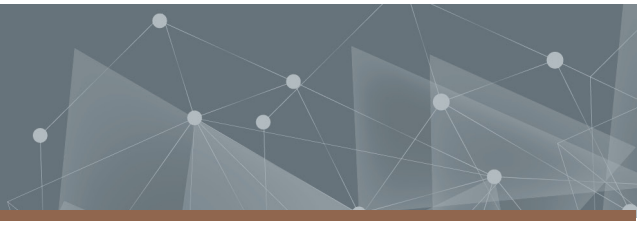
CHALMERS



Wall modeled computational fluid dynamics using machine learning

Using CFD coupled machine learning to create wall models for increased LES knowledge for future industrial use

Master's thesis in Applied Mechanics

RASMUS OLAUSSON

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES DIVISION OF FLUID DYNAMICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg 2024
www.chalmers.se

Master's thesis 2024

Wall modeled computational fluid dynamics using machine learning

Using CFD coupled machine learning to create wall models for increased LES knowledge for future industrial use

RASMUS OLAUSSON

CHALMERS

Department of Mechanics and Maritime Sciences Division of Fluid Dynamics
Chalmers University of Technology
Göteborg 2024

Wall modeled computational fluid dynamics using machine learning
Using CFD coupled machine learning to create wall models for increased LES knowledge for future industrial use
RASMUS OLAUSSON

Supervisor: Magnus Carlsson, SAAB Aeronautics
Examiner: Lars Davidson, Division of Fluid Dynamics

Master's thesis 2024
Department of Mechanics and Maritime Sciences Division of Fluid Dynamics
Chalmers University of Technology
SE-412 96 Göteborg
Telefon +46 31 772 1000

Typeset in LaTeX
Göteborg 2024

Wall modeled computational fluid dynamics using machine learning
Using CFD coupled machine learning to create wall models for increased LES knowledge for future industrial use
RASMUS OLAUSSON
Department of Mechanics and Maritime Sciences Division of Fluid Dynamics
Chalmers University of Technology

Abstract

Machine learning is becoming a useful tool in many parts of engineering and science and computational fluid dynamics is no exception. At the same time the need for more accurate simulations with resolved turbulence is also increasing. But to make turbulence resolving methods such as Large Eddie Simulations (LES) accessible for industrial use computational speed up is still required. One method is to employ wall modeled LES. Classical wall models works well for simple flows but they get inaccurate for flows containing adverse pressure gradient. A solution to this problem is to use a machine learning based wall model.

In this project a machine learning based method for wall models is investigated and a new proposed way of coupling a CFD solver to the training process is tested. The method combines a supervised learning method with an unsupervised learning approach to take numerical inaccuracies of the CFD solver into the wall model. The models in this project were trained on channel flow and boundary layer flow. The results shown that machine learning based methods do work and give accurate results. These models are compared to classical wall functions and they can be more accurate. Some problems in the training process was found and especially the training time could become unreasonable if coupled with a LES solver. Solutions to these problems are discussed and in the future more complicated flows with adverse pressure gradients and LES simulations should be investigated.

Nyckelord: CFD, LES, RANS, Machine Learning, Wall models.

Acknowledgements

I want to give a massive thanks to my supervisor Magnus Carlson at SAAB who has been great. He has guided me throughout thus project and his advice and support has been invaluable.

I also want to thank Sebastion Arvidsson at SAAB for giving me this opportunity and showing his interest in the project his input and support has been greatly appreciated.

Lastly I want to thank my examiner Lars Davidson who's input and knowledge in the field has been needed in many of the cross roads faced in this project.

Rasmus Olausson, Göteborg, June 2024

Contents

List of Figures

1
Introduction

As computing power increases and the need for more accurate computational fluid dynamics (CFD) simulations is on the rise, more engineers are looking at Large Eddy Simulations (LES) as a tool to use when Reynolds-averaged Navier–Stokes (RANS) does not give accurate results. The academic world of research in turbulence modelling are also increasingly choosing LES over RANS. The reason for this shift over the last years is the added accuracy of LES. In LES the large scale turbulence is resolved in time and space while the sub-grid scale turbulence is modelled. In RANS simulations all turbulent scales are modelled using a turbulence model.

The added accuracy of a LES simulation comes with an additional cost to computational power caused by the strict mesh requirements needed to resolve turbulence, both in space and time. Near walls the computational mesh needs to be even finer to resolve near-wall eddies in the buffer layer and log-layer. This kind of simulation where the near-wall eddies are resolved is called wall-resolved LES (WRLES) and while being accurate a lot of computational cost is spent on resolving the flow in the viscous sublayer and log-layer. This is not ideal since in engineering application the mean flow and large scale properties of the flow is often the desired outcome of a CFD simulation. This strict mesh requirement at the wall has lead many researchers to look into wall modeled LES (WMLES). In WMLES the boundary layer flow is not fully resolved instead a wall model is used to set the boundary conditions at the wall. This means the mesh requirements of the near-wall region are not as strict and will save on computational time if done correctly leading to the same large scale flow properties.

In recent years machine learning (ML) and artificial intelligence have been shown to be a powerful tool in many fields of science. In CFD research more scientist are looking towards ML methods to improve CFD methods both in turbulence modeling and creating wall functions. The natural non-linearity and ability to find relationships in a large data set makes ML methods a promising candidate for modeling complex fluid flow.

This rapport is a first step in the creation of a ML based wall model for CFD. This work will focus on creating a working wall function for RANS using machine learning method. And will be a important first step of using such methods to improve the viability of large scale LES simulations in the future. The report will discuss different wall models which have been used in the past and how new methods are emerging especially using machine learning (ML). A review on excising literature is outlined

and their methods are analysed. Then the a ML wall model is trained using a CFD-coupled approach and tested on channel-flow and boundary layer flow.

1.1 Purpose

The aim of this project is to understand how ML methods have been used in CFD and how it can used in the future to improve wall models in more complicated flow cases. A ML based wall model is presented and trained using a new proposed method called CFD coupled machine learning.

1.2 Limitations

The project will be limited to 2D flow cases to decrease training time and complexity. The goal was to include flow cases which with recirculation regions but due to time limitations these cases were not includede in the training process.

2
Theory

This chapter will explain the governing equations and how using a wall model can decrease cell count and simulation time. A study of available literature will be preformed where both classical and ML based wall models are covered. Most of the literature discusses the use of ML methods for WMLES but their methods and learning's are equally relevant in RANS cases.

2.1 The governing equations

The RANS equations are derived from time averaging the Navier-Stokes equations. The chosen time interval is chosen to be smaller than the macroscopic flow change yet bigger than turbulent eddy fluctuations [3]. The properties of the flow is split into two parts $\Phi = \bar{\Phi} + \Phi'$, where $\bar{\Phi}$ is the averaged quantity while Φ' is a fluctuation. The resulting RANS equations are seen below

$$\frac{\partial \bar{v}_i}{\partial x_i} = 0$$
$$\frac{\partial \bar{v}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{v}_i \bar{v}_j\right) = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu \frac{\partial^2 \bar{v}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j}$$

where the stresses $\tau_{ij} = \rho\overline{v_i' v_j'}$ are called the Reynolds stresses.[5]

2.1.1 The closure problem

The RANS equations are not a closed system of equations since the Reynolds stresses are unknowns and a turbulence model is needed to solve them. This is referred to as the closure problem. The Reynolds stresses are often modeled using the Boussinesq assumption

$$\overline{v_i' v_j'} = -\nu_t \left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right) + \frac{1}{3}\delta_{ij}\overline{v_k' v_k'}$$

where ν_t is the turbulent viscosity and is unknown.[5]

