Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Marine and Offshore Technology	Spring 2019 Open
Author: Seyedeh Mona Nakhostin	……Mona Nakhostin…… (signature of author)
Programme coordinator: Muk Chen Ong Supervisor(s): Knut Erik Teigen Giljarhus	
Title of master's thesis: Investigation of transitional turbulence models to predict drag crisis for flow over spheres and cylinder	
Credits: 30	
Keywords: CFD, OpenFOAM, Turbulence model, Transition model, Drag coefficient, Drag crisis, Wind tunnel	Number of pages: 90 Stavanger, 14th of June 2019

Title page for Master's Thesis

Faculty of Science and

Universitetet
i Stavanger

Investigation of transitional turbulence models to predict drag crisis for flow over spheres and cylinder

Seyedeh Mona Nakhostin

Supervisor:

Knut Erik Teigen Giljarhus

Faculty of Science and Technology

Department of Mechanical and Structural Engineering and Materials Science

This page was intentionally left blank

Preface

This thesis was written as a part of a master degree of Marine and Offshore technology at the University of Stavanger. The work has been carried out in the spring semester 2019.

I would like to express my deepest appreciation to my supervisor Associate professor Knut Erik Teigen Giljarhus for all his guidance and support, valuable advice and practical suggestions, patience that cannot be underestimated and encouraging me all the way during this work.

I am also grateful to Ola Elfmark and Lars Morton Bardal at Norwegian University of Science and Technology in Trondheim (NTNU) for their valuable contribution in the experiment in the wind tunnel that has been done in this work.

I would like to extend my sincere thanks to my mother Giti Mojahed and my uncle Behzad Mojahed who provided me with encouragement and patience throughout the duration of my thesis.

In the end thanks should also go to my second family, Dunia, Pulkit, Clarissa, David, Julie-Ann and Brendan for being great friends and all their help and support.

June 2019, Stavanger, Norway

Seyedeh Mona Nakhostin

Abstract

The current work performed experimental and numerical studies of flow over cylinders and spheres. The purpose is to investigate whether the drag crisis phenomenon can be predicted, and to gain insight into the transition from laminar to turbulent flow in the boundary layer.

An experiment was performed in a wind tunnel for flow over the sphere. The experiment has been performed on a sphere with smooth surface, and also with a trip wire at various locations on the surface. The results have shown that the placement of the trip wire on the surface changes the drag behavior significantly. Unfortunately the rod which was used in order to hold the sphere in the wind tunnel caused some unwanted changes in the wake area. Therefore, most of the simulations have been done on the sphere and cylinder in the free stream, and the results compared well to other experiments from the scientific literature.

The simulations have been done for several Reynolds numbers covering the laminar and transitional flow regimes in the boundary layer. The turbulence models used were RANS with k-ω SST and DES with k-ω SST as RANS model. For transition modelling, the Langtry-Menter model was used. Simulations for the cylinder have been performed for three Reynolds numbers, 3900, 10 000 and 50 000 in both models. For the sphere, seven cases has been defined with different Reynolds numbers, 10^4, 4×10^4, 10^5, 2×10^5, 2.5×10^5, 5×10^5 and 10^6, and were simulated with both models.

For the cylinder, the computed drag coefficients were in a good agreement with experimental results at the lower Reynolds number. At the higher Reynolds numbers in the transition region, the computed results with transitional model were closer to experimental results compared to the fully turbulent model.

In the sphere case, there was also good agreement with experimental results at subcritical Reynolds numbers. Also, at supercritical Reynolds numbers, the fully turbulent model performs well. In the transition region, the transition model gives higher drag than the fully turbulent, and

the results are closer to the experimental results. However, some numerical instabilities were also observed at the highest Reynolds numbers.

Contents

Figures

Tables

List of Symbols

Latin Letters

Symbol	Description	Unit
A	Frontal area of the body	m^2
C_D	Drag Coefficient	-
C_f	Friction variation	-
C_μ	Constant of equation (2-7)	-
e_{ra}	Expansion ratio	-
F_1	Blending function given by Equation (2-24)	-
F_2	Blending function given by Equation (2-31)	-
F_D	Drag Force	N
K	Turbulence kinetic energy	$Kg\ m^2s^{-2}$
L_t	Turbulent length	m
N	Direction normal to the boundary	-
N_{ce}	Number of cell layers	-
P	Mean pressure	Pa
p	Pressure	Pa
p'	Fluctuating pressure	Pa
Re	Reynolds number	-
$Re_{\theta t}$	Transition momentum thickness Reynolds number	-
s_{ij}	Strain rate tensor	-
T_i	Turbulence intensity	-
t	Time	s
U_∞	Free-stream velocity	m/s
U	Mean velocity in the x-direction	m/s
u	Velocity	m/s
u'	Fluctuating velocity	m/s
u^+	Dimensionless velocity	-
u_τ	Frictional velocity	m/s
x	x-direction	m
y_{cl}	The height of the cells in the layer furthest from the wall(in the viscous mesh)	m
y_{dc}	The height of the first cell from the wall	m
y_d	Distance normal from the wall	m

List of symbols (continuing)

Greek letters

Symbols	Description	Unit
β	Constant of equation (2-24)	-
β_1	Constant given by Equation (2-12), it is equal to 0.075	-
β^*	Constant given by equation (2-28)	-
γ	intermittency	-
δ_{ij}	Kronecker delta	-
k	Von Karmans constant, equal to 0.41	-
μ	Dynamic viscosity	Pa.s
μ_t	Eddy viscosity	Pa.s
ν	Kinematic viscosity	m^2/s
ν_T	Turbulent viscosity	m^2/s
ρ	Density	Kg/m^3
τ_ω	Wall shear stress	N/m^2
ω	Rate of dissipation of *k*	s^{-1}

Abbreviations

CFD	Computational fluid dynamics
DES	Detached Eddy Simulation
LES	Large Eddy Simulation
LM	Langtry-Menter
NTNU	Norwegian University of Science and Technology
RANS	Reynolds-averaging Navier-Stokes
SST	Shear Stress Transport
UiS	University of Stavanger
URANS	Unsteady Reynolds-Averaged Navier-Stokes

1 Introduction

The resistance caused by fluid flow over bodies, both gas and liquid, is important in many different industries, such as automotive industries, aerospace industries and marine industries. Although real structures often have complex geometry, the study of simpler geometries can lead to greater insight into physical phenomena. They are also directly relevant in some cases, such as flow over cylinder representing flow around pipelines or wind turbine towers. For this study, flow over spheres and cylinders have been chosen as the main cases. Experiments show a sudden and dramatic reduction in drag force at higher Reynolds numbers for these geometries, which is known as the drag crisis. This phenomenon is caused by transition from laminar flow regime to turbulent in the boundary layer. The turbulent boundary layer is able to remain attached to the body longer which leads to a smaller wake and thereby a reduction in drag force.

Various studies have been done for flow over cylinder and sphere in which the simulations have been done at either subcritical or supercritical Reynolds numbers each using different methods. Mittal and Moin [1], Breuer, case D3 [2], Franke and Frank [3], Alkishriwi et al. [4] and Meyer et al. [5] performed simulations of flow over cylinder using LES (Large eddy simulations) in the subcritical area with Re=3900 using structural grid type with finite volume method as their solution method for the filtered Navier-Stokes equations. Kravchenko and Moin [6], Parnaudeau et al. [7] and Mani et al. [8] used LES at Re=3900 using structured grid type and [7] high order finite difference method.

P. Catalano et al. [9] did simulations for Re = 1×10^6 with structural mesh first with LES and then with steady and unsteady Reynolds-averaged Navier –Stockes solutions (RANS and URANS).

Most of the studies over flow around sphere has been done through lab results and only few studies based on the numerical simulations. As an example, Constantinescu and Squires have done their simulation for Reynolds number in the range of 10^4 and 10^6 using DES technique. The results were in the good agreement with the experimental results. [10]

There are other studies in various fields which are dedicated to the behavior of drag force in different situations. Dean and Bhushan [11] studied the effect of shark-skin riblets in order to maximize the drag reduction and the theories have been discussed from experiment and simulation data. Moreover, there is another study over mechanism of drag reduction by dimples on a sphere such as golf-ball dimples which shows that the dimples delay the main separation and reduce drag and generation of separation bubble is a good way to control the flow for drag reduction on a bluff body like sphere or cylinder. [12]

Park and Lee presented a wind tunnel experiment and large-eddy simulation in order to examine the drag reduction performance in flow over 2 dimensional bluff body with a blunt trailing. [13]

Daniello, Waterhouse and Rothstein presented their study about flow over superhydrophhobic surfaces. They concluded that superhydrophobic surfaces provide a considerable drag reduction for marine vessels. [14]

As described, most of the previous studies focused on either the subcritical or supercritical area, while little work has been done in the transition region. Therefore, this work investigates the use of transitional turbulence models for prediction of drag crisis. Many transitional flow models are available in the literature. One of the most practical and generic is the Langry-Menter model [15] which is based on empirical correlations to predict the transition. These models have been primarily studied in relation to airfoil geometries, such as flow over airfoils, hydroturbines or wind turbines, and not for bluff bodies such as spheres or cylinders. This study will investigate the performance of the Langtry-Menter model for predicting the drag crisis for flow over spheres and cylinders, using both RANS and DES formulations.

1.1 Outline of the work

Chapter 2 introduces the theory and concepts behind all of the simulations and the methods that have been used in this work. Chapter 3 explains the experiment that has been done on the flow over sphere in the Norwegian University of Science and Technology in Trondheim (NTNU). This

experimental study covers the drag force measurement on the flow over smooth sphere and the sphere with an attached trip wire to study the effect of the transition location. In Chapter 4, flow over a flat plate is performed to investigate the transition behavior for a simpler test case. Chapters 5 and 6 gives the simulation results for flow over a sphere with RANS and DES, respectively. Chapter 7 presents the results for flow over a cylinder. Finally, Chapter 8 gives conclusions and recommendations for further work.

2 Theory

The Reynolds number is the ratio of inertial force to viscous force, defined as

$$Re = \frac{\rho u D}{\mu} = \frac{u D}{\nu} \qquad (2\text{-}1)$$

Here, ρ is the density, μ is the dynamic viscosity, u is the velocity, D is the diameter and ν is the kinematic viscosity of the fluid. The Reynolds number is used to classify the flow regime and to predict the behavior of the fluid.

The force that a flowing fluid exerts on a body in the flow direction is called drag. Most of the times, it is an undesirable effect and we try to minimize it. A moving fluid exerts normal pressure force and tangential shear forces on the surface as drag forces. Tangential shear forces are due to no-slip condition caused by viscous effect. Both of these forces have component in the direction of flow, hence the drag force is due to combined effects of pressure and wall shear forces in the flow direction. To study the effect of drag force we consider the effect of the dimensionless parameter known as drag coefficient,

$$C_D = \frac{2F_D}{\rho U^2 A} \qquad (2\text{-}2)$$

Here, F_D is the total drag force, ρ is the density U is the velocity and A is the projected area facing the direction of the flow.

When it comes to the external flow over a bluff body there are two regions that matter for flow study: the region around the front, which is called stagnation region, and a low pressure wake region at the back of the body that is caused due to separation. It happens when the fluid

separates from the body and it forms a separated region between the body and the fluid stream. This region where the effects of the body on velocity are felt is called the wake. Figure 1 and 2 show the wake area behind the cylinder. [16]

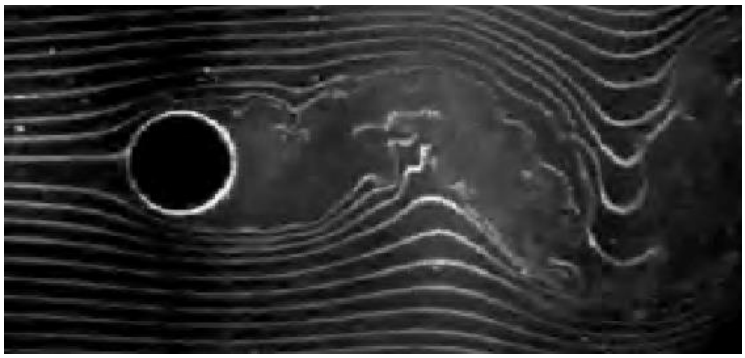


Figure 1 Wake at the back of a cylinder

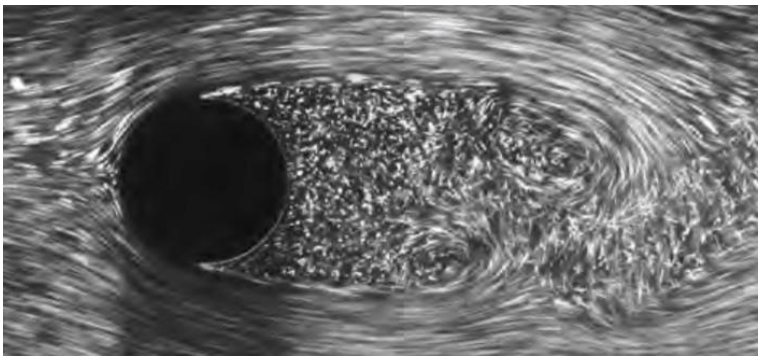


Figure 2 Flow over a circular cylinder at Re = 2000; Laminar boundary layer separation with a turbulent wake

The separated region is an enclosed volume, which comes to an end when the two flow stream reattach. While the wake only stops growing when the fluid regains its velocity back to its normal profile. The size of the separation region depends on Reynolds number, roughness and turbulence intensity. The larger the separated region, the larger the pressure drag. There is a chance to lower the drag force by reducing the separation region. More than 90% of the total drag of bluff bodies is due to the wake region. [17] [18]

Fluid is predicted to be laminar at low Reynolds numbers. Hence, the layers of the fluid is meant to slide smoothly past each other due to high viscosity force. As the Reynolds numbers increase, the flow behavior is rapidly fluctuating in time and space and the flow change to be turbulent. Transition flow have combination of laminar and turbulent aspects.

At some critical Reynolds number the shear layer separating from upper and lower surface of sphere starts to be unstable and boundary layer faces a transition from laminar to turbulent. This transition leads to a delay of the separation of flow from sphere surface causing a substantial reduction in the drag force that the sphere experiences. This is often referred to as the drag crisis and due to its effect on drag reduction, it is the matter of our interest.

Surface roughness also has a significant effect on the drag. Surface roughness in general increases the viscous drag. However, the roughness also promotes transition to turbulence which can reduce the pressure drag due to delayed separation.

2.1 Computational fluids dynamics (CFD)

CFD is a methodology to solve fluid flow problems numerically. The open-source software OpenFOAM has been used as the CFD tool in the current thesis. Analysis with CFD follows different steps. First of all, the geometry has to be defined and meshed, fluid properties have to be inserted and boundary conditions should be set. In the next level, the desired method of solving the problem is applied and the arising equations solved. In the last steps, the results are visualized and evaluated with the help of post-processing tools such as Paraview.

2.2 Boundary Conditions

The drag force is the main topic in the current work, therefore the turbulence near the wall is our main interest. The fluid in contact with the wall as a solid object has zero relative velocity which is called no slip-condition. As per definition of the Reynolds number, having zero velocity leads

to zero Reynolds number which means the flow near the wall is more effected by the viscous force therefore the mean velocity is the function of the ρ, μ, τ_w as the shear stress at the wall and y_d as the distance from wall. Mean velocity can be shown as,

$$U = f(\rho, \mu, \tau_w, y_d) \qquad (2\text{-}3)$$

In order to have the dimensionless distance from the wall, mean velocity will be divided by the frictional velocity, u_τ , to define the equation, known as the law of the wall, that has been created based on work on the turbulence near the wall by the Prandtl and von Karaman in 1931 [19],

$$u_\tau = \sqrt{\frac{\tau_\omega}{\rho}} \qquad (2\text{-}4)$$

Where τ_ω is the shear stress along the wall. This gives

$$u^+ = \frac{U}{u_\tau} = f\left(\frac{\rho u_\tau y_d}{\mu}\right) = f(y^+) \qquad (2\text{-}5)$$

Where u^+ is the dimensionless velocity, u_τ is the frictional velocity and y^+ is the dimensionless distance from the wall.

Having a viscous fine mesh near the walls are important in order to capture the effect of the law of the wall.

2.2.1 Inlet

At the inlet we use the following conditions for turbulence,

$$k = \frac{3}{2}(U_\infty T_i)^2 \qquad (2\text{-}6)$$

$$\omega = \frac{\rho k}{\mu}\left(\frac{\mu_t}{\mu_t}\right)^{-1} \qquad (2\text{-}7)$$

Where T_i is the turbulence intensity, C_μ is a constant equal to 0.09, and μ_t/μ is the eddy viscosity ration.

2.2.2 Viscous mesh parameters

To select the distance to the first cell we use expressions for a flat plate. We define y_{dc} as the distance to the first cell. From the law of the wall, we have

$$y = y_{dc} = \frac{y^+ \mu}{u_\tau \rho} \qquad (2\text{-}8)$$

Where y^+ is the dimensionless wall distance, which we want equal to one on the surface of the wall. Based on the definition of the u_τ, shear stress along the wall can be rewritten as,

$$\tau_\omega = \frac{C_f \rho U_\infty^2}{2} \qquad (2\text{-}9)$$

Where C_f is the friction variation and is calculated by equation (2-10), [20]

$$C_{f=\frac{0.027}{Re^{1/7}}} \qquad (2\text{-}10)$$

Therefore we have,

$$\tau_\omega = 0.0135\ \frac{\rho U_\infty^2}{Re^{1/7}} \qquad (2\text{-}11)$$

And,

$$u_\tau = \sqrt{\frac{0.0135 U_\infty^2}{Re^{1/7}}} \qquad (2\text{-}12)$$

In the end to calculate y_{dc} we have,

$$y_{dc} = \frac{1}{\sqrt{0.0135}} \frac{y^+ Re^{1/14} \mu}{U_\infty \rho} \qquad (2\text{-}13)$$

There are other parameters that matter to us in order to have a good viscous mesh. Expansion ratio, e_{ra}, defines how much each cell grow and usually is set to 1.2. Cells closer to the wall has lower height compared to cells further from the wall. Number of cell layers in viscous mesh, n_{ce}, can be calculated by inserting the fixed value for the mentioned parameters. The value of n_{ce} helps to have the same cell height in the last cell in viscous mesh with first cell in the background mesh. This value can be computed by,

$$n_{ce} = \frac{\ln y_{cl} - \ln y_{dc}}{\ln e_{ra}} \qquad (2\text{-}14)$$

Where y_{cl} is the height of the last cell in viscous mesh which is equal to the height of the first cell in the background mesh. In order to calculate most of these values, online calculators can be useful.

2.3 kOmegaSST-RANS model

A DES model is a combination of RANS and LES models. In regions near surfaces and regions where the grid resolution is lower than the turbulent length scale, the RANS model is applied, while the LES model is applied elsewhere. This makes the model particularly suited to e.g. bluff body simulations since it accounts for the highly fluctuating nature of the flow while relaxing the grid requirements of a full LES simulation. Moreover, in the RANS simulations the velocity represents the average velocity, while for DES, it is the filtered velocity. The flow equations solved are the unsteady RANS equations,

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2-15}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j^2} - \frac{\partial \overline{\acute{u}_i \acute{u}_j}}{\partial x_j} \tag{2-16}$$

Here, *P* is the dynamic pressure and *ρ* is the density of the fluid. In LES regions, u represents the filtered instantaneous velocity, while in RANS regions, the velocity is the mean velocity and the influence of turbulence is accounted for by the Reynolds stress tensor, $\overline{\acute{u}_i \acute{u}_j}$. Here, u_i' denotes the fluctuating part of the velocity.

The Reynolds stress component is modelled using the Boussinesq approximation, where it is expressed in terms of a turbulent viscosity, ν_t, and the mean flow gradients,

$$-\overline{\acute{u}_i \acute{u}_j} = \nu_T \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3} k \delta_{ij} \tag{2-17}$$

Here, *k* is the turbulent kinetic energy and δ_{ij} is the kronecker delta function. [21]

For the RANS part, we are using the k-ω SST model. This model combines the best features of the k-ε model and the k-ω model. In the boundary layer, the model uses a k-ω model which means

that it can be used all the way to the wall without the use of wall functions. In the free stream, it avoids issues with the k-ω by switching to a k-ε formulation here. [22]

The model solves two additional transport equations, one for the turbulent kinetic energy,

$$\frac{\partial k}{\partial t} + \frac{\partial (u_i k)}{\partial x_i} = P_k - D_k + \frac{\partial}{\partial x_i}\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_i}\right] \quad (2\text{-}18)$$

And one for the turbulence specific dissipation rate, ω,

$$\frac{\partial \omega}{\partial t} + \frac{\partial (u_i \omega)}{\partial x_i} = P_\omega - \beta\omega^2 + \frac{\partial}{\partial x_i}\left[\left(\nu + \frac{\nu_T}{\sigma_\omega}\right)\frac{\partial \omega}{\partial x_j}\right] + 2(1 - F_1)\sigma_{\omega 2}\,\frac{1}{\omega}\,\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \quad (2\text{-}19)$$

The production term in the previous equation is

$$P_\omega = \alpha\, S^2 \quad (2\text{-}20)$$

Where S is the invariant measure of the strain rate, which is defined by

$$S = \sqrt{2\ \bar{s}_{ij}\ \bar{s}_{ij}} \quad (2\text{-}21)$$

And the strain rate tensor of filtered velocity is

$$\bar{s}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_i}\right) \quad (2\text{-}22)$$

Now we need a function to switch the SST model between the k-ω and k-ε formulation by changing between the value 1 close to wall to 0 in the outer part of the boundary layer as well as outside of it. This is represented by the last term of the ω-equation. The blending function is defined as

$$F_1 = \tanh\left(\left(min\left[max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\,\nu}{y^2\omega}\right), \frac{4\,\sigma_{\omega 2} k}{CD_{k\omega} y^2}\right]\right)^4\right) \qquad (2\text{-}23)$$

The term CD_ω is defined by equation (2-24),

$$CD_{k\omega} = \max\left(2\sigma_{\omega 2}\,\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-10}\right) \qquad (2\text{-}24)$$

The SST model also uses a shear stress limiter that will switch from the eddy viscosity model to the Johnson King model in regions where the shear stress becomes too large. It can be found in the following formula;

$$\nu_T = \frac{k a_1}{\max(w a_1, S F_2)} \qquad (2\text{-}25)$$

Where $a_1 = \sqrt{\beta^*}$ and F_2 is blending function that makes the Johnson King model be active only in the boundary layers. It is defined as;

$$F_2 = \tanh\left[\left[max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\,\nu}{y^2\omega}\right)\right]^2\right] \qquad (2\text{-}26)$$

Where the term y, stands for the distance to closest wall. The limiter from the production of turbulent kinetic energy is defined as

$$\tilde{P}_k = \min(P_k\,, 10\,\beta^* k\omega) \qquad (2\text{-}27)$$

$$P_k = \nu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \frac{\partial \bar{u}_i}{\partial x_j} \qquad (2\text{-}28)$$

To incorporate the DES features we must define the term F_{DES} in the dissipation term of the k-equation,

$$D_k = \beta^* k\omega\, F_{DES} \qquad (2\text{-}29)$$

$$F_{DES} = \max\left(\frac{L_t}{C_{DES}\,\Delta}\,, 1 \right) \qquad (2\text{-}30)$$

Where the turbulent length scale is

$$L_t = \frac{\sqrt{k}}{(\beta^*\,\omega)} \qquad (2\text{-}31)$$

And,

$$\Delta = \max(\Delta x_1, \Delta x_2, \Delta x_3) \qquad (2\text{-}32)$$

Which is the largest side of a cell at the present point of the grid and

$$C_{DES} = 0.61$$

The values for the remaining constants in the model are

$$\alpha_1 = \frac{5}{9}\,, \qquad \alpha_2 = 0.44, \qquad \beta_1 = \frac{3}{40}, \qquad \beta_2 = 0.0828$$

$$\beta^* = \frac{9}{100}\,,\ \sigma_{k1} = 0.85\,,\ \sigma_{k2} = 1\,,\ \sigma_{\omega 1} = 0.5\,,\ \sigma_{\omega 2} = 0.85$$

2.4 Langtry- Menter Model

For transition modelling, we use the Langtry-Menter model. Since the model contains a large number of correlations, only a brief overview of the model is given here, more details can be found in [15]. The model is also known as the γ-$Re_{\theta t}$ –SST model, because it makes use of two additional variables, the intermittency, γ, and the transition momentum thickness Reynolds number, $Re_{\theta t,}$ for the boundary layer in addition to the k and ω equations. The intermittency or γ is used to trigger the transition locally. It is coupled with the production of turbulent kinetic energy and used to turn on production downstream of the transition. The transition momentum thickness Reynolds number is used to capture the effect of the freestream behavior on the boundary layer. The two extra equations for $Re_{\theta t}$ and γ that are solved in this model are

$$\frac{\partial(\rho\gamma)}{\partial t} + \frac{\partial(\rho\gamma u_j)}{\partial x_j} = P_\gamma - E_\gamma + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_f}\right)\frac{\partial\gamma}{\partial x_j}\right] \qquad (2\text{-}33)$$

$$\frac{\partial(\rho Re_{\theta t})}{\partial t} + \frac{\partial(\rho u_j Re_{\theta t})}{\partial x_j} = P_{\theta t}(Re_{\theta t0} - Re_{\theta t}) + \frac{\partial}{\partial x_j}\left[\sigma_{\theta t}(\mu + \mu_t)\frac{\partial Re_{\theta t}}{\partial x_j}\right] \qquad (2\text{-}34)$$

The production term in the intermittency equation is designed to be switched whenever the local strain rate exceeds the transition criteria. It contains multiple correlations that are tuned based on experiments performed of flow over a flat plate. The sink term ensures that the intermittency is close to zero in the laminar boundary layer and can also be used to model relaminarization.

