Master Thesis

Variational Multiscale Method coupled with an Artificial Neural Network for 3D turbulent channel flow

by

Mohan Mohith Kanala

To obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday, April 24, 2023 at 2:30 PM.

Student number:	5306515	
Project duration:	November 2, 2021 – March 21, 2023	
Thesis Committee:		
Supervisor:	Dr. S.J. Hulshoff	TU Delft
Chair:	Dr.ir. B.W. van Oudheusden	TU Delft
Examiner:	Dr. M. L. Baptista	TU Delft

TU Delft

Preface

As I near the end of my journey as a Master's student, I am overwhelmed with gratitude and appreciation for all those who have supported and helped me along the way. Firstly, I want to express my deepest gratitude to my parents and brother. Their unwavering love and encouragement have been the driving force throughout my academic journey. I owe every accomplishment of mine to their guidance and support. Presenting this thesis is not just my achievement, but theirs too.

I would also like to extend my sincere thanks to Dr Steven Hulshoff, my thesis supervisor. Steve's guidance, encouragement, and support have been instrumental in shaping my research and thesis. His invaluable insights and feedback have always motivated me to keep working. Additionally, I am grateful to Sharath and Andrea for their unwavering support and assistance, which helped me in my research work.

Moreover, my time in Delft has allowed me to meet some incredible people, and I am grateful for the friendships. Thank you, Sam, Ashwin, and Ravi for making my academic journey enjoyable, memorable, and full of laughs. I would also like to take this opportunity to send my love and appreciation to all my friends and family back home. You have been my constant support system, and I cannot thank you enough.

Finally, I want to express my gratitude to anyone who has taken the time to read this. I hope my thesis is an enjoyable and informative read for you. Once again, thank you to everyone who has played a part in shaping my academic journey.

Abstract

Turbulence is a ubiquitous phenomenon in nature, characterized by complex and unpredictable fluid motion that is challenging to simulate. Turbulence modelling aims to create mathematical models that can predict turbulent flow behaviour in a given system, and there are various types of models available, including Reynolds-Averaged Navier-Stokes (RANS), Large Eddy Simulation (LES), and Direct Numerical Simulation (DNS). LES provides high-fidelity predictions of turbulent flow phenomena by resolving most of the energetic scales of turbulence, making it a promising candidate for accurate predictions. However, LES has limitations, particularly in capturing near-wall phenomena, necessitating fine meshes to resolve these regions. Deep learning, a sub-field of machine learning, has been actively used to improve the shortcomings of traditional LES models, particularly in predicting subgrid-scale (SGS) terms. This thesis aims to explore the extension of existing multilayer perceptron (MLP) from previous studies to the case of 3D turbulent channel flow and compare net hyperparameters, feature sets, and training data to find the most compact model that can accurately reproduce the closure terms. The study will build on previous work at TU Delft on LES-VMM and focus on developing an effective network with a simple architecture.

Contents

1 Introduction **1**
1.1 Turbulence Models . . . 1
1.2 LES and SGS models . . . 1
1.3 Deep Learning . . . 2
1.4 Previous work at TU Delft . . . 3
1.5 Research Objective . . . 3
1.6 Research Questions . . . 3
1.7 Thesis Outline . . . 3

2 ANN in Turbulence Modelling **5**
2.1 Neural Networks in Turbulence . . . 5
2.2 Deep Learning for RANS. . . . 6
2.3 Deep Learning for LES . . . 7
2.4 Other Approaches . . . 10
2.5 Reflection . . . 11

3 Theory **12**
3.1 Large Eddy Simulations (LES). . . . 12
3.1.1 Subgrid-scale Modelling . . . 13
3.2 Weak Form of the Navier-Stokes Equation . . . 14
3.2.1 Starting Point . . . 14
3.3 Variational Multiscale Modelling (VMM) . . . 15
3.3.1 VMM in LES . . . 16
3.4 Formulation of the closure terms . . . 16
3.4.1 Residual Based LES . . . 16
3.4.2 DNS reference data notation. . . . 18

4 Previous work at TU Delft **20**

5 Methodology **22**
5.1 Main Test Case . . . 22
5.2 Training Data . . . 22
5.2.1 Closure . . . 23
5.2.2 Feature Set . . . 23
5.2.3 Stencil. . . . 23
5.2.4 Boundary Conditions . . . 24

6 Results **25**
6.1 Baseline Network. . . . 25
6.1.1 Network Hyperparameters . . . 25
6.1.2 Network Structure . . . 25
6.1.3 Training and Validation . . . 26
6.2 Increasing the training data . . . 27
6.2.1 Reflection . . . 31
6.3 Phase II: Post-Normalization Correction . . . 31
6.3.1 Neuron-Hidden Layer Analysis . . . 31
6.3.2 Feature Set (NO U, Ut). . . . 33
6.3.3 Stencil Studies . . . 34
6.3.4 Batch Size Variation . . . 34
6.3.5 Reflection . . . 35

6.4 Phase III: Post-Batch Size Correction . . . 36
6.4.1 Neuron-Hidden Layer Analysis . . . 36
6.4.2 Data-Set Size Variation. . . . 37
6.4.3 Epochs . . . 39
6.4.4 Reflection . . . 40

7 Conclusions 42
7.1 Research Questions and Answers. . . . 43
7.2 Recommendations . . . 44

A Derivation of the closure terms 45

List of Figures

1.1 A simple Neural Network (Tracey *et al.* (2015)) 2

2.1 Reduction in wall-shear stress for various control schemes, · · · offline trained control scheme, — online trained (Lee *et al.* (1997)) 5
2.2 Reconstruction performance. From left to right: original sample, linear reconstruction, and neural net reconstruction (Milano & Koumoutsakos (2002)) 6
2.3 Comparison of source terms; inferred vs neural network prediction (Duraisamy & Durbin (2014)) 7
2.4 Prediction; Skin Friction (Duraisamy & Durbin (2014)) 7
2.5 Contours of velocity (u) fluctuations (Völker *et al.* (2002)) 8
2.6 turbulent viscosity coefficient (c_s) at $y^+ = 10$ (a) Neural Network (b) Bardina SSM (c)u'w' component of Reynolds stress tensor (Sarghini *et al.* (2003)) 8
2.7 CNN with transfer learning and encoder-decoder (Guan *et al.* (2022)) . 9
2.8 Architeture of D-ANN (Yuan *et al.* (2022)) 10
2.9 Schematic of PINNMeng *et al.* (2020) 11