In this project the Wilcox k-ω turbulence model is used, the turbulent viscosity is modeled as $\nu_t = k/\omega$. For both k, the turbulent kinetic energy, and ω, the turbulent frequency, a transport equation is solved.[4]

2.2 Mesh requirements and different wall treatments

In CFD the RANS equations are either solved by resolving the boundary layer of by using a wall model. In wall resolved RANS the first cell is put at $y^+ \sim 1$ thus all velocity gradients are resolved. The non-dimensional wall distance y^+ is defined as

$$y^+ = \frac{yu_\tau}{\nu}$$

where $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity and τ_w is the wall shear stress. In larger geometries most of the work of the CFD-solver is spent in the boundary layer region which is not always the point of interest. Thus a reduction in near wall cells leads to a large decrease in computational time.[1]

To decrease the near wall cells a wall function can be used. In RANS simulations with wall models the first cell is placed higher up around $30 < y^+ < 250$. This increase in height of the first cell decreases the number of cells and enables bigger domains to be simulations. Since the first cell is placed further from the wall than in wall resolved RANS a no-slip boundary condition is no longer valid. Instead the wall shear stress is approximated by the wall function.[11]

2.3 Classical wall functions

Most classical wall functions are derived from momentum conservation in the near-wall cells either by assuming a near-wall velocity profile or solving RANS momentum equations near the wall to set the wall-shear stress. But solving the full RANS equations becomes expensive and often the thin boundary layer equations (TBLE) are solved instead. In these equations the near-wall turbulence have been filtered out and u_2, the wall-normal component, is calculated from the continuity equation.

$$\frac{\partial \widetilde{u_i}}{\partial t} + \frac{\partial \widetilde{u_i}\widetilde{u_j}}{\partial x_j} + \frac{\partial P_m}{\partial x_i} = \frac{\partial}{\partial y}\left[(\nu + \nu_t)\frac{\partial \widetilde{u_i}}{\partial y}\right], \; i = 1, 3 \tag{2.1}$$

$$\widetilde{u_2} = -\int_0^y \frac{\partial \widetilde{u_i}}{\partial x_i} dy' \tag{2.2}$$

$$\widetilde{u_i}(y_w) = 0, \; \widetilde{u_i}(y^*) = U(y^*) \tag{2.3}$$

These equations are solved with a no-slip boundary condition near the wall and matching velocities to LES solution at the upper boundary. The eddy viscosity is usually modeled with a one-equation model [11].

These equations can be further simplified to the equilibrium stress model which is the most commonly used wall function. In the equilibrium stress model the convection and pressure gradient terms are assumed to balance each other exactly and the left hand side of equation (2.1) can be assumed to be zero. The resulting ODE gives the famous law of the wall solution. For $y^+ \lesssim 5$ the solution is linear $u^+ = y^+$ and for $y^+ \gtrsim 30$ the solution is logarithmic $u^+ = \log y^+/\kappa + B$ [2].

Another approach is to fit a function to the solution of the ODE, the most used wall function using this methods is Reichardt's wall function. It can be written on the form

$$u^+ = \frac{1}{\kappa}\ln\left(1+\kappa y^+\right) + 7.8\left[1 - \exp\left(-\frac{y^+}{11}\right) - \frac{y^+}{11}\exp\left(-\frac{y^+}{3}\right)\right]$$

where $\kappa = 0.41$ is the con Karman's constant. By Taylor expansion it can be shown that for small values of y^+ the solution is linear. And for large values of y^+ the exponential go towards zero, giving a logarithmic solution. But Reichardt's model captures the buffer region as well by carefully choosing the coefficients in the expression.[10]

These models have worked well in the past and have been proven to be accurate in academic studies on flows such as channel flow. But when the flow to be studied contains an adverse pressure gradient and a re-circulation zone the assumptions of the equilibrium stress model break down and the wall model becomes far less accurate [1]. The full ODE or even the full RANS equations could still give accurate results but these methods add additional computational complexity. Due to recent success of ML methods in other fields many researchers have tried to apply ML methods to improve classical wall-functions.

2.4 Data driven wall functions

In recent years many papers and studies on different ways to apply ML methods to the field of WMLES have been published. The focus of this report is on wall modeled RANS but the methods and results of their ML methods are still relevant. Various methods have been tested such as supervised ML with feed-forward neural networks and unsupervised learning. Challenges with ML based wall models include what inputs to be used, how these inputs should be normalized and at what cell or cells should the input data be sampled. These questions will be discussed and outlined using the available literature in the following chapters.

2.4.1 Supervised neural network approach

Training a neural network with a supervised learning approach requires a large amount of data. This data is usually acquired from Direct Numerical Simulations (DNS) data but could also be LES data or even RANS data if it is deemed accurate enough. The requirement of a large dataset is one of the main drawbacks of supervised learning especially in the field of fluid dynamics. This is because DNS is very expensive and not feasible for complex flow cases. For this reason many wall-functions based on this type of machine learning is often trained on simple flows such as channel flow with the hope that the model generalise to more complicated geometries. But this often fails to give accurate predictions for complex geometries and may be only valid for a certain range of Reynolds numbers.

In the work of Vadrot *et al.* [12] three different ML based wall-models were compared to the equilibrium wall model described above. These models used were supervised

neural network wall-model trained on rotating channel flow by Huang *et al.*[13], and supervised neural network wall model trained on periodic hills by Zhou *et al.* [14]. They also compared a unsupervised model which will be discussed later in this chapter.

The inputs for the neural network in Huang's rotating channel model are y^+ and y^+/l_Ω^+ where $l_\Omega^+ = u_\tau/(2\Omega)$ and Ω is the rotation rate. The output from the model is u^+ from which the wall-shear stress can be calculated iteratively. The network is therefor built to predict the law of the wall and is called physics-informed wall modelling by the authors. The network was make up of 3 hidden layers of size 4,4,2 which is on the smaller side but works well for them since the network is already physics informed and the curve the network needs to fit is not as complicated. The network training converged after 10 000 epochs and the resulting network can predict the velocity profile well. They also try to improve their model by sampling LES data from the third grid cell instead of the first to improve log-layer mismatch. They also test their model against the equilibrium wall model on unseen data, which is rotating channel flow at different rotation speeds and Reynolds numbers where the ML methods predicts more accurate result.

The second wall function analyzed by Vadrot is Z. Zhou nerual network trained on periodic hill flow. They argued that using LES data from only one cell is not enough to accurately predict wall-shear stress. Instead they use data from three cells, the first cell is at height h_{wm} and can be chosen arbitrarily while the other lies at $h_{wm} + 0.03\delta$ and $h_{wm} + 0.06\delta$, where δ is the boundary layer thickness. They also used more inputs and a different normalization than Huang's model, the normalized inputs and outputs are summarized in table 2.1. They tried two different outputs for their neural networks either two components of the stress tensor $\tau_{xz_{wm}}$ and $\tau_{yz_{wm}}$ or just one component $\tau_{xz_{wm}}$. The network consists of 6 hidden layers with 20 neurons each, a much larger network then seen previously. The model was trained on WRLES data from periodic-hill flow at Reynolds numbers 5600 and 10595. They compared the models predicted wall-shear stress to a WRLES at different Reynolds number 19000. They later tried it on a different geometry hill flow where the errors in predicted C_f are larger but still improved results. Lastly they used their trained model in a WMLES channel flow and hill flow and compared it to available DNS data. These results were not as accurate and in the hill flow the velocity field was not accuratly predicted anywhere in the domain.