The equation for $Re_{\theta t}$ is used to transfer information from the free-stream to the boundary layer to give a local criterion for transition. The production term in this equation contains a correlation for the transition onset based on experiments.

The only modification to the SST model is that the production and dissipation terms in the equation for turbulent kinetic energy now include the intermittency to control the production/dissipation as follows,

$$\overline{P_k} = \gamma P_k \tag{2-35}$$

$$\overline{D_k} = \min(\max(\gamma, 0.1), 1.0)\, D_k \tag{2-36}$$

2.5 Implementation of DES version of the Langtry-Menter model in OpenFOAM

The standard version of OpenFOAM does not include a DES version of the Langtry-Menter model. However, it is implemented as a RANS model for the k-ω SST model, called kOmegaSSTLM. The standard k-ω SST model is also included in a DES version, called kOmegaSSTDES. To implement the LM model in a DES version, a new solver was created, called kOmegaSSTLMDES, where the modifications from the k −ω SST LM solver were included in the k −ω SST DES model. The full source code for the model is included in the appendix.

Although the source code contains many lines, a lot of this is just boilerplate code to make it fit into the OpenFOAM framework. The key parts of the code are relatively easy to understand and is closely related to the actual equations due to the equation-mimicking interface of OpenFOAM. For instance, the equation for the transition momentum thickness Reynolds number,

$$\frac{\partial(\rho Re_{\theta t})}{\partial t} + \frac{\partial(\rho u_j Re_{\theta t})}{\partial x_j} = P_{\theta t}(Re_{\theta t0} - Re_{\theta t}) + \frac{\partial}{\partial x_j}\left[\sigma_{\theta t}(\mu + \mu_t)\frac{\partial Re_{\theta t}}{\partial x_j}\right]$$

Is implemented in OpenFOAM as

```
// Transition onset momentum-thickness Reynolds number equation
tmp<fvScalarMatrix> ReThetatEqn
(
    fvm::ddt(alpha, rho, ReThetat_)
```

```
      + fvm::div(alphaRhoPhi, ReThetat_)
      - fvm::laplacian(alpha*rho*DReThetatEff(), ReThetat_)
     ==
        Pthetat*ReThetat0(Us, dUsds, nu) - fvm::Sp(Pthetat, ReThetat_)
      + fvOptions(alpha, rho, ReThetat_)
    );
```

We see that there is a close correlation to the original equation. The functions prepended by `fvm::` are functions which return a matrix with the finite volume operator applied to it using the schemes defined by the user. Additionally, the `alpha()` term is included to make the model also applicable to multiphase flow models, and the `fvOptions` function gives support for various source terms, such as porous regions or user-defined functions.

Interestingly, the free-stream transition momentum thickness Reynolds number, $Re_{\theta t0}$, is based on a correlation that is purposely not provided in the original paper. This is because the authors of the model are employed by ANSYS and keep some details proprietary. The implementation used here is based on reverse engineering done in [23].

3 Experiment over sphere

This chapter covers the experiment over sphere with and without surface modification which has been done in the wind tunnel facilities in Norwegian University of Science and Technology (NTNU) in Trondheim.

The purpose of the experiment was to obtain a greater understanding of the drag crisis behavior. First, we wanted to see if we could reproduce the drag crisis for a smooth sphere. Next, we introduce a small trip wire to force the flow in the boundary layer to transition from laminar to turbulent flow. By doing this at different positions on the sphere, we can understand better how the drag coefficient is influenced by the transition behavior. Finally, we wanted to see if we could gain more insight into the transition behavior by using oil surface visualization.

3.1 Introduction to wind tunnel

First of all we take a look at how a wind tunnel works. Wind tunnels are hollow tubes which have powerful fans to create a flow of air inside the tunnel. There are sensors and instruments inside the wind tunnel that gives hard data regarding an object's interaction with wind, and with the help of a window, we can observe experiment visually.

A wind tunnel has five basic sections: the settling chamber, contraction cone, test section, diffuser and drive section. Settling chamber help to settle and straighten the air through the use of panels with honeycomb-shaped holes or a mesh screen. Contraction cone help to increase the airflow velocity. Test section is where sensors record the data and visual observation of the model is possible. Diffuser smoothly slows the air's velocity without causing turbulence in the test section. There is an axial fan in the drive section that creates high speed airflow. This fan is at the end of the tunnel in order to allow the fan to pull air into smooth stream instead of pushing it. The walls of the tunnel are smooth because any imperfections could act as speed bumps and cause turbulence.

There are two kind of wind tunnels. Open-circuit (open return) tunnels which are just long and straight boxes. Or closed-circuits (closed return), which are ovals that send the air around the same path using vanes and honeycomb panels to precisely guide and direct the flow. The wind tunnel in which we have done our experiment is the closed-circuit wind tunnel. The dimension of the wind tunnel is 1 meter length to 0.5 meter width.

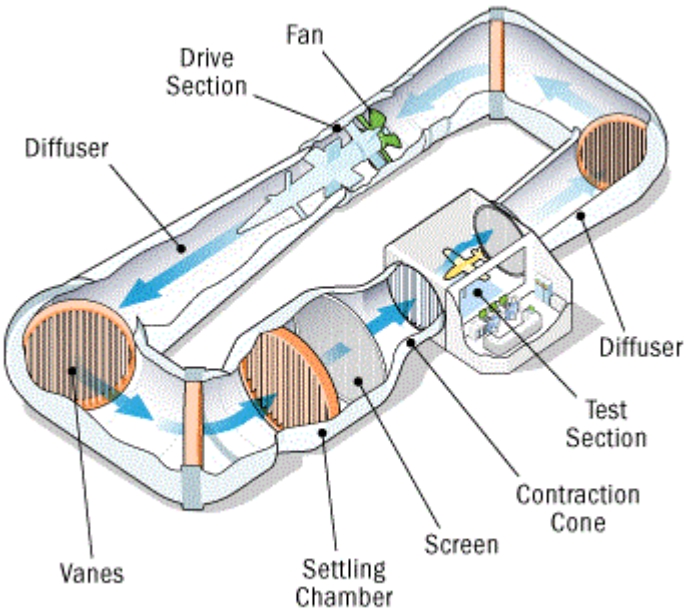


Figure 3 Closed circuit wind tunnel illustration

3.2 Experimental setup

The tested sphere has a diameter of 170mm. The sphere has been placed on a rod with 19mm diameter, and 130mm distance from sphere to the floor of the wind tunnel. The setup is shown in Figure 4. The rod is connected to a force balance to calculate the forces on the geometry.

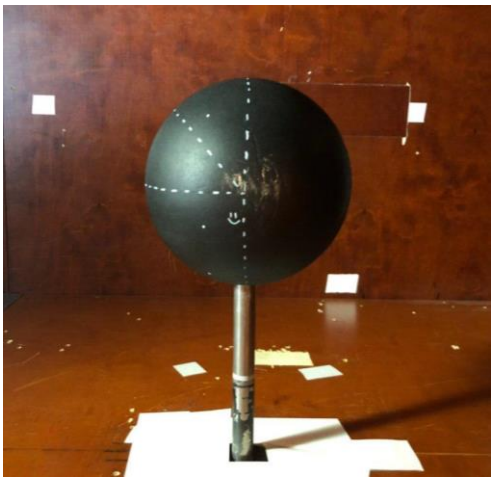


Figure 4 Sphere with 170mm diameter with normal surface

The first step was to measure the drag coefficient and hence the drag force with different wind speeds. The test has been done in different levels. First level was to test the sphere without any modification on its surface to have an indication to compare for other levels.

In order to modify the surface of the sphere, a wire has been attached on the sphere to act as a trigger for transition on the surface. The wire has small diameter and was made with the shape of circle and attached in different angles. The defined cases for the wire angle are 30, 70 and 90 degrees, which are shown in Figure 5.

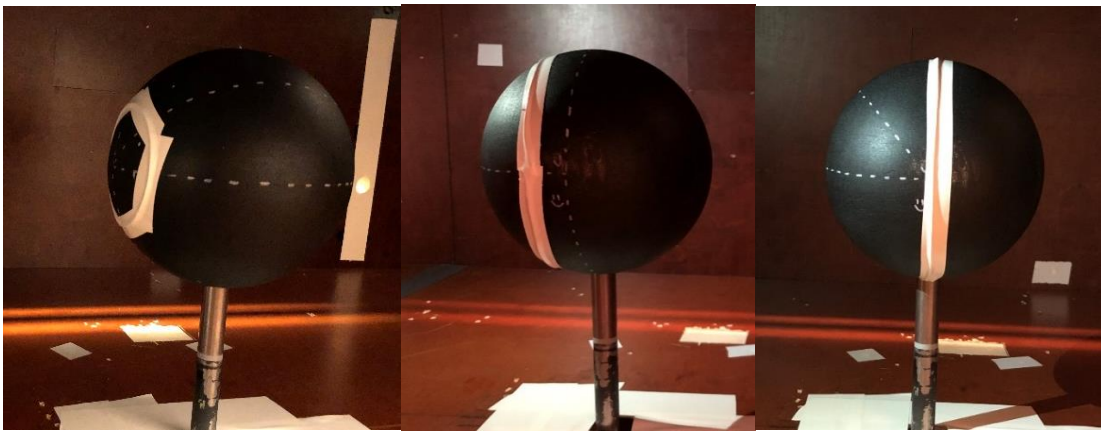


Figure 5 Sphere with wire attached on the angle of 30 (left), 70 (middle) and 90 (right)

In order to see the behavior of the flow on the surface, surface visualization has been done with the application of an oil film to the surface. By selecting an oil with suitable viscosity, the flow patterns can be seen by the behavior of the oil on the surface, without influencing the flow significantly. A mixture of motor oil and canola oil was used, together with white dry pigment, as illustrated in Figure 6.

Figure 6 Material needed in order to paint the surface for visualization

3.3 Results

The drag coefficient for different cases is measured with the help of the sensors in the test section of the wind tunnel and has been shown in Figure 7.

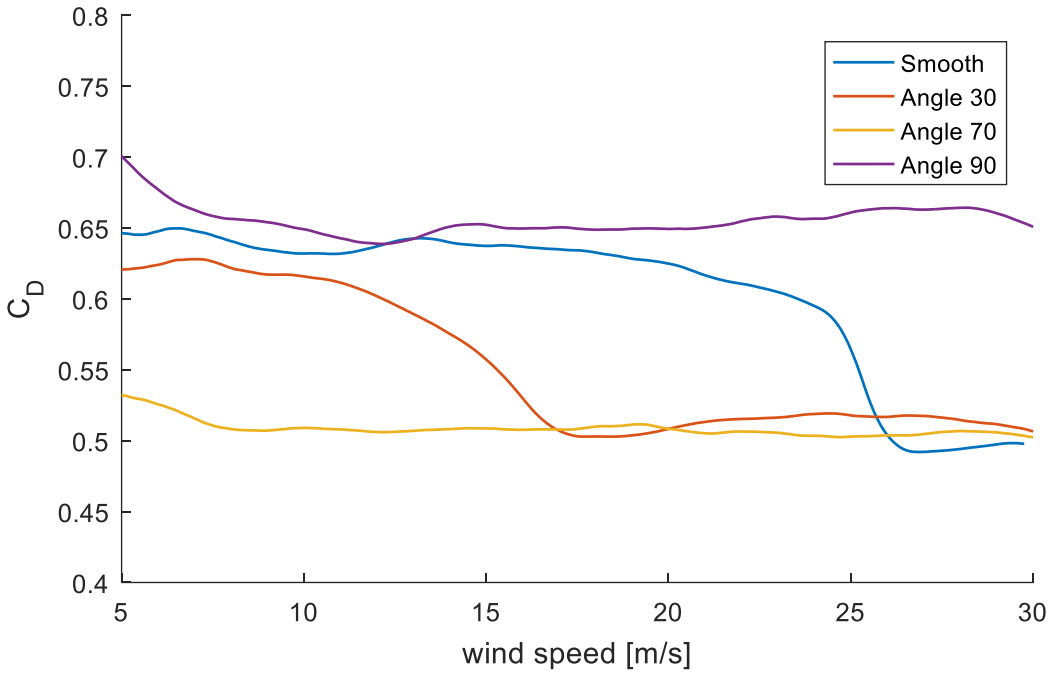


Figure 7 Drag coefficient behavior in different cases for sphere based on different wind speed

For the smooth sphere, there is a drop in drag coefficient around 25 m/s wind speed. This corresponds to a Reynolds number of 280 000, which is close to where other experiments predict the drag crisis. However, the drag coefficient is significantly higher, and the drop in drag is lower than reported in experiments (see Results section). This is believed to be due to the rod, as will be discussed later. Adding the trip wire to the sphere makes it behave in different ways based on the placement of the wire. The drop in the drag coefficient happens earlier when the wire is placed at a 30 degree angle. At 70 degrees it happens even earlier, and the drag behavior is close to supercritical for most of the tested velocity range. There is no specific drop in 90-angle case,

which is because at this angle, the trip wire initiates separation instead of just transitioning the boundary layer. The 70 degree case is the optimal case in order to have low drag at any Reynolds number.

Surface visualization has been done in order to monitor the behavior of the flow over the surface. The flow pattern for wind speed 30 m/s is shown in Figure 8. As it is marked the first line is assumed to be the place where the flow regime changes from laminar to turbulent and the second line is the place where the flow separates from the surface. Additionally, some strange behavior of the flow on the surface behind the sphere is also visible and marked by red circle. This behavior is cause by the presence of the rod, which holds the sphere in the wind tunnel during the experiment.

Figure 8 Surface visualization in wind speed of 30m/s

3.4 Preliminary CFD simulations

To further investigate the influence of the rod on the flow, some preliminary CFD simulations were performed. Figure 9 shows the resulting surface streamlines over the surface, colored by wall shear stress magnitude. The separation location can clearly be seen. The anomalous behavior behind the rod can also be gleaned from the picture. This is further illustrated Figure 10 which shows streamlines in the wake region. Flow going around the cylinder is then dragged up into the wake, creating two counter-rotating vortices in the wake. The velocity contours, shown in Figure 11, also show a significant influence on the wake by the presence of the cylinder. Therefore, we instead continue with simulations of geometries in free flow to isolate the effects and avoid issues with the interaction between the rod and the sphere.

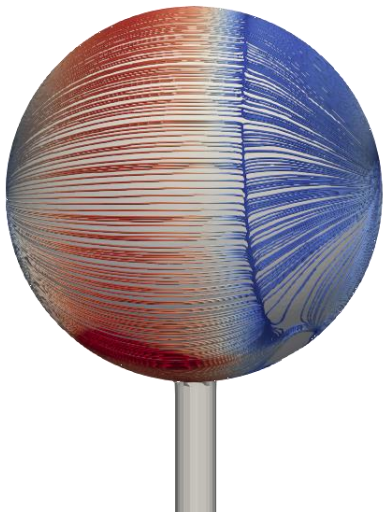


Figure 9 Surface streamlines over the surface at the supercritical flow regime

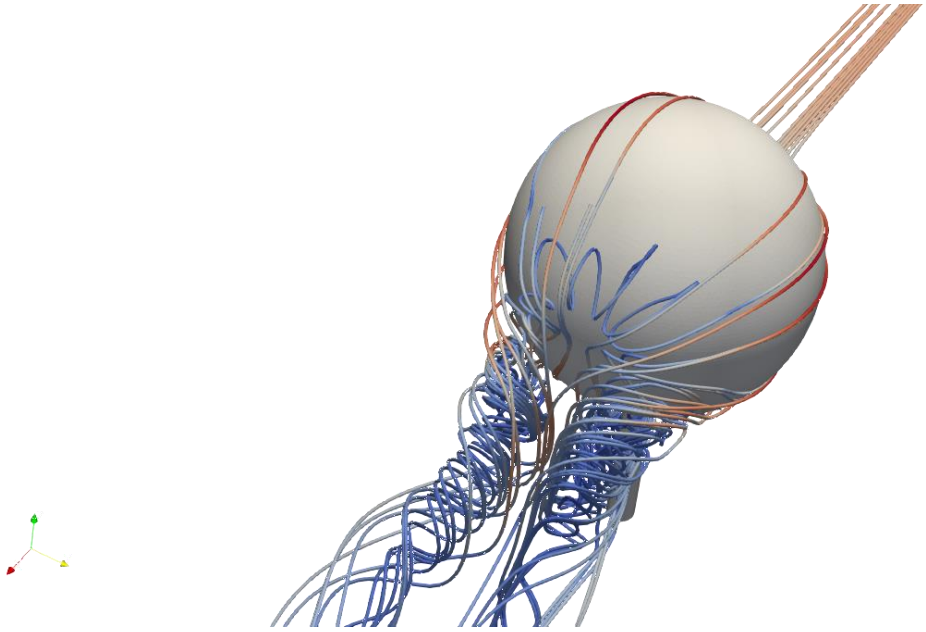


Figure 10 Wake area in supercritical regime

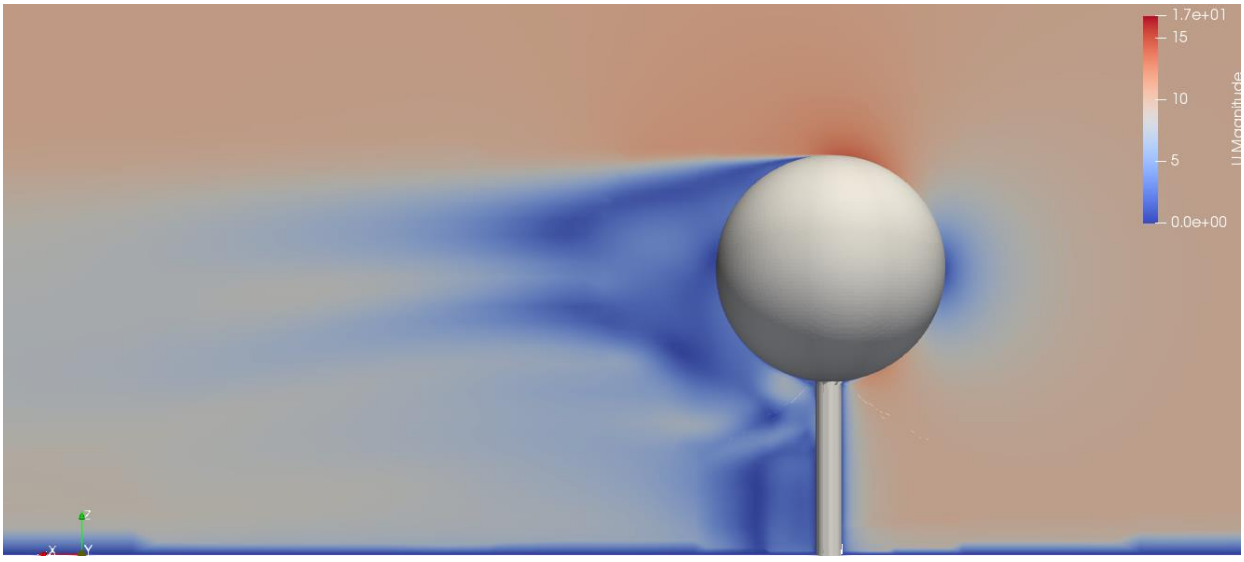


Figure 11 Velocity slice of subcritical region

Chapter 4

4 Simulation of flow over flat plate

To test the implementation of the Langtry-Menter transition model in OpenFOAM, and to ensure proper usage of the model, we first consider a simple test case. The test case is flow over a flat plate, and to verify the results we use experiments from the European Research Community on Flow Turbulence and Combustion (ERCOFTAC). [24] [25] The chosen experiment has zero pressure gradient and a medium turbulence intensity of 3.3 % in the free stream. It should be noted that these experiments are commonly used as benchmark experiments for transitions models, see e.g. the recent model proposed in [26]. These experiments were also used for developing the Langtry-Menter model [27]. We therefore expect good agreement since the models have been explicitly tuned for these particular conditions.

4.1 Computational setup

The plate considered has a thickness of 1.5 mm and has an arc-shaped leading edge. A 2D computational domain is chosen that extends 3 m downstream of the leading edge and 1 m above the plate. The inlet is placed 0.4 m from the leading edge, with an inlet velocity of 5.4 m/s, a turbulent intensity of 3.3 % and a turbulent eddy viscosity ratio of 12. The computational mesh is generated using blockMesh. The grid consists of 28 620 cells and is stretched towards the leading edge of the plate to get high resolution in the boundary layer. The calculated y^+ values give an average value of 0.5 along the plate. The mesh for the full computational domain is illustrated in Figure 12, and Figure 13 shows a close-up of the mesh near the leading edge of the plate.

Steady-state simulations are performed using the simpleFoam solver, with the second-order linearUpwind scheme for all convective terms. The turbulence models considered are laminar flow, k-ω SST and the Langtry-Menter modification of k-ω SST.