5.1 3D Turbulent Channel Flow 22
5.2 Output File 23
5.3 Visualisation of the element stencil 24

6.1 Training and Validation loss (Baseline Network) 26
6.2 X and Y Momentum Terms (Baseline Network) 27
6.3 Z Momentum and Continuity Terms (Baseline Network) 27
6.4 Training and Validation loss (Baseline – 2 Time Steps) 28
6.5 X and Y Momentum Terms (Baseline – 2 Time Steps) 28
6.6 Z Momentum and Continuity Terms (Baseline – 2 Time Steps) 29
6.7 Continuity Terms (Baseline Network) 29
6.8 Z Momentum Terms (Baseline Network) 30
6.9 Fixing Normalization (Baseline Network – 4 Time Steps) 30
6.10 Change in Loss graph (Baseline Network – 4 Time Steps) 31
6.11 Effect of network structure optimization: Training and Validation Loss – 4 Time Steps 33
6.12 Effect of network structure optimization: X-Momentum Terms – 4 Time Steps 33
6.13 Effect of batch size variation: Training and Validation Loss – 4 Time Steps, Two Hidden Layers, 2000 Neurons 35
6.14 Effect of batch size variation: X-Momentum Terms – 4 Time Steps, Two Hidden Layers, 2000 Neurons 35
6.15 Training and Validation loss – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size–750 38

6.16 X and Y Momentum Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size–750 38
6.17 Z Momentum and Continuity Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size–750 38
6.18 Training and Validation loss – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size – 750, – 1000 Epochs 39
6.19 X and Y Momentum Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size – 750, – 1000 Epochs 40
6.20 Z Momentum and Continuity Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size – 750, – 1000 Epochs 40

1 Introduction

Turbulence is a complex phenomenon characterised by chaotic and unpredictable motion. It occurs nearly everywhere in nature, from the airflow around an aircraft to cumulus clouds, rivers, oceans (Sreenivasan (1991)), and even the flow of blood in arteries (Ghalichi *et al.* (1998)). It is highly non-linear and involves energy transfer across various scales, from large-scale motions of eddies to small-scale turbulent fluctuations, making it challenging to simulate.

1.1. Turbulence Models

Turbulence modelling refers to creating mathematical models by developing equations and algorithms to predict the behaviour of turbulent flows in a given system. There are various types of turbulence models, including Reynolds-Averaged Navier-Stokes (RANS) models, large-eddy simulation (LES) models, and direct numerical simulation (DNS) models.

Reynolds-Averaged Navier-Stokes (RANS) equations are a set of partial differential equations describing the motion of time or ensemble-averaged mean flow. They are obtained by decomposing all instantaneous quantities into averaged and fluctuating components in a turbulent flow field (Breuer *et al.* (1995)). The solution of the RANS equations helps predict the mean flow properties, such as velocity, pressure, and temperature. However, the RANS models do not adequately represent the effects of the unsteady and turbulent flow phenomena, such as vortex shedding and flow separation, which can be important in many engineering applications. Therefore, advanced turbulence modelling techniques, such as LES and DNS, are used to account for these effects.

Direct Numerical Simulation (DNS) is a computational method used to numerically solve the Navier-Stokes equations (Equation 3.1), which describe the motion of a fluid (Besson & Laydi (1992)). It attempts to solve the equations directly without making any assumptions. However, it is computationally expensive because it requires resolving all the scales of motion associated with turbulence, down to the Kolmogorov scale (Georgiadis & DeBonis (2006)).

Large Eddy Simulations (LES), on the other hand, is a CFD technique that provides high-fidelity predictions of turbulent flow phenomena by resolving most of the energetic scales of turbulence (Feng *et al.* (2020)). Furthermore, it directly computes the large-scale motions (large eddies) of turbulent flow and models only the small-scale (sub-grid scale (SGS)) motions (Zhiyin (2015)). LES bridges the gap between RANS and DNS compared to the other CFD methods.

1.2. LES and SGS models

LES models are based on phenomenological assumptions because the physical equations that govern fluid flow are not entirely solvable. Therefore, numerical methods are used to solve the equations, and assumptions such as equilibrium in turbulent scales and isotropy in smaller scales are often made to simplify the modelling process. The resulting models provide closure for the equations governing large-scale structures and allow for accurate predictions.

One of the limitations of LES is that it requires a fine mesh in certain areas, like near-wall regions. This is because the turbulent structures are anisotropic and strongly influenced by viscous forces, which models often fail to capture accurately. Fine meshes capture the sharp velocity gradients and accurately represent the wall shear stress. Similarly, capturing laminar-turbulent transition requires high-resolution meshes, especially near the wall, where the initial instabilities and subsequent turbulent structures form.

In addition to phenomenological assumptions, existing SGS models also come with their own set of shortcomings. For instance, the Smagorinsky (Smagorinsky (1963)) and Dynamic Smagorinsky (Germano *et al.* (1991)) models are stable but have low correlation with exact subgrid-scale stresses. While the Scale Similarity (Bardina *et al.* (1980)) and Approximate Deconvolution (Stolz *et al.* (1999)) Models have higher correlation but are unstable as they do not drain an adequate amount of energy from the resolved scales.

The mathematical requirements of the commutation of filtering and differentiation must be satisfied in the LES approach. When these conditions are not met, LES accuracy is compromised, which led to the development of the Variational Multiscale Method (VMM). The concept of VMM was introduced by Hughes (1995), and it involves the a prior division of the velocity field (u) into resolved and unresolved scale velocities ($\bar{u}$ and u'). Unlike traditional LES, the VMM does not use an explicit filtering scheme but uses a projector (P) instead. VMM considers the discretisation error and representation of unresolved (SGS) scales as the same. It thus avoids having separate sources of model and discretisation error and considers only the approximate model of unresolved scales.

1.3. Deep Learning

Deep learning is a sub-field of machine learning and artificial intelligence that encompasses training artificial neural networks in tasks that typically involve recognising patterns in data and learning the correlation among two or more variables (Chassagnon *et al.* (2020)). Neural networks carry out the data processing in these algorithms and form the backbone of deep learning (Jindal *et al.* (2022)). Deep refers to neural networks used in deep learning, typically having many layers of interconnected nodes that allow them to model complex relationships and hierarchies in the data.