Inputs	Outputs
$\ln \frac{h_{wm}}{y^*}$, $\frac{\delta}{h_{wm}}\frac{u}{u_b}$, $\frac{\delta}{h_{wm}}\frac{v}{v_b}$, $\frac{h_{wm}}{\rho u_b^2}\frac{\partial p}{\partial x}$, $\frac{h_{wm}}{\rho u_b^2}\frac{\partial p}{\partial z}$	$\frac{\tau_{xz_{wm}}}{u_b^2}$, $\frac{\tau_{yz_{wm}}}{u_b^2}$ or $\frac{\tau_{xz_{wm}}}{u_b^2}$

Table 2.1: Inputs for Zhou neural network. The inputs are normalized using a near wall length scale $y^* = \frac{\nu}{u_{\tau,p}}$ where $u_{\tau,p} = \sqrt{u_v^2 + u_p^2}$, $u_v = \sqrt{\left|\frac{\nu u}{h_{wm}}\right|}$, $u_p = \left|\frac{\nu}{\rho}\frac{\partial p'}{\partial x}\right|^{1/3}$

Vadrot compared these two models for regular channel flow using eleven different Reynolds numbers between 180 and 10^{10}. The first model trained by Huang on rotating channels outperformed or preformed similarly as the equidistant wall-model

in the test cases. This result is promising and shows that machine learning models can be generalized. Even if the model has been trained on rotating channels it can accurately predict the log-law for before unseen Reynolds numbers. As for the second model trained on hill-flows by Zhou it did not preform as well. It over predicted the velocity compared to the log law for all Reynolds numbers and the error increased for larger Reynolds numbers. Vadrot mentions how the models reliance on three inputs at different locations might be the issue with the combination of lack of training data. Since the network is larger with many more inputs it is more sensitive to incorrect or unseen inputs which may arise when increasing the Reynolds number in this case. A. Vadrot also discusses how the choice of activation function might also be problematic. The model uses the ReLU activation functions which goes to ∞ for large inputs. This is not the correct asymptotic behavior expected of the shear stress for large inputs.

Another approach to supervised learning is the work of Lozano-Durán and Bae [15]. They trained a wall-model with DNS data with the goal of testing it on the NASA Juncture flow, which is data taken from a windtunnel experiment of a full-span wing-fuselage body. The Reynolds number of the experiment was 2.4 million and they trained on data using an angle of attack of 5 degrees. Their approach was to use two networks one classifier and one predictor. The purpose of the classifier is to decide what type of flow is present at the current cell in the simulation. They used three different flow types to describe the flow around the body: flat plate flow, juncture flow and separated flow. The result of the classifier is then inserted into the predictor which then predicts the wall-shear stress at the current cell. The way the classifier communicates with the predictor is by changing the inputs to the predictor network or rather chaining the normalization of the inputs. For flat plate turbulence they use the stress at the wall and distance from the wall to nondimensionalize the inputs. And in separated flow they use the pressure gradient and viscosity. The benefit of breaking the problem up into these two components is that the normalization is based on which are the dominating factors in the different cases. It also avoids problems like normalizing with a very small or even negative shear stresses in separated flow. However, there are drawbacks since they only consider three different flow "base cases" and the flow around an aircraft is too complex to be accurately described by only these three cases. A generalized model needs to include more "base cases" to work on any geometry. It also complicates the training process and it is almost certain that a larger dataset and longer training time would be needed to reach a good model.

Lozano-Durán and Bae also used a different way of inputting the LES data into the model. They use what they call a seven-point stencil, where the center of the stencil is placed at a height $h = 2\Delta$ from the wall which corresponds two the second cell of the wall. The other six values are then acquired by taking a step of length Δ in the three cardinal directions. The argument of doing this is that this process enables the classifier to discern different types of flow. It also enables the network to differentiate flow types in all three directions which is beneficial in more complicated separation cases on a full 3d airplane. The drawback of doing it this way is that

the inputs need to be interpolated which will be slower than just using data from a single cell. Also if using a unstructured mesh it will be more difficult locating all the required neighbors and may require extensive interpolation.

The predictor network consisted of 4 hidden layers of 30 neuron each while the classifier was a bit smaller with only 3 hidden layers of 15 neurons. Interestingly a combination of sigmoid and ReLU activation functions were used between the layers. The model was then trained using DNS data from channel flows, ducts and boundary layers with separations and a variety of Reynolds numbers. The network was then compared to the equilibrium wall-model on the NASA juncture flow case with promising results. When the classifier identified flat plate flow and corner flow their results were very good and outperforming the equilibrium wall-model by some margin. However in separations regions on the aircraft the network did not classify the flow correctly and predicted is was corner flow giving a bad predicted shear stress.

2.4.2 Reinforcement learning

In reinforcement learning (RL) the neural network is not trained against a large database as in supervised learning. Instead the network is given a target for example is turbulence modeling the target maybe a correct shear stress och a velocity profile the network should try to achieve. The benefit of this approach is that you are able to train a model without a large amount of data. This is especially great in fluid dynamics where DNS och WRLES DATA is very expensive and could be hard to get a hold of.

Since DNS data is not used the match input to output it is not used in RL. Instead the model has to explore its environment and figure out how to optimise the current problem. Instead of inputs and output pairs a RL model is formulated using agents and states. The environment is always changing but can be described currently as a state (s). An agent is connected to the environment by taking inputs (i) from the state (s). The agents job is then to choose an action (a) which then will change the environment. But for the agent to learn it also needs feedback so the change in environment is communicated to the agent using a reinforcement signal (r). The reinforcement signal (r) is a scalar number often $\{0, 1\}$ or $\mathbb{R}$ and the goal for the agent is to maximize the long term sum of the signal (r). [16]

Bae and Koumoutsakos used [17] used reinforcement learning to create a wall-model for turbulent channel flow. They distributed agents along the channel walls and each timestep the agents got the state s_n, preformed the action a_n and got the response r_n, where n is the agent in question. The goal of the model and thus the agents was to predict the wall-shear stress τ_w. Importantly they also non-dimensionless their state s_n using u_τ and ν as has been seen in the previous section. They trained two different models. In their first model they defined the state s_n as the velocity $u^*(x, h, z, t_n)$, wall-normal derivative $\frac{\partial u^*}{\partial y^*}(x, h, z, t_n)$ and wall normal location y^* the sampling point. The agents then had to decide the action a_n which was defined

as a scalar multiplication factor in the interval $a_n \in [0.9, 1.1]$ from which the next wall-shear stress could be determined from $\tau_w(x, h, z, t_{n+1}) = a_n \tau_w(x, h, z, t_n)$. The reward r_n was then proportional to the improvement in prediction of the wall-shear stress. This model preformed well in other channel flows predicting the shear stress with an error of maximum 4% even in different Reynolds numbers than it had been trained on. But they note that for higher Reynolds numbers the mesh had to be refined otherwise the sampling point y^* would lie outside the interval it had been trained on.

Bae and Koumoutsakos also trained another model using the same inputs but instead predict the coefficients in the log-law $u^+ = \frac{1}{\kappa} \log y^+ + B$. The actions and rewards were defined in the same way but now predicting two numbers. The benefit of this model is that is works independently of the location of the sampling point y^+ as long as it lies within the log-law region. This was also seen in their result where the mesh did not need to be refined for higher Reynolds numbers which the previous model needed. It did however still get worse than the equilibrium wall-model for high Reynolds numbers and the error in prediction of u_τ did increase. They also tested both models on boundary layer flow, a case neither model had seen before. The log-law based model adapted very well and also predicted C_f well. The first velocity based model behavior was worse, for small Re_θ it matched the log-law well, but as the Reynolds number increased the errors kept getting bigger.