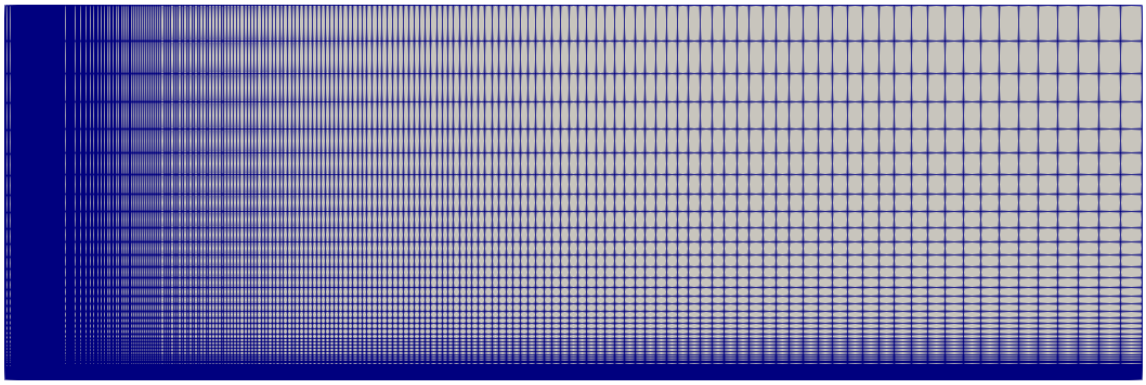


Figure 12 Computational domain and mesh for flow over flat plate

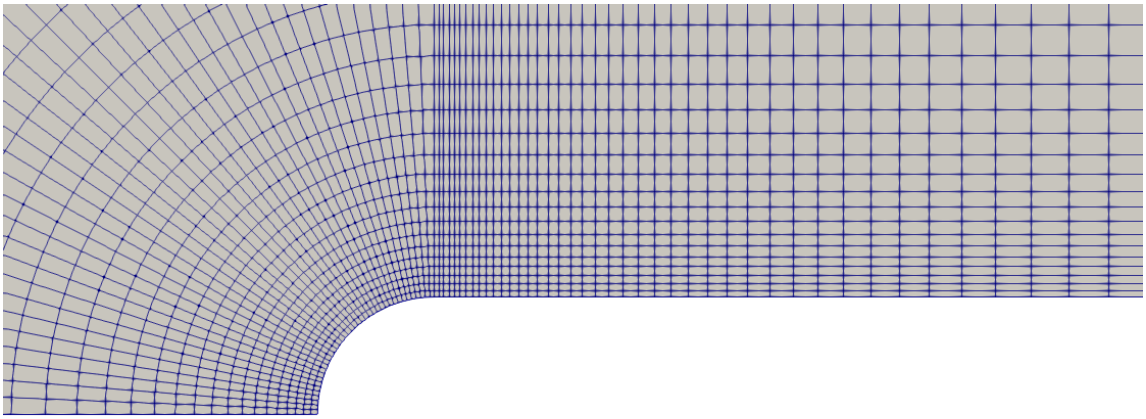


Figure 13 Close-up of mesh near flat plate leading edge

4.2 Results

Figure 14 shows the velocity magnitude near the leading edge of the plate. We see that there is a stagnation point at the tip of the arc-shaped edge and the flow is accelerated over the arc-shaped edge. Then a boundary layer is established and growing along the flat plate.

Figure 15 compares the skin friction as a function of plate Reynolds number between the simulations and the experiment. We see that initially, there is excellent agreement between the laminar simulation and the experiment, indicating that here the flow is laminar. Between

Reynolds numbers 2×10^5 and 3×10^5 there is a sharp increase in the skin friction, which is due to transition to turbulence. The fully turbulent simulation agrees well with the friction after the flow has fully transitioned, but fails to capture the lower friction in the laminar region. The transition model, however, is able to successfully capture both the laminar region, the transition region and the fully turbulent regions.

The experiment also measured the turbulent intensity at a height of 5 cm above the plate. A comparison between the turbulent simulations and the experiment is shown in Figure 16. There is excellent agreement for both the turbulent models.

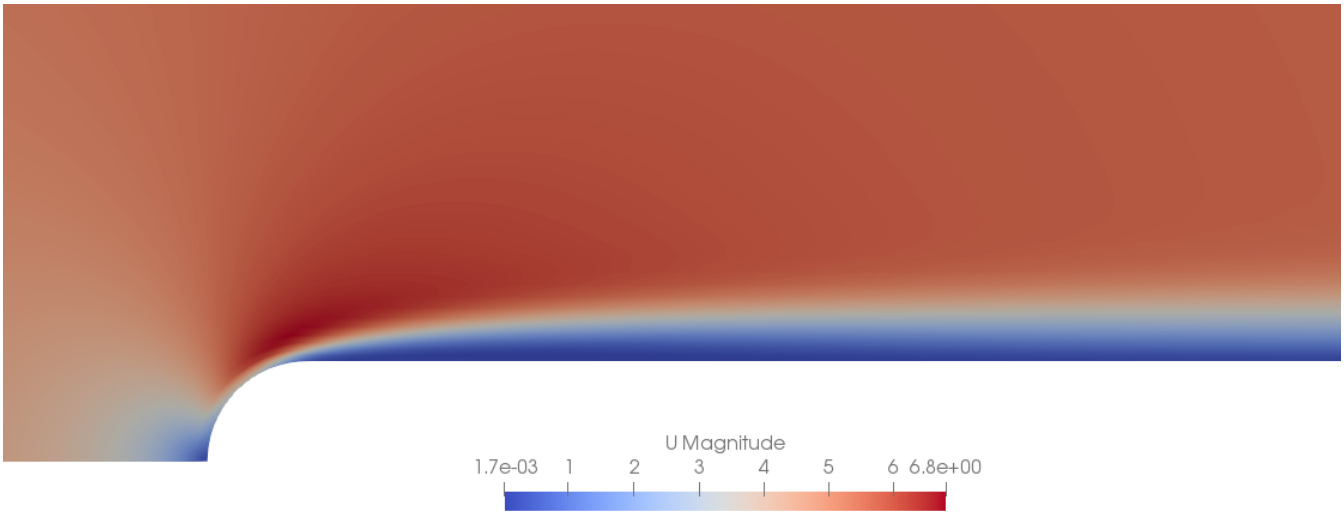


Figure 14 Contour plot of velocity magnitude near the leading edge

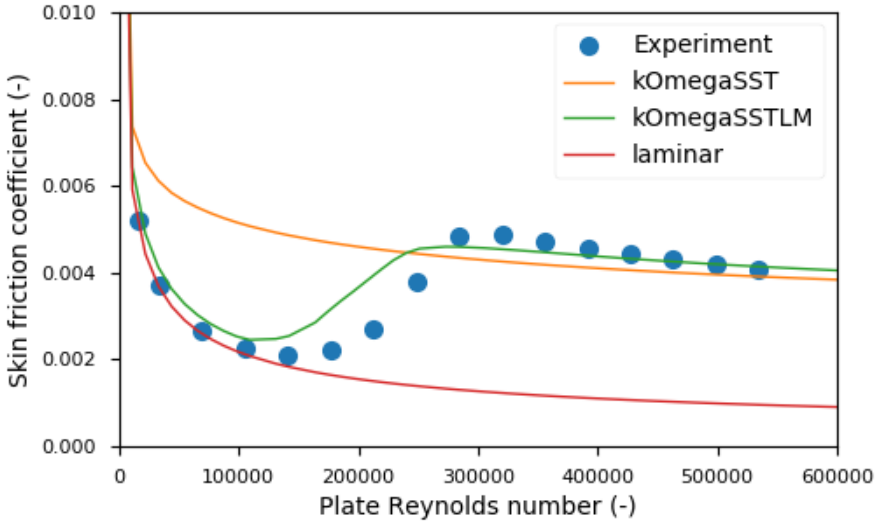


Figure 15 Comparison of skin friction along the plate.

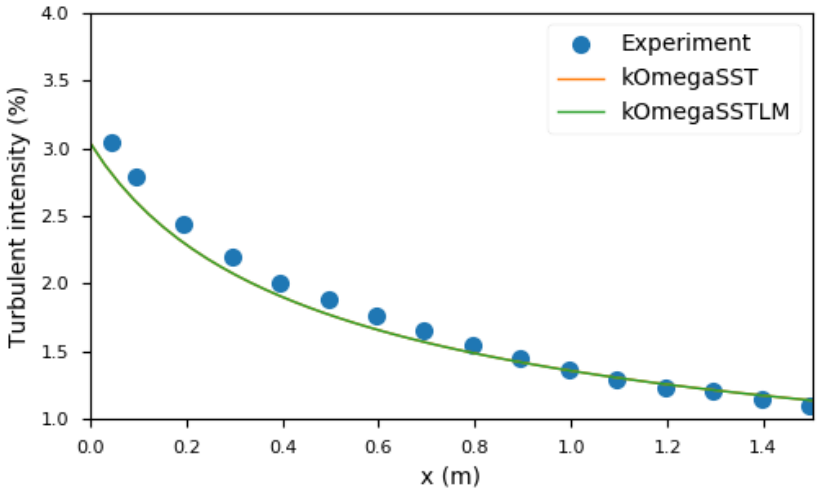


Figure 16 Comparison of turbulent intensity along the plate at a height of 5 cm above the plate

5 Flow over Sphere

Achenbach [17] studied the flow past spheres in the Reynolds number at the range of 5×10^4 and 6×10^6 for smooth surface. Wieselberger [28] studied the flow over sphere which the result of that study has been shown in the Figure 17 next to Achenbach. Maxworthy [29] has done experiments for Reynolds numbers between 2×10^5 and 6×10^6 by measuring the pressure distribution around the circle of longitude under a variety of conditions including the effect of the various boundary layer trip arrangement. These three experimental data are used in the current thesis in order to compare with the numerical results at the end of the current study. Based on the Achenbach research, the flow in the Wieselsberger's experiments has been influenced by the support system. This might explain the difference between these two experimental results, which are shown in the Figure 17. The results of Maxworthy show similar behavior as our experiments on the influence of tripping the boundary layer.

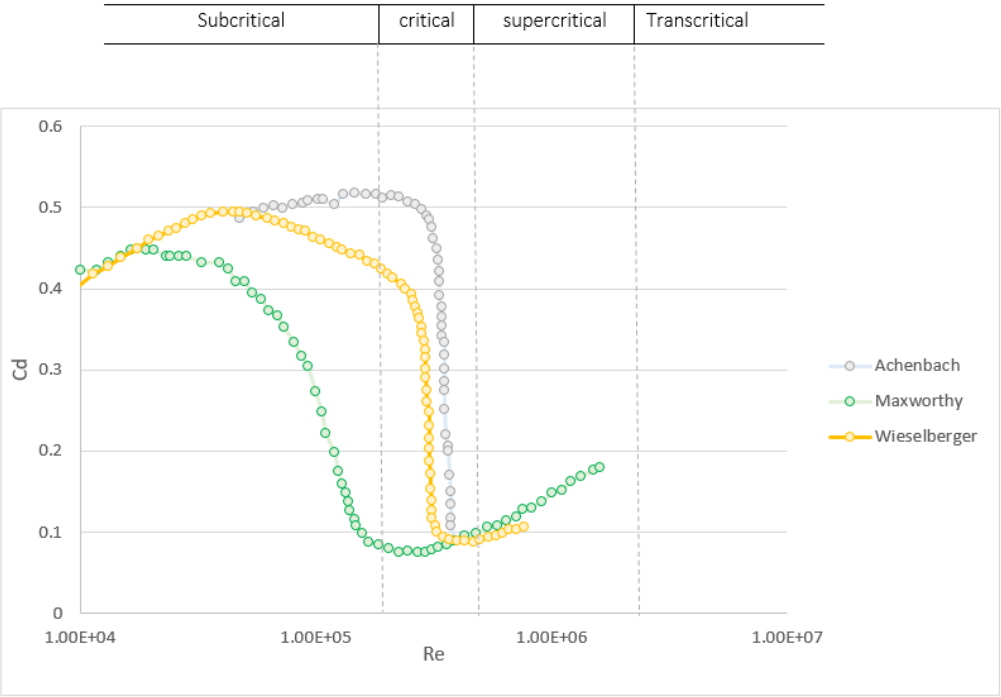


Figure 17 Experimental results used in the current thesis

5.1 Computational setup

In total, fourteen simulations are performed. Seven Reynolds numbers have been defined for both turbulent and transition models, 10^4, 4×10^4, 10^5, 2×10^5, 2.5×10^5, 5×10^5 and 10^6.

In order to create the mesh in the computational domain, two utilities have been used. blockMesh as the background mesh and snappyHexMesh for the mesh around the sphere. The whole calculation setups are set in order to have $y^+ = 1$ using layers near the surface. An illustration of the grid used is shown in Figure 18 and Figure 19.

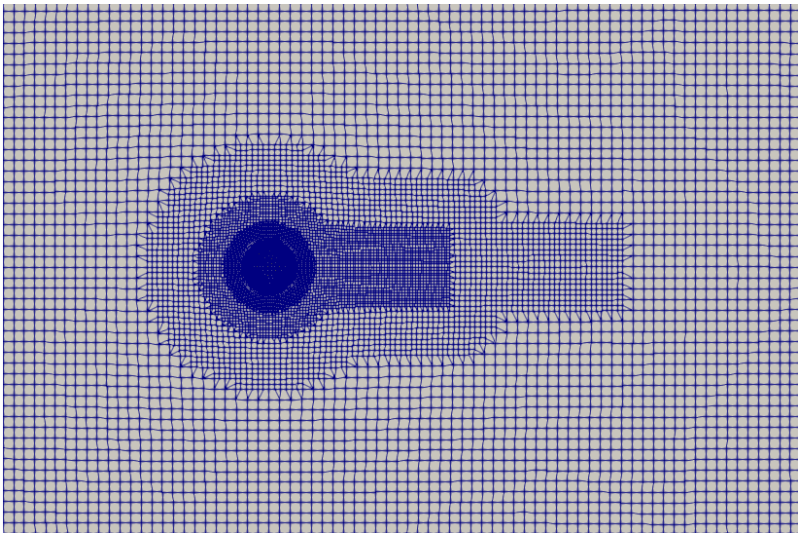


Figure 18 Overview of the mesh in computational domain for Re=5×10^5

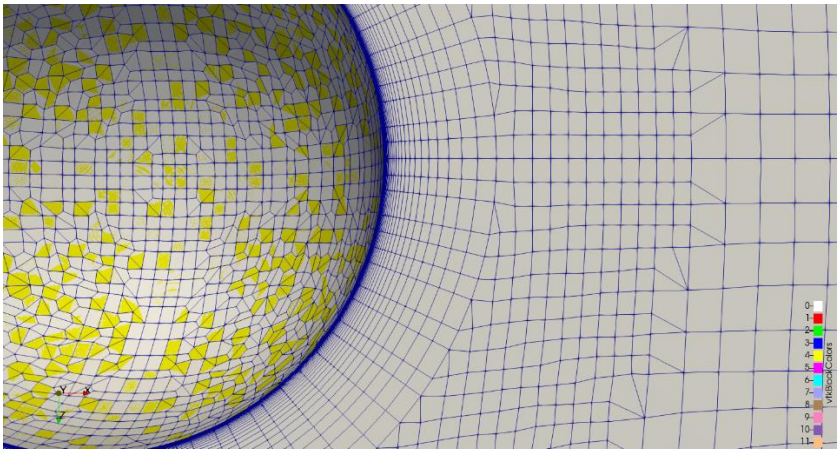


Figure 19 Mesh as close up around the sphere

5.2 Grid sensitivity study

In order to find out the best grid size for our cases, grid sensitivity study has been done for the Reynolds number, 5×10^5. For this Reynolds number, the distance to the first grid cell is set to 0.0448 mm. Table 1 shows the thickness of the first layer that has been used for the rest of the cases.

Table 1 Thickness of the first layer cell based on the Reynolds number calculated by online calculator

Case	Reynolds number	First layer thickness [m]
1	1.00E+04	0.0017
2	4.00E+04	0.000467
3	1.00E+05	0.0002
4	2.00E+05	0.000105
5	2.50E+05	0.0000825
6	5.00E+05	0.0000448
7	1.00E+06	0.000024

For the convergence study, the distance to the first cell is kept constant and the overall grid resolution elsewhere is increased. Figure 20 shows the drag coefficient for four grid sizes. For the two last grids, the diagram converges to the value of 0.166, which is in a good agreement with the experimental results. Therefore the grid size of 493 456 cells is sufficient to reach convergence for Reynolds number, 5×10^5 . For the sake of simplicity, this resolution is kept for all cases and only the distance to the first cell is modified for the remaining cases.

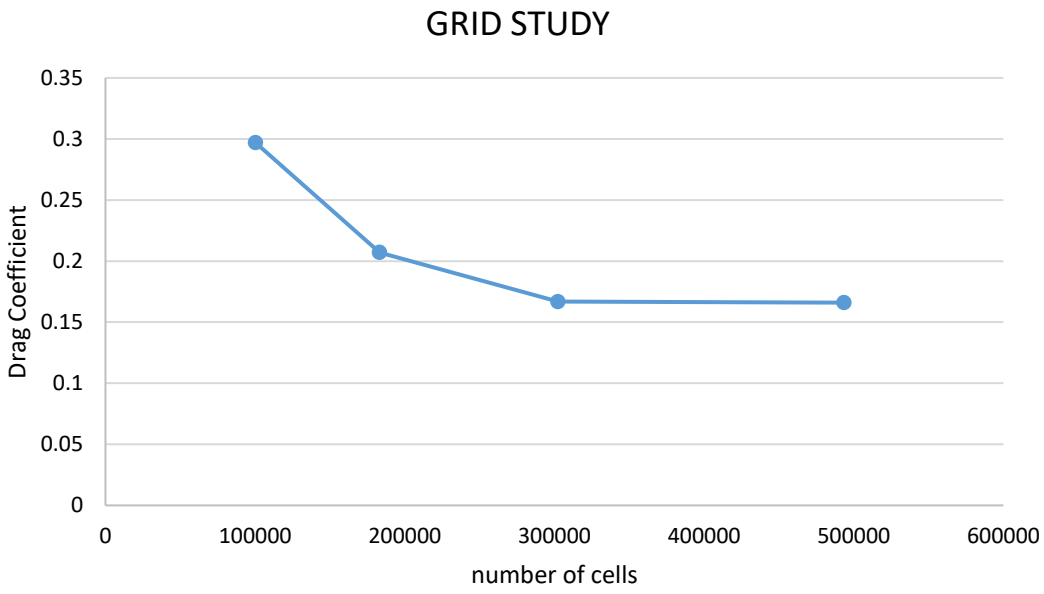


Figure 20 Drag coefficient for 4 grid sizes

5.3 Numerical settings

Flow over the sphere is assumed as an incompressible Newtonian turbulent fluid in steady-state so, the solver for OpenFoam is *simpleFoam*. Two models have been used in this case as it was mentioned. K-Omega SST as fully turbulent boundary layer model and K-Omega SSTLM as transient model. A uniform constant horizontal velocity has been chosen for each case based on the defined Reynolds number and zero gradient for pressure at the inlet of the fluid domain. On the other side the outlet pressure is assumed as static pressure with slip for velocity. For the sphere no slip condition for velocity and zero gradient pressure has been chosen. For the turbulent quantities, the blended wall functions are used that account for the y^+ values to select proper behavior according to the law of the wall.

The discretization scheme used for the convective term is the limited linear scheme, which is a central-upwind scheme using the Sweby limiter. Standard schemes in OpenFOAM are used for the remaining terms.

5.4 Results

The final result is shown in Figure 21, where the numerical results are compared with experimental data from Wieselberger and Achenbach and Maxworthy. [17] [28] [10]

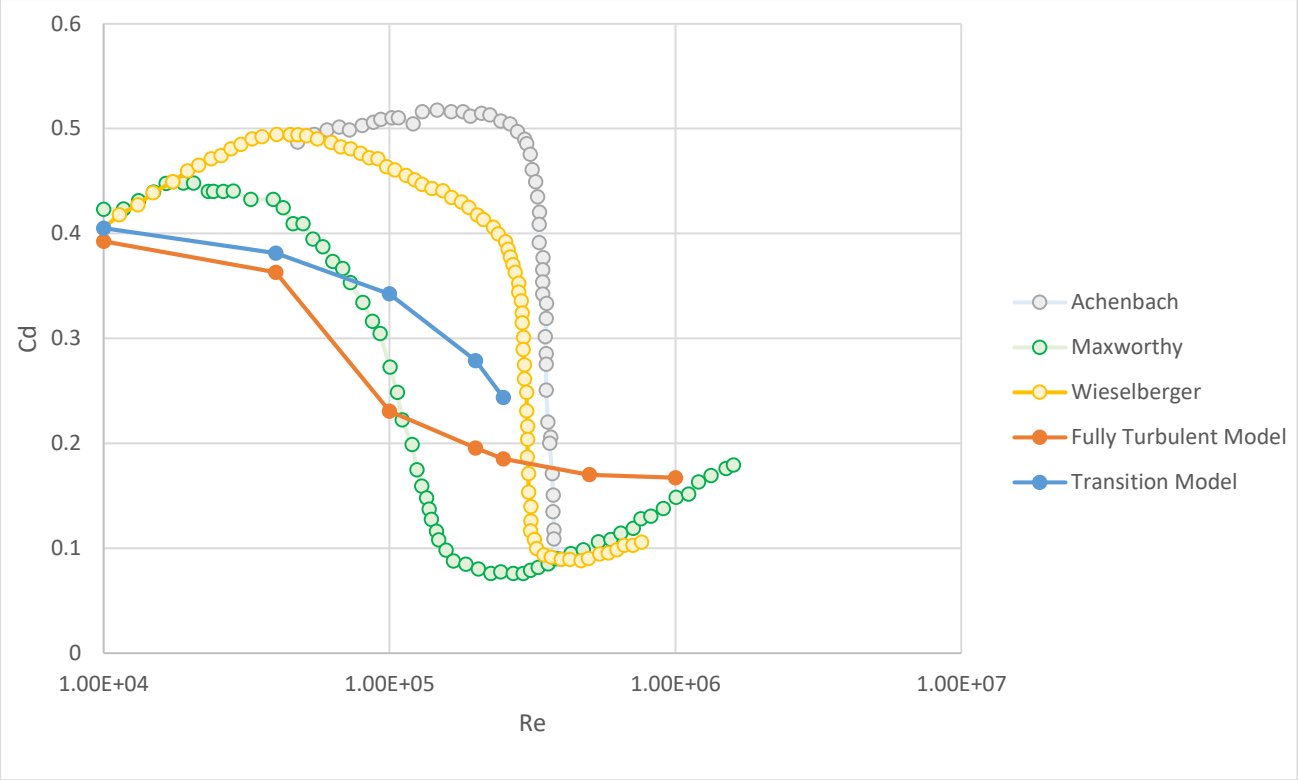


Figure 21 Drag coefficient result for two types of models for seven different Reynolds numbers

Comparing both models with the experimental results shows that in the subcritical region, there is good agreement between the experiments and the simulations. However, in the transition region, the fully turbulent model deviate significantly from the experiments. This is expected, as for the fully turbulent model, the turbulent boundary layer will lead to later separation and thereby smaller wake and pressure drag. In the transcritical region, however, the results are in good agreement with the experiments again.

