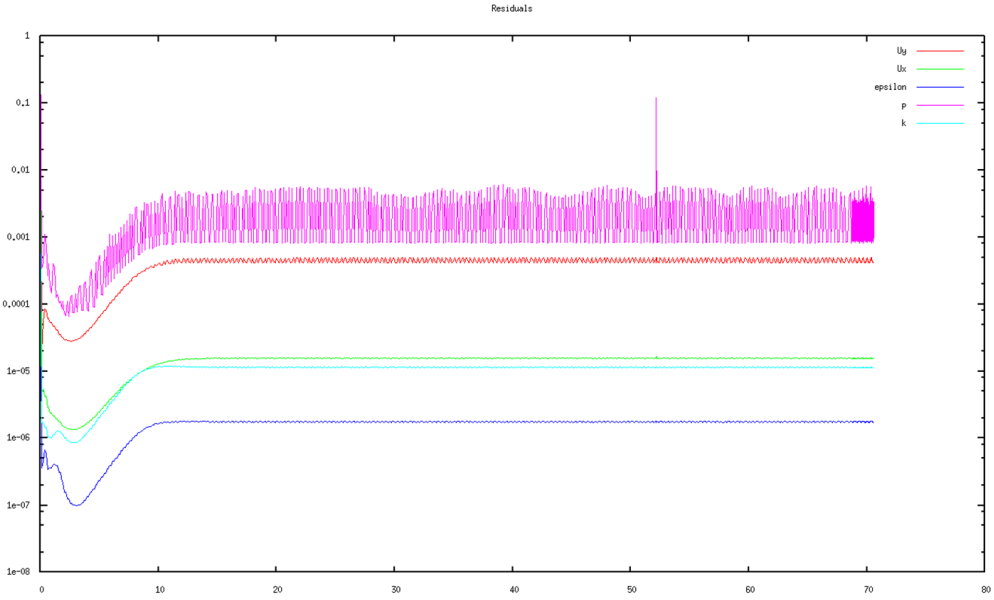
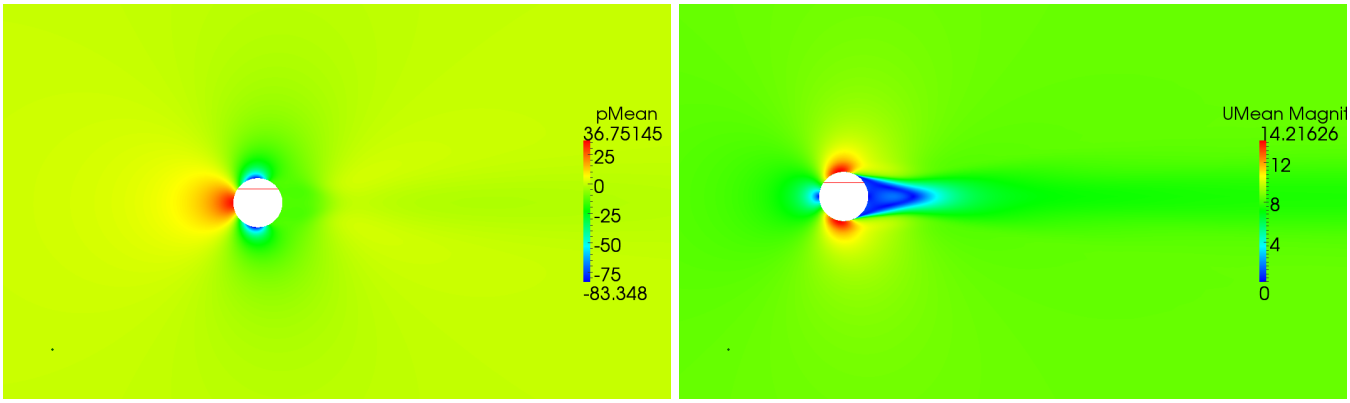


Figure 3.6: *Convergence for the uRANS circular cylinder case with Re* 1e6

Besides these data, also some velocity profiles are extracted from the data. The points at which they are extracted are deduced from the paper [13] as well and can be found in figure 3.9. From figure 3.9a can be seen that the model has to much artificial dissipation, the black line (our results) are a little smooth in comparison with the results of the paper. The gradient at those points is very high (since it is the area of the bubble). In the graph, a smooth transition is clearly visible which is caused by the numerics used to solve the problem. Furthermore it seems that the bubble width is longer since the velocity peak in figure 3.9b is lower than that of the paper. The velocity in y-direction also differs in the peaking points (figures 3.9c and 3.9d). The rounding can be caused by the artificial dissipation again.
The lift coefficient and drag coefficient are also measured and plotted as a function of time in figure 3.10. The blue part is the part after the second restart.



(a) *pressure* (b) *Velocity*

Figure 3.7: *Results for pressure and velocity of the circular cylinder case with Re* 1e6

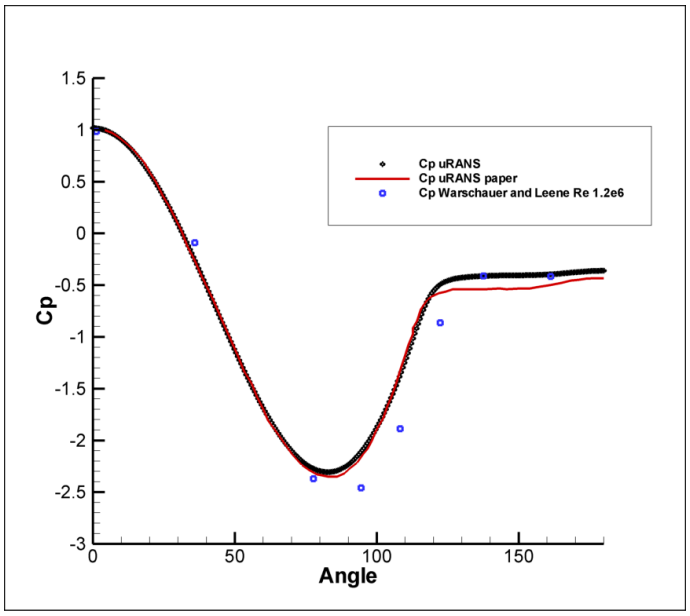


Figure 3.8: *Results for pressure distribution at the circular cylinder wall with Re* 1e6 *with the ones of the paper [13]*

The Strouhal number is defined as:

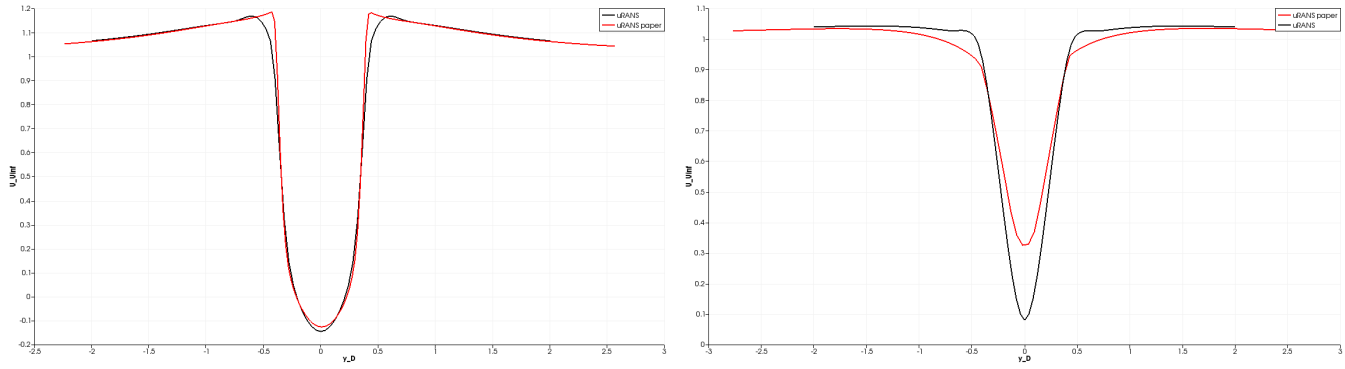
$$St = \frac{fL}{U} \tag{3.1}$$

with f being the frequency of the vortex shedding, L the diameter of the circular cylinder and U the freestream velocity. The frequency of the vortex shedding can be determined from the lift or drag coefficient. In order to check the results both methods were used. The results of this case are summarized in table 3.3. It should be noted that the bubble length is the length from the center of the circular cylinder to the end of the bubble.

Variable	OpenFOAM simulation	uRANS paper	LES paper	Exp. Shih et al	Exp. others see paper
Separation Angle θ (in degrees)	122.151	-	-	-	-
Bubble Length L (in diameters)	1.41	1.37	1.04	-	-
C_D	0.33	0.4	0.31	0.24	0.17-0.40
C_{Lmax}	0.078	0.12	-	-	-
St	0.344	0.31	0.35	0.22	0.18-0.50

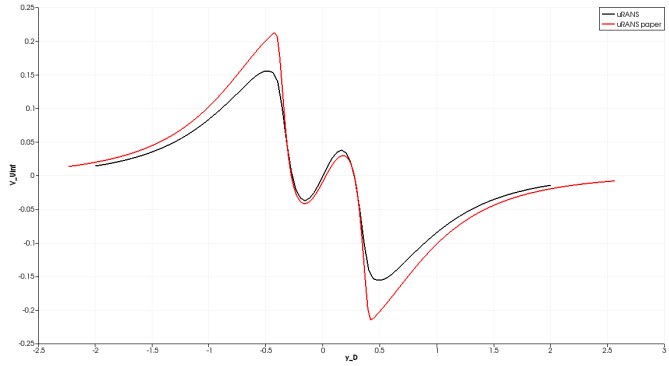
Table 3.3: *Summary results of Re* 1e6*, computations from the paper and experiments [13]*

Conclusion With all the results in place, the computations seem quite reasonable. If the pressure coefficient of figure 3.8 is compared to the results of the uRANS computation of the paper, it can be found that the pressure recovery is a little higher. The last part of the graph looks much more like experimental data found by Warschauer and Leene