But these two models were only trained on flow containing non recirculating flows. Bae and Koumoutsakos kept working on their model and decided to train it on periodic hill flow as well [18]. In the periodic hill flow domain 512 agents where distributed along the bottom wall uniformly but this time their wall-distance or sample point y^+ was randomized in a reasonable interval. They decided to keep working with the log-law model but for the model to work in separation regions the state or inputs to the agents had to be modified. They now included the turbulent strain rate and wall-parallel pressure gradient in the state as well. To circumvent issues with normalization as u_τ goes towards 0 at separation points they choose an other normalization. The chosen normalization they choose is the same as Z. Zhou and has been talked about in table 2.1. They also changed the action of the agents, instead of directly influencing the wall-shear stress they choose a multiplication factor which changes the eddy viscosity at the wall $\nu_{t,w}(t_{i+1}) = a\nu_{t,w}(t_i)$ where a is the action. The wall-shear stress is then calculated from $\tau_w = \rho\nu(1+\nu^*_{t,w})(\partial u_s/\partial y)_w$ and the reward is still proportional to the improvement of the prediction.

The model was tested on the same hill flow at the same Reynolds number as it was trained on and yielded a good results and the model captures the separation region well. It also predicts the C_f well in nearly the whole domain and greatly outperforms the equilibrium wall model. There are large deviations compared to DNS data at the top of the hill where the separation occurs where it greatly over predicts the C_f. The model was also tested on different Reynolds number with similar results still over predicting C_f at the separation point but outperforming the equilibrium wall model. They also tested their model on flow over a Boeing Gaussian bump which

the model had never seen before. This time the errors in C_f and flow field where larger than in the training cases but again it still outperformed the equilibrium wall model.

3
Method

This chapter discusses how the data was gathered and how the model created in this project was trained. It also explains how the process of coupling a CFD solver to ML method and how the training was done.

3.1 Neural network as a wall function

There are multiple ways of constructing a wall function using a neural network depending on what inputs and outputs are used. For example the output of the network could be the wall-shear stress τ_w directly. The wall model created in this project instead predicts the normalized velocity u^+ from which τ_w is calculated. The input of the network created is y^+ and the network consists of four hidden layers of size: 30, 40, 30, 20 and is summarised in the table 3.1. A schematic picture of the network can also be seen in figure 3.1. With this configuration the wall-shear stress can be calculated implicitly since both y^+ and u^+ depend on u_τ. This is done by iterating and 5 iterations are enough for a good convergence.

Input	H_1	H_2	H_3	H_4	Output
y^+	30	40	40	20	u^+

Table 3.1: The network architecture used for training both channel flow and boundary flow. H_i is hidden layer i with the number of nodes.

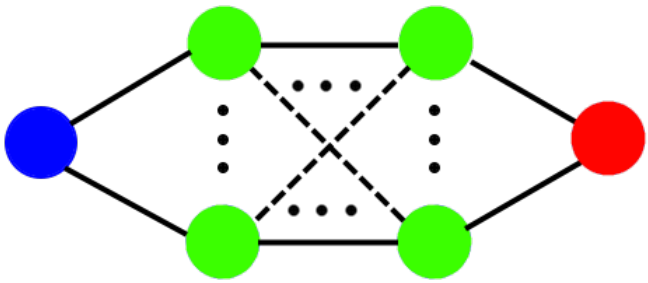


Figure 3.1: Schematic picture of the neural network used. Blue node is the input node and is connected to the hidden layers seen in green. These are connected to the red output node.

3.2 Flow cases

The first step in training the model is to create a initialization dataset. The creation of this dataset requires a choice of what flow cases will be included in the training process. In theory as many cases as the user wants can be trained against but in this project only channel flow and boundary layer flow was considered.

3.2.1 Channel flow

The first model was trained to work as a wall model for fully developed channel flow. Channel flow is driven by a pressure gradient between to infinite parallel plates using periodic boundaries seen in figure 3.2. Since the plates are of infinite size the flow does not vary in the z-direction, out of the paper in the figure. And since the flow is considered fully developed is also does not vary in the x-direction making this a 1-dimensional problem.

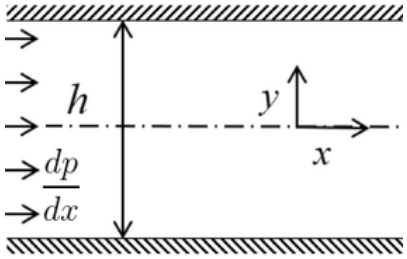


Figure 3.2: Fully developed channel flow.

3.2.2 Boundary layer flow

The second model created was trained on boundary layer flow data. Where a flow with a velocity u hits a infinitely long and wide plate. A boundary layer develops along the plate and will continue to grow further down the plate. This makes the problem 2-dimensional which makes it a natural next step to the channel flow case. The dataset does not include data near the leading edge of the plate as the turbulence in the simulation is modeled the beginning contains a lot of error from the turbulence model. To avoid this a initial simulation was run and at a certain x-position the flow was taken as a initial condition for an other simulation. Data from the second simulation was used to create the dataset. The case is illustrated in figure 3.3.

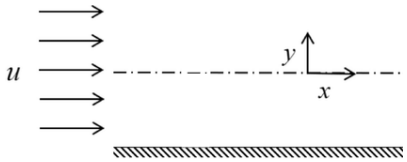


Figure 3.3: Boundary layer flow.

3.3 Training process

The training and testing of the neural network will be broken down into three steps. First the initialization step, then the CFD coupled training and lastly the testing and validation of the model. Before the training starts all the data is scaled by removing the mean and dividing by the variance and then turned into pytorch tensors. The training data was then split into two sets, 80% of the data as training data and 20% as testing data. The training will be done using the library pytorch. Pytorch is a easy to use library and is also very optimised and can run on graphics cards which decreases the required training time.

3.3.1 Initialization training

The initial training of the network will be done using regular supervised learning. The network will be trained against DNS data taken from channel flow with $Re = 5200$. This was done for both models even the model trained to work on boundary layer flow. DNS data is only taken for values of y^+ between $5 < y^+ < 250$ as range covers the usual y^+ values for wall functions. The initialization step is done using regular supervised learning using the least mean square loss function,

$$C_1(x) = \frac{1}{2}\sum_i (y_i - \hat{y}_i(x))^2.$$

The sum runs over all the data in the training set where y_i is the correct output and $\hat{y}_i(x)$ is the predicted output from the network. In the networked developed in this report the prediction $\hat{y}_i(x)$ is the predicted velocity U^+, while $y_i(x)$ is the correct velocity. A L2 regularization was also added to the loss function to avoid overfittning

$$C_2 = \lambda \sum_i \beta_i^2.$$

This sum runs over all the weights and biases in the network and adds up the square of their magnitudes with scale factor λ. A value of 0.005 was chosen for this project. This will penalize large weights which often is a sign overfitting. The total loss function which will be minimized can then be written as

$$C(x) = C_1(x) + C_2 = \frac{1}{2}\sum_i (y_i - \hat{y}_i(x))^2 + \lambda \sum_i \beta_i^2. \tag{3.1}$$

A pseudocode implementation of the training function is seen below.

```
function epoch_train(train_data, model):
    # Get correct output from train data
    y = train_data.out

    # Get predicted output from neural network
    # by feeding training input data through network
    y_pred = model(train_data.in)
```

```
    # Calculate the loss between predicted output and
    # correct data output
    cost = loss_function(y_pred, y)

    # Add l2 regularization to minimize overfitting
    for p in model.parameters():
        loss += lambda * pow(p, 2)

    # Update weights and biases in network
    model.optimize()

    return cost
```