The transition model gives better agreement to the experiment in the transition region. However, the deviation is still significant.

Figure 22 show contours of velocity for the two turbulent models at $Re = 10^5$. The fully turbulent model has a smaller wake area than the transition model which explains the difference in drag values.

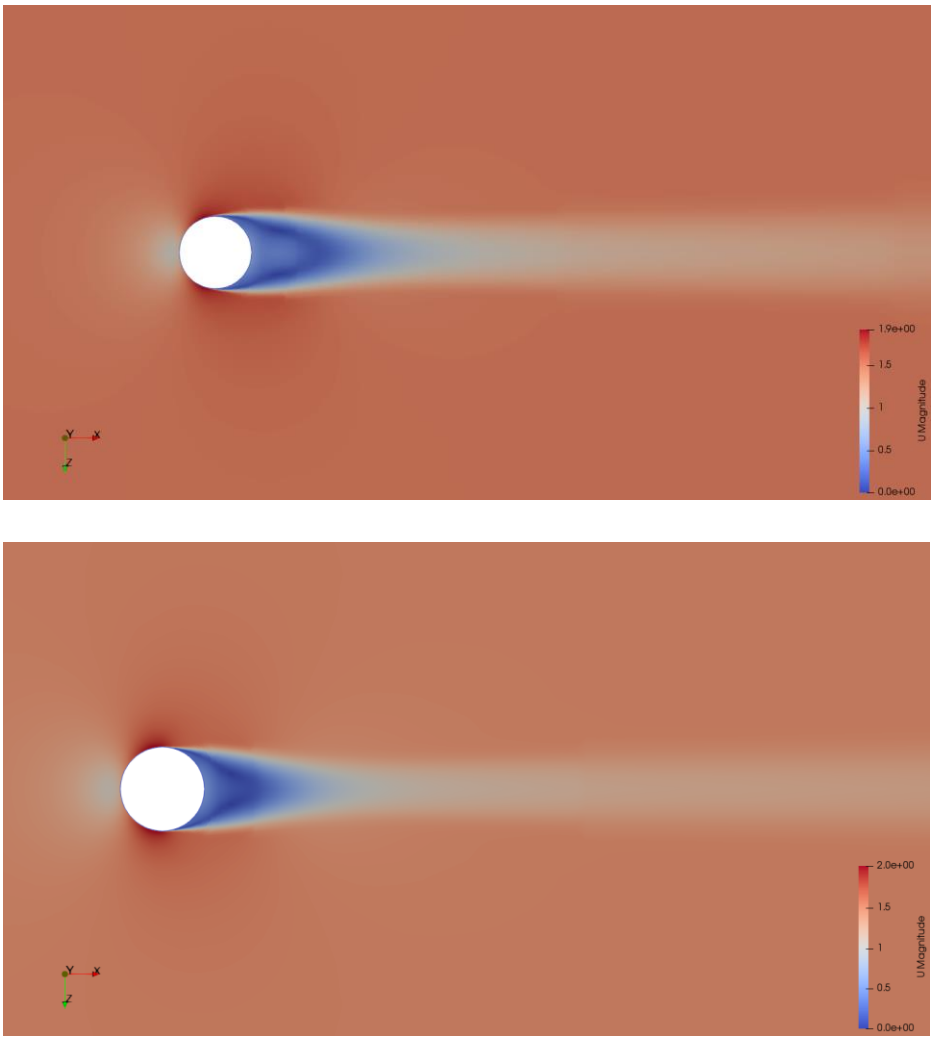


Figure 22 Comparison of velocity contours for fully turbulent model (top) and transition model (bottom) for $Re = 10^5$

As the Reynolds number increases, the separation point moves further back on the sphere and the wake area shrinks. Figure 23 shows the wake for all simulations for the fully turbulent model,

where this effect is clearly seen. There is also some asymmetry in the wake at the larger Reynolds numbers. This was also observed in the simulation performed in [10].

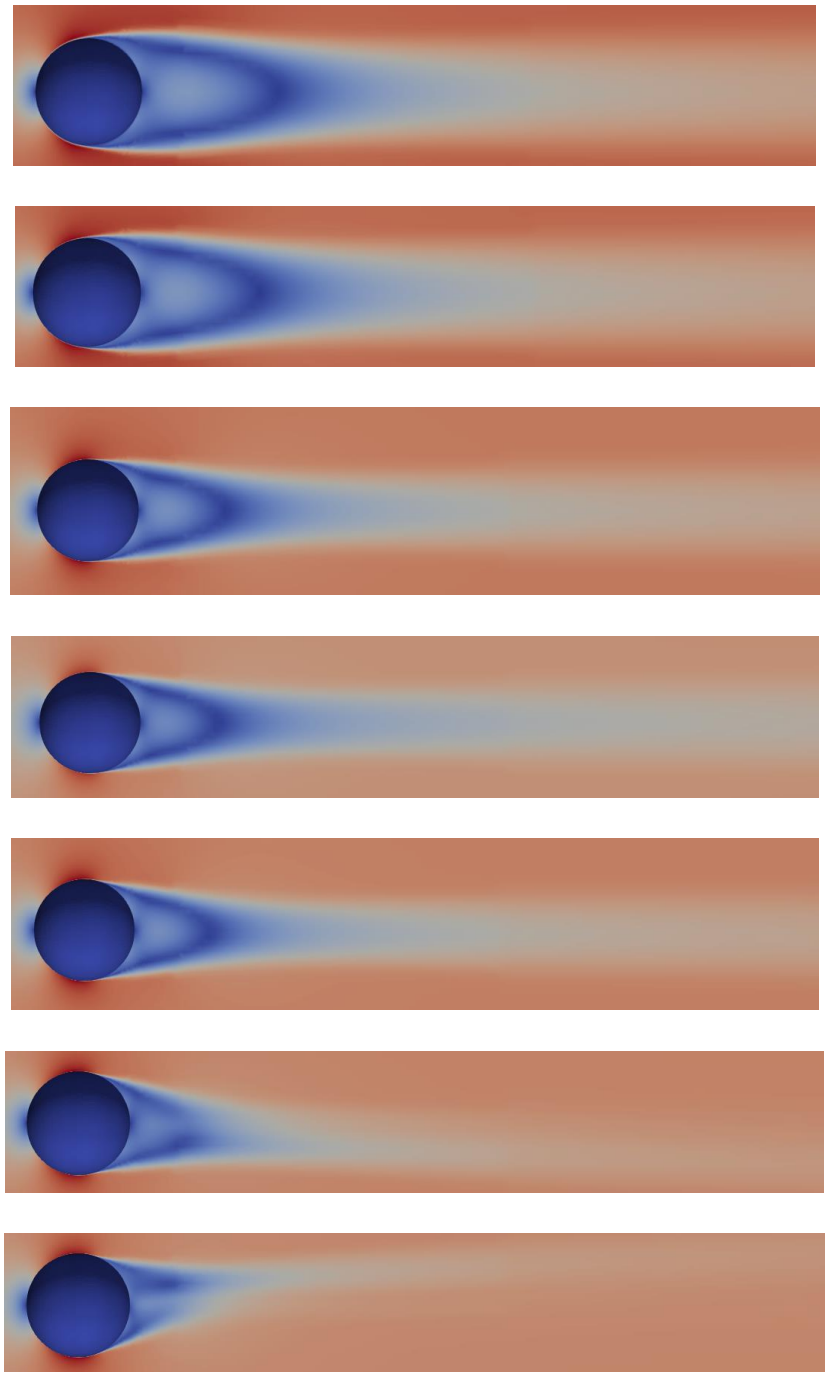


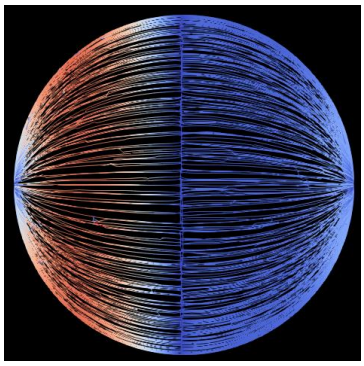
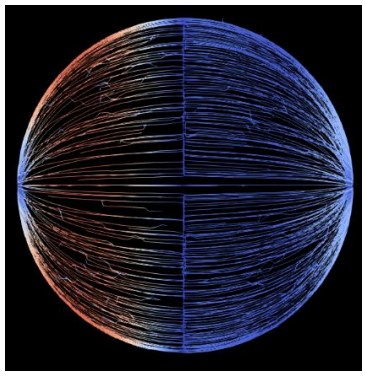
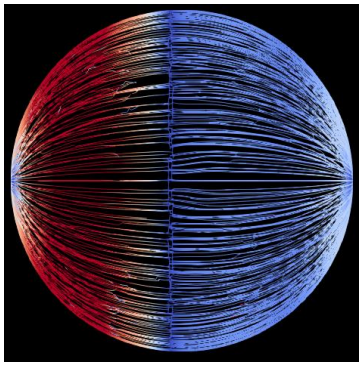
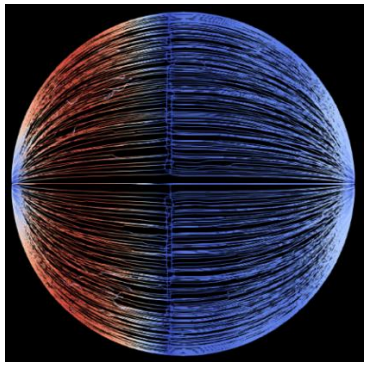
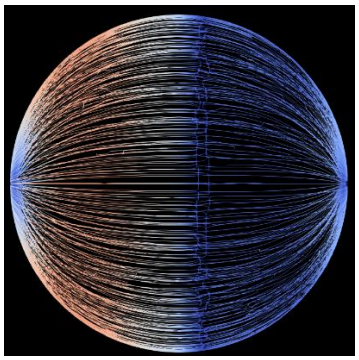
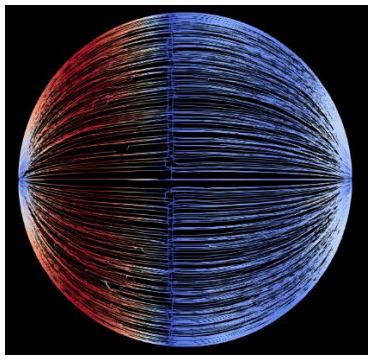
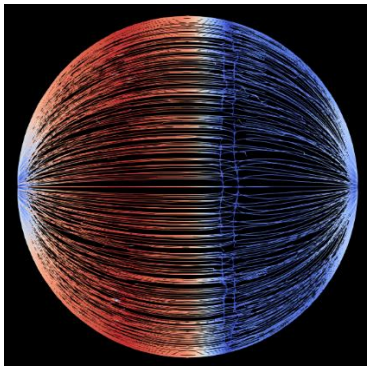
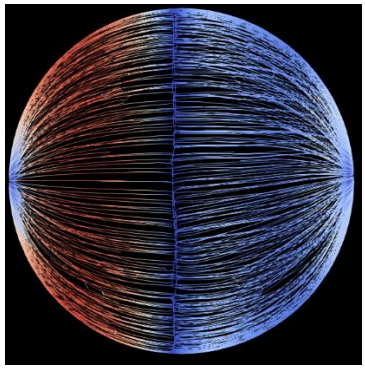
Figure 23 Velocity field for fully turbulent model cases as the Reynolds number increases

Developing a better understanding of the behavior of the flow comparison using surface streamlines is also done.

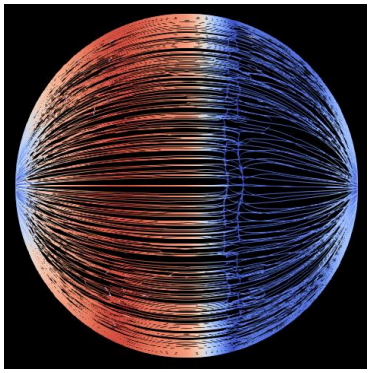
Figure 24 shows the streamline in all the cases and how the separation moves further towards the back of the sphere as the Reynolds number increase. Moreover the difference between the fully turbulent and transition model is visible, with the transitional model having separation earlier.

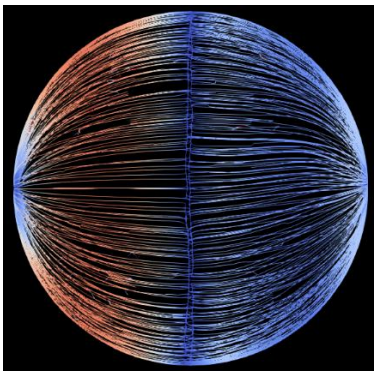
At higher Reynolds numbers, the double separation lines are also visible, especially for the fully turbulent model. This was also observed in the experiment. However, since this behavior is most pronounced for the fully turbulent model, this is most likely not due to transition and should be investigated further.

Finally, a comparison of pressure coefficient along the surface is performed. The simulation values are compared against the experiments in [17]. The experiment at Re = 134 000 are in the subcritical range while the experiment at Re = 340 000 are in the supercritical range. The fully turbulent simulation agree well with the pressure in the supercritical range, which is expected due to the fully turbulent boundary layer. The transitional model is slightly closer to the subcritical pressure, but there are still significant differences, which were also observed for the drag coefficient values.

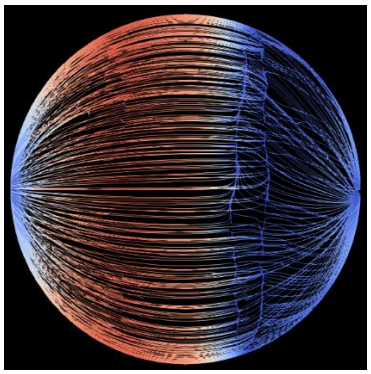
Re	Fully Turbulent Model	Transitional Model
10^4		
4×10^4		
1×10^5		
2×10^5		

2.5×10^5





5×10^5



10^6

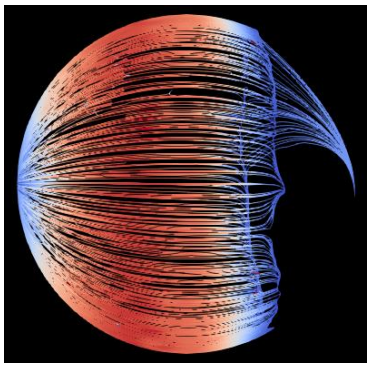


Figure 24 Surface streamlines colored by wall shear stress magnitude for the fully turbulent model and transitional model

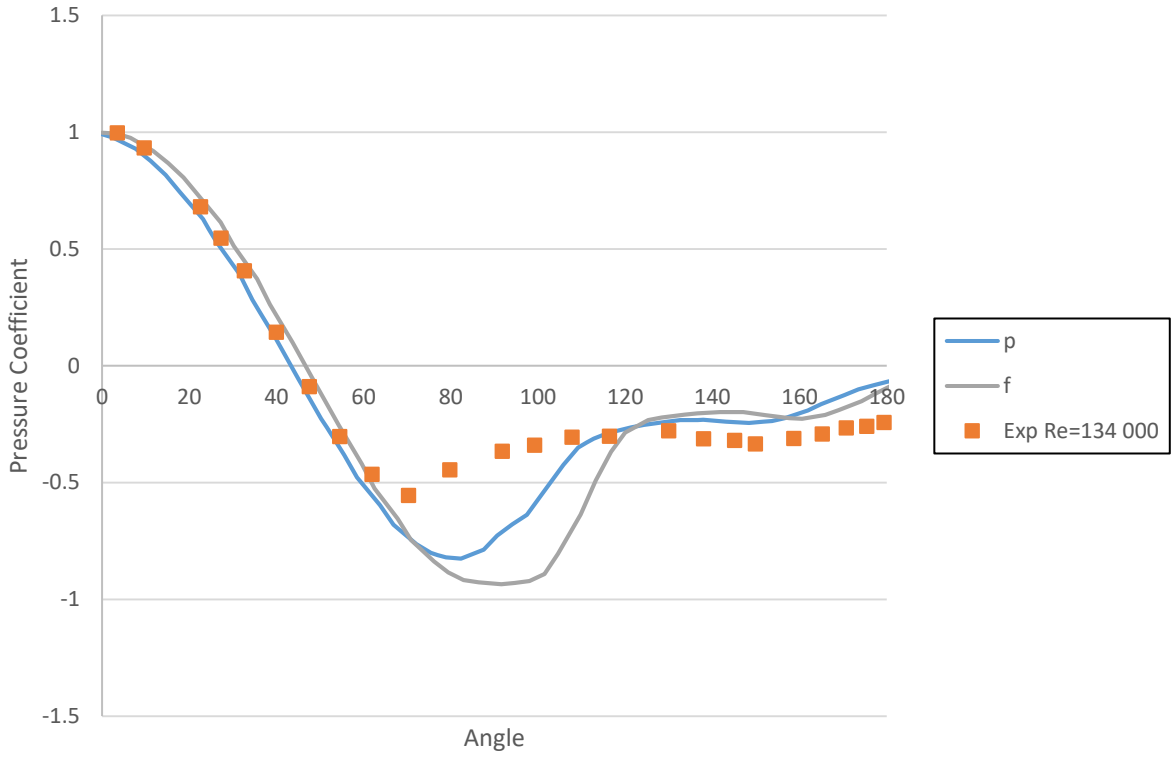


Figure 25 Comparison of pressure coefficient along sphere surface for $Re = 10^5$

5.4.1 Behavior of the drag coefficient in the higher Reynolds numbers in transition model

As it was shown in Figure 21 the transition model is in a reasonable agreement with the experiments in the lower Reynolds numbers. There is a strange behavior in the last two cases, Re= 5×10^5 and Re=10^6, that shows a sudden increase in the drag coefficient amount which was not expected. This is because of numerical instabilities, causing an increase in the velocity fields and wall shear stress. This is shown in Figure 26. These instabilities indicate that the transition model is not as robust as the fully turbulent model. The reason for these instabilities were not investigated further, but could be due to grid quality issues or boundary conditions.

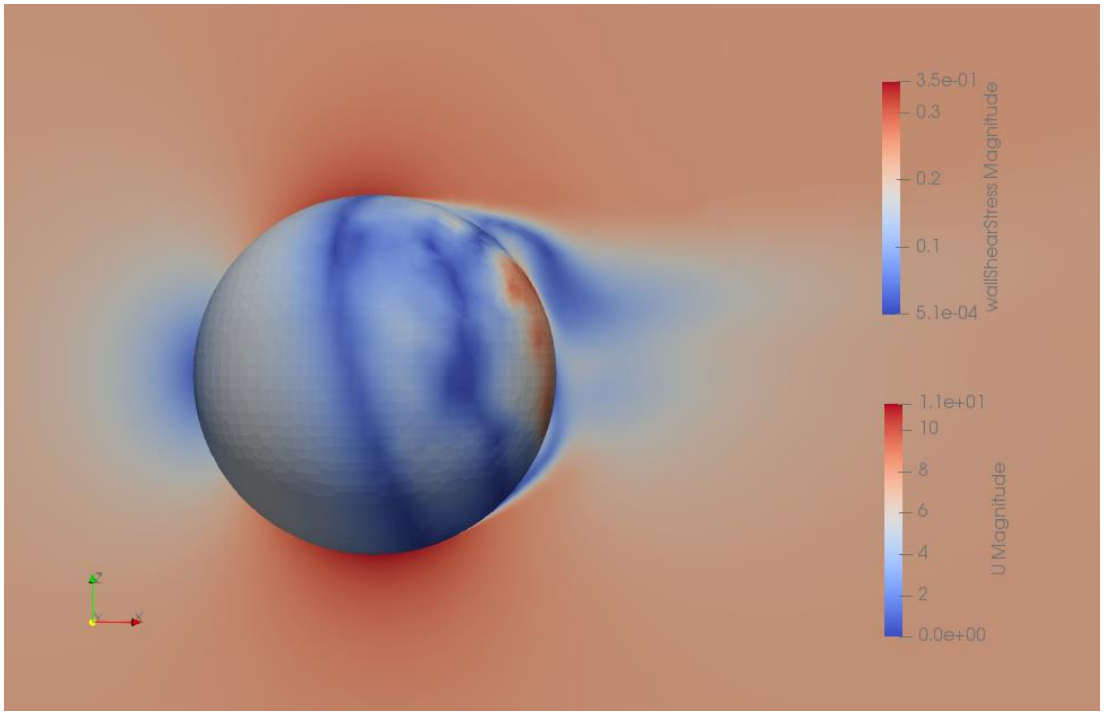


Figure 26 Instability in the results in transient model for high Reynolds numbers

6 Simulation of flow over sphere using DES

In this chapter, DES simulations of flow over the sphere are performed. This is to see if resolving the turbulent eddies shedding from the sphere will lead to a higher drag, closer to the experimental values.

6.1 Computational setup

The same setup as for the RANS simulations are used, with some modifications. First of all, the overall grid resolution is doubled. Additionally, a larger refinement box is applied near the sphere and in the wake region, to better resolve the turbulent eddies. The mesh structure near the sphere is shown in

For the time discretization, the second-order backward scheme is applied. The time step is dynamically based on the Courant number of the flow,

$$Co = \frac{1}{2}\max\left(\frac{\mathrm{sum}(\phi)}{V_c}\right) \qquad (6\text{-}1)$$

Where ϕ is the flux in the momentum equations and V_c is the grid cell volume. The maximum Courant number is set to 0.7.

Note that a rigorous study of grid resolution and time step is not performed, since the purpose of the present simulations is mainly to see the qualitative impact of changing from a RANS model to a scale-resolving model.

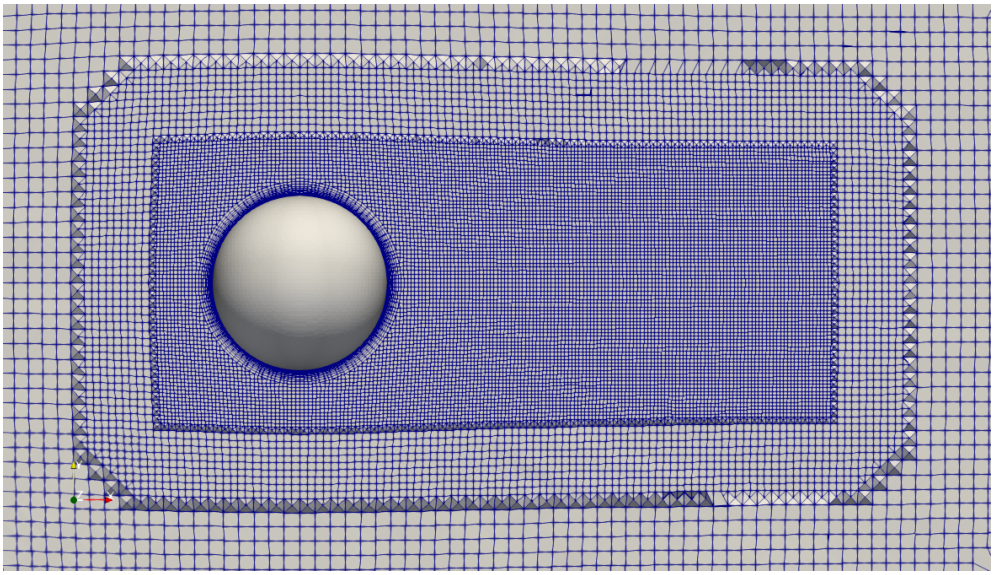


Figure 27 Computational mesh near the sphere for DES simulation

6.2 Results

The drag history for all three simulations is shown in Figure 28. We see that the values eventually oscillate around a mean value, indicating a quasi-steady solution. The average coefficients are given in Table 2, along with the RANS results. The DES values are very close to the RANS values. This means that for a sphere, the time-dependent behavior of the vortices do not significantly influence the mean drag.

