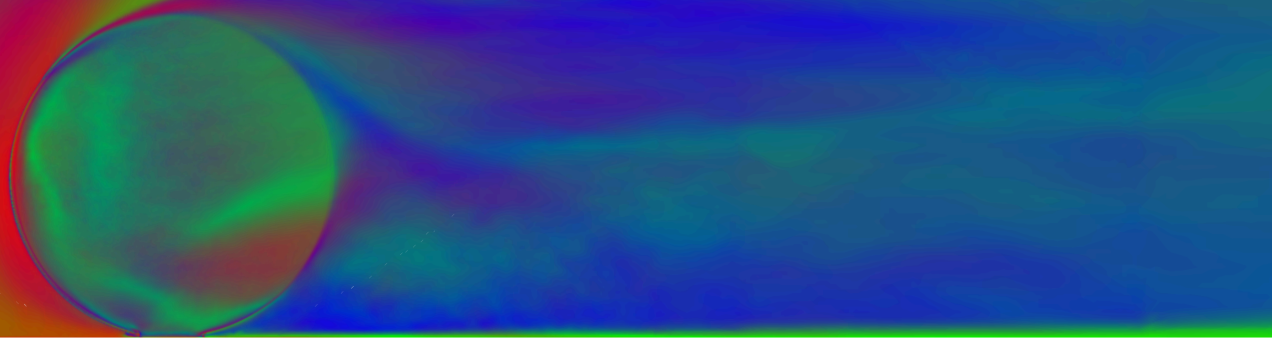
MSc. Thesis Data Driven Turbulence Modelling

McLaren Racing Ltd.
J.P. Huijing



Technische Universiteit Delft

TU Delft

MSc. Thesis Data Driven Turbulence Modelling

McLaren Racing Ltd.

by

J.P. Huijing

Student number:	4281195	
Project duration:	September 1, 2019 – September 2, 2020	
Supervisor:	Dr. R.P. Dwight	
Thesis Committee:	Dr. R.P. Dwight	TU Delft
	Dr. S.J. Hulshoff	TU Delft
	Dr.r. E.J. van Kampen	TU Delft
	Dr. F. Bottone	McLaren Racing ltd.

Confidential

TU Delft

Abstract

RANS simulation are one of the most used tools for aerodynamic analysis. The advantage of RANS simulations is the reduced computational cost compared to other methods such as LES or DNS. This is because by solving the RANS equations one only solves for the mean flow. However, this reduction in computational cost comes at the price of uncertainty. During the derivation of the RANS equations a closure problem is introduced. The turbulence models that solve this closure model of the Reynolds stress introduce the largest level of uncertainty. In this research symbolic regression is used to augment the $k-\omega-SST$ turbulence models with corrections directly inferred from high fidelity data sources such as DES, LES or DNS. This model is augmented with a correction to the anisotropic part of the Reynolds stress and a direct correction to the turbulence production. These correction fields are calculated through the so called frozen approach, where the $k-$ and ω-transport equations are solved using the frozen mean flow values from the high fidelity data. Symbolic regression with Pope's eddy viscosity hypothesis as a basis is used to find models for these correction fields. With these correction models the $k-\omega-SST$ is extended to a non-linear turbulence model for the prediction of the eddy viscosity. Elastic net regression forms the basis of the machine learning method and is used to promote sparsity of the correction models. The sparsity is needed for stability and to keep the computational efficiency of the RANS method. This symbolic regression method is named SpaRTA and has been successfully applied to 2D cases in literature. In this work that is extended to 3 fully 3D cases, namely a wall mounted cube, an infinite circular cylinder and an idealised rotating wheel on a road surface. This study has shown the applicability of the SpaRTA methodology to these cases, resulting in models that have the extrapolating capabilities to improve the results on all 3 test geometries.

Preface

Before you lies my MSc. thesis "Data driven turbulence modelling", the research presented in this work focuses on applying machine learning techniques to find new turbulence models. The project for me started out from a combined interest in machine learning and aerodynamics, where through this project I have been able to expand my knowledge in both fields.

I would like to thank my supervisor Dr. Richard P. Dwight for his excellent guidance and advice during the project. I would also like to thank my colleagues at McLaren Racing whose help has been crucial for the successful completion of this work. To be able to do this project at McLaren has been a real privilege. I have enjoyed their sincere interest in the project and involved discussions about the next steps for this methodology and I believe this has also helped to get this work to a higher level.

Working on this project has been exciting with the every new results showing encouraging signs of improvement. I am happy with the work that has been performed and hope to have made a contribution to the next step in the improvement of RANS simulations.

J.P. Huijing
Delft, July 2020

Contents

List of Figures

List of Tables

List of Symbols and abbreviations

Symbols

A_k	=	Anti-symmetric tensor for the gradient of the turbulent kinetic energy	$[-]$
$\mathbf{B}$	=	Library of combinations of Invariants	$[-]$
$\mathbf{C}_{b_{ij}^{\Delta}}$	=	Function library for b_{ij}^{Δ}-correction	$[-]$
$CD_{k\omega}$	=	$k-\omega-SST$ cross diffusion term	$[-]$
$\mathbf{C}_R$	=	Function library for R-correction	$[-]$
$C_\mu, C_{\epsilon 1}, C_{\epsilon 2}$	=	$k-\epsilon$ turbulence model closure coefficients	$[-]$
$C1_c$, $C2_c$, $C3_c$	=	Barycentric coordinate	$[-]$
$\tilde{D}$	=	Dimensionless turbulence destruction	$[-]$
I	=	Identity Matrix	$[-]$
II	=	Lumley triangle coordinate	$[-]$
III	=	Lumley triangle coordinate	$[-]$
$N(\cdot)$	=	Navier-Stokes equations	$[-]$
P	=	Non-persistence-of-straining tensor	$[-]$
$\tilde{P}$	=	Dimensionless turbulence production	$[-]$
R	=	Turbulent kinetic energy correction term	$[m^2/s^3]$
$\mathbf{R}$	=	Reynolds stress tensor	$[m^2/s^2]$
R_{ij}	=	Rotation rate tensor	$[-]$
S_{ij}	=	Strain rate tensor	$[-]$
$T_{ij}^{(n)}$	=	Input tensor	$[-]$
$\mathbf{U}$	=	Velocity vector	$[m/s]$
V	=	Eigenvectors	$[-]$
a_{ij}	=	Reynolds anisotropy	$[m^2/s^2]$
b_{ij}	=	Dimensionless Reynolds anisotropy	$[-]$
b_{ij}^{Δ}	=	Anisotropy correction field	$[-]$
$\mathbf{b}_B$, b_{ij}^0	=	Anisotropy field from Boussinesq approximation	$[-]$
$\mathbf{b}_{ML}$	=	Anisotropy field predicted by the machine learning algorithm	$[-]$
k	=	Turbulent kinetic energy	$[m^2/s^2]$
p	=	Pressure	$[Pa]$
r	=	Reynolds force vector	$[m/s^2]$
u	=	Velocity	$[m/s]$
$y+$	=	Dimensionless wall distance	$[-]$
y_e	=	Simulation result	$[-]$
y_m	=	Measured result	$[-]$

Greek Symbols

Λ	=	Eigenvalues	$[-]$
Φ_m	=	Measurement data	$[-]$
Φ_s	=	Simulation data	$[-]$
$\tilde{\Omega}$	=	Non-dimenionalized vorticity	$[-]$
β, β^*	=	$k-\omega$-SST closure coefficients	$[-]$
γ	=	Anisotropy ramping factor	$[-]$
δ_{ij}	=	Isotropic part of the Reynolds stress	$[-]$
η	=	Turbulence triangle coordinate	$[-]$
$\theta(n)$	=	Model coefficients	$[-]$
λ	=	Eigenvalue	$[-]$
ν_t	=	Turbulence viscosity	$[m^2/s]$
ν	=	Kinematic viscosity	$[m^2/s]$
ξ	=	Turbulence triangle coordinate	$[-]$
$\xi(n)$	=	Model performance	$[-]$
χ	=	Ratio of the Spallart Allmaras working variable/kinematic viscosity	$[-]$
ρ	=	Density	$[kg/m^3]$
ρ_{RS}	=	Boussinesq validity indicator	$[-]$
$\sigma, \sigma^*, \sigma_d$	=	$k-\omega$-SST closure coefficients	$[-]$
σ^2	=	Standard deviation	$[-]$
τ_{ij}	=	Reynolds Stress	$[m^2/s^2]$
τ_{wall}	=	Wall shear stress	$[N/m^2]$
ω	=	Specific dissipation rate	$[m^2/s^3]$
$\tilde{\omega}$	=	Effective specific dissipation rate used to compute eddy viscosity	$[m^2/s^3]$

Abbreviations

BFS	Backward-facing step
CART	Classification and Regression Tree
CBFS	Curved backward-facing step
DNS	Direct Numerical Simulation
ET	Expression Tree
GEP	Genetic evolutionary programming
JPDF	Joint probability density function
LES	Large Eddy Simulation
LEVM	Linear eddy viscosity model
MAP	Maximum aposteriori estimate
ML	Machine Learning
MLP	Multi-layer Perceptron Network
NN	Neural Network
OoB	Out-of-Bag
PH	Periodic Hill
QEVM	Quadratic Eddy Viscosity Model
RANS	Reynolds Averaged Navier-Stokes
TBNN	Tensor Basis Neural Network
TBRF	Tensor Basis Random Forest

1 Introduction

The purpose of this chapter is to give the reader an introduction to the topic and the motivation for this research. The problem is described accompanied by a presentation of some existing solutions. After this the research questions that are attempted to be answered in this research are presented. Finally an outline of the contents of this report is given, guiding the reader where to find what information.

1.1. Motivation

A good compromise between computational cost and accuracy for the simulation of flows is achieved through a large eddy simulation (LES). With the advancements made in LES methodology and computer hardware continuing to become faster, the use of the LES methodology and the availability of high fidelity data has significantly increased.

In most industrial application where resources and time are limited the choice is often made to solve the Reynolds averaged Navier-Stokes equations (RANS) instead of doing an LES. Performing RANS simulations is a cheaper alternative over the computationally much more expensive LES method. The main difference between LES and the RANS approach is that in LES the equations are solved in discrete time steps for the instantaneous solution, requiring the solution to be solved over a large period of time to attain the averaged solution. While by solving the RANS equations, one only solves for the mean solution directly. This assumption has as a result that the method is cheaper and therefore also faster. However, this reduced cost also brings uncertainty in the form of inaccuracy. During the derivation of the RANS equations assumptions are made and as a result of the averaging procedure to arrive at the equations a closure problem arises.

This closure problem is introduced in the form of the Reynolds stress tensor that needs to be modelled. The models for this Reynolds stress are called turbulence models and are all approximations based on assumptions that introduce some form of modelling error. Because all models have assumptions there does not exist a perfect model and each model has its own strengths and weaknesses.

The increase in popularity and development of machine learning algorithms and the advancements made in the areas of hard- and software has resulted in research starting to focus on the challenge of turbulence modelling in a more data driven way. This is done in different ways but usually involves employing a machine learning algorithm on a set of high fidelity data, gathered by simulation methods such as LES and DNS or by experiments. Previous works have proven that these machine learning techniques can be used to improve the prediction of 2D flows. This forms the motivation to further investigate the application of and develop one of these techniques.

1.2. Problem statement and research context

The strength of machine learning algorithms is the ability to find patterns in large sets of data. This feature of machine learning and the availability of high fidelity data opens up the question: Can the accuracy of RANS solvers be improved by augmenting the turbulence model with a machine learned model? Specifically for this research the turbulence model that is selected to be improved is the $k-\omega-SST$ turbulence model. This turbulence model is a combination of the $k-\omega$ and $k-\epsilon$ turbulence models and is described in more detail in Section 3.1.

There are multiple other research groups that have been applying data driven techniques for the improvement of RANS simulations. These works can generally be split up into three categories. The first category uses these data driven techniques to tune the model coefficients of existing models. Examples of this are the Bayesian approach applied by Edeling, Cinella and Dwight [3]. Who used this to tune the coefficients of the $k-\epsilon$ turbulence model. Li, Hoag, Martin and Bailey [9] have developed an iterative approach where the model coefficients are tuned while solving a RANS equation, high fidelity or experimental data is used as a reference for the tuning. These approach are described in more detail in Section 2.3.1.

The second category is that of the field inversion methods. These methods use the predictive qualities of the machine learning methods to apply a correction to an existing turbulence model. The target for the machine learning algorithm is an output of the RANS simulation. Singh Duraisamy and Pan [20] have applied a version of this field inversion method on the Spalart Allmaras turbulence model. They have used a neural network for the prediction of the correction field. This method is explained in more detail in Section 2.3.2. This approach is called field inversion because it looks at the full field and through a single correction tries to match measurements or high fidelity data with the simulated data.

A similar but fundamentally different approach is that applied by Ling, Kurzawski and Templeton [10]. They applied a feed-forward neural network in combination with a tensor basis as proposed Pope [16]. This method is used to correct the anisotropic part of the Reynolds stress, with the neural network predicting this correction. This predicted correction field is then added to a converged solver after which it allowed to converge again. This approach is described in Section 2.3.3 and the tensor basis is described in more detail in Section 3.2.1. This work belongs to the third category, those are the approaches that use this tensor basis.

Many other works have followed up on this approach using the tensor basis to correct the anisotropic part of the Reynolds stress. It has been combined with many different machine learning techniques such as the regression tree algorithm applied through a random forest as applied by Kaandorp and Dwight [6]. This approach is presented in Section 2.3.4.

Weatheritt and Sandberg [26] have attempted to make this method using the tensor basis generally applicable without the need for a machine learning algorithm implemented in the turbulence model. Instead they used evolutionary programming to find correction models that can be directly implemented into existing turbulence models. This method is closest to the methodology followed in this research and is explained in more detail Section 2.3.5.

Finally two other approaches are highlighted in this report that do not use the tensor basis for the correction of turbulence models. The first of these is presented by Cruz et al. [1], instead of correcting for the Reynolds stress they correct for the divergence of the Reynolds stress, the Reynolds force vector. A neural network is used to predict the correction needed in the Reynolds force vector to match the high fidelity data. The reason to use this Reynolds force vector is to mitigate the error that is made in calculating the Reynolds stress from high fidelity data, as there is a discrepancy between the real Reynolds stress and the calculated one. A more detailed explanation of this method is found in Section 2.3.7. The other method uses the decomposition of the anisotropic part of the Reynolds stress into eigenvalues and eigenvectors to find perturbation and transformation predicted by a neural network to get the correct Reynolds stress. The full details of this approach are shown in Section 2.3.8.

In this work the methodology presented by Schmelzer and Dwight [18] is followed. This methodology is closest to evolutionary algorithm from Weatheritt and Sandberg and is called Sparse Regression of Turbulent Stress Anisotropy (SpaRTA). This methodology attempts to find correction models to be directly implemented in existing turbulence models.

In the SpaRTA methodology the turbulence model that has been selected is the $k-\omega-SST$ model, but in theory other turbulence models can be used as well. In contrary to the evolutionary algorithm two corrective fields are added to the turbulence model. One field is correcting the anisotropic part of the Reynolds stress tensor and the other is directly correcting the turbulence production. These corrective fields are determined using the so called frozen approach described in Section 3.2.2.

To make the corrections applicable to a range of problems symbolic regression is performed to find correction models that can be directly implemented into the $k-\omega-SST$ turbulence model. For this symbolic regression the elastic net algorithm is used to promote sparsity and keep coefficients small. These are two of the requirements for the stability of a turbulence model. The exact machine learning technique used to infer models from the high fidelity data is presented in Section 3.2.3.

1.3. Research objective

The objective for this research is to further examine if the accuracy of RANS solvers be improved by augmenting the turbulence model with a machine learned model. The methodology that has been selected more specifically as stated above is the SpaRTA methodology.

Schmelzer and Dwight [18] have proven this improvement for 2D flow cases. In order to fully answer the main research question in this work the extension is made to 3D flow cases. This is an important step as most flow cases are 3D and in most industrial applications the interest is in 3D flows.

When the correction models improve the solution of 3D flows there are some other questions that need to be answered before the SpaRTA methodology can be used in an industrial environment. The first important question that needs to be answered is that of the computational cost. Are the correction models found using the SpaRTA methodology computationally cheap enough to validate their use? This is an important question as the main benefits of the RANS methodology are the low computational cost and the robustness of the method.

To answer this question it is not only required to look at the computational cost per calculation step but also to look at the combination of convergence rate, computational cost of the additional equations and the accuracy of the solution. Finally a question that is tried to be answered is: What are the extrapolating qualities of the correction models found using the SpaRTA methodology. In other words, how well do the correction models perform on geometries other than the training geometry? What are the limitations of such a correction model? Are there certain types of flows that the correction model is unable to correctly resolve? These are all questions that are tried to be answered in this research.

Some other questions that need to be answered are about the assumptions made in the methodology. An example is the choice of not using the full tensor basis but only a part of it for the model regression. This looked to be sufficient for 2D flow cases but will it also be sufficient for 3D cases where more mean flow features are present?

Finally when looking at the application of the models discovered using the SpaRTA methodology it is interesting to know what types of flows or what areas can be improved the most. This is interesting knowledge for industrial applications that deal with very specific flow types.

1.4. Report Outline

The report is structured in the following way: Chapter 2 covers in more detail the literature that was briefly discussed above in the problem statement and research context section. This is to give the reader an idea of what has been done before and where this work falls within the existing research frame-work.

In the following chapter the methodology as is followed in this research is presented. This chapter start with a short overview of the RANS method, explaining the assumptions that are made by using the RANS method. This is done together with an overview of the high fidelity methods that have been used in this research. Also, turbulence modelling and what assumptions they are based upon will be briefly discussed. This theoretical background can be found in Section 3 before addressing the methodology specific to the

SpaRTA approach in Section 3.2. Finally the implementation of the method in an industrially relevant solver is presented in Section 3.3, this implementation is verified for correct implementation against the results from Schmelzer and Dwight [18].

To answer the research questions posted above the SpaRTA methodology has been applied to 3 different test cases. In Chapter 4 these test cases are described and the results are presented. The first test case that has been used is that of a wall mounted cube. This test case has been used to test several things besides the general improvement in performance of the correction models generated with the SpaRTA methodology. It was also used to prove the assumption that a reduced tensor basis can be used for the model regression. Other things that have been tested using this geometry are the stability of the model that seems to depend on the size of the anisotropy correction and the good extrapolating qualities of this model. When the model is applied to an extended cube similar improvements in performance are noted.

To further test the extrapolating qualities of the model that was found for this flow case the model has also been applied to the two other test cases that have been tested. The results of this model and that trained directly on the geometry of an infinite circular cylinder are shown in Section 4.2.

The other test case for which the results are shown is that of an idealised rolling wheel on a road surface. This is the most complex test case that has been done, the results of which can be found in Section 4.3.

In Chapter 5 the conclusions that can be drawn from the results are presented and discussed. From these conclusions it is not yet possible to fully answer all the research questions. With the results achieved in this research some questions remain and new questions about how to furhter improve the method have arisen. To answer these, more research is required, recommended steps for further research to get a complete answer to the research questions can be found in Section 6. This is accompanied by recommendations for possible further development of this method.

2

Overview of data driven techniques in literature

In this chapter a summary is given of the literature regarding machine learning methods applied to turbulence modelling. This should provide the reader with some background information to see where this research falls within the machine learning and turbulence modelling field. For a short overview of the works discussed here the reader is referred to Section 1.2 of the introduction.

Before discussing these methods different levels of modelling inaccuracy are discussed as this gives a good idea of the differences between several methods. This is needed because not all methods that act on the same modelling level. After that a short part of the work from Emory and Iaccarino [4] is discussed, they have described methods for visualising turbulence. The visualisation of turbulence is an essential part of the discussion as several methods use this approach for the visualisation of the results. It will also be used later in the presentation of the results in chapter 4 and is therefore presented here. After that most relevant literature in relation to this work is presented.

2.1. Levels of modelling inaccuracy

Before discussing the different data-driven methods for turbulence modelling it is good to asses the different types of inaccuracies that are introduced at the different levels of turbulence modelling. Duraisamy, Iaccarino and Xiao [2], have produced an overview of different data driven techniques and have presented the different stages during which errors are made. They labelled each of these levels L1-L4 and are important to understand at which level the data driven models operate.

The first level (L1) of modelling inaccuracy is that introduced during averaging. The fact that the average of the Navier-Stokes equation is not equal to the Navier-Stokes equations of average values is the first level of inaccuracy. The derivation of the Reynolds averaged Navier-Stokes equation is shown in section 3.1.1

$$\langle N(\cdot)\rangle \neq N(\langle\cdot\rangle) \tag{2.1}$$

The second level (L2) of inaccuracy is in the assumption that the additional terms that appear in the averaging process can be modelled using a set of averaged variables as the model variables. The inaccuracies on this level come from the choice of variables and the fact that averaged variables are used. An example of this is the Boussinesq approximation, this is discussed in more detail in section 3.1.2.

The third level of inaccuracy (L3) is the model $M(\cdot)$ and the assumptions that are made while deriving the model. Each assumption that is made while deriving the model is a source of model inaccuracies.

$$\langle N(\cdot)\rangle \neq N(\langle\cdot\rangle) + M(\cdot) \tag{2.2}$$

Finally the fourth level (L4) that they have identified is the inaccuracy in model coefficients. These coefficients determine model behaviour and can be changed according to different conditions. Over the last decades a

lot of effort has gone into determining the coefficients that give the most accurate solutions in general or for very specific flow cases. This is classically done by tuning the coefficients using a database of fundamental flows. [21].

2.2. Visualising Turbulence

A lot of the Machine learning methods focus on fixing the anisotropic part of the Reynolds stress. In order to compare and understand the modifications that are made by the machine learning algorithms to the the turbulence models it is of critical importance to have tools available to compare turbulence models and the prediction of turbulence.

[4] Emory and Iaccarino have described multiple methods that can be used to visualise the turbulence anisotropy. The Reynolds stress anisotropy tensor as given by (2.3), is the anisotropic part of the Reynolds stress and gives a good amount of information on the turbulence model. However, a tensor is difficult to visualise effectively.

$$a_{ij} = \frac{\overline{u_i' u_j'}}{2k} - \frac{\delta_{ij}}{3} \tag{2.3}$$

A technique that is used frequently is to use anisotropy invariant maps [12]. The technique uses the invariants of the the anisotropy tensor, in this case the eigenvalues λ_{1-3} and eigenvectors e_{1-3} resulting from the diagonalisation of a_{ij}. Using these eigenvalues different maps can be made to visualise the state of turbulence. The general commonality between all the invariants maps is that the independent variables and axes variables are functions of the anisotropy eigenvalues. The eigenvalues can be used to describe the relative strengths of the fluctuating velocity components. There are three limiting states that define the boundaries of the invariant maps. The first of the limiting states is one component. It describes a flow where the turbulent fluctuations only exist along one direction. The second is where fluctuations exist in two components and finally there is isotropic turbulence which is fluctuating in all three directions.

The first map that can be used is the so called Lumley triangle [12]. It creates the coordinate system (II, III), which results in a triangle with curved sides as shown in figure 2.1(a). The corners of the triangle are the limiting states of isotropic x_{3_c}, axisymmetric x_{2_c} and one-component x_{1_c} turbulent fluctuations. On the right hand side the turbulence triangle is shown. This triangle is similar to the Lumley triangle but has had a coordinate transformation:

$$II = a_{ij}a_{ji}/2 = \lambda_1^2 + \lambda_1\lambda_2 + \lambda_2^2 \tag{2.4}$$

$$III = a_{ij}a_{jn}a_{ni}/3 = -\lambda_1\lambda_2(\lambda_1 - \lambda_2) \tag{2.5}$$

$$\xi^3 = III/2, \quad \eta^2 = II/3 \tag{2.6}$$

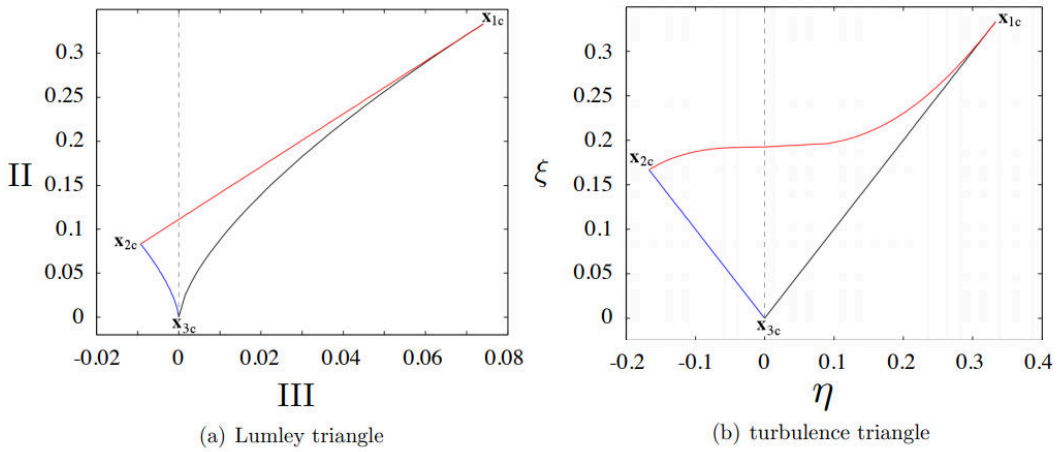
In figure 2.2 two more invariant maps are shown. In figure 2.2(a) the eigenvalue map is shown, which uses the first two eigenvalues as coordinates. The dashed line indicates the plane-strain limit. This limit, where at least one of the eigenvalues is zero.

In figure 2.2b the barycentric map is shown. This map spans an Euclidian domain, in this case constructed as an equilateral triangle described by the corner points $X_{1c} = (1,0), X_{2c} = (0,0), X3c = (1/2, \sqrt{3}/2)$. All the states of a turbulent flow should fall into the triangles. And with that comes the downside of the method. For large sets of data a point cloud is the result of plotting in the barycentric map, cluttering the domain and reducing visibility. Another downside of plotting in the invariant maps is the lack of physical locations. The data points in the domain are hard to link back to physical locations in the domain. This makes observing patterns and global componentiality behaviour difficult. There are methods to resolve this issue of the lacking link with the physical domain, by labelling and colouring the points in the map according to their location.

$$C1_c = \lambda_1 - \lambda_2 \tag{2.7}$$

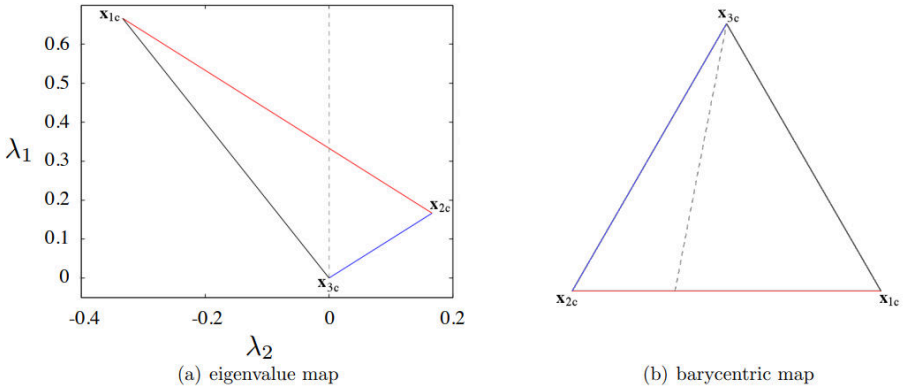
$$C2_c = 2(\lambda_2 - \lambda_3) \tag{2.8}$$

$$C3_c = 3\lambda_3 + 1 \tag{2.9}$$



(a) Lumley triangle (b) turbulence triangle

Figure 2.1: Diagram of two non-linear anisotropy invariant maps. Figure taken from [4].



(a) eigenvalue map (b) barycentric map

Figure 2.2: Diagram of two linear anisotropy invariant maps. Figure taken from [4].

$$x_B = C_{1c}x_{1c} + C_{2c}x_{2c} + C_{3c}x_{3c} = C_{1c} + C_{3c}\frac{1}{2} \tag{2.10}$$

$$y_B = C_{1c}y_{1c} + C_{2c}y_{2c} + C_{3c}y_{3c} = C_{3c}\frac{\sqrt{3}}{2} \tag{2.11}$$

Another option is to instead of colouring the points in the triangle according to their physical location to colour the points in the domain according to their place in the domain. This can be done by applying an RGB map to the domain where 100% red is equal to full X_1C, green to X_{2c} and blue to x_{3c} (figure 2.3).

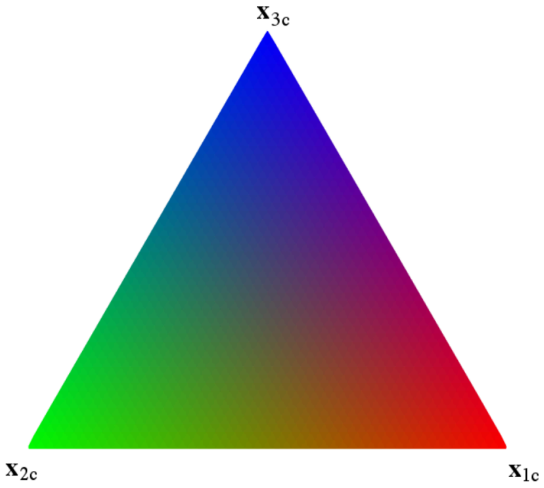


Figure 2.3: RGB map according to the Barycentric map. Figure taken from [4].

2.3. Machine learning applied to Turbulence modelling

In this section the application of machine learning to turbulence modelling is presented. Where machine learning is identified as a class of algorithms that do not use explicit instructions but relies on pattern recognition and inference instead [7].

2.3.1. Model calibration

One of the approaches that is used to optimise the accuracy of existing turbulence models is to tune the model coefficients such that the model performs well under certain flow types. The standard coefficients for models have usually been determined using fundamental flow types such as turbulent channel flow or flat plate flow. This type of model calibration as acting on the L4 uncertainty level as it only modifies model coefficients and not model form or any fundamental assumptions in the derivation of the model.

To get more accurate solutions for more complex flows it is useful to re-calibrate these model coefficients to values that give more accurate results. The calibration of these coefficients can be done by hand or from experience.

Bayesian Approach If there is high fidelity data available of a similar flow, statistical methods can be used to determine the best fitting model coefficients for RANS models. Edeling, Cinella and Dwight [3] have used an approach where using a Bayesian approach the optimal coefficients for multiple turbulence models for multiple flow cases have been generated by using a statistical method.

The model is matched to experimental data by sampling for the posterior of the model coefficients. This sampling is done using the last 5000 samples of a converged Markov Chain. The flow cases that were tested were that of a flat plate boundary layer with different pressure gradients. What was found was that the posterior model probabilities vary greatly with the pressure-gradient applied. It was therefore fair to say that the model coefficients for the best fit depend on the flow case and there are no coefficients that suit all conditions well.

Ray et al [17] have used a similar Bayesian approach to calibrate the coefficients of the $k-\epsilon$ turbulence model for a jet in crossflow conditions. This flow case is a complex flow with a jet blowing perpendicular to the oncoming flow. This results in the plume of the jet rolling over and producing two counter-rotating vortices.

The problem that needs to be solved is an inverse problem to calculate the joint probability density function (JPDF)of the model coefficients $\mathbf{C} = (C_\mu, C_{\epsilon 2}, C_{\epsilon 1})$. In this case the data from which this inverse problem is solved is wind-tunnel data gathered using PIV and DNS simulations. In this case the equations that is tried to be maximised is the following:

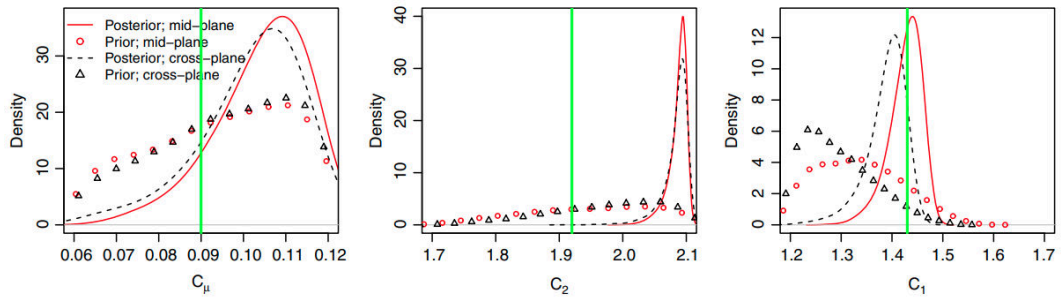


Figure 2.4: Prior and posterior densities obtained from midplane and cross-plane measurements. Posterior densities are plotted using solid and dashed lines. The corresponding priors are pltted with ∘ and Δ. The vertical lines are the "nominal" values of the parameters. Figure taken from [17].

$$P(\mathbf{C},\sigma^2|\mathbf{y}_e) \propto L(\mathbf{y}_e|\mathbf{C},\sigma^2)\Pi_1(\mathbf{C})\Pi_2(\sigma^2) \tag{2.12}$$

Where it is assumed that the likelihood of the simulation result $\mathbf{y}_e$ has a normal distribution and can be compared to the measurement which gives the following expression for the JPDF:

$$P(\mathbf{C},\sigma^2|\mathbf{y}_e) \propto \frac{1}{\sigma^{N_p}} exp\Big(-\frac{||\mathbf{y}_e-\mathbf{y}_m(\mathbf{C})||_2^2}{2\sigma^2}\Big)\Pi_1(\mathbf{C})\Pi_2(\sigma^2) \tag{2.13}$$

With the measurements being in this case the velocity deficit in streamwise and normal direction with reference to the free stream velocity. Solving this equation (2.13) requires specifying $\Pi_1(\mathbf{C})$ and $\Pi_2(\sigma^2)$. Where it is assumed that the prior believe of σ^2 is represented by its reciprocal, $\Pi_2(\sigma^{-2})$, which is modelled using a Gamma distribution. The distribution of (2.13) is complicated and is realised by drawing samples of $(\mathbf{C},\sigma^2)$. The sampling is done by running a simulation for each sample of $\mathbf{C}$ that is a realistic combination, ie. physical combination of coefficients that gives a converging solution, using the Markov Chain Monte Carlo algorithm.

The result of this sampling procedure is the JPDF for the coefficients, with a maximum a posteriori estimate (MAP). Which can be seen in figure 2.4, where it can be seen that the MAP estimate for the coefficients is different from the values of the coefficients used in the original model.

In figure 2.5 results are plotted for both the standard coefficients C_{nom} and the MAP estimate values by taking the mean of the JPDF (ensemble mean) and the optimal coefficient from the analytic solution C_a. These results show the improved correspondence with the experimental results, especially the location of the velocity deficits and vertical flow has been much improved with some slight deviations in the magnitude.

They concluded from this that two-equation RANS models can be tuned to improve results for a jet in crossflow conditions. They also showed that the analytical solution of the model coefficients followed the trends seen in the JPDFs. The outcomes makes them believe that the effect of the model-form errors on the JPDFs are small and the predictive skill of $\mathbf{C}_a$.

Iterative Approach

An iterative approach that does this model tuning 'on the go' is presented by Li, Hoagg, Martin and Bailey [9]. They have chosen the $k-\omega$ turbulence model to iteratively determine the coefficients for. The standard transport model and coefficients have the following form.

For the implementation of their method they have used the KATS computing framework. The method which is called the retrospective cost adaptation algorithm (RCA-RANS) uses the known flow-field measurements (velocity or pressure) at certain locations in the flow to adapt the $k-\omega$ closure coefficients, which are required to solve the transport equations. The closure coefficients are updated with time step $\Delta t_{\text{adapt}} \triangleq \eta_{\text{adapt}}\Delta t$. With that the closure coefficients on adaptation step n are given by (2.14)

$$\theta(n) = [\alpha(n)\,\beta(n)\,\beta^*(n)\,\sigma_k(n)\,\sigma_\omega(n)]^T \in \mathbb{R}^{l_\theta} \tag{2.14}$$

The performance of the model with these model coefficients $\theta(n)$ is described by the following parameter

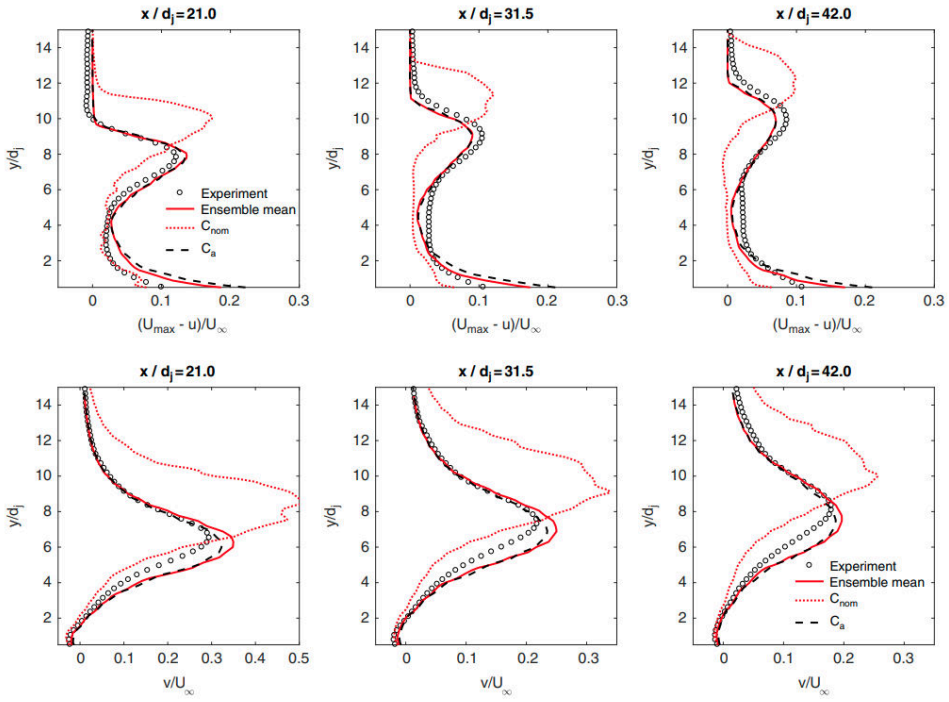


Figure 2.5: Streamwise velocity deficit (top) and normalised vertical velocity (bottom) computed using C_{nom} and C_a (analytical estimate) compared with the ensemble mean from the "pushed-forward posterior" test and experimental measurements. Results are plotted for the (M=0.8, J=10.2) test case, at three streamwise locations ($x/d_j = 21, 31.5$ and 42). Figure taken from [17].

$$\xi(n) \stackrel{\Delta}{=} \Phi_s(n) - \Phi_m(n) \tag{2.15}$$

Where $\Phi_m(n)$ is the measurement data and $\Phi_s(n)$ is the simulation data at each reference point in the domain. This performance ξ is used to adapt the model coefficients $\theta(n)$.

To proof the working of the method, it has also bee applied to a surface mounted cube, The measurement data from experiments were used to do the iterative approach. By iteratively altering the closure coefficients as described above they produced result using a RANS simulation that were closer to the experimental data then a RANS simulation with the default coefficients, but it did not match the measurement result.

2.3.2. Field inversion methods

A method that acts on the L3 level is that applied by Singh, Duraisamy and Pan [20], who applied the technique of inverse modelling for augmenting the Spalart Allmaras (SA) turbulence model. The idea behind this method is to improve the accuracy of the model by adding additional corrective terms. These terms are in this method determined by a neural network (NN).

The main form of the SA turbulence model is the following:

$$\frac{D\tilde{\nu}}{Dt} = P(\tilde{\nu}, \mathbf{U}) - D(\tilde{\nu}, \mathbf{U}) + T(\tilde{\nu}, \mathbf{U}) \tag{2.16}$$

The augmentation of the model happens by multiplying a spatially-varying term $\beta(\mathbf{x})$ to the production term. Which is equivalent to adding a source term to the equation.

$$\frac{D\tilde{\nu}}{Dt} = \beta(\mathbf{x})P(\tilde{\nu}, \mathbf{U}) - D(\tilde{\nu}, \mathbf{U}) + T(\tilde{\nu}, \mathbf{U}) \tag{2.17}$$

The problem that is being solved to get the optimal value of the corrective term β is shown in (2.18). Where N_d data points and where $G_j(\beta)$ is the output of the RANS model.

$$\min_{\beta} \sum_{j=1}^{N_d} \left[G_{j,exp} - G_j(\beta)\right]^2 + \lambda \sum_{n=1}^{N_m} \left[\beta(x_n) - 1\right]^2 \tag{2.18}$$

$$\min_{\beta} \sum_{j=1}^{N_d} \left[C_{p_{j,exp}} - C_{pj}(\beta)\right]^2 + \lambda \sum_{n=1}^{N_m} \left[\beta(x_n) - 1\right]^2 \tag{2.19}$$

To solve this, two variables for determining β are proposed. The first is the pressure coefficient C_p in case experimental pressure measurements are used, this leas to the minimisation problem shown in (2.19). Another option is the lift coefficient as in most experimental studies the lift coefficient is measured. Both these functions have led to a similar solution of the inverse problem.

In order to use the corrective field derived here it is necessary to create a map from $\beta(x)$ to something that only depends on the mean local flow variables. Before this is done the variables are non-dimensionalised such that it can be applied to a variety of problems.

To make this method generally applicable the corrective field needs to be predicted, for this machine learning is used. Specifically a neural network has been chosen to form the basis for predicting the corrective field. The solution to the inverse problem from (2.18) is the target for the machine learning algorithm.

The machine learning algorithm that was chosen for this specific case is the feed forward neural net. The fields from the flow field that forms the input basis for the neural network algorithm are stored in a vector named η, such that the corrective field $\beta(x) \approx \beta(\eta)$. The features that are chosen for η are all variables that are also seen in the Spalart Allmaras turbulence model, it is important the variables are non-dimensional.

The first is $\chi = \hat{\nu}/\nu$, the ratio of the SA working variable over the kinematic viscosity. Secondly a non dimensionalized vorticity $\bar{\Omega}$ is used as input (2.20)

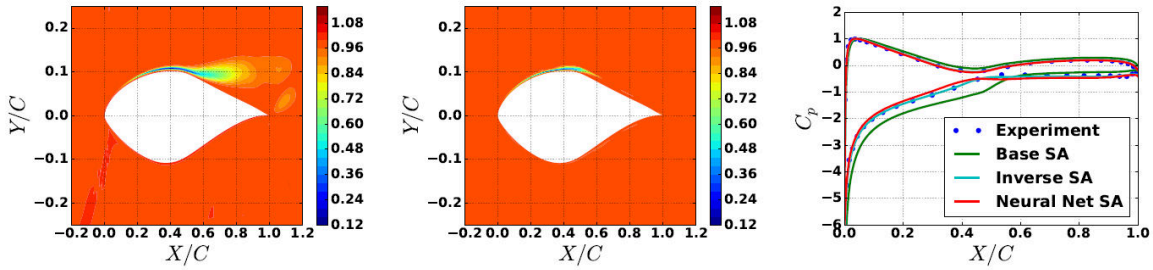
$$\bar{\Omega} = \frac{d^2}{\hat{\nu} + \nu} \tag{2.20}$$

Besides that the dimensionless versions of the production and destruction term are used, (2.21) (2.22).

$$\bar{P} = \frac{d^2}{(\hat{\nu} + \nu)^2} s_p = c_{b1}(1 - f_{t2})\left(\frac{\chi}{\chi + 1}\right)\left(\hat{\Omega} + \frac{1}{\kappa^2}\frac{\chi}{\chi + 1} f_{t2}\right) \tag{2.21}$$

$$\bar{D} = \frac{d^2}{(\hat{\nu} + \nu)^2} s_d = \left(\frac{\chi}{\chi + 1}\right)^2 c_{w1} f_w \tag{2.22}$$

With that the complete input vector has the following form: $\{\bar{\Omega}, \chi, S/\Omega, \tau/\tau_{wall}, P/D\}$. Where S is the strain rate and τ and $\tau_w all$ are the Reynold stress and wall shear stress.



(a) β (**x**) from inverse SA (b) β (**U**) from NN-augmented SA (prediction) (c) Pressure coefficient

Figure 2.6: Results comparing the inverse method to the neural network prediction applied to an S809 airfoil at an angle of attack of 14° and $Re = 2 \times 10^6$. Figure taken from [20].

.

The input layer of the neural net takes from the input vector one variable each and has two proceeding hidden layers and an output layer. The number of nodes in the hidden layer is set to around 100. The neural network is trained using the high data from the inverse approach and is then applied to a different RANS simulation to correct this and thereby improve the accuracy of the result.

In Figure 2.6 an example of the application of this trained neural network is shown. In this figure the results are compared from the inverse approach figure 2.6(a) (data not in the training set) and the neural net prediction Figure 2.6(b). In Figure 2.6(c) the pressure coefficient is compared between a baseline simulation, experimental data, the full inverse problem and the neural net corrected solution. What can be seen is that inverse approach matches with the experimental data is is expected and that the neural net correction comes close to the result but does not exactly match. What can be concluded is that with the correction there is a definite improvement over the base SA model.

The downside of this approach is that the inverse approach is computationally expensive for larger problems and that the training of the neural network will require a lot of computational power when full scale 3D problem data sets are trained and the dimensionality of the problem is increased.

2.3.3. Deep Neural Networks with Embedded Invariance

Pope [16] derived a new effective viscosity hypothesis. In this hypothesis he derived a basis of tensors and invariants of which a combination should be capable of representing the anisotropic part of the Reynolds stress. This theory and the tensor basis is described in section 3.2.1. It forms the basis for the work of Ling Kurzawski and Templeton [11]. They were the first to combine this tensor basis with a feed-forward neural network and compared this approach to that of standard linear eddy viscosity models and of the results of a neural net without the tensor basis.

The goal of the network in the Tensor-Basis approach is to determine the scalar coefficients $g^{(n)}$ from (3.38). The neural network that does this has a layout as shown in figure 2.7, with an input layer containing the invariants $I_1 - I_5$, which is followed by a number of hidden layers. The final hidden layer has 10 nodes for the 10 tensors which are combined in the merge output layer.

The neural networks were trained on a database of nine flows. In this database are DNS data-sets or well-resolved LES data-sets, as well as RANS results. The inputs to the neural network are the RANS results as this is the only known information when doing a RANS simulation with the neural network.

The trained tensor basis neural network and the network without the tensor basis were used to predict the Reynolds stress anisotropy tensor. The results for these predictions are compared to linear and quadratic eddy-viscosity models and can be seen in Figure 2.8. The figure shows the improved predictions of the tensor basis neural network over the linear and quadratic eddy viscosity models and also the better predictive capability when compared to the neural network without the tensor basis.

To truly asses the performance of a neural network augmented turbulence models it was of interest to

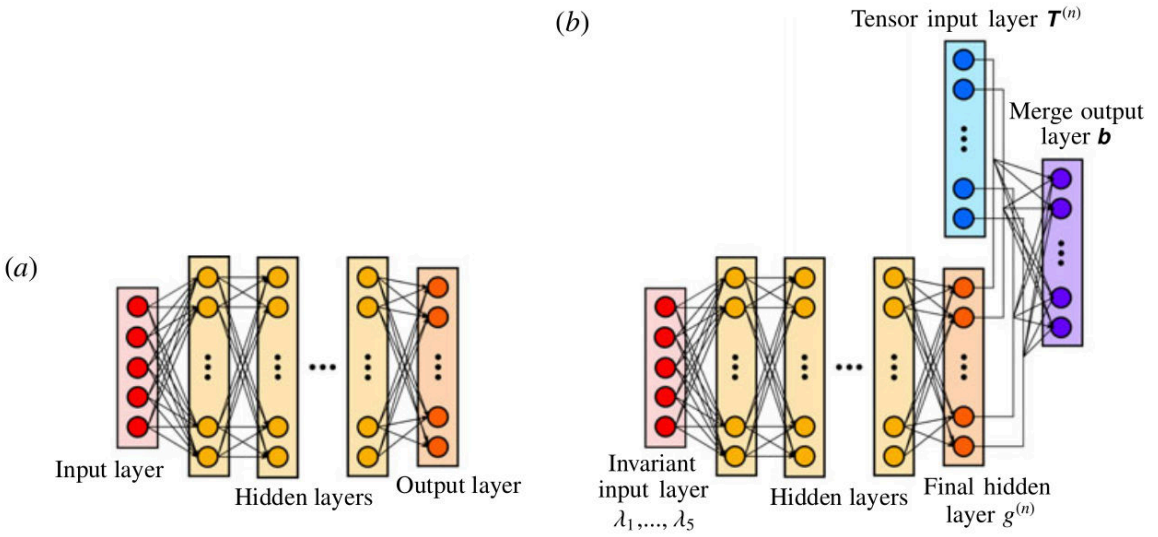


Figure 2.7: Schematic of Feed forward neural network architectures, with and without Tensor basis. Figure taken from [11].

test the predictive capabilities for predicting mean flow velocities. Therefore the Reynolds stress anisotropy tensor predicted by the tensor basis neural network was implemented in a RANS solver. The solver was first allowed to converge to a result after which the found Reynolds anisotropy was added and it was allowed to re-converge. The results of this approach showed an improvement in the results of a flow over a wavy wall where separation occurs. It was compared to the prediction of a linear and quadratic eddy viscosity model, the DNS data and a RANS model with the exact Reynolds stress anisotropy from the DNS inserted. The linear and quadratic eddy viscosity models fail to predict the flow separation that occurs on the wall. The DNS inserted anisotropy RANS correctly captures the shape and size of the separation while the tensor basis neural network predicted anisotropy augmented RANS successfully captures the separation but underestimates the size of the separation.

2.3.4. Stochastic Random Forest with Invariance

A second method that uses the tensor basis approach is that presented by Kaandorp and Dwight [6]. They have used a Random Forest machine learning algorithm to determine the coefficients of the tensor basis. For this approach the random forest method had to be modified to work with the Tensor basis.

The training of the algorithm happens just as in the Neural net approach using LES and DNS data. The final goal is to find a nonlinear eddy-viscosity model. The training takes the invariant features from a converged $k-\omega$ simulation. And the target, the anisotropy Tensor comes from the high fidelity simulation. This approach is a corrective approach where a single machine learning prediction provides an update to the solution. It is also possible to apply the network in an iterative manner such that at each solver iteration the machine learning algorithm updates the solution. In that case the training should be done on the DNS mean flow solution and not on the RANS mean flow, as the assumption is made that the RANS model with the machine learned correction converges to the DNS solution. This approach is however more expensive, as the machine learning model is applied each iteration. How expensive this is depends on the type of ML model.

A decision tree bases its predictions on a series of if-then tests on the input. The Random forest consist of a database of decision trees. Each tree has some randomised component that is making the difference between trees. The training of the tree happens by splitting the feature space recursively into two bins. A constant value is used to approximate the training data in each bin. The training data consists of $\mathbf{X} \in \mathbb{R}^{pxN}$ point locations of inputs and $\mathbf{y} \in \mathbb{R}^N$ scalar targets. The splitting is done in such a way that it minimises the mismatch between the response $\mathbf{y}$ and the approximation of the response in both bins (2.23). In this equation $\mathbf{y}_i$ is the response at $\mathbf{X}_i$. The goal is to find the constants C_L, C_R, which basically results in averaging of the

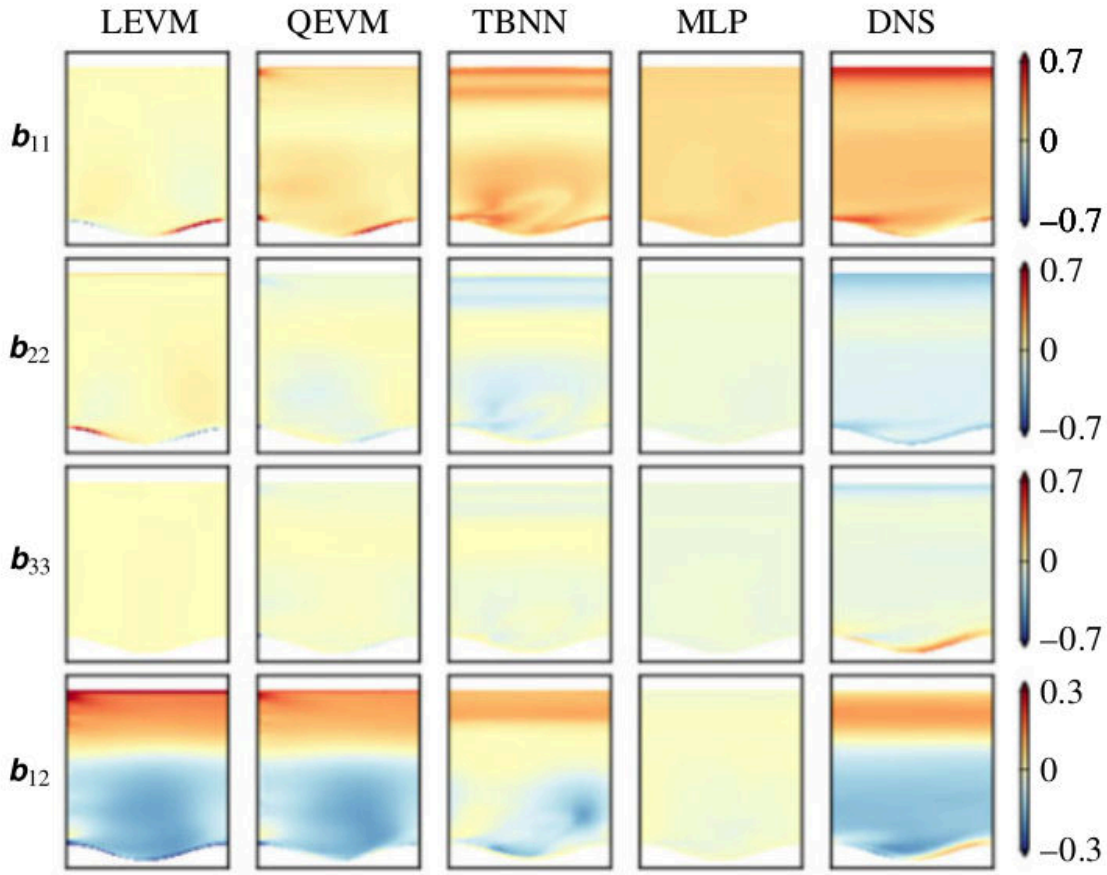


Figure 2.8: Predictions of Reynolds stress anisotropy tensor b_{ij} on a wavy wall testcase. LEVM=linear eddy viscosity model, QEVM= quadratic eddy viscosity model, TBNN=tensor basis neural network, MLP=multi-layer perceptron network (NN without tensor basis). Figure taken from [11].

target over the bins R_R and R_L, minimising the variance in both bins.

$$\min_{j,s}\left[\min_{C_L\in\mathbb{R}}\sum_{x_i\in R_L(j,s)}(\mathbf{y}_i-C_L)^2+\min_{C_R\in R}\sum_{x_i\in R_R(j,s)}(\mathbf{y}_i-c_R)^2\right] \tag{2.23}$$

The outer minimisation loop is solved using a brute-force calculation over the features j, and a one dimensional optimisation over s. This method is then reapplied over the new data-set in each of the bins.

In comparison to a general CART decision tree algorithm the new tensor basis decision tree algorithm is very similar. The difference being that the output is not constant values to approximate the anisotropy tensor, but it outputs coefficients for the tensor basis $g^{(m)}$. The equation that is being solved is (2.24), where i is the index of the samples, $\mathbf{b}_i$. is the anisotropy tensor coming from the high fidelity data and $g_L^{(m)}$ and $g_R^{(m)}$ are the tensor basis coefficients in the left and right bins. The norm that is being used is the Frobenius norm.

$$\min_{j,s}\left[\min_{g_L^{(m)}\in\mathbb{R}^{10}}\sum_{x_i\in R_L(j,s)}\left|\left|\sum_{m=1}^{10}\mathbf{T}_i^{(m)}g_L^{(m)}-\mathbf{b}_i\right|\right|_F^2+\min_{g_R^{(m)}\in\mathbb{R}^{10}}\sum_{x_i\in R_R(j,s)}\left|\left|\sum_{m=1}\mathbf{T}_i^{(m)}g_R^{(m)}-\mathbf{b}_i\right|\right|_F^2\right] \tag{2.24}$$

Finding the coefficients $g_L^{(m)}$ and $g_R^{(m)}$ is now a matter of solving two least-squares problems. This must be repeated for every j and every optimisation of s. This is done explicitly by flattening the tensor at each point as shown in (2.25). Where each of the minimisation problems over $\mathbf{g}$ becomes $\min_{\mathbf{g}} J$ where g is defined as in (2.26), which can be solved separately for R_L and R_R.

$$\hat{\mathbf{T}}_i = \begin{bmatrix} [\mathbf{T}_i^{(1)}]_{11} & [\mathbf{T}_i^{(2)}]_{11} & \dots & [\mathbf{T}_i^{(10)}]_{11} \\ [\mathbf{T}_i^{(1)}]_{12} & [\mathbf{T}_i^{(2)}]_{12} & \dots & [\mathbf{T}_i^{(10)}]_{12} \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{T}_i^{(1)}]_{33} & [\mathbf{T}_i^{(2)}]_{33} & \dots & [\mathbf{T}_i^{(10)}]_{33} \end{bmatrix}, \quad \hat{\mathbf{b}}_i = \begin{bmatrix} [\mathbf{b}_i]_{11} \\ [\mathbf{b}_i]_{12} \\ \vdots \\ [\mathbf{b}_i]_{33} \end{bmatrix} \tag{2.25}$$

$$J = \sum_{i+1}^{N} ||\hat{\mathbf{T}}_i \mathbf{g} - \hat{\mathbf{b}}_i||^2 \tag{2.26}$$

The tensor bases random forest is the result of averaging multiple decision trees. These trees are trained on a collection of bagged data sets. Bagging means randomly drawing with replacement from the full data-set, the idea is that this reduces the variance of the predictor by combining a number of high-variance, low-bias estimators. This is supported by adding some randomness into the individual decision trees. For splitting the data-set into training and validation data the bagging is used. As random samples are drawn until the original size of the data-set is reached there are some samples left, which form the validation data-set, also called the out-of-bag (OoB) samples.

For the decision trees, the inputs that are being used are the tensors and invariants derived by Pope [16]. These are sufficient to describe any possible tensor function. In this approach the assumption is relaxed and more input features are used. The total list of inputs are the Strain and rotation rate tensors and the gradient of the turbulent kinetic energy which is normalised and transformed into an anti-symmetric tensor: $\mathbf{A}_k = -\mathbf{I} \times \nabla k$. Besides that nine extra scalar features are used, such as the wall-distance based Reynolds number.

Set	Features	Normalization	Comment
FS1	$\mathbf{S_{ij}}^2, \mathbf{S_{ij}}^3, \mathbf{R_{ij}}^3, \mathbf{R_{ij}}^2, \mathbf{R_{ij}}^2\mathbf{S_{ij}}$, $\mathbf{R_{ij}}^2\mathbf{S_{ij}}^2, \mathbf{R_{ij}}^2\mathbf{S_{ij}}\mathbf{R_{ij}}\mathbf{S_{ij}}^2$	-	Invariant set based on $\mathbf{S_{ij}}$ and$\mathbf{R_{ij}}$.
FS2	$\mathbf{A_K}^2, \mathbf{A_k}^2\mathbf{S_{ij}}, \mathbf{A_k}^2\mathbf{S_{ij}}^2$, $\mathbf{A_k}^2\mathbf{S_{ij}}\mathbf{A_k}\mathbf{S_{ij}}^2, \mathbf{R_{ij}}\mathbf{A_k}$, $\mathbf{R_{ij}}\mathbf{A_k}\mathbf{S_{ij}}, \mathbf{R_{ij}}\mathbf{A_k}\mathbf{S_{ij}}^2$, $\mathbf{R_{ij}}^2\mathbf{A_k}\mathbf{S_{ij}}^*, \mathbf{R}^2\mathbf{A_k}\mathbf{S_{ij}}^{2*}$, $\mathbf{R_{ij}}^2\mathbf{S_{ij}}\mathbf{A_k}\mathbf{S_{ij}}^{2*}$	-	Added invariants when including $\nabla\mathbf{k}$
FS3	$\frac{1}{2}(\|\mathbf{R_{ij}}^2 - \mathbf{S_{ij}}^2\|)$	$\|\mathbf{S_{ij}}\|^2$	Ratio of excess rotation rate to strain rate
	$k^\dagger$	$\frac{1}{2}\bar{u}_i\bar{u}_i$	Turbulence intensity
	$\min\left(\frac{\sqrt{k}d}{50}, 2\right)$	-	Wall-distance based Reynolds number
	$\frac{k}{\epsilon}$	$\frac{1}{\|\|S_{ij}\|\|}$	Ratio of turbulent time scale to mean strain time scale
	$\sqrt{\frac{\partial p}{\partial x_i}\frac{\partial p}{\partial x_i}}$	$\frac{1}{2}\rho\frac{\partial}{\partial x_k}\bar{u}_k^2$	Ratio of pressure normal stresses to shear stresses
	$\bar{u}_i\frac{\partial k}{\partial x_i}^\dagger$	$\|\overline{u'_j u'_k} S_{jk}\|$	Ratio of convection to production of TKE
	$\|\overline{u'_i u'_j}\|$	k	Ratio of total to normal Reynolds stresses
	$\left\|\bar{u}_i\bar{u}_j\frac{\partial\bar{u}_i}{\partial x_j}\right\|^\dagger$	$\sqrt{\bar{u}_l\bar{u}_l\bar{u}_l\frac{\partial\bar{u}_i}{\partial x_j}\bar{u}_k\frac{\partial\bar{u}_k}{\partial x_j}}$	Non-orthogonality between velocity and its gradient

Table 2.1: Features used for the machine learning algorithms, obtained from [25] and [24]. For features with an * all cyclic permutations of labels of anti-symmetric tensors need to be taken in account. For FS1 and FS2 the trace of the tensor quantitites is taken. Features marked with † are rotationally invariant but not Galilean invariant. Table taken from [6]

For the regression trees, outlier filtering has been used. The reason for this is that decision trees are sensitive to small changes in the data, this is shown during testing as the regression tree can produce irregular and inconsistent predictions in small regions of the spatial domain. Both regularization and pruning were applied to prevent this problem, but an outlier filter proved to be a more robust way of eliminating data-sensitivity. The outlier filtering is described in (2.27). Finally a Gaussian filter is used to smooth the prediction spatially before inserting them into the N-S equation.

$$\frac{|y_i - \text{med}(y)|}{\text{med}(|\mathbf{y} - \text{med}(\mathbf{y})|} > \theta_{\text{med}} \tag{2.27}$$

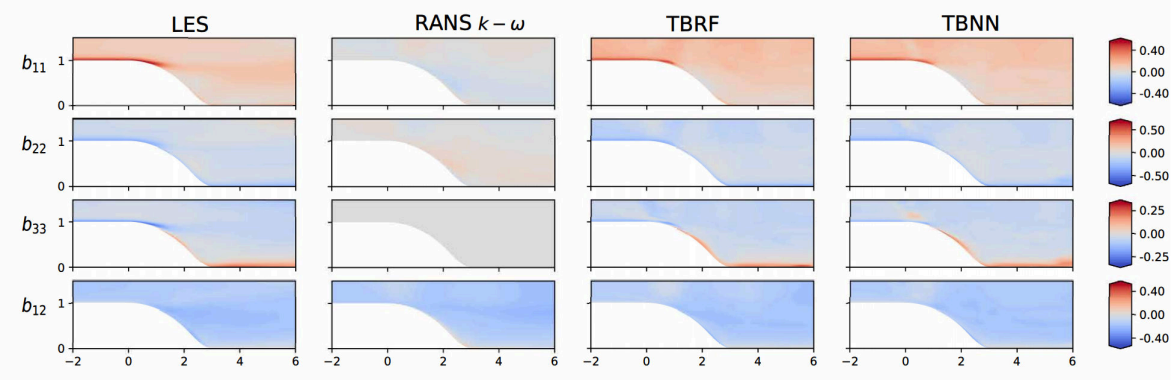


Figure 2.9: The anisotropy tensor components for the curved backward facing step from LES, RANS, TBRF and TBNN. Figure taken from [6].