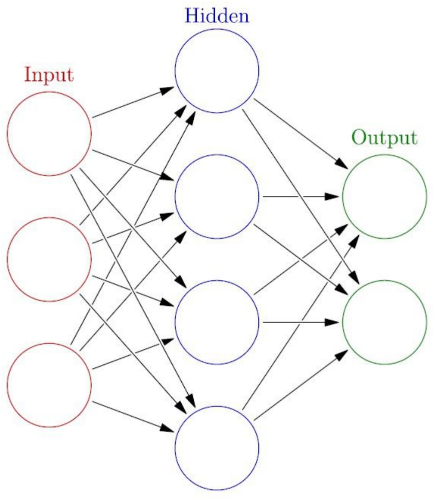


Figure 1.1: A simple Neural Network (Tracey *et al.* (2015))

ANNs are actively used to improve the shortcomings of traditional LES models. Studies like Guan *et al.* (2022) and Yuan *et al.* (2022) show data-driven SGS models outperforming traditional models with higher correlation to exact SGS and reconstructing the velocity spectra and kinetic energy flux. Non-linear function approximation capabilities and studies indicating that data-driven turbulence models are a promising alternative to conventional models are vital motivations for this study.

1.4. Previous work at TU Delft

Much work has been done at TU Delft concerning neural networks and turbulence modelling, particularly in the field of LES-VMM (section 3.3). Most of it revolves around either predicting model coefficients for turbulence (Durieux (2015) and Kurian (2019)) or finding the underlying correlation between resolved and unresolved scales (Robijns (2019)).

Recently, the focus in this area has been to have the neural network predict subgrid scale terms for an LES in an online setting. With research revolving around the stabilisation techniques, including Lagged Feature Set (LFS) and Backscatter Limitting (BL), to help stabilise the VMM-ANN combine (Janssens (2019),Pusuluri (2020), and Rajampeta (2021)). More of which shall be covered in detail in chapter 4. Most recent work, including that of Rajampeta (2021), focuses on a 1D model of turbulent channel flow. It would be of great interest to expand this VMM-ANN technique to 3D flow and understand how difficult it is to reproduce the closure terms.

1.5. Research Objective

Despite the recent developments and contributions of neural networks in turbulence seen in chapter 2, using neural networks in turbulence modelling remains computationally expensive. Building a simple architecture for our application would be particularly interesting in this context. Therefore, the focus of the study would be to find the simplest ANN architecture which could reproduce the closure terms with a correlation of approximately 99%.

A standard multi-layer perceptron (MLP) architecture with a sequential model was considered for our analysis. Furthermore, a low courant number was chosen for the flow. This meant the time step used in the simulation was relatively small compared to grid spacing and flow velocity, which minimised the influence of the specific time march chosen. The network will be trained offline as it is cost-effective with low memory and computational cost. Therefore the objective of the thesis is:

To design and compare feature sets to find the most compact model which can accurately reproduce the correction terms for 3D turbulent channel flow.

1.6. Research Questions

1. Can the neural network (structure and hyper-parameters) used for the 1D turbulent channel flow model (Rajampeta (2021)) be applied to the three-dimensional case?
2. How does the network structure affect the ANN's accuracy?
   - 2.1 What is the effect of widening the network?
   - 2.2 What is the effect of deepening the network?
3. Can the basic, macroscopic properties of the flow (velocity, pressure, and their gradients) be used as features to find the subgrid-scale terms?
   - 3.1 Which of the features are most important for high correlation?
4. What is the simplest ANN architecture which can accurately (99%) reproduce the closure terms?

1.7. Thesis Outline

chapter 2 will review developments and contributions to neural networks in turbulence modelling. chapter 3 will elucidate upon the necessary theory for our study, including basic information related to LES, subgrid-scale modelling, and VMM. chapter 4 will discuss the previous work within TU Delft related to our study and give context to our work. chapter 5 will describe the methodology followed to set up the baseline neural network for our analysis. chapter 6 covers the results obtained with the neural network and further steps undertaken to improve its performance. While chapter 7 talks of the conclusions from the network results and future recommendations.

2
ANN in Turbulence Modelling

The previous chapter outlined key concepts along with the research objective. This chapter will cover recent developments of neural networks in turbulence modelling as a literature review.

2.1. Neural Networks in Turbulence

One of the first attempts to use neural networks in turbulence was that of Lee *et al.* (1997) to control turbulence for drag reduction. The aim was to have wall actuation through distributed blowing and suction on the surface to change wall shear and reduce drag. The neural network was tasked with finding the correlation between wall-shear stress and wall actuation. The network was initially trained offline with high-fidelity controlled turbulence channel flow data and applied to a regular channel flow, resulting in an 18% reduction in skin friction. Later, the network was trained online with an adaptive inverse model control, and a simple control scheme with similar drag reduction was derived.

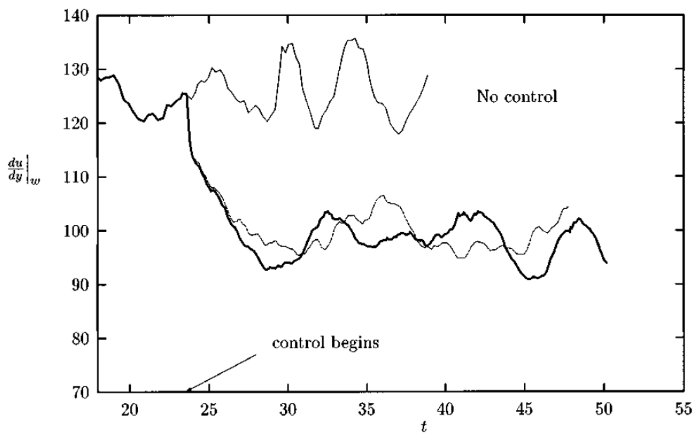


Figure 2.1: Reduction in wall-shear stress for various control schemes, ··· offline trained control scheme, — online trained (Lee *et al.* (1997))