However, running a DES simulation also gives additional information not available from a RANS simulation.

Figure 29 shows the Q-criterion colored by velocity magnitude. The vortex tubes shedding from the sphere can be clearly seen, and the overall pattern is similar for all Reynolds numbers. The tubes shed in a plane parallel to the incoming flow, and there is no fluctuating behavior across the flow direction.

One simulation with the Langtry-Menter model is also performed at $\mathrm{Re} = 1 \times 10^5$. However, the same numerical instabilities that were observed for the RANS simulations were also observed here.

Figure 30 compares the surface streamlines and the velocity contours between the fully turbulent model and the transition model. We see that even though the drag is higher for the transition model, the wake is smaller. This means that the higher drag is generated by numerical instabilities giving high velocity gradients near the sphere and thereby high friction drag. We also observer that these instabilities generate larger fluctuations in the velocity field.

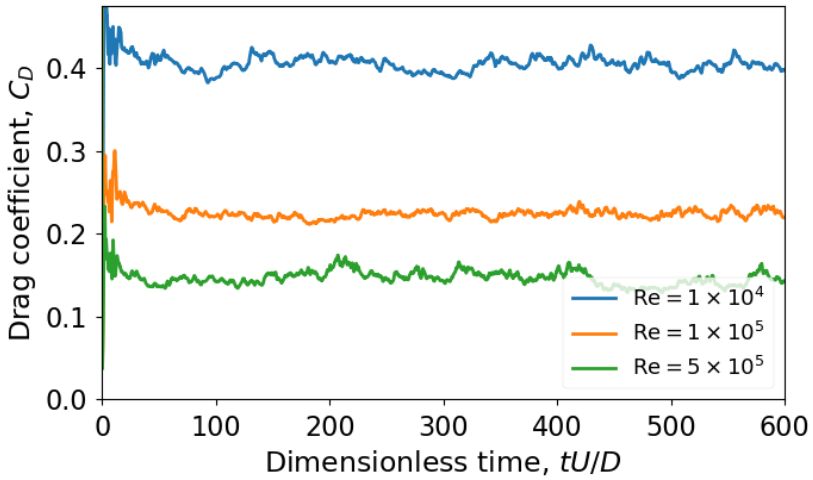


Figure 28 Drag coefficient over time for DES simulations.

Table 2 Drag coefficients and Strouhal numbers for sphere. Results with the transition model in parenthesis

Re	Drag coefficient RANS (LM)	Drag coefficient DES (LM)
1×10^4	0.39 (0.41)	0.40
1×10^5	0.23 (0.34)	0.22 (0.27)
5×10^5	0.17	0,15

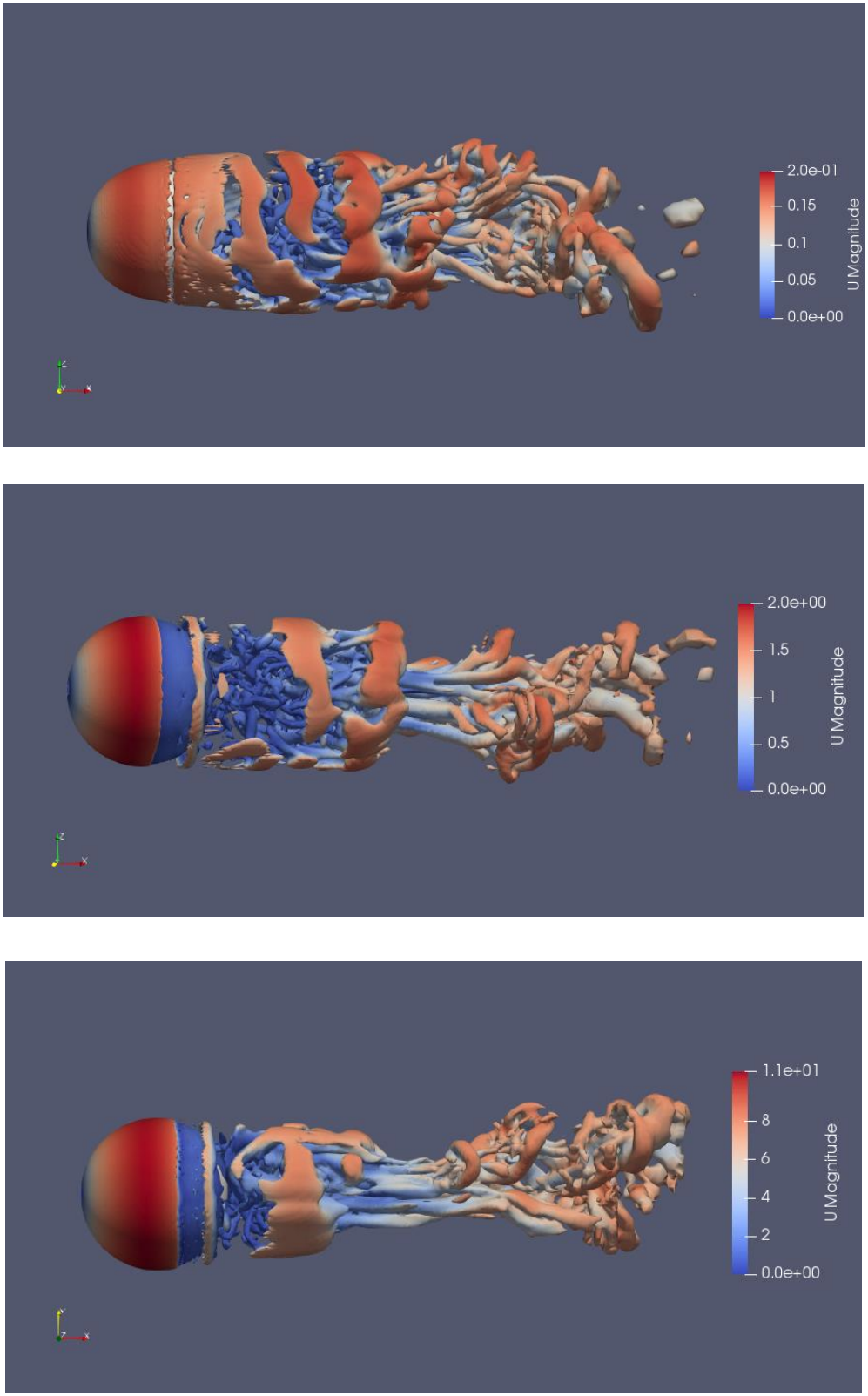


Figure 29 Contour plot of Q-criterion colored by velocity magnitude. Top: Re=10^4. Middle: Re=10^5. Bottom: Re=5×10^5

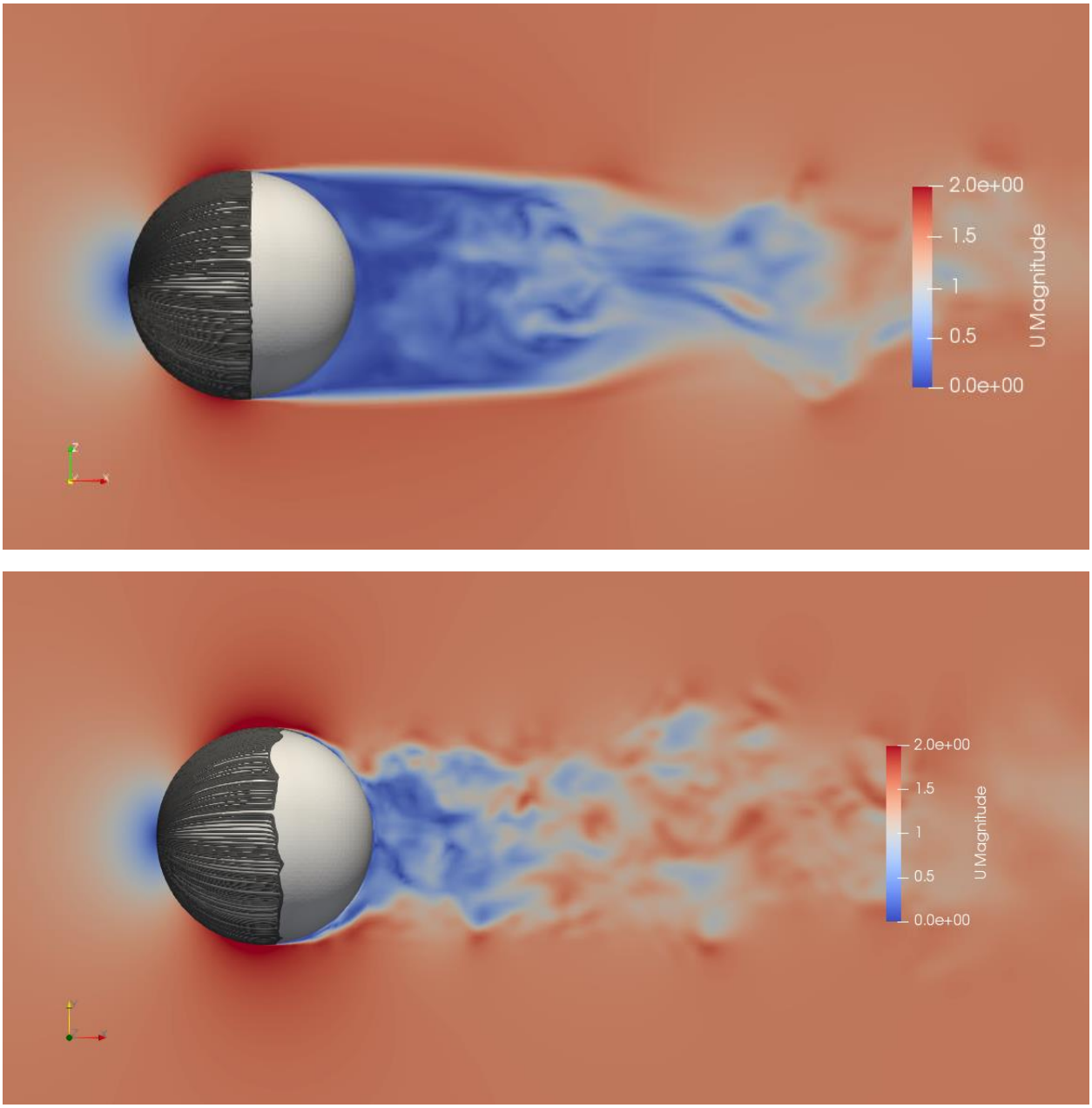


Figure 30 Comparison of surface streamlines and velocity contours for fully turbulent model (top) and transition model (bottom) for Re = 10^5

7 Cylinder case

7.1 Introduction

Analysis of the external flow around the circular cylinder is an interesting area for study. There is a wide range of applications of it in engineering, also in marine engineering. There are a lot of structures where the flow over a cylinder is relevant, such as pipelines, risers, platform legs and wind turbine towers. [21]

The flow past a circular cylinder is associated with various instabilities. These instabilities involve the wake, the separated shear layer and the boundary layer. Williamson has given a comprehensive description of the flow phenomena at different Reynolds numbers. Up to Re = 47, the flow is steady with two symmetric vortices on each side of the wake center line. The first wake instability occurs at Re = 47. Although the flow remains laminar for Re > 47, the flow becomes unsteady and asymmetric. Von Karman vortex shedding is observed for slightly larger Reynolds numbers. At Re=190, three-dimensional instabilities, such as formation of vortex loops, deformation of primary vortices and stream wise and span wise vortices appear in the wake. The wake flow undergoes a series of complex three-dimensional instabilities, eventually making it turbulent. Beyond a certain critical Re, the shear layer separating from the upper and lower surface of the cylinder starts to become unstable via the Kelvin-Helmholtz mode of instability. The transition point, beyond which the separated layer becomes unstable, moves upstream with increase in Re. At Re = 2×10^5, the boundary layer on the cylinder surface undergoes a transition from laminar to turbulent. This transition leads to a delay of the separation of flow from the cylinder surface causing a substantial reduction in the drag force that the cylinder experiences. This is often referred to as the drag crisis.

There are multiple studies of flow over a cylinder, both numerical and experimental. Wieselsberger is one of the first to study the drag coefficient over the entire range of Reynolds numbers. Several attempts have been made to simulate the flow numerically. In Table 3 a summary of results for the drag coefficient and Strouhal numbers for numerical simulations are given. Most studies focus on either the subcritical regime or the supercritical regime. In the

subcritical regime, most successful studies have used a LES (Large Eddy Simulation) method, while for the supercritical regime, good results have also been obtained with RANS (Reynolds-Averaged Navier-Stokes) methods. However, very few studies have been performed in the transitional regime.

Table 3 Summary of drag coefficient and Strouhal numbers found in the scientific literature

Re	Contributors	Cd	St
3900	Cardell [30] [31]		0.215
	Lourenco and shih [32]	0.99	
	Norberg [33] [34]	0.98	
	Ong and Wallace [35]		0.21
	Ma et al. (Case2) [36]	0.84	0.22
	Mittal and Moin [1]	1.00	0.21
	Breuer (case D3) [2]	1.02	0.22
	Kravchenko and Moin [6]	1.04	0.21
	Franke and Frank [3]	0.99	0.21
	Alkishriwi et al. [4]	1.05	0.22
	Mani et al. [8]	0.99	0.21
	Meyer et al. [5]	1.07	0.22
	Ouvrard et al. [37]	0.94	0.22
	Wornom et al. [38]	0.99	0.21
	Lysenko, Ertesvag and Rian (SMAG method) [39]	1.18	0.19
	Lysenko, Ertesvag and Rian (TKE method) [39]	0.97	0.209
1.00E+06	Shih et al. (1993) Experimental [9] [40]	0.24	0.22
	Iaccarino and Moin (LES Method) [9]	0.31	0.35
	Iaccarino and Moin (RANS Method) [9]	0.39	
	Iaccarino and Moin (URANS Method) [9]	0.4	0.31
	Catalano et al. 3D LES [9]	0.31	0.35
	Catalano et al. URANS [9]	0.41	
	Sing and Mittal 2D LES [41]	0.591	

	Ong, Utnes, Holmedal, Myrhaug and pettersen [21]	0.5174	0.2823
	Published experimental data	0.21-0.63	0.18-0.5
3.60E+06	Ong, Utnes, Holmedal, Myrhaug and pettersen [21]	0.0766	0.3052
	Published experimental data	0.36-0.75	0.17-0.29
4.00E+06	Catalano et al (URANS Method) [9]	0.46	

7.2 Computational Setup

A total of six simulations are performed. For three Reynolds numbers, 3900, 10 000 and 50 000, simulations are performed for both fully turbulent boundary layers and with the transition model.

The simulations are performed on an O-grid, with a computational domain extending 25D in the radial direction and π×D in the transverse direction. The full computational domain is shown in Figure 31.

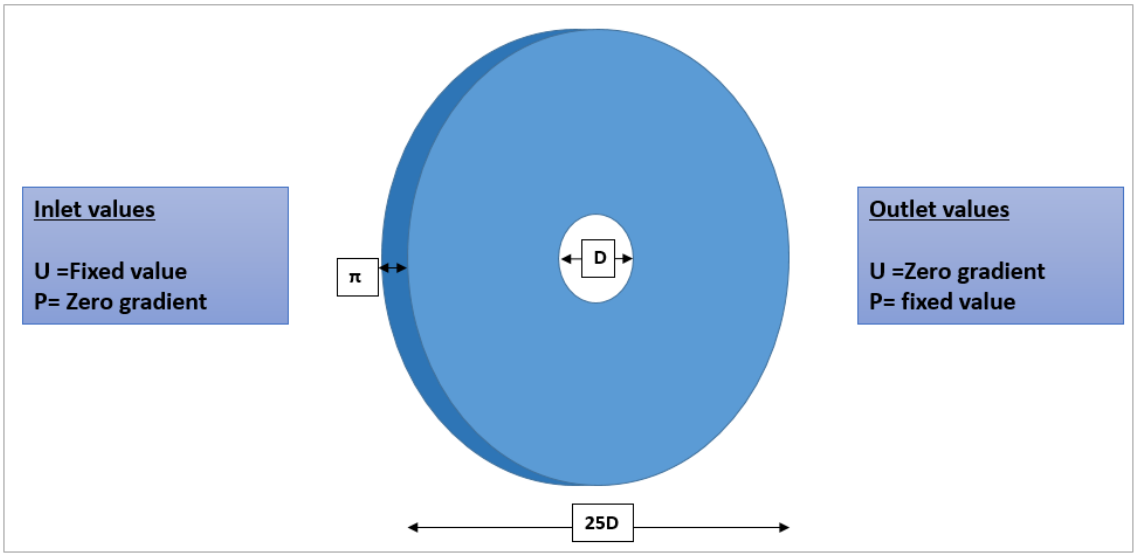


Figure 31 Scheme of the computational domain

The boundary conditions used for the numerical simulations can be defined as follows. Velocity will be assigned as fixed value in the inlet. On the cylinder boundary no slip condition for the velocity will be applied and in the outlet velocity is zero gradient type. The pressure boundary condition in inlet is of a type zero gradient on the cylinder. This is happening because there is no flow through the wall so no pressure gradient should exist normal to the wall. On the other boundaries, the boundary condition is fixed value pressure.

The time step is set to maintain a Courant number of approximately 0.5. The total simulation time is U/D=300 and statistics are taken from the last U/D=100 times.

Since the results of CFD simulations are typically stored at the cell centers, it is important to interpolate the results from cell centers to the face centers, to obtain the fluxes for the surface integrals in the transport equation. The discretization scheme used for the convective term is the Linear Upwind Stabilized Transport (LUST) for the momentum equation and the limited linear scheme for the remaining equations. The time discretization scheme used is the second-order backward scheme.

7.3 Grid sensitivity study

To evaluate the required grid size, a grid sensitivity was performed for the Reynolds number 3900. The number of grid cells in the radial, tangential and transverse directions are denoted N_r, N_θ andN_z, respectively. Here, $N_\theta = N_r$ and$N_z = \frac{N_r}{2}$. Four different grid resolutions were considered, N_r = 60, 80, 100 and 120.

The total expansion ratio has been calculated corresponding to y^+ ~ 1. It is important that the mesh near the wall is properly sized to ensure accurate simulation of flow field. Thus we use online calculator to compute the height of the first mesh cell off the wall required to achieve y^+ ~ 1 using flat-plate boundary layer theory. For a Reynolds number of 3900, this gives a first grid cell size of 4 mm. The grid that has been created for this case has been illustrated in the Figure 32 and Figure 33.

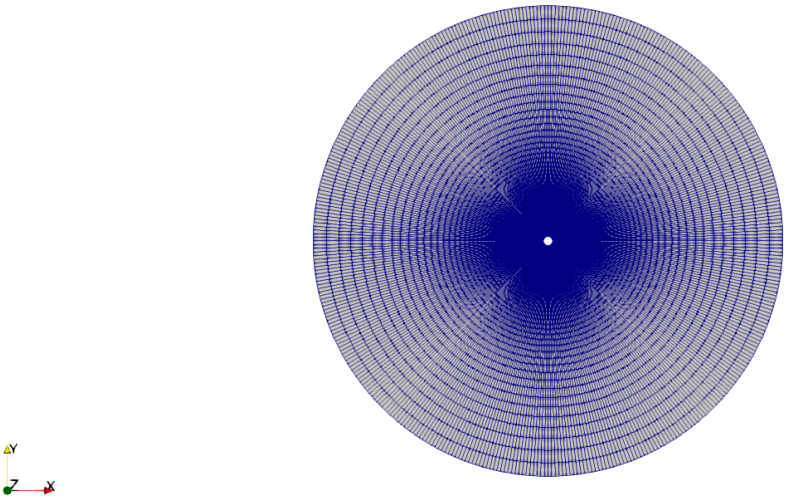


Figure 32 Mesh for the whole domain

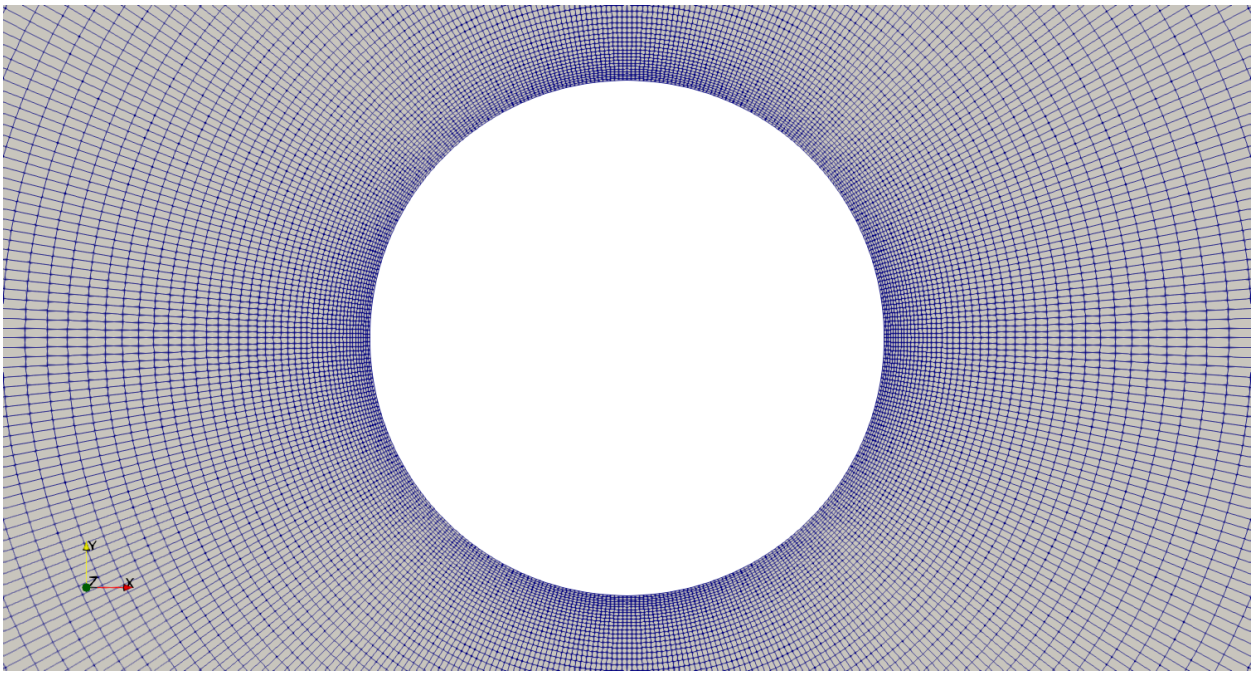


Figure 33 Mesh as close-up

Figure 34 shows the drag coefficient for the four grids. For the last two grids, the coefficient converges to a value of 1 and so the results appear to converge for $N_r = 100$.

The instantaneous flow field at the final time step is shown in Figure 35, for a resolution of $N_r = 100$. The plot shows an isocontour of the Q-criterion with value 0.25, colored by the velocity magnitude. We see the shear layer separating from the cylinder surface and breaking up into vortices.