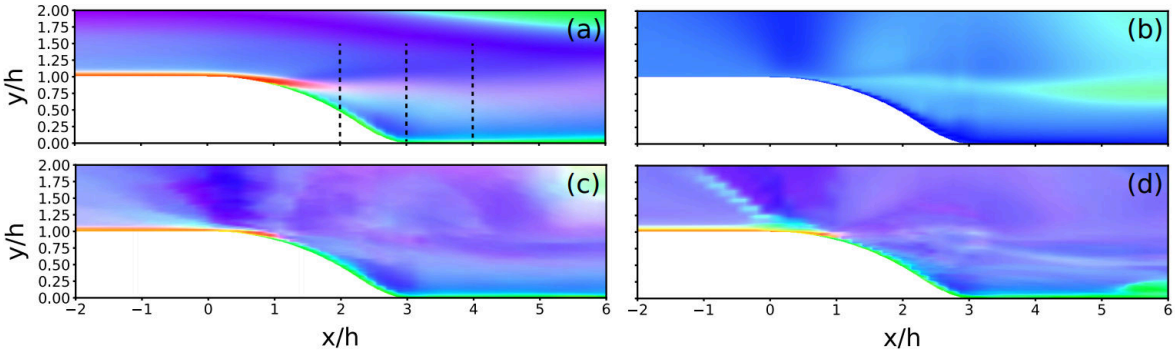


Figure 2.10: The ansitropy tensor displayed through barycentric coordinates for the curved backward facing step from LES (a), RANS (b), TBRF (c) and TBNN (d). Figure taken from [6].

For the insertion of the correction into the N-S equations OpenFOAM was used. The implementation of the predicted anisotropy field is done in a blending way, such that the Boussinesq approximation is slowly replaced by the ML predicted anisotropy field (2.28), Where $\mathbf{b}_B := \nu_t \hat{\mathbf{S}}$. The ramping is introduced acording to (2.29), where $\gamma_{max} \leq 0.8$ was achieved for all simulations and where n_{max} is the number of iterations after which the value of γ is fixed.

$$\tau \simeq \tau_{ML}(\gamma) := \frac{2}{3}k\mathbf{I} + 2k[(1-\gamma)\mathbf{b}_B + \gamma \mathbf{b}_{ML}] \tag{2.28}$$

$$\gamma_n = \gamma_{max} \min\left\{1, \frac{n}{n_{max}}\right\} \tag{2.29}$$

The test cases that have been used to prove this method are the Periodic hills case (PH), the Converging-diverging channel (CD), the curved backward-facing step (CBFS), the backward-facing step (BFS) and the square duct. For each of the cases either DNS or highly resolved LES data was available.

In Figure 2.9 the predicted anisotropy tensor components are compared to the 'true' LES values, the standard $k-\omega$ RANS model and the prediction from the tensor basis neural net that was developed by Ling et al. [11]. What can be concluded is that the standard RANS only is capable to predict the b_{12} tensor component effectively. All the other components are not predicted correctly at all, this is mainly due to the Boussinesq approximation that causes the turbulence anisotropy tensor to be aligned with the mean rate of strain tensor. Besides that it can be seen that the tensor basis random forest produces smoother and more accurate results compared to the tensor basis neural net. This becomes even clearer when using the Barycentric colormap as in Figure 2.10, here it can be seen that the boundary layer anisotropy is much more similar to that of the LES and that it also produces smoother results than the TBNN.

In conclusion the tensor basis random forest performs on par with the tensor basis neural network. As it improves the accuracy of the predicted velocities and shows results very close to the DNS data generally for all test cases.

The disadvantages of the methods described previously is that they are difficult to implement and expensive to train. The neural network from Ling et al. [11] acts as a single correction after the RANS solver has converged. In this the assumption is made that the correct anisotropy tensor can be found from a converged standard RANS solution. Next to that the assumption is made that once this found anisotropy tensor is inserted into a RANS solver the solution converges towards the high fidelity data. The main limitation of this method is the extrapolating qualities of the neural net. Since the network has been trained on a specific flow it is not necessarily true that the correct anisotropy can be found from the RANS mean flow. At least not in a general sense as the discrepancy between high fidelity data and the RANS mean flow might not be correlated at all and depends on many different factors.

2.3.5. Evolutionary algorithm

Weatheritt and Sandberg [26] have used genetic evolutionary programming (GEP) to find a model that predicts the Reynolds stress anisotropy a_{ij}. The GEP is similar to genetic programming, both methods revolve around iteratively improving a population of candidate solutions by making random changes and keeping the solution that fits best, the 'fittest'. The GEP is a tool that can be used for multiple applications but is used in this application as a symbolic regression tool that produces constraint-free solutions.

The overall goal of the evolutionary programming is to find the model which fits the data best. The fitness of a a model is defined as per (2.30). Where the maximum fitness is equal to 1.

$$Fit(f^{\text{guess}}) = 1 - \frac{1}{n}\sum_{k=1}^{n}\frac{||f^{\text{guess}}(x_k, y_k) - f_k||}{||f_k||} \tag{2.30}$$

The GEP is used to form new models for a correction that is applied to a slightly modified $k-\omega-SST$ turbulence model. For which the modified k and ω transport equations are shown.

$$\partial_t k + U_j \partial_{x_j} k = P_k - \epsilon + \partial_{x_j}\left[(\nu + \sigma_k \nu_t)\partial_{x_j} k\right] \tag{2.31}$$

$$\partial_t \omega + U_j \partial_{x_j} \omega = \frac{\gamma\omega}{k} P_k - \beta\omega^2 + \partial_{x_j}\left[(\nu + \sigma_\omega \nu_t)\partial_{x_j}\omega\right] + \sigma_d CD_{k\omega} \tag{2.32}$$

Where a_{ij}^x is the anisotropy correction predicted by the GEP. (2.33), (2.34).

$$\tau_{ij} = \frac{2}{3}\delta_{ij}k - 2\nu_t S_{ij} + a_{ij}^x \tag{2.33}$$

$$a_{ij} = a_{ij}(k, \epsilon, S_{ij}, R_{ij}) \tag{2.34}$$

The GEP method consists of 4 steps:

1. update/create population P^i
2. selection
3. reproduction
4. genetic operators

In the first steps models are formed existing of individual expression trees (ET), as shown in (2.35) and Figure 2.11(a). The total algorithm is build up out of these block by combining these ETs, resulting in multi-genetics as shown in Figure 2.11(b). From these complete models the best fitting model is selected by testing using the training data-set. After that the selected models are altered by reproduction and genetic operations such as mutations.

$$f^{\text{guess}}(x, y) = cos(2 - y)(x + 4) \tag{2.35}$$

For the application in turbulence modelling some alterations had to be made since the standard GEP algorithm works well for scalar field regression. However for tensor regression problems it easily produces

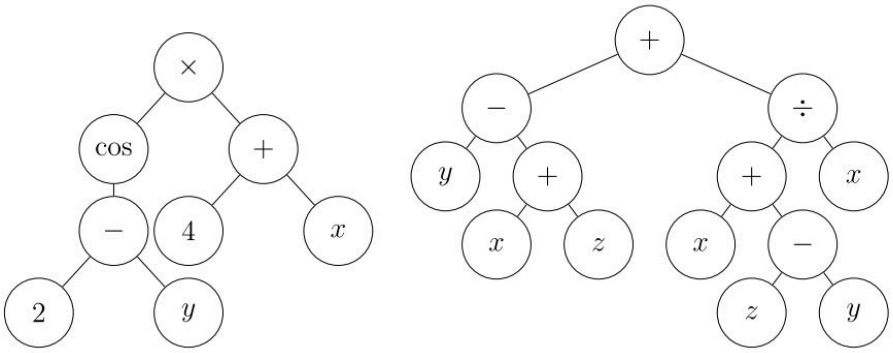


Figure 2.11: (a) Example ET corresponding to (2.35) (b) Example multigenic ET. Figures taken from [26].

'dimensionally invalid' solutions. For tensors the goal is to fit a function of the form $\tau_{ij} = \tau_{ij}(A_{ij}, B_{ij}, x, y) \in \mathbb{R}^{3\times3}$ to the data-set of high fidelity data.

$$\tau_{ij}^{\text{guess}} = \sum_n \alpha_n T_{ij}^n \tag{2.36}$$

Where α_n are the coefficients that need to be optimised and V_{ij}^n are the tensor basis. This is where the tensor basis from Pope [16] comes in. It was chosen in this approach to use the first 4 Tensors and 2 Invariants from the 10 tensors and 5 invariants.

With the reformulation of the problem into tensor form the Fitness expression has also been updated (2.37).

$$Fit(T_{ij}^{\text{guess}}) = \sum_{k=1}^{n} \frac{T_{ij,k} T_{ij,k}^{\text{guess}}}{T_{mn,k} T_{mn,k} T_{pq,k}^{\text{guess}}} \tag{2.37}$$

The result of the evolutionary algorithm are equations of the tensor basis form. For this research the Periodic hill and the backward facing step has been used as test cases.

To asses the quality of the models the evolutionary algorithm has produced the fit to do this the anisotropy is compared to reference data from the LES and DNS along a certain x-distance in the domain (2.38). To get a value for the entire slice this expression is integrated over the entire vertical distance. $\rho_{a,case} = \int_y \rho_a(y) dy$. An other metric of the models performance is the ratio of the normal stresses $\tau_r = \tau_{22}/\tau_1 1$.

$$\rho_a = \rho_a(y) = \frac{a_{ij} a_{ji}^{\text{ref}}}{a_{mn} a_{nm} a_{pq}^{\text{ref}}} \tag{2.38}$$

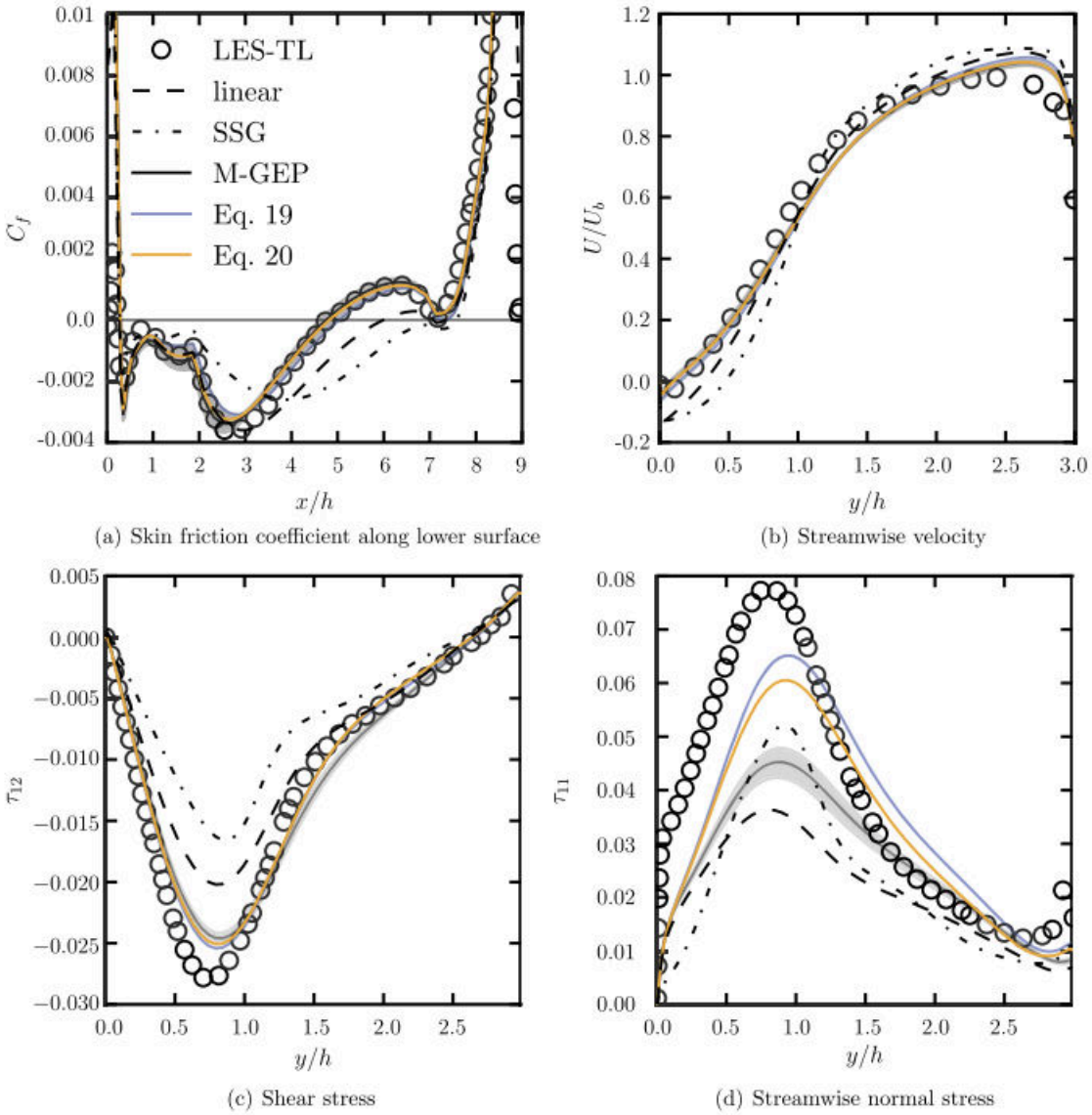
In Figure 2.12 Some results are shown for the periodic hills case. In this figure the results from the baseline, the SSG model are compared to two of the equations that were found by the Genetic programming algorithm equations (2.39), (2.40). These two models have been found independently from each other. In the figures these equations are referred to as Eq. 19 and Eq. 20.

$$a_{ij}^x 2\nu_t T_{ij}^1 \Big[0.02 - 0.0648\tau_I^2\Big(3I_1 + I_2 + 0.0162\tau_1^2 I_1^2\Big)\Big] - 0.18\nu_t \Big[T_{ij}^2 + T_{ij}^3 + 2T_{ij}^4\Big] \tag{2.39}$$

$$a_{ij}^x 2\nu_t T_{ij}^1 \Big[0.01 - 0.0324\tau_I^2\Big(3I_1 - I_2\Big)\Big] - 0.18\nu_t \Big[T_{ij}^2 + T_{ij}^3 + 2T_{ij}^4\Big] \tag{2.40}$$

What can be seen is that compared to a linear SST model and the SSG model the found correction models (2.39), (2.40). It can be seen that for all variables the results from the correction models move the results closer to the LES values. Especially the skin friction coefficient sees a good correspondence with the LES data. In Figure 2.13 more results are shown for the periodic hills case, but in off design conditions. These conditions are different to the training flow and therefore show the extrapolating capabilities of the method

What can be seen is that the velocity and shear stress profiles in 'on design conditions from Figure 2.12 (b-c) have small confidence intervals for the mean solution of the GEP models. However, as can be seen in



(a) Skin friction coefficient along lower surface

(b) Streamwise velocity

(c) Shear stress

(d) Streamwise normal stress

Figure 2.12: Statistical results from the periodic hills case. Profile for stress and velocity is $x/h = 4.0$. The M-GEP sample is represented as a 99.9% confidence interval for the mean. Figure taken from [26].

Figure 2.13 this confidence interval grows significantly as the case is in off-design conditions and this is also seen in the results that are less accurate than the baseline models for the shear stress.

This new method has proven to improve RANS simulations for 2D flow cases it shows to be a useful predictive tool. The need for training on data-sets with a various number of flow cases is shown. As models trained on a single flow regime should not be activated when applied to another regime.

In later work Weatheritt and Sandberg [27] have applied this approach to a diffuser where it showed similar improvements in simulation accuracy. with this showed good extrapolating performance, making this method a useful tool for engineering simulations.

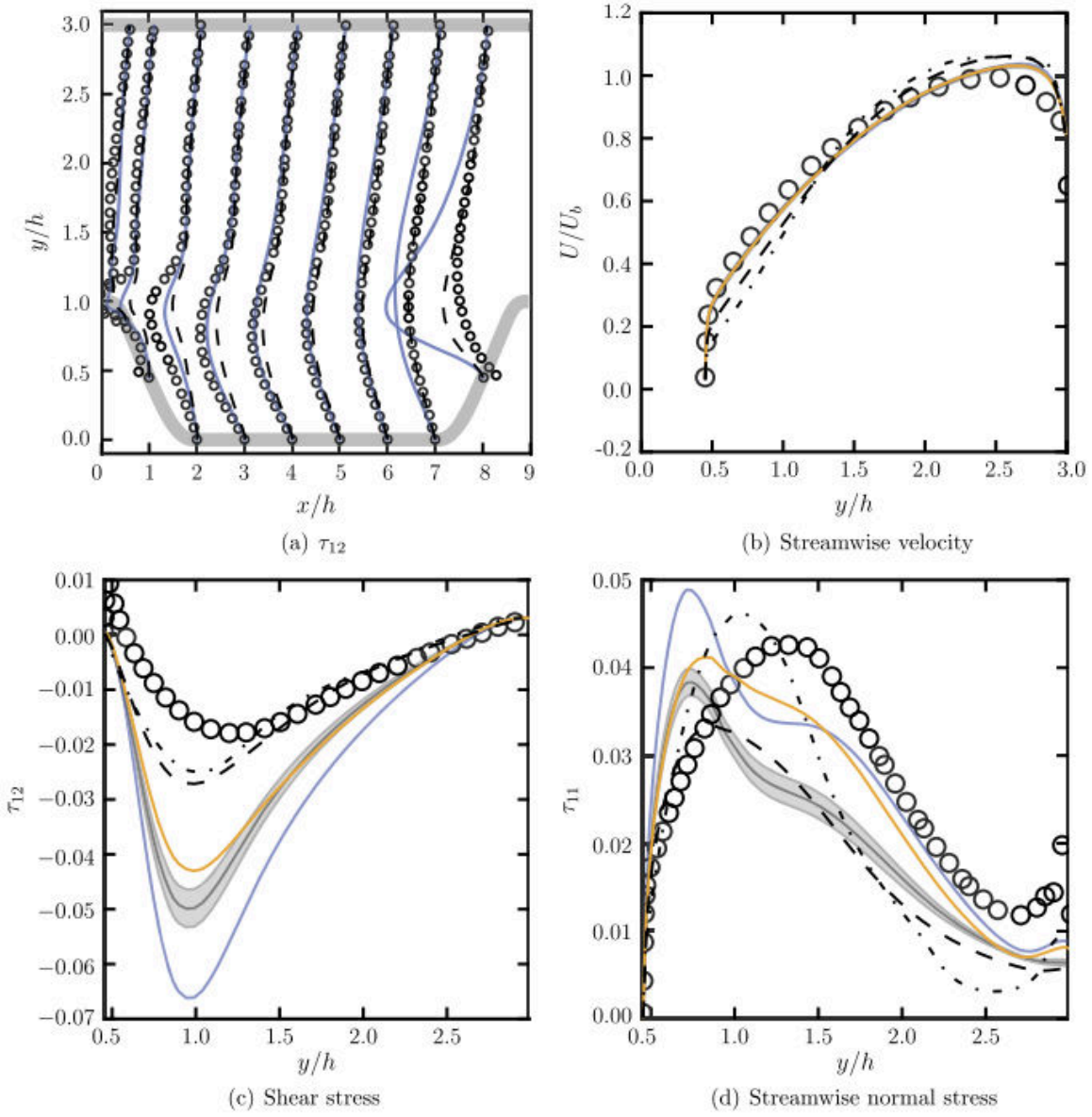


Figure 2.13: Flow quantities in 'off-design' conditions. (a) highlights on-design vs off-design Statistical results from the periodic hills case. Profile for stress and velocity is $x/h = 4.0$. The M-GEP sample is represented as a 99.9% confidence interval for the mean. Figure taken from [26].

2.3.6. Deterministic Symbolic Regression

To improve the extrapolating capabilities Schemlzer and Dwight [18] came up with a novel approach to find new turbulence models by discovering additional corrections to already existing turbulence models. The approach they presented is called Sparse Regression of Turbulent Stress Anisotropy and forms the basis for this work. The theoretical basis for the method is described in Section 3.2. Here some results from the existing literature is presented as similarities between these results is visible with the results presented later in Chapter 4.

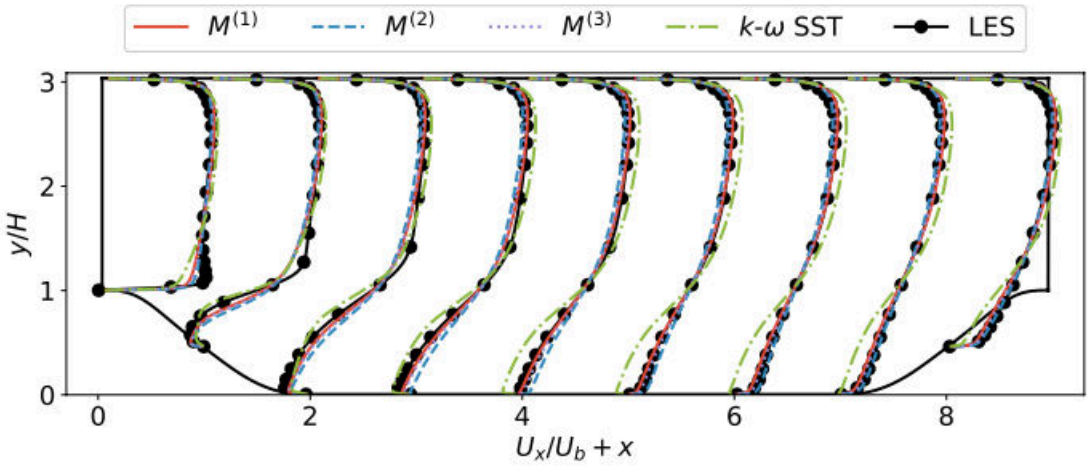
This method has been applied in [18] to three different test cases: the periodic hill at a Reynolds number of 10595, a converging diverging channel at Reynolds number 12600 and a curved backward facing step at a Reynolds number of 13700. For each of the cases a library of correction models was found, ranging in complexity with model coefficients of different magnitude. Three of the models that have been selected from each of the training cases are shown in (2.41), (2.42) and (2.43). These models have been selected based on a cross validation with CFD and have the lowest error when applied to all three test cases. It is possible to apply only the model for correcting R and for correcting b^{Δ}_{ij} separately.

$$\begin{aligned} M^{(1)}_{b^{\Delta}} &= (24.94 I^2_1 + 2.65 I^1_2)\, T^{(1)}_{ij} + 2.96 T^{(2)}_{ij}, \\ &\quad + \Big(2.49 I^1_2 + 20.05\Big) T^{(3)}_{ij} + \Big(2.49 I^1_1 + 14.93\Big) T - ij^{(4)}, \\ M^{(1)}_{R} &= 0.4\, T^{(1)}_{ij} \end{aligned} \tag{2.41}$$

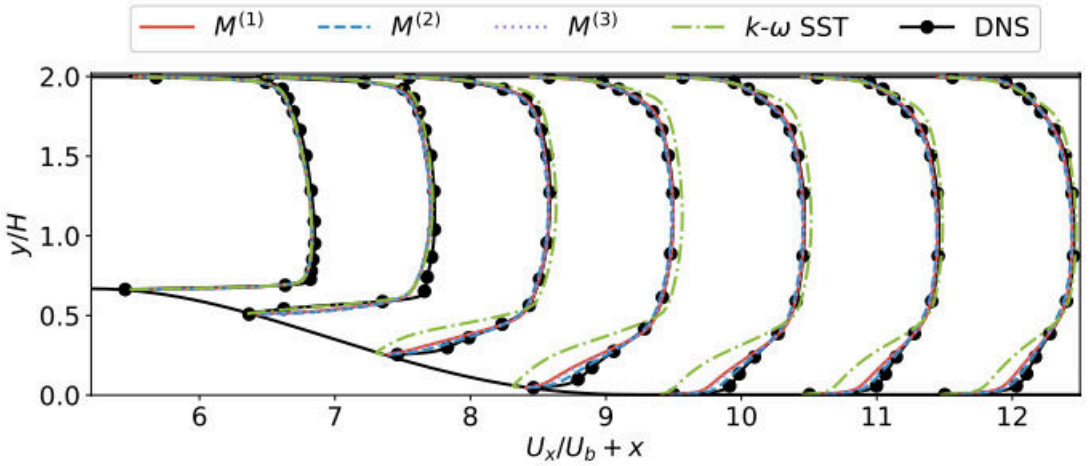
$$\begin{aligned} M^{(2)}_{b^{\Delta}} &= T ij^{(1)} \Big(0.46 I^1_2 + 11.68 I^2_1 - 0.30 I^2_2 + 0.37\Big) \\ &\quad + T^{(2)}_{ij} \Big(-12.25 I^1_1 - 0.63 I^2_2 + 8.23\Big) \\ &\quad + T^{(3)}_{ij} \Big(-1.36 I^2_1 - 2.44\Big) \\ &\quad + T^{(4)}_{ij} \Big(-1.36 I11 + 0.41 I^2_1 - 6.52\Big), \\ M^{(2)}_{R} &= 1.4\, T^{(1)}_{ij} \end{aligned} \tag{2.42}$$

$$\begin{aligned} M^{(3)}_{b^{\Delta}} &= T^{(1)}_{ij} \Big(0.11 I^1_1 I^2_1 + 0.27 I^1_1 I^2_2 - 0.13 I^1_1 I^2_3 + 0.07 I^1_1 I^2_4 \\ &\quad + 17.48 I^1_1 + 0.01 I^1_2 I^2_1 + 1.251 I^1_2 + 3.67 I^2_1 + 7.52 I^2_2 - 0.3\Big) \\ &\quad + T^{(2)}_{ij} \Big(0.17 I^1_1 I^2_2 - 0.16 I^1_1 I^2_3 - 36.25 I^1_1 - 2.39 I^1_2 + 19.22 I^2_1 + 7.04\Big) \\ &\quad + T^{(3)}_{ij} \Big(-0.22 I^1_2 + 1.8 I^2_1 + 0.07 I^2_2 + 2.65\Big) \\ &\quad + T^{(4)}_{ij} \Big(0.2 I^1_2 - 5.23 I^2_1 - 2.93\Big), \\ M^{(3)}_{R} &= 0.93\, T^{(1)}_{ij}, \end{aligned} \tag{2.43}$$

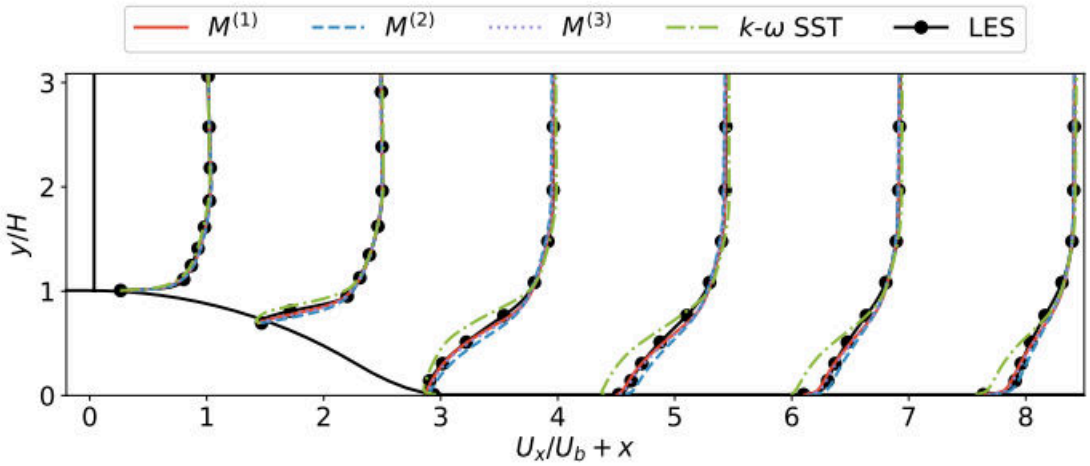
When these models are applied to the test cases an improvement is seen compared to baseline RANS. All models show an improvement, even though they are all trained on different flow cases. The results of these models can be seen in Figure 2.14. Here it can be seen that the model that is not trained on the case is the best performing model. Overall From this it can be concluded that the SpaRTA Method gives a systematic improvement over the baseline turbulence model, with the model discovery being relatively inexpensive.



(a) PH_{10595}



(b) CD_{12600}



(c) CBFS_{13700}

Figure 2.14: The predicted stream-wise velocity for the three different test cases [18]

2.3.7. Reynolds Force Vector

An approach that does not use the Tensor basis approach is that of Cruz et al. [1]. They use a method with which they try to circumvent the issue that there is a discrepancy between the Reynolds stress calculated from the DNS data and the real Reynolds stress.

$$\mathbf{R} = \mathbf{R}_\theta + \delta_{\mathbf{R}_\theta} \tag{2.44}$$

According to Thompson et al. [23] this discrepancy is critical when being used to recover basic quantities of interest. In their work they show that due the fact that the Reynolds stress being a second order statistic is less converged than the first order statistics. They evaluated the difference between reconstructing the mean velocity flow from a DNS which had more converged statistics than the other and noticed a significant difference between the two.

To circumvent this issue Cruz et al. try to use the Reynolds force vector, the divergence of the Reynolds stress. A field that can be constructed from first order statistics. This Reynolds force vector forms the target for the machine learning. The flow case that is used is a series of DNS solutions of square duct flow at Reynolds numbers ranging from 2200 to 3500.

The methodology that is proposed is based on the observation that the turbulence only influences the mean momentum balance equation through the divergence of the Reynolds stress. This is the reason for the name of Reynolds Force vector. The relation between the Reynolds force vector and the data is given by (2.45).

$$r \equiv \nabla \cdot \mathbf{R} = \nabla \cdot \left(\mathbf{R}_\theta + \delta_{\mathbf{R}_\theta}\right) = \mathbf{r}_\theta + \delta_{r\theta} \tag{2.45}$$

It is also possible to derive the Reynolds force vector from first order statistics as shown in (2.46). In case the mean pressure is not available in the data-set, a modified Reynolds force vector can be estimated by (2.47). For the injection of the correction the standard $k-\epsilon$ RANS model was used.

$$\mathbf{r}^\epsilon = \langle \mathbf{U} \rangle_\theta \cdot \nabla \langle \mathbf{U} \rangle_\theta - \nu \nabla^2 \langle \mathbf{U} \rangle_\theta + \frac{1}{\rho} \nabla \langle p \rangle_\theta \tag{2.46}$$

$$\mathbf{t}^\epsilon = \left[\mathbf{r} - \frac{1}{\rho} \nabla \langle p \rangle \right]^\epsilon = \langle \mathbf{U} \rangle_\theta \cdot \nabla \langle \mathbf{U} \rangle_\theta - \nu \nabla^2 \langle \mathbf{U} \rangle_\theta \tag{2.47}$$

The machine learning technique that was employed for training of the corrections is a feedforward neural network. The inputs and the targets are normalized before being used in the neural net training. The inputs that were used in the neural network are the strain rate and the non persistence-of-straining tensor. $P \equiv D \cdot \bar{W} - \bar{W} \cdot D$ where $\bar{W} \equiv W - \Omega^D$ and $D \equiv 0.5(L + L^T)$. These are retrieved from the solution of RANS simulations. In total 8 vectors and 8 tensors are used as inputs for the neural network, these can be found in table 2.2

Tensors ($\mathbf{T}_i$)	Vectors ($\nabla \cdot \mathbf{T}_i$)
$\mathbf{D}$	$\nabla \cdot \mathbf{D}$
$\mathbf{D}$	$\nabla \cdot \mathbf{P}$
$\mathbf{P}$	$\nabla \cdot \mathbf{D}^2$
$\mathbf{D}^2$	$\nabla \cdot \mathbf{P}^2$
$\mathbf{P}^2$	$\nabla \cdot (\mathbf{D} \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{D})$
$\mathbf{D} \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{D}$	$\nabla \cdot (\mathbf{D} \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{D})$
$\mathbf{D}^2 \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{D}^2$	$\nabla \cdot (\mathbf{D}^2 \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{D}^2)$
$\mathbf{P}^2 \cdot \mathbf{D} + \mathbf{D} \cdot \mathbf{P}^2$	$\nabla \cdot \mathbf{P}^2 \cdot \mathbf{D} + \mathbf{D} \cdot \mathbf{P^2})$
$\mathbf{R}$	$r \equiv \nabla \cdot \mathbf{R}$

Table 2.2: Inputs used for the training of the neural network to predict the Reynolds force vector [1].

For the training of the neural network 4 of the 6 RANS cases have been used. The output target for the training are the components of the discrepancy of the Reynolds stress tensor of the RANS simulation and the DNS data. The training data is split up into training and test data for the training of the neural

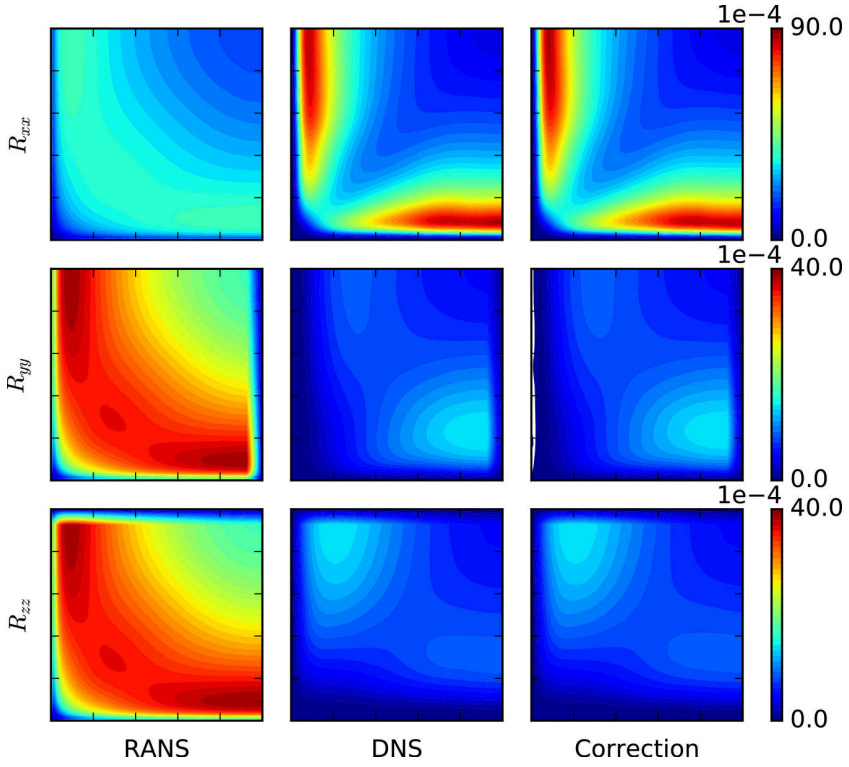


Figure 2.15: Comparison of the normal components of the Reynolds stress tensor for $Re = 2400$. Figure taken from [1].

network. Once the neural network has converged the result is tested. This is done by inserting the outputs from the neural network, the Reynolds Stress tensor and Reynolds force vector corrections inserted into the OpenFOAM platform. The velocity field comes from the double step procedure of neural net and insertion into the momentum balance equation.

In Figure 2.15 the results of the insertion of the Reynolds stress into the RANS solver can be found. What stands out is the large discrepancy between the baseline RANS and the DNS predictions and the accuracy of the NN prediction for the Tensor. The same can be seen in Figure 2.16, where the components of the Reynolds force vector are seen, again the default RANS prediction is far off from the DNS solution and the correction model approaches the DNS solution nicely though some discrepancies can be seen. In Figure 2.17 the velocity components can be seen from the baseline RANS, the DNS, using the Reynolds force vector correction and using the Reynolds stress correction. These results are shown for a Reynolds number of 2400. What can be seen in this figure is that the Reynolds force corrected velocity prediction is more accurate than the Reynolds stress correction for the Velocity in x direction. However, a drawback of using the Reynolds force vector is the loss of symmetry with respect to the diagonal. Comparing the y and z velocity components it is harder to give a definitive conclusion which learning method gives more accurate results.

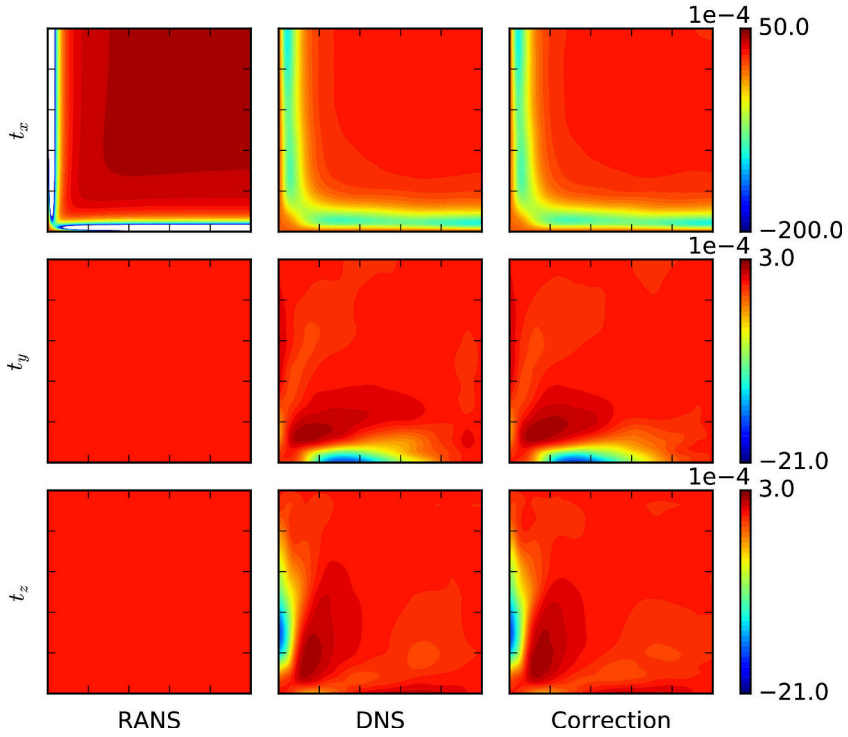


Figure 2.16: Comparison of the Reynolds force vector between RANS, DNS and the NN predicted correction for $Re = 2400$. Figure taken from [1].

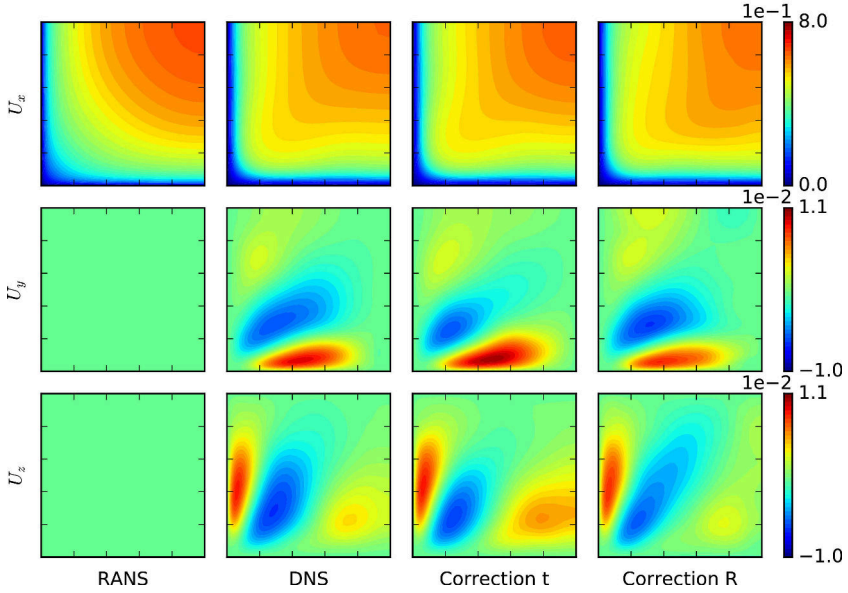
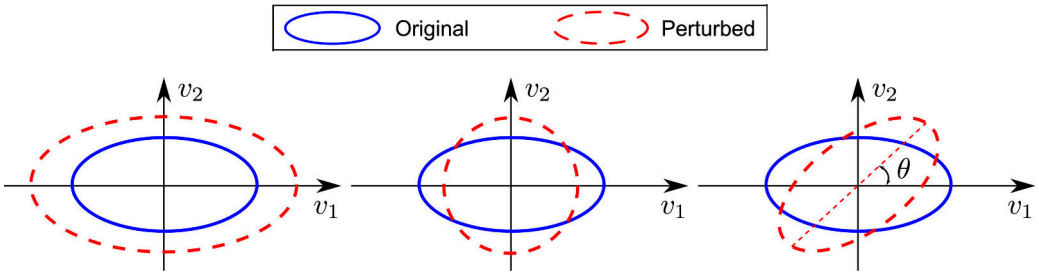


Figure 2.17: Comparison of the velocity components of baseline RANS, DNS, the Reynolds force vector correction and the Reynolds stress correction. $Re = 2400$. Figure taken from [1].



(a) TKE k (b) Eigenvalues $\mathbf{\Lambda}$ (c) Eigenvectors $\mathbf{V}$

Figure 2.18: The perturbations on (a) the magnitude (turbulent kinetic energy k), (b) eigenvalues Λ, and (c) eigenvectors $\mathbf{V}$ of a Reynolds stress tensor $\mathbf{R}$. For clarity, the tensor is represented in the two-dimensional space as an ellipsis instead of a three-dimensional ellipsoid. in addition, perturbations on k, Λ, and $\mathbf{V}$ are shown individually on each panel and not as three consecutive perturbations. Figure taken from [29].

2.3.8. Representation of stress tensor perturbations with application in machine-learning-assisted turbulence modelling

Wu, Sun, Laizet and Xiao, [29] take a different approach in correcting the Reynolds stress, they do this by means of representing the Reynolds stress in the Isotropic and anisotropic parts. The anisotropic part is then split up into its eigenvalues Λ and eigenvectors $\mathbf{V}$ (2.48). The eigenvectors $\mathbf{V}$ consist of three orthonormal vectors but have only three degrees of freedom.

$$2k\left(\frac{I}{3}+A\right)=2k\left(\frac{\mathbf{I}}{3}+\mathbf{V}\Lambda\mathbf{V}^T\right) \tag{2.48}$$

The discrepancy between the Reynolds stress retrieved from the DNS and the modelled Reynolds stress can be expressed in terms of three consecutive transformations, described by Wu et al as: "

1. The size of the ellipsoid (magnitude of the tensor) is scaled by a positive factor of $\gamma_k = k^*/k^{rans}$ while keeping the shape the same, as ilustrated in Figure 2.18 (a)

2. The aspect ratio of ellipsoid is perturbed while keeping the size (sum of the three axes) and orientation unchanged. This is achieved by perturbing the eigenvalues, i.e., $\Lambda^* = \Lambda + \Delta\Lambda$, as illustrated in Figure 2.18(b).

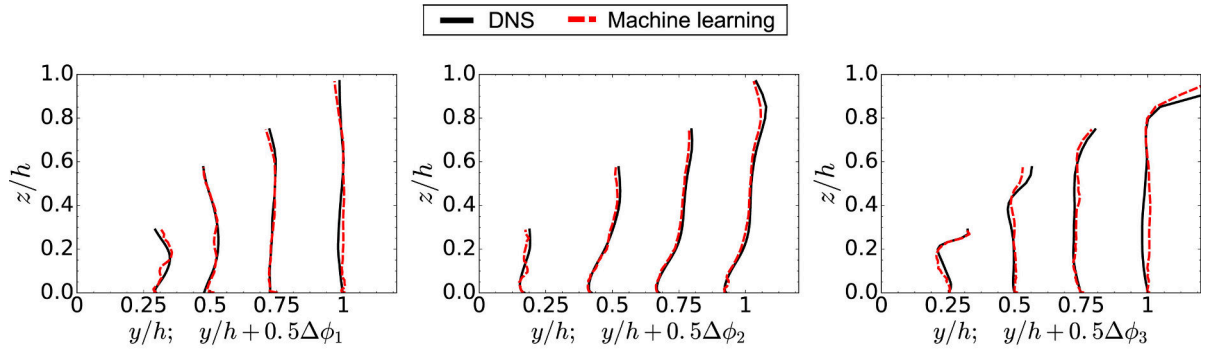
3. Finally, the ellipsoid experiences a rigid-body rotation, which is represented as $\mathbf{V}^* = \mathbf{QV}$, where $\mathbf{Q}$ is a rotationmatrix. This perturbation is illustrated in Fig. 2.18(c)."

Where the machine learning objective is to try and predict a mapping $f : \mathbf{q} \mapsto (\gamma, \Delta\Lambda, \mathbf{Q})$. The transformation that is then equal to $\mathbf{R}^* = 2\gamma_k k\left(\frac{\mathbf{I}}{3}+\mathbf{QV}(\lambda+\Delta\Lambda)\mathbf{V}^T\mathbf{Q}^T\right)$.

These transformations need to be represented in a machine learning environment. The most trivial transformation is using the rotation matrix Q. This matrix has 9 elements but only three intrinsic degrees of freedom due to the orthonormality constraint. This results in a number of constraints which is not trivial to implement in the machine learning algorithm. Therefore, it has been chosen to use a transformation with the same number of variables as the number of intrinsic degrees of freedom. The two options that satisfy this requirement are the Euler angles and unit quaternion.

For the regression function two requirements were imposed, namely the function should be smooth, such that it has a good generalisation performance. Secondly the function itself must be frame-independent, the similar requirement that holds for any other method applicable to RANS.

The Euler angles that are used in this system consist of three rotations (ϕ_1, ϕ_2, ϕ_3) round the $z-axis$, $x-axis$ and $z-axis$ respectively. It is used to describe the current orientation of a fixed body in an axis system. This system is referred to as the "absolute Euler angles". The discrepancy between these two sets of eigenvectors, $\mathbf{V}$ and $\mathbf{V}^*$, can be described by the Euler angles. The machine learning is trained for regressing functions for the Euler angles $\Delta\phi_\alpha$ that describe this transformation. However, there is one problem with this formulation, that is it is not frame independent. To solve this issue the choice was made to directly express the rotation from $\mathbf{V}$ to $\mathbf{V}^*$ in Euler angles $\phi_\alpha^{0\rightarrow\star}$. However these direct transformation angles are non smooth



(a) $\Delta\phi_1$ (b) $\Delta\phi_2$ (c) $\Delta\phi_3$

Figure 2.19: The prediction of the discrepancy-based Euler angles for the inflow in a square duct at Re = 3500, showing the the three components $\Delta\phi_{1-3}$, training flow $Re-2900$. Figure taken from [29].

spatially and are therefore not suitable due to the smoothness requirement.

Therefore, it has been chosen to use the unit quaternion, for which the expression is shown in (2.49). This representation describes a rotation from $\mathbf{V}$ to $\mathbf{V}^*$, which can be expressed in terms of a unit vector $[n_1, n_2, n_3]$ and an angle θ.

$$h = \left[\cos\frac{\theta}{2}, n_1\sin\frac{\theta}{2}, n_2\sin\frac{\theta}{2}, n_3\sin\frac{\theta}{2}\right]^T \tag{2.49}$$

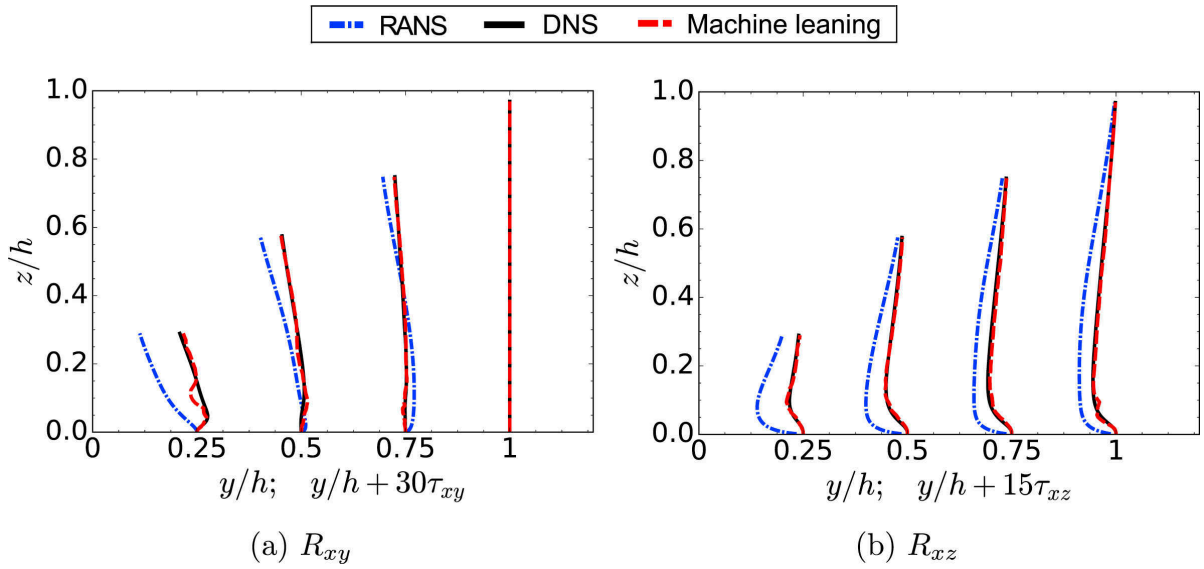
For the machine learing part the random forest regression was used to build mappings from the inputs, the 50 mean flow features and to the Reynolds stress discrepancies. For the forest 300 trees are used and the number of features at each split is set to 7. The predicted discrepancies from the random forest are compared to the true discrepancy between the DNS and the RANS model. From this test it was shown that the unit quaternion is the most accurate method when comparing to the Euler angle representations.

For the proper validation of the method, the random forest has been applied to a test case that is different from the training case. The training and test cases that were used were both periodic hills but with a different ramping geometry. The other two test geometries that were used are the square duct flow at $Re = 2900$ for training and $Re = 3500$ for testing and the same geometry but the coordinate system rotated by 60 degrees anti-clockwise.

Using the two different types of Euler angles described above it was shown that the discrepancy based Euler angles are spatially smooth but the direct-rotation-based Euler angles are not. However, the direct-rotation-based Euler angles are not frame independent as is illustrated in Figure 2.21. In Figure 2.19 the results for the machine learned prediction of the discrepancy based Euler angles are shown for the inflow in a square duct at a Reynolds number of 3500. The comparison is made with the actual angles based on the DNS data. The training was done on the same flow at $Re = 2900$.

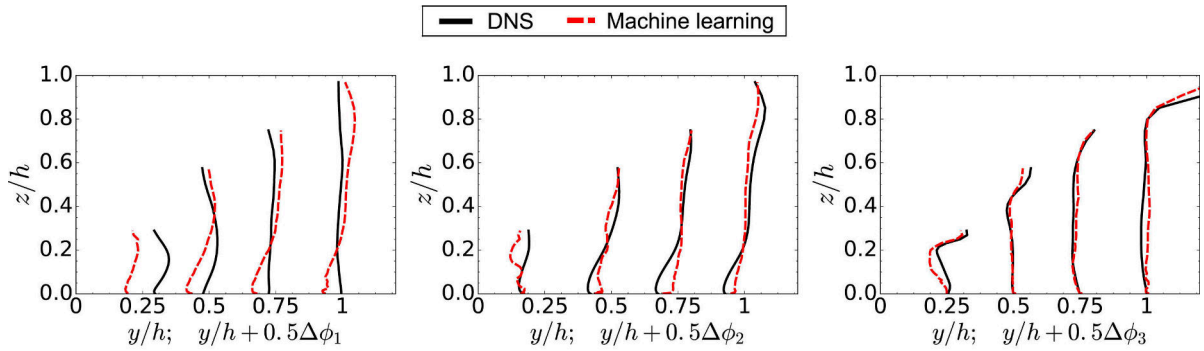
In Figure 2.20 the prediction of the discrepancy based Euler angles is shown, with the resulting predicted Reynolds stress components. It can be seen that the Machine learned model accurately predicts the Reynolds stress components, compared to baseline RANS. However, this brakes down when a rotation of the coordinate system of 60° is introduced as can be seen in Figure 2.22. In this figure it is shown that the Reynolds stress prediction of the machine learning model is worse that that of the baseline RANS. As is also confirmed by the plots shown in Figure 2.21.

In conclusion the only true option that fulfils all the requirements is the unit quaternion decomposition as a target for the machine learning. This method shows a good predictability in terms of the machine learning prediction and capability of correcting the flow velocity. As is seen in Figure 2.23 where the results for the periodic hills are plotted.



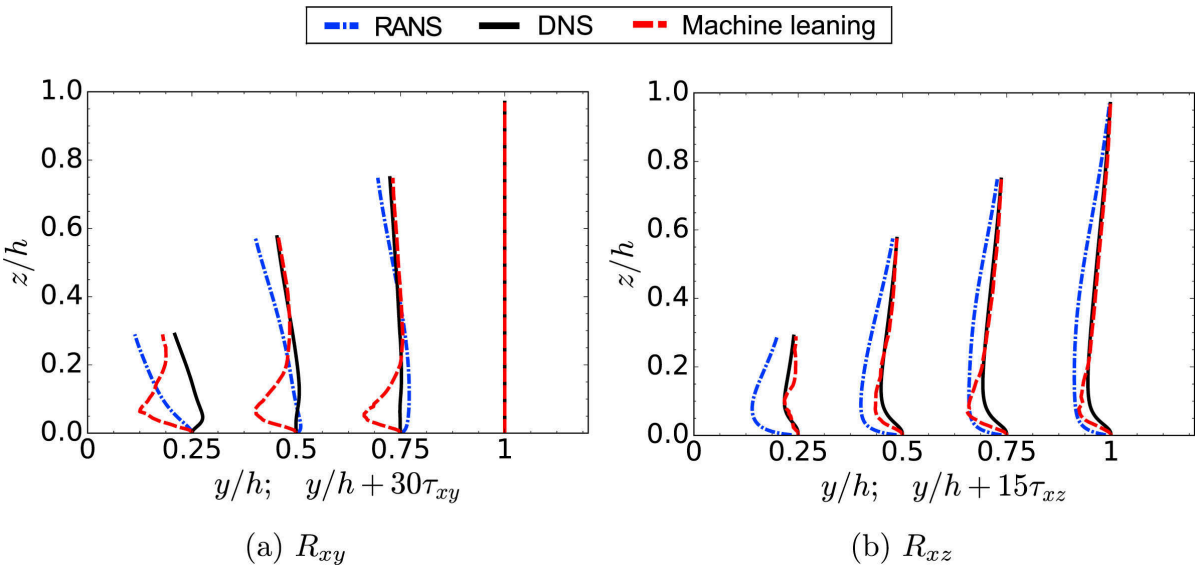
(a) R_{xy} (b) R_{xz}

Figure 2.20: The prediction of Reynolds stress components (a) R_{xy} and (b) R_{xz} based on the discrepancy-based Euler angles for the flow in a square duct at $Re = 3500$. The training flow is at $Re = 2900$. Figure taken from [29].



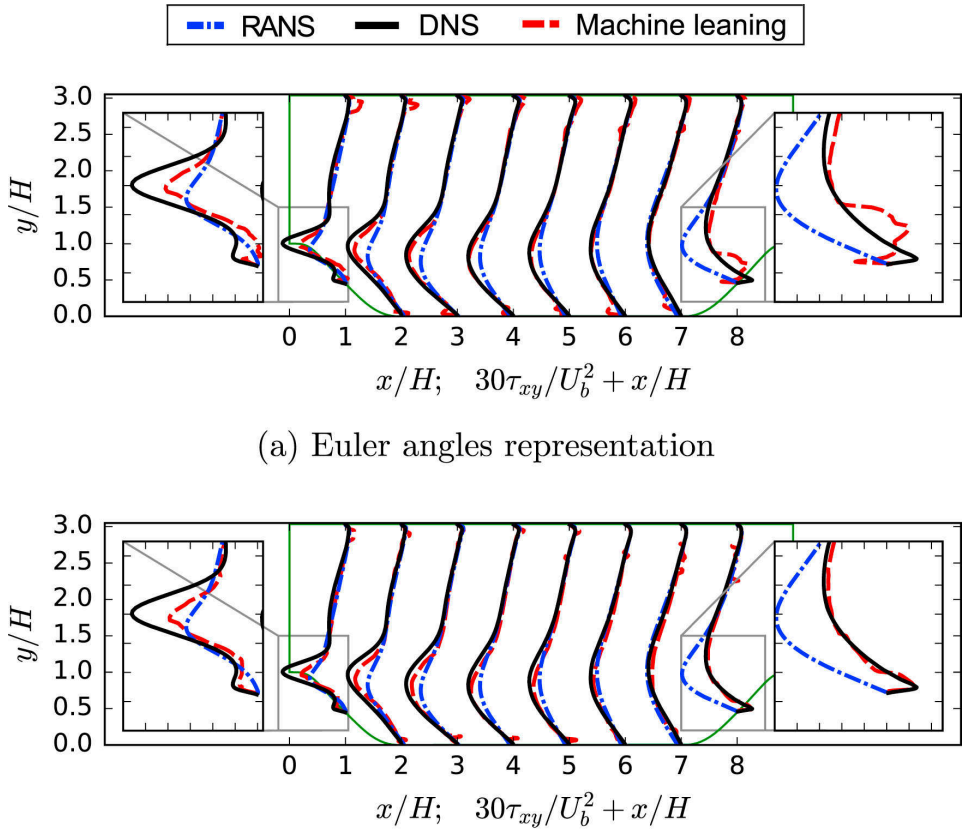
(a) $\Delta\phi_1$ (b) $\Delta\phi_2$ (c) $\Delta\phi_3$

Figure 2.21: The prediction of discrepancy-based Euler angles $\Delta\phi_\alpha$ for the flow in a square duct at $Re = 3500$. Training of at $Re = 2900$ with a coordinate system rotation of 60°. Figure taken from [29].



(a) R_{xy} (b) R_{xz}

Figure 2.22: The prediction of Reynolds stress components based on discrepancy-based Euler angles for the flow in a square duct at $Re = 3500$. The training flow is the flow in a square duct at $Re - 2900$ with the reference frame rotated by 60° anti-clockwise. Figure taken from [29].



(a) Euler angles representation

(b) Unit quaternion representation

Figure 2.23: Results on the periodic hills flow case, comparing different transformations. Figure taken from [29].

3

Methodology for symbolic regression of turbulence model corrections

In this chapter the methodology that is followed is presented. Before discussing the exact methodology where this research focuses on, some theoretical background information is given for the reader to better understand the procedures that will be followed in the remainder of this work. This is done in Section 3.1 where the basics of the RANS procedure and turbulence modelling are covered.

After that the machine learning approach to this problem of turbulence modelling is introduced and the methodology that is followed in the remainder of this report is introduced in section 3.2. With the theory for the basis functions of this method presented in Section 3.2.1, the methodology for attaining the corrective fields to regress models for, the 'frozen-approach' is explained in Section 3.2.2. Finally the machine learning theory is presented in Section 3.2.3 where the goal is to find sparse models with small coefficients to augment to the $k-\omega-SST$ model.

Lastly the implementation of this methodology is discussed. The implementation of the methodology in the industry relevant OpenFOAM v1912 is discussed in Section 3.3 together with the process for the model regression in Section 3.3.5 which is performed in python. In Section 3.3.4 the implementation of the OpenFOAM code is verified against the results from Schmelzer and Dwight [18].

The procedures that have been followed to test all the steps of the methodology presented in this chapter is given in Section 3.4. This section presents what needs to be done in what order to get the results presented in Chapter 4.

3.1. RANS and the need for turbulence modelling

In this section a short overview and some background is given about the theory that will be used in this research. Firstly the Reynolds averaged Navier-Stokes (RANS) equations are presented and the closure problem that is introduced during the derivation of these equations is discussed with the introduction of turbulence modelling, this is done in sections 3.1.1 and 3.1.2.

Secondly some of the higher fidelity methods that have been used to gather high fidelity data used in this research are discussed. The methods that have been used are the large eddy simulation (LES) and the direct eddy simulation (DES). A short description of both methods is given as well as a discussion on the subgrid scale models that have been used together with the inflow conditions for the turbulent inflow that is required in these types of simulations.

3.1.1. RANS

To save computational time Osborne Reynolds time-averaged the Navier stokes equation. The idea behind this is that instead of solving for the flow at a specific time-step one solves for the average flow directly by solving the time averaged equations. This approach is much cheaper when the only interest is the mean flow,

as it does not require calculations for the flow at multiple time steps that then need to be averaged.

The derivation of these equations starts at the incompressible conservation of mass and momentum equations, (3.1) and (3.2).

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.1}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = f_i - \frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{3.2}$$

The derivation starts when the velocity and pressure components are split into their respective mean and fluctuating components. $u_i = \bar{u}_i + u_i'$ and $p_i = \bar{p}_i + p_i'$.

Substitution into the incompressible NS-equations leads to:

$$\frac{\partial(\bar{u}_i + u_i')}{\partial x_i} = 0 \tag{3.3}$$

$$\frac{\partial(\bar{u}_i + u_i')}{\partial t} + (\bar{u}_j + u_j')\frac{\partial(\bar{u}_i + u_i')}{\partial x_j} = (\bar{f}_i + f_i') - \frac{1}{\rho}\frac{\partial(\bar{p}_i + p_i')}{\partial x_i} + \nu \frac{\partial^2(\bar{u}_i + u_i')}{\partial x_j \partial x_j} \tag{3.4}$$

The next step is to take the average of these equations, which will result in the final set of time averaged equations that is solved in the Reynolds averaged approach.

$$\overline{\frac{\partial(\bar{u}_i + u_i')}{\partial x_i}} = 0 \tag{3.5}$$

$$\overline{\frac{\partial(\bar{u}_i + u_i')}{\partial t}} + \overline{(\bar{u}_j + u_j')\frac{\partial(\bar{u}_i + u_i')}{\partial x_j}} = \overline{(\bar{f}_i + f_i')} - \frac{1}{\rho}\overline{\frac{\partial(\bar{p}_i + p_i')}{\partial x_i}} + \nu \overline{\frac{\partial^2(\bar{u}_i + u_i')}{\partial x_j \partial x_j}} \tag{3.6}$$

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \tag{3.7}$$

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} + \overline{u_j' \frac{\partial u_i'}{\partial x_j}} = \bar{f}_i - \frac{1}{\rho}\frac{\partial \bar{p}_i}{\partial x_i} + \nu \frac{\partial^2(\bar{u}_i)}{\partial x_j \partial x_j} \tag{3.8}$$

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = \bar{f}_i - \frac{1}{\rho}\frac{\partial \bar{p}_i}{\partial x_i} + \nu \frac{\partial^2(\bar{u}_i)}{\partial x_j \partial x_j} - \frac{\partial \overline{u_i' u_j'}}{\partial x_j} \tag{3.9}$$

$$\rho \frac{\partial \bar{u}_i}{\partial t} + \rho \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = \rho \bar{f}_i + \frac{\partial}{\partial x_j}\Big[- \bar{p}\delta_{ij} + 2\mu \bar{S}_{ij} - \rho \overline{u_i' u_j'}\Big] \tag{3.10}$$

With $\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$

When this equation is integrated in time the following result is achieved:

$$\rho \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = \rho \bar{f}_i + \frac{\partial}{\partial x_j}\Big[- \bar{p}\delta_{ij} + 2\mu \bar{S}_{ij} - \rho \overline{u_i' u_j'}\Big] \tag{3.11}$$

All the terms in this equation are knowns and can be calculated, except for $\overline{u_i' u_j'}$. This term is called the Reynolds stress tensor and requires modelling to be resolved. That is the closure problem that is introduced by the averaging process.