3.3.2 CFD coupled learning

The CFD coupled training will incorporate a CFD solver into the training of a neural network (NN). The solver used in this work in pyCALC written by Davidsson [19]. The initialized network from the previous section will be the input to this algorithm. The training will then continue in an unsupervised learning method. Each epoch the model uses pyCALC-RANS including wall functions using U^+ and y^+ predicted by the NN. This gives a wall shear stress when converged. This result is then compared to the target which is the wall resolved RANS simulation of the same case, which also gives the wall shear stress when converged. This error is then back propagated through the network. The same criterion, optimized and l2-regularization is used as in the initialization but this process is only run for 100 epochs. The code for the training method is seen below, the difference between this method and the initialization train methods is only the beginning CFD part.

```
function epoch_train(train_data, model):
    # Get correct output from train data
    y = train_data.out

    # Run CFD using the current iteration of the
    # neural network as wall function
    uplus, yplus = CFD(model)

    # Find matching yplus in the wall resolved simulation
    yplus_pred = min(abs(y - yplus))

    # Get the uplus at this yplus
    y_pred = uplus[yplus_pred]

    # Calculate the loss between predicted output and
    # correct data output
    cost = loss_function(y_pred, y)
```

```
    # Add l2 regularization to minimize overfitting
    for p in model.parameters():
        loss += lambda * pow(p, 2)

    # Update weights and biases in network
    model.optimize()

    return cost
```

3.3.3 Testing and validation of model

When the training process is done a wall model is obtained. To test the performance of the model and if the CFD coupling improved the results both these simulations are run. The first cell y^+ values are also varied to see how the model behaves in these cases. The Reynolds number of the flow case should also be varied to ensure that the model has not been overfitted to only work for one Reynolds number. When this process is done the model should be tested on flow cases it has not been trained on to see how it behaves.

4

Results

4.1 Initial training of network

The initial training of the network was done on channel flow DNS data, at a Reynolds number of 5200. The loss function of the initial network as a function of epochs is shown in figure 4.1. The loss drops rapidly in the early epochs before slowing down but still dropping all the way till the training stopped. It also shows that the network is learning the data well as the testing data is decreasing together with the training data. As the losses are still decreasing when the training was stopped it is possible to train the network for longer but the fit was deemed well enough especially as the next step in the training process will train the model further.

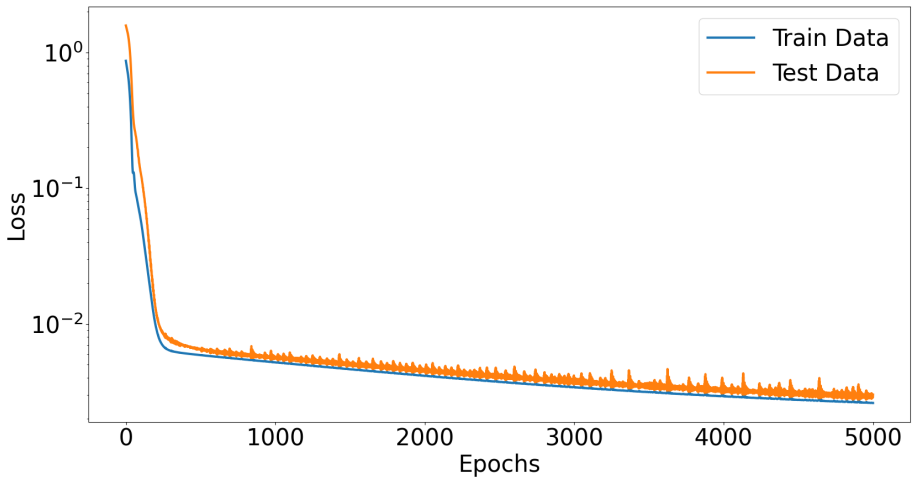


Figure 4.1: The loss as a function of epochs during the initial training process.

The resulting fit of the training data is shown in figure 4.2 and this results confirmed the discussion of the loss figure. The fit follows the training data well and is deemed good enough to continue to the next step. However it can be seen that for lower y^+ values the network does give worse results but these values should never be used since the first cells will be placed at higher y^+. The x-axis is also logarithmic which will compress the results and it is harder to see variations from the training data at higher y^+ values.

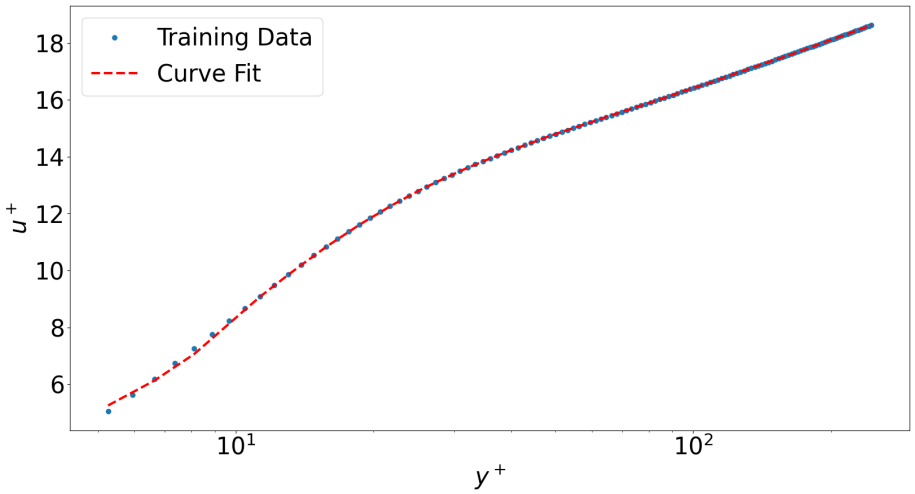


Figure 4.2: The training data and resulting curve fit from the initial training process.

4.2 Channel flow

The CFD coupled training for the channel flow case was preformed using a mesh where the first cell was placed at approximately $y^+ = 60$. The mesh contained three cells in the x-direction and a periodic boundary condition was used for the east and west boundaries. The Reynolds number used was 5200. The loss function for the CFD coupled training process in the channel flow case is seen in figure 4.3. The loss curve is oscillating with a large magnitude in the beginning of the training process but tends towards convergence at later epochs. Again the loss is still decreasing at the end of the training process but is deemed a reasonable stopping location to also prevent overfittning. In figure 4.4 the predicted wall shear stress as well as the wall resolved RANS solution is seen as a function of epoch. The oscillation in the beginning is seen but a convergence has been reached at the end. Interestingly the model shear stress does not converge towards the wall resolved solution but gets close as expected.

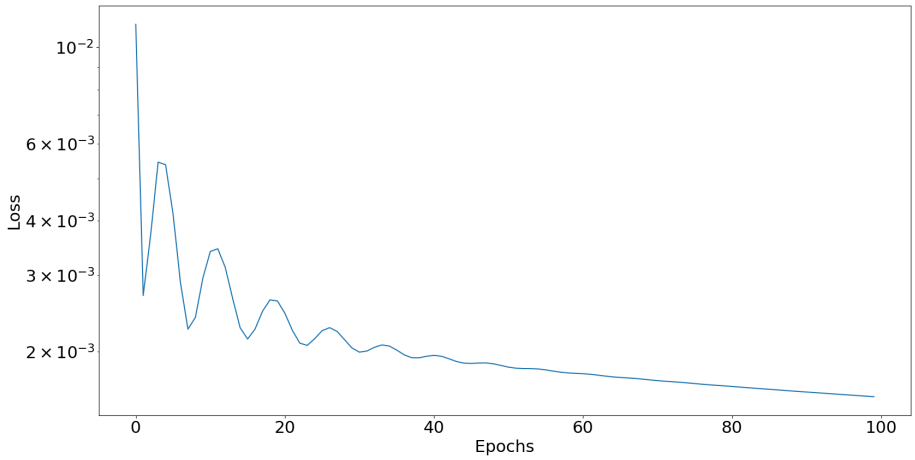


Figure 4.3: The loss as a function of epochs during CFD coupled training.