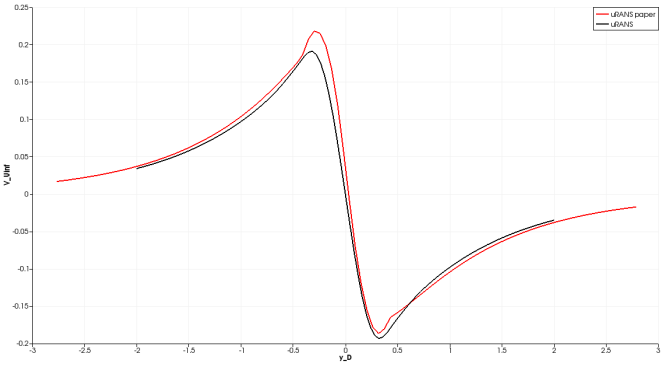


(a) u_x *at* $\frac{y}{D} = 0.75$

(b) u_x *at* $\frac{y}{D} = 1.50$

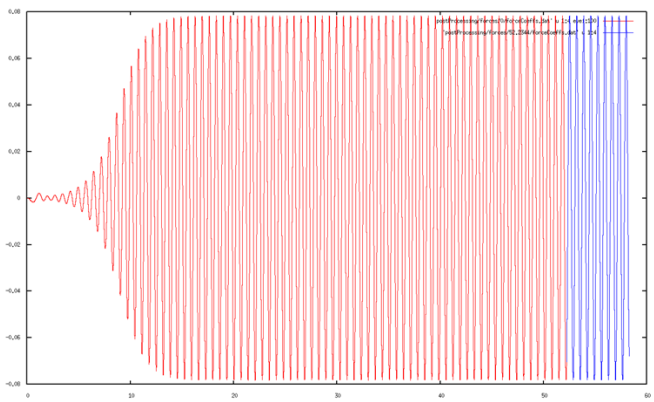


(c) u_y *at* $\frac{y}{D} = 0.75$

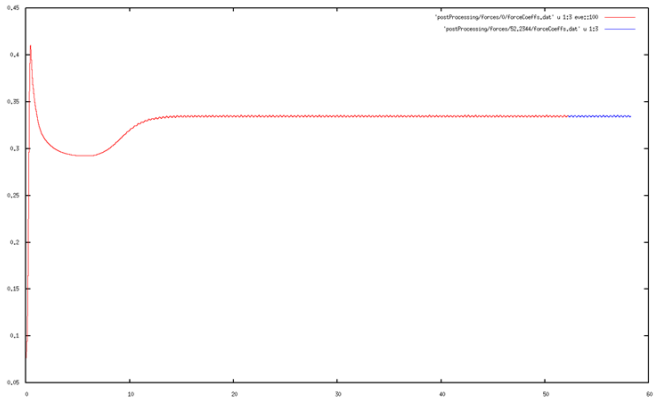


(d) u_y *at* $\frac{y}{D} = 1.50$

Figure 3.9: *Results for velocity distribution in the circular cylinder case with Re* 1e6 *and the results of the paper [13]*



(a) C_L



(b) C_D

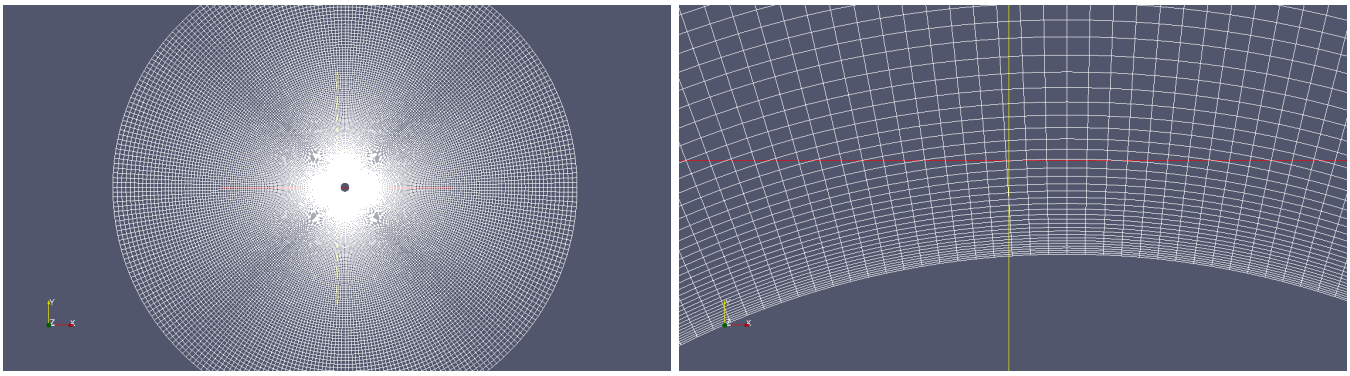
Figure 3.10: *Results for lift and drag coefficient of the circular cylinder case at Re* 1e6

at *Re* 1.2e6. Also when the velocity profiles are studied, it can be seen that a little too much artificial dissipation is added to the system. In next computations, this could be solved by decreasing the cell spacing resulting in a finer grid or by using higher order schemes. The schemes used in the calculations performed in the circular cylinder case are second order. A better result could be found if a third or fourth order scheme is used. However, both increasing the number of cells as well as increasing the order of the schemes will increase the calculation time.
The uRANS computations of the paper need to be inspected carefully, because they do not seem entirely correct. In figures 3.9c and 3.9d it can be clearly seen that the profiles of the paper are not symmetric. Since this is a uRANS simulation, it is expected that these profiles would be symmetric and thus the results should be interpreted with care. The values all seem to be in the right order of magnitude resulting in the conclusion that the model as it is set up here produces some reasonable results.

3.2.4 LES simulation at *Re* 10^6

The main point of this case is to compare the results of this simulation with the results found in literature [13] and the uRANS case in order to determine if the set up of the case is correct.

Set up The major difference in domain between this case and the uRANS case is the domain in z-direction. An LES computation can not be done in two dimensions so the third dimension is extended in comparison to the uRANS computation. The domain in z-direction is set to 4 times the diameter of the circle, as in the reference paper [13], while the rest of the dimensions stay the same. The boundary condition for the front and back patch are now set to cyclic in OpenFOAM which means that will treat these patches as a periodic boundary. The amount of cells used are in respectively radial, circumferential and z direction: 230 elements, 300 elements and 48 elements. Again, a full view and a detailed view of the mesh can be found in figure 3.11. The cell distribution is uniform in z-direction.



(a) *O mesh* (b) *Zoom*

Figure 3.11: *Mesh used for the LES circular cylinder case with Re* 10^6

The LES model chosen in this case is the one equation eddy model. This model solves the conservation equation for the turbulent kinetic energy k. This model is used with the van Driest damping function which applies a cube root volume filter. The initial condition for the velocity and pressure is set to the average pressure and velocity field of the uRANS computations. This is done because the main flow field should not change too much between a uRANS and LES case and by applying this initial solution, it is expected to reach convergence faster. The boundary conditions of this setup are displayed in table 3.4.

Variable	Internal field	Edge of domain	Wall
U	mapped uRANS 8.5	inletOutlet 8.5	fixed value 0
p	mapped uRANS 0	zero gradient	zero gradient
k	uniform 0.00375	inletOutlet 0.00375	kqR wall function
ν_{SGS}	uniform 0	calculated	calculated

Table 3.4: *Boundary conditions for the LES circular cylinder case at* $Re_d = 1e6$

The time step in this case is set to $0.001sec$ with a time frame of $0sec$ to $75sec$. Again an averaging process is started, this time at $30sec$ in order to acquire enough data for the statistics.

Results The residuals of this case are given in figure 3.12. Since these residuals are compared to the initial field, the convergence does not automatically go down a lot. The initial field is already the converged solution of the uRANS solution. Especially in a very turbulent LES case, the residuals are usually not the best way of telling if a case converged since the solution fluctuates a lot. Usually another parameter is used in order to see if the case reaches convergence. This other parameter can be a coefficient such as the drag or lift coefficient but also the pressure in a certain point (probe location).
Because the case is three dimensional, the solution, for example the bubble length, also varies with the span. The average bubble shape can be seen in figure 3.13 and as can be seen, it varies with the spatial coordinate z. The average bubble length is calculated to be $0.75D$ measured from the center of the circular cylinder. The plotted area is the surface at which the mean velocity in x-direction is zero meter per second.
The pressure coefficient is shown in 3.14 and is plotted together with the uRANS results and the experiments of Leendert and Warschauer. The velocity profiles at the different substations can be found in figure 3.15.
The last two figures (figure 3.16a and figure 3.16b) show respectively the lift and drag coefficient.

Conclusion When graphs of this simulation are compared to the results of the paper, it can be noticed immediately that the differences are significant. The length of the bubble is much more short than the bubble length predicted by experiments and the paper. This can also be seen in the different velocity profiles in figure 3.15. It seems that with the used setup, the flow is too long attached to the body which reduces the length of the bubble significantly. The pressure coefficient is going far below the LES case of the reference paper. It can be seen that it follows the high Reynolds number experiments from Leener and Warsteiner. It can be concluded from the results that too little

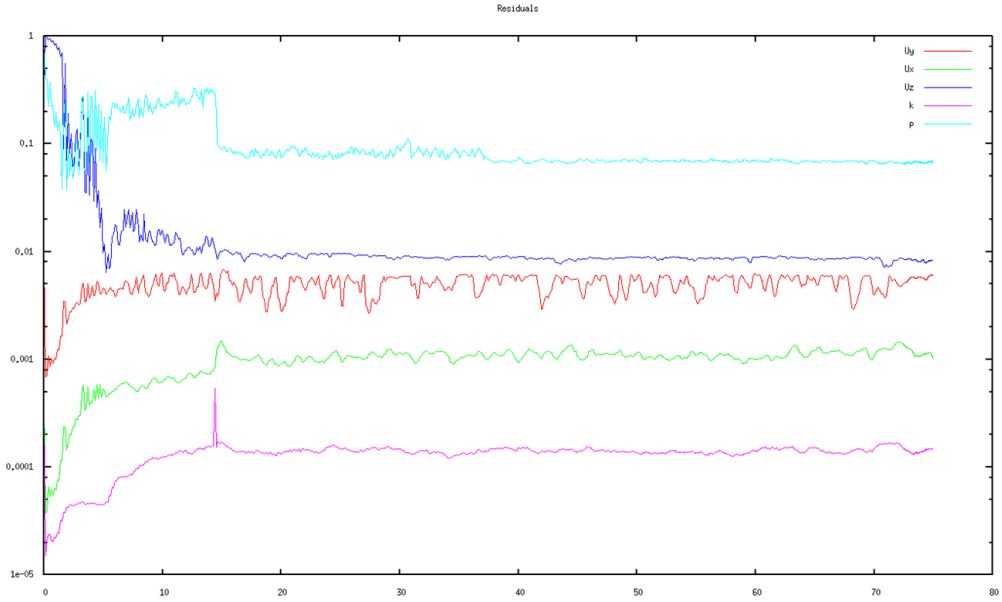


Figure 3.12: *Convergence for the LES circular cylinder case with Re* 1e6

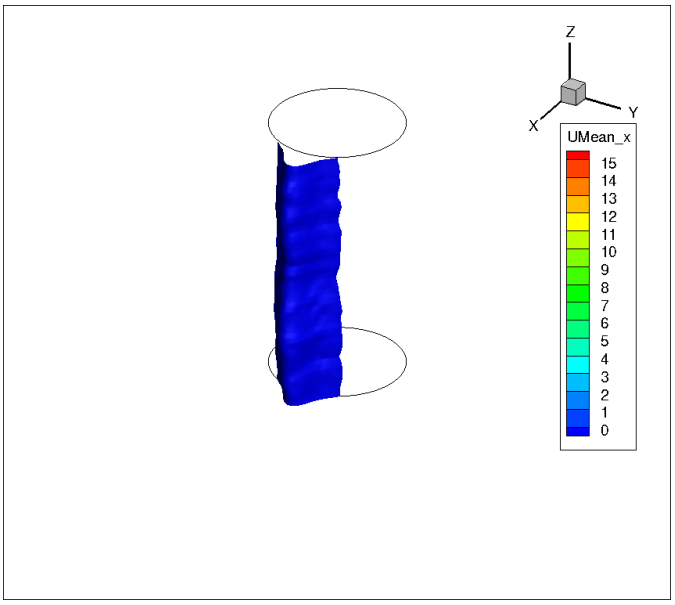


Figure 3.13: *Average bubble shape of the circular cylinder LES case for Re* 1e6.