3.1.2. Turbulence modelling and the k-ω-SST turbulence model

To solve this closure problem of the Reynolds stress a model is required. Over the years a multitude of turbulence models have been created that try to model the Reynolds stress tensor based on the known mean flow features. Of these models a large number are based on the the so called Boussinesq hypothesis. Joseph Boussinesq introduced the concept of eddy viscosity, which is the first approximation to the Reynolds stress in terms of mean flow features and has the following form:

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{3.12}$$

Where in this equation ν_t is the eddy viscosity that requires a model to be calculated. Many of the the currently used turbulence models are based on this Boussinesq hypothesis, such as the $k-\epsilon$ [8] and $k-\omega$ turbulence models. These models approximate ν_t by 2 variables, $\nu_t = \frac{k}{\epsilon}$ and $\nu_t = \frac{k}{\omega}$ [14], where k is the turbulent kinetic energy and ϵ and ω are the turbulence dissipation. For these variables additional transport equations are required. For the $k-\epsilon$ model the turbulence transport equations can be found in (3.13) and (3.14).

Because for these types of models two additional equations are necessary they are called two equation models. There also exist models with a single equation and models with multiple equations, but for this research the focus is on a two equation model. In these models multiple coefficients can be found that can be altered to suit specific types of flows. The standard model coefficients are usually selected such that the resulting flows match well with canonical flows.

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_i)}{\partial x_i} = \frac{\partial}{\partial x_j} \left[\frac{\mu}{\sigma_k} \frac{\partial k}{\partial x_j} \right] + 2\mu_t E_{ij} E_{ij} - \rho\epsilon \tag{3.13}$$

$$\frac{\partial(\rho\epsilon)}{\partial t} + \frac{\partial(\rho\epsilon u_i)}{\partial x_i} = \frac{\partial}{\partial x_j} \left[\frac{\mu_t}{\sigma_\epsilon} \frac{\partial \epsilon}{\partial x_j} \right] + C_{1\epsilon} \frac{\epsilon}{k} 2\mu_t E_{ij} E_{ij} - C_{2\epsilon} \rho \frac{\epsilon^2}{k} \tag{3.14}$$

The turbulence model that will be modified for this research is the $k-\omega$-SST equation [14]. This is a turbulence model based on the $k-\omega$ model from Wilcox [28]. The SST part of the model stands for shear stress transport model and uses the original $k-\omega$ model for the inner half of the boundary layer and switches to the $k-\epsilon$ model in the outer part. The model has the ability to cope with transport of the principal turbulent shear stress in adverse pressure gradient boundary-layers. This is mainly important when trying to simulate massively separated flows and flows involving multiple length-scales. The flows that will be dealt with in this research all have these features and therefore the choice for the $k-\omega$-SST turbulence model is an appropriate choice.

The model equations of the $k-\omega$-SST turbulence model are shown in (3.15) and (3.16).

$$\frac{\partial \rho k}{\partial t} = \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta^* \rho \omega k + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_k \mu_t) \frac{\partial k}{\partial x_j} \right] \tag{3.15}$$

$$\frac{\partial \rho \omega}{\partial t} = \frac{\gamma}{\nu_t} \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + 2\rho(1 - F_1)\sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \tag{3.16}$$

Where $\phi = F_1 \phi_1 + (1 - F_1)\phi_2$ is defined as the switching function, switching between model coefficients of the original $k-\omega$ model ϕ_1 and the transformed $k-\epsilon$ model ϕ_2. The original $k-\omega$ model will be used in the near wall region exclusively. The transformed $k-\epsilon$ model is mainly used in free shear-layers.

For this the blending function F_1 needs to be defined such that it is equal to one for a large part of the boundary layer. This is in order to preserve the desirable features of the $k-\omega$ model in the boundary layer but should go to zero at the boundary layer edge, to preserve free-stream independence.

$$F_1 = \tanh(arg_1^4) \tag{3.17}$$

$$arg_1 = \min\left(\max\left(\frac{\sqrt{k}}{0.09\omega y}; \frac{500\nu}{y^2 \omega} \right); \frac{4\rho\sigma_{\omega 2} k}{CD_{k\omega} y^2} \right) \tag{3.18}$$

With $CD_{k\omega}$ defined as

$$CD_{k\omega} = \max(2\rho_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-20}) \tag{3.19}$$

The eddy viscosity is defined as:

$$\nu_t = \frac{a_1 k}{\max(a_1\omega; \Omega F_2)} \tag{3.20}$$

With $\Omega = \frac{\partial u}{\partial y}$ and where F_2 is also a blending function and is defined as:

$$F_2 = \tanh(arg_2^2) \tag{3.21}$$

$$arg_2 = \max\left(2\frac{\sqrt{k}}{0.09\omega y}; \frac{500\nu}{y^2\omega}\right) \tag{3.22}$$

The model coefficients are defined as in (3.23). These basis coefficients are tuned such that the boundary layer of a flat plate shows the correct behaviour.

$$\begin{aligned} \sigma_{k1} = 0.85, \quad \sigma_k = 0.5, \beta_1 = 0.0750, \quad a_1 = 0.31, \\ \beta^* = 0.09, \quad \kappa = 0.41, \quad \gamma_1 = \beta_1/\beta^* - \sigma_{\omega 1}\kappa^2/\sqrt{\beta^*} \end{aligned} \tag{3.23}$$

This $k-\omega-SST$ model relies heavily on the Boussinesq hypothesis, the accuracy of which is lacking as was among other proven by Schmitt [19], who has investigated the validity of the Boussinesq hypothesis by calculating the validity of the expression in (3.12). This was done evaluating the following expression:

$$\rho_{RS} = \frac{|\mathbf{R_{ij}} : \mathbf{S_{ij}}|}{||\mathbf{R_{ij}}||\,||\mathbf{S_{ij}}||} \tag{3.24}$$

This number is analogous to the cosine of the angle between vectors. This means that ρ_{RS} will be a number between 1 and 0. With 1 being that the Boussinesq hypothesis is valid and as it goes to 0 it becomes less valid. Figure 3.1 shows this relationship for several DNS cases and it shows that near the wall ($y+$ small), the Boussinesq approximation holds well. This is because close to the wall the viscous stress dominates. Further away from the wall the turbulent stress becomes more dominant and therefore the Boussinesq hypothesis holds less.

This inacuracy in the Boussinesq approximation of the Reynolds stress is a good target for a correction. The machine learning part of this research focuses on finding a new better approximation. The theoretical basis of this new approximation is explained in section 3.2.1 with the machine learning approach to finding this correction explained in section 3.2.

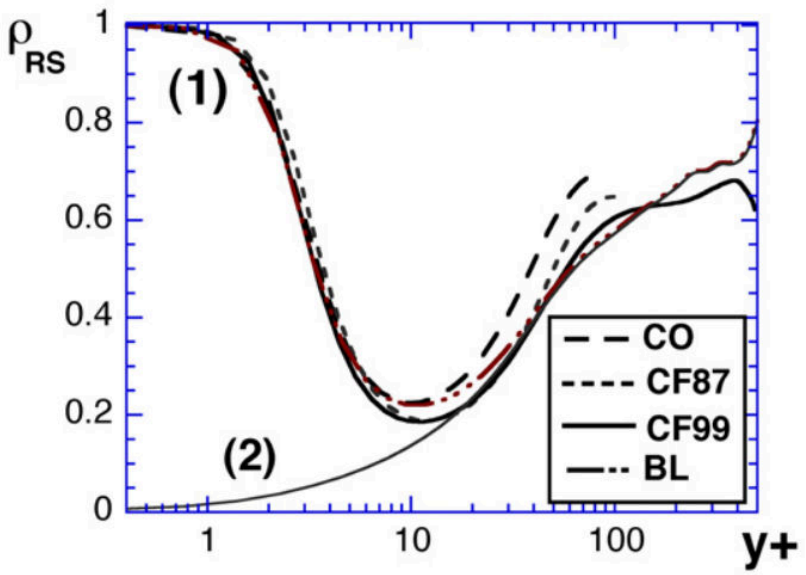


Figure 3.1: Plot of $\rho_{RS}(y^+)$ for DNS databases: (1) total stress (turbulent + viscous stress) and (2) turbulent stress only. [19]

3.1.3. High fidelity methods: LES

It is possible to fully resolve the flow exactly using the so called direct numerical simulation. In this type of simulation no closure models are used and all scales are resolved. The smallest scales that need to be resolved are so small that a lot of computational power is needed. For flows with a higher Reynolds number the smallest scales become so small that calculating all these scales is not a viable option as time scales and length scales become too small, such that a massive amount of computational power is needed. In this case for gathering high fidelity data doing a large eddy simulation (LES) might be the best option.

A large eddy simulation is a type of simulation similar to the Direct numerical simulation where the flow is resolved in time but in contrary to the DNS, LES doe not resolve all scales. It resolves the large scales and models the smaller scales. The size of the scales that are resolved depends on the grid size, the smaller the grid size, the smaller the scales that can be resolved. The scales that are too small for the grid size need to be modelled, this is done using a so called subgrid-scale model (SGS).

SGS

The sub-grid scale model that was used for the LES simulations for this research is the WALE model. The WALE model is the Wall adapting local eddy-viscosity model, this model was introduced by Nicoud and Ducros [5]. Most subgrid-scale models model the subgrid-scale tensor in (3.25) using the eddy viscosity assumption.

$$T_{ij} = \overline{u}_i \overline{u}_j - \overline{u_i u_j} \tag{3.25}$$

$$\overline{S_{ij}} = -\frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) \tag{3.26}$$

One of the most common subrid-scale models is the Smagorinsky model [22]. This model has formed the basis for a large number of subgrid-scale models. In this model the eddy viscosity is modelled in the following way:

$$\nu_t = (C_s \Delta)^2 |\overline{S}|, \quad |\overline{S}| = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \tag{3.27}$$

Here C_s is the Smagorinsky coefficient which can be determined from looking at the energy spectrum and assuming that the dissipation that occurs in the subgrid scales is identical to the disspation at the cut-off wave number k_c.

A downside to the Smagorinsky model is that it does not take into account the presence of walls. The WALE model tries to model the eddy viscosity differently by replacing S_{ij} with the traceless symmetric part of the velocity gradient tensor $S_{ij} = \frac{1}{2}(G_{ij}^2 + G_{ji}^2) - \frac{1}{3}\delta_{ij}G_{kk}^2$ with $G_{ij} = \frac{\partial \overline{u}_j}{x_i}$. Then to get the correct asymptotic wall scaling it uses the following expression for the eddy viscosity.

$$\nu_t = (C_W \Delta)^2 \frac{(S_{ij}^d S_{ij}^d)^{\frac{3}{2}}}{(S_{ij}^d S_{ij}^d)^{\frac{5}{2}} + (S_{ij}^d S_{ij}^d)^{\frac{5}{4}}} \tag{3.28}$$

Cost and stability
One of the things that should be taken into account when doing an LES is a convergence requirement. The maximum timestep is limited by the so called Courant–Friedrichs–Lewy condition (3.29). This is a necessary condition for convergence when solving partial differential equations.

$$CFL = \frac{u_i \Delta t}{\Delta x_i} \leq 1 \tag{3.29}$$

For convergence this CFL number is required to be smaller than 1. This means that with the scaling of the mesh size Δx_i the time-step needs to scale with it to keep the CFL number under 1. This means that when the mesh becomes finer the time-step needs to be smaller. Also when the velocity is increased the time step needs to be adjusted accordingly. This condition needs to hold locally, so the number cannot exceed 1 anywhere in the domain. This means usually that the critical condition is near walls, where there is a combination of a high velocity and small mesh size.

This CFL condition also gives an insight into why the cost of an LES scales with the Reynolds number (3.30). As both the velocity and the mesh size are dependent on the Reynolds number of the flow. When the Reynolds number increases the inertial Range lengthens moving the dissipation range further away into the smaller scales. To have accurate large-scale statistics it is required to have the initial part of the spectrum resolved. In the outer layer far away from walls the resolution needed does not scale strongly with the Reynolds number. However, for the flow close to the wall, in the so called inner layer, the size decreases much quicker with Reynolds number.

$$Re = \frac{\rho U L}{\mu} \tag{3.30}$$

This increased computational cost in relation to an increase in Reynolds number is a result of the decrease in the size of eddies. Forcing the cut off wave number to be lower and thus having a finer mesh requirement. Having a finer mesh results in the necessity of having a smaller time step as well, as is dictated by the CFL convergence requirement (3.29).

Inflow condition
Finally for an LES it is crucial to have the right inflow conditions. The inflow must have a form of turbulence that matches the expected inflow for a given Reynolds number. There are three general ways of generating a turbulent inflow condition.

The first is by creating synthesised turbulence, what this does is that a mean flow is specified, on which artificial fluctuations are created. This is usually done using random numbers with a scaling to create the correct turbulence statistics for the inflow. The implementation that was used in the simulations done in this research used the divergence free synthetic eddy viscosity inflow model implemented OpenFOAM. This OpenFOAM implementation is based on the work from Poletto, Craft and Revell [15]. This type of inflow condition was chosen because of it's good controllability and stability features. It is also computationally cheaper than the other two options.

The second is by running a precursor simulation, this method takes the data from a separately running or stored simulation and maps the velocity data on the inflow of the simulation. The upside to this method is that there is no feedback to the precursor and therefore preventing acoustic interactions. The disadvantage is that a large amount of data needs to be stored to avoid excessive correlations or the precursor simulation must be run concurrently, making the simulation more computationally expensive.

The third is by recycling and rescaling data from downstream of the flow. The data is rescaled to match the desired boundary layer thickness and turbulence intensity. One of the downsides of this method is that there can occur correlations between the extraction plane and the inflow plane if these two are too close too each other. This sort of boundary condition is similar to a periodic boundary condition however, the coupling only works one way.

3.1.4. High fidelity methods: DES

In the case that an LES computation is also too time consuming or too computationally expensive, the choice can be made to perform a hybrid between RANS and LES, also known as the direct eddy simulation (DES).

In this type simulation the scales that are too small to be resolved using an LES method are modelled using a RANS model. This approach is called a hybrid approach and since the cost of an LES scales with the Reynolds number mainly in the regions near walls taking a hybrid approach can result in a significant saving of computational time.

For the DES that has been done in this work the Spalart-Allmaras eddy viscosity model has been used. This eddy viscosity model estimates the eddy viscosity in both the RANS regions as well as for the LES region subgrid-scale model.

Similar to an LES a DES also requires a turbulent inflow condition. For the DES that was done in this research again the synthetic turbulence inflow condition was used.

3.2. Method for symbolic regression: SpaRTA

The methodology that forms the basis of this research is that introduced by Schemlzer and Dwight [18], they came up with a novel approach that infers algebraic stress models for the closure of the RANS equations. It does this directly from high-fidelity data sources that can be either LES or DNS. This is further extended to the use of DES as well in this work.

The goal of this method is to improve already existing turbulence models and specifically the $k-\omega-SST$ model and thereby improve RANS predictions in general. The approach presented is called Sparse Regression of Turbulent Stress Anisotropy. A schematic overview of the method is given in Figure 3.2, in the following section the steps in this schematic are further explained.

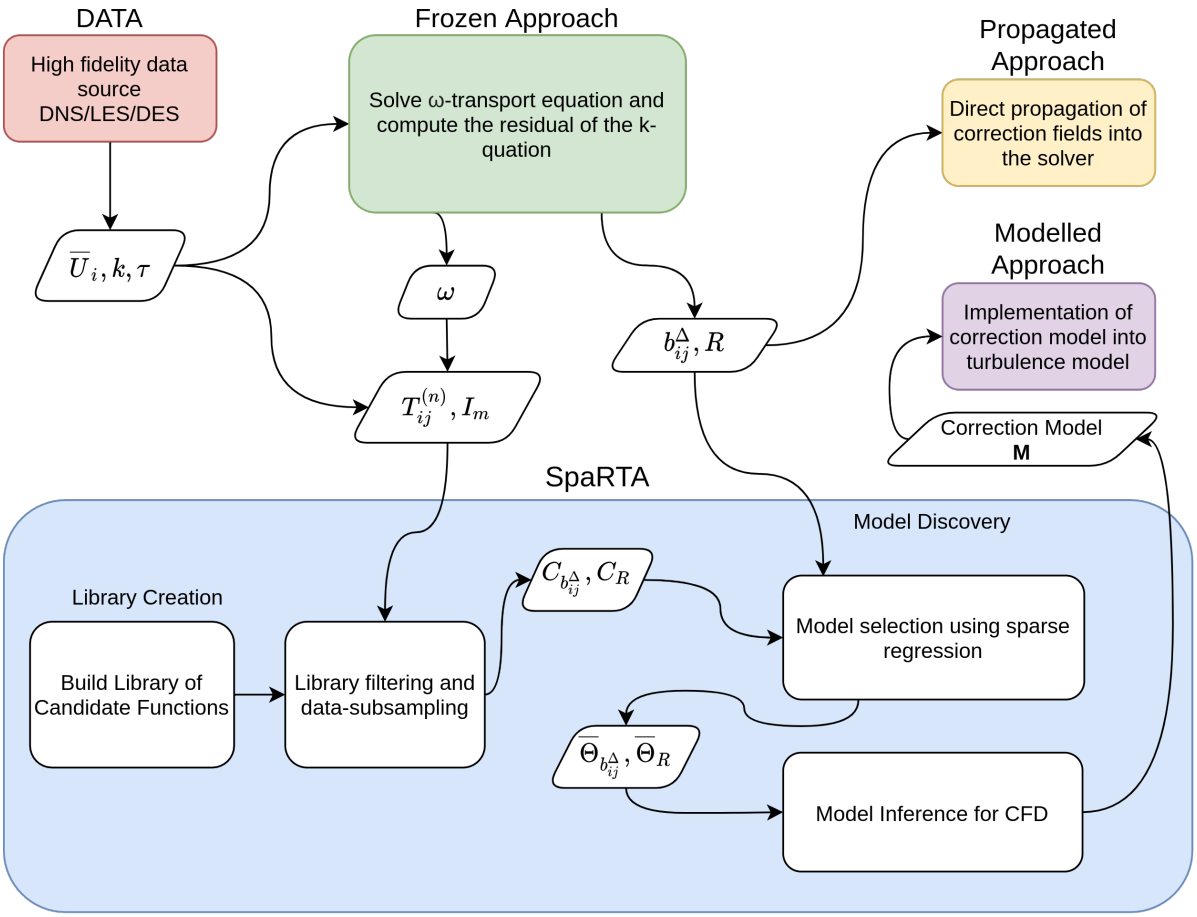


Figure 3.2: Schematic overview of the SpaRTA methodology.

The machine learning framework is focused on finding models to improve the RANS method. Machine learning is a way of finding these models from a large set of high fidelity data. There exist several machine learning algorithms that are able to do this. Within all these methods is a similarity, this is that they rely on the availability of training data, that is the combination of input values and target values. In other words: what values are expected from a trained model for a given set of given inputs? In this research the inputs of the training data are the mean flow features that are available from a RANS simulation. The targets are focused on achieving the correct Reynolds stress.

The training data that the SpaRTA methodology uses is not directly available from the set of high fidelity data and requires an additional processing step. The method that is used for this is the 'k-corrective-frozen-RANS' or Frozen approach as is shown in Figure 3.2. A thorough explanation of this procedure can be found in section 3.2.2 and the implementation of this method in code is discussed in section 3.3.1.

The corrective fields that are found from this frozen approach can be applied in two ways. The first is

by directly inserting these corrections into the model when solving. This method is called the propagated approach and is possible only on the same geometry as the training case. The implementation of this approach is discussed in section 3.3.2. This way of correcting does not involve any machine learning. To make the corrections generally applicable a model needs to be found that predicts the corrective fields from the mean flow features.

This leads to the machine learning approach that is used in this research, which is called Sparse Regression of Turbulent Stress Anisotropy or SpaRTA in short. The application of the correction fields using these machine learned models is called the 'modelled approach' (Figure 3.2). The theory behind SpaRTA is explained in section 3.2.3 and the implementation of this method are explained in section 3.3.4. The modelled approach implementation is discussed in section 3.3.3.

Before all this it is necessary to understand the theory behind the SpaRTA methodology, what the inputs for the method are and how the corrections are applied to existing turbulence models. The basis for the corrections and inputs comes from the theory from Pope [16] who derived a new effective-viscosity hypothesis. This theoretical basis is discussed in section 3.2.1. This theory forms the basis to model from mean flow features a correct prediction of the Reynolds stress tensor.

3.2.1. Pope's effective-viscosity hypothesis

The SpaRTA method regresses correction functions to improve the modelling of the Reynolds stress. The regression relies on a library of basis functions that was introduced by Pope in 1975 [16], a proposition was made for a new effective-viscosity hypothesis. The first effective-viscosity hypothesis was proposed by Boussinesq. The idea behind an effective-viscosity hypothesis is that it relates the Reynolds stresses only to the rates of strain of a fluid and to scalar quantities.

The hypothesis from Boussinesq, which is used in the majority of turbulence models, relates the Reynolds stress to the velocity gradient U_{ij} and an effective viscosity.

$$-\overline{u_i' u_j'} = \mu_{eff} U_{ij} \tag{3.31}$$

This expression can be generalised for higher dimensional problems to give the well known isotropic-viscosity assumption as shown in (3.32) and also shown in (3.12).

$$\overline{u_i' u_j'} = \frac{2}{3} k \delta_{ij} - \mu_{eff}\left(U_{ij} + U_{ji}\right) \tag{3.32}$$

This approximation has been successfully applied but fails in accurately representing the Reynolds-stress tensor. In Pope's work he derives a constitutive relation for the Reynolds stresses. With the following assumptions: The Reynolds stresses are uniquely related to the rates of strain and local scalar quantities. This is an assumption fundamental to an effective-viscosity hypothesis. An other assumption that is implied by an effective-viscosity hypothesis is that the stresses are determined locally.

The two scaling parameters k and ω are used to normalise the Reynolds stress and rates of strain. Where ω is the turbulent dissipation rate and k the turbulent kinetic energy.

$$a_{ij} = \overline{u_i, u_j}/k - \frac{2}{3}\delta_{ij} \tag{3.33}$$

$$S_{ij} = \frac{1}{2}(k/\omega)(U_{ij} + U_{ji}) \tag{3.34}$$

$$R_{ij} = \frac{1}{2}(k/\omega)(U_{ij} - U_{ji}) \tag{3.35}$$

a_{ij} and S_{ij} are dimensionless tensors with zero trace. This is also due to the incompressibile flow assumption which results in $S_{ii} = 0$. From (3.33) it can be seen that determining the Reynolds stress $\overline{u_i u_j}$ is equivalent to determining a_{ij}. On the dimensional grounds S_{ij} and R_{ij} contain all the information from k, ω and $U_{i,j}$.

$$a_{ij} = a_{ij}(S_{ij}, R_{ij}) \tag{3.36}$$

The most general expression for (3.36) can be written as an infinite tensor polynomial of the combinations of S_{ij}, R_{ij} and the coefficients G (3.37). Luckily the Cayley-Hamilton theorem states that the number of independent invariants and the number of linearly independent second-order tensors that can be formed from the strain rate and rotation rate tensors is finite.

$$a_{ij} = \prod_{i=1}^{\infty} \sum_{\alpha_i=0}^{\infty} \prod_{j=1}^{\infty} \sum_{\beta_j=0}^{\infty} G^{\alpha_1,\alpha_2\cdots}_{\beta_1,\beta_2\ldots} s_{ij}^{\alpha_1} R_{ij}^{\beta_1} s_{ij}^{\alpha_2} R_{ij}^{\beta_2} \tag{3.37}$$

[16]

In the general three-dimensional case there are ten tensors and five invariants:

$$\begin{aligned}
T^1 &= S_{ij}, & T^2 &= S_{ij}R_{ij} - R_{ij}S_{ij},\\
T^3 &= S_{ij}^2 - \tfrac{1}{3}Tr(S_{ij}^2)I, & T^4 &= R_{ij}^2 - \tfrac{1}{3}Tr(R_{ij}^2)I,\\
T^5 &= R_{ij}S_{ij}^2 - S_{ij}^2R_{ij}, & T^6 &= R_{ij}^2S_{ij} + S_{ij}R_{ij}^2 - \tfrac{2}{3}Tr(S_{ij}R_{ij}^2)I,\\
T^7 &= R_{ij}S_{ij}R_{ij}^2 - R_{ij}^2S_{ij}R_{ij}, & T^8 &= S_{ij}R_{ij}S_{ij}^2 - S_{ij}^2R_{ij}S_{ij},\\
T^9 &= R_{ij}^2S_{ij}^2 + S_{ij}^2R_{ij}^2 - \tfrac{2}{3}Tr(S_{ij}^2R_{ij}^2)I, & T^{10} &= R_{ij}S_{ij}^2R_{ij}2 - R_{ij}^2S_{ij}R_{ij},
\end{aligned}$$

Table 3.1: The ten tensors for the general three-dimensional case

$$I_1 = Tr(S_{ij}^2) \quad I_2 = Tr(R_{ij}^2) \quad I_3 = Tr(S_{ij}^3) \quad I_4 = Tr(R_{ij}^2S_{ij}) \quad I_5 = Tr(R_{ij}^2S_{ij}^2)$$

Table 3.2: The five invariants for the general three-dimensional case

With this limitation to 10 Tensors and 5 invariants (3.37) can be rewritten to a finite sum:

$$\sum_{n=1}^{10} g^{(n)}(I_1,...,I_5)\mathbf{T}^{(n)} \tag{3.38}$$

The reason to use this tensor basis is the fact that the Reynolds stress tensor is approximated by a tensor and that any continuous function can be determined using this basis. Due to the dimensionless coefficients and the model form, Galilean invariance is ensured. This is a critical property for CFD applications as methods should not give different results for different geometry orientations.

3.2.2. Calculating the correction fields: Frozen approach

The SpaRTA methodology uses the tensor basis to regress a correction model for the turbulent transport model equation. The first correction term b_{ij}^{Δ} acts directly on the Reynolds anisotropy as defined in (3.33). The correction term is added to the Reynolds anisotropy to correct for the assumption that is made in the Boussinesq approximation. Namely that the true anisotropy tensor a_{ij} can be approximated as follows: $b_{ij}^0 = -\frac{\nu_t}{k}S_{ij}$, where S_{ij} is the strain rate tensor and b_{ij}^0 is the Reynolds anisotropy tensor as predicted by the Boussinesq hypothesis.

$$\overline{u_i, u_j} = 2k(b_{ij}^0 + b_{ij}^{\Delta} - \frac{1}{3}\delta_{ij}) \tag{3.39}$$

For the application of this corrective approach the $k-\omega-SST$ model has been selected, this model is described in Section 3.1.2. The correction factors for this model are a correction to the Reynolds stress anisotropy tensor, leading to a correction to the turbulence production term together with a correction to the k-transport equation which also influences the production of turbulence. These corrective fields/targets are calculated in the so called frozen approach also named k-corrective-frozen-RANS method. This method uses the mean velocity and turbulent kinetic energy as inputs to the k and ω equations which are then solved for $\omega, k, b_{ij}^{\Delta}, R$ (3.40) (3.41) (3.42). The method is called frozen because all the inputs from the high fidelity data are frozen. The idea behind this is that with the correct anisotropy and the additional correction to the turbulent kinetic energy equation the RANS method should converge to the high fidelity solution. Therefore the mean velocity of the high fidelity data forms the inputs for the target calculation.

$$\frac{\partial k}{\partial t} + U_j\frac{\partial k}{\partial x_j} = P_k + R - \beta^*\omega k + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_k\nu_t)\frac{\partial k}{\partial x_j}\right] \tag{3.40}$$

$$\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \frac{\gamma}{\nu_t}(P_k + R) - \beta\omega^2 + \frac{\partial}{x_j}\left[(\nu + \sigma_\omega \nu_t)\frac{\partial \omega}{\partial x_j}\right] + CD_{k\omega} \tag{3.41}$$

$$P_k = 2k(b^0_{ij} + b^\Delta_{ij}) \tag{3.42}$$

All the other model coefficients and equations are kept the same as the standard $k-\omega-SST$ model. The correction therefore is on the turbulent kinetic energy and the anisotropy, which uses the standard Boussinesq approximation $b^0_{ij} = \frac{\nu_t}{k} S_{ij}$ and the correction term b^Δ_{ij}.

When looking back at the graphical overview of the methodology in Figure 3.2 the two top blocks represent this procedure. The DATA block forms the input and with this input the k-corrective-frozen-RANS procedure is performed. From this procedure the outputs are the targets: b^Δ_{ij} and R, and the specific dissipation rate ω which is not directly available from the high fidelity data and needs to be calculated. This dissipation rate ω is needed to calculate the tensors $T^1 - T^{10}$ and the invariants $I_1 - I_5$, which are normalised using k and ω as was shown by Pope [16].

With the correction fields and input fields coming from the frozen approach the next step is the inference of a correction model that can calculate these correction fields from mean flow data which is available from a RANS simulation.

3.2.3. Correction model discovery

For the model regression an ansatz is made, it is assumed that the anisotropy correction can be modelled using the first four Tensors T^{1-4} and the first two invariants I_{1-2}. It is also assumed that the correction to the turbulent kinetic energy equation can be written in the similar form as that of the anisotropy tensor, as shown in (3.43). This has the advantage that the same framework can be used for the model regression for the R term.

$$R = 2kb^R_{ij}\frac{\partial U_i}{\partial x_j} \tag{3.43}$$

The model regression consists of two steps namely the model discovery stage and the model inference stage. The model discovery stage is used to construct a large library of nonlinear candidate functions to regress data.

The first step in model discovery consists of the construction of a library with all possible combinations of invariants and tensors. This is done by multiplying all invariants with each other, leading to a maximum degree of 6.

$$\mathbf{B} = \left[c, I_1, I_2, I_1^2, I_2^2, I_1^2 I_2^3, I_1^4 I_2^2, I_1 I_2^2, I_1 I_2^3, I_1 I_2^4, I_1^3 I_2, I_1^2 I_2^4, I_1^2 I_2, I_1 I_2, I_1^3 I_2^2, I_1^2 I_2^2\right] \tag{3.44}$$

To create the library each combination of invariants is combined with each of the tensors T^{1-4} resulting in a library of candidate functions (3.45). For the library of the k-correction R all the variables the double dot product of each $\mathbf{C}_{b^\Delta_{ij}}$ is taken with the velocity gradient tensor leading to the library (3.46).

$$\mathbf{C}_{b^\Delta_{ij}} = \left[cT^{(1)}_{ij}, cT^{(2)}_{ij}, ..., I_1^2 I_2^2 T^{(4)}_{ij}\right]^T \tag{3.45}$$

$$\mathbf{C}_R = \left[cT^{(1)}_{ij}\partial_j U_i, ..., I_1^2 I_2^2 T^{(4)}_{ij}\partial_j U_i\right]^T \tag{3.46}$$

These two libraries of functions are then evaluated at all points in the domain to give a full set of training input features. The target values from the k-corrective frozen are stacked in vectors $\mathbf{b}^\Delta, \mathbf{R}$ and form the targets for the model discovery. With these libraries it is then the goal to form a linear model to regress the target data $\Delta = \mathbf{b}^\Delta$ or $\mathbf{R}$, by finding the coefficient vector Θ.

$$\Delta = \mathbf{C}_\Delta \Theta \tag{3.47}$$

The system that is being solved is a large overdetermined system with many datapoints that need to be fit to the model. To keep the model that is found lightweight to be later used in the turbulence model it is

of importance to keep the number of model variables low in other words to have a sparse model. The other constraint that is applied is to keep the coefficients small. This is done for both stability reasons as well as to prevent overfitting the data. Therefore, the elastic net regression is used to find models that are both sparse and have small coefficients.

$$\Theta = \arg\min_{\hat{\Theta}} \left|\left|\mathbf{C}_\Delta \hat{\Theta} - \Delta\right|\right|_2^2 + \lambda\rho\left|\left|\hat{\Theta}\right|\right|_1 + 0.5\lambda(1-\rho)\left|\left|\hat{\Theta}\right|\right|_2^2 \tag{3.48}$$

The elastic net regression blends the l_1- and l_2-norm regularisation given the mixing parameter $\rho \in [0,1]$ and the regularisation weight λ, to promote sparsity of Θ. The larger the regularisation weight λ the higher the number of nonzero coefficients. Because it is unknown what regularisation parameters give the best results a range of combinations of regularisation weights and mixing parameter are selected. A total of 100 entries of λ uniformly scaled using a log-scale. The elastic net is used to get the model coefficients Θ for this range of regularization weights and mixing parameters. There is a good chance that certain combinations of these regression parameters give models with the same non-zero terms, these models are combined. The model discovery step is done with normalised coefficients to aid the elastic net algorithm.

With the library of model forms complete, ridge regression is used to fit coefficients to the model terms for each of the models in the library. Ridge regression is used to control their size (3.49). For that a range of Tikhonov-regularisation parameters have been selected to give a different range of model coefficient sizes.

$$\Theta_\Delta^{\mathbf{s},\mathbf{d}} = \arg\min_{\hat{\Theta}_\Delta^{s,d}} \left|\left|\mathbf{C}_\Delta^{\mathbf{s}} \hat{\Theta}_\Delta^{\mathbf{s},\mathbf{d}} - \Delta\right|\right|_2^2 + \lambda_r \left|\left|\hat{\Theta}_\Delta^{\mathbf{s},\mathbf{d}}\right|\right|_2^2 \tag{3.49}$$

Once the model regression step has been completed a model can be selected to be implemented in a RANS solver. During this step the model is validated and the convergence is studied. It is not guarenteed that all corrections models will result in a converging result when implemented in a solver. It has been found that $0.1 > \lambda_r > 0.01$ gives model coefficients that produce converged solutions when implemented in a RANS model. These observations are based purely on empirical observations and in case a model did not converge the coefficients were decreased by a factor $\xi = 0.1$ for the b_{ij}^Δ model.

3.3. Code implementation

A large part of the methodology explained above needs to be implemented into a solver. For this the openFOAM solver has been selected. For the model discovery steps python has been used in combination with the scikit-learn toolkit.

The OpenFOAM solver has been selected because it is an opensource platform which allows for the modification of source code. There is also good documentation available of the solvers and the turbulence models that are already implemented in the base version of the code. The original work from Schmelzer and Dwight [18] has been performed in version 2.4.0 of OpenFOAM, but for this work it was chosen to implement the methodology in version v1912. This decision was made because version 2.4.0 was released in 2015 and therefore misses 5 years of development making the code less relevant for industry. Between these versions there are a lot of differences in the way the code is implemented. In this section the implementation of the different methods is documented and the verification procedure to show the correct implementation is shown.

In this section the implementation of the 3 methods in the top right of Figure 3.2. The modification to the original turbulence models are explained in Sections 3.3.1-3.3.3 and the code is available in Appendix B. After this in Section 3.3.4 the implementation of this code is verified against the original implementation from Schmelzer and Dwight in version 2.4.0.

3.3.1. OpenFOAM Frozen approach code

The implementation of the frozen approach requires the modification of two things: the simpleFoam solver and the $k-\omega-SST$ turbulence model. The simpleFoam solver is an incompressible RANS solver which solves the continuity and the momentum equation using the SIMPLE algorithm. After which it calls the turbulence model which then solves the turbulence transport equation(s).

Knowing this, the modification of the simpleFoam solver to make it suitable for the frozen approach as shown in section 3.2.2 is rather easy. The only steps that need to be taken are to remove the lines calling for the solving of the continuity and the momentum equations. Together with renaming of the solver to *frozen_simpleFoam* the code can be compiled and used to perform the k-corrective-frozen-RANS.

The main modifications, however are in the turbulence model, where some alterations need to be made. First of all, the additional fields that will be calculated need to be declared: b_{ij}^{Δ} and R, secondly the existing definition for k needs to be modified, as k is frozen it needs to be loaded from the high fidelity data and there is no need to rewrite k to a file again.

Besides this there are some more variables that are newly defined, either because they are needed in the method or they contribute to the visualisation of the method. These variables are:

- a_{ij}, The Reynolds anisotropy calculated from the high fidelity data.
- a_{ij}^{Δ}, the difference between the Reynolds anisotropy from the high fidelity data and the prediction from the Boussinesq approximation.
- b_{ij}^{Δ}, the non-dimensionalized version of a_{ij}^{Δ}.
- P_k, the turbulence production as calculated from the high fidelity method .
- P_k^0, the turbulence production as predicted by the Boussinesq approximation .
- P_k^{Δ}, the difference between P_k and P_k^0 .
- $timescale_{limited}$, The timescale that is later used when calculating the tensors and invariants.

The code definitions of these variables can be found in appendix B.1, these are the initial definitions of the variables. They are updated later when the turbulence transport equations are solved.

The modified turbulence transport equations are given by (3.40) and (3.41) and the implementation can be found in appendix B.2. The modifications to the original code are in the lines calculating the turbulence production. This includes the calculation of the production as is calculated by the Boussinesq approximation.

The altered production is carried on into the ω-equation which is besides this modified production and the addition of the R term unaltered. The k-equation is a bit different, as from the high fidelity data the turbulence kinetic energy k is already known it is only necessary to solve for R (called kDeficit in the code). This is done by moving all the terms except kDeficit to the right hand side of the k-equation (3.40).

Finally after solving the ω equation and calculating R, the anisotropy correction b_{ij}^{Δ} needs to be updated, this is done at the end of the code. Which concludes all the modifications to the $k-\omega-SST$ turbulence mode.

3.3.2. OpenFOAM Propagated approach code

The propagated approach is where the correction field are directly imported into the solver to correct the RANS simulation. For this only a small number of changes need to be made to the turbulence model only.

These changes are the addition of the definition of the correction fields in a similar manner as was done for the frozen approach, only this time the correction fields must be read from a file and not written to a file. The code for the loading of the correction fields can be found in appendix C.1 together with the modified k and ω equations.

The main modifications in the equations are in the turbulence production term P_k called G in the code, this happens before the declaration of the actual equations. Inside the actual equations R (kDeficit) is added directly to correct the turbulence production in both the ω and k-equations. In the omega equation this R term is divided by ν_t to make it the same dimensions as the $GbyNu0$ term which is equal to the turbulence production devided by the eddy viscosity: P_k/ν_t. In the k-equation the R correction can be added directly.

One more thing that needs to be changed are two functions that are not normally declared in the turbulence model but are used when solving the momentum equation. The first of the two functions is called devReff and returns the effective stress tensor. This function is called when solving for the momentum equation and needs to be modified to include the anisotropy correction b_{ij}^{Δ}. The code can be found in appendix C.2, together with the modified code for the function divDevReff, which returns the source term for the momentum equation, this function needs to be updated as well with the anisotropy correction tensor b_{ij}^{Δ}.

3.3.3. OpenFOAM Modelled approach code

The difference between the modelled approach and the propagated approach is mainly that the correction fields are modelled instead of imported. For this it requires the implementation of the basis functions from Pope's viscosity hypothesis (Section 3.2.1) in the OpenFOAM code. One of the big advantages of the OpenFOAM framework is the implemented methods for tensor operations.

The definitions of the Tensors and invariants that need to be implemented are in shown in table 3.1. There are a couple of useful tensor operators that from openfoam that will be used. The mathematical operators are shown below.

First of there is the 'symm' operator, which takes the symmetric part of the matrix. This operator is used a lot as all the 10 tensors from the basis functions used are symmetric. Due to floating point precision and round-off errors not all tensors were automatically recognised as symmetric. To make sure these tensors were stored as symmetric matrices for memory efficiency the symm operator was used.

$$\text{symm}(\mathbf{A}) = \frac{1}{2}(\mathbf{A}+\mathbf{A}^T) \tag{3.50}$$

The 'skew' operator is the opposite of the 'symm' operator and takes the a-symmetric part of the tensor. This operator has not been used in the final code, but does form a useful tool when debugging and post-

processing.

$$\text{skew}(\mathbf{A}) = \frac{1}{2}(\mathbf{A} - \mathbf{A}^T) \tag{3.51}$$

The final operator that has been used is the 'dev' operator. This operator takes the deviatoric part of the tensor A of which the trace($\mathbf{A}$) = 0. The other part is called the hydrostatic part and is of the form $s\mathbf{I}$, where I is the identity matrix and s is a scalar.

$$\text{dev}(\mathbf{A}) = \mathbf{A} - \frac{1}{3}\text{tr}(\mathbf{A})\mathbf{I} \tag{3.52}$$

In the code the velocity gradient tensor is defined as 'tgradU':

$$\text{tgradU} = \begin{bmatrix} \frac{\partial U_x}{\partial x} & \frac{\partial U_x}{\partial y} & \frac{\partial U_x}{\partial z} \\ \frac{\partial U_y}{\partial x} & \frac{\partial U_y}{\partial y} & \frac{\partial U_y}{\partial z} \\ \frac{\partial U_z}{\partial x} & \frac{\partial U_y}{\partial y} & \frac{\partial U_z}{\partial z} \end{bmatrix} \tag{3.53}$$

With that the two basis components that are used to calculated all the remaining tensors $T^1 - T^{10}$ and invariants $I_1 - I_5$ are the strain rate tensor S_{ij} and the rotation rate tensor R_{ij}.

The strain rate tensor is calculated using the following operators according to (3.34). It is normalised using the definition for 'timel', this is a limited timescale and sets a limit on ω in case values get too large.

$$S_{ij} = \text{dev}(\text{symm}(\text{tgradU}/\text{timel})) \tag{3.54}$$

$$\text{timel} = \max((\sqrt{2 * (\text{symm}(\text{tgradU}) : \text{symm}(\text{tgradU}))})/a31), \omega) \tag{3.55}$$

The rotation rate tensor is also normalised using the limited time-scale and is calculated in the following way.

$$R_{ij} = \text{dev}(\text{tgradU})/\text{timel} \tag{3.56}$$

The implementation of all the 10 tensors and 5 invariants using S_{ij} and R_{ij} can be found in appendix D.1 the definitions are in Table 3.1 and Table 3.2. These same definitions are used during post-processing of the frozen approach. This makes sure that in the later stage during the model discovery exactly the same inputs are used as the output functions of the model.

3.3.4. Verification of code implementation in OpenFOAM

To make sure the method has been implemented correctly various code verifications have to be performed. All the different methods have to be verified separately. This is done using the periodic hills flow case used by Schmelzer and Dwight [18]. This is a small 2D case and the reference data from the original implementation in OpenFOAM 2.4.0 from [18] is available to do the verification.

Frozen approach verification

The first method that needs to be verified for successful implementation is the frozen approach. This is done by taking identical input data as used by Schmelzer and Dwight [18] and running the frozen approach using an identical setup.

The resulting correction fields are then compared to the original, any large differences indicate something is wrong in the implementation. In Figure 3.3 the R correction fields are plotted for the two different implementations in the different OpenFoam versions. In the middle the difference is plotted in a percentage relative to the original. What can be seen is that overall the error is under 1% difference between the versions, with the largest differences being in the regions where the correction is 0 and therefore the tiniest difference gives a large value percentage-wise.

In Figure 3.4 the same is done for the anisotropy tensor. One should keep in mind that this test case is only 2D and therefore since the anisotropy tensor is symmetric, there are only three relevant non-zero tensor

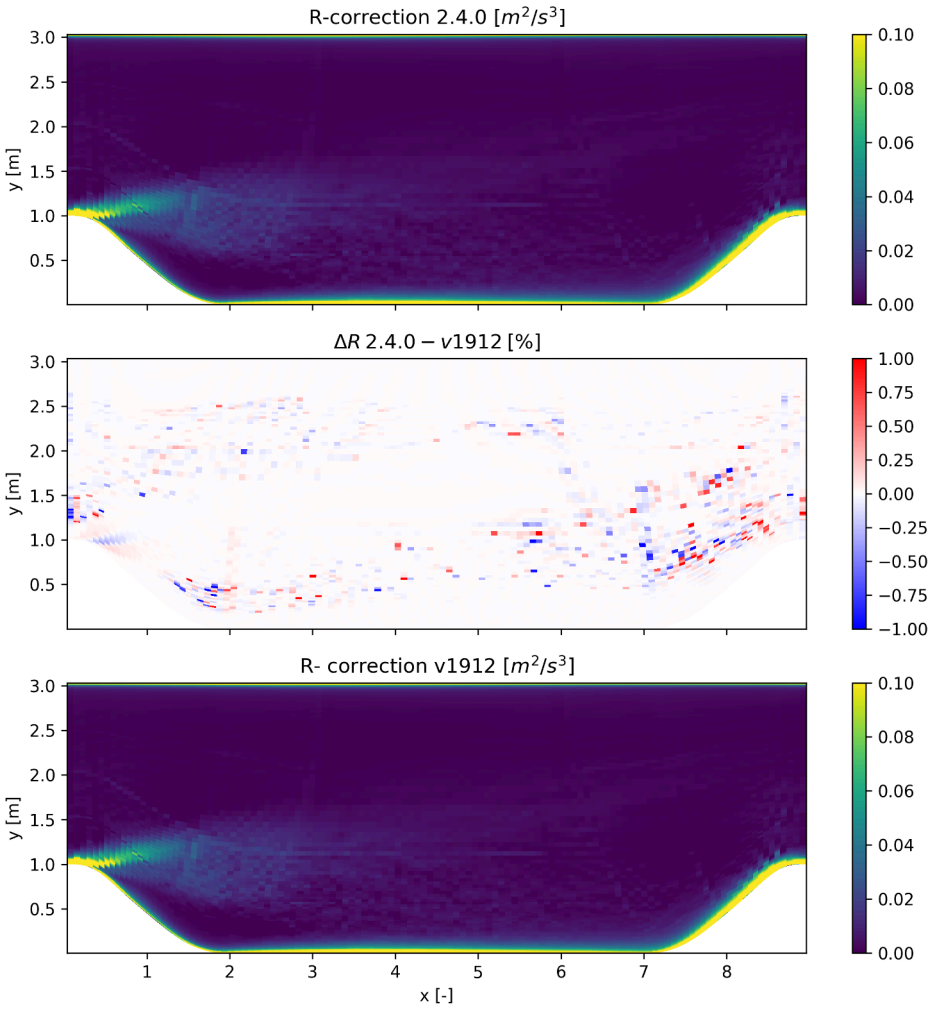


Figure 3.3: Verification of the implementation of the frozen approach by comparing the new implementation in v1912 with the original code in version 2.4.0. Difference between the anisotropy-correction fields in percentage of the original field.

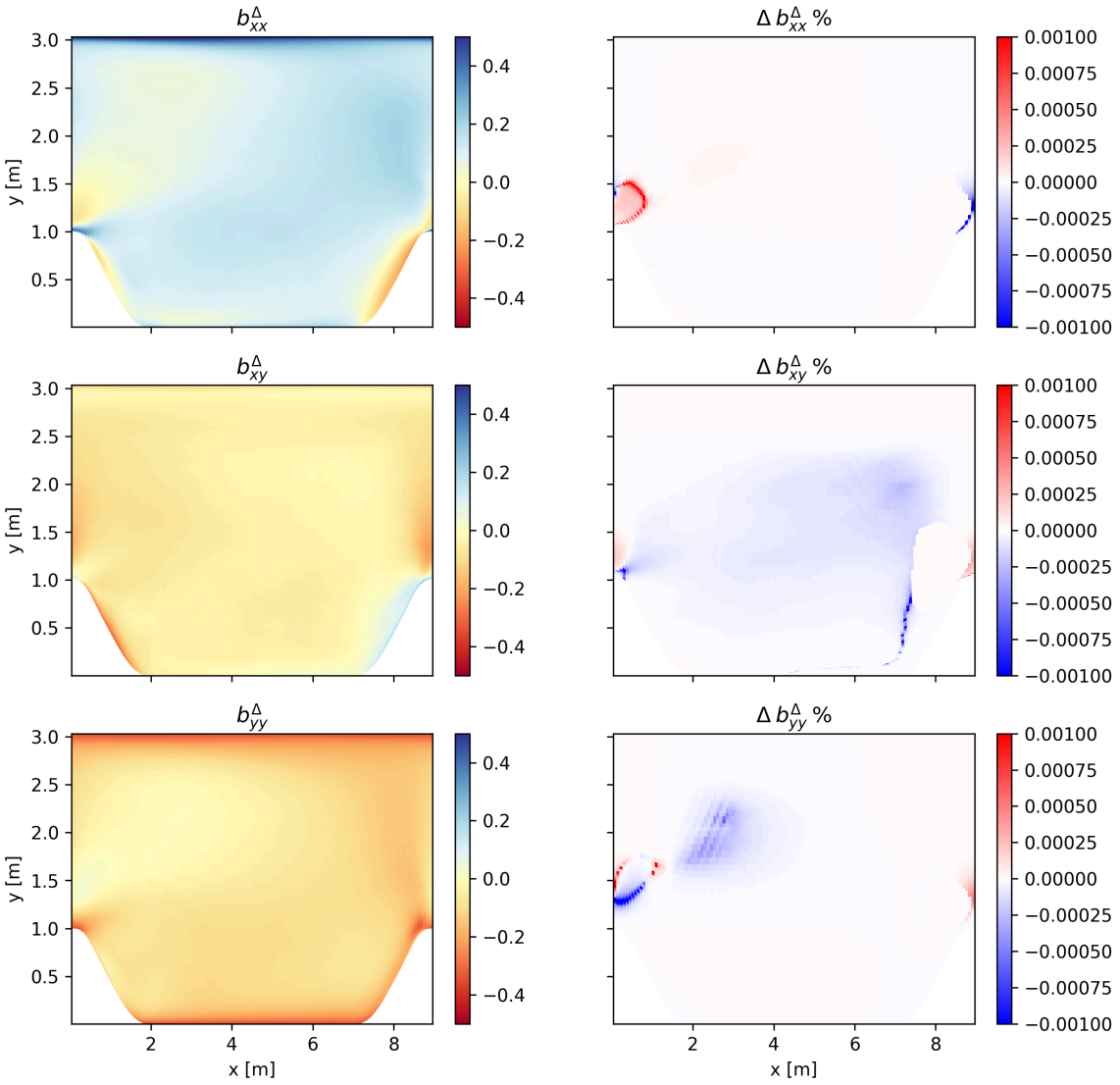


Figure 3.4: Verification of the implementation of the frozen approach by comparing the new implementation in v1912 with the old code in version 2.4.0. On the left: the original field from 2.4.0 on the right: difference between the b^{Δ}_{ij}-correction field in percentage of the original field.

components. In this image on the left are the values as predicted by the original version and on the right the difference between the old version and the new version. What can be seen here is that the difference between the two versions are very small < 0.001%. This ensure the correct implementation of the method, with the final negligible differences coming from the differences in the implementation of the solver and boundary conditions itself, which has changed significantly between the updates.

Code propagation

To check if the propagated approach has been implemented correctly the corrective fields from the frozen approach are fed into the propagated approach solver.

The first test that is done, is to start from the LES solution and the ω field from the frozen approach. With these settings only the transport equations are solved, for which the residuals should be close to zero. If this is the case the implementation of the corrective fields in the turbulence model has been done correctly. After that that the solver is allowed to solve the continuity and momentum equations as well to see if the solution

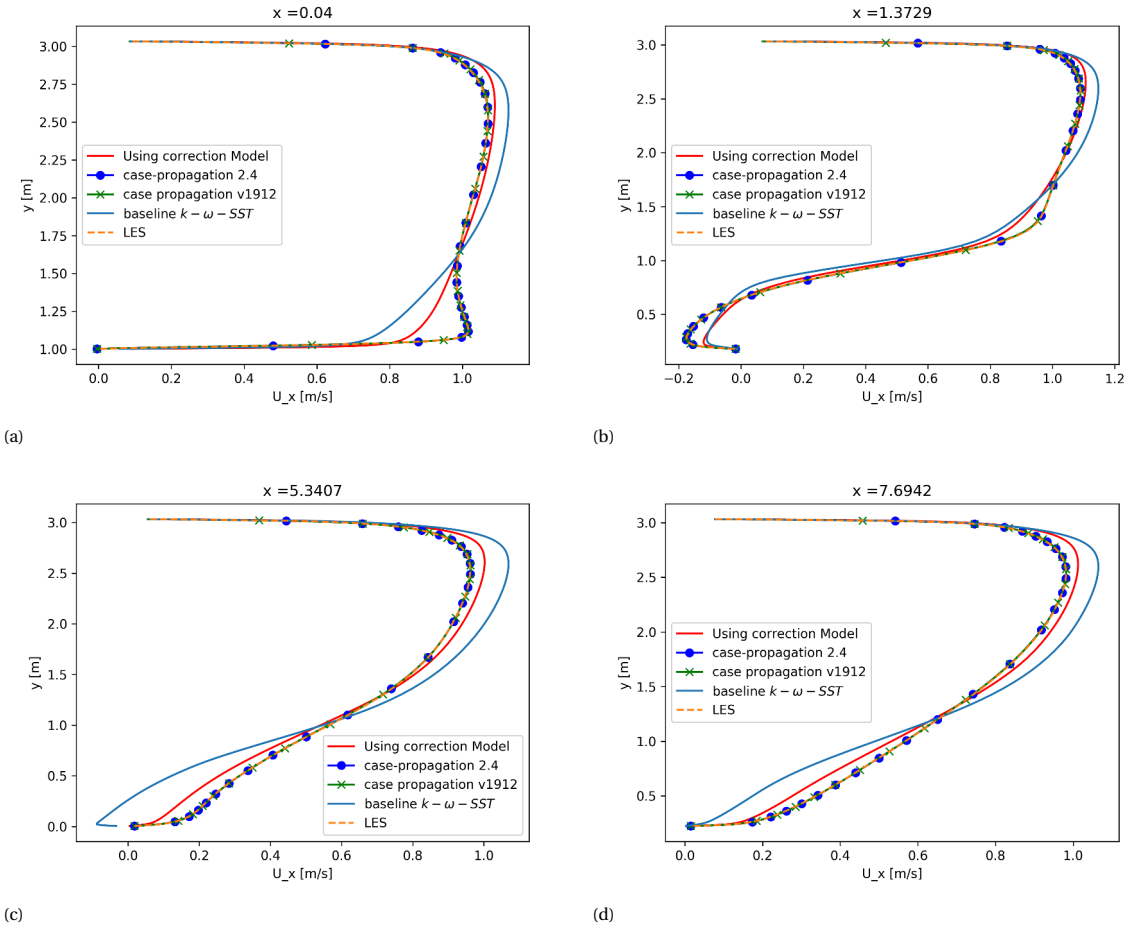


Figure 3.5: X-component velocity comparison between OpenFOAM 2.4.0 and v1912 implementation.

moves away from the LES starting point or stays approximately the same. The Residuals for this case should be small as the final result approximates the LES solution almost perfectly, as was shown by Schmelzer and Dwight [18].

This first test showed that the residuals of the transport equations were equal to zero, ensuring the correct implementation of the correction fields. Secondly when solving for the velocity and pressure the residuals were small and the solution converged to a solution close to the LES solution. This allowed to move forward with the final step in the verification process of the propagated approach.

This final test is to perform the propagated approach starting from the baseline RANS solution. Also this approach starting from the baseline solution should converge to a solution very close to the LES solution. The results of this can be found in Figure 3.5.

Modelled approach

Finally the modelled approach needs to be verified, this is done by implementing one of the models that were discovered by Schmelzer and Dwight [18] and comparing the resulting solution for the periodic hills test case. The model equations are given in (3.57) and (3.57), the results of this can also be found in Figure 3.5, the modelled line is compared to the graph presented by Schmelzer and Dwight [18] in Figure 2.14 and seems to give an identical improvement, with that the modelled approach is also verified to be implemented correctly.

$$b_{ij}^{\Delta} = (24.94{I_1}^2 + 2.65I_2)T^{(1)} + 2.96T^{(2)} + (2.49I_2 + 20.05)T^{(3)} + (2.49I_1 + 14.93)T^{(4)} \tag{3.57}$$

$$R = 0.4T^{(1)} \tag{3.58}$$

3.3.5. Model discovery implentation in Python

The model discovery step is done using python as the main code, making use of the scikit-learn libraries for the regression steps of the method. With the frozen approach completed all the necessary information is available for the model discovery step. The methodology for the model discovery is in basis that of Schmelzer and Dwight, some modifications to this are made in order to make the method more applicable for larger problems. As the geometries that were tested in the original were limited to 2D problems with a small number of cells, the methodology had to be altered and optimised for the larger 3D problems. The number of cells has changed from $O(10^4)$ to $O(10^7)$ which means a much larger memory requirement and longer computation time.

In the original code from Schmelzer and Dwight the calculation of the inputs for the discovery, the tensors $T^1 - T^{10}$ and invariants $I_1 - I_5$ were calculated in the python code. Since python is an object based language and limited to a single thread for computations the calculation of this becomes inefficient, even though the optimised numpy modules are used. Therefore an alteration was made to calculate the inputs for the model discovery step during the postprocessing of the frozen approach. This uses the efficient c++ libraries already available in the OpenFOAM code and is therefore a lot faster. After this the python code loads the data from the files written by the OpenFOAM postprocessing.

This alteration has resulted in a significant speedup, reducing the total importing and computation time from several hours to a couple of minutes for a problem with 1 million data points. This covers the first step of the code, which is the importing ad calculations of the inputs and targets for the machine learning part.

Before continuing with the the model selection step, a library of candidate functions needs to be created. This library consists of lists of models stored as strings. These are created by combining and multiplying the invariants multiple times, after this the simpy module is used to simplify the mathematical expressions. After this list of all different combinations of invariants up to a certain order has been created, all the double entries are removed. This list is then combined with each of the tensors to create the full library of candidate functions.

This list of library functions can be as long as 300 unique functions. The number of candidate functions is of great influence on the computational cost of the elastic net regression. Therefore it is a good idea for computational efficiency to reduce this number. To do this the so called cliqueing approach is used. Functions are grouped according to similarity and from each group the 'simplest' function is selected. The implementation of this method has been done using the numpy covariance function, the limit for functions to be in the same group can be adjusted and as a baseline a value of 0.99 is being used for the covariance limit. Setting this value higher gives more unique functions, having a lower value gives less unique functions but might result in a library that misses features.

This operations is one of the most demanding on the memory, as it evaluates all the candidate functions at all the locations. After this step the number of candidate functions is significantly reduced, reducing the memory requirement for the further operations. To reduce the memory requirement of this first step, sub-sampling is used. The simplest form of sub-sampling is where the data is taken from every Nth point in the data-set. This form of sub-sampling is quick and efficient and does not give preference to any regions in the domain. It might not be the most efficient method, as data can be picked at the locations of interest. Using the simplest case an investigation was done into what the limit of the sub-sampling is to still give the same candidate functions as the output of the cliqueing procedure. It turned out that using $1/10th$ of the original set still gives the same library of functions to do the model selection with.

The reduction of this data-set is of crucial importance to be able to run the model discovery on reasonably sized machines as without this the memory requirements can go up to 100gb or more for problems with 1 million cells. This same problem continues in the first step of the model discovery, doing similar research it turned out that $1/5th$ of the original data-set is enough to get the same functions from the model selection step.

This model selection step uses the elastic net regression function from the scikit-learn library. The net requires as inputs the candidate functions evaluated all the points, the target values b^{Δ}_{ij} or R and the regression

settings. These are the alpha coefficient, which determines the size of the l_1 and l_2 regression coefficient and the ratio, which determines the ratio between l_1 and l_2. Finally the maximum number of iterations and the tolerance need to be given as inputs.

Before the elastic net is used to determine which candidate functions to use in the model the inputs and outputs are normalised. This normalisation is done because the optimisation algorithms work better when all the inputs and outputs are normalised. The normalisation means that all the inputs are scaled to unit norm, making sure all the values are roughly in the range of -1 to 1.

The settings for the elastic net concerning tolerance and maximum number of iterations have an influence on the computational time and the model shape. Since this step is not the final step in the regression it is not of crucial importance to have absolute convergence of the method. However, from experiments it was found that letting the elastic net run for longer, sy setting a lower tolerance and higher number of maximum iterations the models that were found were generally more satisfying in terms of sparsity and coefficient size. Therefore, the decision was made to lower the tolerance from $1e-4$ to $1e-7$ from the original settings.

The final step is the ridge regression, it is used to have control over the coefficient size. This ridge regression step is applied to all the model forms found in the model discovery step. In the original version the input for this method was using a data-set that contains the evaluation of all the candidate functions and taking those values that were needed for the evaluation of the selected model form. In this way the script runs through all the model forms that were found by the elastic net regression. To optimise for memory and performance it was decided to still use the full data-set but reduce the input library to just the functions needed in the model form. This means the computations will take slightly longer because certain candidate functions are recalculated multiple times. However, it does allow to do the final data fit with the full data-set on a smaller machine.

This ridge regression was done using different ridge alpha's to create different coefficient sizes. For the ridge coefficient $\alpha = 0$ the ridge regression is equal to a least squares fit. This ridge step results in the coefficients that combined with the model forms results in a large library of correction functions. Each function with different coefficient sizes and sparsity due to the variation in elastic net coefficients and ridge coefficient. These models are then sorted on their respective error on the training data and visualised in a graph, where using a logarithmic scaling the coefficient size is transformed onto a colourmap. This visualisation is then used to pick a well balanced model for implementation into the turbulence model.

3.4. Methodology test procedure

Three test cases have been selected to test the SpaRTA methodology on. The selected geometries are the wall mounted cube, an infinite circular cylinder and an idealised rotating wheel. These test cases are ranging up in terms of complexity and mesh size, with the surface mounted cube having the smallest mesh size and the idealised wheel the largest.

The test procedure that is followed is the following. The first step is the collection of the high fidelity data, that is by running an LES for the cylinder and the wheel and by running a DES for the surface mounted cube. It was chosen to do a DES for the surface mounted cube, to keep the computational cost relatively low. This helped in the first stages of testing the methodology as it allowed the procedures in the SpaRTA methodology to be run quickly multiple times, helping in the testing and optimisation of the code before moving on to the computationally more expensive test cases. The choice for a DES as high fidelity data source had not been done before for this method and therefore this was the perfect opportunity to test this.

The second step after the collection of the high fidelity data is to run the frozen approach. This step requires the mean flow velocity data, the turbulent kinetic energy and the Reynolds stress tensor as calculated from the high fidelity simulation. The frozen approach also requires the setup of boundary and initial conditions for the variables used in the turbulence models. Correct setup of this is essential for the convergence of the frozen approach. As is the statistical convergence of the high fidelity data as will be shown later in the results chapter.

With the frozen approach converged the next step is post-processing of the results, this step involves the calculation of all the input features for the model regression ie. the tensors and invariants from Pope's eddy viscosity hypothesis. With this all the data required for the model discovery step is available. In the meantime the propagation test can be completed with the addition of the correction fields in the solver.

Depending on the size of the data-set sub-sampling is applied. This is done to stay withing the memory limits of the machine the model discovery is run on. For now the sub-sampling is done by taking data every N points, this leads to a huge reduction in data size. How much the data is sub-sampled depends on the size of the data-set. For optimal use of the data for different steps different levels of sub-sampling are used. This is to maximise the accuracy while remaining within the memory limits. The operations that follow are the creation of the library of candidate functions, reduction of this library through the cliqueing operation after which the model-forms are discovered through the elastic net regression. For each model-form the coefficients are determined through a ridge regression. This completes the model discovery step and results in a large library of correction models is available.

The next step is to pick one of the correction models from this large library of functions. To help with the decision making for which model to pick the models are compared by visualising the number of basis functions and the size of the coefficients and comparing this to the relative error on the training data.