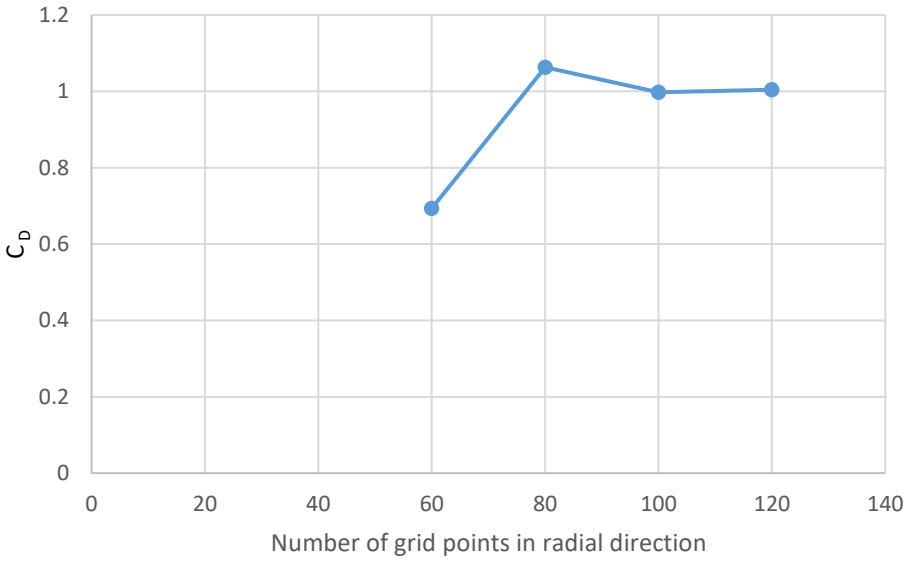


Figure 34 Drag coefficient for 4 grids

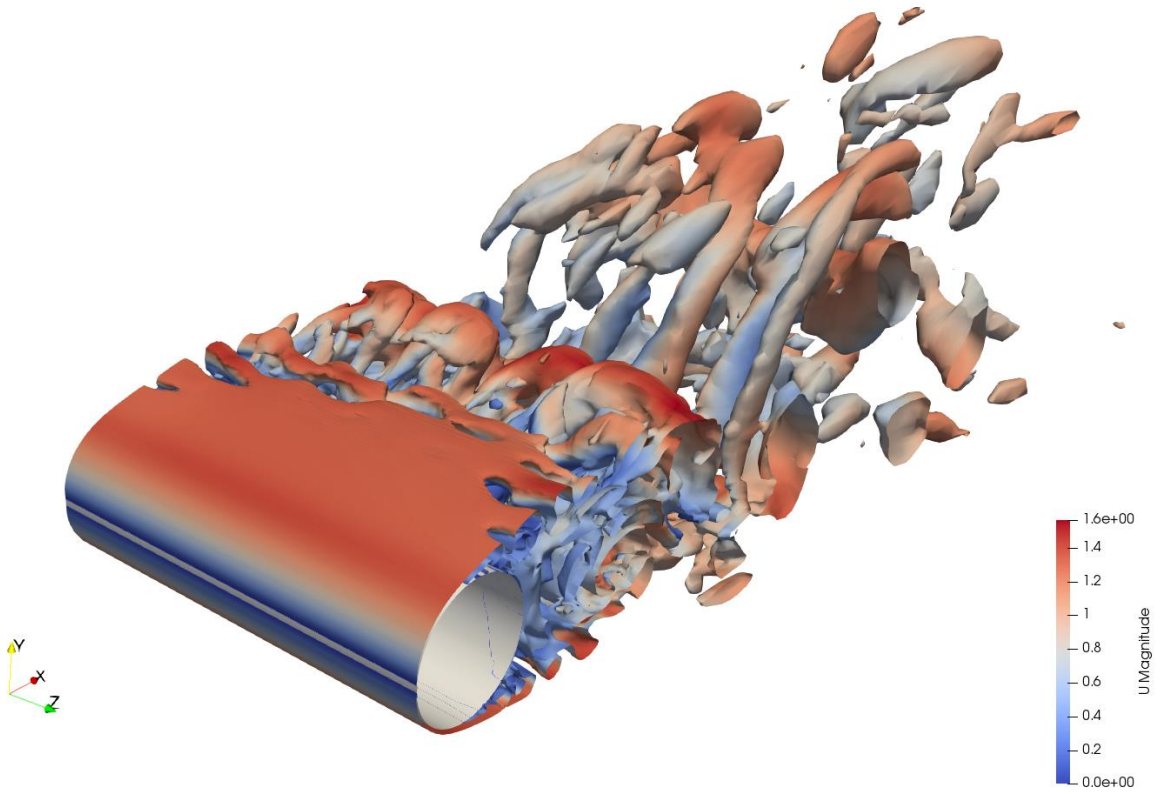


Figure 35 Q-criterion=0.25 colored by velocity magnitude

The results are also compared in terms of pressure coefficient along the surface, and velocity in the wake. Figure 36 illustrate the pressure coefficient with respect to angle of stagnation. There is a good agreement between the simulation results and the experiments.

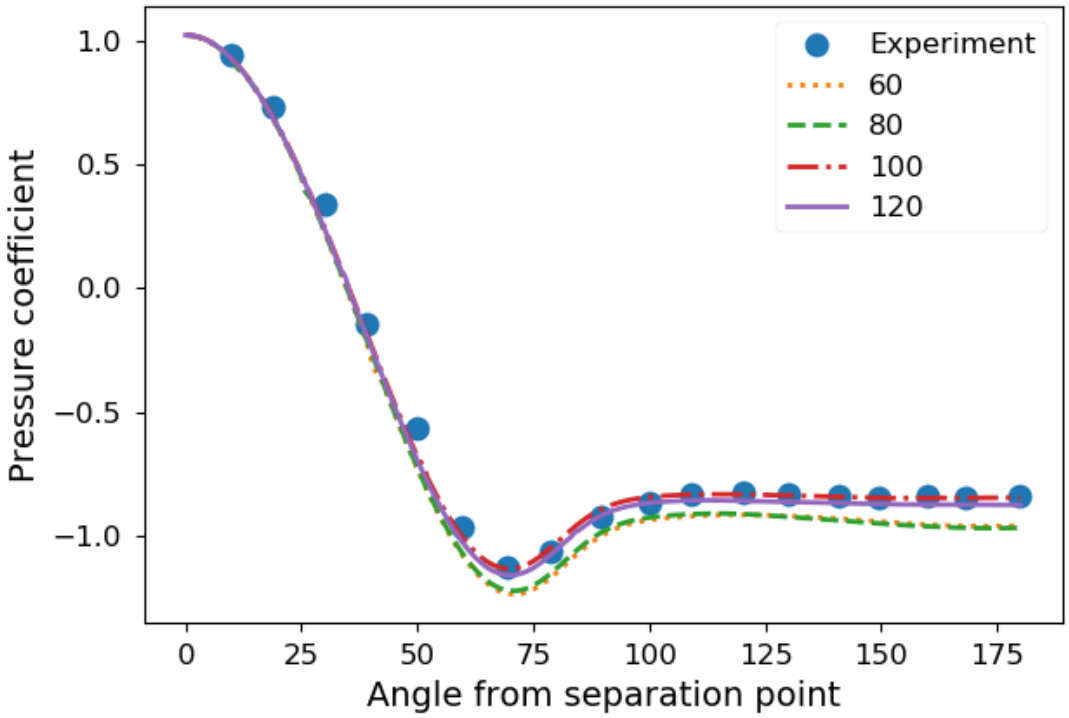


Figure 36 CP coefficient with respect of Angle from separation point

Figure 37 compares the statistics for the near wake of the cylinder with the experiments results at three different downstream locations in the very near wake (X/D = 1.06, 1.54, 2.02) for N_r= 100. Again, there is good agreement between the experiment and the simulation.

With this resolution, the mesh expansion ratio is 1.06. For the remaining simulations, the first cell size is calculated to keep y^+ = 1, and the same expansion ratio is used. This means that the number of grid cells increases with the Reynolds number.

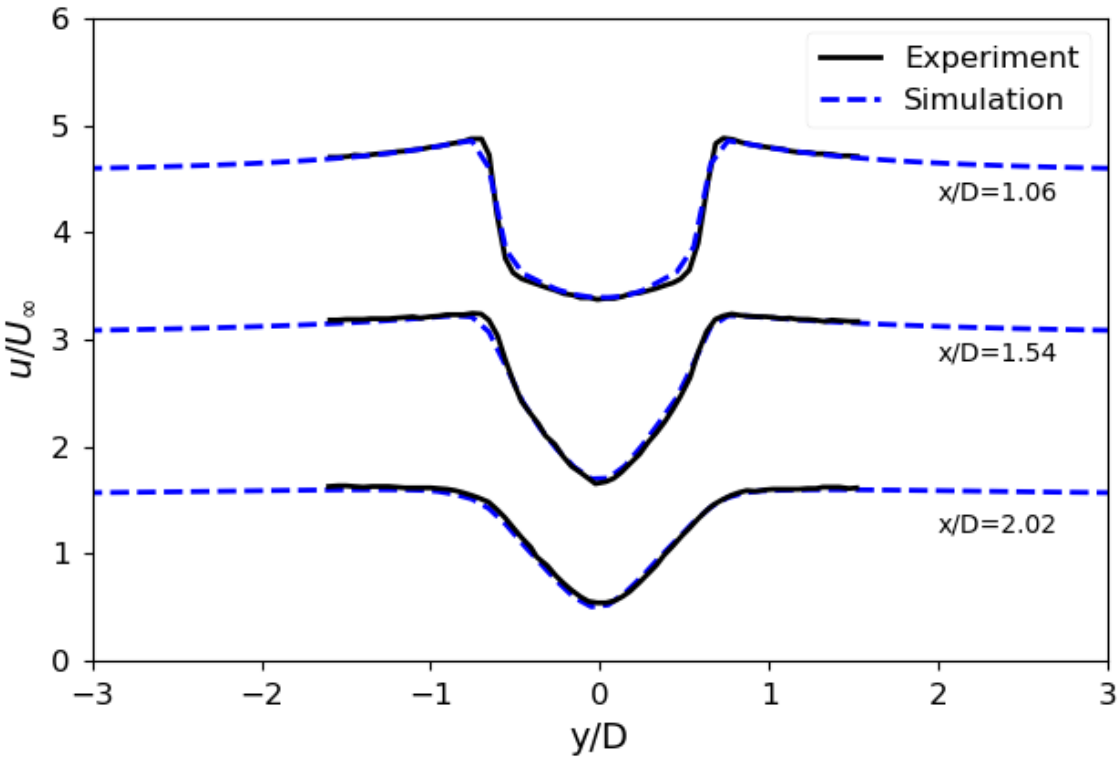


Figure 37 Mean velocity at different locations in the wake of a circular cylinder at Re=3900

7.4 Results

The drag coefficients for three different Reynolds numbers with turbulent model and transitional model are shown in Figure 38 next to the experimental results of Wieselsberger [28].

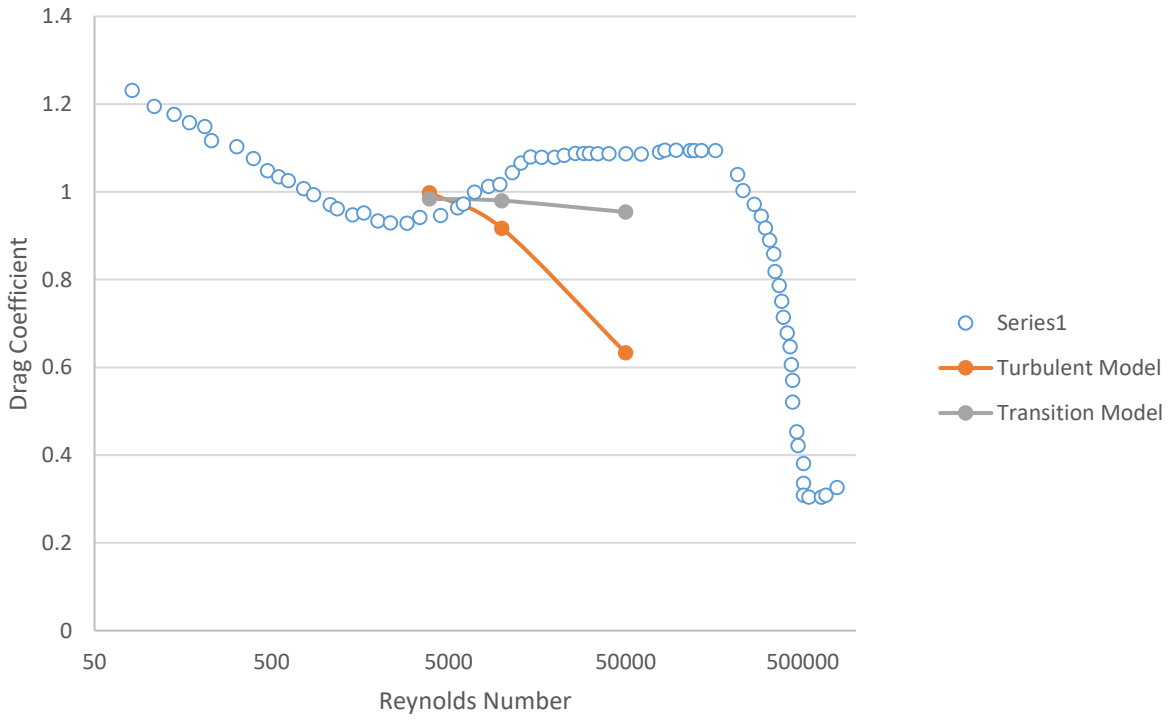


Figure 38 Drag coefficient results for two type of models in three different Re numbers

Comparing the computed drag coefficient as a function of Reynolds number with defined models and experimental results, the transitional model appear to be in better agreement with experimental results than the fully turbulent model. In particular, the fully turbulent results appear not to capture the flattening of the drag coefficients before the drag crisis occurs. This is expected, since for fully turbulent boundary layer, the flow separates later, which for the current geometry, means a smaller wake and lower drag values.

The biggest difference is observed for Reynolds number 50 000. Figure 39 shows a comparison between the mean velocities for the two models. The fully turbulent case clearly has a later separation compared to the transition case. This leads to a smaller wake and hence also lower drag values.

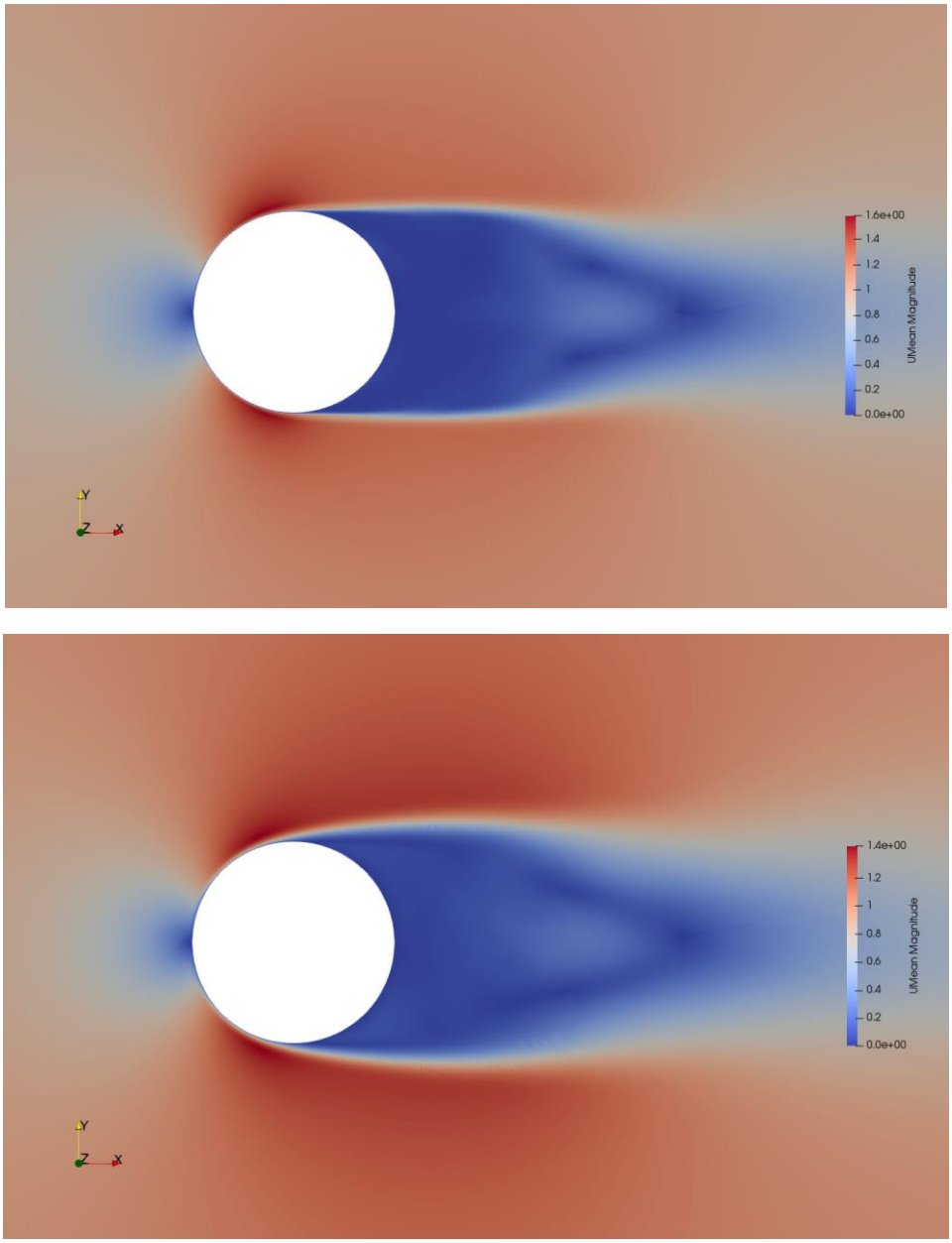


Figure 39 Comparison of mean velocity field for Re=50 000 with fully turbulent boundary layer (top) and transition model (bottom).

For the lower Reynolds numbers, the difference between the two models are smaller. The reason for this can be explained by looking at the intermittency values. Figure 40 compares intermittency values for Reynolds numbers 3900 and 50 000. For Re=3900, the intermittency value is low, meaning the boundary layer is laminar. For Re=50 000, however, there is a transition from laminar to turbulent flow at around the top/bottom of the cylinder.

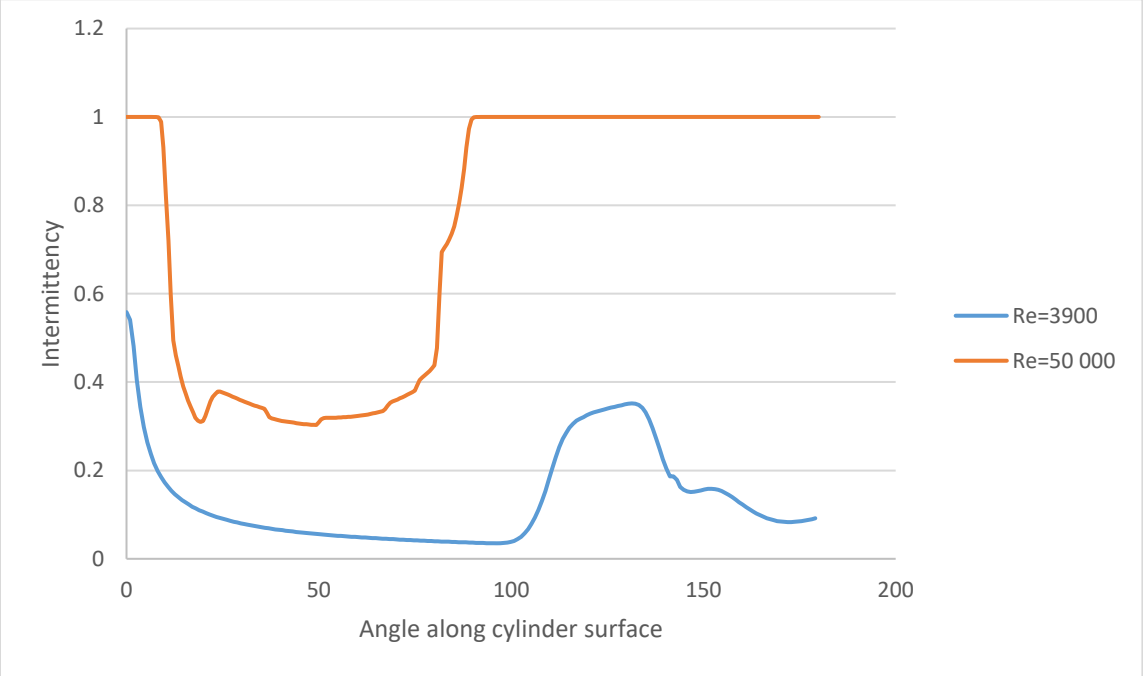


Figure 40 Intermittency values along the cylinder

Following figures compare the velocity magnitude in the boundary layer at a 45 degree angle from the front of the cylinder. We see that for the lower Reynolds number the difference between the transitional and fully turbulent model is not as pronounced as for the higher Reynolds number. For the higher Reynolds number the higher turbulence in the fully turbulent model leads to separation at a later angle.