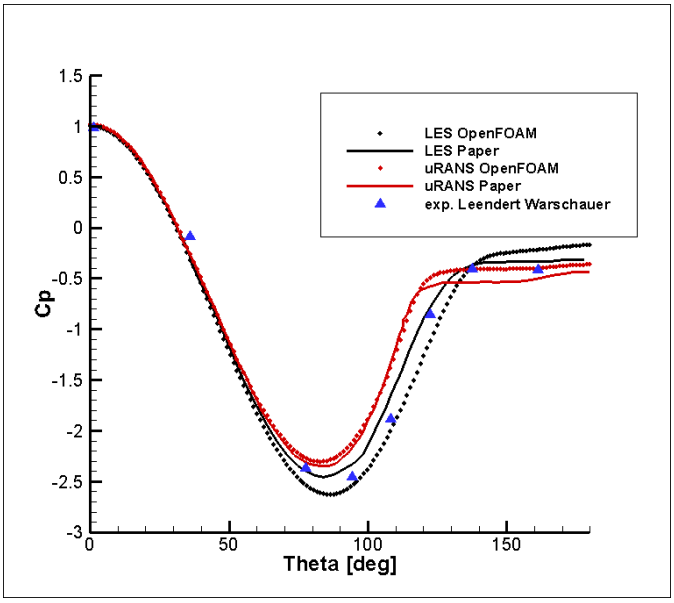
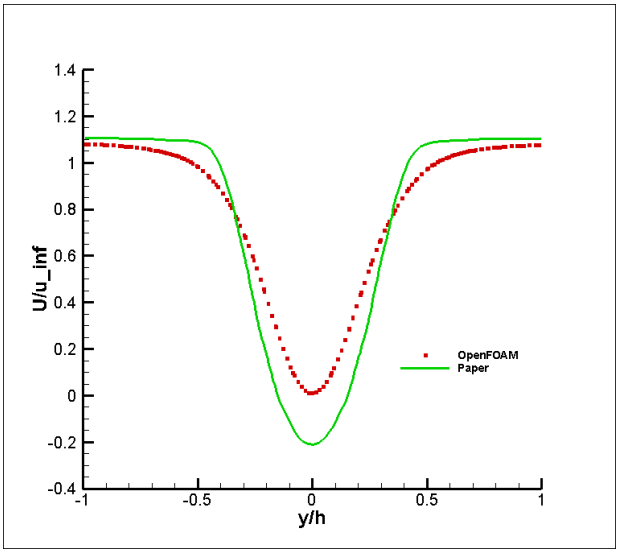
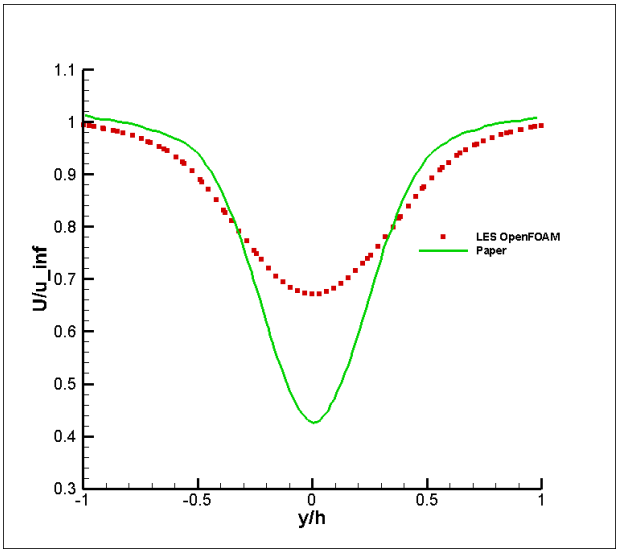


Figure 3.14: *Results for pressure distribution at the circular cylinder wall with Re* 1e6 *in LES case with the ones of the paper [13]*

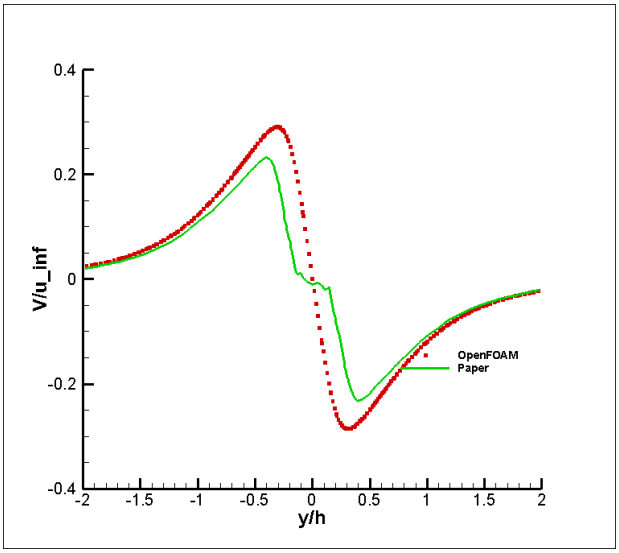
turbulence is generated. This can be caused by the fact that the van Driest damping function is introduced. Maybe it would be better to use the cube root volume method as the delta function. Also the initial condition of the subgrid scale viscosity could be set to a very small value. This will trigger the turbulent behaviour earlier. Another possibility is the use of a different LES method such as the Spalart Allmaras, $k-\omega$ or Smagorinsky could be used to improve the results.



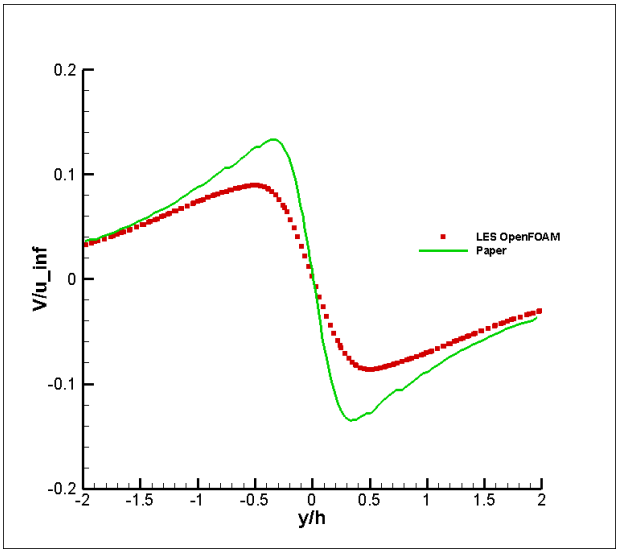
(a) u_x *at* $\frac{y}{D} = 0.75$



(b) u_x *at* $\frac{y}{D} = 1.50$

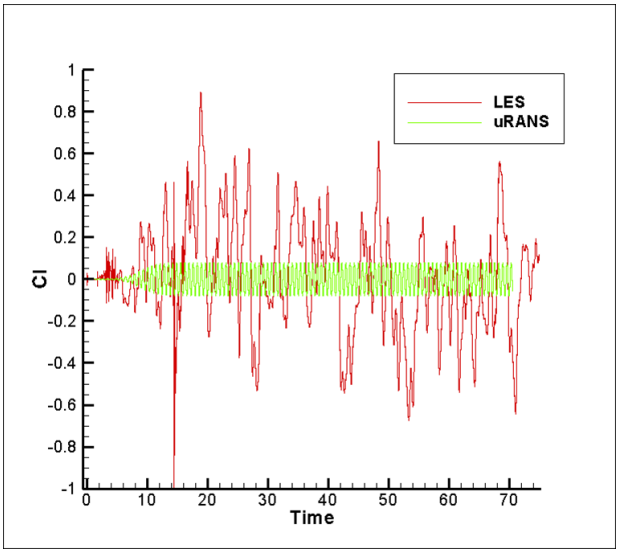


(c) u_y *at* $\frac{y}{D} = 0.75$

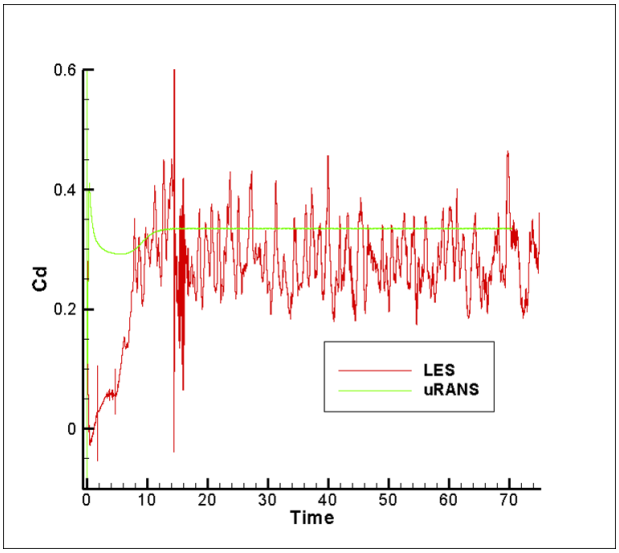


(d) u_y *at* $\frac{y}{D} = 1.50$

Figure 3.15: *Velocity distribution in the LES circular cylinder case with Re* 1e6 *and the results of the paper [13]*



(a) C_L



(b) C_D

Figure 3.16: *Results for lift and drag coefficient of the circular cylinder case at Re* 1e6

3.3 Backward facing step

3.3.1 Problem

The second problem analyzed in this report is the backward facing step. This test case is very widely studied and forms also one of the major test cases in numerical setups. The idea is to get a comparison between different types of LES solvers implemented in OpenFOAM. The results of the simulations can also be used in order to compare them to other types of solvers and the experimental results. There are measurements done at different sections of the backward facing step. This data is available and the results at these stations will be compared to the numerical data. The data available is (seen from the step) at location x equals $-0.43h$, $0.29h$, $1.1h$, $2.3h$, $4.3h$ and $6h$. The data available contains the average velocity profiles and the root mean square of the velocity (a measurement for the fluctuations of the velocity).

3.3.2 Fine grid case

Set up

Given conditions The setup is based on the experiments in [22] and the already performed calculations done at CIRA. The domain consists of a backward facing step with a step height of $1h$. The initial length is $34h$ while the rest of the domain has length $38h$. The width of the domain is $2h$. The factor h is $0.035m$ and is also used to calculate the Reynolds number. The domain is shown in figure 3.17.