After picking a model and the implementation of this model into the turbulence model, test simulations are performed. In this test the correction terms in the turbulence model are slowly increased. Usually the R correction term is directly added in full to the turbulence model. With the anisotropy correction b_{ij}^{Δ} being added slowly up till such an amount that the solution still converges. This test is then done for different models to determine the right balance between model complexity final result accuracy and stability. This corrected solution is then compared to a baseline RANS simulation with the uncorrected turbulence model and the high fidelity simulation to asses the accuracy of the solution. Once a model is found with a good balance between performance and complexity this model is stored to be applied to the other test cases as well to check for the extrapolating qualities.

4

Results

In this chapter the results of the application of the SpaRTA methodology on the various test cases are presented. They are compared to the high fidelity reference data to get an idea of the performance of the method and the extrapolating capabilities of the different models.

In Section 4.1 the results of the method applied to the wall mounted cube are presented. This test case has been used extensively for several tests and experiments to better understand the effects of the correction models. This test case has also been used to verify the assumption that the first 4 tensors and 2 invariants of the tensor basis presented in Section 3.2.1. It has also been used to investigate the stability of the models and the effect of the magnitude of the correction.

Secondly the hardest of the test cases is discussed in Section 4.2, which is the infinite cylinder. This test case is particularly difficult due to its unstable nature. The main challenge for this case was finding stable solutions for the baseline RANS and with correction models applied. This case is also used to test the extrapolating qualities of the correction model that was found on the wall mounted cube test case.

Finally in Section 4.3 a larger case has been explored with the application of the correction model to an idealised rotating wheel. This case also showed improvements with two models trained on different geometries showing very similar results.

4.1. Wall mounted cube

The wall mounted cube has been selected as a test case for a couple of reasons. Firstly it is a geometry similar in topology to the idealised wheel, which will form the most complex test case of this research. The wall mounted cube is expected to create a horseshoe vortex, which is a similar flow features as seen in literature on the idealised wheel. Another reason to use the cube as a test case is because it has separation and reattachment regions of which the location is dependent on the turbulence model. Improvements in the turbulence model will result directly into shifts in the separations and reattachment locations. Besides this the cube is a relatively simple model that allows the use of quickly generated structured meshes with local refinement areas. Finally literature is available on both experimental and computational research that has been performed on the same or similar geometries, this forms a good basis for the following research.

The geometry of this flow case is that of a cube of dimensions $1m$ mounted on a flat plate with another flat plate $1m$ away from the top of the cube. The sides of the domain are treated as symmetry planes, the geometry is shown in Figure 4.1. The inflow velocity is set to 1m/s with a dynamic viscosity of 2.5×10^{-5} to get a Reynolds number of 40000.

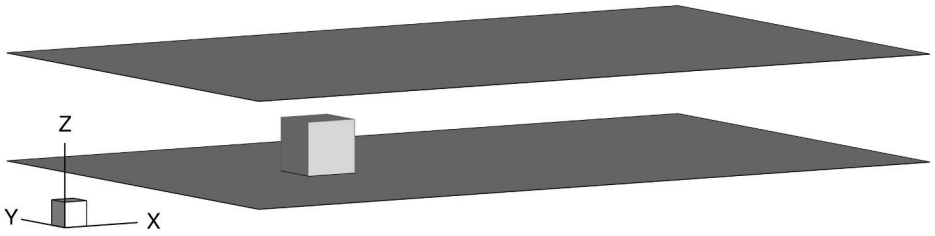


Figure 4.1: Geometry for the Wall mounted cube

The structure of the remainder of this Section is as follows:
In Section 4.1.1 the DES that was run to gather the high fidelity data is discussed and compared to an experimental setup of a similar geometry. After this in Section 4.1.2 the setup of the baseline RANS simulation is presented, this simulation will serve as baseline reference to compare the correction models to.

The results from the frozen approach which is explained in Section 3.2.2 are shown in Section 4.1.3. The direct application of the corrective fields found in the frozen approach is presented in Section 4.1.4, here the first difficulty with the stability of the solution is encountered.

In Section 4.1.5 the library of correction models resulting from the regression are presented, here it is explained how a correction model is picked and the effect of the elastic net and ridge regression methods explained in Section 3.2.3 can be clearly seen in the visualisation of the models.

The main result, the application of the regression model selected from the library of models is presented in Section 4.1.6. Here the improvement in the resulting flowfield is shown and the effect of the correction model on the converged solution is highlighted. The test in this section was not done with the full correction fields due to stability issues. An investigation on the effect of each correction field and the magnitude that is added to the equations is presented in Section 4.1.8. Here results are presented as well with the full correction fields, but with solutions that are averaged over the last 1000 iterations to deal with the instability created by the anisotropy correction field.

In Section 4.1.7 the assumption is tested that the first 4 tensors and 2 invariants from Pope's viscosity hypothesis [16] are sufficient for modelling the correction fields, as explained in section 3.2.1. The application of two models, one with the full tensor basis and the other with a limited number of tensors showed very small differences with a slight favour for the model with the limited number of tensors and invariants, thereby confirming the ansatz made in Section 3.2.3.

Finally the first test of the extrapolating qualities of the correction model is shown in Section 4.1.9. Here the correction is applied to a very similar geometry but slightly modified geometry. For this test the cube has been lengthened to 1.5 times the original length. This first test shows promising signs of the extrapolating qualities of the model. Later in Section 4.2.6 and Section 4.3.5 this model is applied on the cylinder test case and wheel test case to further proof the extrapolating qualities.

4.1.1. DES as high fidelity data source

For the set of high fidelity data that will be used to improve turbulence modules, the decision was made to do a DES. This decision was made because running a DES is significantly cheaper than running an LES. It is also a good test to see if the methods that were previously proven to work by applying an LES data-set as training data also works with DES data as training data.

DES Mesh and simulation setup

The main difference between the DES and LES comes down to the way the flow near the wall is resolved. The

Number of Cells	1216000
Cell Type	Hexahedra
Max Aspect Ratio	81.949
Min Volume	2.471e-6
Max Volume	0.0178
Mesh max non-orthogonality	0
Max skewness	4.092e-13

Table 4.1: Characteristics of the mesh used for the DES of the wall mounted cube.

DES uses RANS models in the near wall regions and uses the LES approach for the cells further away from the walls. For this DES the setup that has been used is to use the Spallart Almaras IDDES turbulence model for the RANS regions and is also used as the SGS model.

The mesh that has been used for this simulation is created using the OpenFOAM blockmesh utility. The aim for this mesh is to achieve a value of max $y+ = 30$. This has been done because wall functions are being used. The computational domain has a size of $14.5 \times 9 \times 2$ ($l \times w \times h$). With the cube placed on the bottom surface in the centre of the domain in terms of width and with the front face 4 meters behind the inlet.

The quality of the mesh was checked using the openFOAM checkMesh utility, of which the results can be found in Table 4.1. A general overview of the mesh is shown in Figure 4.2, where the refinement regions can be seen to be in close proximity to the faces of the cube. Due to the nature of the mesh these refinement regions expand throughout the entire domain.

The Reynolds number is set to 40000 with an inflow velocity of $1 m/s$ and a kinematic viscosity $\nu = 2.5e-5$ to match for the Reynolds number. For solving the DES problem the incompressible pimpleFoam solver is used. For the inlet the turbulent turbulentDFSEMInlet [15] is used together with a mapped field to initiate the boundary layers on the top and bottom wall. This boundary layer was initialized by doing a RANS of a channel, This boundary layer is initiated in a RANS channel flow. The turbulentDFSEMinlet is a divergence free synthetic eddy method to reproduce the turbulent inflow conditions for an LES or DES.

For the wall boundary conditions, the nutUspalding wall function was used for the eddy viscosity, with k being set to 0 at the wall for the turbulent kinetic energy boundary conditions. The walls are enforced by having a no slip condition for the velocity and a Neumann boundary condition for the pressure by enforcing a zero gradient.

The solver was run for a total of 200 seconds of simulated time, with averaging starting at $T = 20s$ resulting in an averaging period of 180 seconds or 12.4 flow-throughs. Also data with a shorter averaging time of $110s$ has been retrieved for a comparison of statistical convergence and to see how much the statistics need to be converged in order to be able to apply the SpaRTA approach.

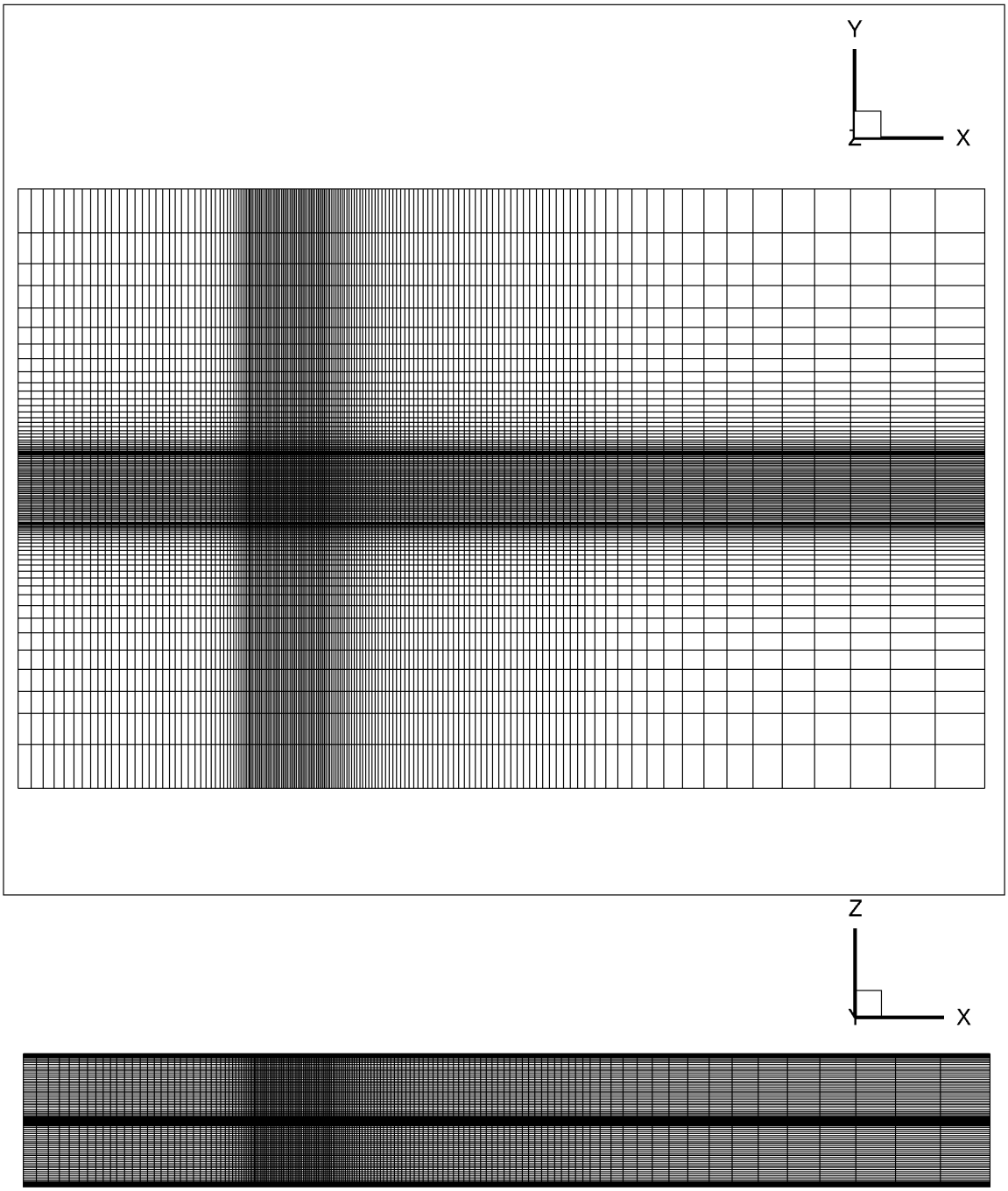


Figure 4.2: The mesh that has been used for the DES of the wall mounted cube, total number of cells is 1.2 million. With a maximum $y+$ of 30.

DES Results
In Figure 4.3 the streamtraces of the average flow around the cube are shown. What can be seen is that the flow around the cube forms a horse-shoe vortex that is formed in front of the cube due to the interaction of the ground boundary layer and the cube stagnation point. This horseshoe vortex travels downstream of the cube the direction of rotation on this vortex pair is symmetrically opposite and is rotating down at the sides of the cube.

The other flow feature that are distinctive for this flow case are the separations that occur at the leading edges of the cube. Behind this separation a recirculation area can be seen where there is backflow, this phenomenon occurs on both the top and the side surfaces of the cube. On the side surfaces with the interaction of the bottom boundary layer this circulation area attaches on the bottom surface, which can be clearly seen in Figure 4.4a where the surface streamlines are plotted on the bottom surface. They are compared to experimental results at a different but comparable Reynolds nuber from Martinuzzi and Tropea [13] in Figure 4.4b. The streamtraces show a good between the DES solution and the experimental results. The shape of the horshoevortex around the cube shows good similarity and the reatachement location of the flow on the bottom of the domain shows good agreement with the reattachment location of the experimental result.

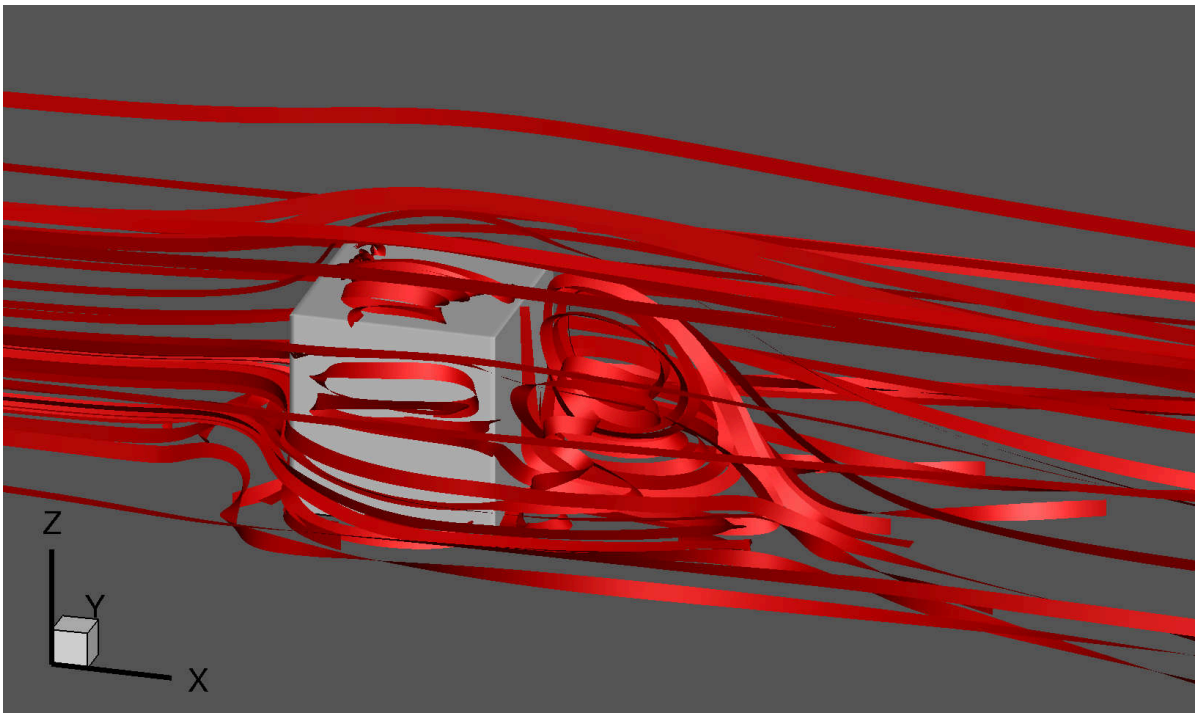
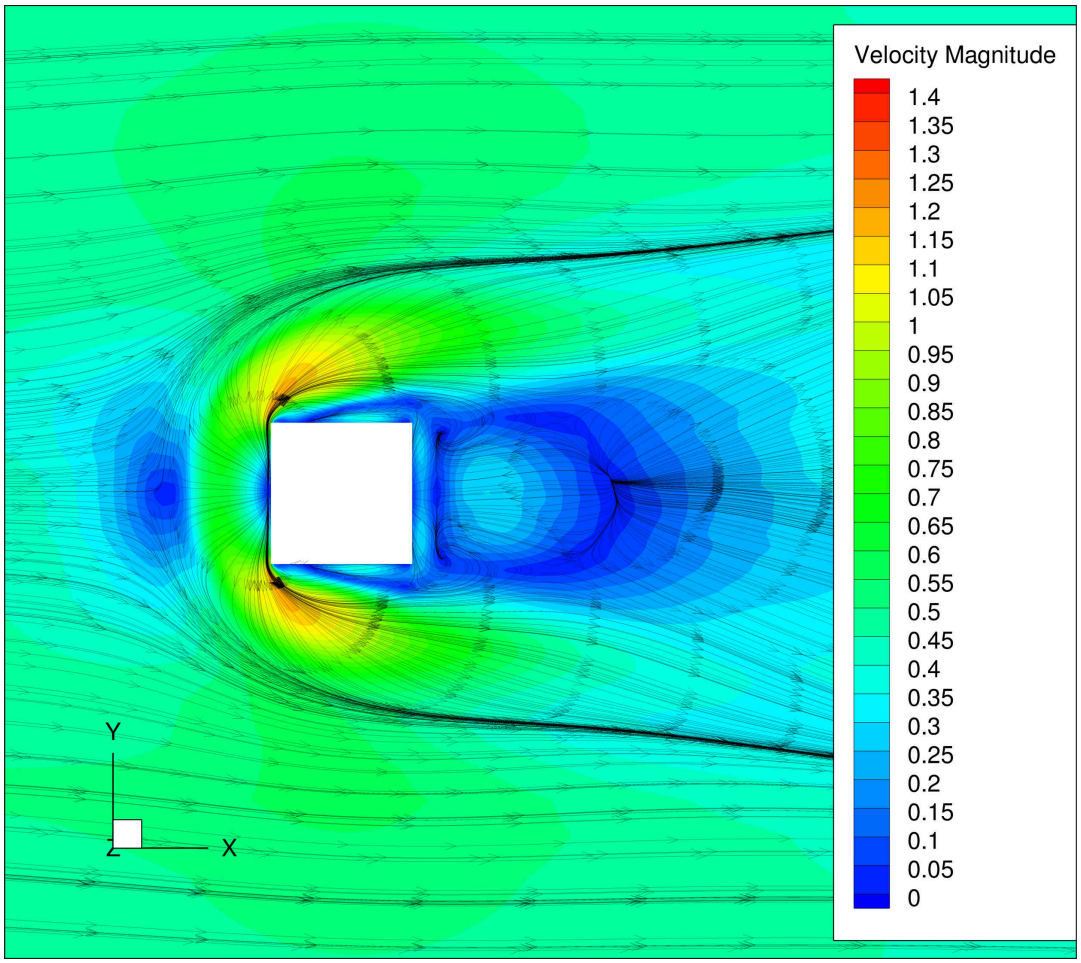


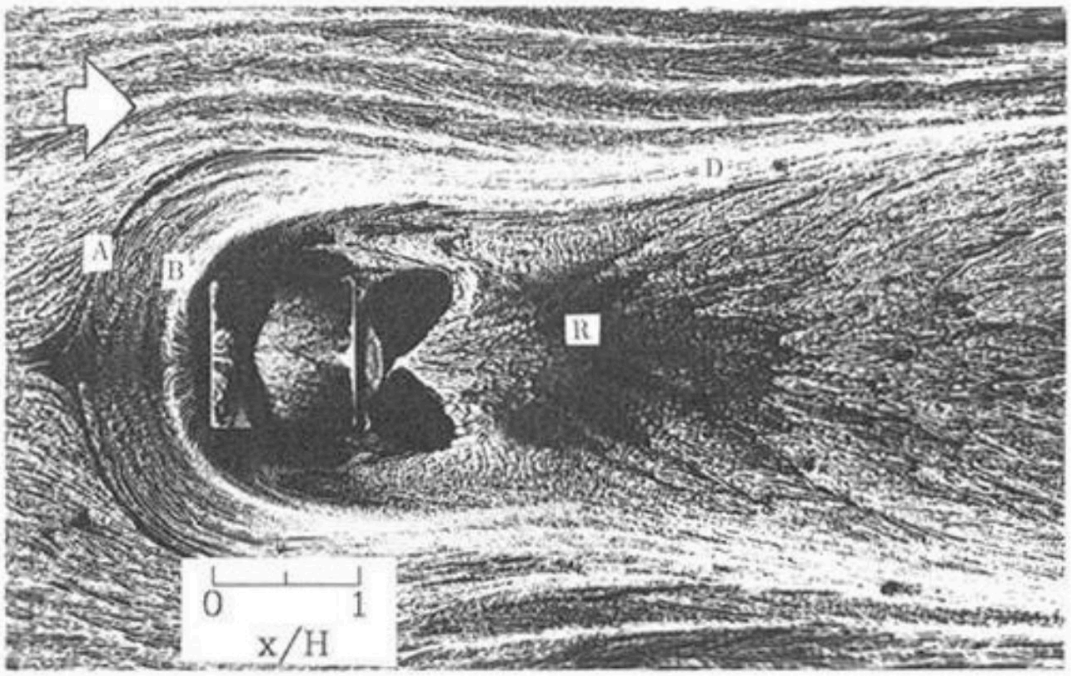
Figure 4.3: Visualisation of the flow around the cube using stream ribbons The results are from the DES simulation.

From the streamtraces it can also be seen that the flow reattaches on the top and side surfaces, to then separate again at the trailing edge of the cube. These separations cause another recirculating area directly behind the cube, shaped in a horseshoe standing up straight. Figure 4.4a also shows very clearly the two recirculation areas that are behind the cube originating from the separating flow coming from the sides of the cube.

The wake behind the cube is characterised by a drop in total pressure behind the cube compared to the total pressure in freestream conditions. Further on the size and shape of the wake is compared to that of different simulations. The shape of the wake is visualised using an isosurface where $Cp_0 = 0.5$. Doing this shows the shape of the wake which is longer than just the recirculation areas behind the cube. The length of the wake shows the effects of turbulence production and dissipation. As turbulence production generally leads to flow that will remain attached for longer. Having more turbulence dissipation will lead to a faster breakdown of turbulent structures into smaller eddies, which should result in the faster dissipation of the horseshoe vortex structure downstream of the cube.



(a) Velocity magnitude and streamtraces at the bottom of the cube from the DES simulation. $Re = 40000$



(b) Experimental results from the wall mounted cube $Re \approx 80000 - 120000$. Figure taken from [13].

Figure 4.4: Comparison of the streamtraces on the bottom of the domain between the DES solution and experimental simulations from Martinuzzi and Tropea [13].

4.1.2. Baseline RANS

To be able to have a comparison and get an estimate of the performance of an unmodified $k-\omega$-SST model a baseline RANS simulation has been run. This baseline RANS simulation was done on the same mesh as the DES for consistency and to make the comparison easier. This RANS simulation was run until the residuals were no longer decreasing.

The inflow and boundary conditions were kept the same to that of the DES, to create a fair comparison. The only thing that was changed was the inflow condition, where the synthetic eddies were replaced with a constant inflow from the mapped inflow condition. The simulation was run using the simpleFoam solver, which is an incompressible RANS solver.

Further on in this report a more in depth comparison will be made between this baseline RANS solution and the DES, when comparing with the correction. From Figure 4.5 a big difference with the DES solution can be spotted immediately. That is the wake of the flow behind the cube is a lot bigger than that of the DES solution. The recirculation area behind the cube is much longer and the point where the flow reattaches to the bottom surface is further downstream. Another thing that is different compared to the DES is the recirculation area on the top of the cube and the sides of the cube. A more detailed comparison will be made later when the corrected model is compared to the DES solution and this Baseline run.

Figure 4.4a gives a good overview of the location of the horseshoe vortex and the shape of the wake. At first sight it can be again seen that the length of the wake is a lot longer than that of the DES solution. Besides that the shape is different and when looking at the a-symmetry of the wake it must be noted that there is still some instability in the solution. This instability is something that can be reduced by changing numerical settings and running the simulation for longer but for now it is a good opportunity to see what applying a correction model does to the stability of the solution.

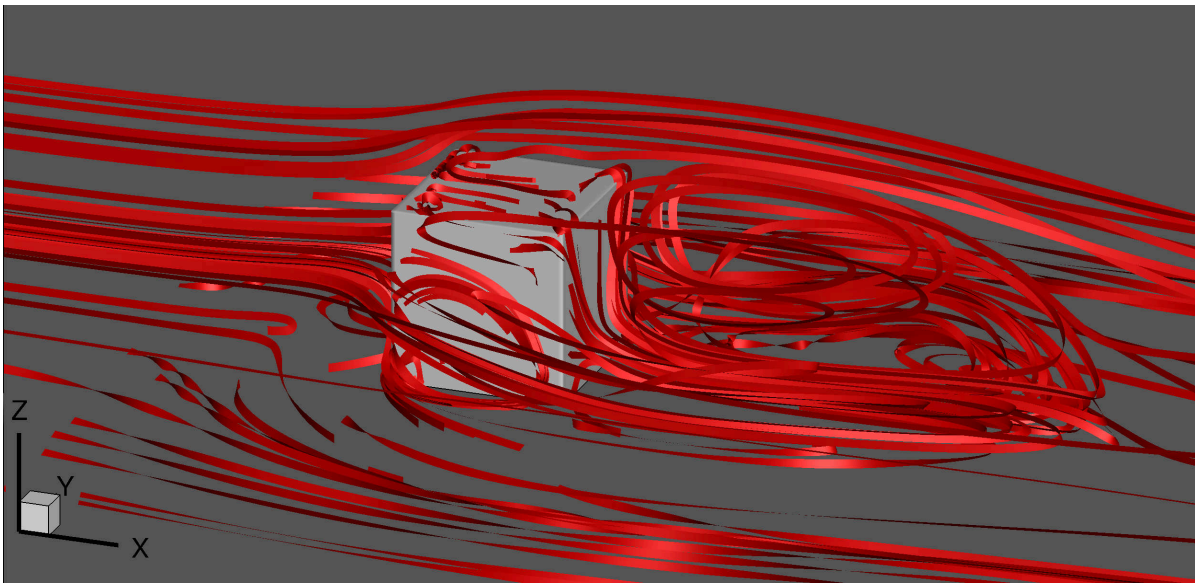
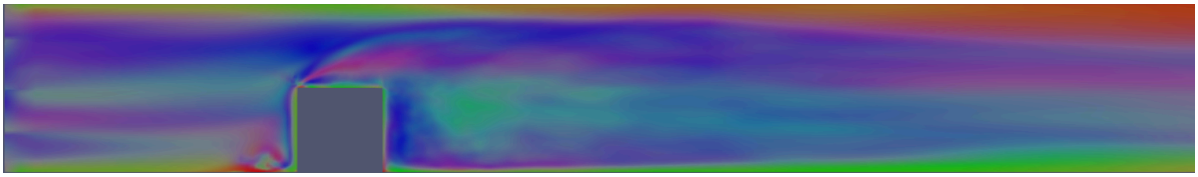


Figure 4.5: Flow visualisation using streamtraces. Simulation is the baseline RANS simulation with standard $k-\omega$-SST turbulence model at $Re = 40000$.

When looking a the shape of the horseshoe vortex, it can be seen that the trajectory of the vortex around the cube is a lot wider and that the trajectory is different compared to the DES. It also seems like there is a second smaller vortex in front of the main vortex, this vortex is not visible in the DES field.



(a) Reynolds Anisotropy as calculated by the DES

(b) Anisotropy field as predicted by the Boussinesq hypothesis

Figure 4.6: Reynolds anisotropy comparison between the DES and the Boussinesq hypothesis, slice at centre of the cube y=4.5. Visualisation through a Barycentric mapping as according to Figure 2.3.

4.1.3. Frozen Approach

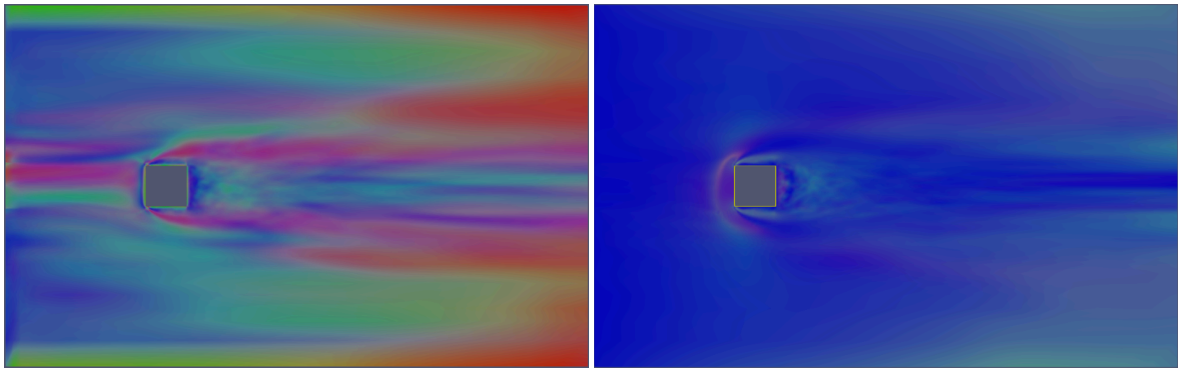
For the frozen approach the starting conditions are taken from the mean DES velocity field, the turbulent kinetic energy that is retrieved from post-processing, similar to the calculated Reynolds stress tensor that will be used to calculate the corrective fields with. The variables that are used in the turbulence model need boundary conditions. These are the conditions that are also used in the baseline RANS. Therefore, for ν_t the nutUspalding wall function is used and for omega the omegaWallfunction is used.

The initial fields for ω and ν_t are set to fixed values that were determined based on the baseline RANS that was run. From experience it should be noted that having the right boundary and initial conditions is critical for the stability of the frozen approach. Having the wrong boundary conditions and an initial value of ω that is too small will result in instabilities and ω values going to infinity locally. Having a large value of ω as a starting condition helps with the stability and helps convergence towards a stable solution for omega and the corrective fields.

In Figure 4.6a the anisotropy field is visualised using the Barycentric map colouring from Figure 2.3. In this figure a slice through the centre of the cube can be seen with the anisotropy field visualised as is calculated by the DES, in Figure 4.6b is the anisotropy field that is predicted by the Boussinesq approximation. This figure highlights the necessity for the need of a correction to the anisotropy field. In the SpaRTA approach the b_{ij}^{Δ} field is trying to correct for this discrepancy. As the Boussinesq hypothesis results in a prediction of 3d turbulence almost everywhere while definitely at the walls the anisotropy should be forced to be 2d and therefore resemble a green colour. The same difference can be seen when looking at the anisotropy field near the bottom wall, in Figure 4.7a the turbulent structures are mainly calculated to be 2D and 1D structures, while the prediction from the Boussinesq hypothesis is mainly 3d turbulence as shown in Figure 4.7b.

A crucial component of the SpaRTA approach is to close and correct the k-equation of the $k-\omega$-SST model. This is necessary because with this newly corrected anisotropy field the production term has been changed and therefore the k-equation requires an additional closure term called R. This corrective field acts on the turbulent kinetic energy and directly adds or removes turbulence production locally. This field is shown in Figure 4.8b, it shows the addition of turbulent kinetic energy all over the entire top surface of the cube. This highlights that additional production is needed to reduce the separation area seen on top of the cube as is predicted by the baseline RANS, which can be seen in Figure 4.8a. Besides this it appears that just above this edge there is a reduction in the turbulent kinetic energy.

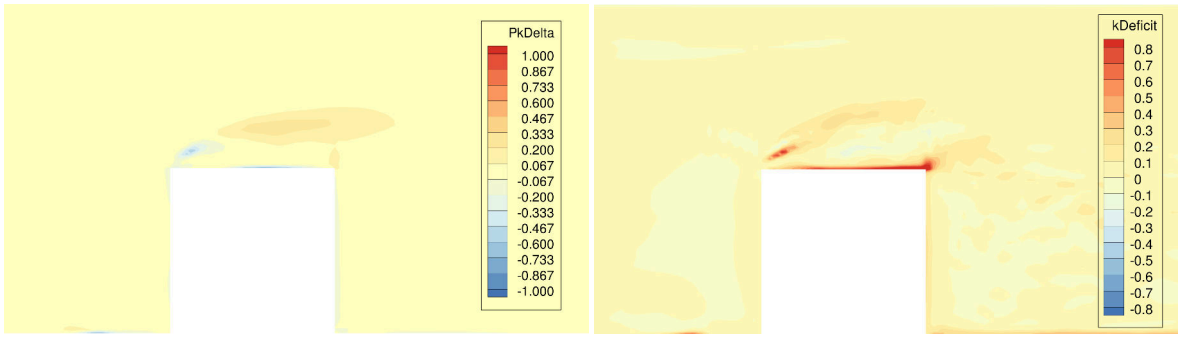
Also when looking at the location of the horseshoe vortex in front of the cube some correction fields can be seen. In the $k-$correction a positive and negative field can be seen, which would result in a different location and shape of the horseshoe vortex. It does not guarantee this but it does proof hat a baseline RANS solution would never reach the position where the horseshoe vortex is in this location.



(a) Reynolds Anisotropy as calculated by the DES

(b) Anisotropy field as predicted by the Boussinesq hypothesis

Figure 4.7: The Reynolds anisotropy comparison between the DES and the Boussinesq hypothesis, slice at z=0.5. Visualisation through a Barycentric mapping as according to Figure 2.3.



(a) Total correction in turbulence production from both the anisotropy correction and the R-correction at the centre of the cube.

(b) Direct correction in the turbulence production R. Calculated from the kinetic energy equation (4.2).

Figure 4.8: The correction field as determined by the frozen approach

4.1.4. Corrective field propagation

A good test is to see if a RANS simulation with the correction fields from the frozen approach converges to the DES solution. This shows if the corrective field is converged enough and if the approach works. The optimal test is to start from the converged baseline RANS solution and add the corrective fields from the frozen approach. This was shown to converge to the LES solution by Schmelzer and Dwight [18]. There is a good chance that the solution will not converge because the corrective field is too disruptive and causes the simulation to diverge. In that case ramping is used, where the corrective fields are slowly added by increasing the amount the correction field is added each couple of iterations once the solution seems converged.

This same approach was applied to the cube. However, with the full correction fields added the solution immediately diverged as ω values went to infinity locally. Therefore the decision was made to slowly introduce the corrective fields into the baseline RANS. However, this approach also failed to get to a stable solution and to correct the RANS solution in the direction of the DES solution. One of the hypothesises for this is that for this specific case the baseline RANS solution is so far off the DES solution that the corrective field is not correcting in the right places to make the solution move towards the DES solution. In the 2D test cases that have been done the baseline RANS solution was already relatively close to the LES solution, therefore the velocity field which the correction fields are based upon is close enough to the velocity field of the baseline RANS. For the cube this is not the case and therefore the solution never starts to move closer to the DES solution with the corrections in these locations.

Another thing that can be tried to verify the correction fields is to start from the DES velocity and turbulent kinetic energy field with the ω field and the eddy viscosity field retrieved from the frozen approach and add the correction fields to this. The residual on the first iteration should be small and the as the simulation progresses the solution should not move away from the initial solution.

This test has been performed for the cube but proved to be difficult due to instabilities. These instabilities originated from the boundaries, where the anisotropy correction is showing extremely large values. This is due to the boundary condition, k is forced to be very small (close to zero) at the wall making the corrections large. This large correction near the wall in combination with the already unstable flow solution for the solver causes the solution to diverge with locally large values for ω. Despite this setback it is still possible to apply the correction through a correction model.

4.1.5. Model Regression

To make the method generally applicable to different meshes and geometries, a model is to be found to reproduce the correction fields. This is done following the regression methodology shown in Section 3.2.3.

The target values for the regression are the correction fields calculated in the frozen approach. The input values for the regression are calculated during post-processing of the frozen approach simulation. These are then loaded into the python script that does the model regression.

The first step of the regression, the model inference step was done using the full data-set consisting of 1.2 million cells. This data-set was scaled and normalised to do the elastic net regression that gives the model form. After that the data-set was split into a train and a testing part, which was then used to the final step the ridge regression. The test part of the data-set was used to determine the mean squared error of the model, which is then used to compare the different models achieved from running with different regression coefficients.

The resulting models from this operation are shown in Figure 4.9 for the models created for b_{ij}^{Δ} and in Figure 4.10 for the models created for the R correction factor. These figures show the size of the coefficients for the basis functions that were used as inputs. In this figure clearly the effect of the ridge α regression coefficient can be seen, the larger the coefficient, the smaller the model coefficients in the final model. Therefore the model that fits the data the best is generated with the ridge $\alpha = 0$ coming down to regular least squares regression. What also can be seen in this figure is the effect of the elastic net coefficients. A total of 16 different model forms were achieved from running the elastic net regression with 500 different coefficient settings. This means that the majority of these settings converge to the same model form. That is the case for the models for b_{ij}^{Δ}, for the R correction more different models came out with the same elastic net settings, namely 29 different model forms.

What can be seen is that from the colours in Figure 4.9 is that there are a couple of dominant model terms that always return in the model with large coefficients are are basis function $1, 2, 6$ and $40-43$ respectively I^1T^1, I^1T^2, I^2T^2, cT^1. cT^2, cT^3, cT^4. It is interesting to see that the simplest basis functions almost always return in the models. The same thing is the case for the models for the R correction, all these models contain the constant times the tensors 1, 3 and 4 term. Tensor number 2 is not present or only with a very small coefficient 10^{-3}. This shows that the R and the b_{ij}^{Δ} correction depend on different mean flow features

1	$I_1T^{(1)}$	9	$I_2T^{(4)}$	17	$I_1I_2T^{(3)}$	25	${I_1}^2{I_2}^2T^{(4)}$	33	${I_1}^3{I_2}^2T^{(4)}$	41	$cT^{(2)}$
2	$I_1T^{(2)}$	10	${I_1}^2T^{(1)}$	18	$I_1{I_2}^2T^{(1)}$	26	$I_1{I_2}^3T^{(1)}$	34	${I_1}^2{I_2}^3T^{(1)}$	42	$cT^{(3)}$
3	$I_1T^{(3)}$	11	$I_1I_2T^{(1)}$	19	$I_1{I_2}^2T^{(2)}$	27	$I_1{I_2}^3T^{(2)}$	35	${I_1}^2{I_2}^3T^{(2)}$	43	$cT^{(4)}$
4	$I_1T^{(4)}$	12	$I_1I_2T^{(2)}$	20	$I_1{I_2}^2T^{(4)}$	28	$I_1{I_2}^3T^{(3)}$	36	${I_1}^2{I_2}^3T^{(3)}$		
5	$I_2T^{(1)}$	13	$I_1I_2T^{(4)}$	21	${I_1}^2I_2T^{(1)}$	29	$I_1{I_2}^3T^{(4)}$	37	${I_1}^2{I_2}^4T^{(1)}$		
6	$I_2T^{(2)}$	14	${I_2}^2T^{(1)}$	22	${I_1}^2I_2T^{(2)}$	30	${I_1}^2{I_2}^2T^{(1)}$	38	${I_1}^2{I_2}^4T^{(2)}$		
7	$I_2T^{(3)}$	15	${I_2}^2T^{(2)}$	23	${I_1}^2I_2T^{(4)}$	31	${I_1}^2{I_2}^2T^{(2)}$	39	${I_1}^2{I_2}^4T^{(3)}$		
8	${I_2}^2T^{(3)}$	16	${I_2}^2T^{(4)}$	24	$I_1{I_2}^2T^{(3)}$	32	${I_1}^2{I_2}^3T^{(4)}$	40	$cT^{(1)}$		

Table 4.2: Basis functions used by the models.

With the library of correction models available it important to pick one that has the best chance of improving the results without making the solution unstable. The general rule of thumb for this is to have coefficients that are relatively small $O(10^1)$ or smaller. Also the number of terms should be limited as a larger variety of terms can have a destabilising effect, especially when terms with a high power are introduced. Also having a lot of terms makes the equations more computationally expensive to calculate. When looking at the library of models however, it becomes clear that the best fitting models are the ones that have the largest coefficients and the highest number of terms. Therefore, a good balance must be sought between model complexity and coefficient size. It is recommended to pick some of the simplest models and verify the performance of these models to then go to more complex models if the results when implemented in the RANS solver is unsatisfying.

The Figures 4.9 and 4.10 help in making a decision what model to pick to implement. The model coefficients

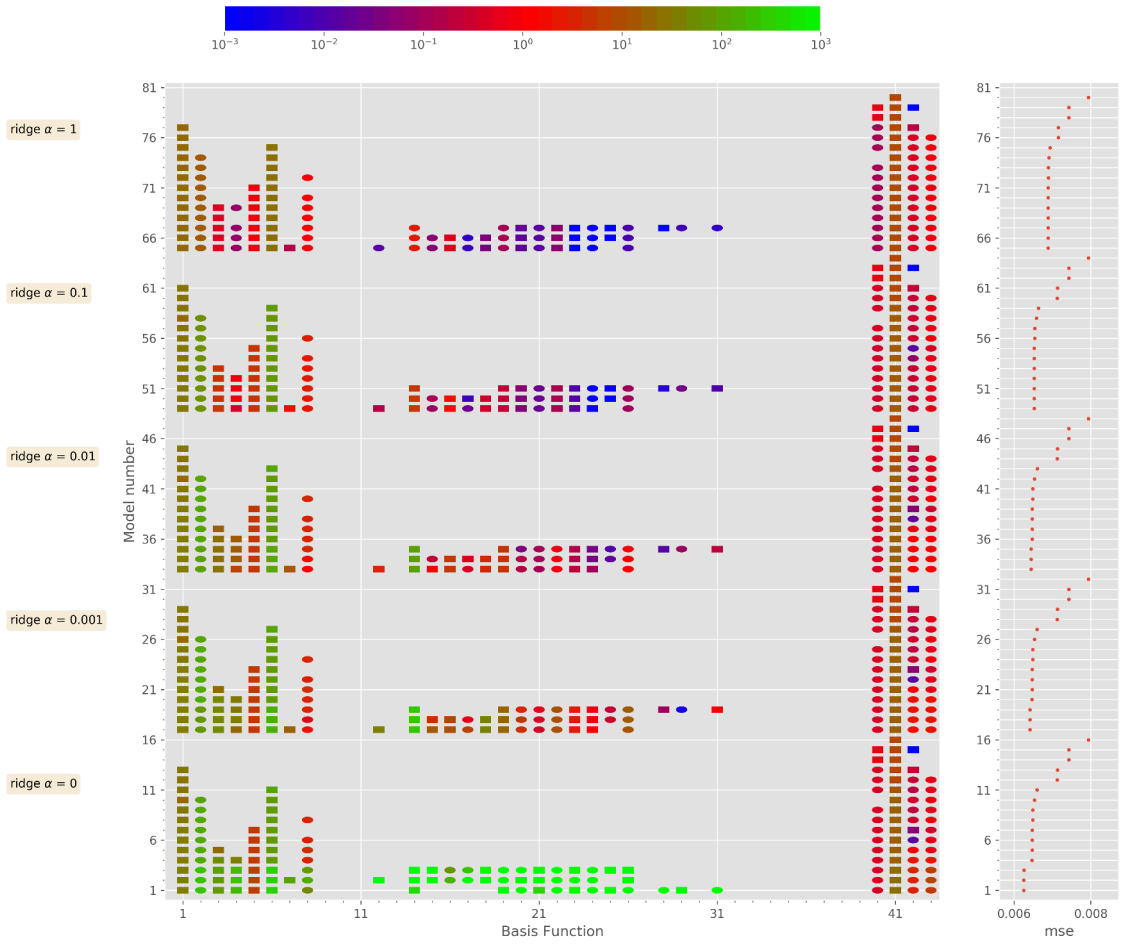


Figure 4.9: Comparing the model form and coefficient sizes for the discovered models for the b_{ij}^{Δ} correction term. Rectangles indicate positive coefficients, ovals indicate negative coefficients.

should not be too large for stability reasons and therefore the models that have green model coefficients are not an option because they are too large. Models with too many coefficients are computationally more expensive and has higher order terms that form terms that form local instabilities.

The models that have been picked for the sections following is shown are shown in (4.1) model 36 in Figure 4.9 and (4.2) model 70 in Figure 4.10. To investigate how well these models approximate the models have been applied to the training flow conditions and the corrective fields are plotted in Figure 4.11 and 4.12. The first figure shows the results just above the ground plane at $z = 0.1$ and shows that the correction model only really predicts very well in the regions where the strain rate in the flow is large, in the areas where the velocity gradients are the largest. In the other regions the model predictions is very poor indicating the incapability of the model to correctly predict the anisotropy correction.

In Figure 4.12 the same correction fields are shown but at the centre height of the cube $z = 0.5$. This image shows similar to the previous image that the corrections mainly focus around the cube and that the corrections are most accurate in the regions with a high strain rate. In some cases the predictions are actually opposite to the training data as can be seen in Figure 4.12e.

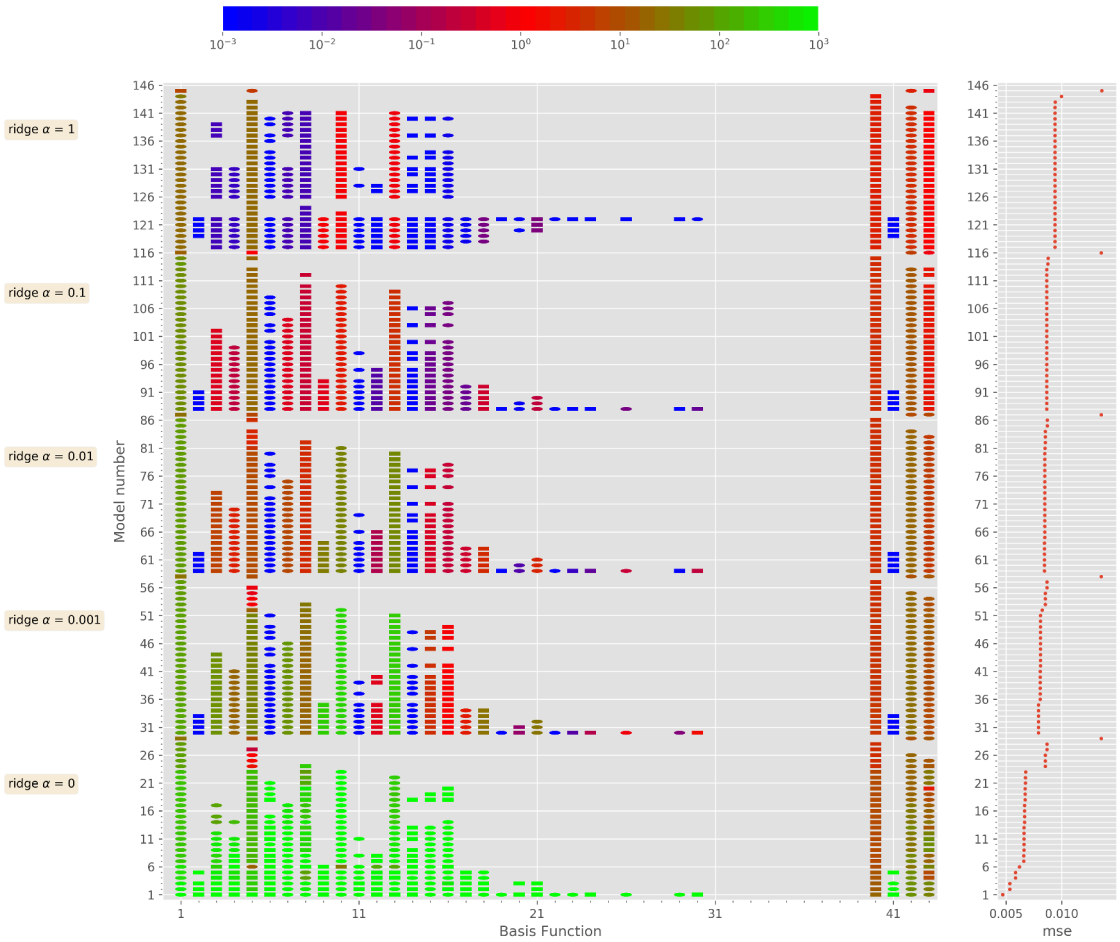
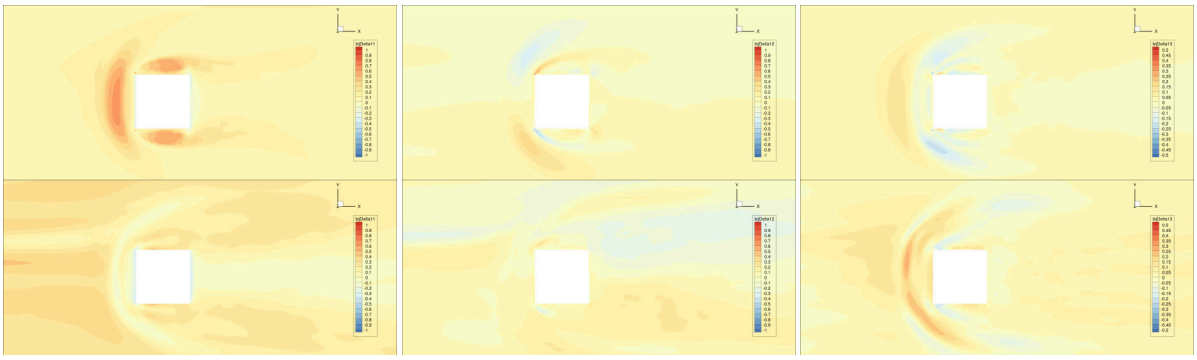
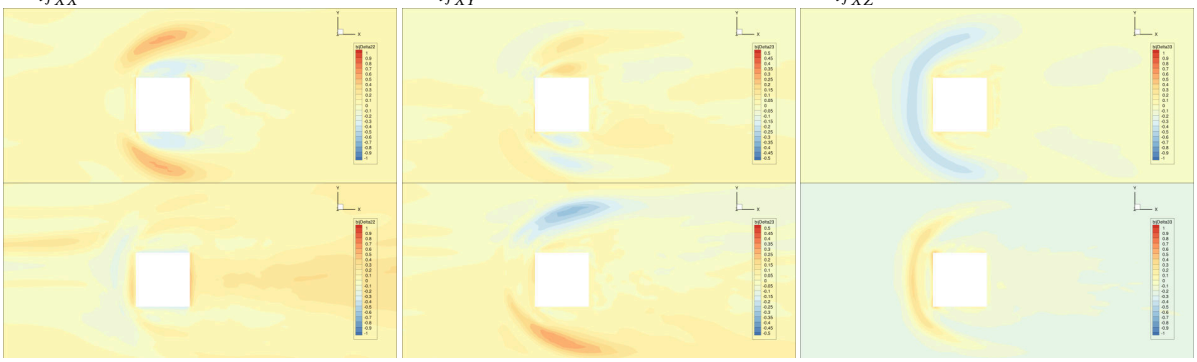


Figure 4.10: Comparing the model form and coefficient sizes for the discovered models for the R correction term. Rectangles indicate positive coefficients, ovals indicate negative coefficients.

In Figures 4.13a and 4.13b a similar comparison is made for the R correction fields at $z = 0.1$ and $z = 0.5$. This shows that the R correction models is mainly predicting the correction near the cube surfaces. Meanwhile the target field also has correction further away from the cube, for example in front of the cube the target value is mainly negative, while the model does not predict a correction there. Overall the biggest correction regions are captured, however the correction values show less contrast compared to the targets with less extreme values.

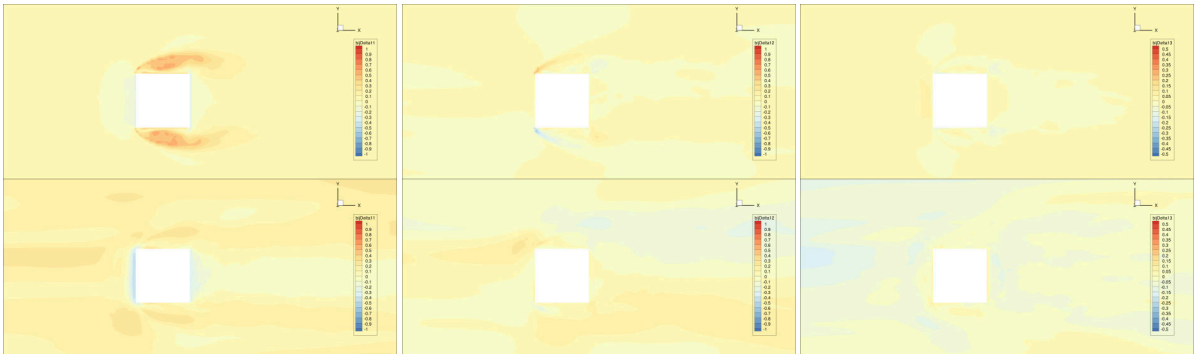


(a) $b^{\Delta}_{ij\,XX}$ (b) $b^{\Delta}_{ij\,XY}$ (c) $b^{\Delta}_{ij\,XZ}$

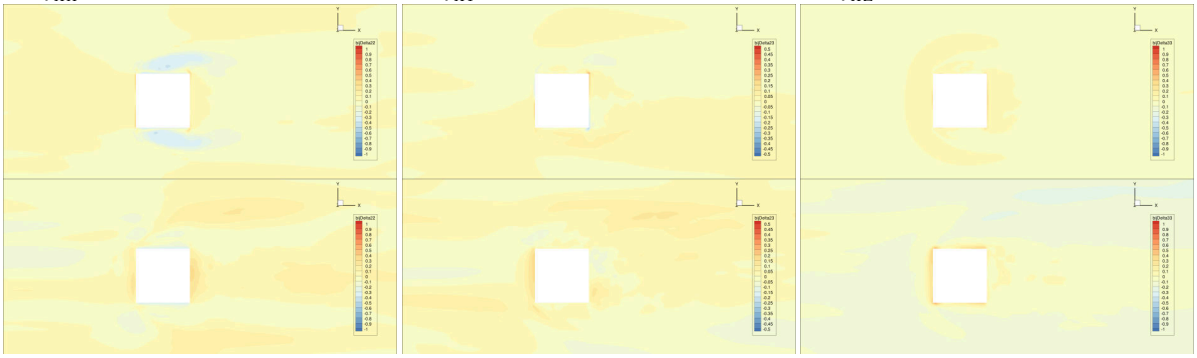


(d) $b^{\Delta}_{ij\,YY}$ (e) $b^{\Delta}_{ij\,YZ}$ (f) $b^{\Delta}_{ij\,ZZ}$

Figure 4.11: Comparison between the training data and the correction model prediction for the anisotropy correction b^{Δ}_{ij} at height $z = 0.1$. Top images are the model approximations of the frozen approach target values shown in the bottom images.

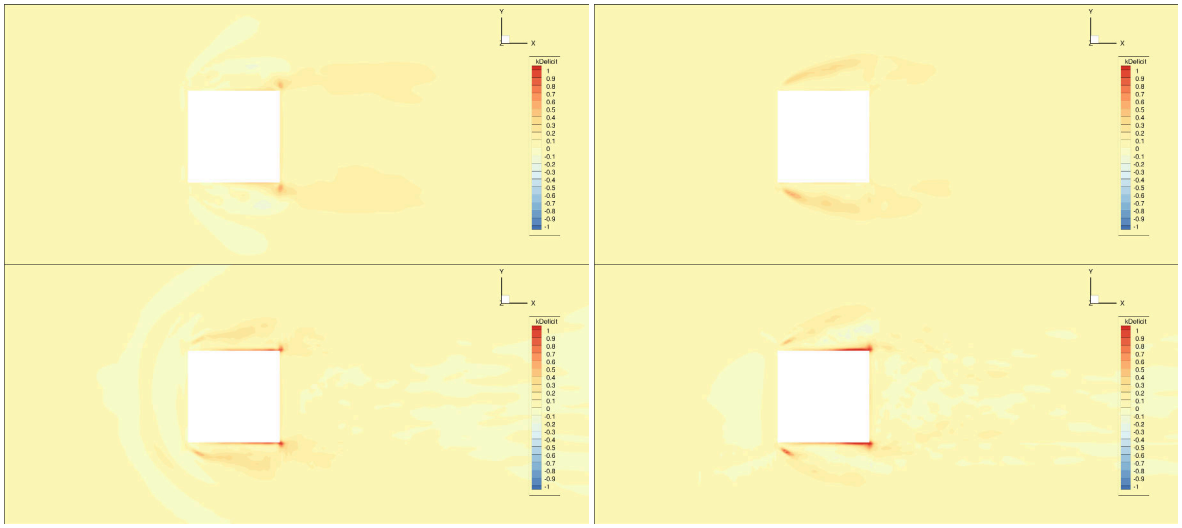


(a) $b^{\Delta}_{ij\,XX}$ (b) $b^{\Delta}_{ij\,XY}$ (c) $b^{\Delta}_{ij\,XZ}$



(d) $b^{\Delta}_{ij\,YY}$ (e) $b^{\Delta}_{ij\,YZ}$ (f) $b^{\Delta}_{ij\,ZZ}$

Figure 4.12: Comparison between the training data and the correction model prediction for the anisotropy correction b^{Δ}_{ij} at height $z = 0.5$. Top images are the model approximations of the frozen approach target values shown in the bottom images.



(a) $z = 0.1$. (b) $z = 0.5$.

Figure 4.13: Comparing the model prediction for R from the correction model to the target value. Top: model prediction, Bottom: target correction field. Cube top view at different heights.

4.1.6. Model corrected RANS

Once one of the correction models has been picked it needs to be implemented in the openFOAM solver. How this is done is described in Section 3.3.3. For the cube the model that has been implemented consists of the two equations shown in (4.1) and (4.2).

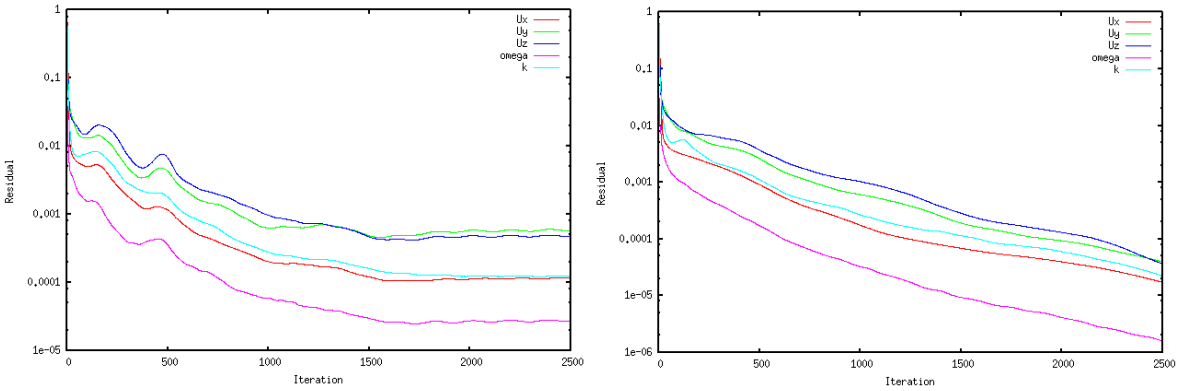
$$\begin{aligned} b_{ij}^{\Delta} = T^{(1)}(28.68361 I_1 + 4.71739 I_2 - 0.35604) \\ + T^{(2)}(-88.99302 I_1 + 68.76692 I_2 + 13.80382) \\ + T^{(3)}(20.59135 I_1 - 0.85935) \\ + T^{(4)}(11.45876 I_1 - 2.76972 I_2 - 0.94415) \end{aligned} \tag{4.1}$$

$$\begin{aligned} R = T^{(1)}(-35.74086 I_1 I_2 - 69.5826 I_1 + 39.73533 {I_2}^2 + 7.57252 I_2 + 3.73854) \\ + T^{(3)}(5.86701 I_1 + 0.3755 {I_2}^2 - 6.78427 I_2 - 16.47849) \\ + T^{(4)}(-2.30422 I_1 - 0.21816 {I_2}^2 + 3.13597 I_2 - 4.82169) \end{aligned} \tag{4.2}$$

The application of this correction model has been done in two different ways. The first is to start from the baseline solution and slowly introduce the corrective equations from here. The other is to start solving directly with these corrective equations applied. It turned out that the latter approach is more effective. Since the baseline solution is so far off from the DES solution it takes more iterations to come from the baseline RANS solution towards a converged corrected solution than to start from 0 and have the corrective fields applied from the start. It turned out as can be seen in Figure 4.14 that the latter approach actually converges faster than the baseline simulation.

An explanation for this can be found by looking at the final converged solutions from the modelled approach and the baseline RANS Figures 4.15. What can be seen is that the correction model generally adds turbulent kinetic energy to the flow and therefore adds a lot of stability. This leads to a convergence rate being higher and continuing lower than using the baseline $k-\omega$-SST model especially since the number of iterations required for each solver iteration of the velocity was reduced from requiring 3 inner loop iterations for the velocity field to just one iteration. This means that the added stability from the correction model actually speeds up the solving process despite being slightly more computationally expensive with the additional terms that need to be calculated.

The results that are shown in the following section are achieved by adding the full correction for the R correction and just 50% of the b_{ij}^{Δ}-correction as a higher number showed stability issues. A more thorough research into how much correction can be added and how this should be done can be found in Section 4.1.8.



(a) Converge history of the baseline RANS simulation

(b) Convergence history of the corrected RANS simulation

Figure 4.14: Comparing the convergence of the baseline and the corrected RANS simulations for the wall mounted cube case

When looking at the results the most obvious thing that stands out is the difference in the size of the wake. This can be very clearly seen when looking at a slice through the centre of the cube in Figures 4.15 and 4.16 and when looking at the streamlines on the bottom surface in Figure 4.17. The most likely explanation for the reduction of the wake is in the additional production from the corrective terms. This has caused the flow that is separated on the top and the sides of the cube to re-attache again. This can be seen in the flow on top of the cube that now has a smaller recirculation area (Figure A.1).

Another thing that has been changed due to the corrective terms in the model is the location of the horseshoe vortex in front and around the sides of the cube. The location is closer to the cube all around as can be clearly seen in Figure 4.17. As can be seen the vortex starting point is moved closer and the trajectory around the cube is also moved closer to the cube, more in line with the DES result but what looks like a bit too close and lacking the outboard motion after the cube. In Figure A.2 the starting location of the vortex is shown in more detail. In this figure it is clear that the location has moved forward to the cube, however it looks like it is moved to far and that the strength has increased too much in comparison to the DES results.