The procedure resulted in a visible reduction of drag yet fell short in capturing activity across the turbulent spectrum. Additionally, as the study did not test the control scheme's generalizability, its applicability to higher flow regimes was unknown. Another significant milestone for data-driven turbulence modelling was the study of Milano & Koumoutsakos (2002), which aimed to find the viscous sublayer velocity profile using a neural network trained on wall-only information provided by DNS. The performances of linear and non-linear Principal Component Analyses (PCA) were compared to a Proper Orthogonal Decomposition (POD) for reconstructing and predicting a near-wall field. It was found that the non-linear PCA performed better than its linear counterpart and that, in general, the neural network was a better option for the task than POD. The author concludes by saying that **neural networks may be helpful for flow solvers using RANS or LES due to their compression capabilities and that any wall model should include large-scale information.**

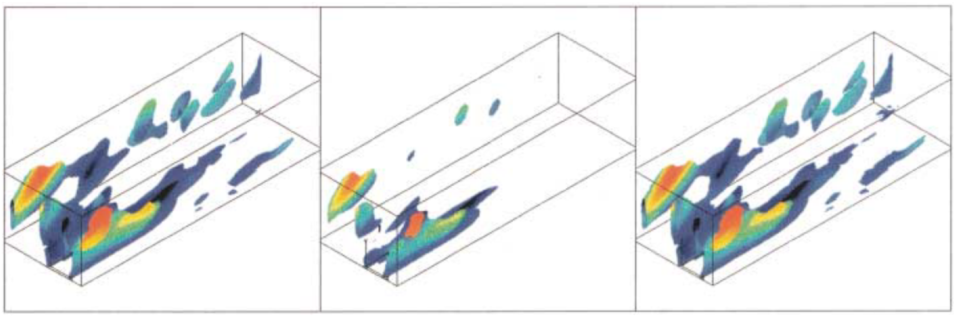


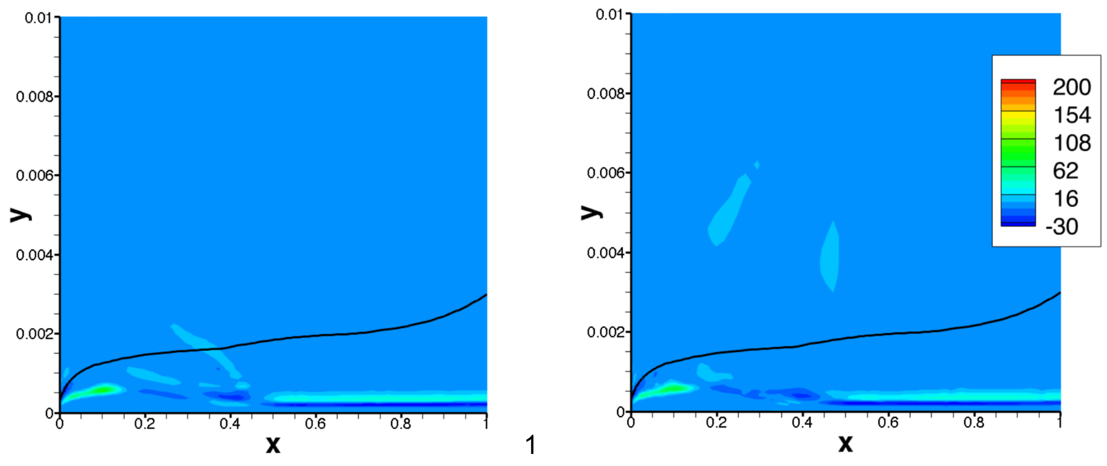
Figure 2.2: Reconstruction performance. From left to right: original sample, linear reconstruction, and neural net reconstruction (Milano & Koumoutsakos (2002))

In recent years, there has been a ”Cambrian explosion” in deep learning technology (Aggarwal (2018)), boosting efforts by researchers to create an ideal, generalisable data-driven turbulence model. This study has categorised these efforts into deep learning for RANS, LES, and other approaches.

2.2. Deep Learning for RANS

Duraisamy & Durbin (2014), Duraisamy *et al.* (2015) modelled bypass transition by multiplying the production term in the turbulence kinetic energy (k) equation of the k-ω model (Wilcox (1998)), with an intermittency factor (γ). The factor followed the condition that it shall be absent for laminar flow and increase with the degree of turbulence until unity, indicating turbulent flow. Inverse modelling and machine learning were used to predict the source term ($S_{\gamma}(\zeta)$) and skin friction coefficient (C_f), and their performances were compared with the bypass transition model proposed by Ge *et al.* (2014). The studies established the potential of inverse modelling and machine learning in transition modelling, which gave rise to Field Inversion Machine Learning (FIML).

Further, in Parish & Duraisamy (2016), as a part of the FIML framework, a correction factor was added to the production term in the k equation of the k-ω model. Point-wise corrections were then made to the factor based on observational DNS data from several problems, and a generalised factor was derived using machine learning. Finally, comparing the velocity profile's machine learning predictions with the prior distribution in different flow regions, it was observed that the ML model performed well only in the inner layer.



(a) Inferred term (b) Predicted term

Figure 2.3: Comparison of source terms; inferred vs neural network prediction (Duraisamy & Durbin (2014))

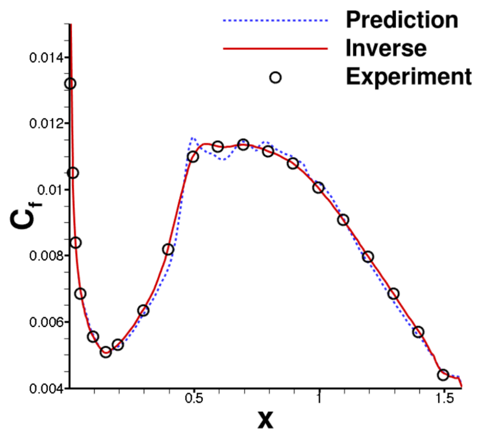


Figure 2.4: Prediction; Skin Friction (Duraisamy & Durbin (2014))

The poor performance in the outer layer can be attributed to the high variance of the machine learning approach, leading to an overestimation of eddy viscosity and an underestimation of velocity. Tracey *et al.* (2015) used the methodology of field inversion to predict the source terms of the Spalart-Allmaras model. The neural network was trained with DNS, LES, and experimental data to learn the underlying relation between the unresolved and source terms. Even for new cases, the model gave promising results, accurately predicting the C_f of a NACA 0012 airfoil in flat plate flow. Singh *et al.* (2017) also used the FIML framework to develop NN-enhanced corrections to the S-A model.