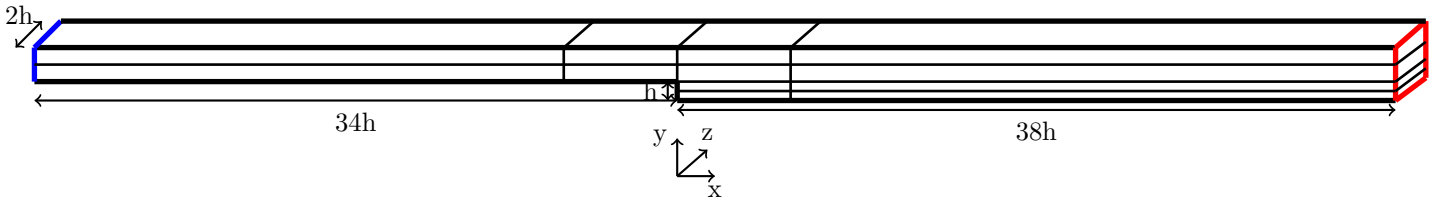


Figure 3.17: Backward facing step

The Reynolds number of this case is 40.000 with an inlet velocity of $50m/s$. With the step height h as characteristic length it is possible to calculate the kinematic viscosity. This viscosity is needed in OpenFOAM in order to determine the Reynolds number. The stated boundary conditions can be found in table 3.5.

Variable	Inflow	Outflow
U	$50\frac{m}{s}$	
p	$99.990Pa$	$100.400Pa$
T_s	$520K$	
M	$0,1$	
M	40.000	

Table 3.5: *Inflow and outflow conditions for the backward facing step.*

The simulations for this case will be done with an incompressible LES setup for the backward facing step. The incompressibility can be assumed because of the low Mach number. The viscosity given in OpenFOAM can be calculated with the help of the stated conditions. The kinematic viscosity (ν) is set to $4.375 \cdot 10^{-5}\frac{m^2}{s}$ in order to have the same Reynolds number in the OpenFOAM simulations. In order to check if this viscosity is a realistic value, a quick check was performed with the help of the ideal gas law and Sutherlands formula which can be found in 3.2.

$$\begin{aligned} P &= \rho R T \quad \text{ideal gas law} \\ \mu &= \mu_0 \frac{T_0 + C}{T + C}\left(\frac{T}{T_0}\right)^{\frac{3}{2}} \quad \text{Sutherlands formula} \end{aligned} \tag{3.2}$$

The universal gas constant R is $287.058J/(kg \cdot K)$, while the reference temperature T_0 and the reference viscosity mu_0 are stated as $291.15K$ and $18.27\mu Pa \cdot s$ respectively. The kinematic viscosity can now be calculated as in 3.3.

$$\rho = \frac{P}{RT} = 0.66986 \frac{kg}{m^3}$$
$$\mu = \mu_0 \frac{T_0 + C}{T + C} \left(\frac{T}{T_0} \right)^{\frac{3}{2}} = 2.80149 \cdot 10^{-5} Pa \cdot s \tag{3.3}$$
$$\nu = \frac{\mu}{\rho} = 4.1822 \cdot 10^{-5} \frac{m^2}{s}$$

The ν used in the OpenFOAM calculations is $4.375 \cdot 10^{-5} \frac{m^2}{s}$ while the calculated ν is equal to $4.1822 \cdot 10^{-5} \frac{m^2}{s}$. It is assumed that this is a good approximation for the kinematic viscosity. The actual Reynolds number is also calculated and was found to be 41843.97.
The time simulated is $0.42s$ with the last $0.02s$ being the time over which the averages of the pressure and velocity are taken. The time step used for the averaging is the dimensionless time step of 0.0014 where the dimensionless time step is given in 3.4. This will result in 20408 steps during the averaging period. Before the time of $0.4s$, the time step can be chosen as desired.

$$\Delta t = \frac{\Delta t^* h}{U} = 9.8 \cdot 10^{-7} s \tag{3.4}$$

Method The used method for solution is the IDDES method implemented in OpenFOAM. The function to determine the filter width is the maxDeltaxyz function in OpenFOAM which uses the maximum size of Δx, Δy and Δz. The reason for this choice is that the results can be compared with the results from an earlier simulation done at CIRA. After the IDDES run, another run can be made with a different LES model to compare the results and find the differences between the models. The rest of the setup will be held constant in order to make a fair comparison.

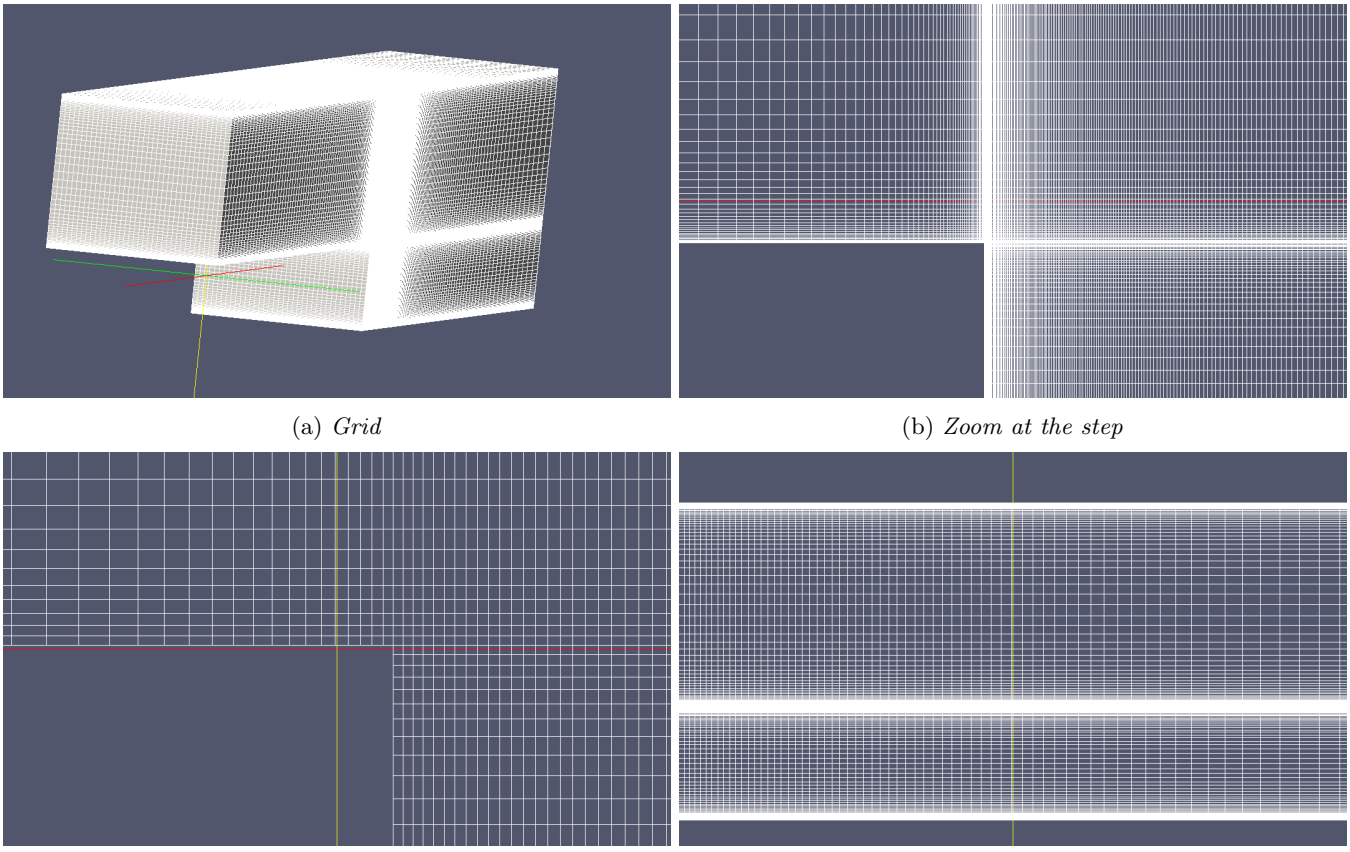
Mesh The mesh is very important in any CFD case so in this case it is no different. As can be seen in figure 3.17, the domain is divided in 12 blocks in such a way that it will be easy to mesh them with the blockMesh utility from OpenFOAM and get a good resolution. There will be no wall functions used in this simulation so in order to get a good result for the boundary layer, the mesh should be very fine at the walls. The height of the cell aimed for is a y^+ of at maximum 1 corresponding to $1.5 \cdot 10^{-5} m$. This can be achieved by using the formulas in appendix B.
As mentioned before, the cell size in the vicinity of the wall should be very small in order to have enough resolution. The other important part is the step until the point of reattachment (should be around $5.3L/h$). The mesh should be quite fine in this region, therefore it has been chosen to set the block lengths of the blocks just before and the after the step to $6L/h$. In this region, the mesh will be small in order to get a good approximation. The first two blocks and last four blocks have a coarser mesh because the solution in these areas is less important. The number of cells in the z-direction is set to 48 cells with equal spacing in order to keep this constant with other computations. The number of cells from block to block is shown in 3.6 with the blocks numbered as shown in figure 3.18. With this setup the total number of cells is 3.6 million and some parts can be seen in figure 3.19.
The mesh was then checked with the checkMesh utility from OpenFOAM. It was found that some regions have a large aspect ratio which should be prevented, but because these cells were located on the far end of the domain it was assumed that the influence would be minimal.

1	3	7	11
0	2	6	10
		5	9
		4	8

Figure 3.18: *Distribution of the blocks over the backward facing step.*

With the distribution of the cells as shown in table 3.6, the mesh is very fine near the wall and at the step.

Boundary conditions The next step is to find the boundary conditions. Just like with the LES case for the circular cylinder, the input of a RANS solution is used to set the boundary conditions and initial conditions for the LES case. The first calculation therefore was a steady run with the SIMPLE algorithm. This was performed on a very small grid in order to get a fast initial solution. This calculation was performed with the Spalart Allmaras RANS setup. The



(a) *Grid* (b) *Zoom at the step*

(c) *Large zoom at the step* (d) *Cross section behind the step*

Figure 3.19: *The used grid*

Block	# cells x-direction	# cells y-direction	# cells z-direction
0 & 1	50	50	48
2 & 3	100	50	48
4 & 5	350	50	48
6 & 7	350	50	48
8 & 9	75	50	48
10 & 11	75	50	48

Table 3.6: *Cell distribution in the backward facing step*

result was then mapped to the fine grid resulting in the initial and boundary conditions which are displayed in table 3.7. It should be noted that the inflow and the outflow areas are the areas displayed in blue and red respectively in figure 3.17. The cyclic patch is again used for the front and back of the domain (in z-direction). The flowRateInletVelocity boundary condition is the boundary condition used at the inlet because of the boundary layer at the wall. With this boundary condition, the mass flow rate is kept constant. The mass flow rate is given by the coefficient and a value of 0.2205 is consistent with the inlet velocity of $50m/s$.

Variable	internal field	inflow	outflow	wall
U	mapped steady velocity	flowRateInletVelocity 0.2205	zero gradient	fixed value 0
p	mapped steady pressure	zero gradient	fixed value 0	zero gradient
ν_{SGS}	uniform $1 \cdot 10^{-6}$	zero gradient	zero gradient	zero gradient
$\tilde{\nu}$	mapped steady $\tilde{\nu}$	fixed value 0	zero gradient	fixed value 0

Table 3.7: *Boundary conditions for the IDDES simulation*

Numerical schemes and solvers The numerical scheme chosen for the time schemes is the pure Crank Nicolson scheme. The gradient schemes are Gauss linear schemes just like the divergence schemes. The Laplacian schemes are Gauss linear corrected schemes. The numerical solver used for the velocity is the PBiCG method which is also used for the turbulent viscosity. For the pressure, the GAMG and the PCG solvers were tried to check which one has more

accuracy and saves computation time. It turned out that the best option would be the PCG solver.