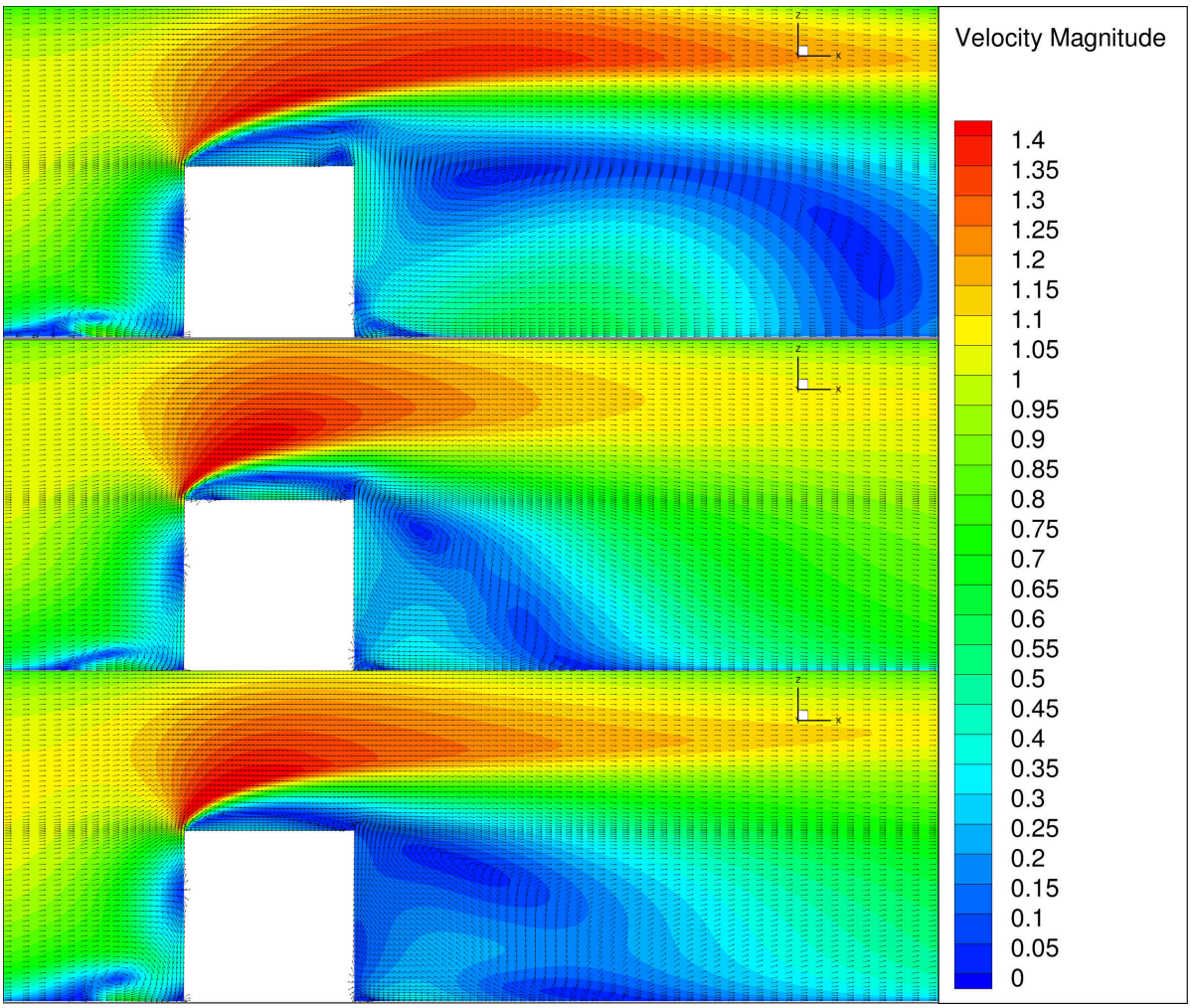


Figure 4.15: Comparison of the velocity magnitude. Top: Baseline RANS, Centre: DES, Bottom: Corrected $100\%R\,50\%b_{ij}^{\Delta}$

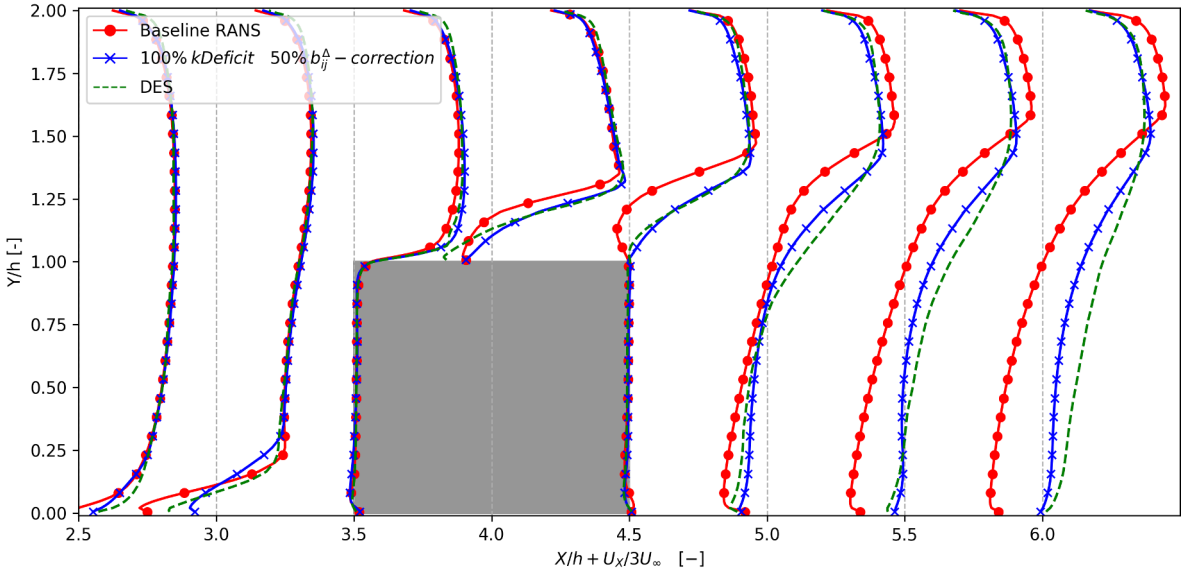


Figure 4.16: Direct comparison of the x-velocity component in the centre plane of the cube $y = 4.5$

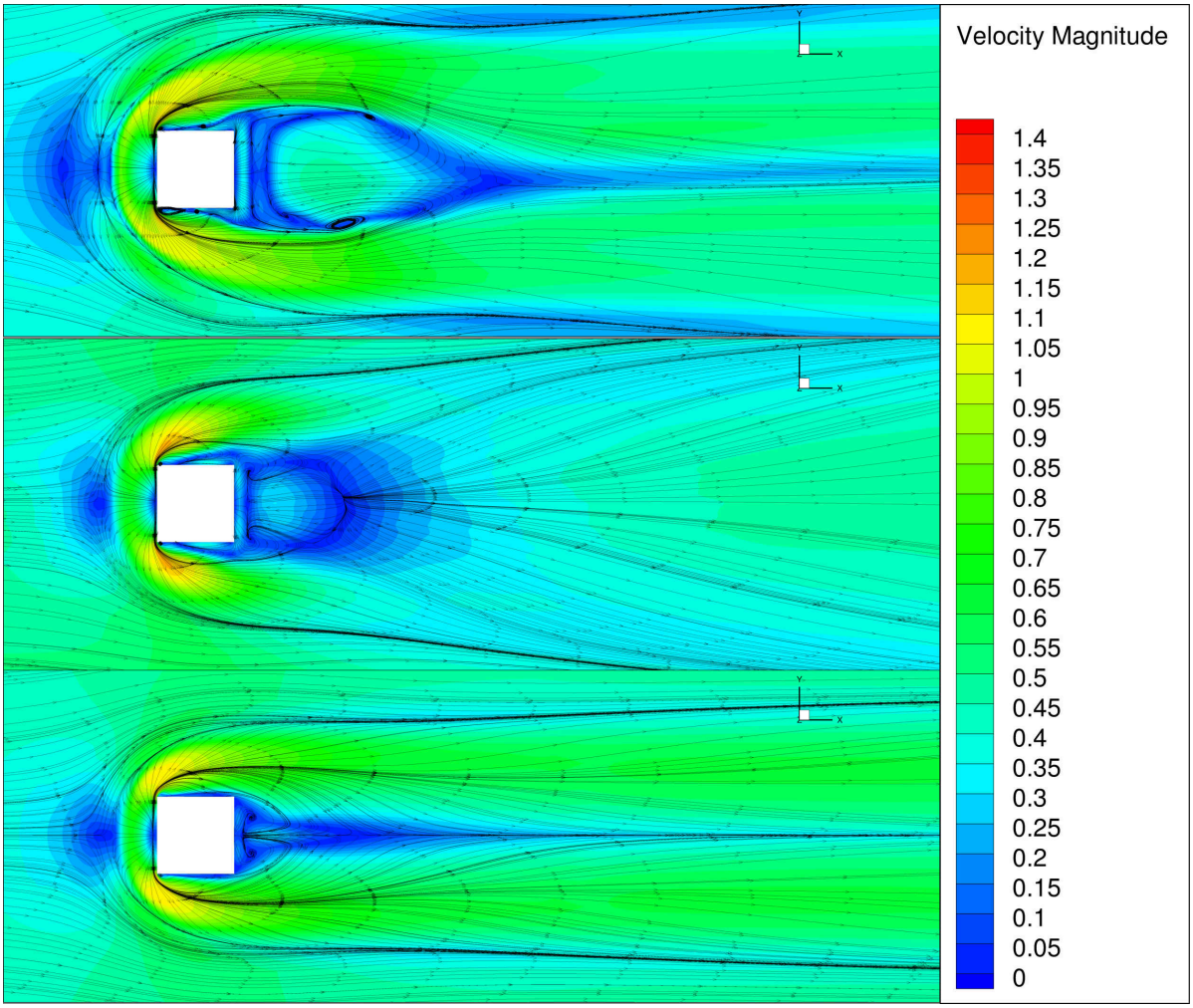


Figure 4.17: Top: Baseline RANS, Middle: DES, Bottom: Corrected 100%R 50%b_{ij}^{Δ}. Velocity magnitude and streamlines at the bottom of the domain.

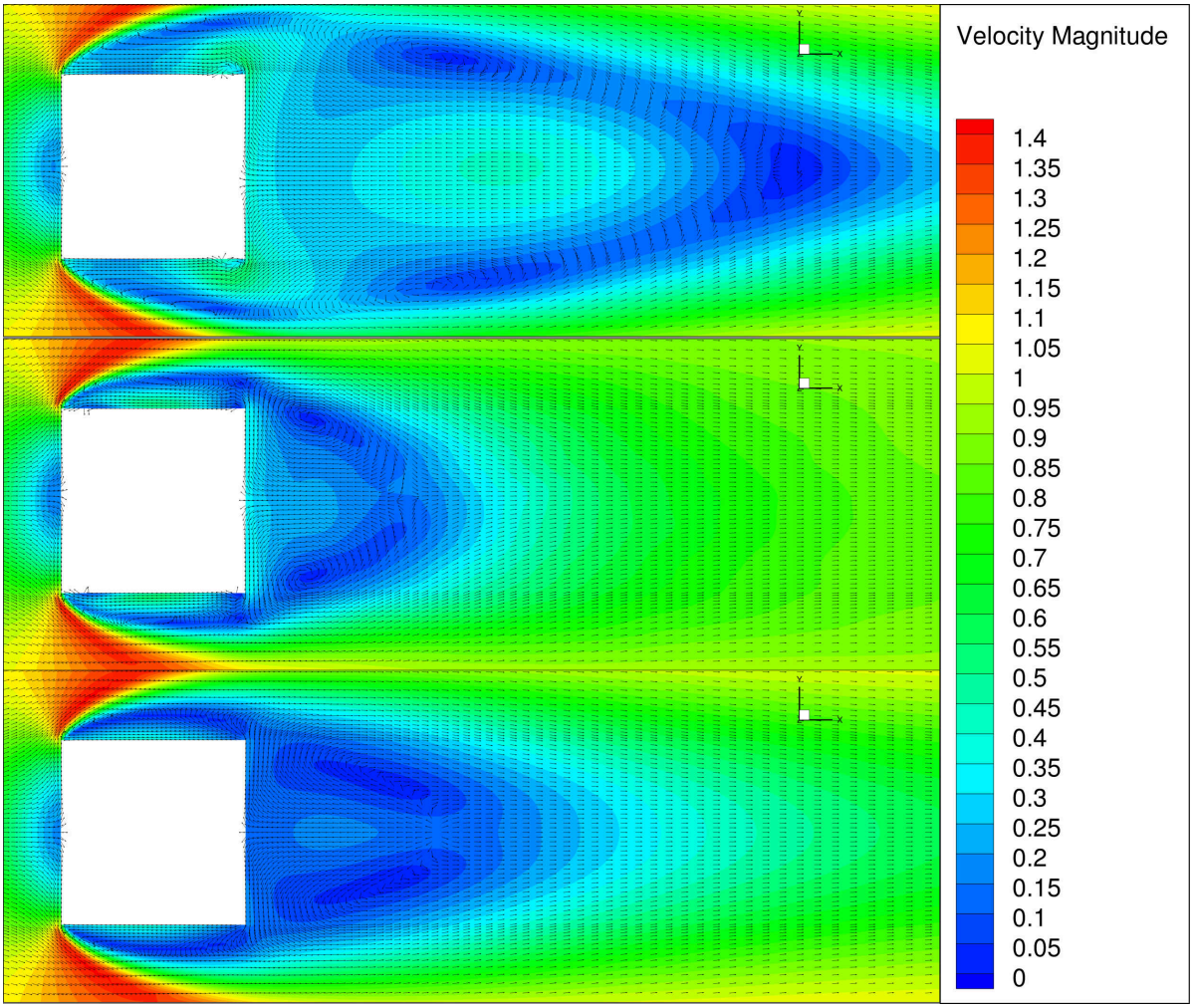


Figure 4.18: Top: Baseline RANS, Middle: DES, Bottom: Corrected $100\%R\,50\%b_{ij}^{\Delta}$, velocity magnitude and vectors at the middle height of the cube $z = 0.5$

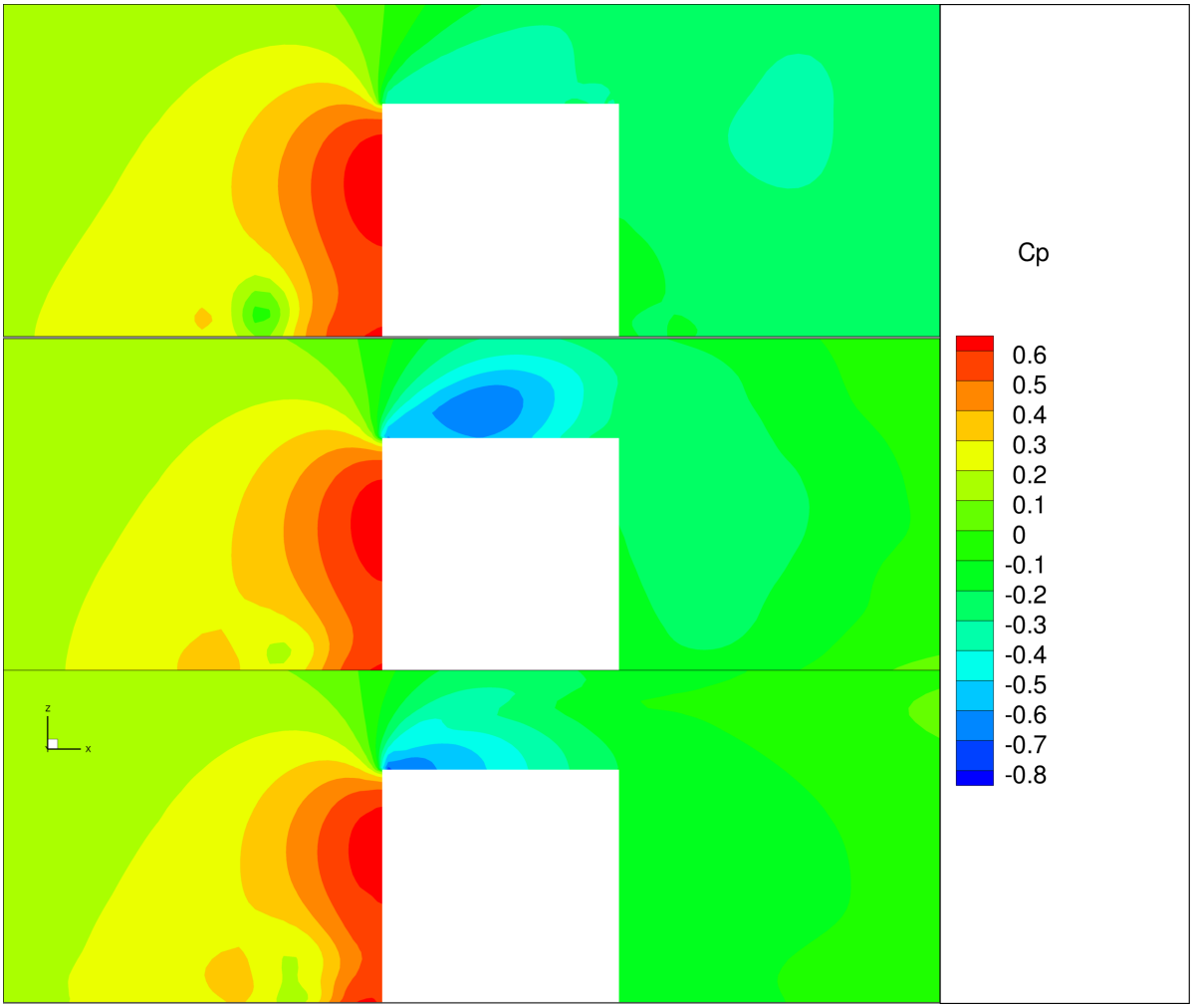


Figure 4.19: Top: Baseline RANS, Middle: DES, Bottom: Corrected 100%R 50%b_{ij}^{Δ}, pressure coefficient at the centre of the cube.

4.1.7. Model regression with 10 tensors and 5 invariants

One of the things that has been investigated is the effect of limiting the number of input Tensors to 4 and the number of invariants to 2. To do this a model regression has been done using all 10 tensors and 5 invariants as basis functions.

In Figure 4.20 and Figure 4.21 the results for the correction models regression can be found. What can be seen in this figure is that from the extended library of complex basis functions mainly the simplest terms are seen back in the models. The list of basis functions can be found in Table 4.3. As expected the accuracy of the model reduces with number of basis functions used and size of coefficients. The models that are found using this list of basis functions generally have a smaller number of basis functions compared to the models found using only the first 4 tensors and 2 invariants. This means that the most complex model has a sufficiently small number of variables to not be too computationally expensive. In order to have coefficients of sufficiently small size a ridge α value of 0.1 was required. The model that was selected is model 55 from Figure 4.20 and is shown in (4.3).

$$\begin{aligned} b_{ij}^{\Delta} = T^{(1)}(18.23125I_1 - 1.35478I_3 - 0.08781) + 7.56892T^{(2)} \\ + T^{(3)}(0.37684I_1 - 0.37384) + T4(-0.06152I_1 - 0.89674I_2 - 0.72209) \\ -2.62734T^{(5)} + 1.96763T^{(6)} - 13.00702T^{(7)} - 11.9126T^{(8)}) \end{aligned} \tag{4.3}$$

The model results for R are shown in Figure 4.21. It can be seen that these models are generally more complex and also contain more variations of the invarariants, where in the model for b_{ij}^{Δ} the model tensors were mainly multiplied with constants. The model that was picked from this list is model 175 and is shown in (4.4).

If one takes a good look at Table 4.3, where the basis functions used are shown one can see that there are a few of duplicates. This is due to a bug in the cliquing part of the code. he cliquing works by grouping function according to similarity, after which from each group the simplest function is selected. With this large set of basis functions it has occurred that some of the simplest basis functions show up in multiple groups and are therefore selected twice. This should not matter for the end result as identical functions are balanced by each other and can simply be summed up to get the final coefficient. It is however computationally inefficient and therefore in future an additional check will be performed to remove these double entries in the basis functions list. This bug only appears when the full set of 10 tensors and invariants is used, with just the first 4 tensors and 2 invariants this did not happen.

$$\begin{aligned} R = T^{(1)}(-70.72229I_1 + 36.89125{I_2}^2 + 6.3582I_3 - 3.02241I_4 + 3.74826) \\ + T^{(3)}(5.83195I_1 + 0.39076{I_2}^2 - 0.57387I_3 - 16.71976) \\ + T^{(4)}(-2.32321I_1 - 0.21646{I_2}^2 + 3.18368I_2 + 0.17686I_3 - 5.13732) \\ + T^{(5)}(-17.59335I_1 + 0.76685I_5 - 1.78069I2^2 + 18.73711I_2 + 2.99085) \\ + T^{(6)}(-35.18669I_1 + 5.98169) + T^{(9)}(0.1964I_2 + 0.00361I_5 - 7.16874) \\ + T^{(10)}(-3.0e-5{I_1}^2 - 0.00167I_1 + 2.0e-5I_5) \end{aligned} \tag{4.4}$$

Taking a look at the Figures 4.22, 4.23, 4.24 and A.4 a direct comparison is made between the corrective fields described by equations (4.1), (4.2) and (4.3), (4.4). Both these models have a mean squared error when fitted to the training data that is around the same order of magnitude, both models also have coefficients that are of the same order of magnitude and a similar number of model terms to get a equivalent level of complexity.

Looking at the first Figure 4.22 shows a slice through the centre of the centre of the cube in the y plane. The velocity magnitude is plotted and the velocity direction is shown through vectors. There are some small differences between the top and bottom images. The shape of the wake seems to be slightly different and the recirculation area on the top of the cube seems to be larger using the full model in the bottom image. This can be seen in more detail in Figure A.3. This shows the correction model with 10 tensors and 4 invariants to give a slightly better prediction over the model with just 4 tensors and 2 invariants.

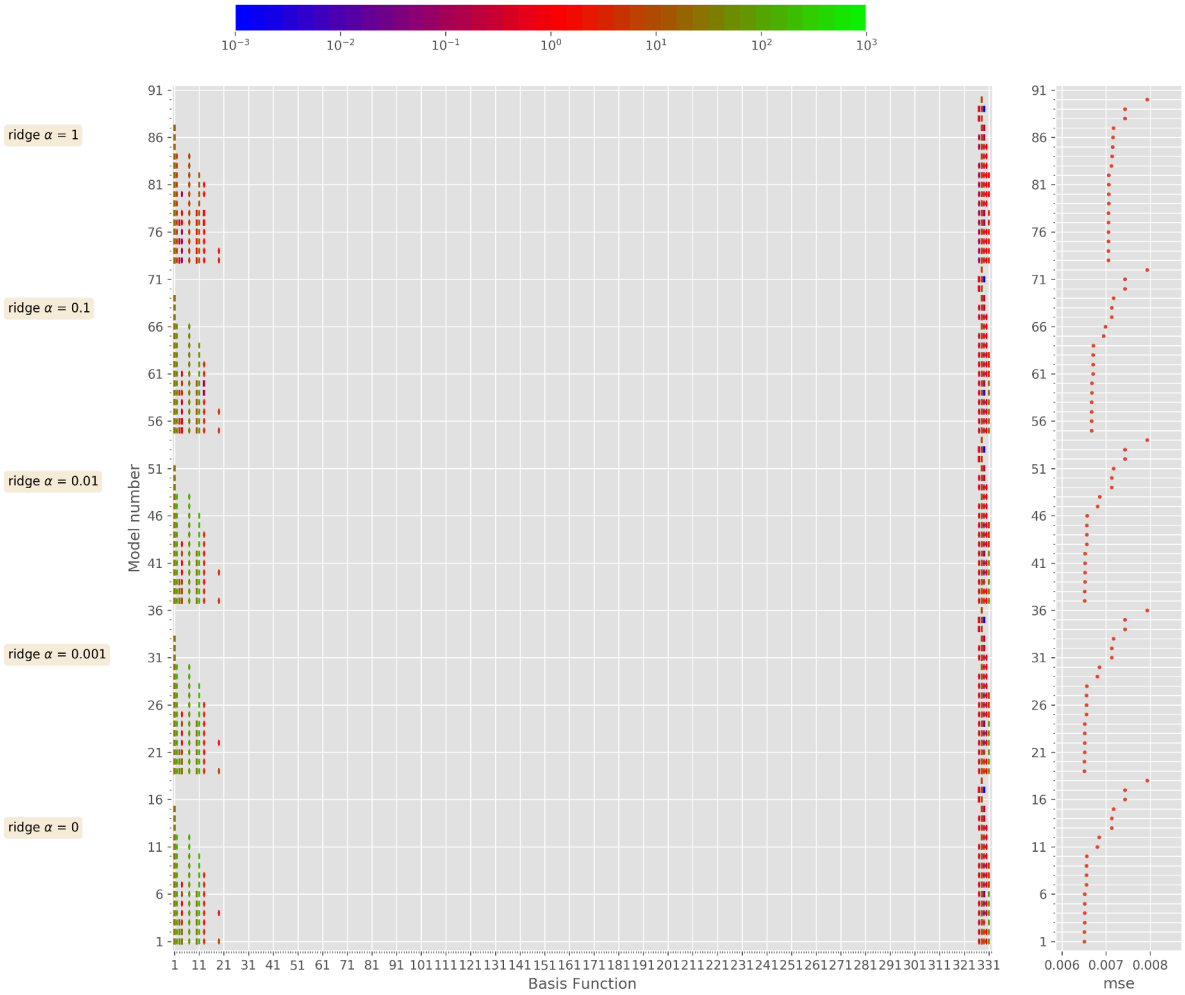


Figure 4.20: Visualising the different models found for b^{Δ}_{ij} from regressing a function with 10 Tensors and 5 Invariants.

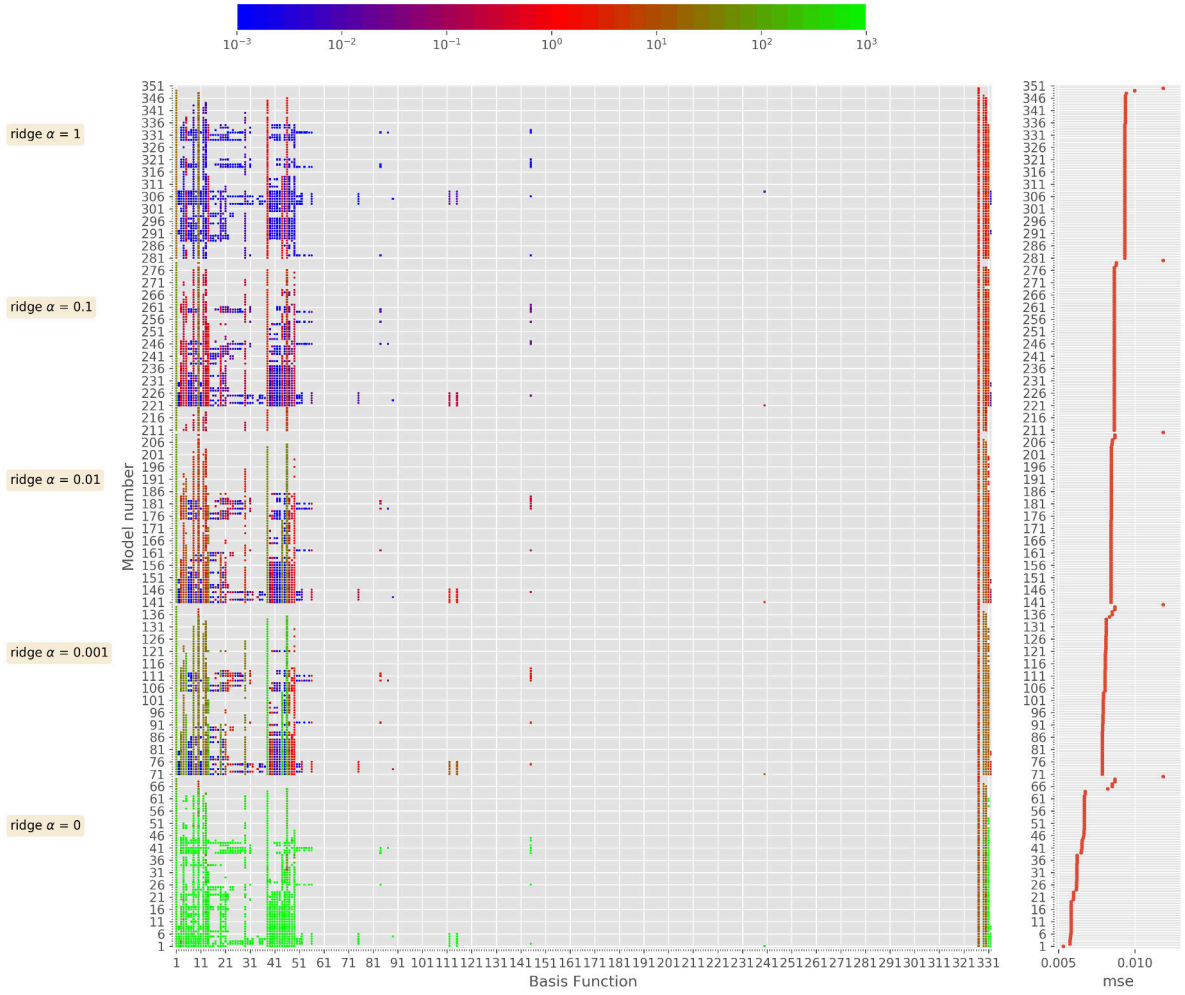


Figure 4.21: Visualising the different models found for R from regressing a function with 10 Tensors and 5 Invariants.

1	$I_1 T^{(1)}$	57	$I_2 I_3 T^{(3)}$	113	$I_5 T^{(8)}$	169	$I_2 I_3^2 T^{(9)}$	225	$I_2 I_4^2 T^{(4)}$	281	$I_2 I_4 I_5 T^{(3)}$
2	$c T^{(8)}$	58	$I_2 I_3 T^{(4)}$	114	$I_1 I_5 T^{(10)}$	170	$I_3^2 I_4 T^{(7)}$	226	$I_2 I_4^2 T^{(5)}$	282	$I_2 I_4 I_5 T^{(4)}$
3	$I_1 T^{(3)}$	59	$I_2 I_3 T^{(5)}$	115	$I_2 I_5 T^{(1)}$	171	$I_1^2 I_4 T^{(2)}$	227	$I_2 I_4^2 T^{(6)}$	283	$I_2 I_4 I_5 T^{(5)}$
4	$I_1 T^{(4)}$	60	$I_2 I_3 T^{(6)}$	116	$I_2 I_5 T^{(4)}$	172	$I_1^2 I_4 T^{(4)}$	228	$I_2 I_4^2 T^{(8)}$	284	$I_2 I_4 I_5 T^{(8)}$
5	$I_1 T^{(5)}$	61	$I_2 I_3 T^{(7)}$	117	$I_2 I_5 T^{(5)}$	173	$I_1^2 I_4 T^{(5)}$	229	$I_2 I_4^2 T^{(9)}$	285	$I_2 I_4 I_5 T^{(9)}$
6	$I_1 T^{(7)}$	62	$I_2 I_3 T^{(8)}$	118	$I_2 I_5 T^{(7)}$	174	$I_1^2 I_4 T^{(6)}$	230	$I_2 I_4^2 T^{(10)}$	286	$I_2 I_4^2 T^{(7)}$
7	$c T^{(8)}$	63	$I_2 I_3 T^{(9)}$	119	$I_2 I_5 T^{(8)}$	175	$I_1^2 I_4 T^{(7)}$	231	$I_3 I_4^2 T^{(1)}$	287	$I_3 I_4 I_5 T^{(1)}$
8	$c T^{(9)}$	64	$I_3 I_4 T^{(7)}$	120	$I_2 I_5 T^{(9)}$	176	$I_1 I_4 T^{(8)}$	232	$I_3 I_4^2 T^{(2)}$	288	$I_3 I_4 I_5 T^{(2)}$
9	$I_1 T^{(10)}$	65	$I_3^2 T^{(1)}$	121	$I_4 I_5 T^{(7)}$	177	$I_1^2 I_4 T^{(9)}$	233	$I_3 I_4^2 T^{(3)}$	289	$I_3 I_4 I_5 T^{(3)}$
10	$c T^{(6)}$	66	$I_3^2 T^{(2)}$	122	$I_3 I_5 T^{(1)}$	178	$I_1^2 I_4 T^{(10)}$	234	$I_3 I_4^2 T^{(4)}$	290	$I_3 I_4 I_5 T^{(4)}$
11	$c T^{(7)}$	67	$I_3^2 T^{(3)}$	123	$I_3 I_5 T^{(3)}$	179	$I_4 I_5 T^{(7)}$	235	$I_3 I_4^2 T^{(5)}$	291	$I_3 I_4 I_5 T^{(5)}$
12	$c T^{(9)}$	68	$I_3^2 T^{(4)}$	124	$I_3 I_5 T^{(4)}$	180	$I_2^2 I_4 T^{(1)}$	236	$I_3 I_4^2 T^{(6)}$	292	$I_3 I_4 I_5 T^{(6)}$
13	$I_2 T^{(4)}$	69	$I_3^2 T^{(5)}$	125	$I_3 I_5 T^{(5)}$	181	$I_2^2 I_4 T^{(3)}$	237	$I_3 I_4^2 T^{(8)}$	293	$I_3 I_4 I_5 T^{(8)}$
14	$I_2 T^{(5)}$	70	$I_3^2 T^{(6)}$	126	$I_3 I_5 T^{(6)}$	182	$I_2^2 I_4 T^{(4)}$	238	$I_3 I_4^2 T^{(9)}$	294	$I_3 I_4 I_5 T^{(9)}$
15	$I_2 T^{(7)}$	71	$I_3^2 T^{(7)}$	127	$I_3 I_5 T^{(7)}$	183	$I_2^2 I_4 T^{(5)}$	239	$I_3 I_4^2 T^{(10)}$	295	$I_3 I_4 I_5 T^{(10)}$
16	$I_2 T^{(8)}$	72	$I_3^2 T^{(8)}$	128	$I_3 I_5 T^{(9)}$	184	$I_2^2 I_4 T^{(7)}$	240	$I_1 I_5 T^{(1)}$	296	$I_4^2 I_5 T^{(1)}$
17	$I_2 T^{(9)}$	73	$I_3^2 T^{(9)}$	129	$I_3 I_5 T^{(10)}$	185	$I_2^2 I_4 T^{(8)}$	241	$I_5 T^{(3)}$	297	$I_4^2 I_5 T^{(2)}$
18	$I_4 T^{(7)}$	74	$I_3^2 T^{(10)}$	130	$I_4 I_5 T^{(1)}$	186	$I_2^2 I_4 T^{(9)}$	242	$I_1 I_5 T^{(7)}$	298	$I_4^2 I_5 T^{(3)}$
19	$I_3 T^{(1)}$	75	$I_1 I_4 T^{(1)}$	131	$I_4 I_5 T^{(2)}$	187	$I_2 I_4^2 T^{(7)}$	243	$I_1 I_5 T^{(9)}$	299	$I_4^2 I_5 T^{(4)}$
20	$I_3 T^{(2)}$	76	$I_1 I_4 T^{(2)}$	132	$I_4 I_5 T^{(3)}$	188	$I_1 I_3 I_4 T^{(2)}$	244	$I_1^2 I_5 T^{(10)}$	300	$I_4^2 I_5 T^{(5)}$
21	$I_3 T^{(3)}$	77	$I_1 I_4 T^{(3)}$	133	$I_4 I_5 T^{(4)}$	189	$I_1 I_3 I_4 T^{(4)}$	245	$I_2^2 I_5 T^{(1)}$	301	$I_4^2 I_5 T^{(6)}$
22	$I_3 T^{(4)}$	78	$I_1 I_4 T^{(4)}$	134	$I_4 I_5 T^{(5)}$	190	$I_1 I_3 I_4 T^{(5)}$	246	$I_2^2 I_5 T^{(4)}$	302	$I_4^2 I_5 T^{(8)}$
23	$I_3 T^{(5)}$	79	$I_1 I_4 T^{(5)}$	135	$I_4 I_5 T^{(6)}$	191	$I_1 I_3 I_4 T^{(6)}$	247	$I_2^2 I_5 T^{(5)}$	303	$I_4^2 I_5 T^{(9)}$
24	$I_3 T^{(6)}$	80	$I_1 I_4 T^{(6)}$	136	$I_4 I_5 T^{(8)}$	192	$I_1 I_3 I_4 T^{(7)}$	248	$I_2^2 I_5 T^{(9)}$	304	$I_4^2 I_5 T^{(10)}$
25	$I_3 T^{(7)}$	81	$I_1 I_4 T^{(7)}$	137	$I_4 I_5 T^{(9)}$	193	$I_1 I_3 I_4 T^{(9)}$	249	$I_2 I_4 I_5 T^{(7)}$	305	$I_5^2 T^{(1)}$
26	$I_3 T^{(8)}$	82	$I_1 I_4 T^{(9)}$	138	$I_4 I_5 T^{(10)}$	194	$I_1 I_3 I_4 T^{(10)}$	250	$I_3 I_5 T^{(2)}$	306	$I_5^2 T^{(8)}$
27	$I_3 T^{(9)}$	83	$I_1 I_4 T^{(10)}$	139	$I_5^2 T^{(4)}$	195	$I_2 I_3 I_4 T^{(1)}$	251	$I_1 I_3 I_5 T^{(4)}$	307	$I_1 I_5^2 T^{(10)}$
28	$I_3 T^{(10)}$	84	$I_2 I_4 T^{(1)}$	140	$I_5^2 T^{(5)}$	196	$I_3 I_4 T^{(7)}$	252	$I_1 I_3 I_5 T^{(7)}$	308	$I_2 I_5^2 T^{(4)}$
29	$I_4 T^{(1)}$	85	$I_4 T^{(7)}$	141	$I_5^2 T^{(7)}$	197	$I_2 I_3 I_4 T^{(3)}$	253	$I_3 I_5 T^{(8)}$	309	$I_2 I_5^2 T^{(5)}$
30	$I_4 T^{(2)}$	86	$I_2 I_4 T^{(3)}$	142	$I_5^2 T^{(9)}$	198	$I_2 I_3 I_4 T^{(4)}$	254	$I_1 I_3 I_5 T^{(10)}$	310	$I_2^2 I_5 T^{(8)}$
31	$I_4 T^{(3)}$	87	$I_2 I_4 T^{(4)}$	143	$I_5^2 T^{(10)}$	199	$I_2 I_3 I_4 T^{(5)}$	255	$I_2 I_3 I_5 T^{(1)}$	311	$I_2 I_5^2 T^{(9)}$
32	$I_4 T^{(4)}$	88	$I_2 I_4 T^{(5)}$	144	$I_2 I_5 T^{(3)}$	200	$I_2 I_3 I_4 T^{(6)}$	256	$I_2 I_3 I_5 T^{(3)}$	312	$I_4 I_5^2 T^{(7)}$
33	$I_4 T^{(5)}$	89	$I_2 I_4 T^{(6)}$	145	$I_1 I_3 T^{(1)}$	201	$I_2 I_3 I_4 T^{(8)}$	257	$I_2 I_3 I_5 T^{(4)}$	313	$I_3 I_5^2 T^{(1)}$
34	$I_4 T^{(6)}$	90	$I_2 I_4 T^{(8)}$	146	$I_1 I_3 T^{(2)}$	202	$I_2 I_3 I_4 T^{(9)}$	258	$I_2 I_3 I_5 T^{(5)}$	314	$I_3 I_5^2 T^{(4)}$
35	$I_4 T^{(8)}$	91	$I_2 I_4 T^{(9)}$	147	$I_1 I_3 T^{(3)}$	203	$I_3 I_4^2 T^{(7)}$	259	$I_2 I_3 I_5 T^{(6)}$	315	$I_3 I_5^2 T^{(5)}$
36	$I_4 T^{(9)}$	92	$I_4^2 T^{(7)}$	148	$I_1 I_3 T^{(5)}$	204	$I_3^2 I_4 T^{(1)}$	260	$I_2 I_3 I_5 T^{(7)}$	316	$I_3 I_5^2 T^{(6)}$
37	$I_4 T^{(10)}$	93	$I_3 I_4 T^{(1)}$	149	$I_1 I_3 T^{(6)}$	205	$I_3^2 I_4 T^{(2)}$	261	$I_2 I_3 I_5 T^{(8)}$	317	$I_3 I_5^2 T^{(7)}$
38	$I_1 T^{(6)}$	94	$I_3 I_4 T^{(2)}$	150	$I_1 I_3 T^{(8)}$	206	$I_3^2 I_4 T^{(3)}$	262	$I_2 I_3 I_5 T^{(9)}$	318	$I_3 I_5^2 T^{(9)}$
39	$I_5 T^{(4)}$	95	$I_3 I_4 T^{(3)}$	151	$I_1 I_3 T^{(9)}$	207	$I_3^2 I_4 T^{(4)}$	263	$I_3 I_4 I_5 T^{(7)}$	319	$I_3 I_5^2 T^{(10)}$
40	$I_5 T^{(5)}$	96	$I_3 I_4 T^{(4)}$	152	$I_1 I_3 T^{(10)}$	208	$I_3^2 I_4 T^{(5)}$	264	$I_1 I_3^2 T^{(7)}$	320	$I_4 I_5^2 T^{(1)}$
41	$I_5 T^{(7)}$	97	$I_3 I_4 T^{(5)}$	153	$I_2^2 I_3 T^{(1)}$	209	$I_3^2 I_4 T^{(6)}$	265	$I_3^2 I_5 T^{(4)}$	321	$I_4 I_5^2 T^{(3)}$
42	$I_5 T^{(9)}$	98	$I_3 I_4 T^{(6)}$	154	$I_2^2 I_3 T^{(3)}$	210	$I_3^2 I_4 T^{(8)}$	266	$I_3^2 I_5 T^{(5)}$	322	$I_4 I_5^2 T^{(4)}$
43	$I_5 T^{(10)}$	99	$I_3 I_4 T^{(8)}$	155	$I_2^2 I_3 T^{(4)}$	211	$I_3^2 I_4 T^{(9)}$	267	$I_3^2 I_5 T^{(6)}$	323	$I_4 I_5^2 T^{(5)}$
44	$I_1^2 T^{(1)}$	100	$I_3 I_4 T^{(9)}$	156	$I_2^2 I_3 T^{(5)}$	212	$I_3^2 I_4 T^{(10)}$	268	$I_3^2 I_5 T^{(7)}$	324	$I_4 I_5^2 T^{(8)}$
45	$I_1^2 T^{(10)}$	101	$I_3 I_4 T^{(10)}$	157	$I_2^2 I_3 T^{(6)}$	213	$I_1 I_4^2 T^{(1)}$	269	$I_3^2 I_5 T^{(10)}$	325	$I_4 I_5^2 T^{(9)}$
46	$I_2^2 T^{(1)}$	102	$I_4^2 T^{(1)}$	158	$I_2^2 I_3 T^{(7)}$	214	$I_1 I_4^2 T^{(2)}$	270	$I_1 I_4 I_5 T^{(1)}$	326	$I_4 I_5^2 T^{(10)}$
47	$I_2^2 T^{(3)}$	103	$I_4^2 T^{(2)}$	159	$I_2^2 I_3 T^{(8)}$	215	$I_1 I_4^2 T^{(3)}$	271	$I_1 I_4 I_5 T^{(2)}$	327	$c T^{(1)}$
48	$I_2^2 T^{(4)}$	104	$I_4^2 T^{(3)}$	160	$I_2^2 I_3 T^{(9)}$	216	$I_1 I_4^2 T^{(4)}$	272	$I_1 I_4 I_5 T^{(3)}$	328	$c T^{(2)}$
49	$I_2^2 T^{(5)}$	105	$I_4^2 T^{(4)}$	161	$I_2 I_3 I_4 T^{(7)}$	217	$I_1 I_4^2 T^{(5)}$	273	$I_1 I_4 I_5 T^{(4)}$	329	$c T^{(3)}$
50	$I_2^2 T^{(7)}$	106	$I_4^2 T^{(5)}$	162	$I_2 I_3^2 T^{(1)}$	218	$I_1 I_4^2 T^{(6)}$	274	$I_1 I_4 I_5 T^{(5)}$	330	$c T^{(4)}$
51	$I_2 I_5 T^{(2)}$	107	$I_4^2 T^{(6)}$	163	$I_2 I_3^2 T^{(3)}$	219	$I_1 I_4^2 T^{(7)}$	275	$I_1 I_4 I_5 T^{(6)}$	331	$c T^{(5)}$
52	$I_2^2 T^{(9)}$	108	$I_4^2 T^{(8)}$	164	$I_2 I_3^2 T^{(4)}$	220	$I_1 I_4^2 T^{(8)}$	276	$I_1 I_4 I_5 T^{(7)}$	332	$c T^{(10)}$
53	$I_2 I_4 T^{(7)}$	109	$I_4^2 T^{(9)}$	165	$I_2 I_3^2 T^{(5)}$	221	$I_1 I_4^2 T^{(9)}$	277	$I_1 I_4 I_5 T^{(8)}$		
54	$I_1 I_3 T^{(4)}$	110	$I_4^2 T^{(10)}$	166	$I_2 I_3^2 T^{(6)}$	222	$I_1 I_4^2 T^{(10)}$	278	$I_1 I_4 I_5 T^{(9)}$		
55	$I_1 I_3 T^{(7)}$	111	$I_1 I_5 T^{(4)}$	167	$I_2 I_3^2 T^{(7)}$	223	$I_2 I_4^2 T^{(1)}$	279	$I_1 I_4 I_5 T^{(10)}$		
56	$I_2 I_3 T^{(1)}$	112	$I_5 T^{(6)}$	168	$I_2 I_3^2 T^{(8)}$	224	$I_2 I_4^2 T^{(3)}$	280	$I_2 I_4 I_5 T^{(1)}$		

Table 4.3: List of basis functions used for the model regression with 10 tensors and 5 invariants.

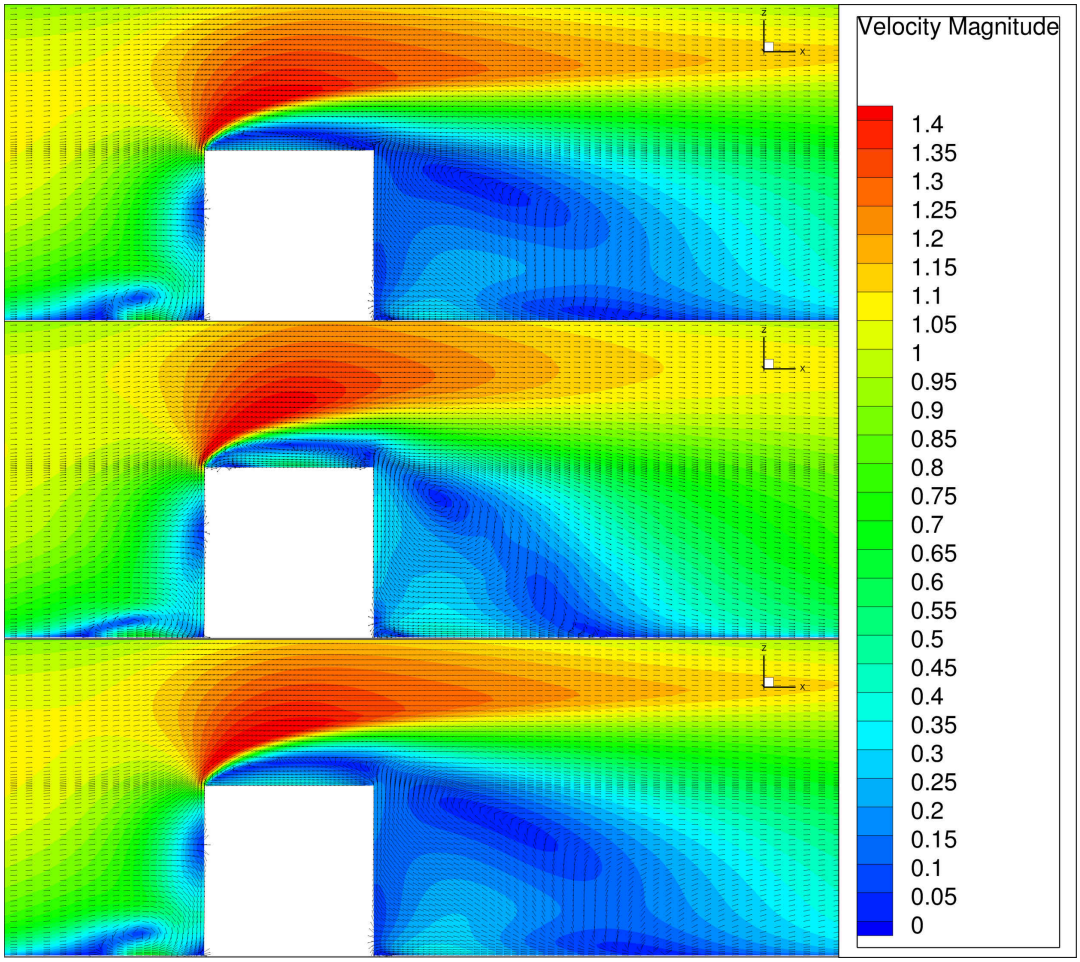


Figure 4.22: Comparing a correction model using 4 tensors and 2 invariants (Top) with a correction model using 10 tensors and 5 invariants (Bottom). The DES solution is shown in the centre. Velocity magnitude with vectors.

Looking at the streamtraces on the bottom of the domain in Figure 4.24 no big differences can be seen, there are some very small differences in the velocity magnitude near the recirculation areas directly behind the cube, but in terms of re-attachment locations and vortex trajectories no differences can be seen.

Moving the slice in the z-plane up to the centre of the cube gives Figure 4.25. This region showed the biggest improvement from both the corrections and the closest match to the DES results. In this comparison the result shows some very small differences in the wake as the velocity magnitude is generally lower in the model with 10 tensors and 5 invariants compared to the model limited to 4 tensors and 2 invariants. Overall the shape and size of the wake is the same for both models. This can also be seen when an iso-surface is plotted of the total pressure coefficient in Figure A.4. At first sight no differences can be spotted both in the shape and the colouring indicating the pressure values at these points in the domain.

What can be concluded from this comparison is that by using the full set of tensors and invariants no significant improvement in the final result is achieved and therefore due to the additional computational cost the computation of these additional terms introduce the models limited to 4 tensors and 2 invariants are preferable. Another benefit of limiting the number of tensors and invariants is that the number of basis functions used in the regression is limited and therefore the computational requirements for the regression are reduced, both in terms of computational time and memory usage. Because of this conclusion all the further models in this research will use the 4 tensors and 2 invariants for the basis functions.

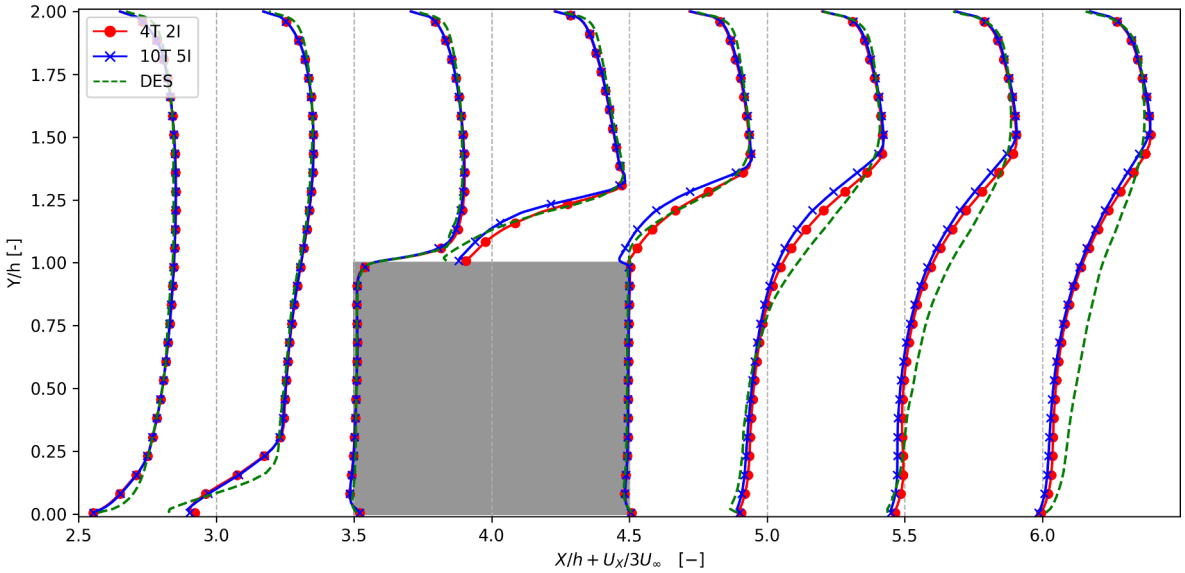


Figure 4.23: Comparing the velocity x-component for the correction models with 10 tensors and 5 invariants with the correction models using 4 tensors and 2 invariants.

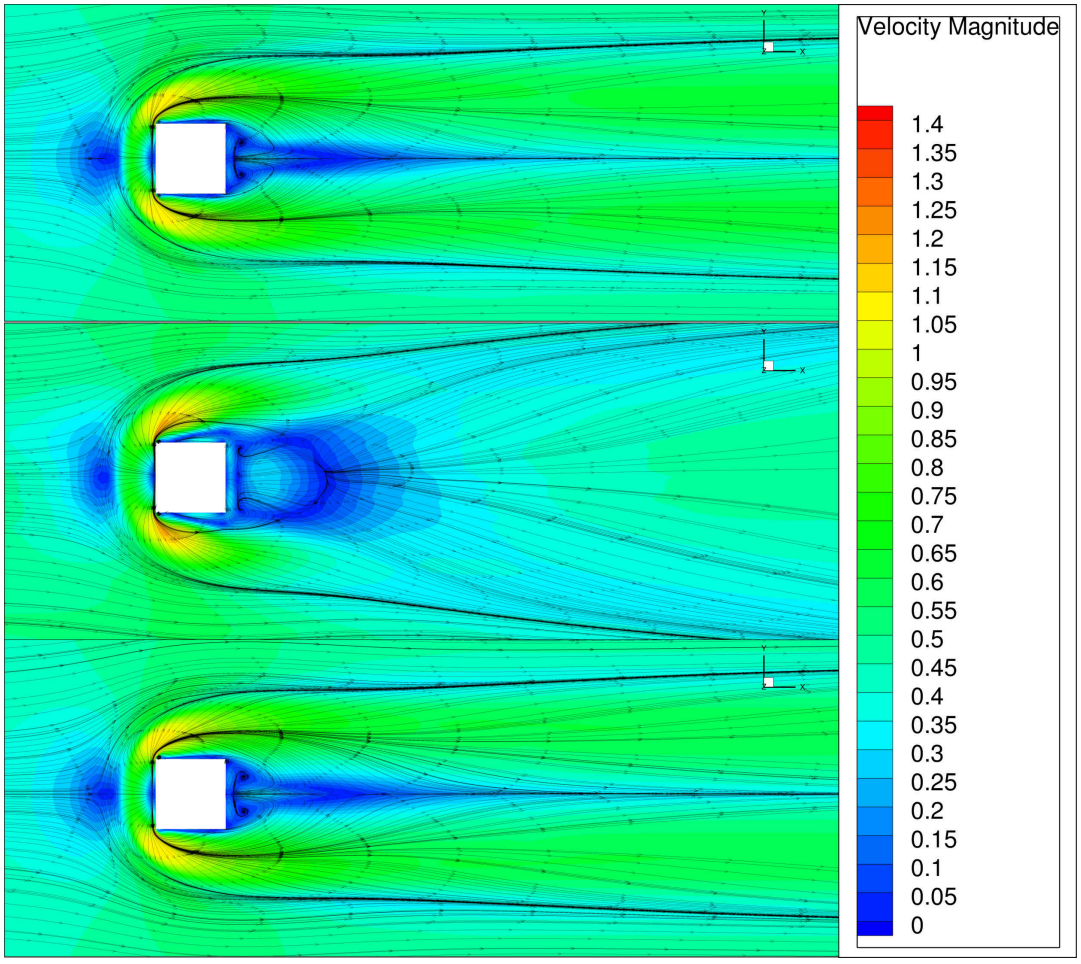


Figure 4.24: Comparing a correction model using 4 tensors and 2 invariants (Top) with a correction model using 10 tensors and 5 invariants (Bottom). The DES solution is shown in the middle. Velocity magnitude with surface streamlines on the bottom of the domain.

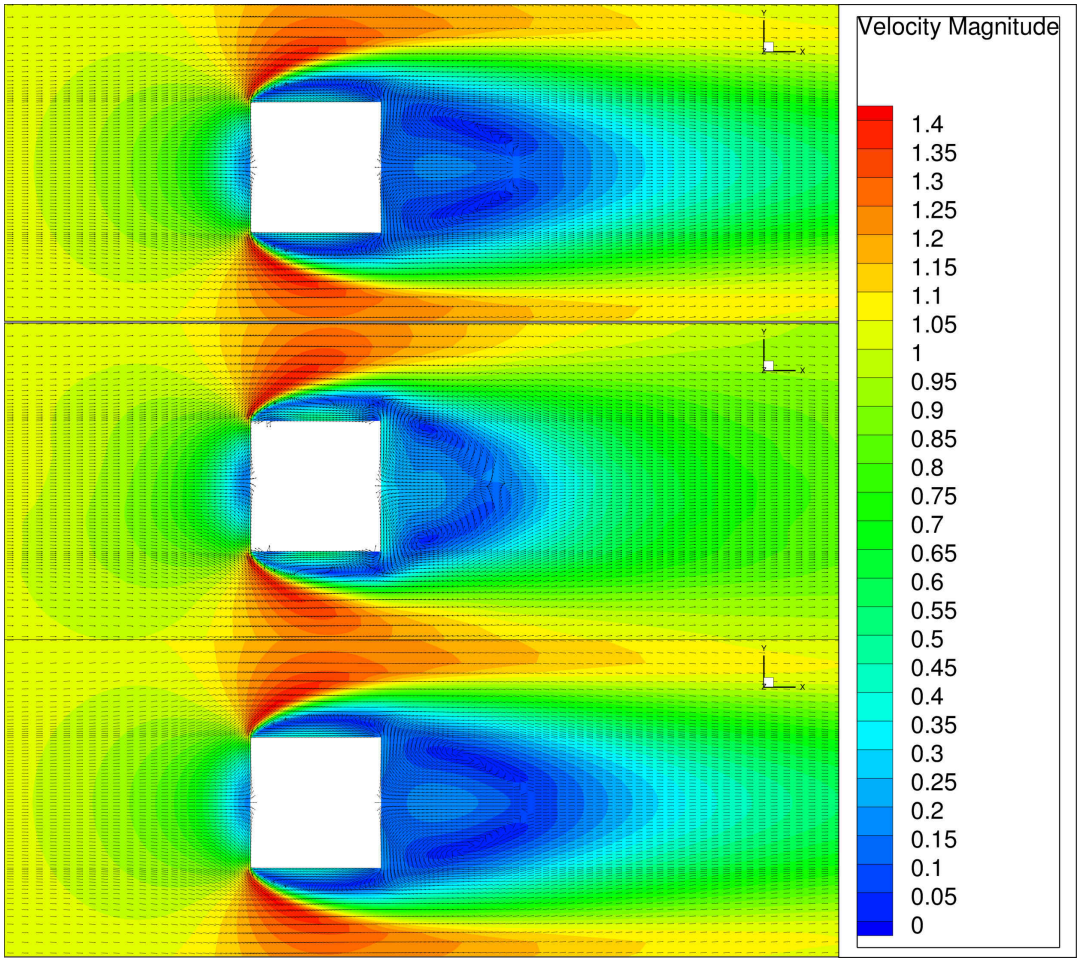


Figure 4.25: Comparing a correction model using 4 tensors and 2 invariants (Top) with a correction model using 10 tensors and 5 invariants (Bottom). The DES solution is shown in the centre. Velocity magnitude with vectors at half height of the cube z=0.5

4.1.8. Different levels of anisotropy correction

All the simulations that have been shown up to this point have been done with 100% R and 50% b_{ij}^{Δ} for stability reasons. What is interesting is to gradually apply the R and b_{ij}^{Δ} corrections and see how the solution changes and investigate the effect of each of the correction fields.

To see this change happen in Figure 4.26 sections are plotted for different converged fields with various levels of b_{ij}^{Δ} correction. The top field shows the baseline RANS solution without any correction fields, the second shows the solution of adding 50% R correction and the third with the full R-correction. The third image is of the solution where an additional 25% of the b_{ij}^{Δ} correction is added and finally the fourth image is of the solution with a total of 50% b_{ij}^{Δ} correction added. Adding more correction makes the solution unstable.

What can be seen is that the first image with half the R correction the result is already improved compared to the baseline solution and that when the full correction is added the solution is improved further. By adding the 25% of the b_{ij}^{Δ} correction the shape of the wake is modified and the re-attachment point is more in line with the DES solution. Also, the recirculation area on the top of the cube is lengthened making it closer to the DES solution. Adding 50% of this correction improves the circulation on the top even more, as well as the re-attachment region, however the shape of the wake is not necessarily more accurate in terms of the velocity magnitude.

Looking at the streamtraces in Figure 4.28 on the bottom of the domain gives an interesting insight in how both correction fields change the flow in proximity to the wall. The R correction mainly decreases the size

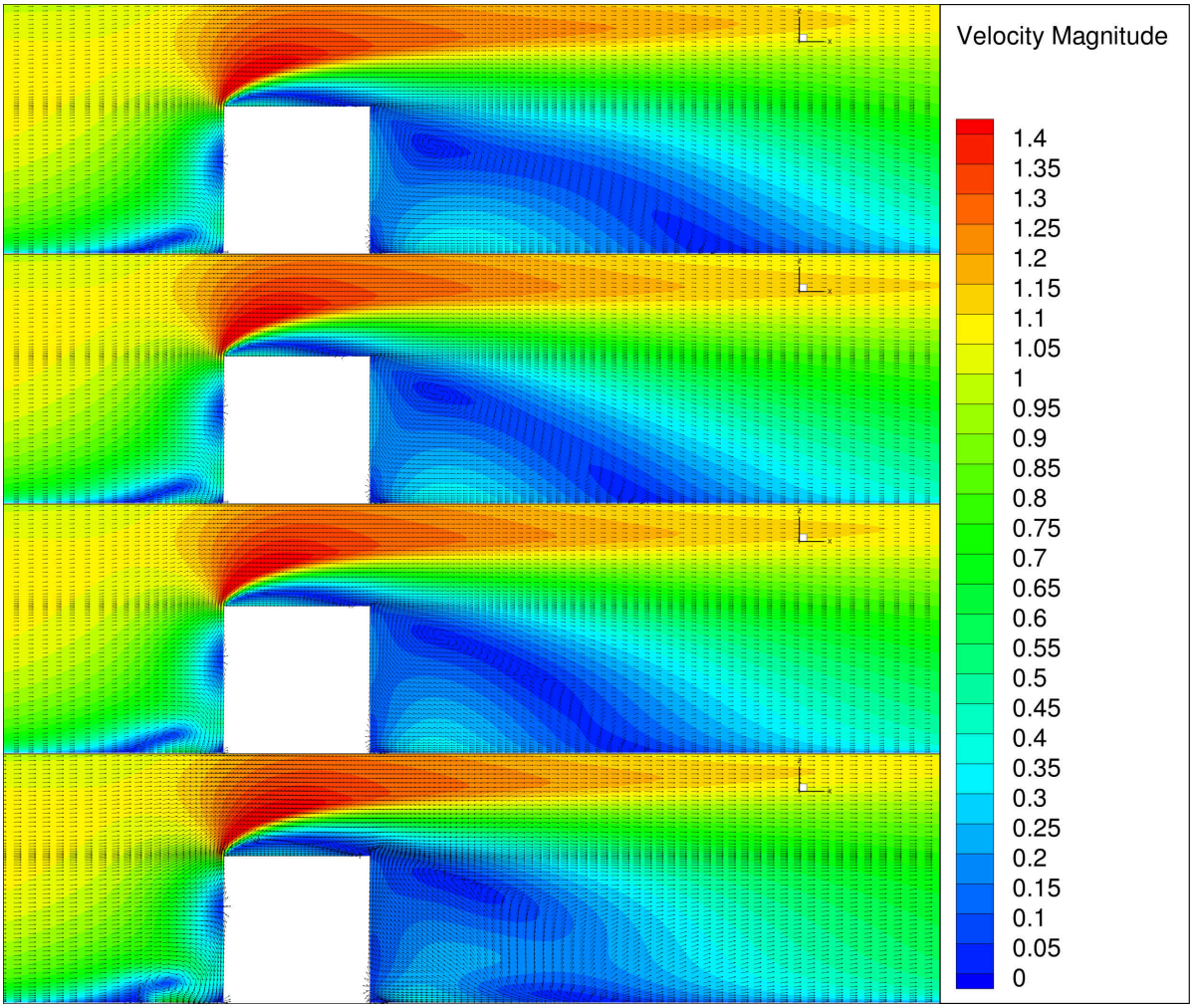


Figure 4.26: Visualisation of the effect of the corrective fields. 1st: 50%R 0%b_{ij}^{Δ} 2nd: 100%R 0%b_{ij}^{Δ} 3rd 100%R 25%b_{ij}^{Δ} 4th 100%R 50%b_{ij}^{Δ}. Velocity magnitude with vectors.