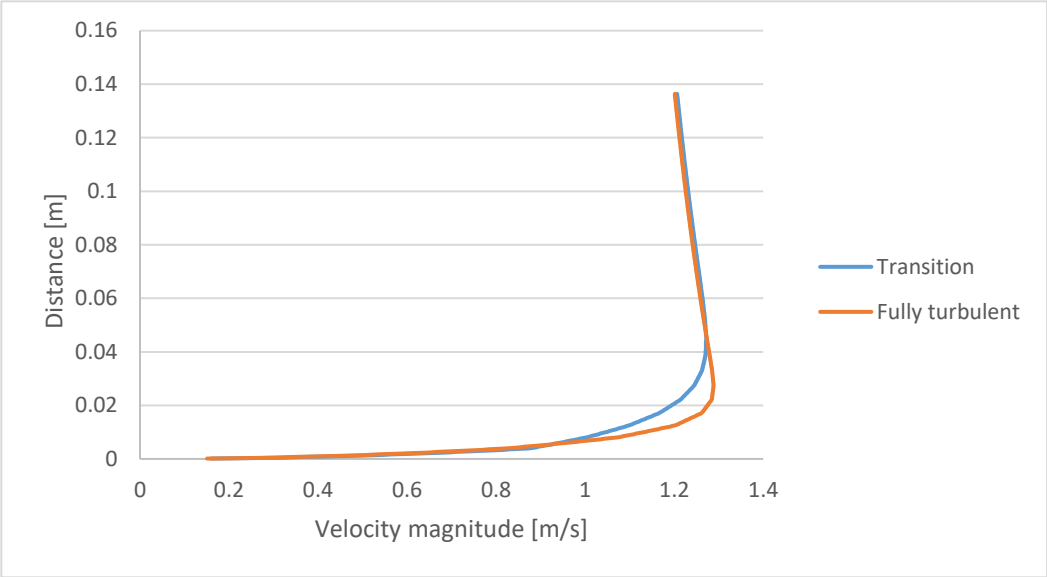


Figure 41 Comparison between transitional and fully turbulent model at Re=3900

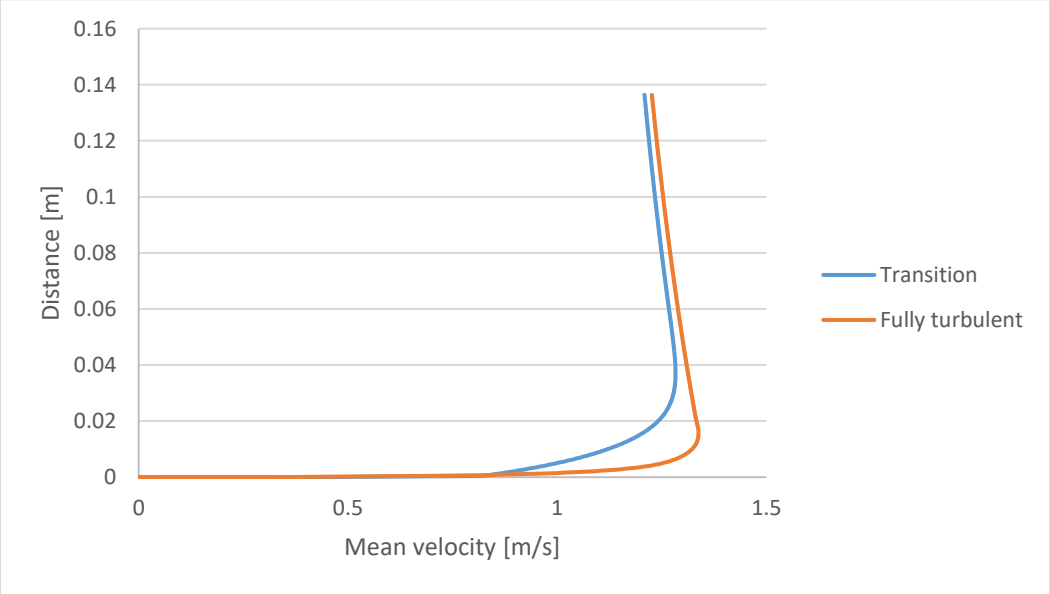


Figure 42 Comparison between transitional and fully turbulent model at Re=50000

8 Conclusion

The current thesis performed an investigation of a transitional turbulence model to predict drag crisis for flow over bluff bodies. Sphere and cylinder has been chosen as an example for bluff bodies. An experiment has been done in the wind tunnel for flow over the sphere, as it has been chosen as the main geometry in this study. The experiment was performed on sphere with smooth surface, and also while a bumper has been made on the surface in order to do a modification on the surface. The results have been compared and presented in a diagram. Additionally, surface visualization shows a good picture of how flow behave on the surface of a buff body.

Flow over circular cylinder and sphere has been simulated for several Reynolds numbers covering the laminar and transitional flow regimes in the boundary layer. The turbulence model used was DES with k-ω SST as RANS model. For transition modelling, the Langtry-Menter model was used. Simulation for the cylinder has been performed for three Reynolds numbers, 3900, 10 000 and 50 000 in both models. For the sphere, seven cases have been defined with different Reynolds numbers, 10^4, 4×10^4, 10^5, 2×10^5, 2.5×10^5, 5×10^5 and 10^6, and were simulated with both models.

The computed drag coefficient was similar in both models for cylinder. However, at the highest Reynolds number, the computed results with transitional model were closer to experimental results compared to the fully turbulent model.

In the sphere case, there is a small difference between the two models at the low Reynolds numbers and, similar to the cylinder, the difference grew as the Reynolds number increased. However, the simulation cannot converge at the higher Reynolds numbers.

For further work, simulations for the higher Reynolds numbers in the transitional model should be performed for the sphere case to investigate the reason why it doesn't converge and to create a better plot in the supercritical range. Moreover, for the cylinder, case simulations at higher Reynolds numbers due to the high computational time required was not feasible in the current work. Therefore, more simulations should be performed to investigate the full range of flow

behavior during the drag crisis and into the post-critical domain. Investigation into other transitional models can also help create a better plot that covers full range of the flow regime.

9 References

[1] R. Mittal and P. Moin, "suitablity of upwind biased schemes for large- eddy simulation," *AIAA Journal,* vol. 30(8), pp. 1415-1417, 1997.

[2] M. Breuer, "Large eddy simulation of the sub-critical flow past a circular cylinder," *numerical and modeling aspects. Int. J. Numer. Methods Fluid,* vol. 28, pp. 1281-1302, 1998.

[3] J. Frank and W. Frank, "Large eddy simulation of the flow past a circular cylinder at Re=3900.," *J. Wind Eng. Ind. Aerod.,* vol. 90, pp. 1191-1206, 2002.

[4] N. Alkishriwi, M. Meinke and W. Schroder, "A large-eddy simulation method for low Mach number flows using preconditioning and multigrid," *J. Copmut. Fluids,* vol. 35, pp. 1126-1136, 2006.

[5] M. Meyer, S. Hickel and N. A. Adams, "Assessment of implicit large eddy simulation with a conservative immersed interface method for turbulent cylinder flow," *Int. J. Heat Fluid Flow,* vol. 31, pp. 368-377, 2010.

[6] A. kravchenko and P. Moin, "Numerical studies of flow over a circular cylinder at Re=3900," *Phys. Fluids,* vol. 12, pp. 403-417, 2000.

[7] P. Parnaudeau, J. Carlier, D. Heitz and E. Lamballais, "Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900," *Phys.Fluids,* vol. 8, p. 20, 2008.

[8] A. Mani, P. Moin and M. Wang, "Computational study of optical distortions by seperated shear layers and turbulent wakes," *J. Fluid Mech,* vol. 625, pp. 273-298, 2009.

[9] P. Catalano, M. Wang, G. Iaccarino and P. Moin, "Numerical simulation of the flow around a circular cylinder at high Reynolds numbers," *International journal of heat and fluid flow,* vol. 24, pp. 463-469, 2003.

[10] G. Constantinescu and K. Squires, "Numerical investigations of flow over a sphere in the subcritical and supercritical regimes," *Phys. Fluids,* vol. 16, no. 2004, pp. 1449-1466, 2003.

[11] B. Dean and B. Bhushan, "Shark-skin surfaces for drag reduction in turbulent flow," *Philosophical Transactions of Royal society,* pp. 4775-4806, 2010.

[12] J. Choi, W.-P. Jeon and H. Choi, "Mechanism of drag reduction by dimples on a sphere," *Physics of Fluids,* vol. 18, no. 2006, 2006.

[13] H. Park, D. Lee, W.-P. Jeon, S. Hahn, J. Kim, J. Kim, J. Choi and H. Choi, "Drag reduction in flow over two dimensional bluff body with a blunt trailing edge using a new passive device," *Fluid Mechanics,* vol. 563, pp. 389-414, 2006.

[14] R. J. Daniello, N. E. Waterhouse and J. P. Rothstein, "Drag reduction in turbulent flows over superhydrophobic surfaces," *Physics of fluids,* vol. 21, 2009.

[15] R. B. Langtry and F. R. Menter, "Correlation-Based transition modelling for unstructured parallelized computational fluid dynamics codes," *AIAA Journal,* vol. 47, pp. 2894-2906, 2009.

[16] Y. A. Cengel and J. M. Cimbala, Fluid Mechanics: Fundamental and applications, NewYork: McGraw-Hill, 2006.

[17] E. Achenbach, "Experiments on the flow past spheres at very high Reynolds numbers," *J. Fluid Mech,* vol. 54, 1972.

[18] J. M. Cimbala and Y. A. Cengel, Fluid Mechanics, New York: MacGraw-Hill, 2006.

[19] T. Von Karman, "Mechanical similitude and Turbulence," *Tech.Mem. NACA,* vol. 5, pp. 58-76, 1931.

[20] F. M. White, Fluid Mechanics 7th ed., NewYork: McGraw Hill Higher Education, 2011.

[21] M. C. Ong, T. Utnes, L. E. Holmedal, D. Myrhaug and B. Pettersen, "Numerical simulation of flow around a smooth circular cylinder at very high Reynolds numbers," *Marine structures,* vol. 22, pp. 142-153, 2009.

[22] D. Lindblad, "Implementation and run-time mesh refinement for the k-Omega SST DES turbulence model when applied to airfoils," 2014.

[23] P. Malan, K. Suluksna and E. Juntasaro, "Calibrating the gamma-Re_theta tarnsition model for commercial CFD," *AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition,* vol. 47th, p. 1142, 2009.

[24] A. M. Savill, "Some Recent progress in the Turbulent Modeling of By-Pass Transition, Near-Wall Turbulent Flows,," *Elsevier,* p. 829, 1993.

[25] A. M. Savill, "One-Point Closures Applied to Transition, Turbulence and Transition Modeling," *The Netherlands,* pp. 233-268, 1996.

[26] M. Lopez and D. K. Walters, "A recommended correction to the Kt-Kl-Omega transition sensitive eddy-viscosity model," *Journal of Fluids Engineering,* vol. 139(2), p. 024501, 2017.

[27] R. Langtry, "Extending the Gamma-Retheta Correlation Based Transition Model for Crossflow Effects," in *45th AIAA fluid dynamics conference*, 2015.

[28] C. Wieselsberger, "New data on the laws of fluid resistance," National advisory committe for Aeronautics, 1922.

[29] T. Maxworthy, "Experiments on the flow around a sphere at high Reynolds numbers," *Journal of Applied Mechanics,* pp. 598-607, 1969.

[30] G. S. Cardell, "Flow past a circular cylinder with permeable splitter plate," 1993.

[31] R. Mittal, "Progress on LES of flow past a circular cylinder," *Annual Research Briefs, Center of Turbulence Research,* 1996.

[32] L. M. Lourenco and C. Shih, "Characteristics of the plane turbulent near wake of a circular cylinder, a particle image velocimetry study," 1993.

[33] C. Norberg, "Experimental investigation of the flow around the circular cylinder," *Influence of aspect ratio. J. Fluid Mech,* vol. 258, pp. 287-316, 1994.

[34] C. Norberg, "Flow around circular cylinder," *Aspect of fluctuating lift. J. Fluids Struct.,* vol. 20, pp. 459-469, 2001.

[35] L. Ong and J. Wallace, "The velocity field of the turbulent very near wake of a circular cylinder," *Exp. Fluids,* vol. 20, pp. 441-453, 1996.

[36] X. Ma and G. S. Karamanos, "Dynamics and low-dimensionality of a turbulent near wake," *J.Fluid Mech,* vol. 410, pp. 29-65, 2000.

[37] H. Ourvard, B. Koobus, A. Dervieux and M. V. Salvetti, "Classical and Variational multiscale LES of the flow around a circular cylinder on unstructured grids," *Comput. Fluids,* vol. 39, pp. 1083-1094, 2010.

[38] S. Wornom, H. Ourvard, M. V. Salvetti, B. Koobus and A. Dervieux, "Variational multiscale large-eddy simulations of the flow past a circular cylinder, Reynolds number effects," *Comput. Fluids,* vol. 47(1), pp. 44-50, 2011.

[39] D. A. Lysenko, I. S. Ertesåg and K. E. Rian, "Large-Eddy simulation of the flow over a circular cylinder at Reynolds number 3900 using the OpenFOAM toolbox," *Flow Turbulence Combust,* vol. 89, pp. 492-517, 2012.

[40] W. C. L. shih, C. Wang, D. Coles and A. Roshko, "Experiments on flow past rough circular cylinders at large Reynolds numbers," *Wind Eng. Indust. Aerodyn.,* vol. 49, pp. 351-368, 1993.

[41] S. Singh and S. Mittal, "Flow past a cylinder, Shear layer instability and drag crisis," *Int. J. Numer Meth Fluids,* vol. 47, pp. 75-98, 2005.

[42] D. Årrestad Stave, "Cyclist posture optimisation using CFD," Universitet i stavanger, Stavanger, 2018.

[43] H. K. Versteeg and W. Malalasekera, An introduction to CFD finite volume method, Glasgow: Pearson Education, 2007.

[44] E. D. Robertson, "Verification, validation and implementation of numerical methods and models for OpenFOAM 2.0 for incompressible flow," Mississippi State University, Mississippi, 2015.

Appendix

Contents of Appendix

A: Source code for kOmegaSSTLMDES

- Header file - kOmegaSSTDESLM.H
- Source file – kOmegaSSTLMDES.C

Appendix A – source code for kOmegaSSTLMDES

Header file - kOmegaSSTDESLM.H

```
#ifndef kOmegaSSTLMDES_H
#define kOmegaSSTLMDES_H

#include "LESeddyViscosity.H"


namespace Foam
{
namespace LESModels
{

/*---------------------------------------------------------------------------*\
                        Class kOmegaSSTLMDES Declaration
\*---------------------------------------------------------------------------*/

template<class BasicTurbulenceModel>
class kOmegaSSTLMDES
:
    public LESeddyViscosity<BasicTurbulenceModel>
{
    // Private Member Functions

        // Disallow default bitwise copy construct and assignment
        kOmegaSSTLMDES(const kOmegaSSTLMDES&);
        void operator=(const kOmegaSSTLMDES&);

protected:

    // Protected data

        // Model constants

            //- DES coefficient
            dimensionedScalar CDES_;

            //- Zonal filter choice
            //
            //  - 0: no filtering
            //  - 1: (1 - F1)
            //  - 2: (1 - F2)
            direction FSST_;

            // From kOmegaSSTLM
            dimensionedScalar ca1_;
            dimensionedScalar ca2_;

            dimensionedScalar ce1_;
            dimensionedScalar ce2_;

            dimensionedScalar cThetat_;
            dimensionedScalar sigmaThetat_;

            //- Convergence criterion for the lambda/thetat loop
```

```
            scalar lambdaErr_;

            //- Maximum number of iterations to converge the lambda/thetat loop
            label maxLambdaIter_;

            //- Stabilization for division by the magnitude of the velocity
            dimensionedScalar deltaU_;

// Fields

            // from kOmegaSSTLM

            //- Transition onset momentum-thickness Reynolds number
            volScalarField ReThetat_;

            //- Intermittency
            volScalarField gammaInt_;

            //- Effective intermittency
            volScalarField::Internal gammaIntEff_;



            // From kOmegaSST

            dimensionedScalar alphaK1_;
            dimensionedScalar alphaK2_;

            dimensionedScalar alphaOmega1_;
            dimensionedScalar alphaOmega2_;

            dimensionedScalar gamma1_;
            dimensionedScalar gamma2_;

            dimensionedScalar beta1_;
            dimensionedScalar beta2_;

            dimensionedScalar betaStar_;

            dimensionedScalar a1_;
            dimensionedScalar b1_;
            dimensionedScalar c1_;

            Switch F3_;

// Fields
            // from kOmegaSST

            //- Wall distance
            //  Note: different to wall distance in parent RASModel
            //  which is for near-wall cells only
            const volScalarField& y_;

            volScalarField k_;
            volScalarField omega_;


    // Protected Member Functions

        //- Return the turbulent length-scale
        tmp<volScalarField::Internal> Lt() const;

        //- The DES dissipation-rate multiplier with options zonal filtering
```

```
//  based on either F1 or F2
virtual tmp<volScalarField::Internal> FDES
(
    const volScalarField::Internal& F1,
    const volScalarField::Internal& F2
) const;

//- Return epsilon/k which for standard RAS is betaStar*omega
virtual tmp<volScalarField::Internal> epsilonByk
(
    const volScalarField::Internal& F1,
    const volScalarField::Internal& F2
) const;


//- Modified form of the k-omega SST k production rate
virtual tmp<volScalarField::Internal> Pk
(
    const volScalarField::Internal& G
) const;


//- Freestream blending-function
tmp<volScalarField::Internal> Fthetat
(
    const volScalarField::Internal& Us,
    const volScalarField::Internal& Omega,
    const volScalarField::Internal& nu
) const;

//- Empirical correlation for critical Reynolds number where the
//  intermittency first starts to increase in the boundary layer
tmp<volScalarField::Internal> ReThetac() const;

//- Empirical correlation that controls the length of the
//  transition region
tmp<volScalarField::Internal> Flength
(
    const volScalarField::Internal& nu
) const;

//- Transition onset location control function
tmp<volScalarField::Internal> Fonset
(
    const volScalarField::Internal& Rev,
    const volScalarField::Internal& ReThetac,
    const volScalarField::Internal& RT
) const;

//- Return the transition onset momentum-thickness Reynolds number
// (based on freestream conditions)
tmp<volScalarField::Internal> ReThetat0
(
    const volScalarField::Internal& Us,
    const volScalarField::Internal& dUsds,
    const volScalarField::Internal& nu
) const;

//- Solve the turbulence equations and correct the turbulence viscosity
void correctReThetatGammaInt();

// from kOmegaSST
```

```
//- Modified form of the k-omega SST F1 function
virtual tmp<volScalarField> F1(const volScalarField& CDkOmega) const;
virtual tmp<volScalarField> F2() const;
virtual tmp<volScalarField> F3() const;
virtual tmp<volScalarField> F23() const;

tmp<volScalarField> blend
(
    const volScalarField& F1,
    const dimensionedScalar& psi1,
    const dimensionedScalar& psi2
) const
{
    return F1*(psi1 - psi2) + psi2;
}

tmp<volScalarField::Internal> blend
(
    const volScalarField::Internal& F1,
    const dimensionedScalar& psi1,
    const dimensionedScalar& psi2
) const
{
    return F1*(psi1 - psi2) + psi2;
}

tmp<volScalarField> alphaK(const volScalarField& F1) const
{
    return blend(F1, alphaK1_, alphaK2_);
}

tmp<volScalarField> alphaOmega(const volScalarField& F1) const
{
    return blend(F1, alphaOmega1_, alphaOmega2_);
}

tmp<volScalarField::Internal> beta
(
    const volScalarField::Internal& F1
) const
{
    return blend(F1, beta1_, beta2_);
}

tmp<volScalarField::Internal> gamma
(
    const volScalarField::Internal& F1
) const
{
    return blend(F1, gamma1_, gamma2_);
}

virtual void correctNut
(
    const volScalarField& S2,
    const volScalarField& F2
);

virtual void correctNut();

virtual tmp<fvScalarMatrix> kSource() const;

virtual tmp<fvScalarMatrix> omegaSource() const;
```

```
        virtual tmp<fvScalarMatrix> Qsas
        (
            const volScalarField::Internal& S2,
            const volScalarField::Internal& gamma,
            const volScalarField::Internal& beta
        ) const;


public:

    typedef typename BasicTurbulenceModel::alphaField alphaField;
    typedef typename BasicTurbulenceModel::rhoField rhoField;
    typedef typename BasicTurbulenceModel::transportModel transportModel;


    //- Runtime type information
    TypeName("kOmegaSSTLMDES");


    // Constructors

        //- Construct from components
        kOmegaSSTLMDES
        (
            const alphaField& alpha,
            const rhoField& rho,
            const volVectorField& U,
            const surfaceScalarField& alphaRhoPhi,
            const surfaceScalarField& phi,
            const transportModel& transport,
            const word& propertiesName = turbulenceModel::propertiesName,
            const word& type = typeName
        );


    //- Destructor
    virtual ~kOmegaSSTLMDES()
    {}


    // Member Functions

        //- Read model coefficients if they have changed
        virtual bool read();

        //- Access function transition onset momentum-thickness Reynolds number
        const volScalarField& ReThetat() const
        {
            return ReThetat_;
        }

        //- Access function to intermittency
        const volScalarField& gammaInt() const
        {
            return gammaInt_;
        }

        //- Return the effective diffusivity for transition onset
        //  momentum-thickness Reynolds number
        tmp<volScalarField> DReThetatEff() const
        {
```

```
    return tmp<volScalarField>
    (
        new volScalarField
        (
            "DReThetatEff",
            sigmaThetat_*(this->nut_ + this->nu())
        )
    );
}

//- Return the effective diffusivity for intermittency
tmp<volScalarField> DgammaIntEff() const
{
    return tmp<volScalarField>
    (
        new volScalarField
        (
            "DgammaIntEff",
            this->nut_ + this->nu()
        )
    );
}

 //- Return the effective diffusivity for k
tmp<volScalarField> DkEff(const volScalarField& F1) const
{
    return tmp<volScalarField>
    (
        new volScalarField("DkEff", alphaK(F1)*this->nut_ + this->nu())
    );
}

//- Return the effective diffusivity for omega
tmp<volScalarField> DomegaEff(const volScalarField& F1) const
{
    return tmp<volScalarField>
    (
        new volScalarField
        (
            "DomegaEff",
            alphaOmega(F1)*this->nut_ + this->nu()
        )
    );
}

//- Return the turbulence kinetic energy
virtual tmp<volScalarField> k() const
{
    return k_;
}

//- Return the turbulence kinetic energy dissipation rate
virtual tmp<volScalarField> epsilon() const
{
      return tmp<volScalarField>
    (
        new volScalarField
        (
            IOobject
            (
                "epsilon",
                this->mesh_.time().timeName(),
                this->mesh_
```

```
                    ),
                    betaStar_*k_*omega_,
                    omega_.boundaryField().types()
                )
            );
        }

        //- Return the turbulence kinetic energy dissipation rate
        virtual tmp<volScalarField> omega() const
        {
            return omega_;
        }


        //- Correct nuTilda and related properties
        virtual void correct();
};



} // End namespace LESModels
} // End namespace Foam

#ifdef NoRepository
    #include "kOmegaSSTLMDES.C"
#endif

#endif
```