Time step With all the parameters filled in and the setup determined, the starting time step had to be determined. This was done by trying different time steps. The idea was to keep the maximum Courant number below one which was assumed to be the best for this simulation, and try to find a value for the time step as high as possible in order to speed up the calculations. The time step would be adjusted before the averaging period. However after several test runs it turned out that the time step had to be very small for the simulation not to blow up. It turned out that the time step had to be $1 \cdot 10^{-7} s$ for the simulation to not diverge directly. This is roughly a factor 10 smaller than the averaging time step needed. This posed a problem because the simulations would take a lot of time steps to find a solution. It was found that the time step used should remain below a maximum Courant number of 0.4 to keep a stable converging setup.

Processors The case was decomposed to speed up the calculation. A performance graph was made in order to evaluate the optimum amount of processors to be used. When more processors are used, time is saved because each processor has to do less calculations. However there is an optimal number of processors. With the increment in the number of processors used, the communication time between the processors is increased as well. This effects the efficiency of the calculations. From some preliminary tests it turned out that 8 processors would be good for this case, also because the time saved with 16 processors was not significant and the waiting period for the queue at the calculation computer of CIRA would be less if 8 processors were used. It should be noted however that these tests were done with a little different setup than the final case, because the steady state run was not finished yet.

Results

The velocity profile at $x/h = -0.35$ station is given in figure 3.20 determined from the experimental as well as from the OpenFOAM steady simulation. This solution shows that the initial solution is a little high in the center of the flow, but this can change in the actual solution. This profile shows that a reasonable accurate initial setup is found so this is used. It was also found that the pressure drop over the backward facing step is around $294Pa$. From the experiments it turns out that this step should be around $400Pa$, so this should improve as well. The turbulent viscosity is also plotted as can be found in figure 3.21.

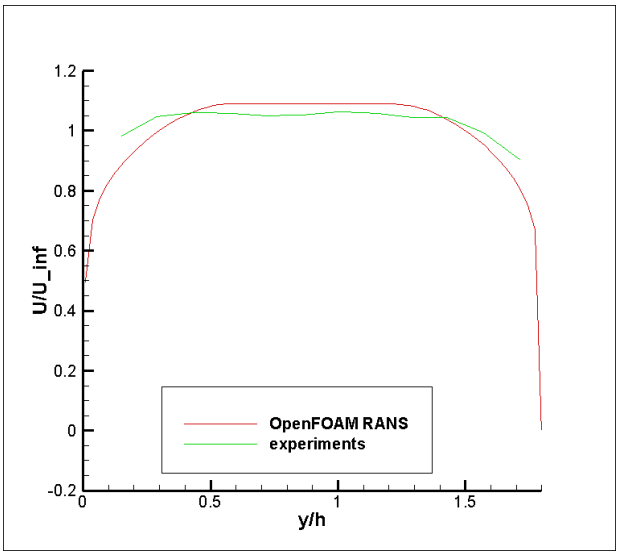


Figure 3.20: *Velocity profile at $x/h = -0.35$ for the steady case*

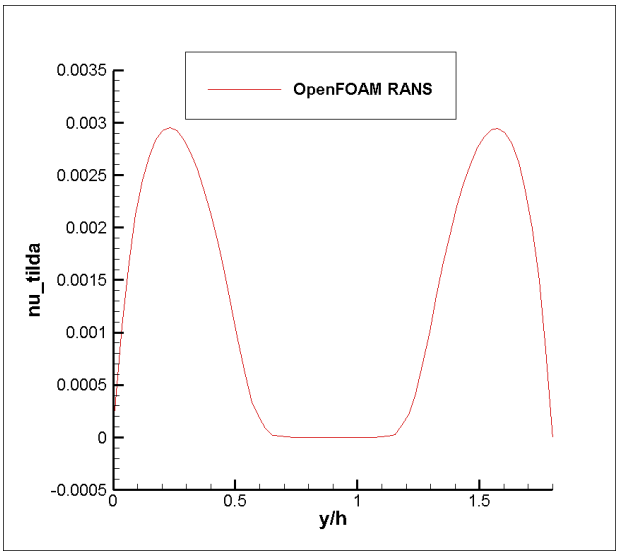


Figure 3.21: *$\tilde{\nu}$ profile at $x/h = -0.35$ for the steady case*

The final simulation was setup and run, but it took a very long time to finish. There were a lot of iterations needed to find a solution for the pressure which results in a very long calculation time. Also because many time steps have to be calculated which takes much time, the results of this simulation are not found.
In order to have some results, the decision was made to try and simulate the problem on a bigger coarser mesh. This will be discussed now.

3.3.3 Coarse grid case

Set up

The case setup is very similar to the setup for the fine mesh. The major difference between the two cases is, of course, the mesh which is much coarser in order to save calculation time. The devision of the blocks remains the same, the number of cells however is changed significantly as can be seen in table 3.8. The wall spacing aimed for is a y^+ of 30 which is around $4.4 \cdot 10^{-4} m$, which, with the new setup, resulted in a mesh with 224.520 cells. A partial zoom of the mesh can be found in figure 3.22 and it should be noted that this case did not have any cells with too large aspect ratio as the fine mesh had.

Block	# cells x-direction	# cells y-direction	# cells z-direction
0 & 1	25	20	15
2 & 3	50	20	15
4 & 5	150	12	15
6 & 7	150	20	15
8 & 9	37	12	15
10 & 11	37	20	15

Table 3.8: *Cell distribution in the backward facing step with a course mesh*

With the change in cell size, the amount of points is significantly lower than the case for the fine mesh. This saves time in the amount of work done per time step, but, because the cells are bigger, it is also possible to increase the time step which saves time because less time steps have to be taken. The time step taken in this case is $1 \cdot 10^{-6} s$. This run was also run from a steady state solution.
The numerical solvers chosen in this setup are the same as in the large case setup. The numerical schemes are also the same with the exception of the time scheme which is set to a second order backward scheme in the IDDES case, because this is a more commonly used time differencing scheme.
For the boundary conditions, it was chosen to keep them the same as in the fine mesh. It must be kept in mind that the wall cell height in y-direction is quite large so there could be some differences between the boundary layer in the simulation of the experiments and the one of the OpenFOAM simulations.

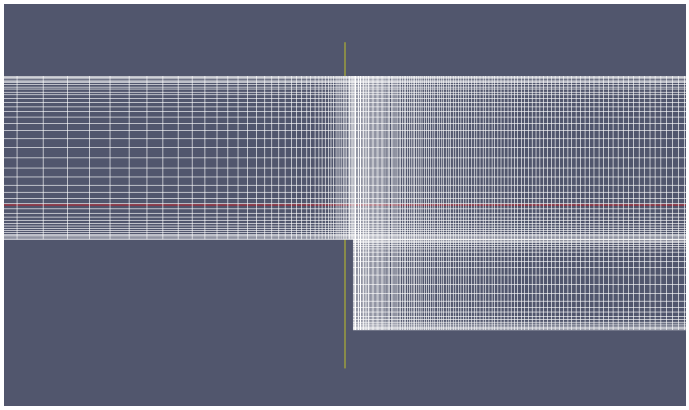


Figure 3.22: *Zoom of the coarse mesh*

After the simulation was run, it seemed that the averaging time was not chosen well. It seemed that the mean velocity profile was not stationary as was expected. In order to see if this suspicion was indeed correct, the simulation time was increased to $0.5s$ which results in an averaging period of $0.1s$ (five times longer than the initial).

Results

The residuals of the simulation are given in figure 3.23. The peaks in the residuals are caused by the different restarts from the simulation. The simulation ran on a server with a limited calculation time so it was restarted every time. The oscillations in the residuals are caused by the flow being not steady.
The next things to check are the velocity profiles. The average velocity profiles provided are at the positions $-0.43h$, $0.29h$, $1.1h$, $2.3h$, $4.3h$ and $6h$. Except the mean velocity profiles, the data also contains the fluctuations expressed as u_{rms} in 3.5. These can be used to see if the turbulence production is accurate.

$$\begin{aligned} U_{mean} &= \frac{1}{T} \int_{t_0}^{t_0+T} U dt \\ u_{rms} &= \frac{1}{T} \int_{t_0}^{t_0+T} \left(U - U_{mean}\right)^2 dt \end{aligned} \tag{3.5}$$

Because the IDDES method is a solving method in 3D, there are fluctuations in three directions. In order to compare the velocity profiles, the average over z-direction was taken. The plotted velocity profiles are the found velocity profiles

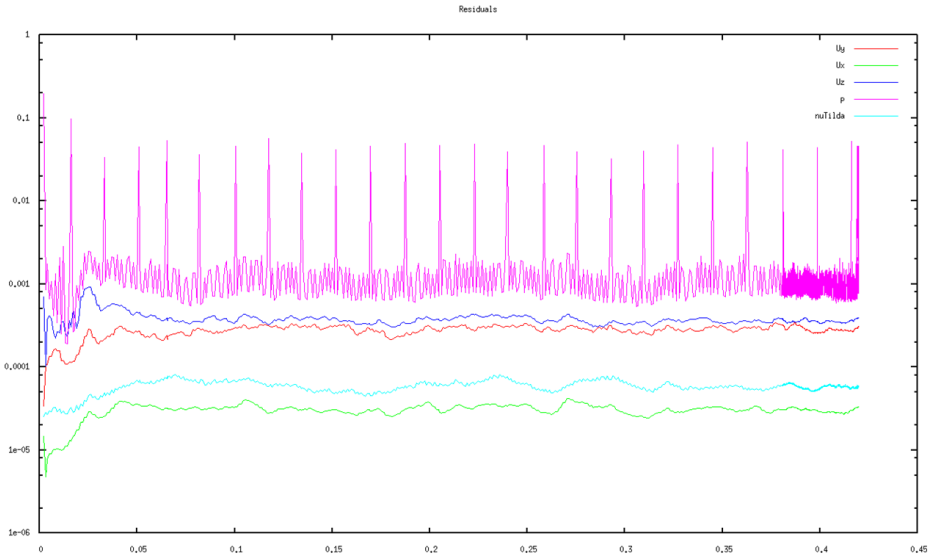


Figure 3.23: *Residuals of the coarse mesh backward facing step simulation*

from average velocity with an averaging time of $0.02s$ and $0.1s$ as well as the experimental results. The first velocity profile checked is at the location before the step. The velocity profile here is steady, because the step has not been passed. The velocity profile is shown in figure 3.24.