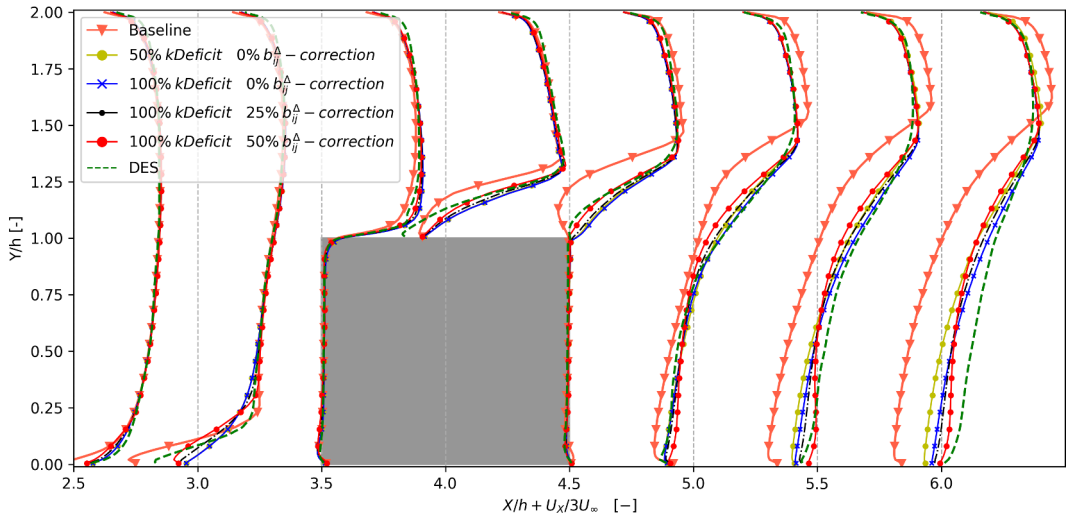


Figure 4.27: Comparing the velocity at the centre plane of the cube $y = 4.5$ for different levels of the correction fields

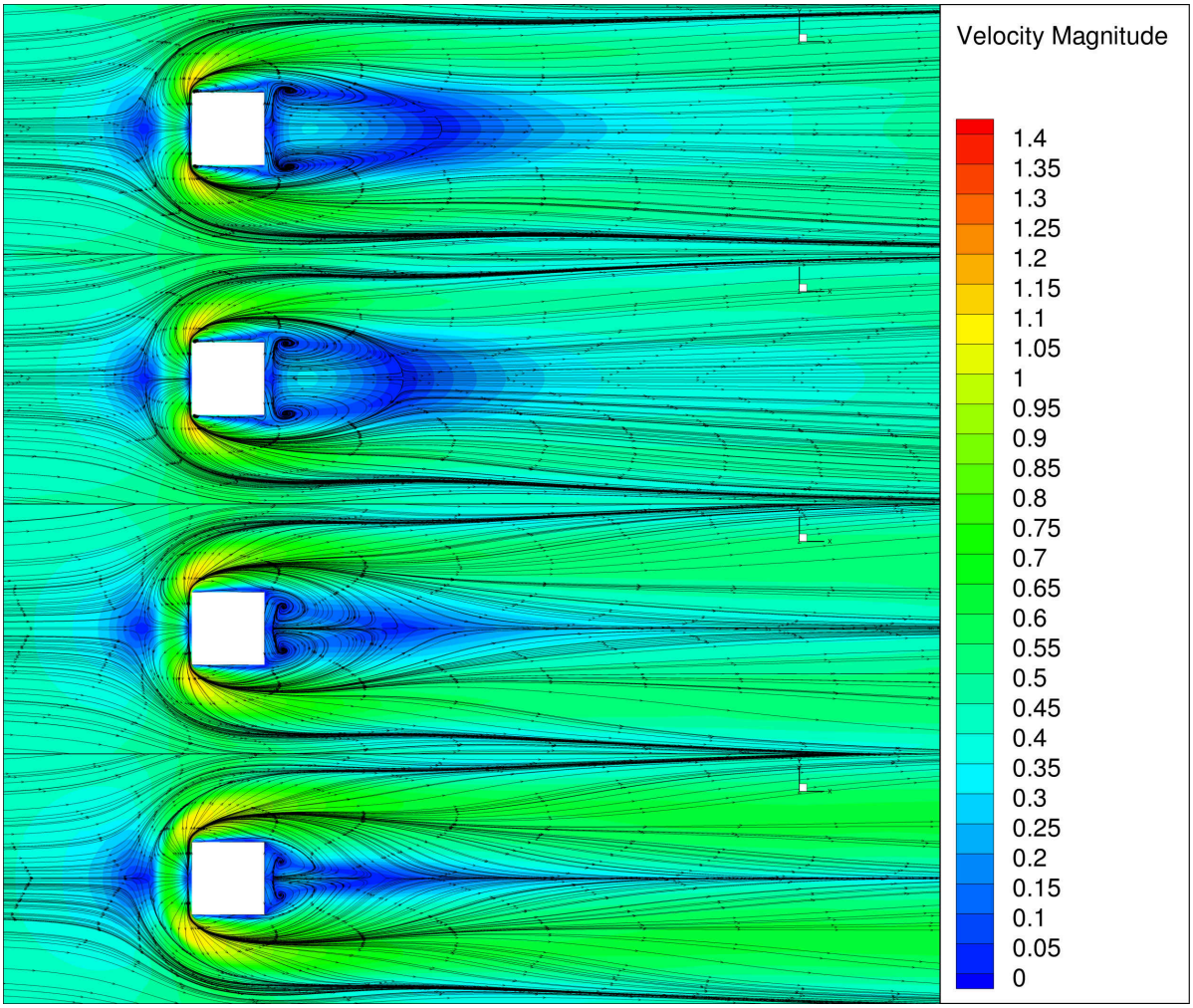


Figure 4.28: Visualisation of the effect of the corrective fields. 1st: 50%R 0%b_{ij}^{Δ} 2nd: 100%R 0%b_{ij}^{Δ} 3rd 100%R 25%b_{ij}^{Δ} 4th 100%R 50%b_{ij}^{Δ}. Velocity magnitude with streamtraces on the bottom surface.

of the wake, as adding half the R correction already moves the recirculation zone centre closer to the cube. Adding the full correction moves these further inboard and reduces the size of the wake even more. The b_{ij}^{Δ} correction field mainly moves these recirculation zones even more inboard, only in the centre of the cube the wake shape remains longer as it moves the wake in a pointy shape.

This pointy shape only happens near the bottom of the boundary as the flow on a plane a bit higher, shown in Figure 4.29 shows less of a pointy shape but clearly shows the reduction in size and a movement of the recirculation area when more correction is added. This is supported by the images shown in Figure A.5 which shows the velocity magnitude at the height of the middle of the cube.

Finally by looking at the total pressure coefficient through isocontours at $C_{p0} = 0.5$ in Figure A.6 the general reduction in wake shape is visualised. It also shows the narrowing of the wake near the bottom surface. The colouring of the pressure shows that the effect of the correction fields on the pressure field is small and that the correction mainly influences the resulting velocity field.

Averaging the unstable solutions

In all the simulations that have been with the model corrections the b_{ij}^{Δ} correction had to be limited to 50% of the original correction predicted by the model. After some investigation with slowly adding more correction as iterations progress it was found that with the full field the solution does not diverge but oscillates.

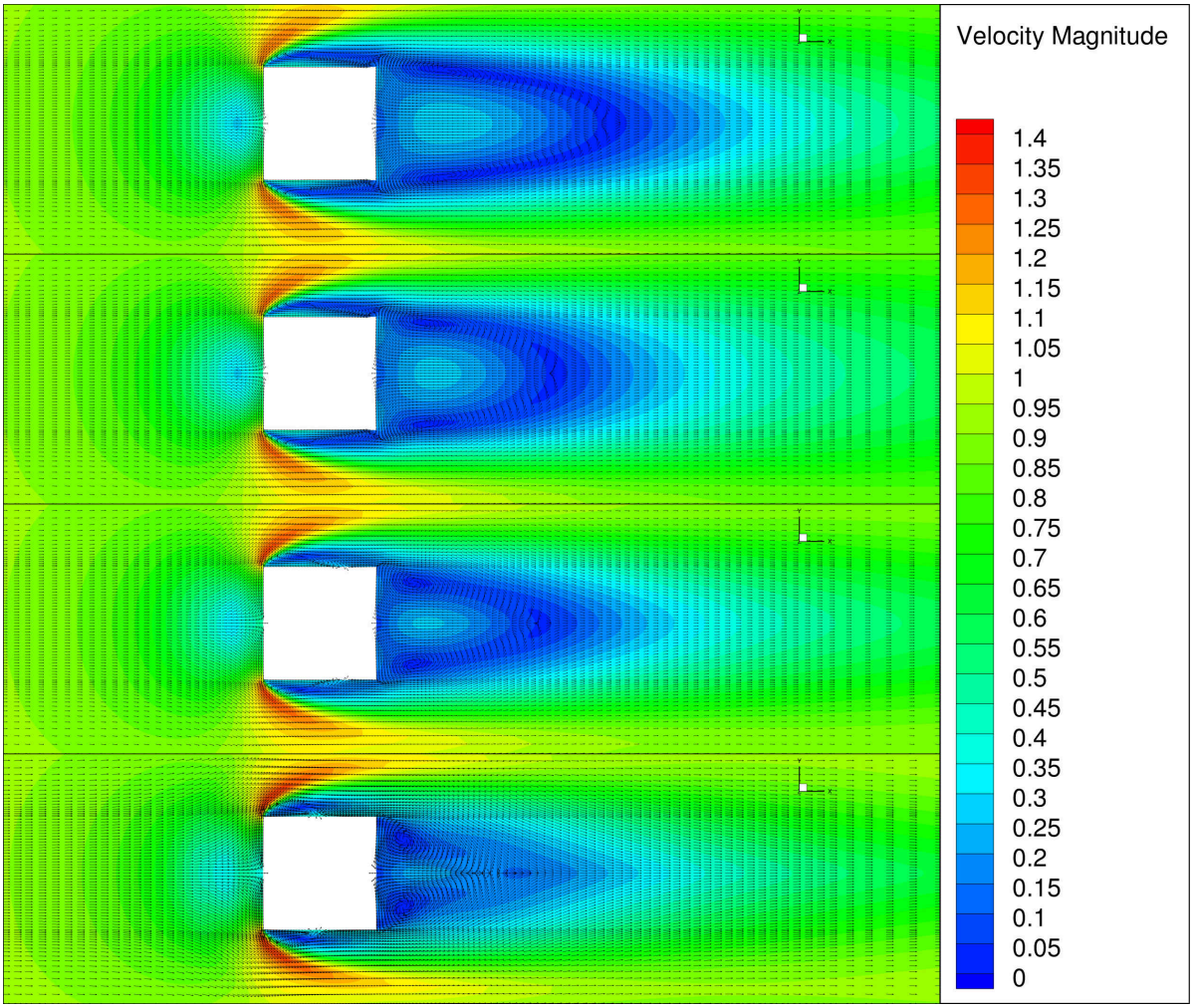


Figure 4.29: Visualisation of the effect of the corrective fields. 1st: 50%R 0%b_{ij}^{Δ} 2nd: 100%R 0%b_{ij}^{Δ} 3rd 100%R 25%b_{ij}^{Δ} 4th 100%R 50%b_{ij}^{Δ}. Velocity magnitude with vectors $z = 0.2$.

The hypothesis is that these oscillations are around a solution that comes close to the mean flow of the DES.

To investigate this a simulation has been done where the solution is first allowed to converge to a stable solution with the b_{ij}^{Δ} at 50% of the original. After this the full 100% of the correction is added to the solution. What can be seen in the residuals is a stable trend, they are not converging or diverging but staying around the same magnitude. This is left running for 1000 iterations after which the averaging is started for another 500-1000 iterations.

The results of this procedure are found below and they are compared to the solution of the flow at 50% b_{ij}^{Δ} correction and the DES mean flow. The averaged results are achieved by first allowing the solver to run for 1000 iterations from the 50% b_{ij}^{Δ} converged solution. After that the solver is run for another 1000 iteration over which the solution is averaged.

Looking at the results in Figure 4.30 it can be be seen that with the averaging parts of the flow show a more accurate prediction and others become worse. The part that has improved is the wake and the re-attachment zone at the bottom of the cube whose location has moved forward, more in line with the DES result. What has worsened is that the recirculation area on the top has reduced in size and as a result the circulation area behind the top trailing edge of the cube is positioned lower. The centre of the main recirculation region however is now in line with the prediction from the DES in contrary to the location of the 50% b_{ij}^{Δ} solution.

When a closer look is taken at the flow near the bottom of the domain only one small change can be observed and that is that the length of the wake is still a pointy shape but shorter, the recirculation area

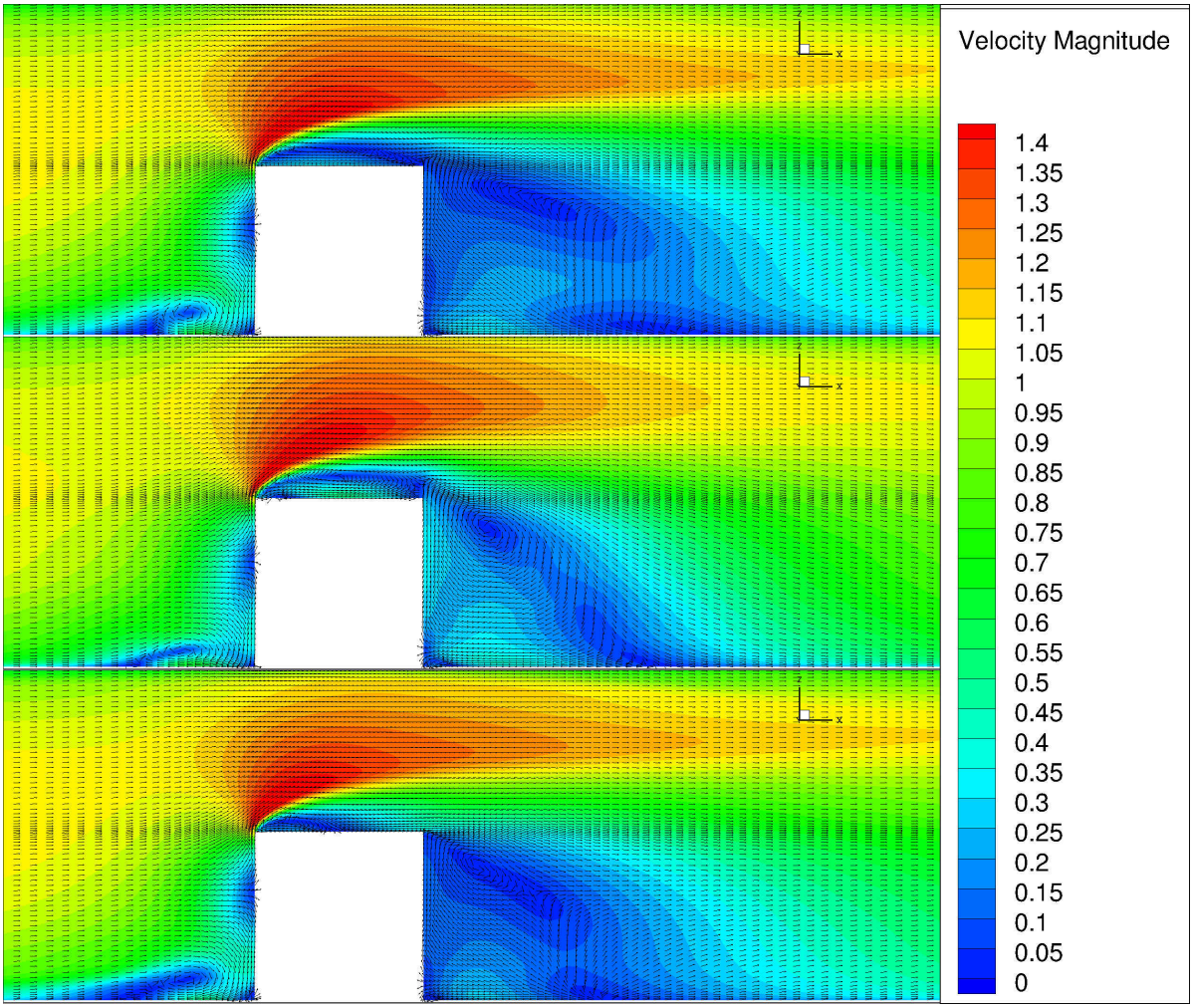


Figure 4.30: Comparing 50% b^{Δ}_{bij} correction (Top) with an averaged solution with 100% b^{Δ}_{ij} correction (Bottom). Both have 100% R correction. The middle is the DES solution for reference. Velocity magnitude comparison with vectors on the centre of the y-plane.

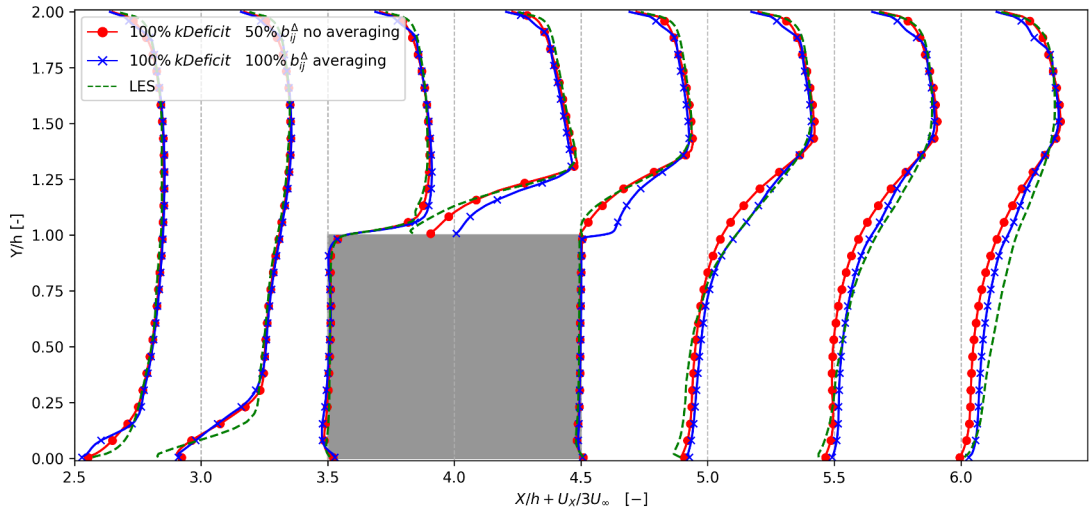


Figure 4.31: Comparing the velocity x-component for different levels of b^{Δ}_{ij}: 50%b^{Δ}_{ij} without averaging and 100%b^{Δ}_{ij} with averaging

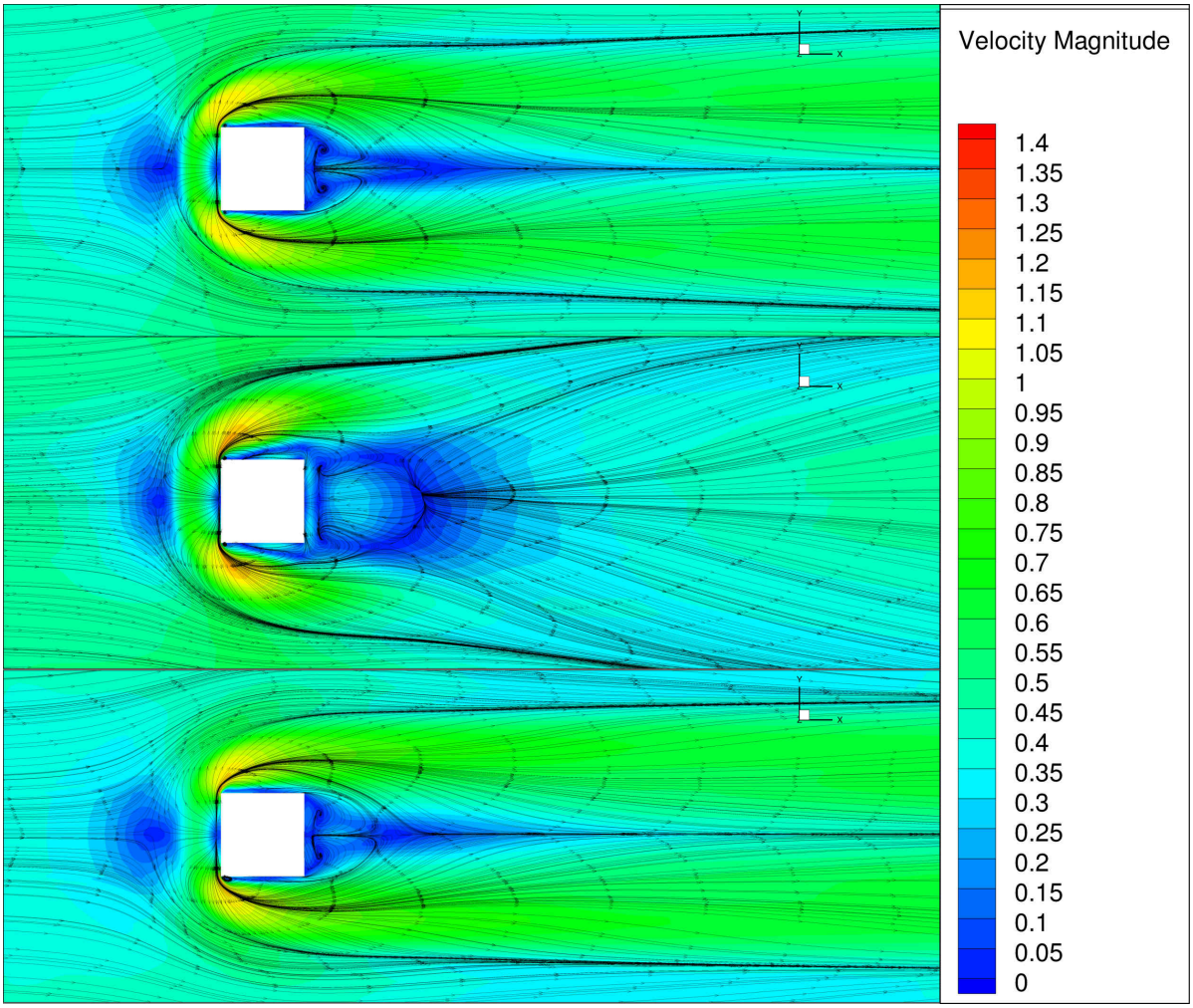


Figure 4.32: Comparing 50% b^{Δ}_{bij} correction (Top) with an averaged solution with 100% b^{Δ}_{ij} correction (Bottom). Both have 100% R correction. The middle is the DES solution for reference. Streamtraces on the bottom of the domain.

behind the cube is still the same shape but a bit smaller, meaning that the rotation centres are still too far inboard. The velocity profile a bit higher at $z = 0.2$ in Figure 4.33 shows a similar change, a shorter wake but the recirculation areas remain too close to the centre. The separation regions on the sides of the cube are smaller which is a move further away from the DES solution. Overall the shape of the wake is more in line with the DES solution, however the flow close too the cube is worse.

This trend seen at the bottom of the domain and at $z = 0.2$ is also visible at the middle height of the cube at $z = 0.5$ as can be seen in Figure A.7. Again the shape of the wake is more in line with the DES solution, however the flow near the sides of the cube is showing a smaller separation and recirculation region.

Finally when the total pressure coefficient is compared in Figure A.8 it can again be seen that the wake is shorter and in terms of shape more in line with the DES solution. The biggest difference however, can be seen at the bottom of the domain where the wake in the corrected RANS solution is a lot narrower than that of the DES solution. The re-attachment location and the recirculation are the same or close to the DES solution, especially for the averaged solution with the full correction field. However, the velocity magnitude and the total pressure do not recover as quickly as in the DES and therefore the wake is a lot longer and stronger in the corrected RANS solutions compared to the DES solution. Also what is missing is the phenomenon where the horshoe vortex as it traveles downstream is forced away from the centre of the middle of the domain. In all RANS simulations, both corrected and uncorrected this phenomenon is never captured and the horseshoe vortex goes travels downstream in an almost straight line.

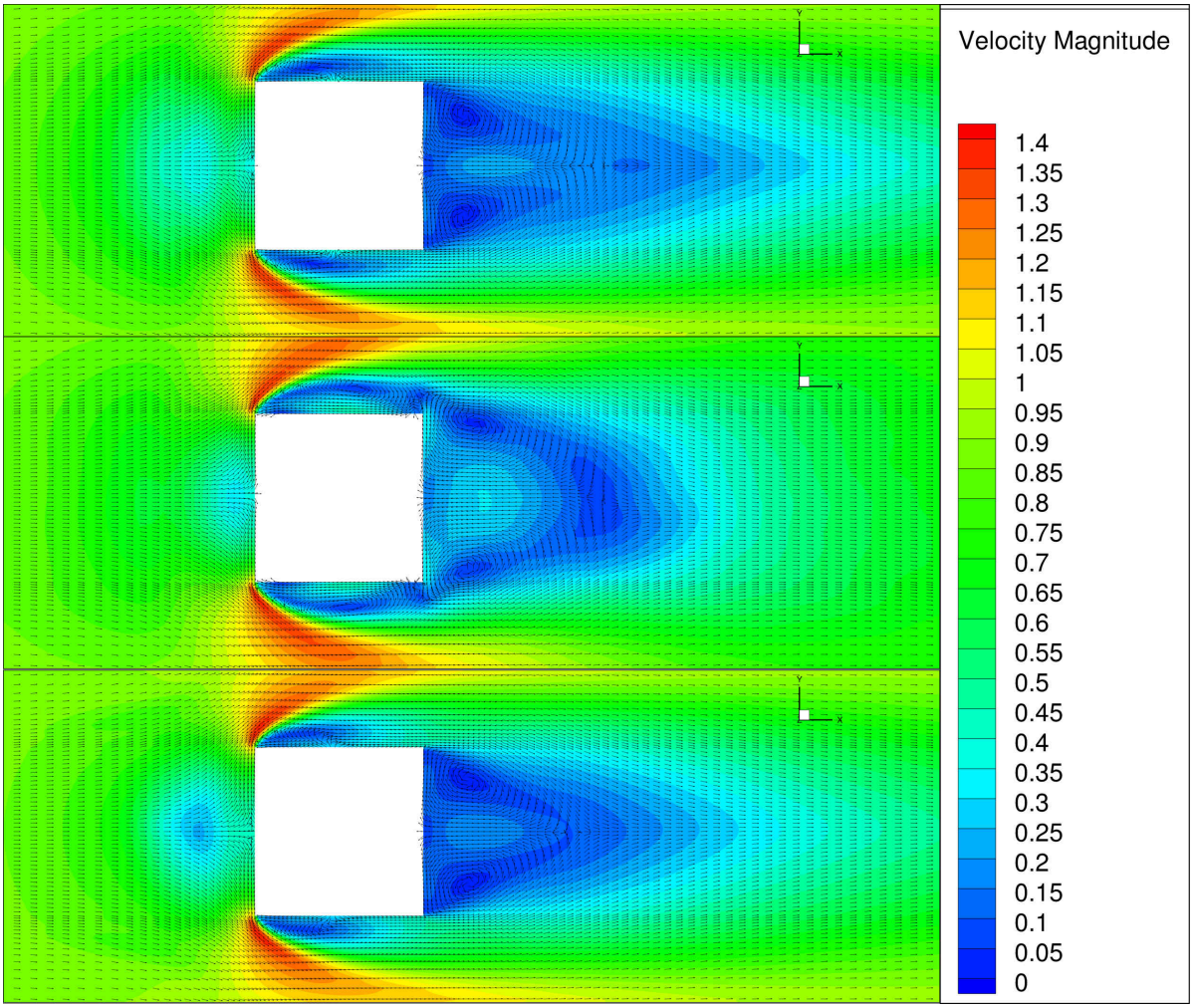


Figure 4.33: Comparing 50% b^{Δ}_{bij} correction (Top) with an averaged solution with 100% b^{Δ}_{ij} correction (Bottom). Both have 100% R correction. The middle is the DES solution for reference. Velocity magnitude on a plane at z=0.2

To find an explanation for why the solution becomes unstable for large correction values of b^{Δ}_{ij} a look is taken again at the corrective fields from the frozen approach in Figures 4.8a and 4.8b. These show the total correction in turbulence production and the direct contribution from the correction on the turbulent kinetic energy. Figure 4.8a shows that the total correction in turbulence is mainly a reduction of turbulence production. When comparing this figure to the direct contribution from the correction on the turbulent kinetic energy in Figure 4.8b it can be seen that this contribution is mainly positive thereby adding turbulence production and making the solution more stable. This concludes that the correction coming from the anisotropy correction is mainly a reduction in turbulence production. This is in line with the experience from the corrective models. The corrective model term directly correcting the turbulent kinetic energy is adding stability to the solution, while the anisotropy correction is making the solution more unstable.

4.1.9. Testing the model's extrapolating qualities

In order to asses the extrapolating qualities of the correction model, the model using 4 tensors and 2 invariants (4.1) (4.2) has been applied to other geometries as well. The results of which will follow in Section 4.2.6 and Section 4.3.5. These geometries are completely different and are therefore the ultimate test for the extrapolating qualities of this correction model. To get an idea of the performance of the model on a similar but slightly different geometry the cube test case has been altered to a cube that is 1.5x longer than the original, making it a wall mounted box of size $1.5 \times 1 \times 1$ ($l \times w \times h$).

Taking a first look at the results in Figure 4.34 shows a similar improvement to what was seen in the original box. The length of the wake has reduced and is almost similar to that of the DES, be it slightly longer and with a lower velocity magnitude directly behind the box. The recirculation area on top of the box has improved drastically with a reattachment area now visible.

This image is confirmed when looking at the x-component of the velocity in Figure 4.35 This shows that in all the areas the x-component of the velocity has moved towards the DES solution. Both in front of the box, near the horseshoe vortex location and in the wake the velocity profile has moved towards the DES solution, with the largest change visible on top of the box and directly behind the box, sometimes over-correcting.

Also when looking at the streamtraces at the bottom of the domain in Figure 4.36 a similar change due to the correction model can be seen as before. What is clear is that the recirculation areas directly behind the box are moved inboards and closer to the box. he correction near the bottom plane is not necessarily an improvement as the anisotropy correction causes an over-correction in this area, as was also seen in the previous results. The improvement is more obvious when looking at the velocity magnitude at the middle height of the box in figure A.10, this shows a velocity profile similar to that of the DES solution and a good improvement over the baseline.

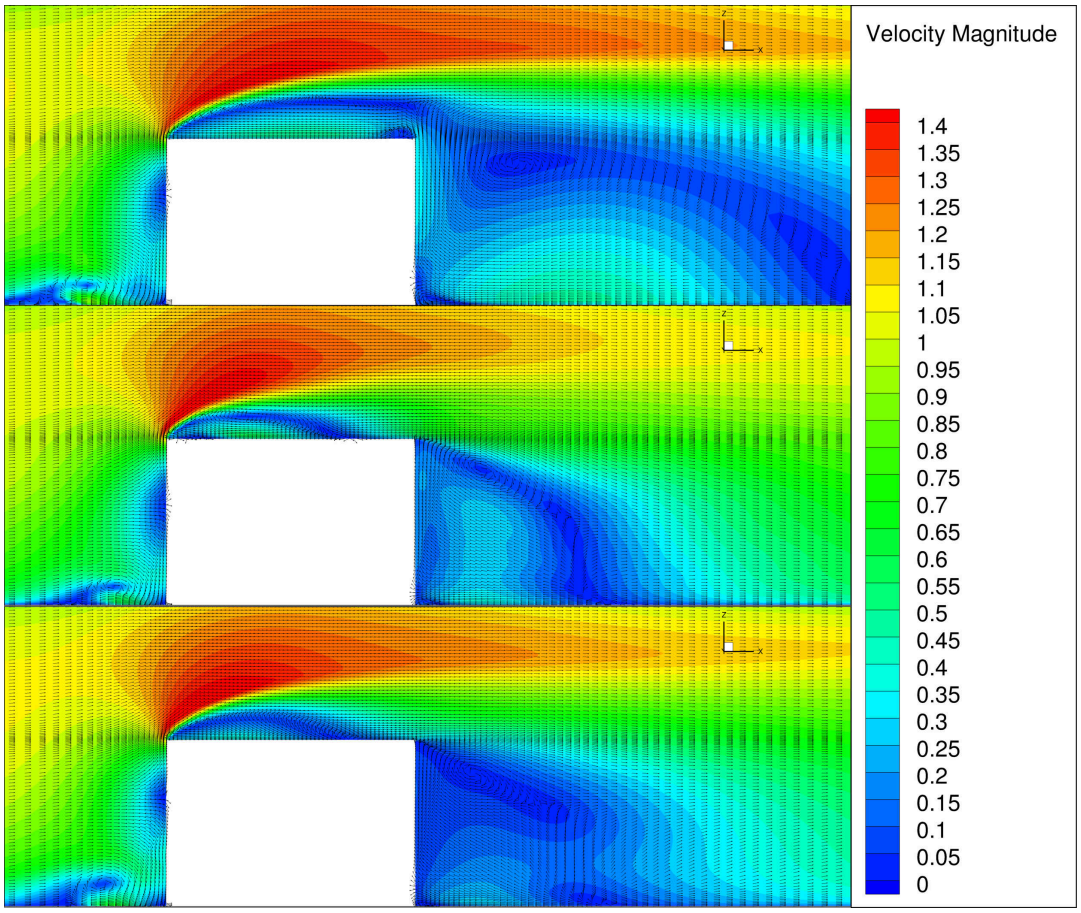


Figure 4.34: Velocity magnitude with vectors at the centre of the box. Top: baseline, middle: DES, bottom: 100%R 50%b_{ij}^{Δ} correction.

Finally when the wake is visualised through an isosurface of the total pressure in figure A.9 the reduction in wake size and the change in shape is clearly visible. In this image the influence of the correction model on the wake shape especially near the bottom plane is also clearly visible. It also shows that even though the wake has reduced significantly with the application of the correction model it is not matching the DES yet, for this a greater correction is needed.

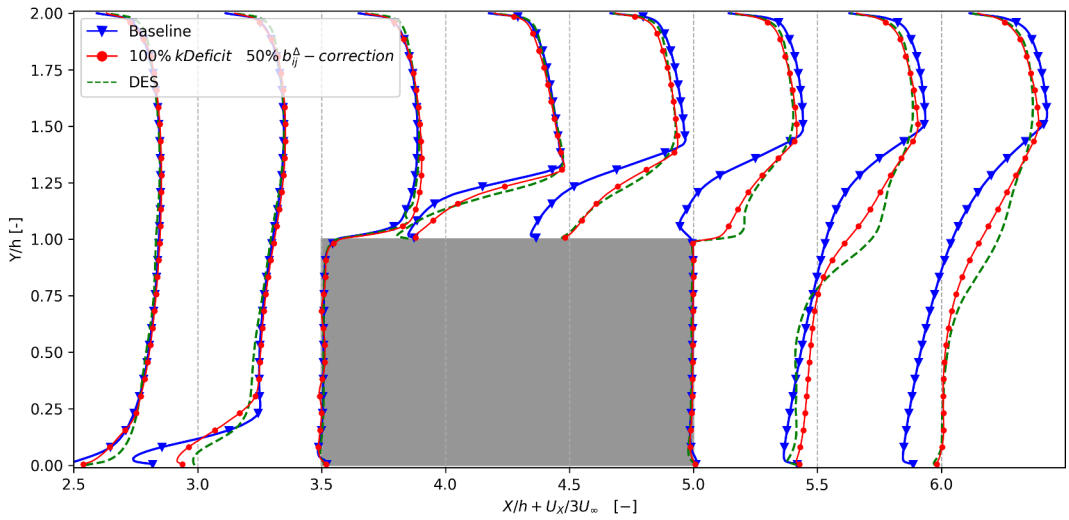


Figure 4.35: Directly comparing the X-velocity component in the centre plane of the box

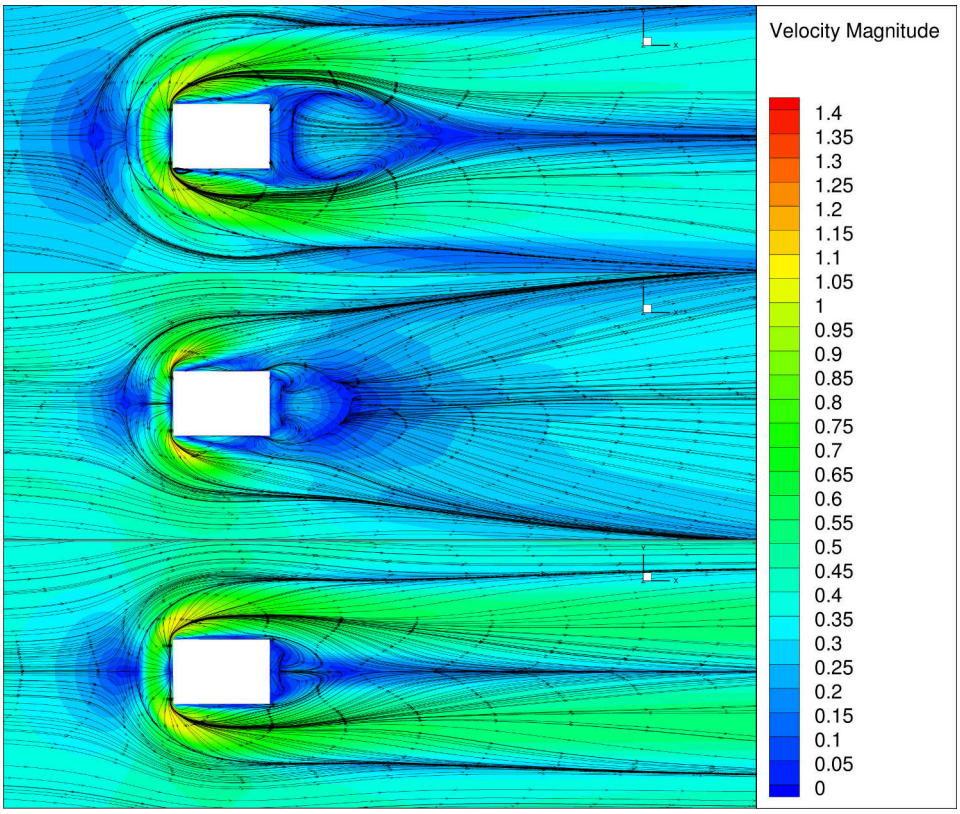


Figure 4.36: Velocity magnitude with streamtraces on the bottom of the domain of the box. Top baseline, middle DES, bottom $100\%R, 50\%b_{ij}^{\Delta}$

4.2. Infinite circular cylinder

A very challenging case for a RANS solver is that of the infinite circular cylinder. As the flow is generally unstable especially at low Reynolds numbers, the RANS solvers generally struggle to converge to the correct mean flow.

The setup for this case is a circular cylinder placed in the centre of a domain that has a length of 29m, a height of 18m and a width of π. The diameter of the cylinder is set 1, combined with an inflow velocity of 1m/s and a dynamic viscosity of 7.14285×10^{-6} the Reynolds number is set to 140000. The cylinder centre is placed 9 m after the inflow plane and in the centre, height wise. The boundary conditions for the sides are set to symmetry planes to create the theoretically infinite cylinder.

In contrary to the wall mounted cube for this flow case, not a DES but a wall resolved LES was performed to be used as the high fidelity training data for the SpaRTA methodology. This high fidelity data set is described in Section 4.2.1. After this the setup of the baseline RANS simulation is discussed in Section 4.2.2 with the frozen approach results in Section 4.2.3. From the results of the frozen approach the propagated approach was tried but resulted in diverging results, this is discussed in Section 4.2.4. The model regression from the frozen approach results are shown in Section 4.2.5 of which the results when applied in the turbulence model are shown in Section 4.2.6, where they are compared to results from the model trained on the Cube, the baseline and the LES results.

4.2.1. Cylinder high fidelity data source: LES

The LES was performed on a structured mesh created with the gmesh software. This software allow for the creation of fully structured and unstructured meshes. The mesh can be seen in figure A.11 with the characteristics shown in Table 4.4. The maximum values are mainly dominated by the corners where the circular regions transforms into the straight sections.

Number of Cells	6071490
Cell Type	Hexahedra
Max Aspect Ratio	684.600
Min Volume	9.910e-08
Max Volume	0.013102709
Mesh max non-orthonogonality	44.556061
Max skewness	0.54174513

Table 4.4: Characteristics of the mesh used for the LES of the infinite cylinder.

The setup of the LES used explicit time-stepping with the WALE SGS model and synthetic turbulent inflow as described in Section 3.1.3. For this LES the pimpleFoam solver has been used with dynamic time step adjustments to keep the CFL number constant at a value of 0.8.

4.2.2. Baseline RANS

To have a reference for the model performance a baseline RANS simulation has been run. This simulation was run on a smaller mesh than the LES. The topology of the mesh is the same but the number of cells has been reduced as the mesh gets coarser more quickly from the cylinder outboards. The mesh is made for a wall resolved RANS with a target of $y+ = 1$.

Number of Cells	851776
Cell Type	Hexahedra
Max Aspect Ratio	1207.8738
Min Volume	3.7991e-07
Max Volume	0.1335
Mesh max non-orthonogonality	44.2376
Max skewness	0.6308

Table 4.5: Characteristics of the mes used for the RANS of the infinite cylinder.

The stability of the solution has proven troublesome as for this flow case the flow is generally unstable as was discovered from the LES. The flow generally flaps behind the cylinder as the separation areas on the

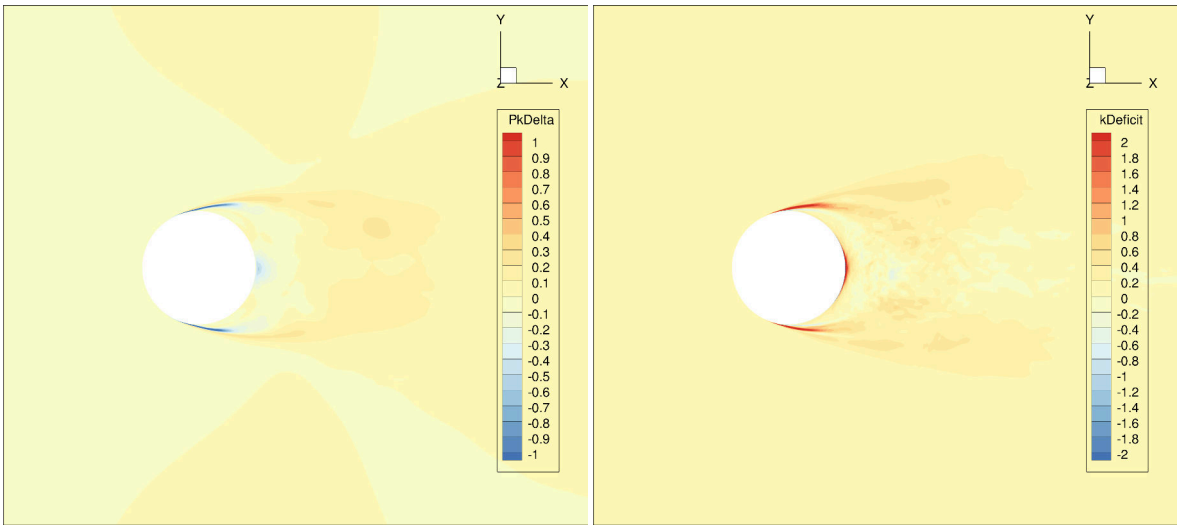
bottom and top sides of the cylinder shift locations constantly creating the so called von Kármánn vortex street. This flapping motion was still present in the RANS simulation, even though the method is solving for the average solution. Therefore the decision was made to let the solver run until the Residuals were no longer decreasing, to then average the solutions over the last 500 iterations to get an estimate for the mean flow.

4.2.3. Correction fields calculation: Frozen Approach

With the averaged LES results the frozen approach was run on the same mesh as the LES. The method converged in 2500 iterations to a stable solution for ω and the correction fields b_{ij}^{Δ} and R.

The averaged flow in this case should be identical at each section through the cylinder, therefore a slice through the centre of the cylinder will be used to visualise the flow and correction fields. This assumption of identical flow at each section is not necessarily true as can be seen in Figure 4.41, but it also shows that the centre location is the best to compare results as there the influence of the symmetry planes is the smallest.

In Figure 4.37a the total correction in turbulence production from both the anisotropy correction and the direct k correction is shown. This figure shows that there is a sharp region where there is the biggest correction in turbulence production and dissipation. Close to the cylinder there is a reduction in turbulence production with slightly further away from the cylinder an area of additional production. When looking at the direct correction in the turbulent kinetic energy correction in figure (4.6) it can be seen that this correction mainly adds turbulence production. From this it can be concluded that the correction from the anisotropy correction then must be mainly negative.



(a) Total turbulent kinetic energy production correction

(b) Direct turbulent kinetic energy correction, R.

Figure 4.37: Cylinder frozen approach corrective fields.

4.2.4. Correction field propagation

The first test done with the corrective fields is to start from the LES solution and add the corrective fields from the frozen approach and let a RANS simulation run from here. This method worked as intended, even though the initial residuals were not extremely small, but in contrary to this approach applied to the wall mounted cube where the solution diverged. The hypothesis for this is that this cylinder is wall resolved in contrary to the cube where wall models are used.

This led to a final result that did not converge to a stable solution, as instabilities caused the solution to remain flapping behind the cylinder, with the average solution close to the LES solution. The next step is then to propagate these correction fields on a baseline RANS solution. This baseline RANS is run on the same mesh as the LES and frozen approach, in contrary to the smaller mesh that will be used for the modelled approach. This approach led to an immediately diverging solution, with the correction fields immediately causing the dissipation values to shoot to infinity locally. This was to be expected as what is generally seen

in this propagated approach is that if the baseline solution is far off the original and is already struggling to converge properly, adding the corrective fields directly rarely leads to a stable solution.

4.2.5. Correction model Regression

With the partial success of the propagated approach the corrective field is now added through modelling of the corrective fields. The resulting models that were found for b_{ij}^{Δ} and R through the SpaRTA methodology are shown in figures 4.38 and 4.39. With the basis functions on the x-axis being available in Table 4.6. These images show a similar buildup of model structures compared to the models of the cube. With the simplest functions always occurring in the models and the error compared to the training data increasing with less basis functions used and smaller coefficients.

This observation is consistent for the b_{ij}^{Δ} term but the models for the R term show some new observations. It shows very large coefficients for the regression without the l_2 norm applied (ridge $\alpha = 0$). It is also not necessarily the case that the models with a higher number of basis functions fit the training data better.

1	$I_1 T^{(1)}$	7	$I_2 T^{(4)}$	13	$I_2^2 T^{(2)}$	19	$I_2^3 T^{(2)}$	25	$I_2^4 T^{(1)}$	31	$I_1 I_2^3 T^{(1)}$
2	$I_1 T^{(2)}$	8	$I_1^2 T^{(1)}$	14	$I_2^2 T^{(4)}$	20	$I_2^3 T^{(4)}$	26	$I_2^4 T^{(2)}$	32	$I_1 I_2^3 T^{(2)}$
3	$I_1 T^{(3)}$	9	$I_1 I_2 T^{(1)}$	15	$I_1^2 T^{(2)}$	21	$I_1^2 I_2 T^{(1)}$	27	$I_2^4 T^{(4)}$	33	$c T^{(1)}$
4	$I_1 T^{(4)}$	10	$I_1 I_2 T^{(2)}$	16	$I_1^2 T^{(3)}$	22	$I_1 I_2^2 T^{(1)}$	28	$I_1^2 I_2 T^{(2)}$	34	$c T^{(2)}$
5	$I_2 T^{(1)}$	11	$I_1 I_2 T^{(4)}$	17	$I_1 I_2 T^{(3)}$	23	$I_1 I_2^2 T^{(2)}$	29	$I_1^2 I_2 T^{(4)}$	35	$c T^{(3)}$
6	$I_2 T^{(2)}$	12	$I_2^2 T^{(1)}$	18	$I_2^3 T^{(1)}$	24	$I_2^3 T^{(3)}$	30	$I_2^4 T^{(3)}$	36	$c T^{(4)}$

Table 4.6: Basis functions used for the regression of the correction model for the cylinder

The models that have been selected to be implemented into the turbulence model are model number 84 (4.5) for the b_{ij}^{Δ} term and model number 340 (4.6) for the R term. These have been selected because they are some of the simplest functions with a relatively small error on the training data. For the b_{ij}^{Δ} term the model was selected with the smallest number of basis functions without the jump in error on the training data, that can be seen between models 12 and 13 for example.

In this case a ridge α value of 1 was necessary to get model coefficients of reasonable magnitude. This was done after experiencing divergence during experiments where first models were selected from the ridge $\alpha = 0.01$ and $\alpha = 0.1$ settings. This is because in contrast to the cube case the coefficients remained relatively large, especially with a large anisotropy correction. This large anisotropy correction will cause some trouble in terms of stability as can be seen later in the results.

$$\begin{aligned} b_{ij}^{\Delta} = T^{(1)}(19.24392 I_1 + 57.85607 I_2 + 2.93929) \\ +9.69506 T^{(2)} + 7.80504 T^{(3)} + 1.17078 T^{(4)} \end{aligned} \tag{4.5}$$

$$\begin{aligned} R = T^{(1)}(-2.82333 I_1 + 33.82069 I_2 + 2.58612) \\ +11.16664 T^{(3)} - 3.10679 T^{(4)} \end{aligned} \tag{4.6}$$

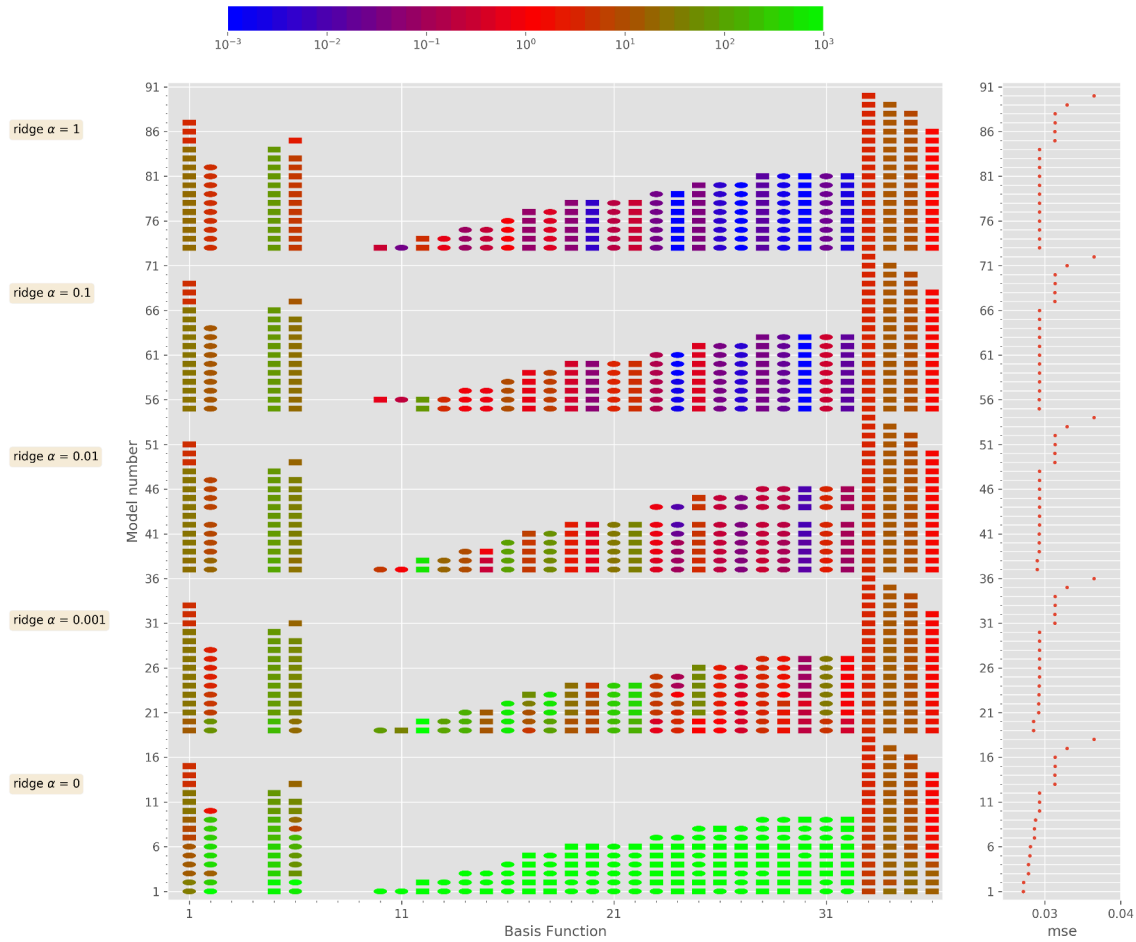


Figure 4.38: Overview of the models found for b_{ij}^{Δ} from both regression steps performed on the cylinder training data.

4.2.6. Model corrected RANS

With the correction models selected (4.5) (4.6) and implemented the corrected $k-\omega-SST$ model can be tested. This is compared directly with the correction model that is already available from the cube case. This model is found in (4.1) (4.2) and tested extensively in Section 4.1.6.

The first big difference between the two correction models became clear quickly during the first simulation runs. The way the correction model in (4.5) and (4.6) were added to the solver, was by first adding the R correction, which made the solution a bit more stable, but not by much. When adding the b_{ij}^{Δ} correction on top of this it showed that the maximum percentage of this correction field that could be added was 25% of the original equation. When adding more of the correction, the solution would diverge and crash. This is in contrary to the cube correction model, of which the full fields could be added to the turbulence model without the solution diverging.

For the results shown in the following sections the maximum correction fields are added to the solution for which the solution did not diverge. Due to the instabilities for all RANS simulations an average of the solution is taken over the last 1000 iterations, where solver residuals remained of the same magnitude. For the quality of the solution specific attention is paid to the size of the wake behind the cylinder, the separation point on the cylinder and the symmetry of the averaged solution.

Firstly when looking at the x-component of the velocity in the wake of the cylinder in Figure 4.40 shows that both correction models have improved the solution. With the model that has been trained on the cube

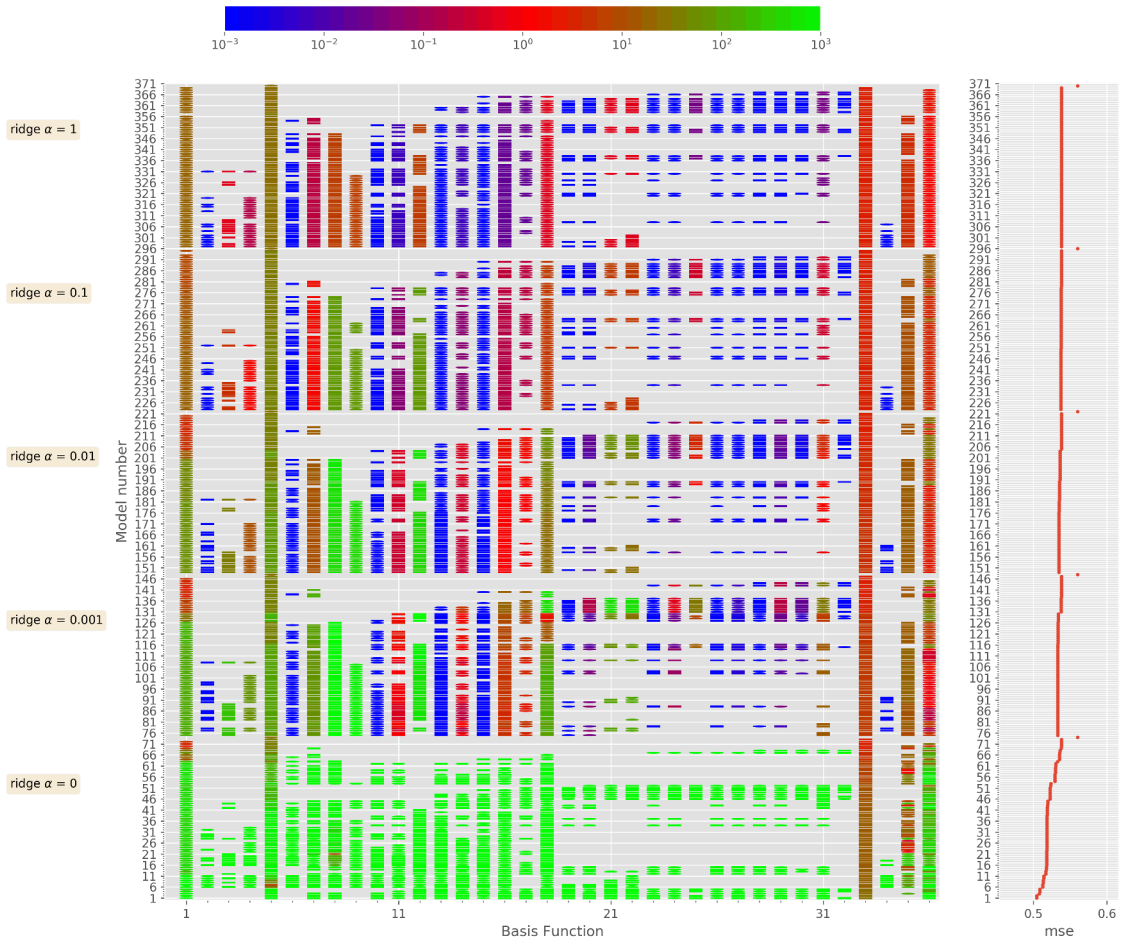


Figure 4.39: Overview of the models found for R from both regression steps performed on the cylinder training data.

matching the velocity profile of the LES generally better than that trained on the cylinder case.

To illustrate the differences in stability between the different simulations and correction models a slice is done through the middle of the cylinder to show the differences in velocity magnitude behind the cylinder. This is shown in Figure 4.41 where there are some clear differences shown between the span-wise solution similarity. It shows in all simulations an effect of the symmetry boundary condition. Either locally increasing the wake length, which is the case for the LES and the cylinder trained model. Or decreasing the wake length, as is the case for the baseline RANS and the cube trained correction model. It also shows some instabilities near these sides for the cylinder trained correction model, being indicative of the local instabilities these correction fields bring.

The second interesting point is to take a look at the separation point on the cylinder. This is again done at the centre of the cylinder by looking at the velocity magnitude in Figure 4.42. In this figure it becomes clear that the baseline solution predicts the separation point to be ever so slightly too far aft on the cylinder and that with the corrections this point moves even further aft, with the cylinder correction model moving the solution the furthest back.

Taking a look at the total pressure coefficient in Figure 4.43 shows the size and magnitude of the wake. In this again some of the instabilities in the solution of both correction models can be seen. The strange behaviour at the stagnation point might be due to the averaging procedure. The averaging of a RANS simulation does not work the same as the averaging of an LES, as the time step of a RANS simulation does not represent

physical time, the solutions over which are averaged might be mainly the two extremes on both sides and not the solution between the two.

One of the things that definitely did not improve is the pressure coefficient in Figure 4.44, with a slight improvement in the wake but a very large increase at the stagnation point which is not visible in the LES solution, besides this the pressure drop at the high velocity areas just before the separation region, this is something that was not seen in the cube case where the pressure coefficient saw is slight improvement towards the DES solution.

In conclusion, both the model trained on the cylinder and the model on the cube improve the velocity profile, but create a bigger difference in terms of pressure, this flow case remain very challenging for RANS and the requirements for a correction model to make sure the solution converges towards a stable solution.