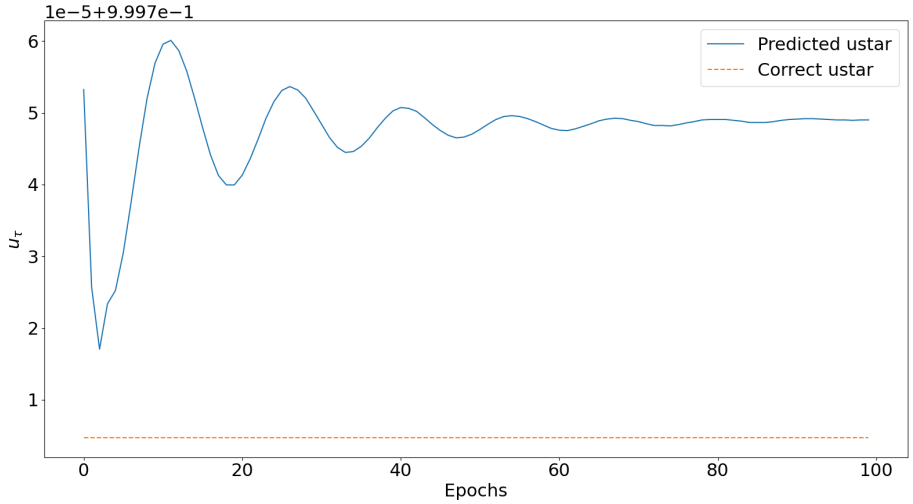


Figure 4.4: τ_w as a function of epochs during CFD coupled training.

4.3 Model performance at different first cell y^+

The height of the first cell was varied to see how the model accuracy changed with values it was not trained on. The four meshes used and their fist cell y^+ values are seen in table 4.1. Figures 4.5, 4.6, 4.7, 4.8 show velocity profiles for the model at different first cell y^+. The plots also contains DNS data, the wall resolved RANS solution, a wall modeled RANS with Reichardt's model and a wall modeled RANS using the initialization network as the wall model. It is seen that the network preforms worse for first cells closer to the wall also not outperforming the initialization network. The reason for this is a oversight in the training process. In the CFD coupled training only one mesh is used namely the mesh which has the first cell at $y^+ = 64$. So the models has been overfitted towards that mesh. But using the $y^+ = 64$ mesh the network does give results that are close to the wall resolved solution and beating Reichart's. To improve the performance of the model it should be trained on a set up meshes using different y^+ values. However this will also increase the training time by a large factor.

Case	Mesh 1	Mesh 2	Mesh 3	Mesh 4
First cell y^+	33	44	64	103

Table 4.1: First cell y^+ for the different meshes used to test model preformance.

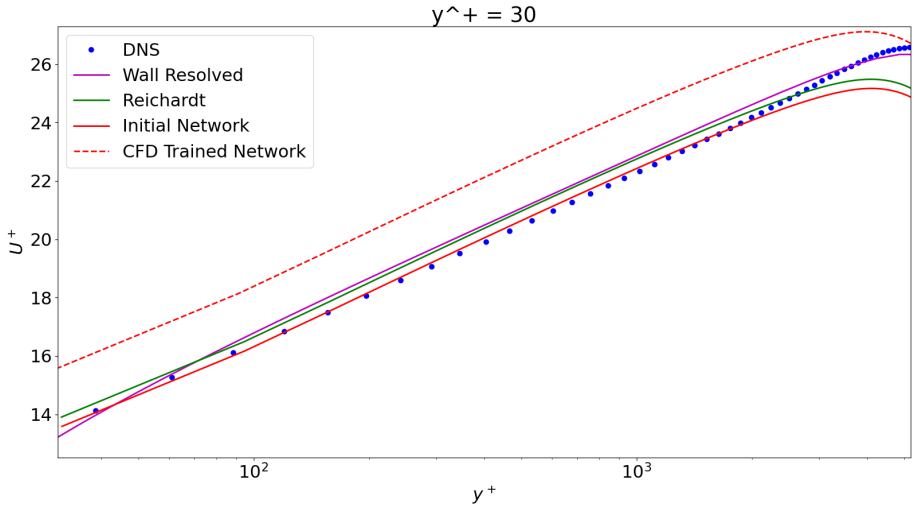


Figure 4.5: Velocity profile for first cell placed at $y^+ = 30$ for different wall functions.

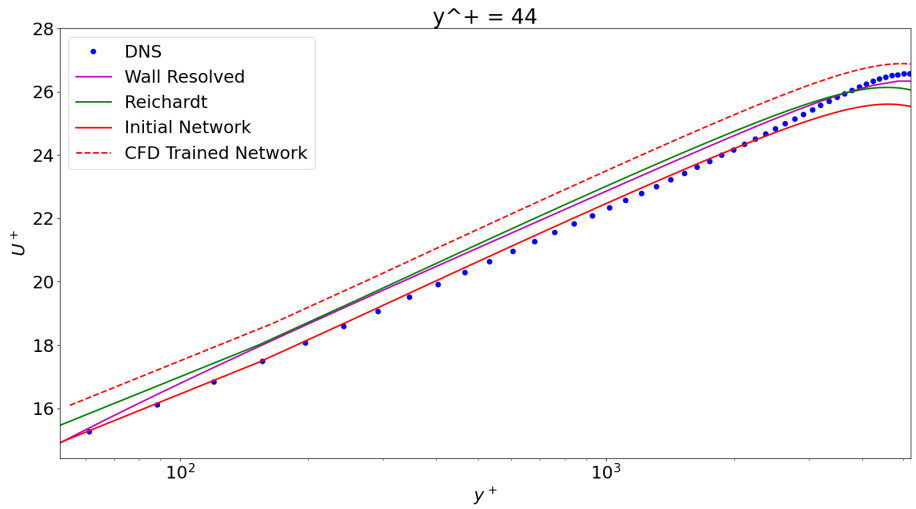


Figure 4.6: Velocity profile for first cell placed at $y^+ = 44$ for different wall functions.

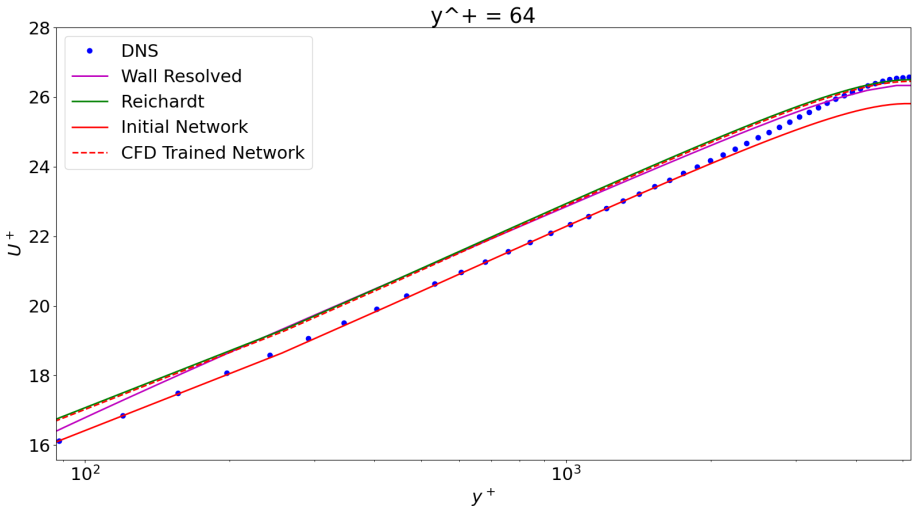


Figure 4.7: Velocity profile for first cell placed at $y^+ = 64$ for different wall functions.