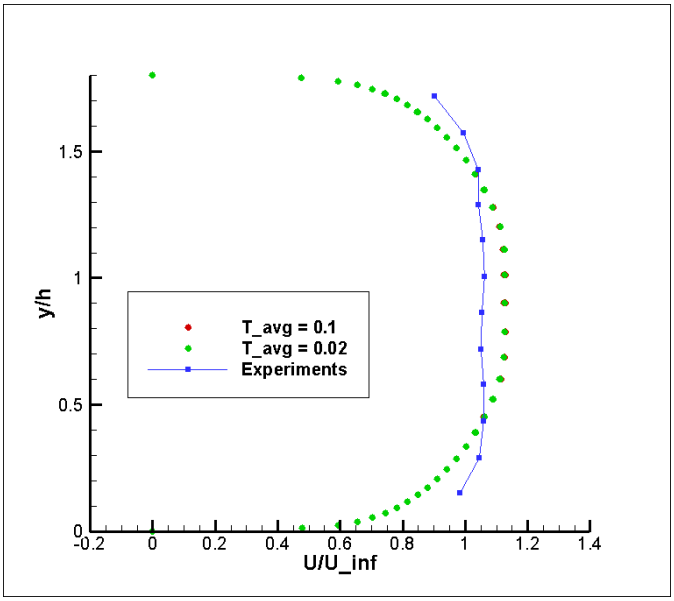
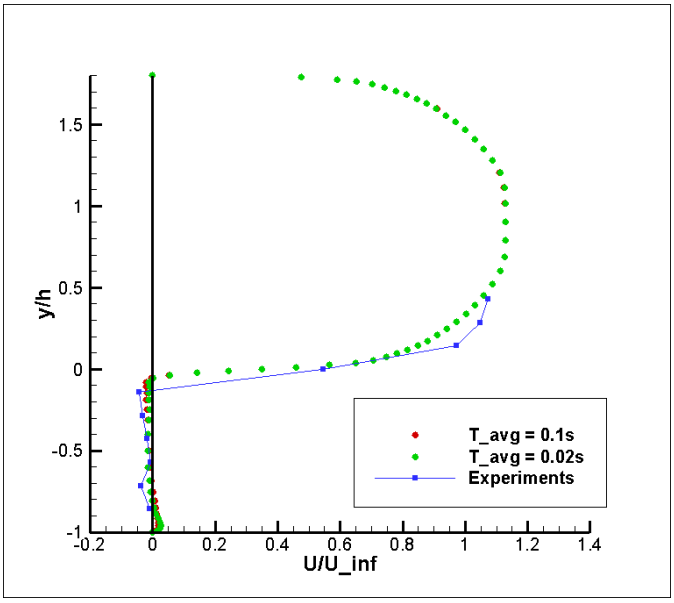


Figure 3.24: *Velocity profile at station* $\frac{x}{h} = -0.43$

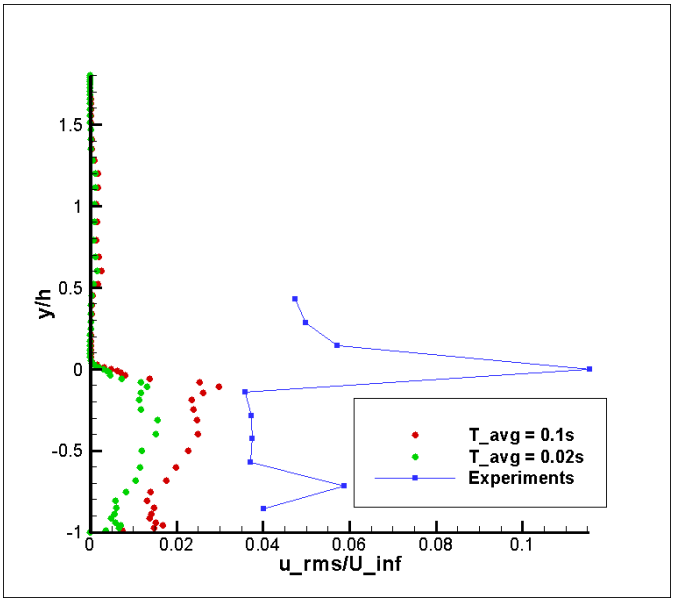
The velocity profiles at the next two stations are shown in figure 3.25.
Station number three after the step is given in figure 3.26.
And the final two stations are plotted in figure 3.27.
An impression of the velocity is given in figure 3.28a and of the pressure in figure 3.28b. These are the distribution of the velocity and pressure at the front of the domain. From the bubble shape in figure 3.29 it is clearly visible that the average velocity in x-direction varies with the span wise coordinate z.
The pressure at the beginning of the domain is $-171Pa$ to $-173Pa$ and at the end of the domain it is zero so the pressure increment is around $172Pa$. The length of the bubble was estimated to be around $13h$.

Conclusion

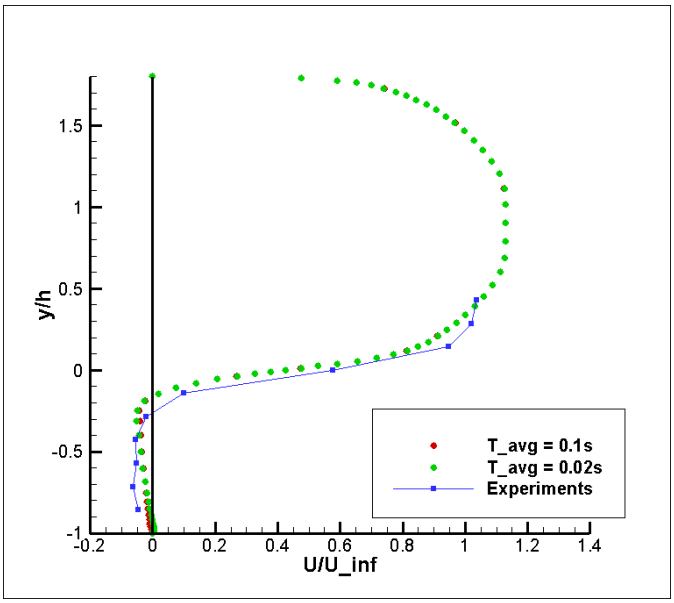
The velocity profiles in figures 3.24 and 3.25 seem quite accurate. It can be seen that before the step, the velocity is quite good in comparison to the experiments. The boundary layer is a little thicker and causes a small overshoot in the flow velocity in the center of the channel. The flow just after the step has a same overshoot in the mean flow velocity but it seems quite a good approximation. The flow in the bottom part of the channel contains a lot of fluctuations



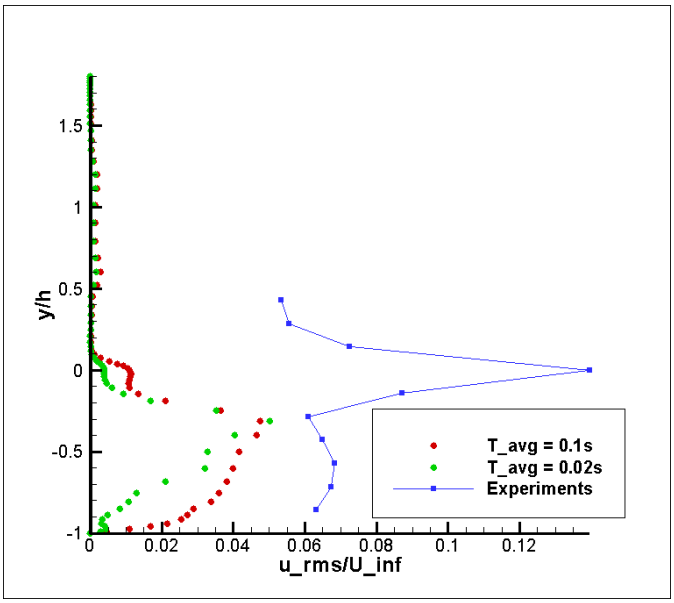
(a) *Velocity profile at station* $\frac{x}{h} = 0.29$



(b) *Fluctuations profile at station* $\frac{x}{h} = 0.29$



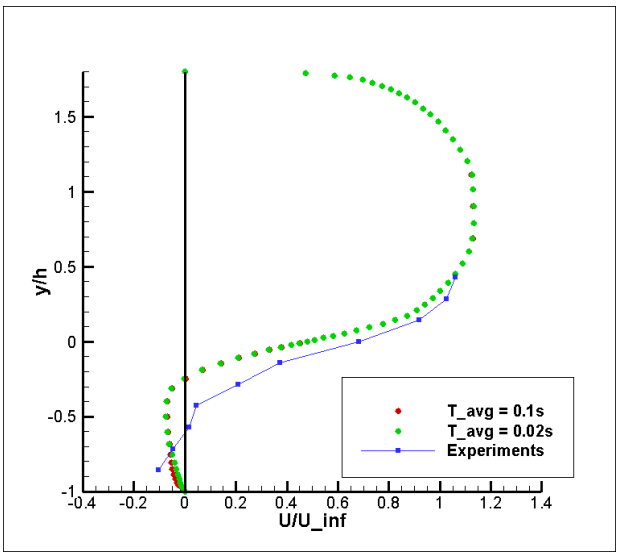
(c) *Velocity profile at station* $\frac{x}{h} = 1.1$



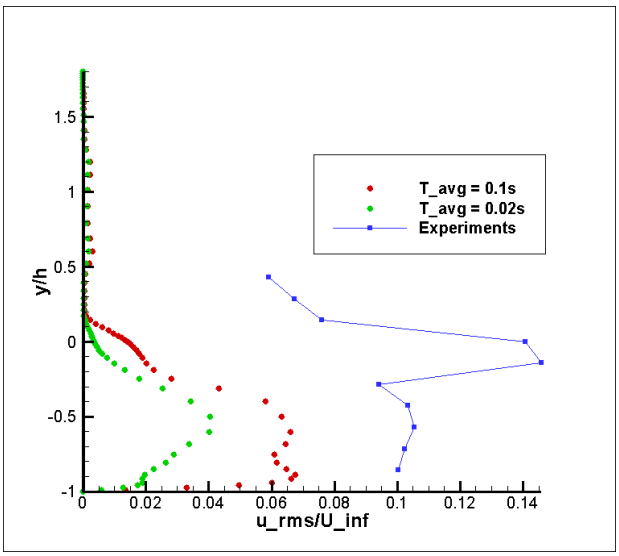
(d) *Fluctuations profile at station* $\frac{x}{h} = 1.1$

Figure 3.25

although the predicted fluctuations are less than the ones obtained from the experiments. For the average velocity profiles of the simulations, there is not much difference, but for the fluctuating parts, a significant difference can be observed. From the velocity profiles in 3.26 it can be seen that the bubble in the OpenFOAM simulation is thicker than the one encountered in the experiments. The fluctuations at this station are also not as high as found in the experiments but seem to increase from station to station. This trend is continued in the next two stations (figure 3.27). The experimental data shows a decrease in the thickness of the boundary layer especially and no boundary layer in figure 3.27c. The bubble is, however, still present in the simulations. The fluctuations of the experiments in the station at $x/h = 6$ has decreased because there is less turbulence at that station. In the OpenFOAM simulations however, the bubble is still going on and the fluctuations are very high. The pressure prediction in the OpenFOAM simulations is almost half of the one found from the experiments. When the bubble length is compared, it turns out that the length is almost three times longer in the OpenFOAM simulations in comparison with the experiments. It can be concluded that the results from the simulations as done by the OpenFOAM simulations are not in comparison with the experiments done. The profiles found in the beginning of the channel and just after the step seem to be promising but development of the bubble is not correct. It can also be seen that there is a significant difference in time taken for the calculation of the fluctuations. The fluctuations seem to be higher than predicted by the smaller averaging time.