2.3. Deep Learning for LES

Völker *et al.* (2002) explored creating an ideal or optimal LES using ”stochastic estimation”. The test case considered for developing the ideal LES was turbulent flow in a channel, and the idea was to create a model applicable to various test cases. The proposed model was non-generalisable in flow regimes outside the Re used in the DNS training data and hence unfit for LES applications. Though the study could not meet its primary objective, it effectively pointed out the various criterion to be considered for developing subgrid-scale models for wall-bounded turbulence.

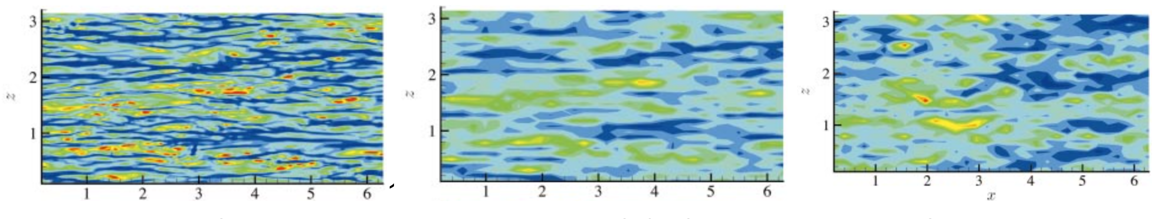

(a) DNS (b) Filtered DNS (fDNS) (c) LES3 (Data-driven)

Figure 2.5: Contours of velocity (u) fluctuations (Völker *et al.* (2002))

There are two approaches to deep learning research in LES; the first is to use the ANN to predict the model coefficients for a specific case, and the second is to discover the underlying relationship between the resolved and unresolved scales. Sarghini *et al.* (2003) was one of the first papers to inspire the use of ANNs in LES and is an example of the former approach. The model used was Bardina's scale similarity model (Bardina *et al.* (1980)) and the coefficient of interest, the turbulent eddy viscosity c_s. The test case was 2D turbulent channel flow at a Re of 180. The study succeeded in creating a model generalisable to other Re with the help of ANN re-training and suggested that **higher fidelity data like DNS be used for training**.

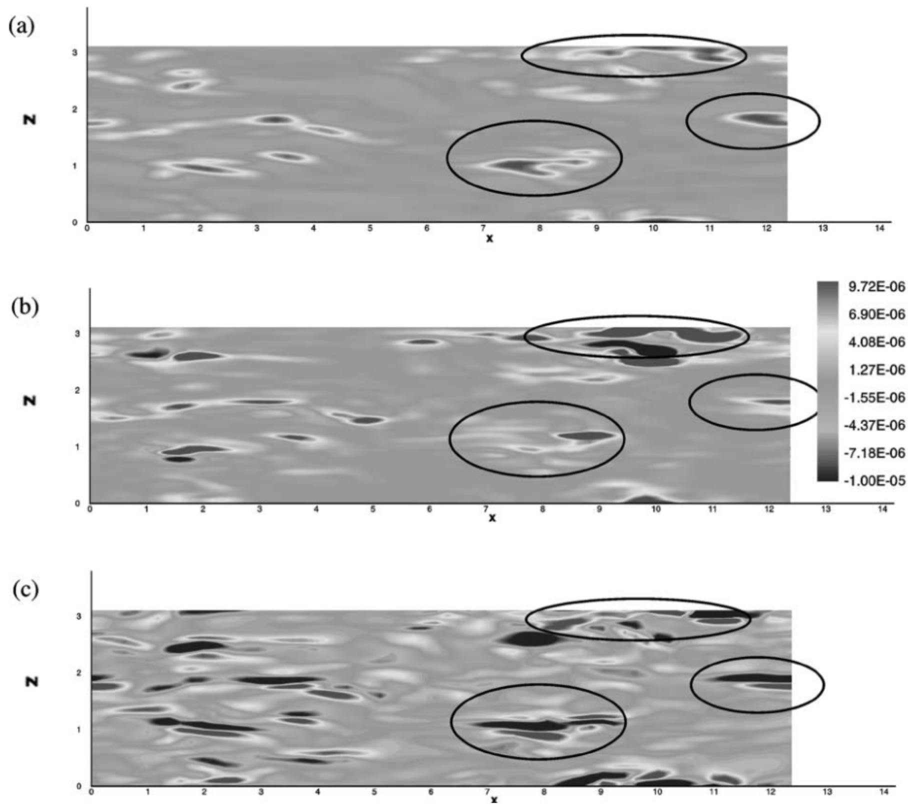


Figure 2.6: turbulent viscosity coefficient (c_s) at $y^+ = 10$
(a) Neural Network (b) Bardina SSM (c)u'w' component of Reynolds stress tensor (Sarghini *et al.* (2003))

More recently, Vollant *et al.* (2017) focused on performing an LES for the concentration of a chemical species while ANNs produced the subgrid-scale model for the simulation. Two approaches were explored in the study for developing the subgrid-scale model. The first was to use the ANN as a surrogate model constrained by the least square error cost function, and the other was to find the coefficients of Noll's formula (Noll (2009)). The first approach produced SGS models that were only accurate within the realm of the training data, while the second approach proved incrementally better in creating a generalisable model. The key takeaway from the study was that the **performance of ANNs depended on**

their application.

Kurz & Beck (2021) used an ANN with Multilayer perceptron (MLP) and Gated recurrent units (GRU) architecture to predict the filtered DNS divergence $\overline{R(F(U))}$ using the coarse-scale solution $\overline{U}$ as the input for Decaying Homogenous Isotropic Turbulence (DHIT). Three distinct LES filters, including two based on the solution representation of Discontinuous Galerkin (DG) and Finite Volume (FV) schemes, and one global Fourier cutoff filter were employed to understand the effect of filter selection on the closure terms. Though the GRU and the MLP were able to learn the unknown closure terms, the GRU outperformed the MLP in accuracy and showed excellent agreement with the exact closure terms.

GRU networks trained with one particular filter were tested on unseen-training data from the two other filters to gauge the generalisation ability of the networks. The LES filter width was changed as a part of this attempt and proved that the networks were generalisable. Additionally, the ANN-predicted closure terms were used to compute the full closure term $\tilde{R}(U) - \overline{R(F(U))}$ and compared with the predictions of the common gradient model (CGM) and smagorinsky model (SSM). The results showed that the predictions by the **ANN were higher in accuracy than CGM and SMM** and that the DG filter was the best choice among the filters.