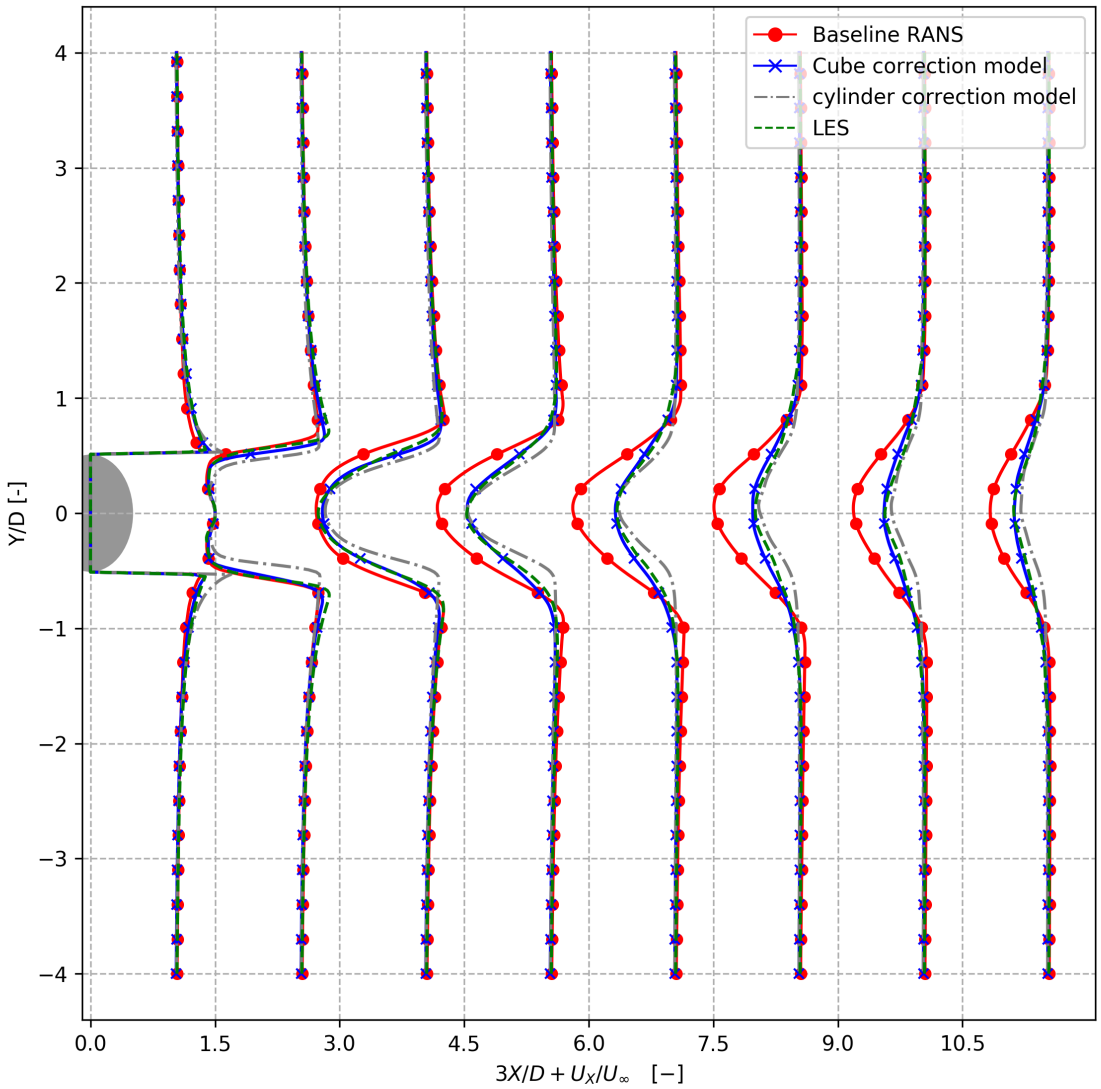


Figure 4.40: Velocity x-component comparison for the cube correction model

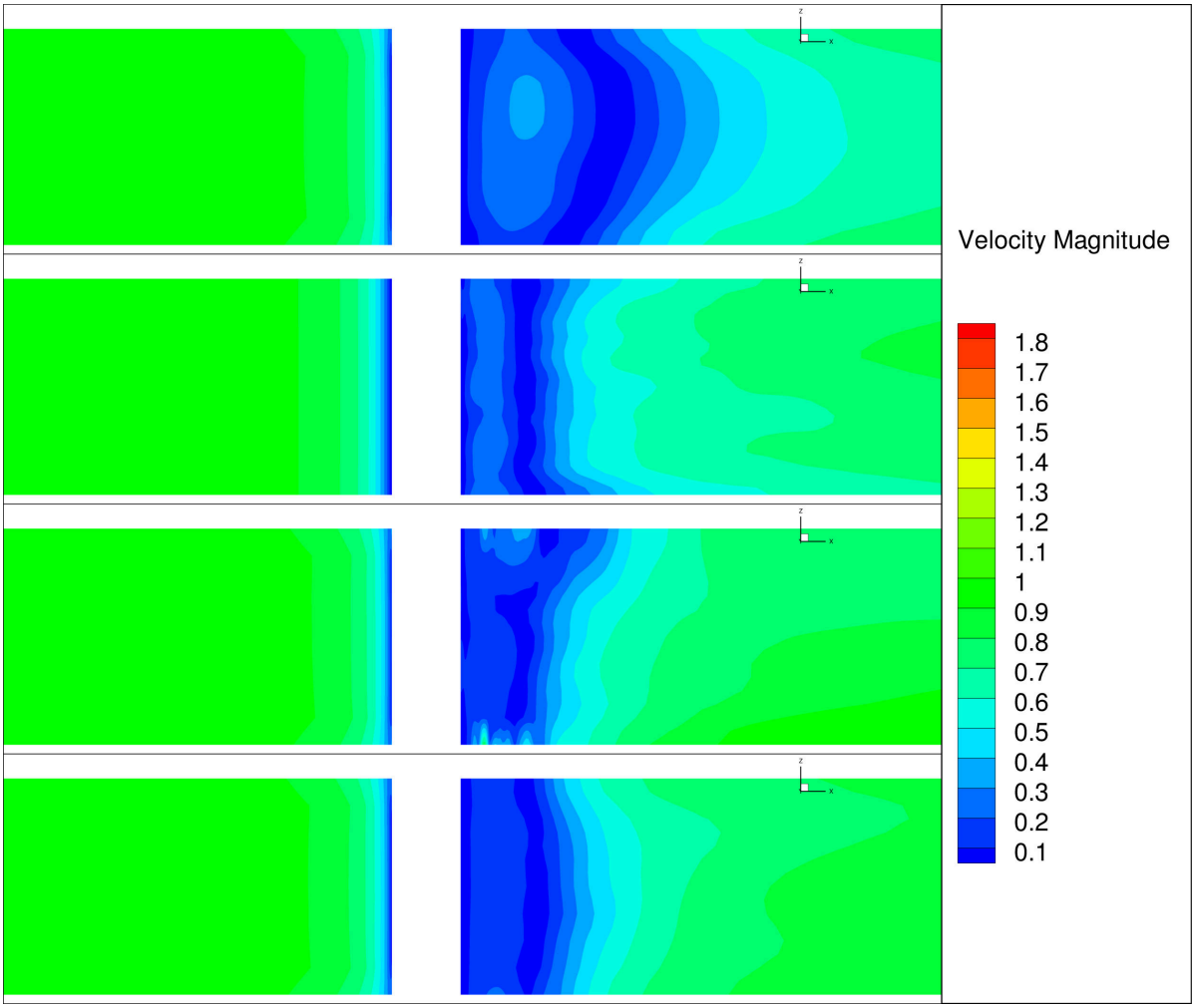


Figure 4.41: Comparison of span-wise velocity magnitude distributions. Top: Baseline RANS, second: LES, third: RANS cylinder correction model, Bottom: RANS cube correction model.

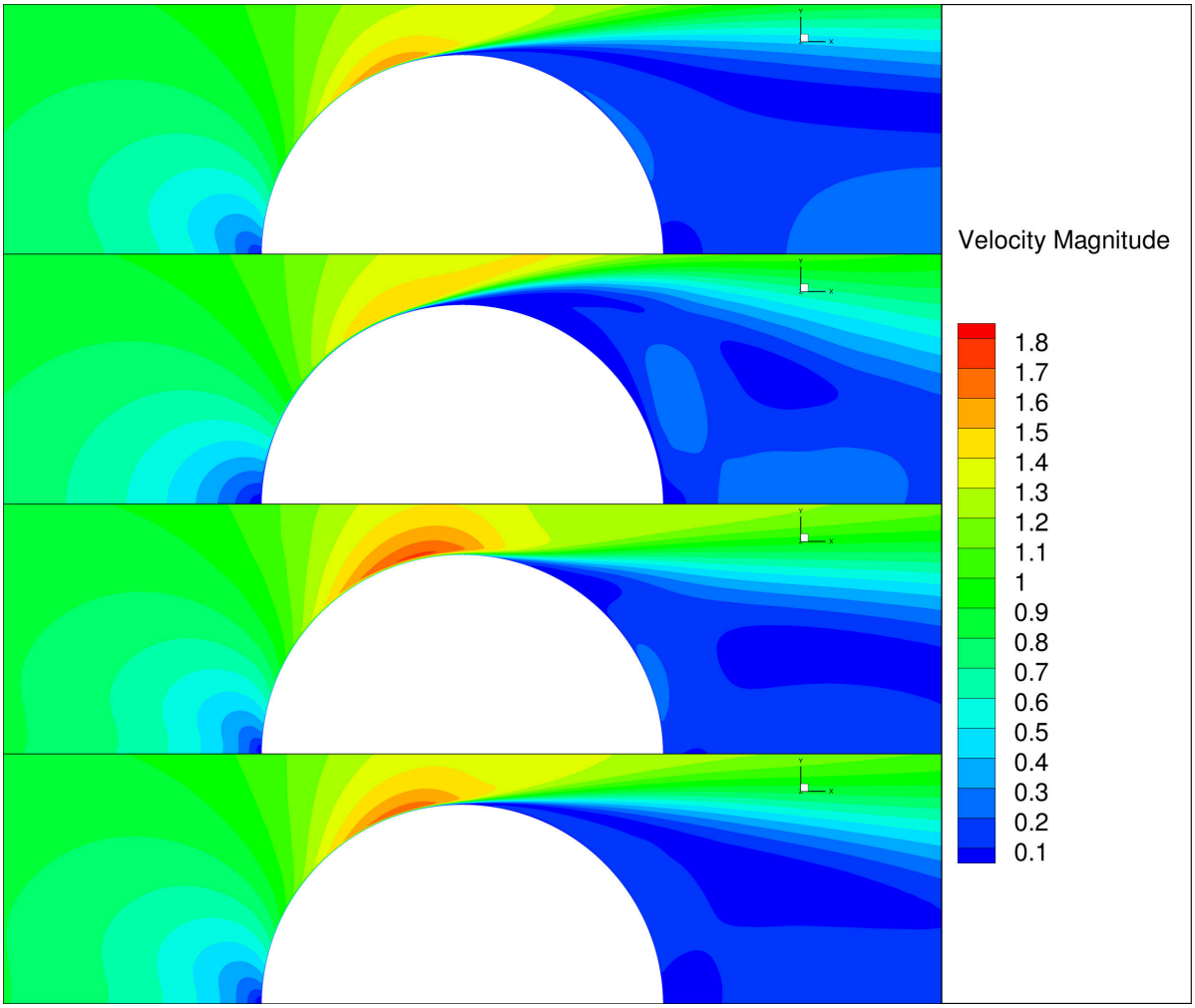


Figure 4.42: Visualisation of the separation point through the Velocity magnitude, Top Baseline RANS, second: LES, third: RANS cylinder correction model, Botom: RANS cube correction model.

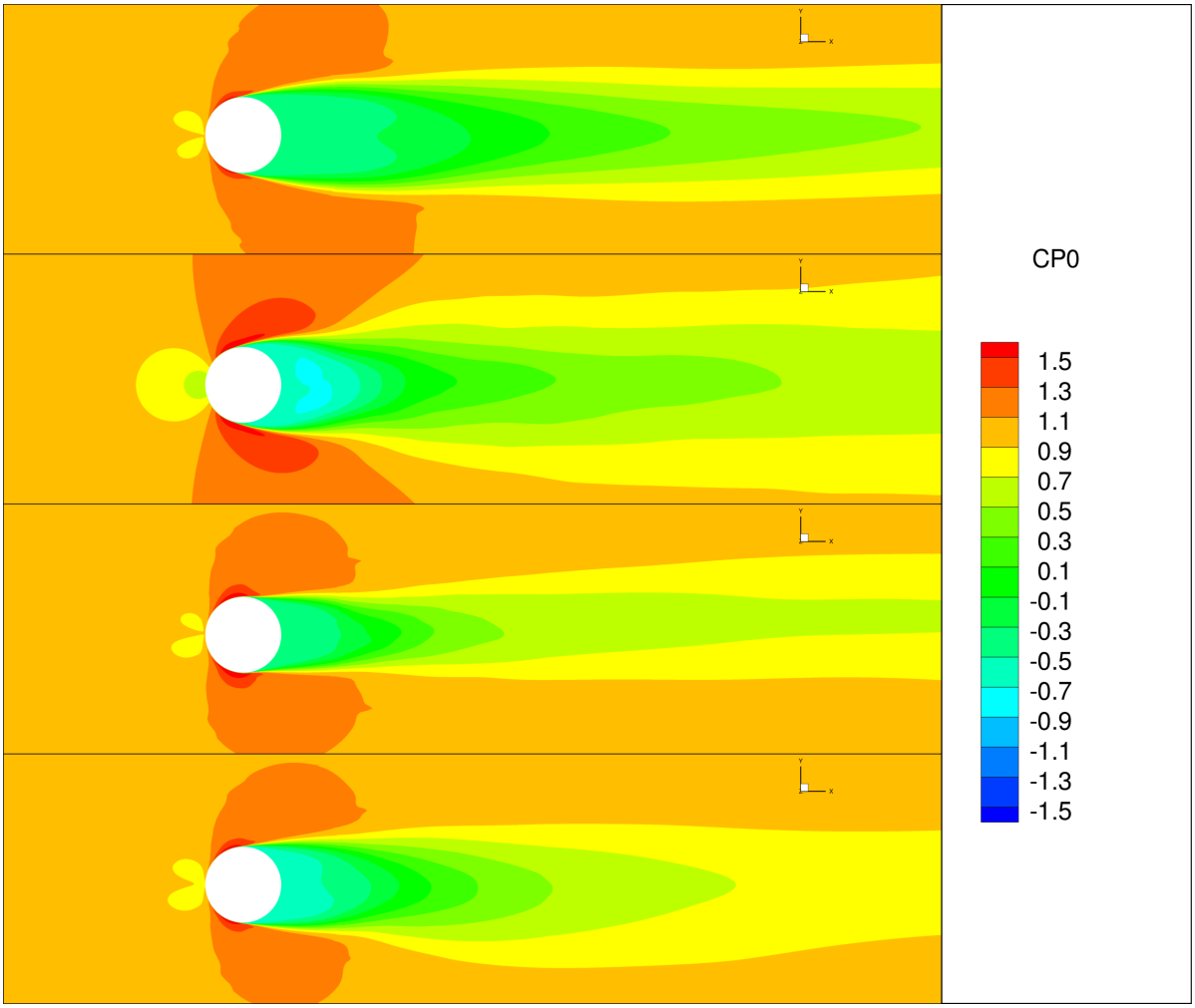


Figure 4.43: Total pressure coefficient. Top Baseline RANS, second: LES, third: RANS cylinder-correction model, Botom: RANS cube-correction model.

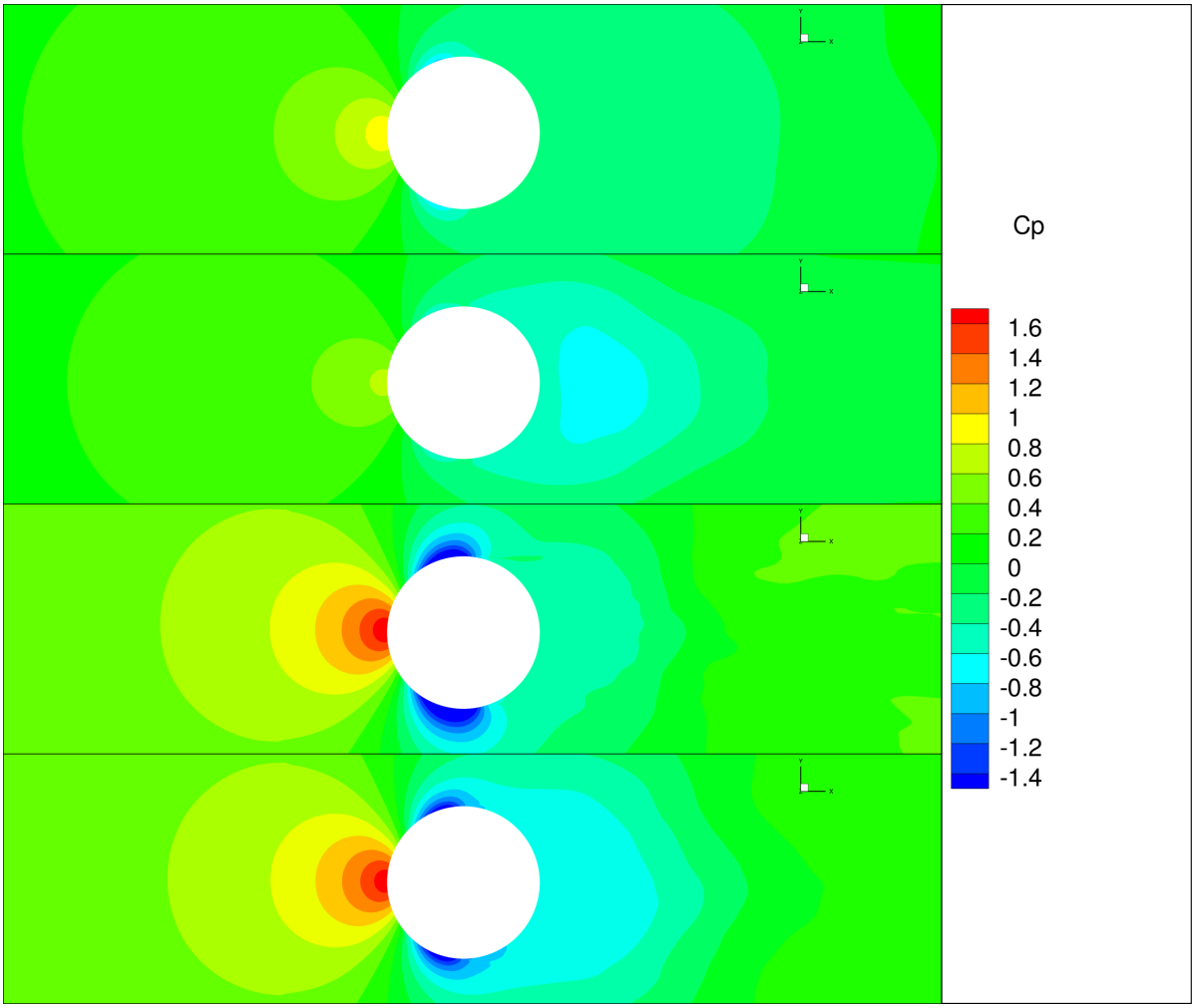


Figure 4.44: Pressure coefficient. Top Baseline RANS, second: LES, third: RANS cylinder-correction model, Botom: RANS cube-correction model.

4.3. Idealised wheel

The most complex flow case that has been investigated is that of the idealised rotating wheel. This flow case is an idealised version of an isolated wheel that is rotating on a road surface. The wheel has been modelled as a cylinder in contact with the ground. The boundary conditions simulate the rotation of the wheel and the ground plane. With the velocity set to 1 m/s and the rotational velocity of the wheel set to match this. The dynamic viscosity is set to $1e-6$ to get in combination with a wheel diameter of $1m$ a Reynolds number of 1e6. The domain has a length of $30m$ with a width and height of $10m$. The boundary conditions for the inlet are a fixed inlet velocity with for the LES synthetically generated turbulence.

4.3.1. Wheel high fidelity data source: LES

The LES that forms the basis for the frozen approach has been run on a 102 million cell grid for a wall resolved LES. The data from this simulation has been interpolated to a 50 million cell grid for the frozen approach and the further post processing. The subgrid-scale model that has been used for this is the WALE model. The solver that was used for this LES was not the OpenFOAM solver but the commercial solver Star-CCM+. The solver was run for a total of 450s simulated time to complete 15 flowthroughs.

The topology of the mesh can be found in figures A.12 and A.13. The mesh has been created using the snappyHexMesh Utility using refinement boxes and refinement layers for the surfaces. The minimum layer height for the wheel was set to

- 0.008 m for the ground wall.
- 0.001 m for the main tyre wall
- 0.002 m for the tyre side wall.
- 0.002 m for the tyre contact patch.

4.3.2. Baseline RANS

For the RANS simulation it was chosen to run the simulation on a much smaller of around 8 million cells. This was done to resemble a realistic scenario with a grid that might be used for such a RANS simulation. This has also been done to reduce the computation cost and it is a good opportunity to see how the correction model performs on a mesh that is a lot coarser than the mesh that has been used for the training data. This mesh does have the same topology and also still aims for a low y^+ value near the boundary layers.

4.3.3. Correction fields calculation: Frozen Approach

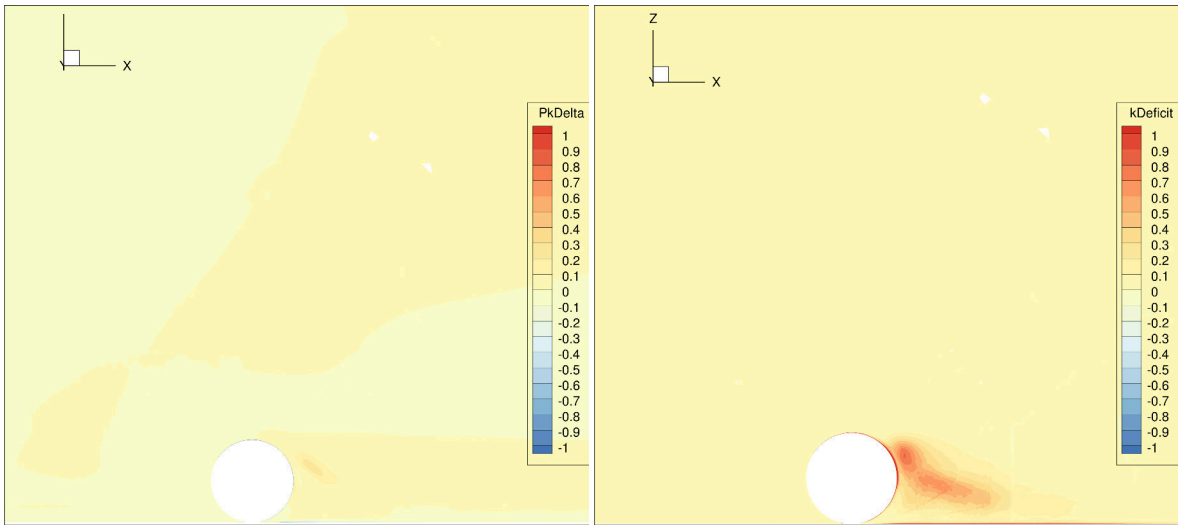
The frozen approach that was run till converged in about 600 iterations which was possible because the LES was statistically sufficiently converged. The corrective fields coming from this frozen approach are shown in Figure 4.45. In this figure it can be seen that the main correction happens close to the wheel in the boundary layer. This correction can be seen in more detail in Figure 4.46. This shows that the largest corrections are mainly focused on the boundary layer with a relatively small correction in the wake of the wheel as well. It shows again that the total correction is a reduction of the total turbulence production with the direct correction R being an addition the anisotropy correction must have mainly a negative effect on the turbulence production. This is in line with the findings from the previous two to test cases.

4.3.4. Correction model regression

The models that were found during the model discovery step are shown in Figure 4.47 for the anisotropy correction and in Figure 4.48 for the correction to the turbulent kinetic energy.

4.3.5. Model corrected RANS

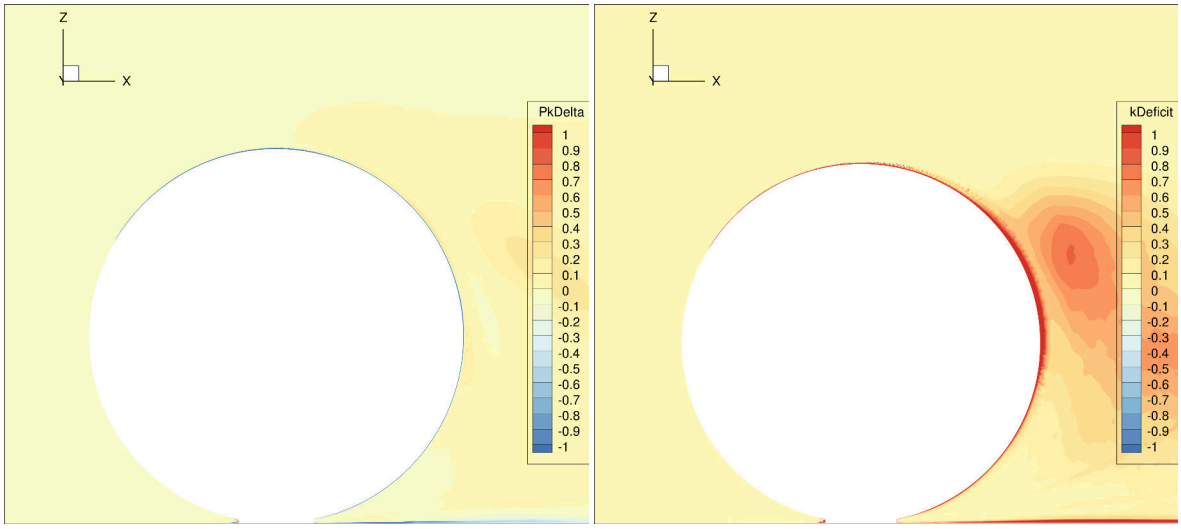
Due to the differences in meshes between the RANS and LES it was chosen not to do a correction propagation but to directly apply the correction through the modelled approach. The correction models that were selected are model number 12 for the anisotropy correction (4.7) and model number 343 for the R term (4.8). What stands out is that the magnitude of the coefficients for the basis functions that always return do not differ with different ridge coefficients.



(a) Total correction in turbulence production

(b) Direct correction in the turbulent kinetic energy. R, (3.40)

Figure 4.45: Wheel frozen approach correction fields.



(a) Detail of the total correction in turbulent kinetic energy

(b) Direct correction in the turbulent kinetic energy equation. R (3.40)

Figure 4.46: Wheel frozen approach correction fields detail.

1	I_1T_1	7	$I_1I_2T_4$	13	$I_1I_2T_3$	19	$I_1I_2^3T_1$	25	$I_1^2I_2^3T_4$	32	$I_1^2I_2^4T_3$
2	I_1T_3	8	$I_2^2T_1$	14	$I_1I_2^2T_2$	20	$I_1I_2^3T_2$	26	$I_1^3I_2^2T_4$	33	cT_1
3	I_1T_2	9	$I_2^2T_2$	15	$I_1I_2^2T_4$	21	$I_1I_2^3T_3$	27	$I_1^2I_2^3T_1$	34	cT_2
4	I_1T_4	10	$I_2^2T_4$	16	$I_1I_2^2T_1$	22	$I_1I_2^3T_4$	28	$I_1^2I_2^3T_3$	35	cT_3
5	I_2T_1	11	$I_1I_2T_1$	17	$I_2^2T_3$	23	$I_1^2I_2^2T_2$	29	$I_1^2I_2^4T_1$	36	cT_4
6	I_2T_4	12	$I_1I_2T_2$	18	$I_1^2I_2^2T_4$	24	$I_1^2I_2^3T_2$	30	$I_1^2I_2^4T_2$		

Table 4.7: Basis functions used for the correction model regression for the anisotropy correction b_{ij}^{Δ}

1	$I_1 T_1$	9	$I_2^2 T_2$	17	$I_1 I_2^2 T_4$	25	$I_1 I_2^3 T_1$	33	$I_1^2 I_2^2 T_2$	41	$I_1 I_2^4 T_2$
2	$I_1 T_2$	10	$I_2^2 T_4$	18	$I_2^3 T_1$	26	$I_1 I_2^3 T_2$	34	$I_1 I_2^4 T_1$	42	$I_1 I_2^4 T_3$
3	$I_1 T_3$	11	$I_1^2 T_1$	19	$I_2^3 T_2$	27	$I_1 I_2^3 T_3$	35	$I_1 I_2^4 T_4$	43	$c T_1$
4	$I_1 T_4$	12	$I_1 I_2 T_2$	20	$I_2^3 T_3$	28	$I_1 I_2^3 T_4$	36	$I_1^3 I_2^2 T_4$	44	$c T_2$
5	$I_2 T_1$	13	$I_1^2 T_3$	21	$I_2^3 T_4$	29	$I_2^4 T_1$	37	$I_1^2 I_2^3 T_1$	45	$c T_3$
6	$I_2 T_4$	14	$I_1 I_2 T_3$	22	$I_1 I_2^2 T_1$	30	$I_2^4 T_2$	38	$I_1^2 I_2^3 T_2$	46	$c T_4$
7	$I_1 I_2 T_4$	15	$I_1 I_2 T_1$	23	$I_2^2 T_3$	31	$I_2^4 T_3$	39	$I_1^2 I_2^3 T_3$		
8	$I_2^2 T_1$	16	$I_1 I_2^2 T_2$	24	$I_1^2 I_2^2 T_4$	32	$I_2^4 T_4$	40	$I_1^2 I_2^3 T_4$		

Table 4.8: Basis functions used for the regression of the correction model for R for the idealised wheel

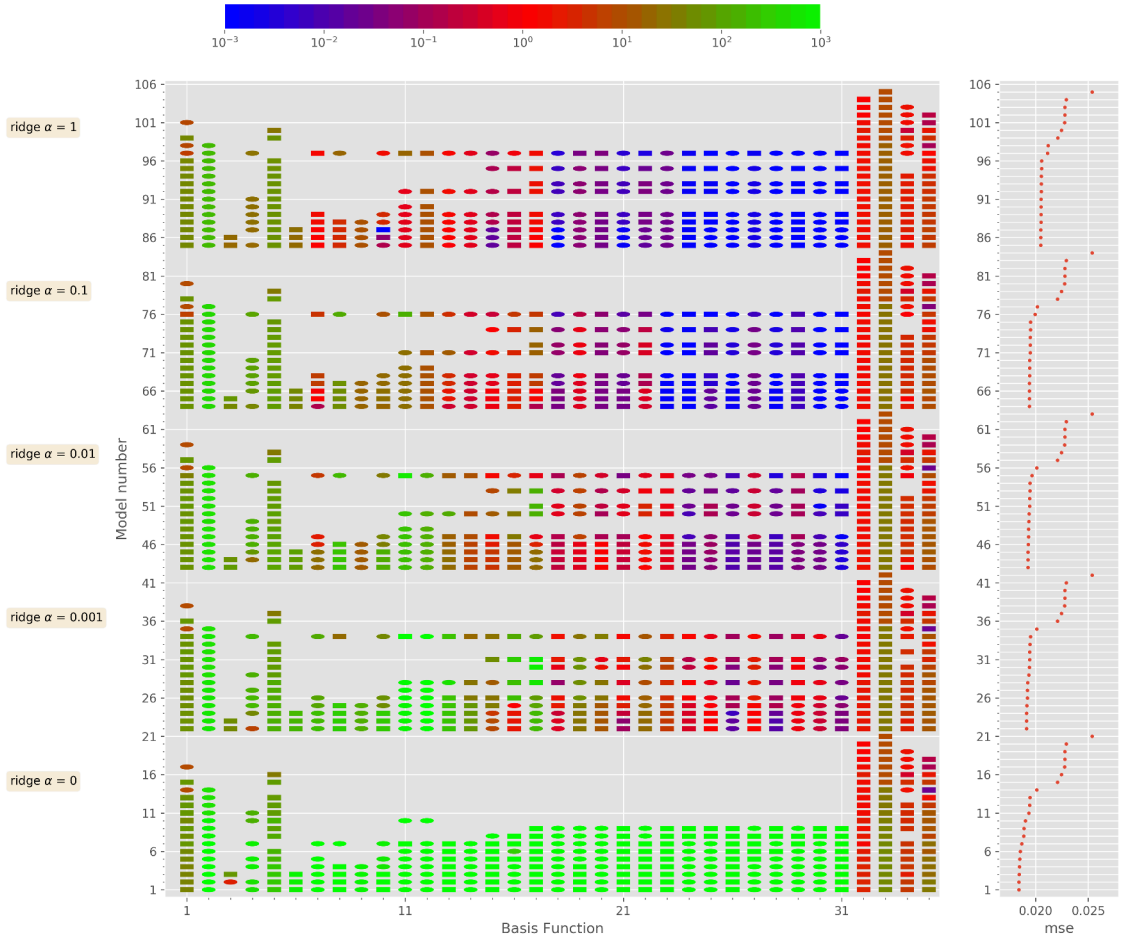


Figure 4.47: Overview of the correction models for the anisotropy correction b_{ij}^{Δ}

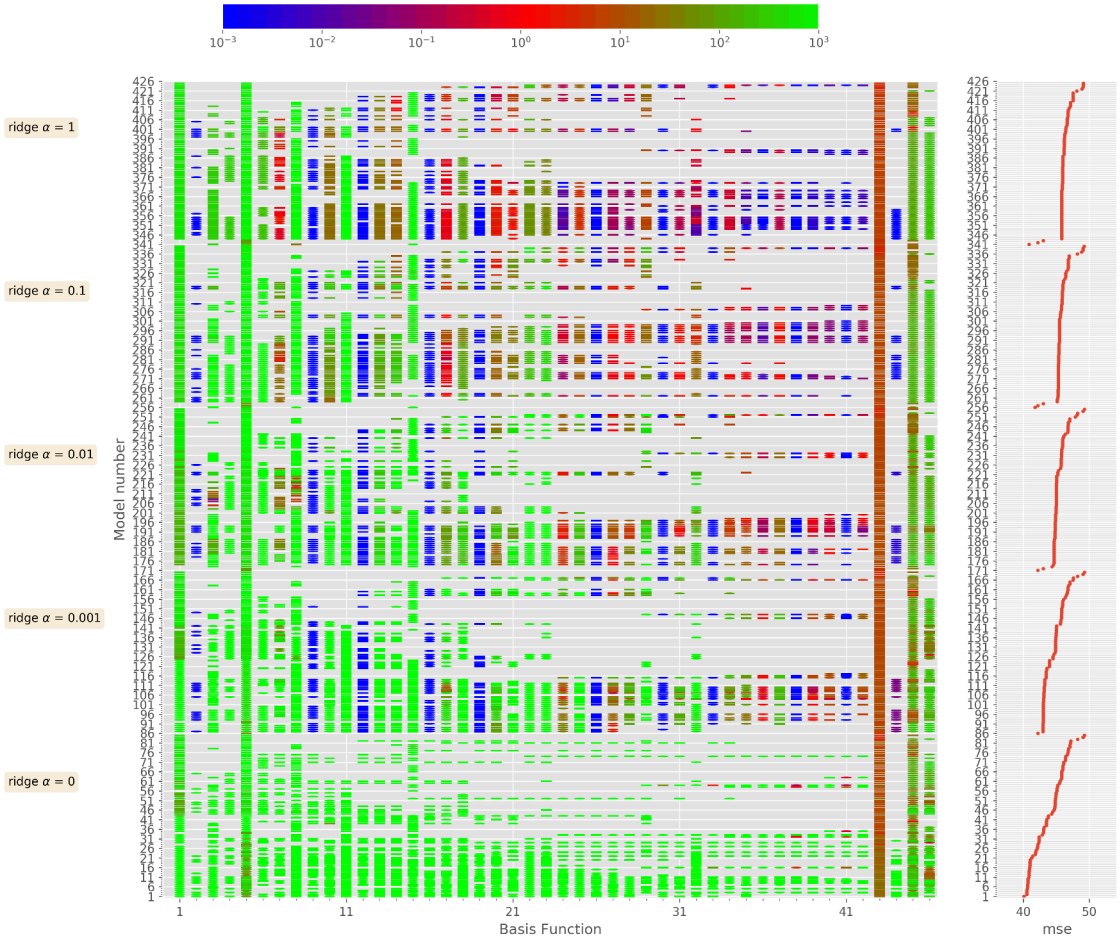


Figure 4.48: Overview of the correction models for the anisotropy correction R

To also get an idea about the extrapolating qualities of this approach these correction models are also compared again to the model that was trained on the wall mounted cube case and was also successfully applied to cylinder case. The equations for this correction model can be found in equations (4.1) and (4.2).

$$\begin{aligned} b_{ij}^{\Delta} = T^{(1)}(58.53851 I_1 + 67.69593 I_2 + 1.34969) \\ + T^{(2)}(26.56142 - 430.87047 I_1) \\ + 3.3792 T^{(3)} + 4.89236 T^{(4)} \end{aligned} \tag{4.7}$$

$$R = T^{(1)}(68.57017 I_2 + 3.95008) + 30.62762 T^{(3)} \tag{4.8}$$

In Figure 4.49 the velocity magnitude is plotted for the different simulations. What can be seen here is that for the baseline RANS a massive separation area is predicted on top of the wheel, whilst for the LES the flow remains attached resulting in an area of high velocity flow directly behind the wheel. With the wheel correction model this separation area is reduced significantly with also the area of low velocity magnitude behind the wheel moving towards the LES solution. The Cube correction model further decreases the separation area on top of the wheel. However, with both correction models there are still some differences in the velocity magnitude on top of the wheel and directly behind the wheel near the ground, where the velocity magnitude is lower than in the LES solution.

Looking at the x-component of the velocity at different sections in front of and behind the wheel in Figure 4.50 confirms this improvement, with both correction models moving the solution closer to the LES solution and both showing a very similar result. When a comparison of the pressure coefficient is made for the different simulations in Figure 4.51 it can be seen that both models show a significant improvement over the baseline model, but with again a close match between for the Wheel correction model and the Cube correction.

Looking at the wake of the wheel in Figure 4.52 shows that the shape of the wake is improved by the use of both correction models, however it looks like when looking at the bulk length behind the wheel, the cube trained model performs better. The only downside to this model is the strange double wave on the sides of the wheel that does not correspond to the LES solution.

Finally when comparing the different simulations near the ground plane in Figure 4.53 the big differences between the Baseline and the LES become very obvious with the biggest differences in the size of the wake and the direction of the flow behind the wheel. When looking at the streamtraces from the LES solution one can see that there is a small portion of where there is some backflow, this is just in front of the low velocity area that marks the point where the two vortices originating from the interactions between the low pressure on the sides and with the high pressure on the frontal area come together. It can be seen that this portion of backflow is not visible in both the baseline RANS and the RANS with the wheel correction model. What also stands out is that none of the RANS simulations seem to capture the outboard movement of the horseshoe vortices as the flow moves further downstream. This missing prediction was also seen in the wall mounted cube case, where the vortices travel downstream in almost straight lines.

In conclusion it possible to state that both correction models improve the result of the RANS in comparison to the LES simulation. Both models do not manage to exactly recreate the LES solution, with differences showing in the magnitude and direction of the wheel wake being the biggest differences still visible between the two. The model that improves the solution the most is that found on the wall mounted cube case in Section 4.1. This model also improved the stability of the solution the most leading to faster convergence.

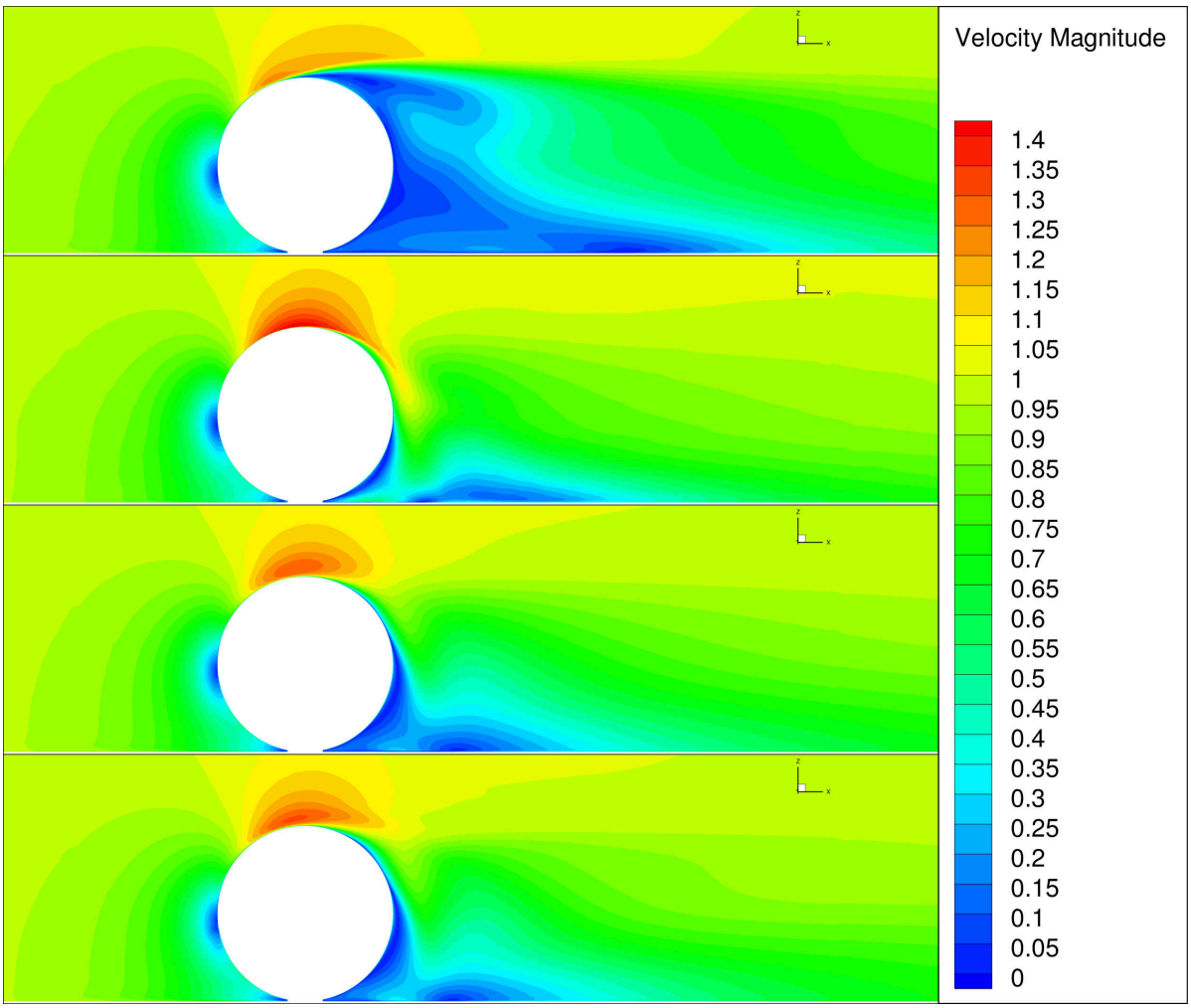


Figure 4.49: Comparing the Velocity magnitude at the centre of the wheel. Top: Baseline RANS, 2nd: LES, 3rd: Wheel correction model, bottom: Cube correction model.

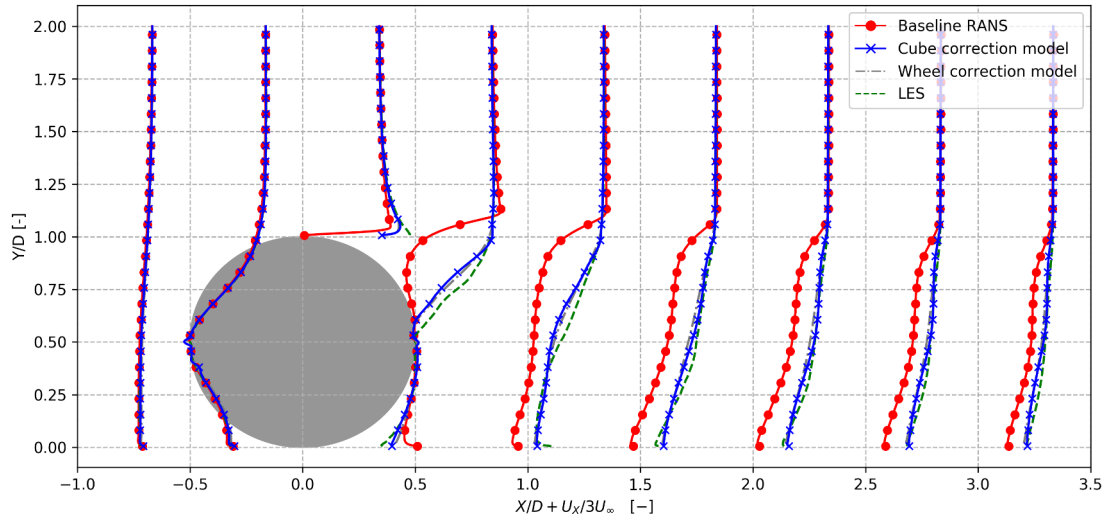


Figure 4.50: X-component velocity profile at the centre of the wheel.

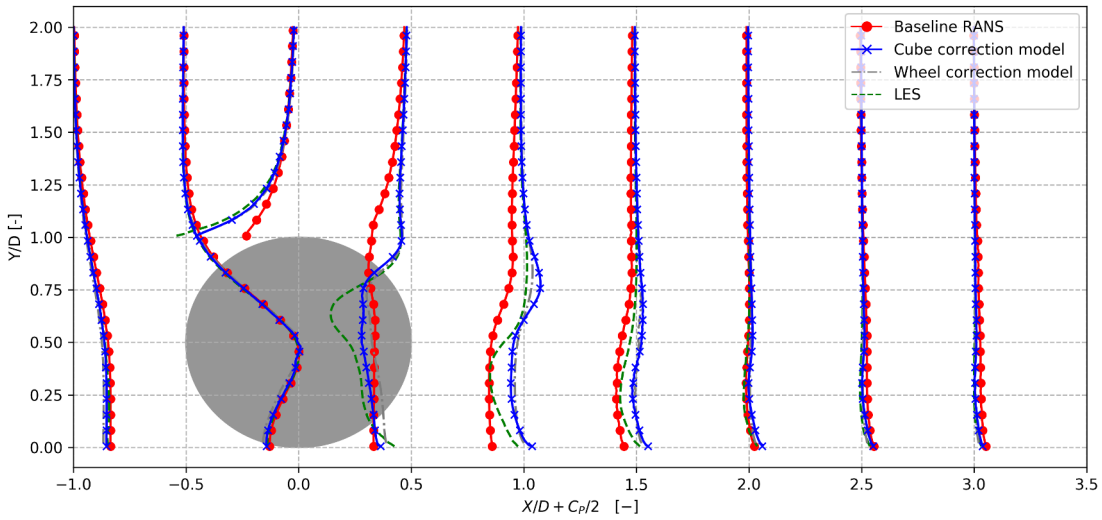


Figure 4.51: Pressure Coefficient profile at the centre of the Wheel.

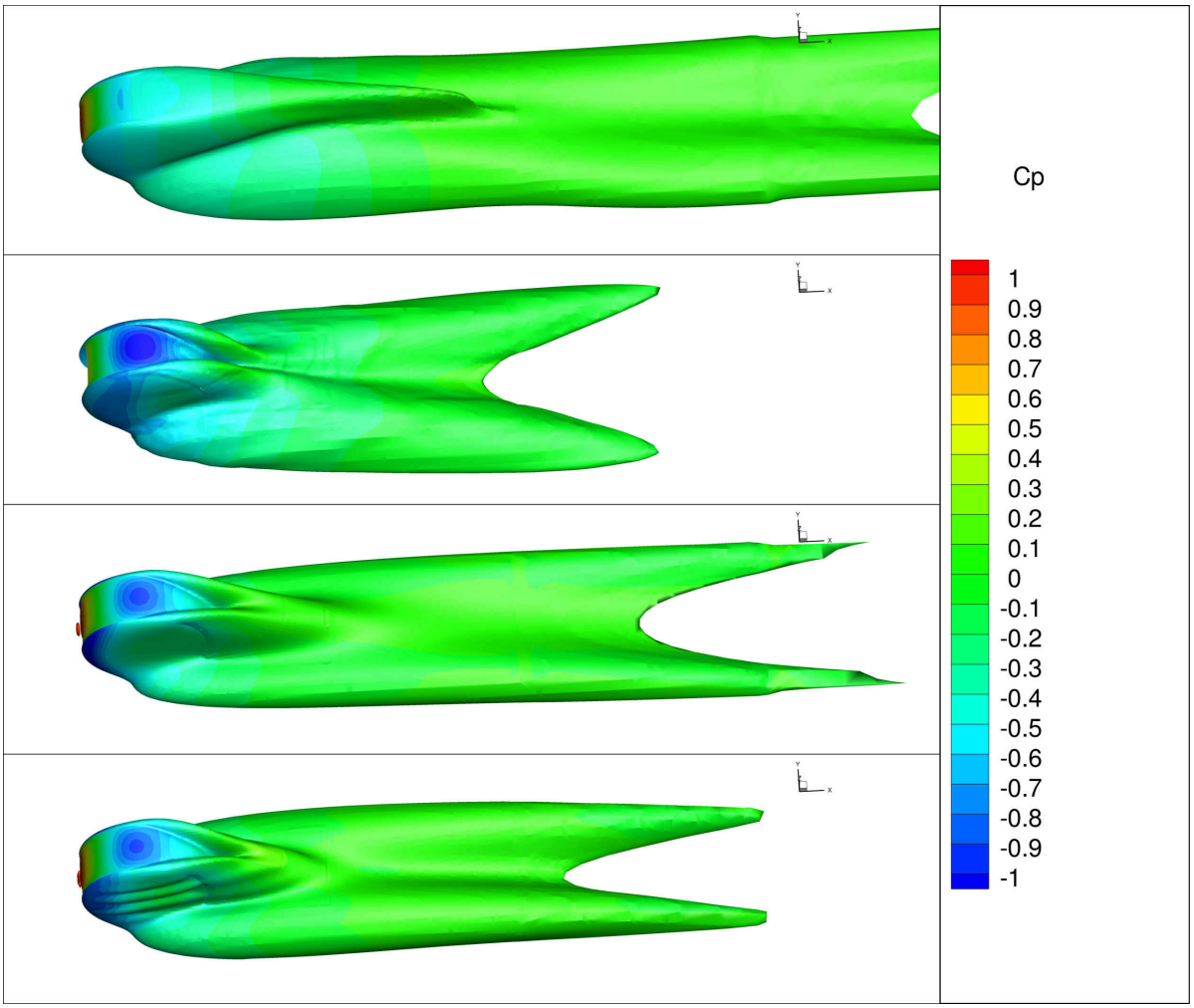


Figure 4.52: Visualisation of the wheel wake through isosurface of the total pressure, $C_{p0} = 0.5$, coloured by the pressure coefficient. Top: Baseline RANS, 2nd: LES, 3rd: Wheel correction model, bottom: Cube correction model

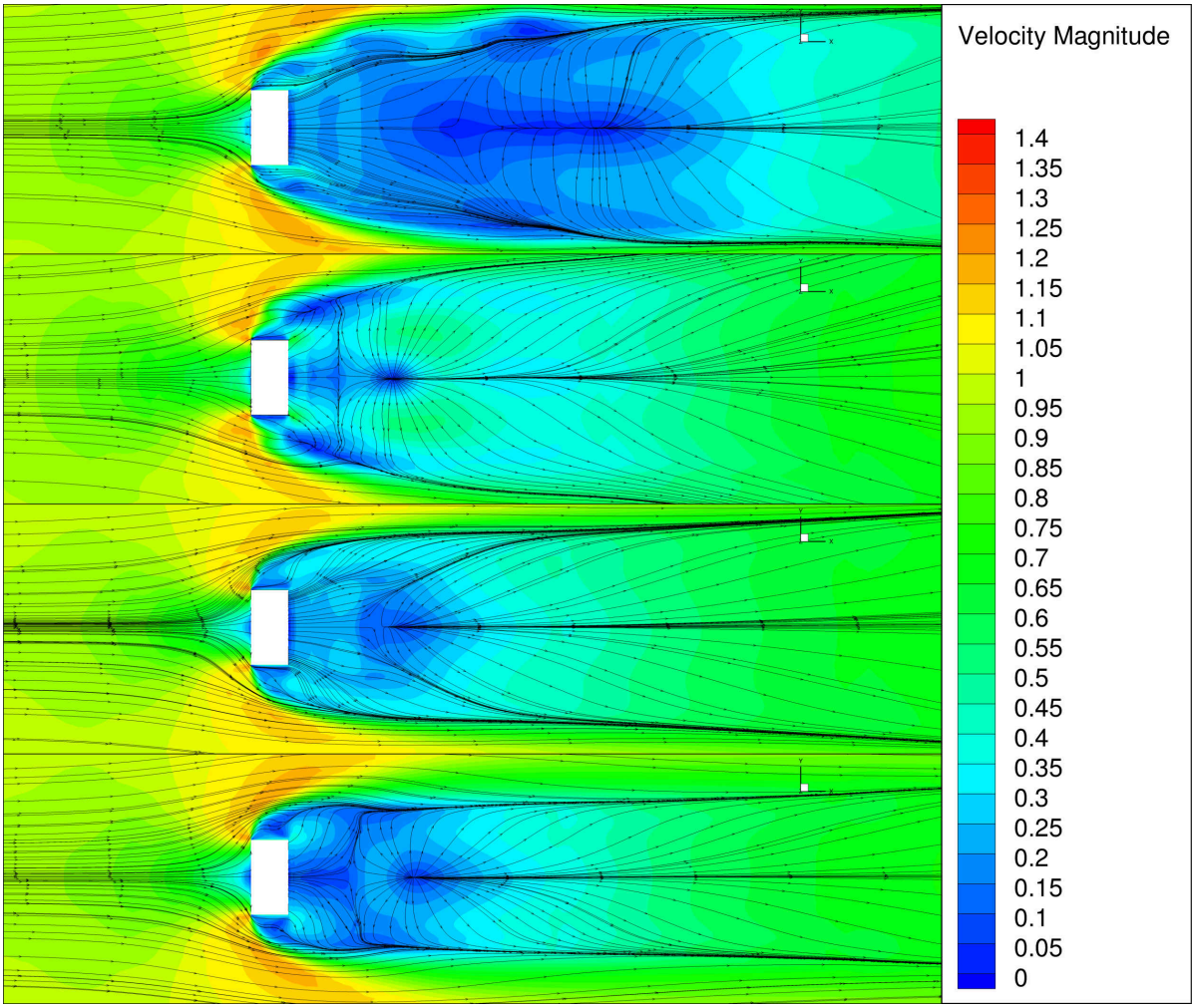


Figure 4.53: Streamtraces on the bottom of the domain combined with the velocity magnitude. Top: Baseline RANS, 2nd: LES, 3rd: Wheel correction model, bottom: Cube correction model

5

Conclusion

From the results presented in Chapter 4 a number of conclusions can be drawn. The main question that was to be answered is if the accuracy of RANS solvers could be increased by augmenting existing turbulence models with machine learned models. From literature this had already been shown for 2D flow cases and with the results from the three test cases it has been shown that this is also the case for 3D flows. With the correction models improving the solution for all three cases compared to the standard $k-\omega-SST$ model.

Different training geometries resulted in different correction models with some models performing better on non flows they were not trained on. This is in line with the results found by Schmelzer and Dwight [18] who also found that models that were trained using a different geometry outperformed models trained on this geometry.

The ansatz that was made in the SpaRTA methodology that only the first 4 tensors and 2 invariants from Pope's tensor basis are needed to get to be able to model the anisotropy. When looking at the results of the surface mounted cube and comparing the results of the correction models for both the full tensors basis and the reduced tensor basis, the results are similar enough to be able to conclude that in this application of the reduction of the basis is justified.

From all three simulations that have been done, experience has shown that two correction fields for which models are found both have different effects on the stability of the solution. All three simulations seemed to benefit in terms of stability with the addition of the R-correction term. On the contrary with the addition of the anisotropy correction the stability of the solution decreased making it harder for the simulation to converge.

The finding of this instability due to the addition of the anisotropy correction led to an investigation of what happens when more anisotropy is gradually added. It is shown that the more anisotropy correction is added the more the solution moves towards the LES reference. It was also shown that when the solution is then averaged over the last couple of hundredths of iterations the average of this unstable solution might improve further towards the LES solution. In this process of adding the anisotopy correction it was also shown that the largest correction effect comes from the R correction term. With the solution generally already improving significantly with just this correction added.

Finally one of the most important conclusions that can be drawn is that the correction models can be applied to different geometries and have a positive effect on the final result. This shows the extrapolating qualities of the correction models. One might actually say that with the correction models found on the cube a new turbulence model has been discovered. As it performed better on all three tested geometries than the baseline and has shown a good stability to be able to generally apply it to all sorts of problems. However, more testing is needed on more different flow cases to prove the general applicability of the model.

A decisive conclusion cannot be drawn yet if there is an increase or reduction in computational cost. This is because the computational cost depends heavily on the stability and convergence rate of the solution.

This stability depends on a couple of things: the geometry/flow itself, the correction model that is being used and the amount of anisotropy correction that is being applied. This means that with a relatively stable correction model and a small amount of anisotropy correction the stability of the solution and the converge rate is increased leading to a reduction in computational cost compared to the baseline turbulence model. However, when more anisotropy is added or a correction model is used that is less stable the convergence rate of the solution is reduced resulting in a higher computational cost. All in all the computational cost depends mainly on the choices the user makes.

The areas that see the largest improvement overall are the separation areas, the point of separation is slightly adjusted in most cases and the resulting separated regions and re-circulation areas are reduced in size. what can be concluded as well is that the areas that are already accurate in comparison to the high fidelity data are not worsened with the application of the correction model. The biggest changes in the solution seem to be visible in the velocity magnitude and the total pressure and not so much in the pressure directly. With the instability of the solution it seems that the resulting pressure is becoming more inaccurate.

6

Recommendations for future research

From this research it is possible to draw multiple conclusions and a new avenue for further research is opened up. The conclusions that can be drawn from this research are generally positive, however there is still good number of questions that need to be answered. The results also leads to new questions and insights for alterations to the method that might improve the final result.

From the research it was shown that the model that was found through the SpaRTA methodology on the surface mounted cube case performed well on all three test cases. It would be interesting to know what the limitation of this model are and if the improvement is also seen on completely different flows and more complex cases with multiple flow device interactions.

This research has mainly focused on bluff body flows with large separation areas, for further research it would be interesting to see the performance of these models on flows without separations such as the flow over a wing at low angles of attack.

For the method itself it would be interesting to investigate the extension of the library of basis function with other non-dimensional quantities such as used in the research from Kaandorp and Dwight [6]. They extended the library of basis functions formed from the tensors and invariants from Pope's viscosity hypothesis with additional variables such as the non-dimensional wall-distance. It is important that variables are selected that are non dimensional to keep the Galilean invariance of the method.

The non-dimensional wall distance is specifically interesting to include in the library as it was seen that both correction fields were generally largest near the walls. Having a form of the wall-distance in the library of candidate functions might allow the regression to more easily adapt for these large corrections near the walls while maintaining the smaller corrections further away from the wall. That being said in the propagated approach it was mainly due to the large corrections near the wall that the solution started to become unstable. For the regression it might be a good idea to explicitely filter out the first cell for the regression of the anisotropy correction model. As this first cell is dominated by boundary conditions anyway and with a wall model in the wall modelled approach a correction should not have to be necessary.

One of the other things that requires further investigation is the effect of the convergence of the statistics of the high fidelity data-set. In Section 4.1.5 the anisotropy correction field is shown together with the model prediction. The anisotropy correction field still looks quite rough with model predictions that do not match the target values very well. This mismatch could be due to two reasons, the first is that there is not enough correlation between the inputs and the correction field. The second is that the Reynolds stress calculated in the high fidelity method is not converged enough. Trying to make a prediction of this unconverged field could be very difficult. To test this it would be required to let the LES or DES run for much longer to collect more converged statistics.

Something else that needs to be investigated is the method of sub-sampling. The method that has been used in this research is straightforward and fast but unsophisticated. There must be smarter ways of sampling the data for the model discovery steps. What is important in this is that the balance should not be lost between

areas where very little or no correction is needed and areas where the largest corrections are needed. This means that it is not possible to just ignore the areas where no correction is needed as that will lead to skewed models that will also correct in regions where no correction is required. There exist clever algorithms that are able to determine which cells to filter out without throwing away too much information, this requires more testing and is definitely needed if more large problems are to be used for training.

Another route that requires investigation is that of combining data-sets. The ultimate goal is to find a correction model that is generally applicable. For now the research has focused on single cases where it appeared that the correction models are also applicable to different cases. It would be interesting to see if this general applicability is increased when multiple problems are combined in the model discovery step. This should increase the chance of correcting more errors in the baseline model. There are a couple of things that will require some thought and experiments if this method of combining data-sets is to be used. Something that should be taken into account is the size of the data-set, how much data is used from one set and how much from the other? Can you take data-sets with completely different Reynolds numbers or should they be similar?

Finally for further investigation it might be beneficial to further optimise the elastic net algorithm. The elastic net module from scikit learn that is being used now is not fully optimised for parallel computing. For optimal performance it would be recommended to write a bespoke code which is properly optimised for parallel running on multiple machines. Switching the elastic net regression for a support vector machine with l_1 and l_2 regularization might also open up the opportunity to include GPU processing in this step as is shown to be possible by Zhou, Chen, Song and Gardner [30].

Bibliography

[1] Matheus A. Cruz, Roney L. Thompson, Luiz E.B. Sampaio, and Raphael D.A. Bacchi. The use of the reynolds force vector in a physics informed machine learning approach for predictive turbulence modeling. *Computers & Fluids*, 192:104258, 2019. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2019.104258.

[2] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence Modeling in the Age of Data. *Annual Review of Fluid Mechanics*, 51(1):357–377, Jan 2019. doi: 10.1146/annurev-fluid-010518-040547.

[3] W.N. Edeling, P. Cinnella, and R.P. Dwight. Predictive rans simulations via bayesian model-scenario averaging. *Journal of Computational Physics*, 275:65–91, 2014. doi: 10.1016/j.jcp.2014.06.052.

[4] M. R. Emory and G. Iaccarino. Visualizing turbulence anisotropy in the spatial domain with componentality contours. In *Center for Turbulence Research Anual Research Briefs 2014*, 2014.

[5] Nicoud F. and F. Ducros. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow Turbulence and Combustion*, 62:183–200, 09 1999. doi: 10.1023/A:1009995426001.

[6] M. L. A. Kaandorp and R. P. Dwight. Stochastic Random Forests with Invariance for RANS Turbulence Modelling. *arXiv e-prints*, art. arXiv:1810.08794, Oct 2018.

[7] J.R. Koza, F.H. Bennett, D. Andre, and M.A. Keane. *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*, pages 151–170. Springer Netherlands, Dordrecht, 1996. ISBN 978-94-009-0279-4. doi: 10.1007/978-94-009-0279-4_9.

[8] Spalding D.B. Launder B.E. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269 – 289, 1974. ISSN 0045-7825. doi: 10.1016/0045-7825(74)90029-2.

[9] Z. Li, J. B. Hoagg, A. Martin, and S.C.C. Bailey. Retrospective cost adaptive reynolds-averaged navier–stokes k–ω model for data-driven unsteady turbulent simulations. *Journal of Computational Physics*, 357:353 – 374, 2018. ISSN 0021-9991. doi: 10.1016/j.jcp.2017.11.037. URL `http://www.sciencedirect.com/science/article/pii/S0021999117308756`.

[10] J. Ling and J. Templeton. Evaluation of machine learning algorithms for prediction of regions of high reynolds averaged navier stokes uncertainty. *Physics of Fluids*, 27(8):085103, 2015. doi: 10.1063/1.4927765.

[11] J. Ling, A. Kurzawski, and J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016. doi: 10.1017/jfm.2016.615.

[12] John L. Lumley and Gary R. Newman. The return to isotropy of homogeneous turbulence. *Journal of Fluid Mechanics*, 82(1):161–178, 1977. doi: 10.1017/S0022112077000585.

[13] R. Martinuzzi and C. Tropea. The Flow Around Surface-Mounted, Prismatic Obstacles Placed in a Fully Developed Channel Flow (Data Bank Contribution). *Journal of Fluids Engineering*, 115(1):85–92, 03 1993. ISSN 0098-2202. doi: 10.1115/1.2910118.

[14] F. Menter. Zonal two equation k-w turbulence models for aerodynamic flows. In *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*, January 1993. doi: 10.2514/6.1993-2906.

[15] Craft T. Poletto R. and Revell A. A. A new divergence free synthetic eddy method for the reproduction of inlet flow conditions for les. *Flow Turbulence Combust*, 91:519–539, 2013. doi: 10.1007/s10494-013-9488-2.

[16] S. B. Pope. A more general effective-viscosity hypothesis. *Journal of Fluid Mechanics*, 72(2):331–340, 1975. doi: 10.1017/S0022112075003382.

[17] J. Ray, L. Dechant, S. Lefantzi, J. Ling, and S. Arunajatesan. Robust bayesian calibration of a $k-\epsilon$ model for compressible jet-in-crossflow simulations. *AIAA Journal*, 56(12):4893–4909, 2018. doi: 10.2514/1.J057204.

[18] M. Schmelzer, R.P. Dwight, and P. Cinnella. Machine Learning of Algebraic Stress Models using Deterministic Symbolic Regression. *arXiv e-prints*, art. arXiv:1905.07510, May 2019.

[19] F. G. Schmitt. About boussinesq's turbulent viscosity hypothesis: historical remarks and a direct evaluation of its validity. *Comptes Rendus Mécanique*, 335(9):617 – 627, 2007. ISSN 1631-0721. doi: https://doi.org/10.1016/j.crme.2007.08.004. URL `http://www.sciencedirect.com/science/article/pii/S1631072107001386`. Joseph Boussinesq, a Scientist of bygone days and present times.

[20] A. P. Singh, K. Duraisamy, and S. Pan. Augmentation of turbulence models using field inversion and machine learning. In *55th AIAA Aerospace Sciences Meeting*, 2017. doi: 10.2514/6.2017-0993.

[21] T. Sjögren and A. V. Johansson. Development and calibration of algebraic nonlinear models for terms in the reynolds stress transport equations. *Physics of Fluids*, 12(6):1554–1572, 2000.

[22] J. Smagorinsky. General circulation experiments with the primitive equations: I. The basic experiment*. *Monthly Weather Review*, 91(3):99–164, 03 1963. ISSN 0027-0644. doi: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[23] Roney L. Thompson, Luiz Eduardo B. Sampaio, Felipe A.V. de Bragança Alves, Laurent Thais, and Gilmar Mompean. A methodology to evaluate statistical errors in dns data of plane channel flows. *Computers & Fluids*, 130:1 – 7, 2016. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2016.01.014.