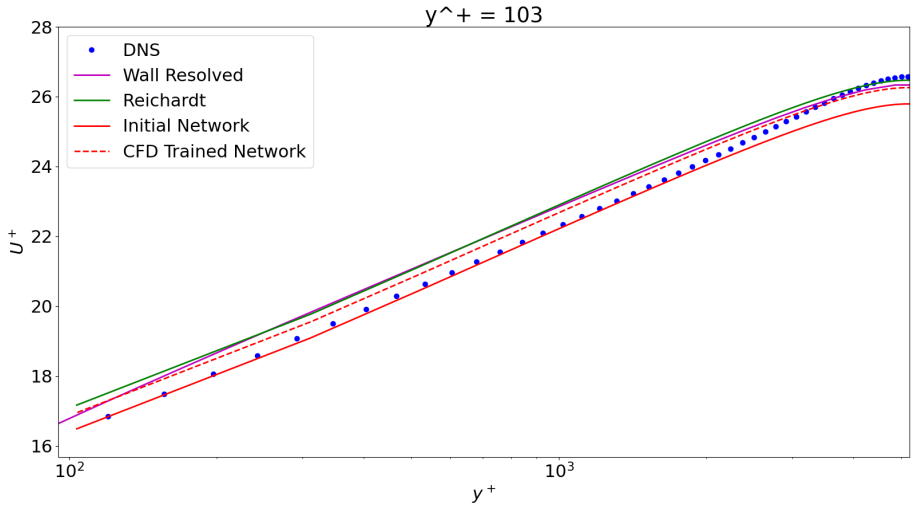


Figure 4.8: Velocity profile for first cell placed at $y^+ = 103$ for different wall functions.

4.4 Boundary layer flow

The inlet data was sampled from at $Re_\theta = 2200$ from a initial RANS simulation to reduce errors from the turbulence model. The northern and eastern boundary assumes fully developed flow and no gradient in the y-direction. The velocity at the eastern boundary is modified to satisfy continuity. The mesh used in the wall modeled training simulations had a first cell $y^+ \sim 30$. The mesh then grows in the y-direction with a growth rate of 1.05. A curve of how the resulting first cell y^+ varied with distance along the plate is seen in figure 4.9.

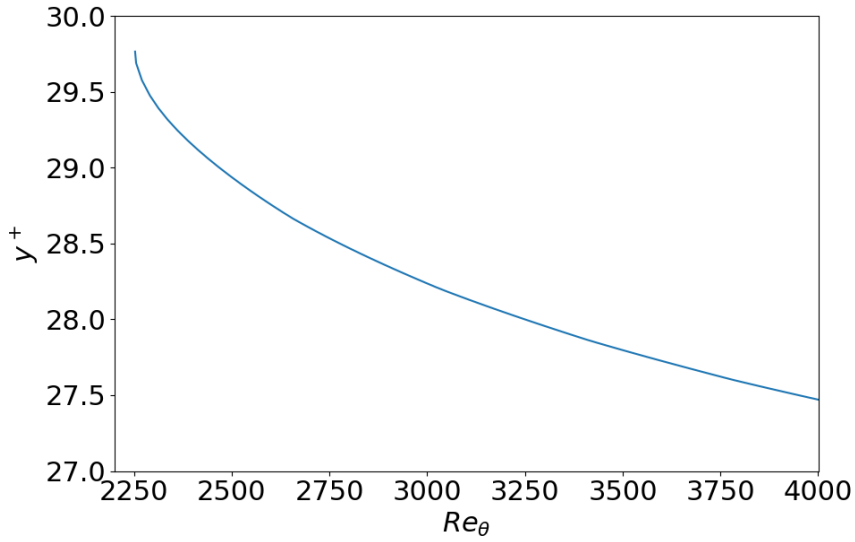


Figure 4.9: First cell y^+ changing along the plate.

The resulting loss function during the training of the boundary layer network is shown in figure 4.10. As in the channel flow case the values oscillate with large amplitudes in the beginning before finding a stable solution at the end. This is the most converges model as the loss is constant for the last epochs. In figure 4.11 C_f as a function of Re_θ is shown for DNS data, wall resolved RANS, wall modeled RANS with initial network and four epochs of the training network. The oscillations seen in the loss functions are also present here and the last iteration of the training process does give a more accurate result then the initial network. The comparison with Rechardt's can be seen in figure 4.12, where DNS and wall resolved RANS is shown as well. The ML based wall functions are more accurate than Rechardt's and it is seen that the CFD coupled training also increases the accuracy further. The first cell y^+ has not been varied but this model also suffers from the same overfitting problem as the channel flow case.

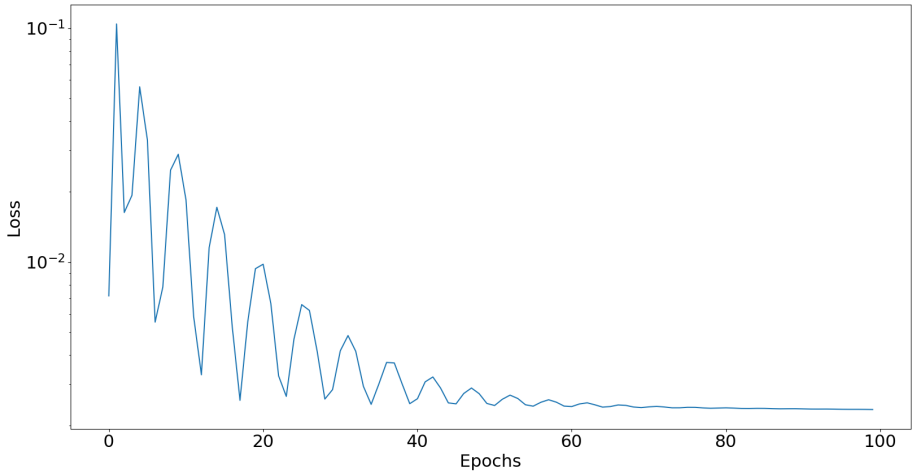


Figure 4.10: The loss as a function of epochs during CFD coupled training.

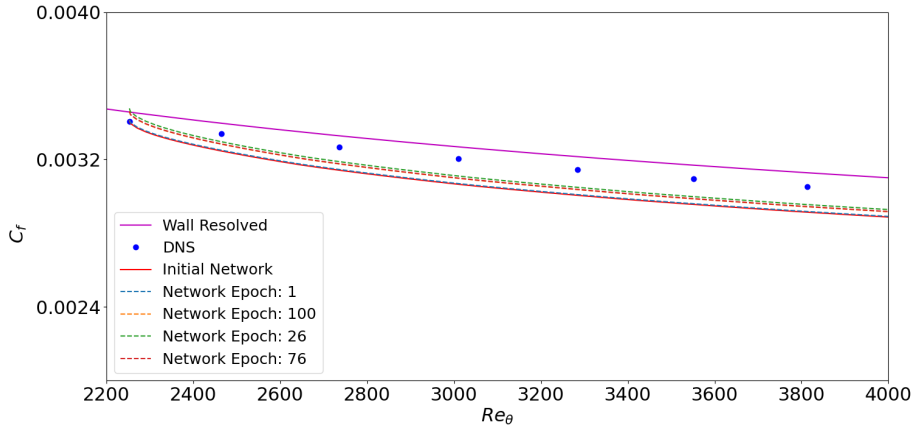


Figure 4.11: C_f at four training steps as a function of Re_θ during CFD coupled training.