Guan *et al.* (2022) used a Convolutional Neural Network (CNN) trained with Filtered DNS (FDNS) data to estimate the SGS forcing term for the case of 2D Decaying Homogenous Isotropic Turbulence. The study further compared the results of CNN with DSM, DMM, and ANN. *A priori* analysis of the correlation between the models' SGS terms, inter-scale transfer predictions, and FDNS data showed that the CNN-based data-driven SGS model outperformed the rest of the models. Furthermore, the CNN coupled with a numerical LES solver outperformed the other models in *a posteriori* analysis by capturing backscatter, forecasting short-term spatiotemporal evolution, and reproducing long-term statistics such as TKE spectrum and probability density function (PDF) of vorticity, yet failed to generalise to higher Re. The study overcame this problem using transfer learning. The last two layers of the network, already trained for low Re distribution, were re-trained with a small amount of data from the higher Re distribution. While for cases requiring generalisation to higher numerical resolution, the study included encoder-decoder layers in the transfer learning architecture.

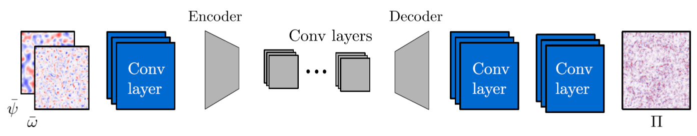


Figure 2.7: CNN with transfer learning and encoder-decoder (Guan *et al.* (2022))

Yuan *et al.* (2022) followed a deconvolutional artificial neural network (D-ANN) approach to model subfilter-scale (SFS) terms; subfilter stress τ_{ij} and heat flux Q_{ij} of compressible turbulent flow. Filtered variables such as density, momentum, and pressure at the neighbouring locations were input by the network to predict unfiltered variables at the local point, further inserted in scale-similarity form to reconstruct the SFS terms. *A priori* analysis of SFS predictions revealed that the D-ANN showed a far more significant correlation than the velocity-gradient model (VGM) and approximate-deconvolution method (ADM) while accurately reconstructing the velocity spectra and kinetic energy flux. Furthermore, in *a posteriori* investigation, the D-ANN-LES coupling outperformed the implicit LES (ILES), DSM, and DMM in the reproduction of velocity spectra, SFS-kinetic energy and thus came closest to the actual filtered DNS. Finally, the *a priori* and *a posteriori* analyses were conducted in two Mach numbers of 0.4 and 0.8, proving that compressibility does not affect the results.

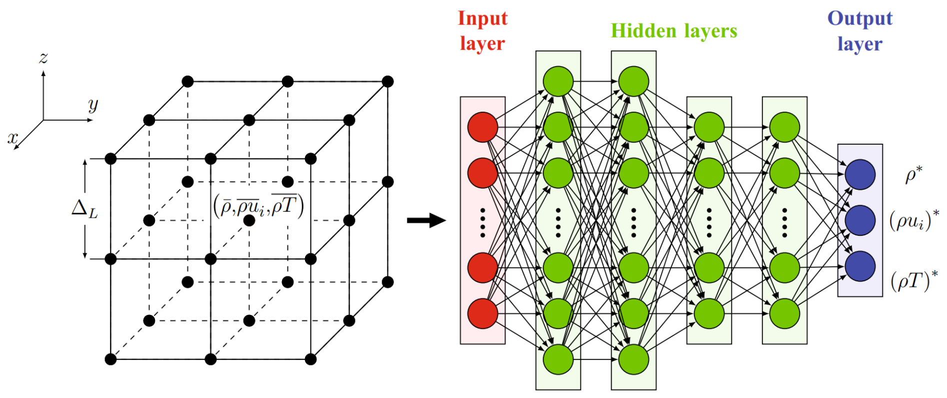


Figure 2.8: Architeture of D-ANN (Yuan *et al.* (2022))

The neural networks emerged as viable alternatives in both investigations by outperforming traditional LES models like DSM, DMM, VGM, and ADM.

2.4. Other Approaches

Another approach that utilised deep learning in turbulence modelling was that of Physics Informed Neural Networks (PINNS), an idea floated in 2017 by Raissi *et al.* (2017). As the name suggests, the neural networks were physically informed by regularisers in the loss function in the form of initial and boundary conditions that limit search space scope to physically viable solutions. The proposed model used the runge kutta time-stepping scheme and did well in solving the burgers equation with acceptable results. The success of the architecture has given way to further research in PINNS and successor models such as the Parareal PINS or PPINS introduced by Meng *et al.* (2020). The idea behind the PPINNS was that it would divide significant integration problems into more minor, independent problems, thus reducing the burden of training on the neural networks. According to the study, this was an improvement over PINNS as it argued that PINNS was unsuitable for large-time steps. In the end, the PPINS solved the 1-D burgers equation more efficiently than the PINNS and proved an improvement. **Providing physical information improves the prediction capabilities of the neural network.**

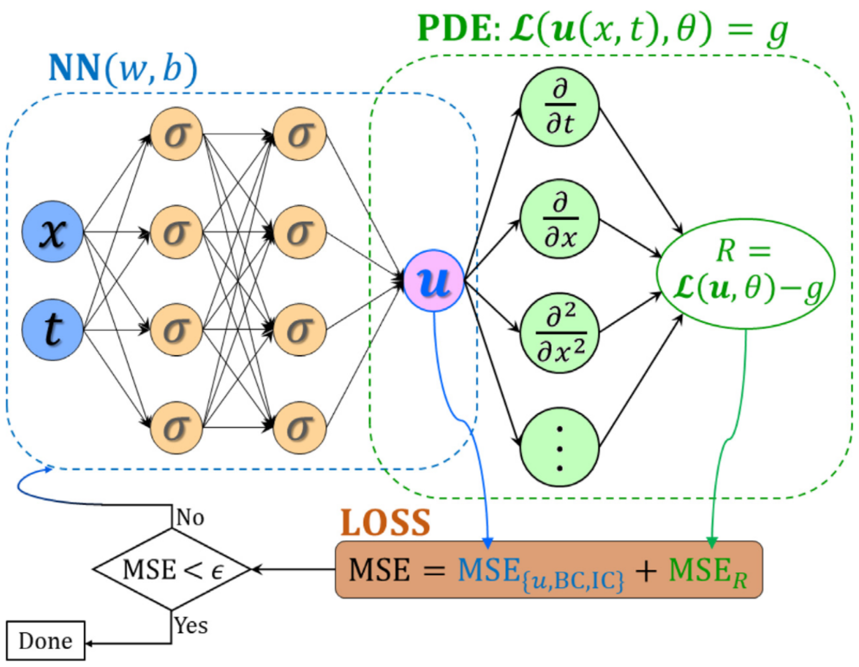

PDE: $\mathcal{L}(u(x,t),\theta) = g$
NN(w,b)
x
t
σ
u
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
$R = \mathcal{L}(u,\theta) - g$
LOSS
MSE $= \text{MSE}_{\{u,BC,IC\}} + \text{MSE}_R$
MSE $< \epsilon$
No
Yes
Done