Source file – kOmegaSSTLMDES.C

```
#include "kOmegaSSTLMDES.H"
#include "fvOptions.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
namespace LESModels
{

// * * * * * * * * * * * * * Protected Member Functions  * * * * * * * * * * * * * //

// From kOmegaSST
template<class BasicTurbulenceModel>
tmp<volScalarField>
kOmegaSSTLMDES<BasicTurbulenceModel>::F1
(
    const volScalarField& CDkOmega
) const
{
    const volScalarField Ry(this->y_*sqrt(this->k_)/this->nu());
    const volScalarField F3(exp(-pow(Ry/120.0, 8)));


    tmp<volScalarField> CDkOmegaPlus = max
    (
        CDkOmega,
        dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
    );

    tmp<volScalarField> arg1 = min
    (
        min
        (
            max
            (
                (scalar(1)/betaStar_)*sqrt(k_)/(omega_*y_),
                scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omega_)
            ),
            (4*alphaOmega2_)*k_/(CDkOmegaPlus*sqr(y_))
        ),
        scalar(10)
    );
                // Old F1       , new)
    return max(tanh(pow4(arg1)), F3);
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
kOmegaSSTLMDES<BasicTurbulenceModel>::F2() const
{
    tmp<volScalarField> arg2 = min
    (
        max
        (
            (scalar(2)/betaStar_)*sqrt(k_)/(omega_*y_),
            scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omega_)
        ),
        scalar(100)
```

```
    );

    return tanh(sqr(arg2));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
kOmegaSSTLMDES<BasicTurbulenceModel>::F3() const
{
    tmp<volScalarField> arg3 = min
    (
        150*(this->mu()/this->rho_)/(omega_*sqr(y_)),
        scalar(10)
    );

    return 1 - tanh(pow4(arg3));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
kOmegaSSTLMDES<BasicTurbulenceModel>::F23() const
{
    tmp<volScalarField> f23(F2());

    if (F3_)
    {
        f23.ref() *= F3();
    }

    return f23;
}


template<class BasicTurbulenceModel>
void kOmegaSSTLMDES<BasicTurbulenceModel>::correctNut
(
    const volScalarField& S2,
    const volScalarField& F2
)
{
    this->nut_ = a1_*k_/max(a1_*omega_, b1_*F2*sqrt(S2));
    this->nut_.correctBoundaryConditions();
    fv::options::New(this->mesh_).correct(this->nut_);

    BasicTurbulenceModel::correctNut();
}

template<class BasicTurbulenceModel>
void kOmegaSSTLMDES<BasicTurbulenceModel>::correctNut()
{
    correctNut(2*magSqr(symm(fvc::grad(this->U_))), F23());
}


template<class BasicTurbulenceModel>
tmp<fvScalarMatrix>
kOmegaSSTLMDES<BasicTurbulenceModel>::kSource() const
{
    return tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            k_,
```

```
            dimVolume*this->rho_.dimensions()*k_.dimensions()/dimTime
        )
    );
}


template<class BasicTurbulenceModel>
tmp<fvScalarMatrix>
kOmegaSSTLMDES<BasicTurbulenceModel>::omegaSource() const
{
    return tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            omega_,
            dimVolume*this->rho_.dimensions()*omega_.dimensions()/dimTime
        )
    );
}


template<class BasicTurbulenceModel>
tmp<fvScalarMatrix> kOmegaSSTLMDES<BasicTurbulenceModel>::Qsas
(
    const volScalarField::Internal& S2,
    const volScalarField::Internal& gamma,
    const volScalarField::Internal& beta
) const
{
    return tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            omega_,
            dimVolume*this->rho_.dimensions()*omega_.dimensions()/dimTime
        )
    );
}



template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::Lt() const
{
    return sqrt(this->k_())/(this->betaStar_*this->omega_());
}


template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::FDES
(
    const volScalarField::Internal& F1,
    const volScalarField::Internal& F2
) const
{
    switch (FSST_)
    {
        case 0:
            return max(Lt()/(CDES_*this->delta()()), scalar(1));
        case 1:
            return max(Lt()*(1 - F1)/(CDES_*this->delta()()), scalar(1));
        case 2:
            return max(Lt()*(1 - F2)/(CDES_*this->delta()()), scalar(1));
```

```
        default:
            FatalErrorInFunction
                << "Incorrect FSST = " << FSST_ << ", should be 0, 1 or 2"
                << exit(FatalError);
            return F1;
    }
}


template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::epsilonByk
(
    const volScalarField::Internal& F1,
    const volScalarField::Internal& F2
) const
{
    return
        min(max(gammaIntEff_, scalar(0.1)), scalar(1))*
        this->betaStar_*this->omega_()*FDES(F1, F2);
}

// From kOmegaSSTLM

template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::Pk
(
    const volScalarField::Internal& G
) const
{
    return gammaIntEff_*min(G, (c1_*betaStar_)*this->k_()*this->omega_());
}


template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::Fthetat
(
    const volScalarField::Internal& Us,
    const volScalarField::Internal& Omega,
    const volScalarField::Internal& nu
) const
{
    const volScalarField::Internal& omega = this->omega_();
    const volScalarField::Internal& y = this->y_();

    const volScalarField::Internal delta(375*Omega*nu*ReThetat_()*y/sqr(Us));
    const volScalarField::Internal ReOmega(sqr(y)*omega/nu);
    const volScalarField::Internal Fwake(exp(-sqr(ReOmega/1e5)));

    return tmp<volScalarField::Internal>
    (
        new volScalarField::Internal
        (
            IOobject::groupName("Fthetat", this->U_.group()),
            min
            (
                max
                (
                    Fwake*exp(-pow4((y/delta))),
                    (1 - sqr((gammaInt_() - 1.0/ce2_)/(1 - 1.0/ce2_)))
                ),
                scalar(1)
            )
        )
```

```
    );
}


template<class BasicTurbulenceModel>
tmp<volScalarField::Internal>
kOmegaSSTLMDES<BasicTurbulenceModel>::ReThetac() const
{
    tmp<volScalarField::Internal> tReThetac
    (
        new volScalarField::Internal
        (
            IOobject
            (
                IOobject::groupName("ReThetac", this->U_.group()),
                this->runTime_.timeName(),
                this->mesh_
            ),
            this->mesh_,
            dimless
        )
    );
    volScalarField::Internal& ReThetac = tReThetac.ref();

    forAll(ReThetac, celli)
    {
        const scalar ReThetat = ReThetat_[celli];

        ReThetac[celli] =
            ReThetat <= 1870
          ?
            ReThetat
          - 396.035e-2
          + 120.656e-4*ReThetat
          - 868.230e-6*sqr(ReThetat)
          + 696.506e-9*pow3(ReThetat)
          - 174.105e-12*pow4(ReThetat)
          :
            ReThetat - 593.11 - 0.482*(ReThetat - 1870);
    }

    return tReThetac;
}


template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::Flength
(
    const volScalarField::Internal& nu
) const
{
    tmp<volScalarField::Internal> tFlength
    (
        new volScalarField::Internal
        (
            IOobject
            (
                IOobject::groupName("Flength", this->U_.group()),
                this->runTime_.timeName(),
                this->mesh_
            ),
            this->mesh_,
            dimless
```

```
        )
    );
    volScalarField::Internal& Flength = tFlength.ref();

    const volScalarField::Internal& omega = this->omega_();
    const volScalarField::Internal& y = this->y_();

    forAll(ReThetat_, celli)
    {
        const scalar ReThetat = ReThetat_[celli];

        if (ReThetat < 400)
        {
            Flength[celli] =
                398.189e-1
              - 119.270e-4*ReThetat
              - 132.567e-6*sqr(ReThetat);
        }
        else if (ReThetat < 596)
        {
            Flength[celli] =
                263.404
              - 123.939e-2*ReThetat
              + 194.548e-5*sqr(ReThetat)
              - 101.695e-8*pow3(ReThetat);
        }
        else if (ReThetat < 1200)
        {
            Flength[celli] = 0.5 - 3e-4*(ReThetat - 596);
        }
        else
        {
            Flength[celli] = 0.3188;
        }

        const scalar Fsublayer =
            exp(-sqr(sqr(y[celli])*omega[celli]/(200*nu[celli])));

        Flength[celli] = Flength[celli]*(1 - Fsublayer) + 40*Fsublayer;
    }

    return tFlength;
}


template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::ReThetat0
(
    const volScalarField::Internal& Us,
    const volScalarField::Internal& dUsds,
    const volScalarField::Internal& nu
) const
{
    tmp<volScalarField::Internal> tReThetat0
    (
        new volScalarField::Internal
        (
            IOobject
            (
                IOobject::groupName("ReThetat0", this->U_.group()),
                this->runTime_.timeName(),
                this->mesh_
            ),
```

```
        this->mesh_,
        dimless
    )
);
volScalarField::Internal& ReThetat0 = tReThetat0.ref();

const volScalarField& k = this->k_;

label maxIter = 0;

forAll(ReThetat0, celli)
{
    const scalar Tu
    (
        max(100*sqrt((2.0/3.0)*k[celli])/Us[celli], scalar(0.027))
    );

    // Initialize lambda to zero.
    // If lambda were cached between time-steps convergence would be faster
    // starting from the previous time-step value.
    scalar lambda = 0;

    scalar lambdaErr;
    scalar thetat;
    label iter = 0;

    do
    {
        // Previous iteration lambda for convergence test
        const scalar lambda0 = lambda;

        if (Tu <= 1.3)
        {
            const scalar Flambda =
                dUsds[celli] <= 0
              ?
                1
              - (
                 - 12.986*lambda
                 - 123.66*sqr(lambda)
                 - 405.689*pow3(lambda)
                )*exp(-pow(Tu/1.5, 1.5))
              :
                1
              + 0.275*(1 - exp(-35*lambda))
               *exp(-Tu/0.5);

            thetat =
                (1173.51 - 589.428*Tu + 0.2196/sqr(Tu))
               *Flambda*nu[celli]
               /Us[celli];
        }
        else
        {
            const scalar Flambda =
                dUsds[celli] <= 0
              ?
                1
              - (
                  -12.986*lambda
                  -123.66*sqr(lambda)
                  -405.689*pow3(lambda)
                )*exp(-pow(Tu/1.5, 1.5))
```

```
                    :
                      1
                    + 0.275*(1 - exp(-35*lambda))
                     *exp(-2*Tu);

                thetat =
                    331.50*pow((Tu - 0.5658), -0.671)
                   *Flambda*nu[celli]/Us[celli];
            }

            lambda = sqr(thetat)/nu[celli]*dUsds[celli];
            lambda = max(min(lambda, 0.1), -0.1);

            lambdaErr = mag(lambda - lambda0);

            maxIter = max(maxIter, ++iter);

        } while (lambdaErr > lambdaErr_);

        ReThetat0[celli] = max(thetat*Us[celli]/nu[celli], scalar(20));
    }

    if (maxIter > maxLambdaIter_)
    {
        WarningInFunction
            << "Number of lambda iterations exceeds maxLambdaIter("
            << maxLambdaIter_ << ')'<< endl;
    }

    return tReThetat0;
}


template<class BasicTurbulenceModel>
tmp<volScalarField::Internal> kOmegaSSTLMDES<BasicTurbulenceModel>::Fonset
(
    const volScalarField::Internal& Rev,
    const volScalarField::Internal& ReThetac,
    const volScalarField::Internal& RT
) const
{
    const volScalarField::Internal Fonset1(Rev/(2.193*ReThetac));

    const volScalarField::Internal Fonset2
    (
        min(max(Fonset1, pow4(Fonset1)), scalar(2))
    );

    const volScalarField::Internal Fonset3(max(1 - pow3(RT/2.5), scalar(0)));

    return tmp<volScalarField::Internal>
    (
        new volScalarField::Internal
        (
            IOobject::groupName("Fonset", this->U_.group()),
            max(Fonset2 - Fonset3, scalar(0))
        )
    );
}
```

```
// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * //

template<class BasicTurbulenceModel>
kOmegaSSTLMDES<BasicTurbulenceModel>::kOmegaSSTLMDES
(
    const alphaField& alpha,
    const rhoField& rho,
    const volVectorField& U,
    const surfaceScalarField& alphaRhoPhi,
    const surfaceScalarField& phi,
    const transportModel& transport,
    const word& propertiesName,
    const word& type
)
:
    LESeddyViscosity<BasicTurbulenceModel>
    (
        type,
        alpha,
        rho,
        U,
        alphaRhoPhi,
        phi,
        transport,
        propertiesName
    ),

    CDES_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "CDES",
            this->coeffDict_,
            0.61
        )
    ),
    FSST_(this->coeffDict_.lookupOrDefault("FSST", 2)),
     ca1_
    (
        dimensionedScalar::lookupOrAddToDict
        (
            "ca1",
            this->coeffDict_,
            2
        )
    ),
    ca2_
    (
        dimensionedScalar::lookupOrAddToDict
        (
            "ca2",
            this->coeffDict_,
            0.06
        )
    ),
    ce1_
    (
        dimensionedScalar::lookupOrAddToDict
        (
            "ce1",
            this->coeffDict_,
            1
        )
```

```
),
ce2_
(
    dimensionedScalar::lookupOrAddToDict
    (
        "ce2",
        this->coeffDict_,
        50
    )
),
cThetat_
(
    dimensionedScalar::lookupOrAddToDict
    (
        "cThetat",
        this->coeffDict_,
        0.03
    )
),
sigmaThetat_
(
    dimensionedScalar::lookupOrAddToDict
    (
        "sigmaThetat",
        this->coeffDict_,
        2
    )
),
lambdaErr_
(
    this->coeffDict_.lookupOrDefault("lambdaErr", 1e-6)
),
maxLambdaIter_
(
    this->coeffDict_.lookupOrDefault("maxLambdaIter", 10)
),
deltaU_("deltaU", dimVelocity, SMALL),

ReThetat_
(
    IOobject
    (
        IOobject::groupName("ReThetat", U.group()),
        this->runTime_.timeName(),
        this->mesh_,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    this->mesh_
),

gammaInt_
(
    IOobject
    (
        IOobject::groupName("gammaInt", U.group()),
        this->runTime_.timeName(),
        this->mesh_,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    this->mesh_
),
```

```
gammaIntEff_
(
    IOobject
    (
        IOobject::groupName("gammaIntEff", U.group()),
        this->runTime_.timeName(),
        this->mesh_
    ),
    this->mesh_,
    dimensionedScalar("0", dimless, 0)
),
alphaK1_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "alphaK1",
        this->coeffDict_,
        0.85
    )
),
alphaK2_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "alphaK2",
        this->coeffDict_,
        1.0
    )
),
alphaOmega1_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "alphaOmega1",
        this->coeffDict_,
        0.5
    )
),
alphaOmega2_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "alphaOmega2",
        this->coeffDict_,
        0.856
    )
),
gamma1_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "gamma1",
        this->coeffDict_,
        5.0/9.0
    )
),
gamma2_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "gamma2",
        this->coeffDict_,
```

```
        0.44
    )
),
beta1_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "beta1",
        this->coeffDict_,
        0.075
    )
),
beta2_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "beta2",
        this->coeffDict_,
        0.0828
    )
),
betaStar_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "betaStar",
        this->coeffDict_,
        0.09
    )
),
a1_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "a1",
        this->coeffDict_,
        0.31
    )
),
b1_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "b1",
        this->coeffDict_,
        1.0
    )
),
c1_
(
    dimensioned<scalar>::lookupOrAddToDict
    (
        "c1",
        this->coeffDict_,
        10.0
    )
),
F3_
(
    Switch::lookupOrAddToDict
    (
        "F3",
        this->coeffDict_,
```

```
            false
        )
    ),

    y_(wallDist::New(this->mesh_).y()),

    k_
    (
        IOobject
        (
            IOobject::groupName("k", U.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),
    omega_
    (
        IOobject
        (
            IOobject::groupName("omega", U.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    )




{
    bound(k_, this->kMin_);
    bound(omega_, this->omegaMin_);
    if (type == typeName)
    {
        this->printCoeffs(type);
    }
    Info << "fuck";
}


// * * * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

template<class BasicTurbulenceModel>
bool kOmegaSSTLMDES<BasicTurbulenceModel>::read()
{
    if (LESeddyViscosity<BasicTurbulenceModel>::read())
    {
        // kOmegaSSTLM
        ca1_.readIfPresent(this->coeffDict());
        ca2_.readIfPresent(this->coeffDict());
        ce1_.readIfPresent(this->coeffDict());
        ce2_.readIfPresent(this->coeffDict());
        sigmaThetat_.readIfPresent(this->coeffDict());
        cThetat_.readIfPresent(this->coeffDict());
        this->coeffDict().readIfPresent("lambdaErr", lambdaErr_);
        this->coeffDict().readIfPresent("maxLambdaIter", maxLambdaIter_);

        // kOmegaSST
```

```
        alphaK1_.readIfPresent(this->coeffDict());
        alphaK2_.readIfPresent(this->coeffDict());
        alphaOmega1_.readIfPresent(this->coeffDict());
        alphaOmega2_.readIfPresent(this->coeffDict());
        gamma1_.readIfPresent(this->coeffDict());
        gamma2_.readIfPresent(this->coeffDict());
        beta1_.readIfPresent(this->coeffDict());
        beta2_.readIfPresent(this->coeffDict());
        betaStar_.readIfPresent(this->coeffDict());
        a1_.readIfPresent(this->coeffDict());
        b1_.readIfPresent(this->coeffDict());
        c1_.readIfPresent(this->coeffDict());
        F3_.readIfPresent("F3", this->coeffDict());


        CDES_.readIfPresent(this->coeffDict());
        this->coeffDict().readIfPresent("FSST", FSST_);

        return true;
    }
    else
    {
        return false;
    }
}

template<class BasicTurbulenceModel>
void kOmegaSSTLMDES<BasicTurbulenceModel>::correctReThetatGammaInt()
{
    // Local references
    const alphaField& alpha = this->alpha_;
    const rhoField& rho = this->rho_;
    const surfaceScalarField& alphaRhoPhi = this->alphaRhoPhi_;
    const volVectorField& U = this->U_;
    const volScalarField& k = this->k_;
    const volScalarField& omega = this->omega_;
    const tmp<volScalarField> tnu = this->nu();
    const volScalarField::Internal& nu = tnu()();
    const volScalarField::Internal& y = this->y_();
    fv::options& fvOptions(fv::options::New(this->mesh_));

    // Fields derived from the velocity gradient
    tmp<volTensorField> tgradU = fvc::grad(U);
    const volScalarField::Internal Omega(sqrt(2*magSqr(skew(tgradU()()))));
    const volScalarField::Internal S(sqrt(2*magSqr(symm(tgradU()()))));
    const volScalarField::Internal Us(max(mag(U()), deltaU_));
    const volScalarField::Internal dUsds((U() & (U() & tgradU()()))/sqr(Us));
    tgradU.clear();

    const volScalarField::Internal Fthetat(this->Fthetat(Us, Omega, nu));

    {
        const volScalarField::Internal t(500*nu/sqr(Us));
        const volScalarField::Internal Pthetat
        (
            alpha()*rho()*(cThetat_/t)*(1 - Fthetat)
        );

        // Transition onset momentum-thickness Reynolds number equation
        tmp<fvScalarMatrix> ReThetatEqn
        (
            fvm::ddt(alpha, rho, ReThetat_)
          + fvm::div(alphaRhoPhi, ReThetat_)
```

```
          - fvm::laplacian(alpha*rho*DReThetatEff(), ReThetat_)
         ==
            Pthetat*ReThetat0(Us, dUsds, nu) - fvm::Sp(Pthetat, ReThetat_)
          + fvOptions(alpha, rho, ReThetat_)
        );

        ReThetatEqn.ref().relax();
        fvOptions.constrain(ReThetatEqn.ref());
        solve(ReThetatEqn);
        fvOptions.correct(ReThetat_);
        bound(ReThetat_, 0);
    }

    const volScalarField::Internal ReThetac(this->ReThetac());
    const volScalarField::Internal Rev(sqr(y)*S/nu);
    const volScalarField::Internal RT(k()/(nu*omega()));

    {
        const volScalarField::Internal Pgamma
        (
            alpha()*rho()
           *ca1_*Flength(nu)*S*sqrt(gammaInt_()*Fonset(Rev, ReThetac, RT))
        );

        const volScalarField::Internal Fturb(exp(-pow4(0.25*RT)));

        const volScalarField::Internal Egamma
        (
            alpha()*rho()*ca2_*Omega*Fturb*gammaInt_()
        );

        // Intermittency equation
        tmp<fvScalarMatrix> gammaIntEqn
        (
            fvm::ddt(alpha, rho, gammaInt_)
          + fvm::div(alphaRhoPhi, gammaInt_)
          - fvm::laplacian(alpha*rho*DgammaIntEff(), gammaInt_)
        ==
            Pgamma - fvm::Sp(ce1_*Pgamma, gammaInt_)
          + Egamma - fvm::Sp(ce2_*Egamma, gammaInt_)
          + fvOptions(alpha, rho, gammaInt_)
        );

        gammaIntEqn.ref().relax();
        fvOptions.constrain(gammaIntEqn.ref());
        solve(gammaIntEqn);
        fvOptions.correct(gammaInt_);
        bound(gammaInt_, 0);
    }

    const volScalarField::Internal Freattach(exp(-pow4(RT/20.0)));
    const volScalarField::Internal gammaSep
    (
        min(2*max(Rev/(3.235*ReThetac) - 1, scalar(0))*Freattach, scalar(2))
       *Fthetat
    );

    gammaIntEff_ = max(gammaInt_(), gammaSep);
}


template<class BasicTurbulenceModel>
```

```
void kOmegaSSTLMDES<BasicTurbulenceModel>::correct()
{
    if (!this->turbulence_)
    {
        return;
    }

// Local references
    const alphaField& alpha = this->alpha_;
    const rhoField& rho = this->rho_;
    const surfaceScalarField& alphaRhoPhi = this->alphaRhoPhi_;
    const volVectorField& U = this->U_;
    volScalarField& nut = this->nut_;
    fv::options& fvOptions(fv::options::New(this->mesh_));

    BasicTurbulenceModel::correct();

    volScalarField::Internal divU
    (
        fvc::div(fvc::absolute(this->phi(), U))()()
    );

    tmp<volTensorField> tgradU = fvc::grad(U);
    volScalarField S2(2*magSqr(symm(tgradU())));
    volScalarField::Internal GbyNu(dev(twoSymm(tgradU()())) && tgradU()());
    volScalarField::Internal G(this->GName(), nut()*GbyNu);
    tgradU.clear();

    // Update omega and G at the wall
    omega_.boundaryFieldRef().updateCoeffs();

    volScalarField CDkOmega
    (
        (2*alphaOmega2_)*(fvc::grad(k_) & fvc::grad(omega_))/omega_
    );

    volScalarField F1(this->F1(CDkOmega));
    volScalarField F23(this->F23());

    {
        volScalarField::Internal gamma(this->gamma(F1));
        volScalarField::Internal beta(this->beta(F1));

        // Turbulent frequency equation
        tmp<fvScalarMatrix> omegaEqn
        (
            fvm::ddt(alpha, rho, omega_)
          + fvm::div(alphaRhoPhi, omega_)
          - fvm::laplacian(alpha*rho*DomegaEff(F1), omega_)
         ==
            alpha()*rho()*gamma
           *min
            (
                GbyNu,
                (c1_/a1_)*betaStar_*omega_()
               *max(a1_*omega_(), b1_*F23()*sqrt(S2()))
            )
          - fvm::SuSp((2.0/3.0)*alpha()*rho()*gamma*divU, omega_)
          - fvm::Sp(alpha()*rho()*beta*omega_(), omega_)
          - fvm::SuSp
            (
                alpha()*rho()*(F1() - scalar(1))*CDkOmega()/omega_(),
                omega_
```

```
            )
          + Qsas(S2(), gamma, beta)
          + omegaSource()
          + fvOptions(alpha, rho, omega_)
        );

        omegaEqn.ref().relax();
        fvOptions.constrain(omegaEqn.ref());
        omegaEqn.ref().boundaryManipulate(omega_.boundaryFieldRef());
        solve(omegaEqn);
        fvOptions.correct(omega_);
        bound(omega_, this->omegaMin_);
    }

    // Turbulent kinetic energy equation
    tmp<fvScalarMatrix> kEqn
    (
        fvm::ddt(alpha, rho, k_)
      + fvm::div(alphaRhoPhi, k_)
      - fvm::laplacian(alpha*rho*DkEff(F1), k_)
     ==
        alpha()*rho()*Pk(G)
      - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
      - fvm::Sp(alpha()*rho()*epsilonByk(F1, F23), k_)
      + kSource()
      + fvOptions(alpha, rho, k_)
    );

    kEqn.ref().relax();
    fvOptions.constrain(kEqn.ref());
    solve(kEqn);
    fvOptions.correct(k_);
    bound(k_, this->kMin_);


//    --------------- LM PART ------------------
    correctReThetatGammaInt();


}

} // End namespace LESModels
} // End namespace Foam
```

This page was intentionally left blank