[24] J. Wang, J. Wu, J. Ling, G. Iaccarino, and H. Xiao. A Comprehensive Physics-Informed Machine Learning Framework for Predictive Turbulence Modeling. *arXiv e-prints*, art. arXiv:1701.07102, January 2017.

[25] J. Wang, J. Wu, and H. Xiao. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Physical Review Fluids*, 2(3):034603, March 2017. doi: 10.1103/PhysRevFluids.2.034603.

[26] J. Weatheritt and R.D. Sandberg. The development of algebraic stress models using a novel evolutionary algorithm. *International Journal of Heat and Fluid Flow*, 68:298 – 318, 2017. ISSN 0142-727X. doi: https://doi.org/10.1016/j.ijheatfluidflow.2017.09.017.

[27] J. Weatheritt and R.D. Sandberg. The development of algebraic stress models using a novel evolutionary algorithm. *International Journal of Heat and Fluid Flow*, 68:298 – 318, 2017. ISSN 0142-727X. doi: 10.1016/j.ijheatfluidflow.2017.09.017.

[28] D.C. Wilcox. Formulation of the k-w turbulence model revisited. *AIAA Journal*, 46(11):2823–2838, 2008. doi: 10.2514/1.36541. URL `https://doi.org/10.2514/1.36541`.

[29] Jinlong Wu, Rui Sun, Sylvain Laizet, and Heng Xiao. Representation of Reynolds Stress Perturbations with Application in Machine-Learning-Assisted Turbulence Modeling. *arXiv e-prints*, art. arXiv:1709.05683, Sep 2017.

[30] Q. Zhou, W. Chen, S. Song, J. R. Gardner, K. Q. Weinberger, and Y. Chen. A Reduction of the Elastic Net to Support Vector Machines with an Application to GPU Computing. *arXiv e-prints*, art. arXiv:1409.1976, September 2014.

A

Additional Figures

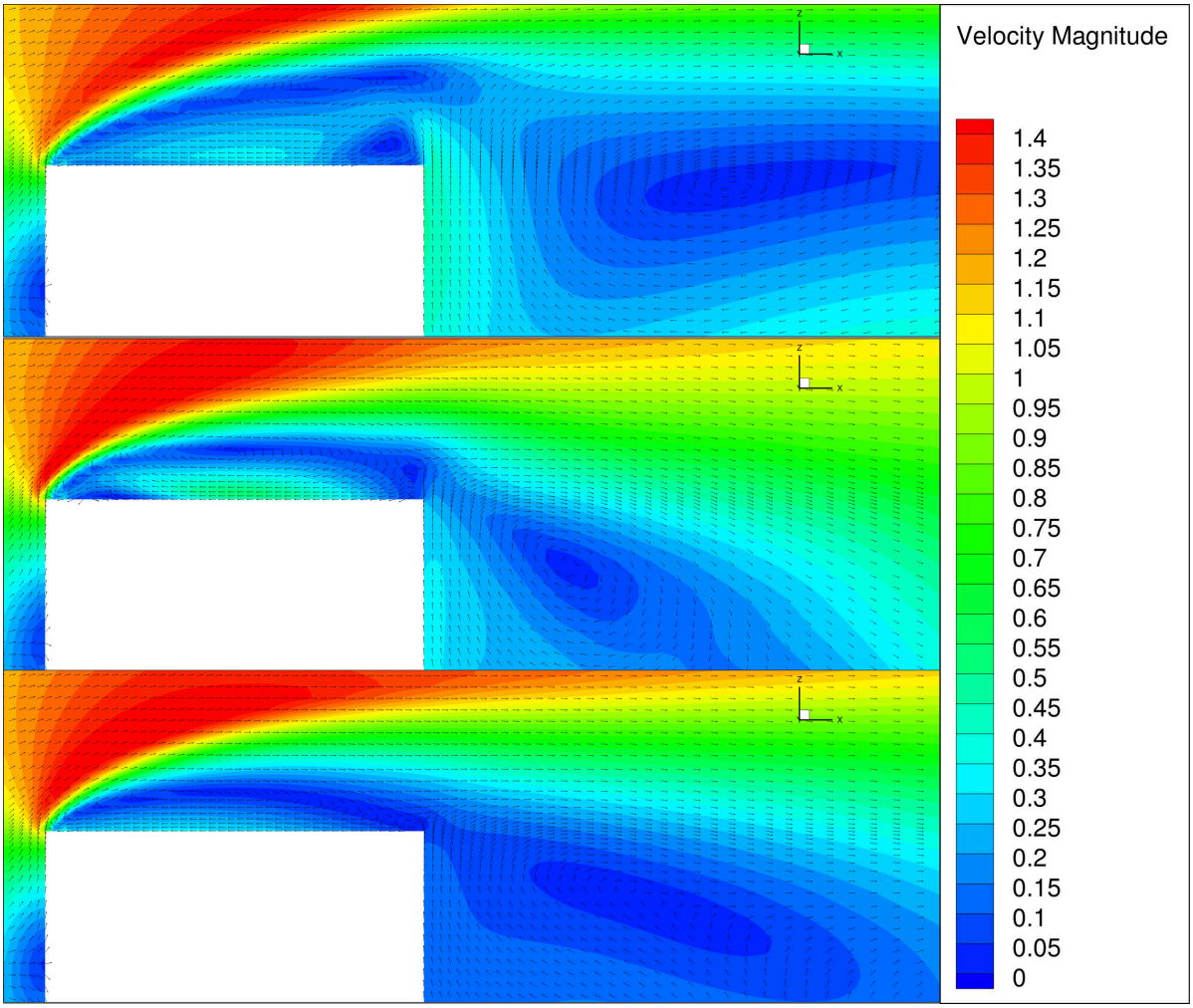


Figure A.1: Wall mounted cube: Comparing the recirculation area on top of the cube for the baseline RANS and correction model to the DES reference. Top: Baseline RANS, Centre: DES, Bottom: RANS 100%R, 50%b_{ij}^{Δ} correction applied. Model equations (4.1)(4.2)

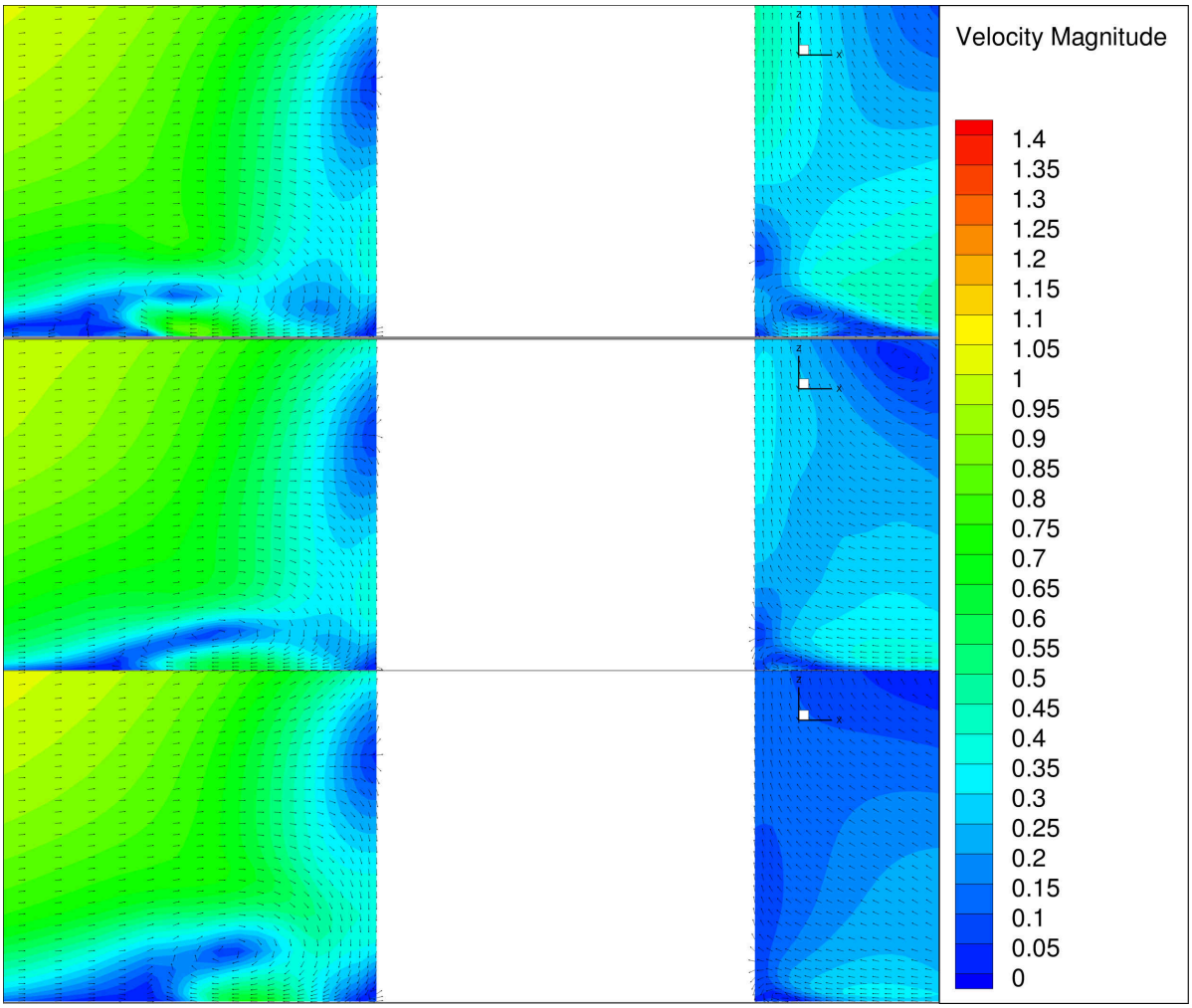


Figure A.2: Wall mounted cube: Comparing the horseshoe vortex location in front of the cube for the baseline RANS and correction model to the DES reference. Top: Baseline RANS, Centre: DES, Bottom: RANS $100\%R, 50\%b_{ij}^{\Delta}$ correction applied. Model equations (4.1)(4.2)

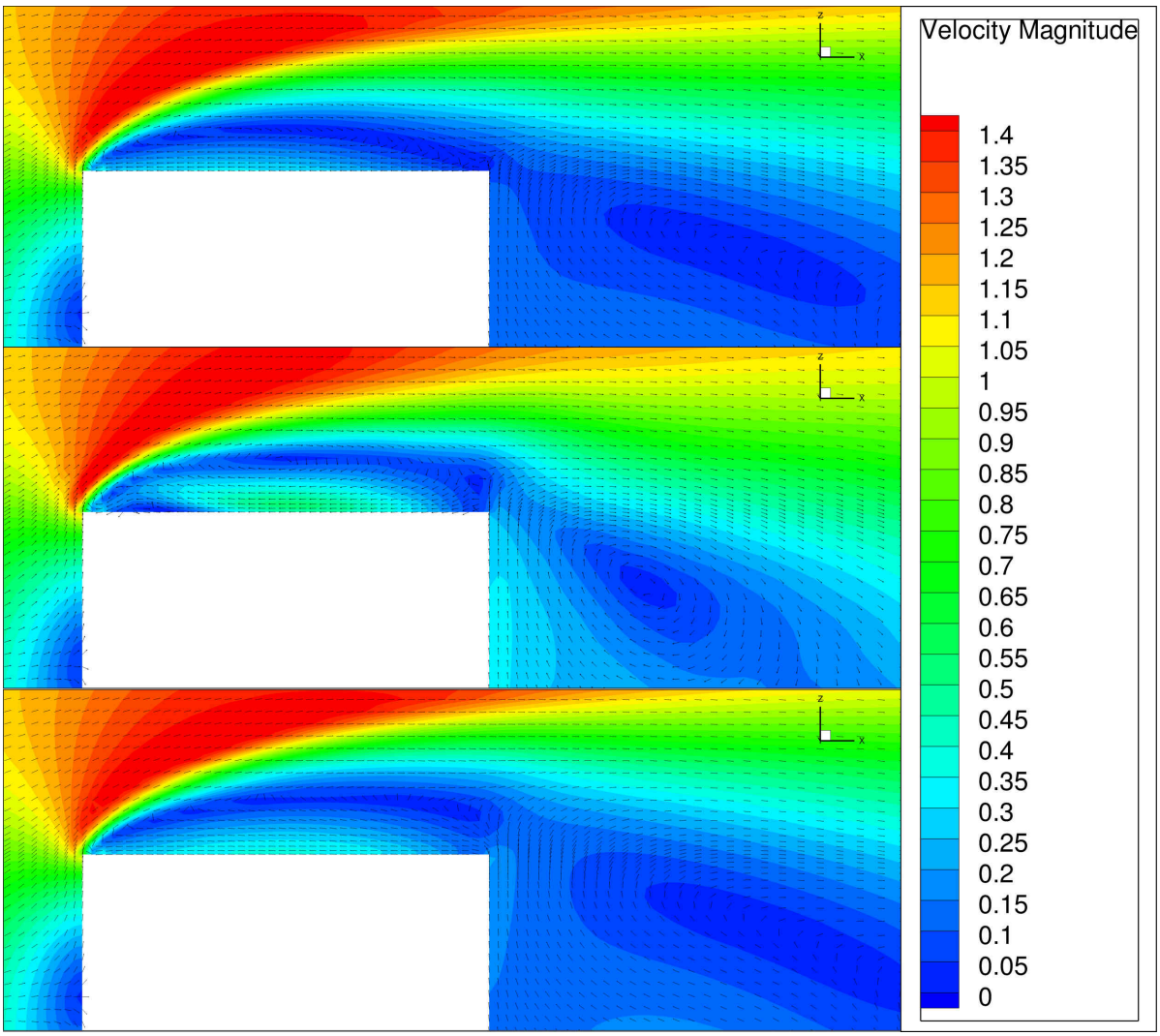


Figure A.3: Wall mounted cube: Comparing a correction model using 4 tensors and 2 invariants (Top) with a correction model using 10 tensors and 5 invariants (Bottom). The DES solution is shown in the centre. Velocity magnitude with vectors, detail of the recirculation area on top of the cube.

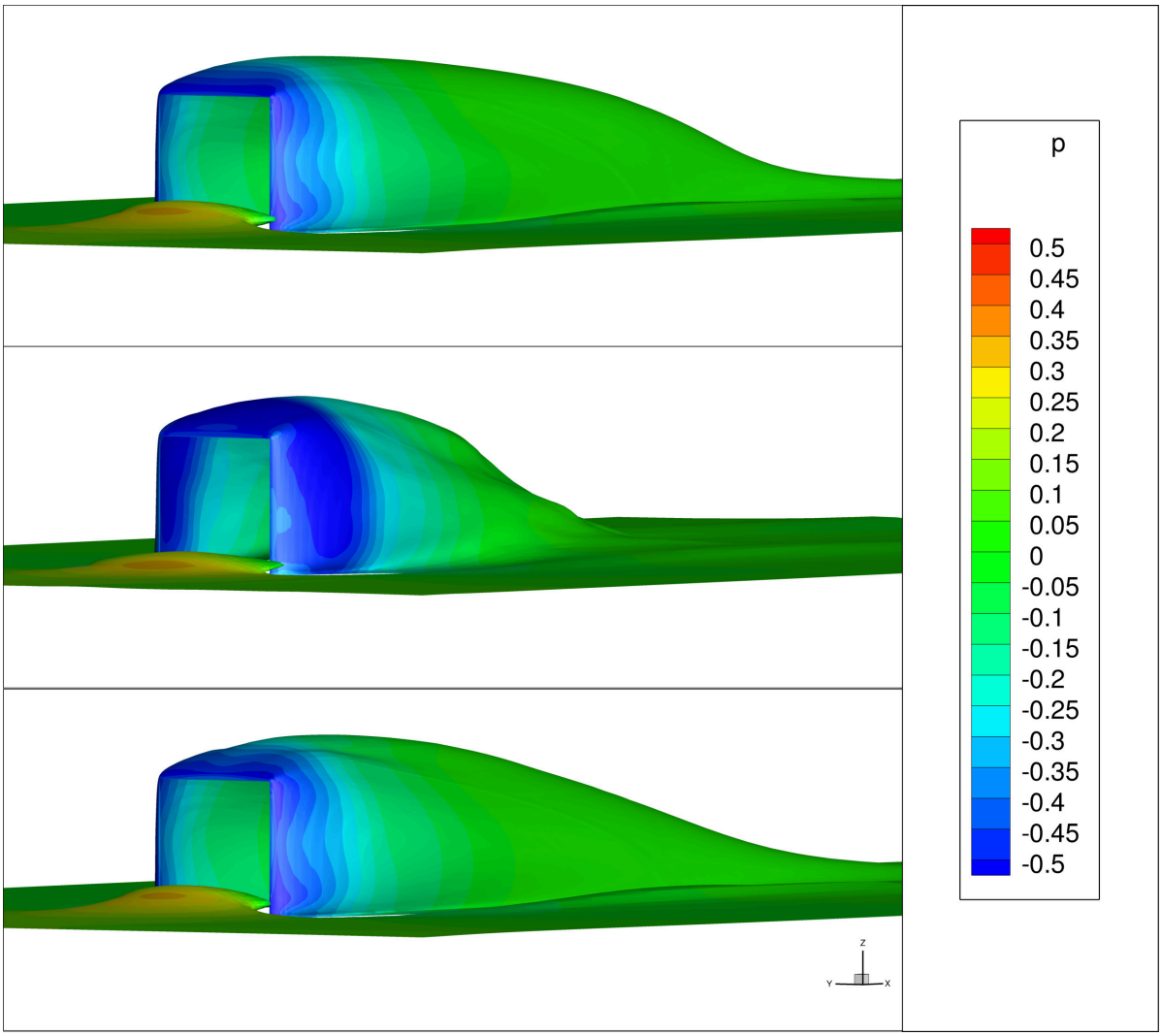


Figure A.4: Wall mounted cube: Comparing a correction model using 4 tensors and 2 invariants (Top) with a correction model using 10 tensors and 5 invariants (Bottom). Isosurface of $C_{p0} = 0.5$.

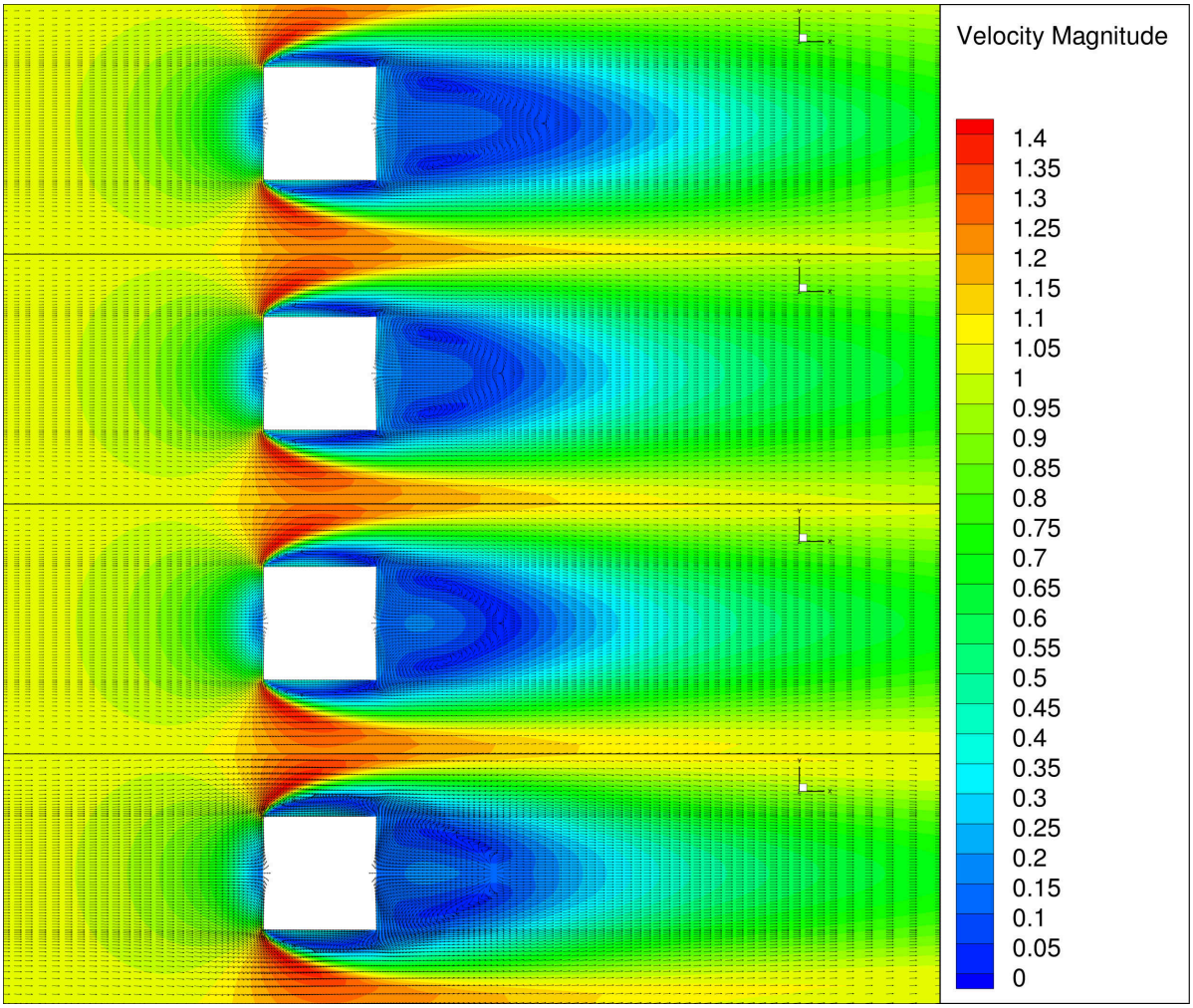


Figure A.5: Wall mounted cube: Visualisation of the effect of the corrective fields. 1st: 50%R 0%b_{ij}^{Δ} 2nd: 100%R 0%b_{ij}^{Δ} 3rd 100%R 25%b_{ij}^{Δ} 4th 100%R 50%b_{ij}^{Δ}. Velocity magnitude with vectors $z = 0.5$.

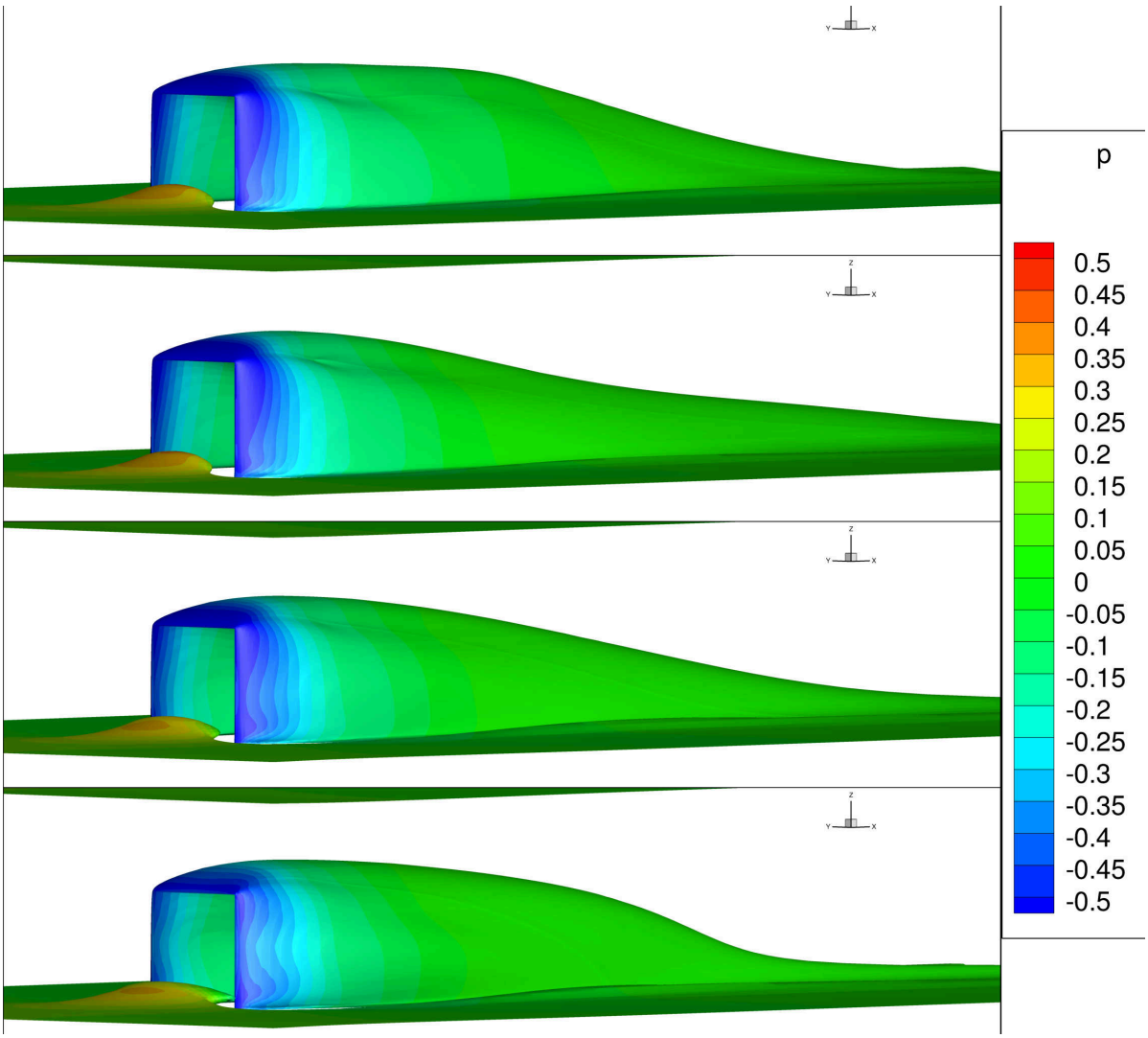


Figure A.6: Wall mounted cube: Visualisation of the effect of the corrective fields. 1st: 50%R 0%b_{ij}^{Δ}, 2nd: 100%R 0%b_{ij}^{Δ}, 3rd 100%R 25%b_{ij}^{Δ} 4th 100%R 50%b_{ij}^{Δ}. Isocontour of the total pressure coefficient $C_{p0} = 0.5$, coloured by the pressure.

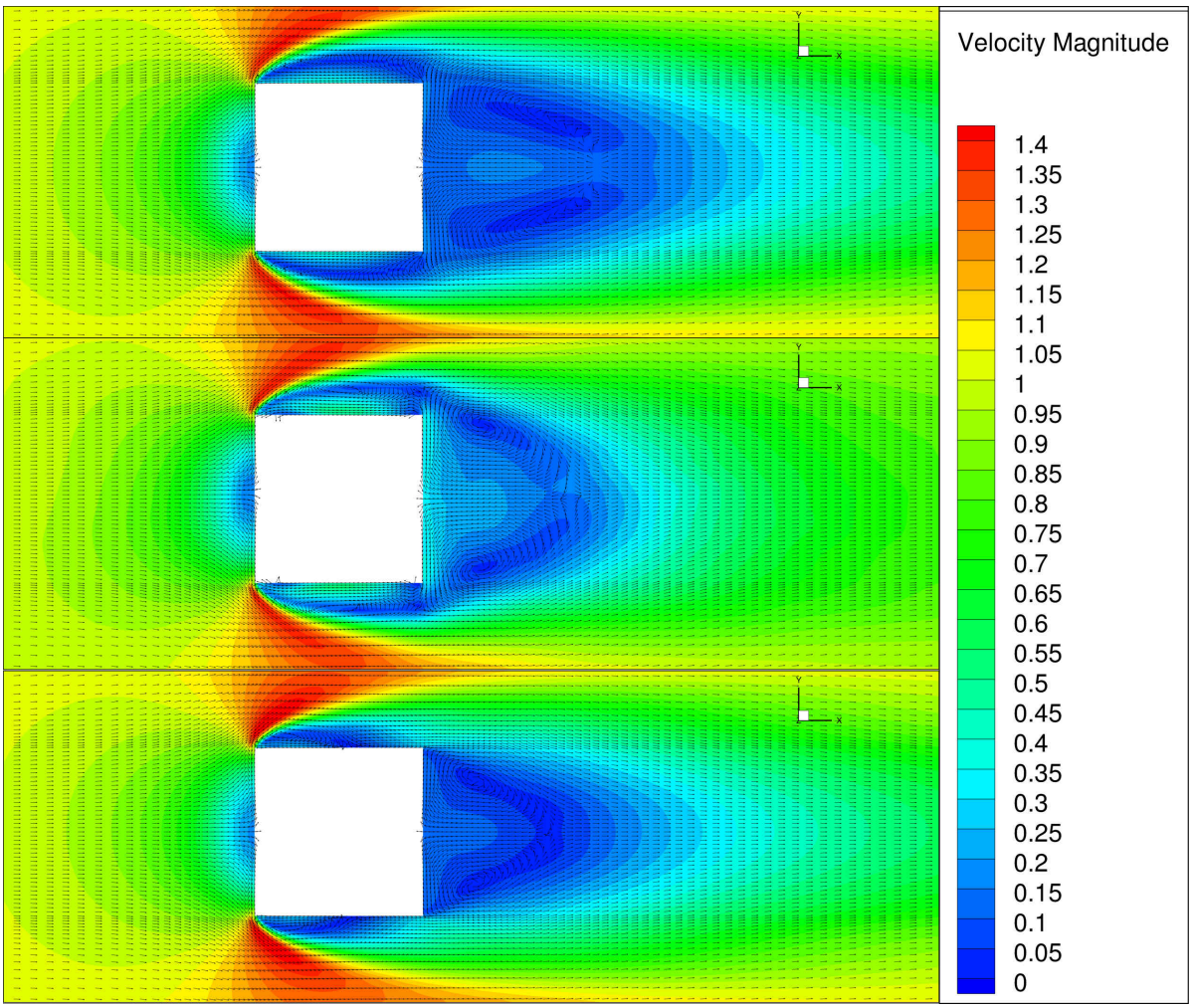


Figure A.7: Wall mounted cube: Comparing 50% b^{Δ}_{bij} correction (Top) with an averaged solution with 100% b^{Δ}_{ij} correction (Bottom). Both have 100% R correction. The middle is the DES solution for reference. Velocity magnitude on a plane above ground at cube centre height at z=0.5

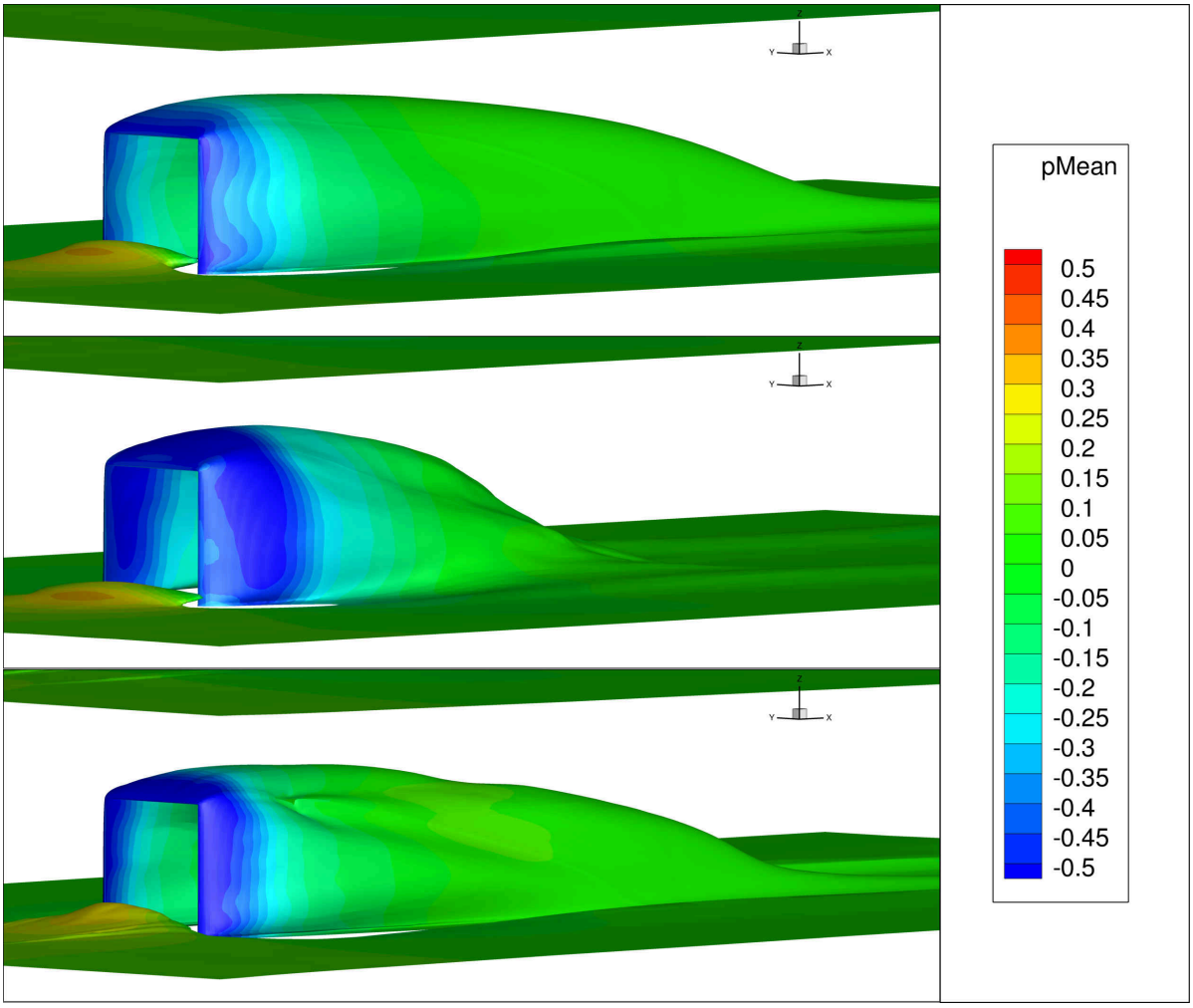


Figure A.8: Wall mounted cube: Comparing 50% b^{Δ}_{bij} correction (Top) with an averaged solution with 100% b^{Δ}_{ij} correction (Bottom). Both have 100% R correction. The middle is the DES solution for reference. Isocontour of the total pressure coefficient ($C_{p0} = 0.5$) coloured by pressure.

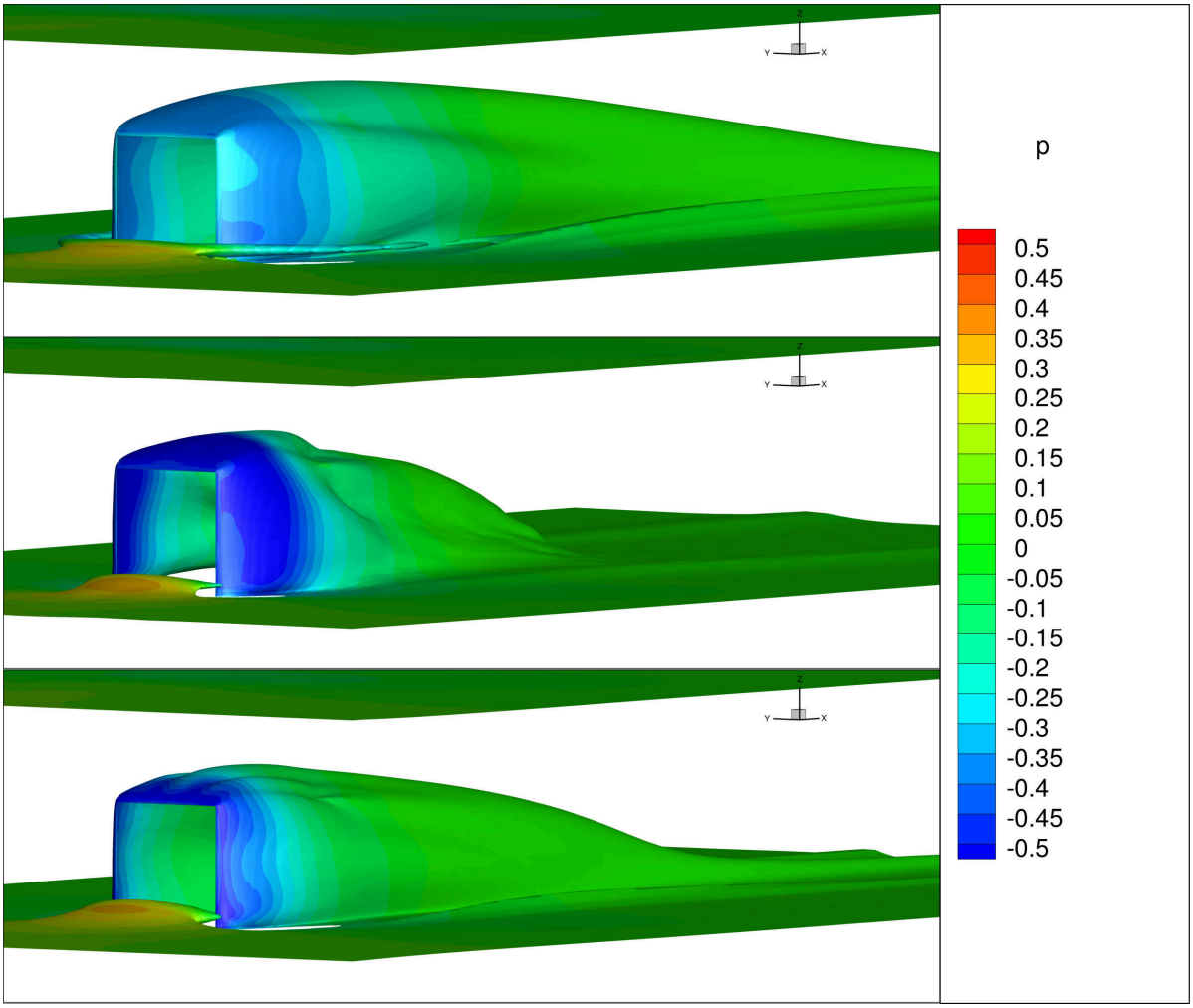


Figure A.9: Wall mounted box: Total pressure iso-surface $C_{p0} = 0.5$, coloured by the pressure. Comparison for the extended box. Top: Baseline RANS, Middle: DES, Bottom: RANS $100\%R, 50\%b_{ij}^{\Delta}$ correction applied. Model equations (4.1)(4.2)

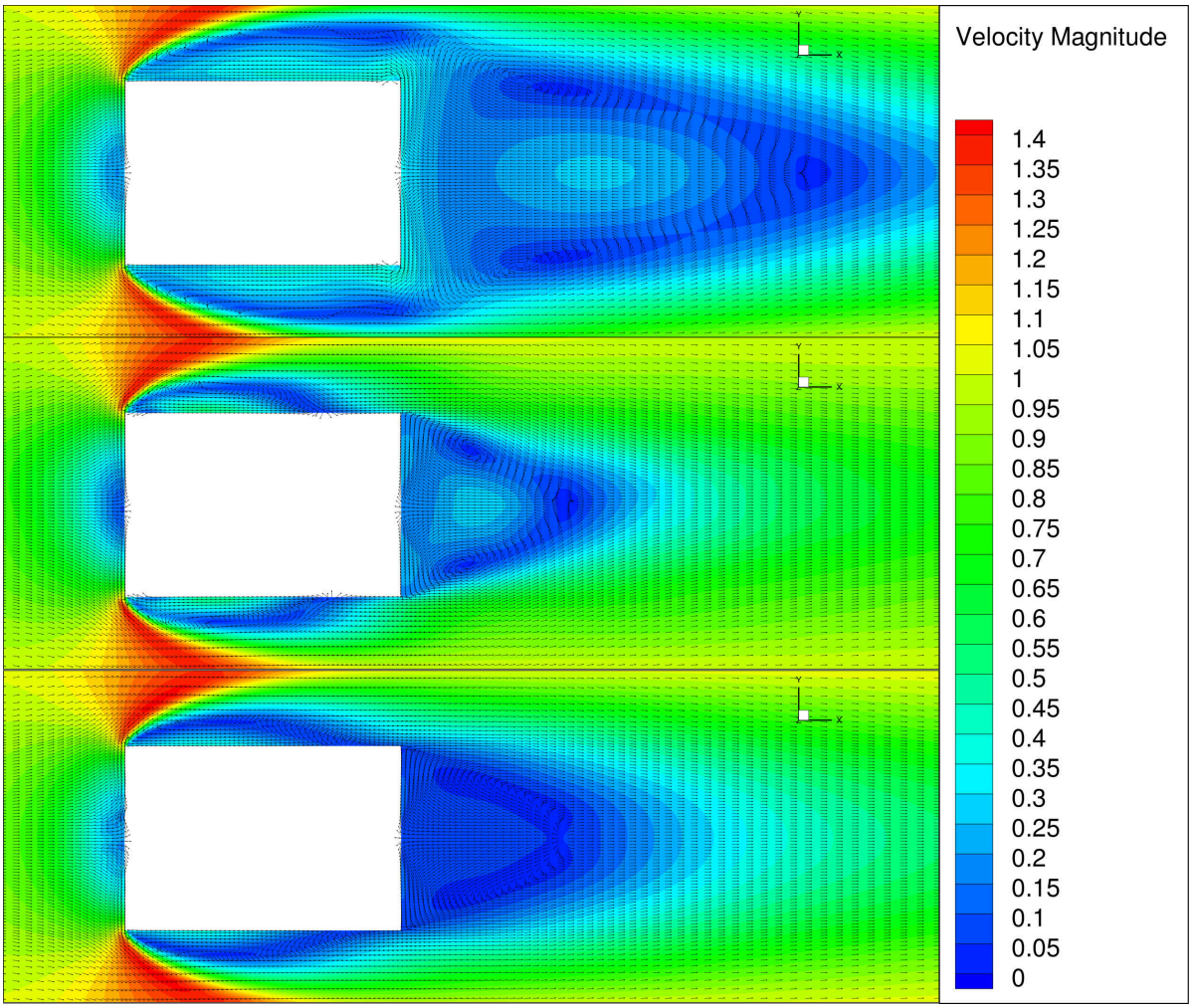


Figure A.10: Wall mounted box: Velocity magnitude with vectors at the centre height of the extended box $z = 0.5$. Top: Baseline RANS, middle: DES, bottom: RANS $100\%R, 50\%b_{ij}^{\Delta}$ correction applied. Model equations (4.1)(4.2)

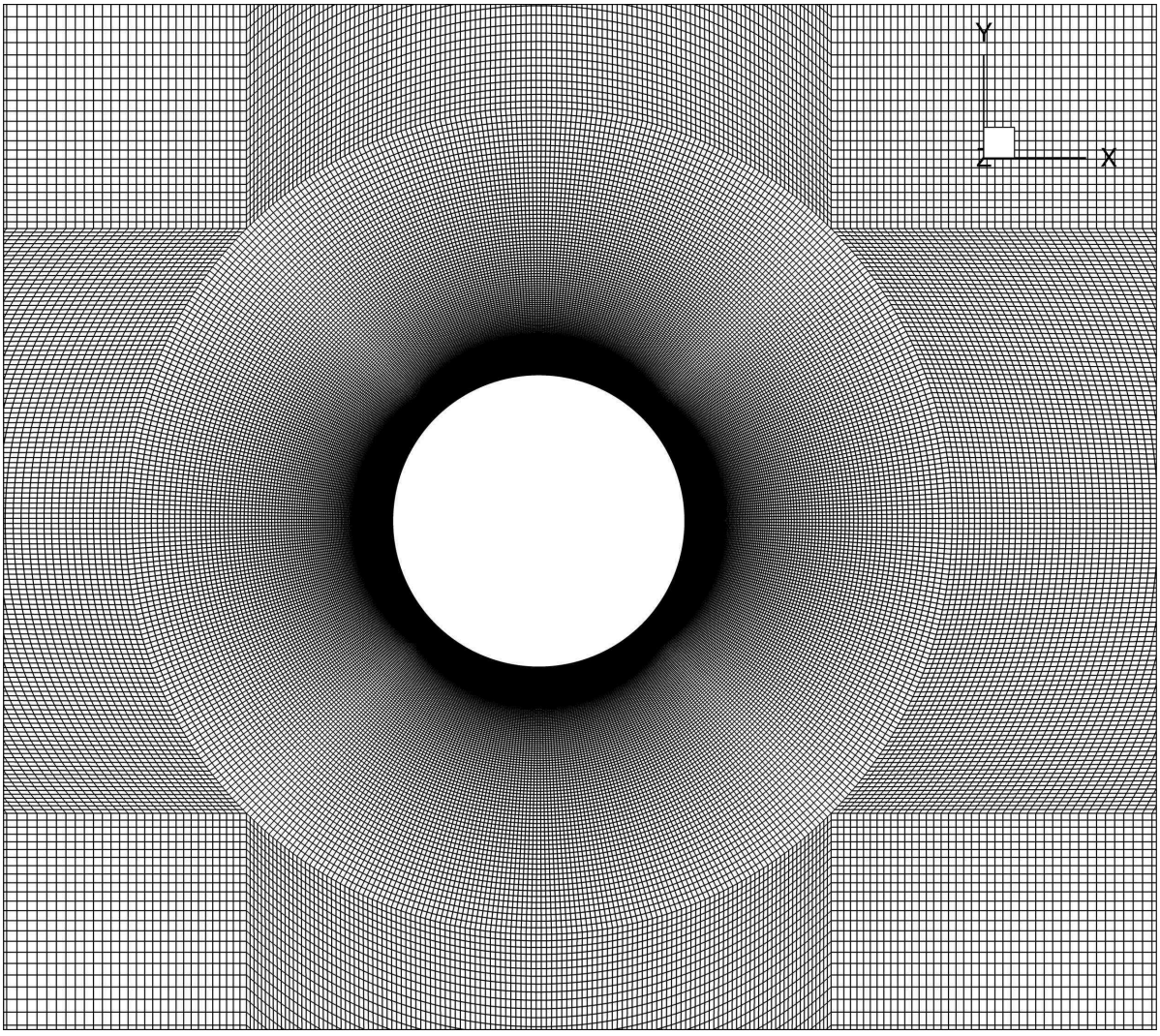


Figure A.11: Infinite circular cylinder: Overview of the mesh structure for the infinite cylinder case.

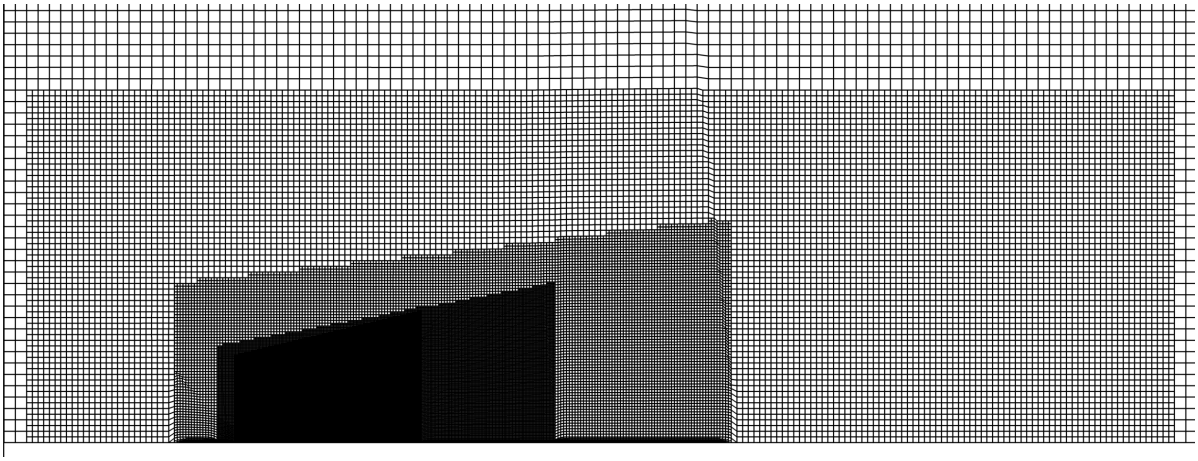


Figure A.12: Idealised wheel: Mesh topology used for the LES

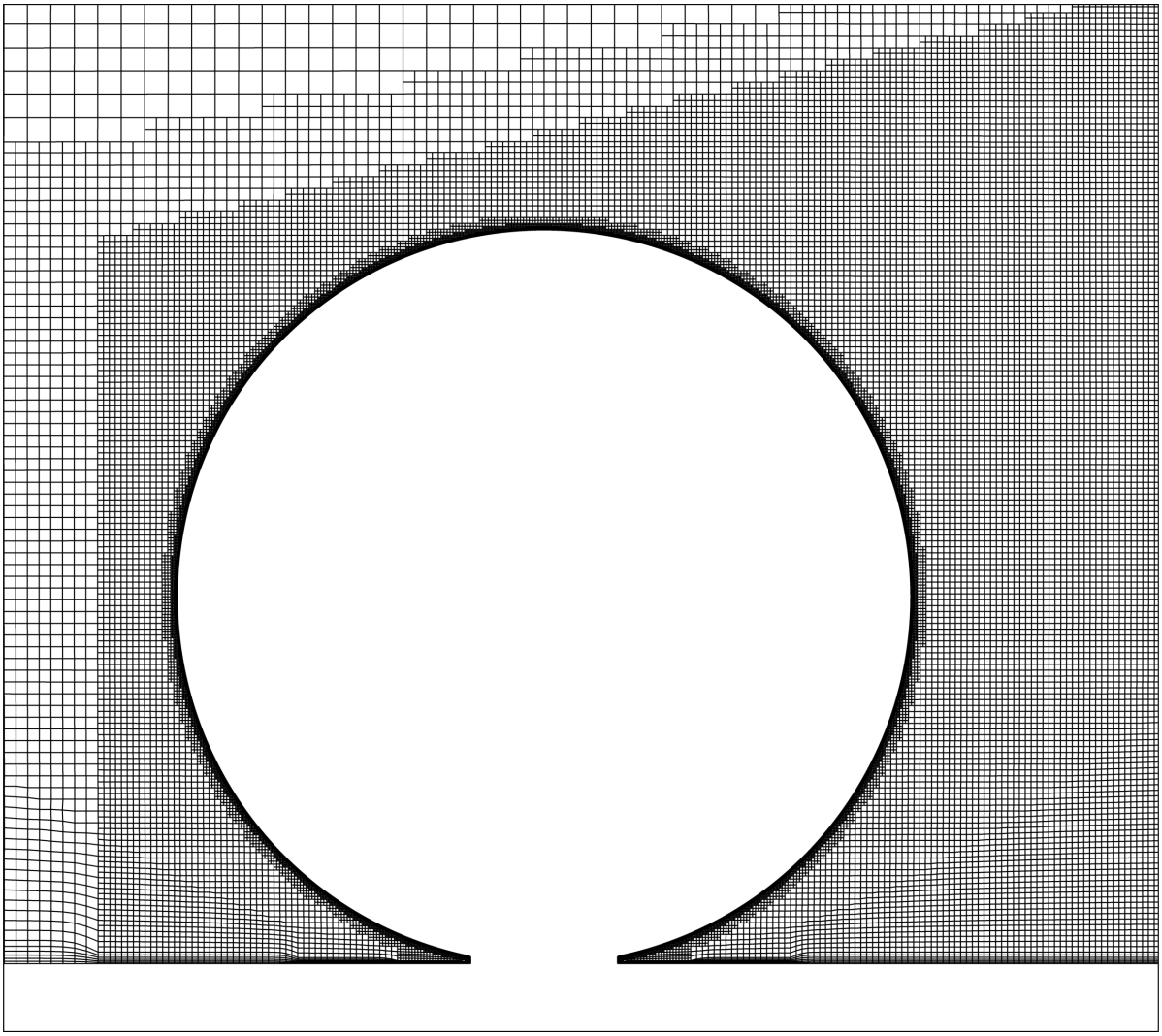


Figure A.13: Idealised wheel: Detail of the refinement region for the LES

B

Frozen Approach code

B.1. Variable Definitions

```
    aij_
    (
        IOobject
        (
            "aij",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        tauij_ - ((2.0/3.0)*I)*k_
    ),
    aijBoussinesq_
    (
        IOobject
        (
            "aijBoussinesq",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        0.0*symm(fvc::grad(this->U_))*this->nut_
    ),
    aijDelta_
    (
        IOobject
        (
            "aijDelta",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        0.0*symm(fvc::grad(this->U_))*this->nut_
    ),
    bijDelta_
    (
        IOobject
        (
            "bijDelta",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        0.0*symm(fvc::grad(this->U_))/omega_
    ),
```

```
    Pk_
    (
        IOobject
        (
            "Pk",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_,
        dimensionedScalar("Pk",dimensionSet(0,2,-3,0,0,0,0), 0.0)
    ),
    PkBoussinesq_
    (
        IOobject
        (
            "PkBoussinesq",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_,
        dimensionedScalar("PkBoussinesq", dimensionSet(0,2,-3,0,0,0,0), 0.0)
    ),
    PkDelta_
    (
        IOobject
        (
            "PkDelta",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_,
        dimensionedScalar("PkDelta",dimensionSet(0,2,-3,0,0,0,0),0.0)
    ),
    timescale_limited_
    (
        IOobject
        (
            "timescale_limited",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_,
        dimensionedScalar("timescale_limited", dimensionSet(0,0,1,0,0,0,0), 0.0)
    ),
    decayControl_
    (
        Switch::lookupOrAddToDict
        (
            "decayControl",
            this->coeffDict_,
            false
        )
    ),
```

B.2. Modified transport equations

```
template<class BasicEddyViscosityModel>
void frozen_kOmegaSSTBase<BasicEddyViscosityModel>::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_));

    BasicEddyViscosityModel::correct();

    volScalarField::Internal divU(fvc::div(fvc::absolute(this->phi(), U)));
    volScalarField divU2(fvc::div(fvc::absolute(this->phi(),U)));

    tmp<volTensorField> tgradU = fvc::grad(U);
    volScalarField S2(2*magSqr(symm(tgradU())));
    volScalarField::Internal GbyNu0(S2-((aijDelta_ && tgradU())/(nut+nutSmall_)));
    volScalarField::Internal G(this->GName(), nut*GbyNu0);

    Pk_ = G;
    PkBoussinesq_ = nut*S2;
    PkDelta_ = -aijDelta_ && tgradU();

    // 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*(GbyNu((G/(nut+nutSmall_)), F23(), S2())+(kDeficit_/(
                nut+nutSmall_)))
            //alpha()*rho()*gamma*(GbyNu(GbyNu0, F23(), S2())+(kDeficit_/(nut+nutSmall_
                )))
          - 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_
            )
          + alpha()*rho()*beta*sqr(omegaInf_)
          + Qsas(S2(), gamma, beta)
          + omegaSource()
          + fvOptions(alpha, rho, omega_)
        );

```

```
68          omegaEqn.ref().relax();
69          fvOptions.constrain(omegaEqn.ref());
70          omegaEqn.ref().boundaryManipulate(omega_.boundaryFieldRef());
71          solve(omegaEqn);
72          fvOptions.correct(omega_);
73          bound(omega_, this->omegaMin_);
74      }
75
76      bound(k_, this->kMin_);
77
78      kDeficit_= fvc::ddt(alpha*rho, k_)
79        + fvc::div(alphaRhoPhi, k_)
80        - fvc::laplacian(alpha*rho*DkEff(F1), k_)
81        - alpha()*rho()*Pk(G)
82        + fvc::SuSp((2.0/3.0)*alpha()*rho()*divU2, k_)
83        + fvc::Sp(alpha()*rho()*betaStar_*omega_, k_)
84        - alpha()*rho()*betaStar_*omegaInf_*kInf_;
85
86      correctNut(S2);
87
88      // Re-calculate aijDelta
89
90      volSymmTensorField::Internal aijBoussinesq_ = -nut*twoSymm(tgradU());
91      volSymmTensorField::Internal aijDelta_ = aij_-aijBoussinesq_;
92      volSymmTensorField::Internal bijDelta_ = aijDelta_/(2*(k_+this->kMin_));
93
94      // Calculate timescale
95      volScalarField S = sqrt(2*magSqr(symm(tgradU())));
96      timescale_limited_ = - 1./max( S/a1_ + this->omegaMin_, omega_ + this->omegaMin_);
97
98      tgradU.clear();
99
100 }
```

C

Propagated Approach code

C.1. Modified transport equations

```
void prop_kOmegaSSTBase<BasicEddyViscosityModel>::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_));

    BasicEddyViscosityModel::correct();

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

    tmp<volTensorField> tgradU = fvc::grad(U);
    volScalarField S2(2*magSqr(symm(tgradU())));
     dev(twoSymm(tgradU())));
    volScalarField::Internal GbyNu0(S2-(2*k_*bijDelta_ && symm(tgradU())/nut));
    volScalarField::Internal G(this->GName(), nut*GbyNu0);

    // 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*(GbyNu(GbyNu0, F23(), S2())+kDeficit_/(nut+nutSmall_))
```

```
            - 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_
              )
            + alpha()*rho()*beta*sqr(omegaInf_)
            + 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)+kDeficit_)
      - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
      - fvm::Sp(alpha()*rho()*epsilonByk(F1, tgradU()), k_)
      + alpha()*rho()*betaStar_*omegaInf_*kInf_
      + kSource()
      + fvOptions(alpha, rho, k_)
    );

    tgradU.clear();

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

    correctNut(S2);
}
```

C.2. devReff

```
template<class BasicEddyViscosityModel>
tmp<volSymmTensorField> ML_kOmegaSSTBase<BasicEddyViscosityModel>::devReff
 (
     //const singlePhaseTransportModel& laminarTransport,
     //const transportModel& laminarTransport,
     const volVectorField& U
 ) const
 {
     return tmp<volSymmTensorField>
     (
         new volSymmTensorField
         (
             IOobject
             (
                 "devRhoReff",
                 this->mesh_.time().timeName(),
                 this->mesh_,
                 IOobject::NO_READ,
                 IOobject::NO_WRITE
             ),
            //-(laminarTransport.nu() + nutRef())*dev(twoSymm(fvc::grad(U))) // linear
                part
            - this->nuEff()*dev(twoSymm(fvc::grad(U))) // linear part
            + dev(2.0*k_*bijDelta_) // non-linear part
         )
     );
 }
```

C.3. divDevReff

```
template<class BasicEddyViscosityModel>
tmp<fvVectorMatrix> ML_kOmegaSSTBase<BasicEddyViscosityModel>::divDevReff(
    volVectorField& U) const
{
    return
    (
      - fvm::laplacian(this->nuEff(), U)
      - fvc::div(this->nuEff()*dev(T(fvc::grad(U)))) //linear part
      + fvc::div(dev(2.0*k_*bijDelta_)) //non-linear part
    );
}
```

D

Modelled Approach Code

D.1. Modified Transport equations

```

{
    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_));

    BasicEddyViscosityModel::correct();

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

    tmp<volTensorField> tgradU = fvc::grad(U);
    volScalarField S2(2*magSqr(symm(tgradU())));


    dimensionedScalar a31
    (
        "a31",
        dimensionSet(0,0,0,0,0,0,0),
        0.31
    );


    // Here comes the expression for the correction.
    volScalarField timel = max((sqrt(2*(symm(tgradU()) && symm(tgradU())))/a31),omega_)
        ;


    volSymmTensorField Sij_t = symm(tgradU())/(timel);

    volSymmTensorField Sij = dev(Sij_t);
    volTensorField Oij = dev(tgradU())/(timel);

    volSymmTensorField T1 = Sij;
    volSymmTensorField T2 =symm((Sij&Oij)-(Oij&Sij));
    volSymmTensorField T3 =symm((Sij & Sij)-(1/3)*tr(Sij&Sij)*tensor::one);
    volSymmTensorField T4 =symm((Oij & Oij)-(1/3)*tr(Oij&Oij)*tensor::one);
    volSymmTensorField T5 =symm((Sij & (Oij & Oij))-((Sij & Sij) & Oij));
    volSymmTensorField T6 =symm((((Oij & Oij) & Sij) + (Sij & (Oij & Oij)))-(2/3)*tr(
        Sij & (Oij & Oij))*tensor::one);
```

```
    volSymmTensorField T7 =symm(((Oij & Sij) & (Oij & Oij)) - ((Oij & Oij) & (Sij & Oij
        )));
    volSymmTensorField T8 =symm(((Sij & Oij) & (Sij & Sij)) - ((Sij & Sij) & (Oij & Sij
        )));
    volSymmTensorField T9 =symm(((Oij & Oij) & (Sij & Sij) + ((Sij & Sij) & (Oij & Oij)
        )) -(2/3)*tr((Sij & Sij) & (Oij & Oij))*tensor::one);
    volSymmTensorField T10 = symm((Oij & (Sij & Sij) & (Oij & Oij)) -((Oij & Oij) & (
        Sij & Sij) & Oij));
    volScalarField I1 = tr(Sij & Sij);
    volScalarField I2 = tr(Oij & Oij);
    volScalarField I3 = tr(Sij & Sij & Sij);
    volScalarField I4 = tr((Oij & Oij) & Sij);
    volScalarField I5 = tr((Oij & Oij) & (Sij & Sij));


    volSymmTensorField bDeltaR = (T1*(-35.74086*I1*I2 - 69.5826*I1 + 39.73533*pow(I2,2)
         + 7.57252*I2 + 3.73854) + T3*(5.86701*I1 + 0.3755*pow(I2,2) - 6.78427*I2 -
        16.47849) + T4*(-2.30422*I1 - 0.21816*pow(I2,2) + 3.13597*I2 - 4.82169));

    bijDelta_= (T1*(28.68361*I1 + 4.71739*I2 - 0.35604) + T2*(-88.99302*I1 + 68.76692*
        I2 + 13.80382) + T3*(20.59135*I1 - 0.85935) + T4*(11.45876*I1 - 2.76972*I2 -
        0.94415));

    aij_=-nut*dev(twoSymm(tgradU()))+2*k_*bijDelta_;


    volScalarField::Internal GbyNu0(S2-(2*k_*bijDelta_ && symm(tgradU())/(nut+
        nutSmallf_)));//(tgradU() && dev(twoSymm(tgradU()))));
    volScalarField::Internal G(this->GName(), nut*GbyNu0);

    // 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*(GbyNu(G/(nut+nutSmall_), F23(), S2())+kDeficit_/(nut
              +nutSmall_))
            alpha()*rho()*gamma*(GbyNu(GbyNu0, F23(), S2())+kDeficit_/(nut+nutSmall_))
          - 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_
            )
          + alpha()*rho()*beta*sqr(omegaInf_)
          + Qsas(S2(), gamma, beta)
          + omegaSource()
          + fvOptions(alpha, rho, omega_)
        );

        omegaEqn.ref().relax();
        fvOptions.constrain(omegaEqn.ref());
        omegaEqn.ref().boundaryManipulate(omega_.boundaryFieldRef());
```

```
108         solve(omegaEqn);
109         fvOptions.correct(omega_);
110         bound(omega_, this->omegaMin_);
111     }
112
113     // Turbulent kinetic energy equation
114     tmp<fvScalarMatrix> kEqn
115     (
116         fvm::ddt(alpha, rho, k_)
117       + fvm::div(alphaRhoPhi, k_)
118       - fvm::laplacian(alpha*rho*DkEff(F1), k_)
119      ==
120         alpha()*rho()*(Pk(G)+kDeficit_)
121       - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
122       - fvm::Sp(alpha()*rho()*epsilonByk(F1, tgradU()), k_)
123       + alpha()*rho()*betaStar_*omegaInf_*kInf_
124       + kSource()
125       + fvOptions(alpha, rho, k_)
126     );
127
128     tgradU.clear();
129
130     kEqn.ref().relax();
131     fvOptions.constrain(kEqn.ref());
132     solve(kEqn);
133     fvOptions.correct(k_);
134     bound(k_, this->kMin_);
135
136     correctNut(S2);
137 }
```