Figure 2.9: Schematic of PINNMeng *et al.* (2020)

2.5. Reflection

This chapter reviews recent developments in neural networks in turbulence modelling. Lee *et al.* (1997) used a neural network for drag reduction and achieved an 18% reduction in skin friction. Milano & Koumoutsakos (2002) found that neural networks outperformed linear and non-linear PCA and POD in predicting the viscous sublayer velocity profile. The chapter further discusses deep learning for RANS, highlighting the work of Duraisamy & Durbin (2014) and Parish & Duraisamy (2016) on Field Inversion Machine Learning (FIML). It also covers deep learning for LES, focusing on the work of Völker *et al.* (2002),Sarghini *et al.* (2003),Vollant *et al.* (2017),Kurz & Beck (2021),Guan *et al.* (2022), and Yuan *et al.* (2022), along with an alternate approach involving PINNs.

The literature review showed that using neural networks for large eddy simulation (LES) was advantageous as ANN predictions were superior to traditional subgrid-scale (SGS) models. Furthermore, neural networks should be trained with higher fidelity data such as DNS and provided with physical information to improve their prediction capabilities..

3 Theory

3.1. Large Eddy Simulations (LES)

The fundamental principle behind LES is decomposing the velocity field into large-scale and small-scale contributions (Talstra (2011)). In the classical approach, resolved flow represents large-scale motion which can be obtained by filtering the velocity field using a filter function. On the other hand, unresolved flow represents small-scale motion obtained by subtracting the resolved flow from the original velocity field (Adrian *et al.* (2000)). It is modelled using a subgrid-scale (SGS) model, which accounts for the effects of the small-scale turbulent motion on the large-scale motion.

LES only simulates the larger turbulence scales while modelling the smaller scales' effects. Classical LES methodology comprises the following steps: filtering, discretization, modelling, and solving. The first step in LES is to filter the Navier-Stokes equations, which describe the conservation of mass, momentum, and energy in a fluid. The filter width is typically chosen to be larger than the grid spacing but smaller than the characteristic size of the turbulent structures in the flow. This helps to remove small-scale fluctuations. Explicit filtering allows the filter size to be chosen independently of the mesh spacing (Vasilyev & Lund (1997)). However, it is also possible to perform LES without an explicit filter operation, i.e. implicit filtering (Klein (2005)). In this case, the mesh would filter by eliminating all wavelengths shorter than twice the mesh spacing.

The incompressible Navier-Stokes equations can be written as:

$$\begin{aligned} \frac{\partial u}{\partial t} + \nabla \cdot (u \otimes u) + \nabla p - \nabla \cdot 2\nu \nabla^s u &= f \quad \text{in } \Omega \\ \nabla \cdot u &= 0 \quad \text{in } \Omega \end{aligned} \tag{3.1}$$

Considering we apply a filtering operation that is strictly commutative with differentiation, Equation 3.1 can be written as:

$$\begin{gathered} \frac{\partial \bar{u}_i}{\partial x_i} = 0 \\ \rho \frac{\partial \bar{u}_i}{\partial t} + \rho \frac{\partial}{\partial x_j} \left(\bar{u}_i \bar{u}_j \right) = -\frac{\partial \bar{p}}{\partial x} + \frac{\partial}{\partial x_j} \left(\bar{\tau}_{ij} - \rho \left(\overline{u_i u_j} - \bar{u}_i \bar{u}_j \right) \right) \end{gathered} \tag{3.2}$$

The additional term ($T_{ij} \equiv \left(\overline{u_i u_j} - \bar{u}_i \bar{u}_j \right)$) that appears after the filtering operation, is referred to as the subgrid-stress (SGS) tensor. The SGS tensor arises as a result of the non-linear convection term ($\nabla \cdot (u \otimes u)$) in Equation 3.1, and signifies the effect of the small, unresolved scales on the large resolved scales. However, Equation 3.2 cannot be solved directly due to the lack of an expression for the SGS tensor. This is known as the closure problem in LES. Hence, it is necessary to use subgrid-scale models to model the term T_{ij} to solve and close the system of equations.

The decomposition of velocity into large and small-scale contributions can be represented as:

$$u_i = \bar{u}_i + u_i'' \tag{3.3}$$

Substituting Equation 3.3 into Equation 3.2, we get the following expression:

$$\tau_{ij} = \underbrace{\overline{\bar{u}_i\bar{u}_j} - \bar{u}_i\bar{u}_j}_{L} + \underbrace{\overline{\bar{u}_i u_j''} + \overline{u_i''\bar{u}_j}}_{C} + \underbrace{\overline{u_i''u_j''}}_{R} \tag{3.4}$$

Where L stands for the Leonard stress tensor:

$$L_{ij} = \overline{\bar{u}_i\bar{u}_j} - \bar{u}_i\bar{u}_j \tag{3.5}$$

The Leonard stress tensor can be computed exactly and shows the difference in the product of velocities as we filter them twice. C stands for the Cross stress tensor:

$$C_{ij} = \overline{\bar{u}_i u_j''} + \overline{u_i''\bar{u}_j} \tag{3.6}$$

The Cross stress tensor term shows that the fluctuations do not disappear and is the summation of products of filtered and unfiltered quantities. R stands for the Reynolds stress tensor:

$$R_{ij} = \overline{u_i''u_j''} \tag{3.7}$$

Once the equations are filtered, the next step is discretization, which involves the division of the calculation domain into a grid of elements of specific geometric shapes (Baliga & Patankar (1983)), such as cells or volumes, and approximating the equations at discrete points within each element. Finally, the Reynolds stress tensor term needs to be modelled. As the exact reconstruction of the subgrid scales is difficult, subgrid-scale models aim to model the effect of the subgrid-scale turbulence on the large, resolved scales. The SGS models must accurately estimate the energy drain from the resolved scales in the cascade.