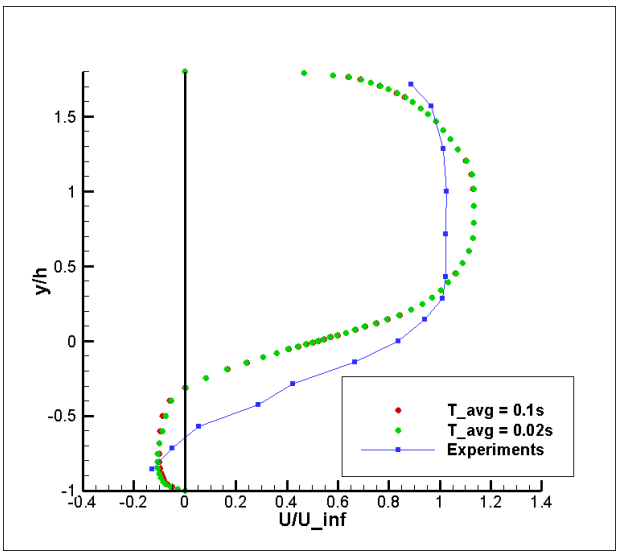


(a) *Velocity profile at station* $\frac{x}{h} = 2.3$

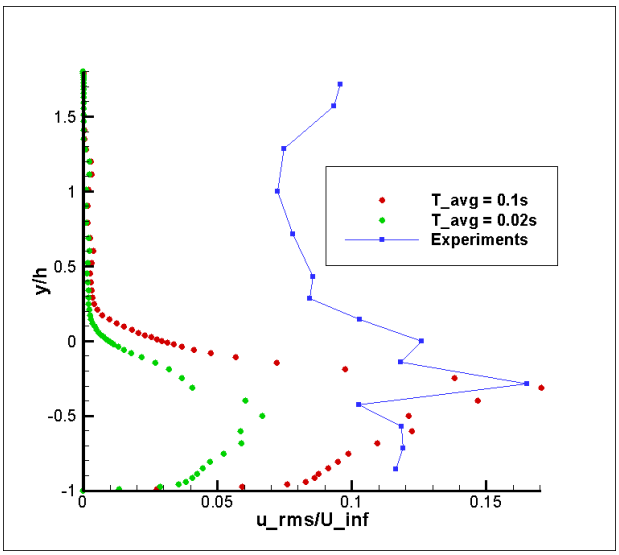


(b) *Fluctuations profile at station* $\frac{x}{h} = 2.3$

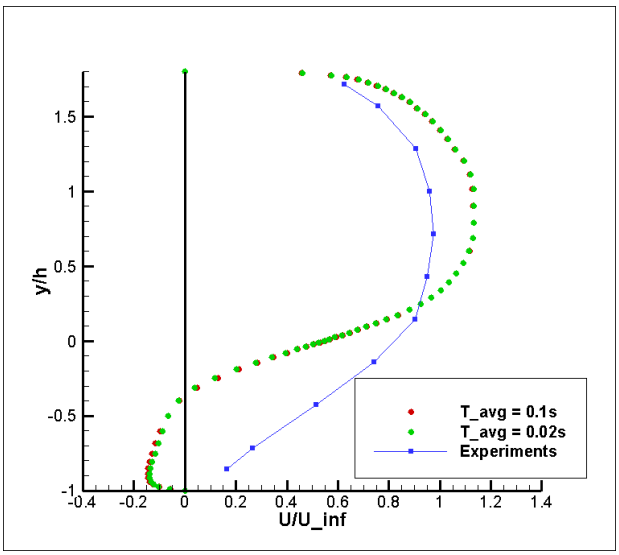
Figure 3.26



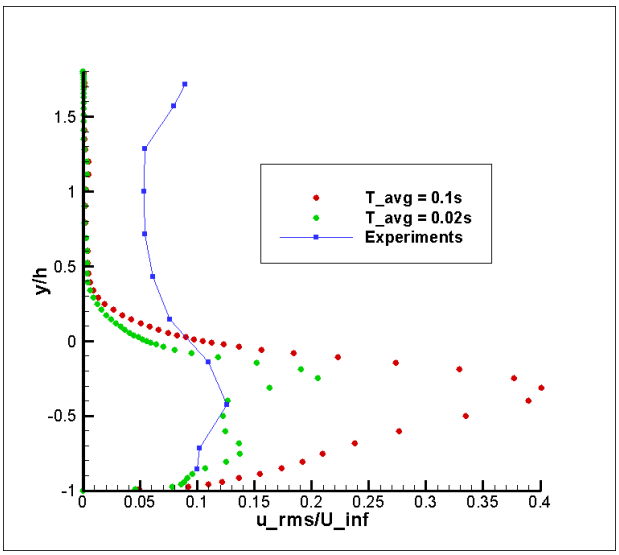
(a) *Velocity profile at station* $\frac{x}{h} = 4.3$



(b) *Fluctuations profile at station* $\frac{x}{h} = 4.3$

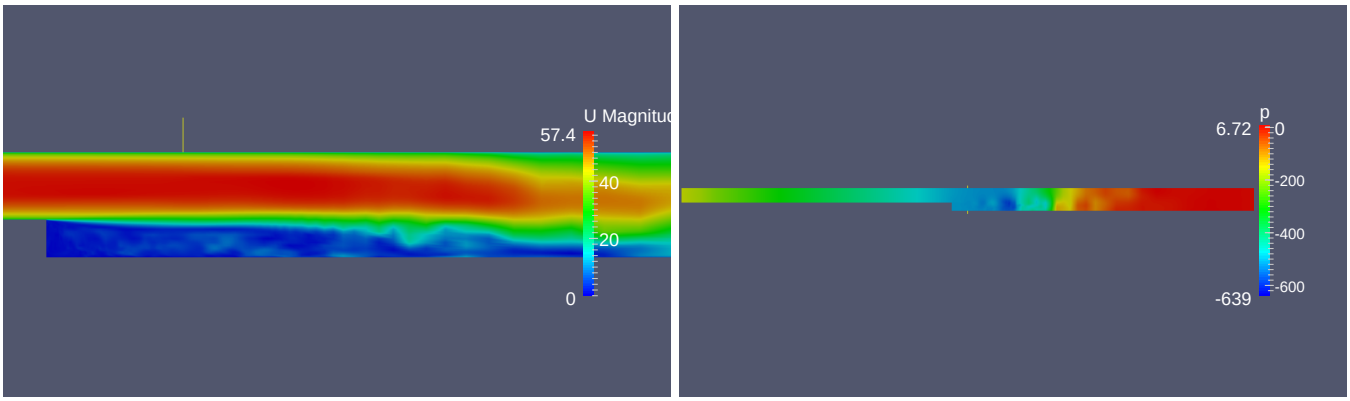


(c) *Velocity profile at station* $\frac{x}{h} = 6$



(d) *Fluctuations profile at station* $\frac{x}{h} = 6$

Figure 3.27



(a) *Magnitude of the velocity at* $t = 0.49998s$ (b) *Pressure at* $t = 0.49998s$

Figure 3.28

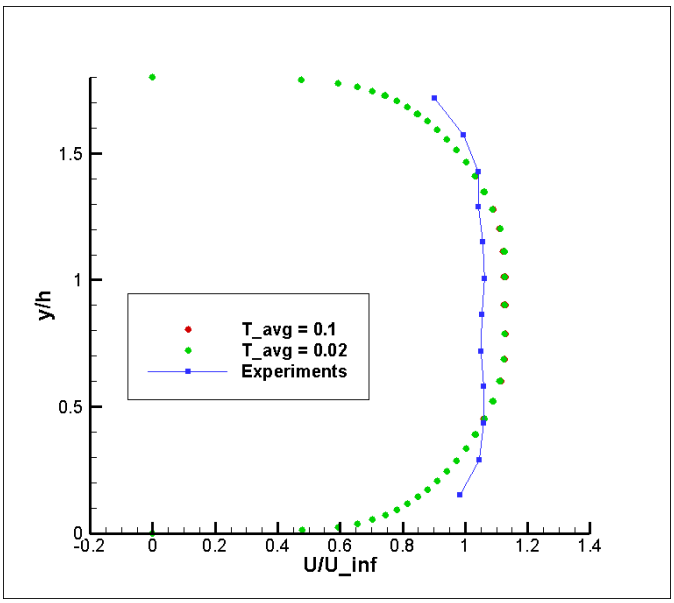


Figure 3.29: *Area where the mean velocity in* x*-direction is negative forming the bubble*

4 Conclusion and Recommendations

4.1 Circular Cylinder

The circular cylinder cases where used to get a feeling for the setup of cases as well as to get familiar with OpenFOAM. The results found were, except in the LES case, quite good in comparison to the reference papers. In the uRANS case, a little too much artificial dissipation reduced the sharp edges in the profiles. By applying a finer mesh in the wake region, this problem could be solved. Also the application of higher order schemes could improve the results. The LES case was a different story because the results seem totally off. The flow remains attached to the cylinder for too long because too little turbulence was created. It would be useful to try a different type of δ-function in order to improve the results because the van Driest damping function damps some of the turbulence. Also by setting the initial condition for the turbulent viscosity to a very small value could trigger the simulation to produce more turbulence and improve the results. It is also interesting to try some different LES methods in order to see if the result will change.

4.2 Backward facing step

The backward facing step was the test case to see the influence of different LES and DES models implemented in OpenFOAM. The original fine grid setup took an exceptionally long time to finish. This could be caused by a mistake in setup as well as with the fact that the simulation would take a long time anyway. Since the cells at the step are very small (to get enough resolution at the wall), the time step has to remain very small in order to keep the maximum Courant number very low.
To get some results for the backward facing step, the decision was made to change to a coarser grid. Two different types of simulations were performed on the coarse grid. The first one is the IDDES simulation while the second one was an one-equation-eddy LES model. This latter case was left out of the report, because the velocity profiles were very wrong. Although in both cases the initial conditions were the same as presented before in this report, the velocity profile before the step changed in the one equation eddy simulation. The boundary layers became very thick and made no sense at all. The reason for this could be that the coarse mesh was too coarse for this type of simulation. With a much finer mesh, the result could significantly improve. It should also be kept in mind that the application of wall models could also mean a significant improvement in the results. The IDDES case looked promising but it turned out that it also over predicted the length of the bubble. From figure 3.29 it is clearly visible how coarse the mesh is. The threshold used here is the mean velocity in x-direction and the region is shown where this velocity is negative. The bubble looks a little like a Lego creation which is far from reality. By applying a finer mesh, the result could become much better. Because of the coarse mesh, the simulated eddy size also increases (since the smallest solvable eddy size depends on the size of the grid). Because these eddies are approximated and not solved exactly, errors are introduced as well. Although the differences with the experiments are clearly visible, it can also be seen that the averaging time influences the fluctuations as well. The differences in stations further down stream were visible especially in the fluctuations. The averaging time chosen should be sufficiently large to 'guarantee' a stationary solution of the mean flow. Setting the average time to a much larger value is one way to make sure that the averages will be correct. Another way would be to make estimations of the frequency of the vortex shedding and choosing the time in such a way that at least a couple of vortexes have been shed. This will give a good solution without increasing the calculation too much.