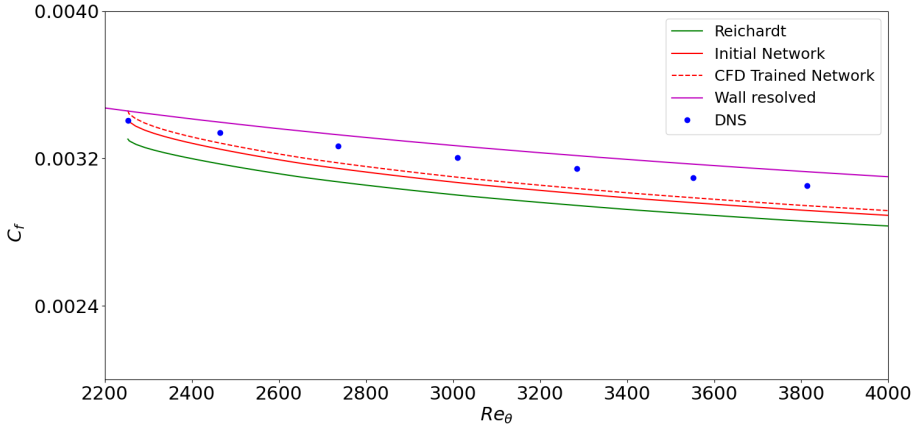


Figure 4.12: Resulting C_f as a function of Re_θ for different wall functions.

5
Conclusions

Wall functions and specifically machine learning based wall functions do give good results and machine learning methods can even outperform classical models such as Reichardt. Coupled CFD learning also shows potential especially the boundary layer results seem promising where the results do clearly improve over the initialization model. It also gives better results than Reichardt which shows the potential of the method.

One problem with machine learning based models and specifically CFD coupled learning is the training time required. Both the cases used in this work the training time was over one minute per iteration and the total training time was around 2 hours for each model. However looking at the loses during the training process it can be seen that maybe not that many iterations are needed for a finished model. The last 50 epochs in both model did not decrease the loss by a large amount. This is promising and shows that not thousands of epochs are needed to reach a good model. But to fix the issue of mesh overfitting encountered in this project multiple simulation needs to be run per epoch which can further increase per epoch run time. These simulations can be run in parallel and the increase in time wont be that bad after all.

If the CFD coupled learning process is to be repeated on LES simulations the training time will increase dramatically. Every simulation will take up to multiple hours or even days which would probably require cluster to run efficiently. But the training only has to be done once which still makes the idea achievable if it does increase the accuracy.

No LES simulations have been tested in this project which was the long term goal and purpose of this project. In LES simulations the computational speed up is necessary. But it was decided to stick to RANS simulations since they are fast to run to narrow out issues and bugs that arose during the project. With these learning's a version working for LES simulations could be implemented and tested given the required computing power.

5.1 Further studies

A lot of work still remains undone these models have not had a formal sweep of Reynolds numbers to ensure they function equally well for different flow conditions.

There is a risk that the models could be overfitted in Reynolds number as they are with the mesh. This needs to be explored before more complicated flow cases can be considered. When this these two problems have been corrected the next step is to try more complicated flow cases and networks with more inputs. 2-dimensional flows such as a bump or diffusor where recirculation can be present is where most classical and ML based methods break down. The implementation of these cases in the training process should not be difficult since they share characteristics with the boundary layer flow.

There is also more to be learned in how these models operate. The ML based wall functions does work well but there is a loss of knowledge in how they work. In this project a simple network with one input and one output is used which does not make this problem to apparent. But if the number of input parameters are increased and the network size grows at the same time it is going to get harder to understand what the model is truly doing. It is important to test and come up with methods to understand how these models work. This is important to understand and maybe find new physics or classical models.

Bibliography

[1] T, Sanjeeb. Bose and George Ilhwan Park. "Wall-Modeled Large-Eddy Simulation for Complex Turbulent Flows", Annual Review of Fluid Mechanics 2018, doi: https://doi.org/10.1146/annurev-fluid-122316-045241.

[2] J. LARSSON et al. "Large eddy simulation with modeled wall-stress: recent progress and future directions", *Mechanical Engineering Reviews* 2015. doi: 10.1299/mer.15-00418.

[3] D. R, Chapman. 1979 Computational aerodynamics development and outlook. AIAA J.

[4] H K, Versteeg and W, Malalasekera. An Introduction to Computational Fluid Dynamics (2007). Pearson Education Limited.

[5] L, Davidsson. "Fluid mechanics, turbulent flow and turbulence modeling" 2024 url: `https://www.tfd.chalmers.se/~lada/postscript_files/solids-and-fluids_turbulent-flow_turbulence-modelling.pdf`

[6] J, Smagorinsky. General circulation experiments with the primitive equations. Monthly Weather Review, 99–165, 1963 doi: `https://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2`.

[7] M, Germane, et al. "A dynamic subgrid-scale eddy viscosity mode". *Physics of Fluids A: Fluid Dynamics* 1991 doi: `https://doi.org/10.1063/1.857955`.

[8] J, Jiménez. 2012. Cascades in wall-bounded turbulence. Annu. Rev. Fluid Mech. 44:27–45. doi:10.1146/annurev-fluid-120710-101039.

[9] H, Choi & P, Moin. "Grid-point requirements for large eddy simulation: Chapman's estimates revisited". *Physics of Fluids* 2012. doi: 10.1063/1.3676783.

[10] H.Reichardt, ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik Volume 31, Issue 7, pages 208–219, 1951.

[11] W, Cabot, P, Moin. Approximate Wall Boundary Conditions in the Large-Eddy Simulation of High Reynolds Number Flow. Flow, Turbulence and Combustion 63, 269–291 (2000). `https://doi.org/10.1023/A:1009958917113`

[12] A, Vadrot. X, Yang. M, Abkar. "A survey of machine learning wall models for large eddy simulation". Phys. Rev. Fluids 2023. doi: `10.1103/PhysRevFluids.8.064603`

[13] X. L. Huang, X. I. A. Yang, and R. F. Kunz, Wall-modeled large-eddy simulations of spanwise rotating turbulent channels—Comparing a physics-based approach and a data-based approach, Phys. Fluids 31, 125105 (2019).

[14] Z. Zhou, G. He, and X. Yang, Wall model based on neural networks for LES of turbulent flows over periodic hills, Phys. Rev. Fluids 6, 054610 (2021).

[15] A. Lozano-Duran & H. J .Bae. Self-critical machine-learning wall-modeled LES for external aerodynamics. Center for Turbulence Research Annual Research Briefs. 2020. https://web.stanford.edu/group/ctr/ResBriefs/2020/20_LozanoDuran.pdf.

[16] L. P. Kaelbling, M. L. Littman, A. W. Moore. "Reinforcement Learning: A Survey". Journal of Artificial Intelligence Research. 4: 237–285 (1996). doi:10.1613/jair.301.

[17] H, J, Bae. P, Koumoutsakos. Scientific multi-agent reinforcement learning for wall-models of turbulent flows. (2022). https://doi.org/10.1038/s41467-022-28957-7

[18] H. J. Bae, P. Koumoutsakos. Wall Modeling of Turbulent Flows with Various Pressure Gradients Using Multi-Agent Reinforcement Learning. (2023) doi: https://doi.org/10.48550/arXiv.2305.02540

[19] L. Davidsson. pyCALC-RANS: A Python Code for Two-Dimensional Turbulent Steady Flow. (2023) https://www.tfd.chalmers.se/~lada/postscript_files/py-calc-rans.pdf.

Department of Mechanics and Maritime Sciences Division of Fluid Dynamics
Chalmers University of Technology
Göteborg, Sverige
www.chalmers.se

CHALMERS