4.3 Future research

For the future it would be useful to perform some more simulations on the backward facing step and the circular cylinder. Because these test cases are very intensively studied, a lot of experimental data is available. In this report, the final LES/DES simulations did not give good results with respect to the experimental data which was most likely caused by errors made in the setup. When the calculation is done again, the results could be improved significantly. New simulations would also give an ideal opportunity to apply more different LES models implemented in OpenFOAM. By using more different models, the knowledge about turbulence in general could be improved. OpenFOAM can be used for these test cases because it supports a lot of different models as well as solving methods. Also because there are constantly people working on the software, the program is kept up to date and the newest methods will be implemented.

A Appendix

Finite difference The first derivative can be found in two different ways with the help of the Taylor series expansion. The first way is writing the value of the next point with the Taylor series expansion by using the current value (u_i), the step size (Δx) and the derivative in the current point ($\frac{du}{dx}$) at the position i (h.o.t. are the higher order terms which are not written out here). Next, an expression for the first derivative can be found as is done in A.1.

$$\begin{aligned} u_{i+1} &= u_i + \Delta x \left.\frac{du}{dx}\right|_i + \frac{(\Delta x)^2}{2}\left.\frac{d^2u}{dx^2}\right|_i + \frac{(\Delta x)^3}{6}\left.\frac{d^3u}{dx^3}\right|_i + \frac{(\Delta x)^4}{24}\left.\frac{d^4u}{dx^4}\right|_i + h.o.t. \\ \left.\frac{du}{dx}\right|_i &= \frac{u_{i+1}-u_i}{\Delta x} - \frac{(\Delta x)}{2}\left.\frac{d^2u}{dx^2}\right|_i + h.o.t. = \frac{u_{i+1}-u_i}{\Delta x} + \mathcal{O}(\Delta x) \end{aligned} \tag{A.1}$$

This is called the forward differencing method and has an accuracy of order Δx. When the previous point is taken, the backward differencing scheme is found as in A.2 and, as can be seen, is also first order accurate.

$$\begin{aligned} u_{i-1} &= u_i - \Delta x \left.\frac{du}{dx}\right|_i + \frac{(\Delta x)^2}{2}\left.\frac{d^2u}{dx^2}\right|_i - \frac{(\Delta x)^3}{6}\left.\frac{d^3u}{dx^3}\right|_i + \frac{(\Delta x)^4}{24}\left.\frac{d^4u}{dx^4}\right|_i + h.o.t. \\ \left.\frac{du}{dx}\right|_i &= \frac{u_i-u_{i-1}}{\Delta x} + \frac{(\Delta x)}{2}\left.\frac{d^2u}{dx^2}\right|_i + h.o.t. = \frac{u_i-u_{i-1}}{\Delta x} + \mathcal{O}(\Delta x) \end{aligned} \tag{A.2}$$

If the first expressions of A.1 and A.2 are subtracted, the central differencing scheme is found as is shown in A.3.

$$\begin{aligned} u_{i+1} - u_{i-1} &= u_i - u_i + \Delta x \left.\frac{du}{dx}\right|_i - (-\Delta x \left.\frac{du}{dx}\right|_i) + \frac{(\Delta x)^2}{2}\left.\frac{d^2u}{dx^2}\right|_i - \frac{(\Delta x)^2}{2}\left.\frac{d^2u}{dx^2}\right|_i \\ &\quad + \frac{(\Delta x)^3}{6}\left.\frac{d^3u}{dx^3}\right|_i - (-\frac{(\Delta x)^3}{6}\left.\frac{d^3u}{dx^3}\right|_i) + \frac{(\Delta x)^4}{24}\left.\frac{d^4u}{dx^4}\right|_i - \frac{(\Delta x)^4}{24}\left.\frac{d^4u}{dx^4}\right|_i + h.o.t. \\ &= 2\Delta x \left.\frac{du}{dx}\right|_i + 2\frac{(\Delta x)^3}{6}\left.\frac{d^3u}{dx^3}\right|_i + h.o.t. \\ \left.\frac{du}{dx}\right|_i &= \frac{u_{i+1}-u_{i-1}}{2\Delta x} + \frac{(\Delta x)^2}{6}\left.\frac{d^3u}{dx^3}\right|_i + h.o.t = \frac{u_{i+1}-u_{i-1}}{2\Delta x} + \mathcal{O}\left((\Delta x)^2\right) \end{aligned} \tag{A.3}$$

It can be seen that the order of accuracy in the central differencing scheme is higher than that of the forward or backward differencing schemes.
When the first expressions in A.1 and A.2 are added, an expression for the second derivative can easily be found. For the forward and backward differencing it is also possible to express the second derivative by using an additional point respectively u_{i+2} and u_{i-2}. The resulting expressions are given in A.4.

$$\begin{aligned} \textbf{Backward } \left.\frac{d^2u}{dx^2}\right|_i &= \frac{u_i - 2u_{i-1} + u_{i-2}}{(\Delta x)^2} + \mathcal{O}(\Delta x) \\ \textbf{Central } \left.\frac{d^2u}{dx^2}\right|_i &= \frac{u_{i-1} - 2u_i + u_{i-2}}{(\Delta x)^2} + \mathcal{O}\left((\Delta x)^2\right) \\ \textbf{Forward } \left.\frac{d^2u}{dx^2}\right|_i &= \frac{u_i - 2u_{i+1} + u_{i+2}}{(\Delta x)^2} + \mathcal{O}(\Delta x) \end{aligned} \tag{A.4}$$

B Appendix

The blockMesh utility in OpenFOAM can be used to define the mesh by defining the number of cells in a direction as well as the grading. This utility creates a mesh. The idea is that 8 points are defined which make up a block. This block is then divided in the different cells. In order to determine the distribution of the cells, a number of parameters should be provided. These parameters are the number of cells (N) and the grading (r). The grading (r) is defined as the length of the last cell in the block divided by the length of the first cell of the block. The cell-to-cell expansion ratio is defined in such a way that it is kept constant from cell to cell (k).
With the grading and the number of cells as degrees of freedom, the mesh can be created in such a way that the cell width at the walls is small enough and that the cell size from block to block is very similar to prevent large jumps in cell size. In order to get the right size for the cells at the beginning and end of the block, the correct parameters should be chosen.

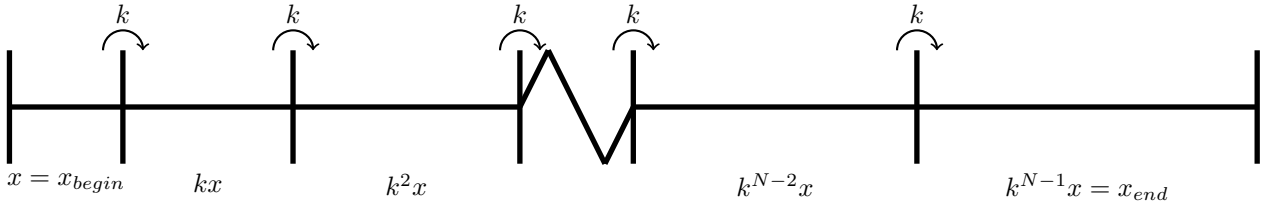


Figure B.1: BlockMesh utility

The division of a line with the method of the BlockMesh utility is given in figure B.1. The length of the line is defined as the length (L) of an arbitrary side of the defined block. The local grading is equal to:

$$k = r^{\frac{1}{N-1}} \tag{B.1}$$

It can then be found that the following is true:

$$L = \sum_{1}^{N} k^{N-1}x = \frac{k^N - 1}{k - 1}x \tag{B.2}$$

It can be found that the first cell has a width of $\frac{k^N-1}{k-1}/L$ or $\frac{k-1}{k^N-1}L$. The last cell has the size of rx_{begin}. In summary this means that the begin and end length can be calculated as function of the grading and the number of cells according to:

$$\begin{aligned} x_{begin} &= \frac{k-1}{k^N-1}L = \frac{r^{\frac{1}{N-1}}-1}{r^{\frac{N}{N-1}}-1}L \\ x_{end} &= rx_{begin} = r\frac{r^{\frac{1}{N-1}}-1}{r^{\frac{N}{N-1}}-1}L \end{aligned} \tag{B.3}$$

Bibliography

[1] Fluid Dynamics Lecture notes Part 1, Prof.dr.ir. H.W.M. Hoeijmakers, September 2012

[2] Turbulent Flows, Stephen B. Pope, Cambridge, 2000

[3] Direct Numerical Simulation: A Tool in Turbulence Research, Parviz Moin and Krishnan Mahesh, Annu. Rev. Fluid Mech. , 1998

[4] Introductory Lectures on Turbulence, J.M. McDonough , Departments of Mechanical Engineering and Mathematics, University of Kentucky, 2007

[5] http://www.cfd-online.com/Wiki/Prandtl's_one-equation_model

[6] The Potential of Large Eddy Simulation for the Modeling of Wall Bounded Flows, Eugene de Villiers, 25th July 2006

[7] M.Germano, U. Piomelli, P. Moin and W.H. Cabot. A dynamic subgrid-scale eddy viscosity model, Phys. Fluids A 3, 1760-1765, 1991

[8] Book, Chapter 5 Flow Around A Circular Cylinder irs.ub.rug.nl/dbi/45a650c293eb9

[9] Thesis Villiers picture page 62, fig. 3.2

[10] Numerical Heat Transfer and Fluid Flow, Suhas V. Patankar, McGRAW-HILL BOOK COMPANY, 1980

[11] Shur M., Spalart P.R., Strelets M. and Travin A. Detached eddy simulation of an airfoil at high angle of attack. In Fourth International Symposium on Engineering Turbulence Modelling and Experiment, Corsica, W. Rodi and D. Laurence, editors, Elsevier, May, 1999.

[12] Von Karman Institute for Fluid Dynamics, Lecture series 2006 Large Eddy Simulation and Related Techniques Theory and Applications, March 13-16, 2006

[13] Numerical simulation of the flow around a circular cylinder at high Reynolds numbers, Pietro Catalano, Meng Wang, Gianluca Iaccarino, Parviz Moin, International Journal of Heat and Fluid Flow 24 (2003) 463-469

[14] http://www.computationalfluiddynamics.com.au/wp-content/uploads/2012/01/inflate5a2.jpg

[15] http://www.openfoam.com/

[16] OpenFOAM Programmers guide foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf

[17] http://www.math-linux.com/mathematics/Linear-Systems/Preconditioned-Conjugate-Gradient

[18] http://www.cfd-online.com/Wiki/File:Non_orthogonal_CV_terminology.jpg

[19] http://en.wikipedia.org/wiki/File:Skewnessquad.PNG

[20] http://www.weatherquestions.com/What_causes_turbulence.htm

[21] http://www.earthzine.org/2010/03/11/easy-access-to-satellite-weather-data/

[22] On the use of DES type methods for reactive flows, B. Sainte-Rose, N. Bertier, S. Deck, F. Dupoirieux