3.1.1. Subgrid-scale Modelling

One of the first and most widely used subgrid-scale models in LES is the Smagorinsky model (Smagorinsky (1963)). It is based on the assumption that the subgrid-scale turbulent eddies behave like viscous eddies and that the energy dissipation rate of the subgrid-scale turbulence is proportional to the local strain rate. Furthermore, the model assumes that the small-scale eddies have a universal behaviour and that an eddy viscosity term can represent their effects. The eddy viscosity term can be represented as (Prof.Dr.ir. S. Hickel (2020)):

$$\tau_{ij} = \frac{1}{3}\tau_{kk}\delta_{ij} - 2\nu_{SGS}\bar{S}_{ij} \tag{3.8}$$

Where $\bar{S}_{ij}$ is the strain rate tensor and can be written as:

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{3.9}$$

Following the principles of dimensional consistency and Galilean invariance, the subgrid-scale viscosity ν_{SGS} can be further expressed as:

$$\nu_{SGS} = (C_s\Delta)^2\sqrt{2\bar{S}_{ij}\bar{S}_{ij}} = (C_s\Delta)^2\,|\bar{\mathbf{S}}| \tag{3.10}$$

Where the filter width $\bar{\Delta} = \sqrt[3]{\text{cell volume}}$ and $|\bar{\mathbf{S}}|$ is the suitable norm of the strain rate tensor. Overall, the Smagorinsky model for the subgrid-stress tensor is:

$$\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j = \frac{1}{3}\tau_{kk}\delta_{ij} - 2\left(C_s \Delta\right)^2 |\bar{\mathbf{S}}| \bar{S}_{ij} \tag{3.11}$$

The Smagorinsky model is relatively simple and computationally efficient as it does not require additional PDEs. It drains a sufficient amount of energy and is stable. It has only one adjustable parameter, the Smagorinsky constant, which can be set to a value that gives good agreement with experimental data. However, it is low in correlation with the exact SGS DNS data (Vreman *et al.* (1995)) and performs poorly in complex flows and near the wall.

Furthermore, the Smagorinsky model is based on the idea that small-scale eddies are isotropic, meaning they have the same behaviour in all directions. However, this assumption is only sometimes accurate, as small-scale eddies can be anisotropic. More advanced models, such as the dynamic Smagorinsky model (Germano *et al.* (1991)), have been developed to account for this.

Another commonly used SGS model is the dynamic mixed model. This model combines the Smagorinsky model with a scale-similarity model (Bardina *et al.,* 1980), which assumes that the small scales have a behaviour similar to the large scales. The dynamic mixed model adjusts the balance between the Smagorinsky and scale-similarity models based on the local flow properties, resulting in a more accurate representation of the subgrid scales.

3.2. Weak Form of the Navier-Stokes Equation

An alternate approach to LES, the (VMM) is based on the weak form of the Navier-Stokes Equation, which is first presented below.

3.2.1. Starting Point

Recalling the incompressible Navier-Stokes equations:

$$\begin{aligned} \frac{\partial u}{\partial t} + \nabla \cdot (u \otimes u) + \nabla p - \nabla \cdot 2\nu \nabla^s u &= f \quad \text{in } \Omega \\ \nabla \cdot u &= 0 \quad \text{in } \Omega \end{aligned} \tag{3.12}$$

Note that $\otimes$ denotes the dyadic product. The symmetric velocity gradient (∇^s) is defined as:

$$\nabla^s u = \frac{1}{2}\left(\nabla u + \nabla u^T\right) \tag{3.13}$$

After applying the incompressibility constraint, the momentum equation in the convective form can be written as:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u + \nabla p - \nu \Delta u - f = 0 \quad \text{in } \Omega \tag{3.14}$$

Equation 3.14 is known as the strong form of the Navier-Stokes equation, and it is also represented as $\mathcal{R}(u)$. Additional boundary conditions applied are:

Homogeneous boundary conditions for the velocity:

$$u = 0 \quad \text{on } \Gamma = \partial\Omega \tag{3.15}$$

Zero mean condition for the pressure:

$$\int_\Omega p d\Omega = 0 \quad \forall t \in]0, T[\tag{3.16}$$

It is possible to define another form of the governing equation, known as the weak form, by requiring weighted integrals of the governing equation's residual $\mathcal{R}(u)$ to be zero for all suitable weighting functions, w:

$$\int_0^1 w\mathcal{R}(u)d\Omega = 0 \tag{3.17}$$

Equation 3.17 can be further written as $(w, \mathcal{R}(u))_\Omega$. The variational formulation of the incompressible Navier-Stokes equations can be stated as follows:

$$B(w,q;u,p) = F(w,q) \quad \forall\{w,q\} \in \left[H_0^1(\Omega)\right]^3 \times L_0^2(\Omega) \tag{3.18}$$

Where the functional spaces are defined as:

$$\begin{aligned} L_0^2(\Omega) &:= \left\{q \in L^2(\Omega) : \int_\Omega q d\Omega = 0\right\} \\ H_0^1(\Omega) &:= \left\{w \in H^1(\Omega) : u = 0 \text{ on } \partial\Omega\right\} \end{aligned} \tag{3.19}$$

Hence, the weak form of the Navier stokes equation that we consider is:

$$\begin{aligned} B(w,q;u,p) =& \left(w, \frac{\partial u}{\partial t}\right) - (\nabla w, u \otimes u) + (q, \nabla \cdot u) \\ & - (\nabla \cdot w, p) + (\nabla^s w, 2\nu\nabla^s u) \\ F(w,q) =& (w,f) \end{aligned} \tag{3.20}$$

Where **F** is the forcing term.

3.3. Variational Multiscale Modelling (VMM)

The Variational Multiscale Method (VMM) was first conceptualized by Hughes (1995). VMM directly separates the resolved discrete solution $\bar{u}$ from the unresolved solution u', where $\bar{u}$ is the desired projection of u onto the discrete space:

$$\bar{\mathbf{u}} = \mathcal{P}(\mathbf{u}) \tag{3.21}$$

$$\mathbf{u}' = \mathbf{u} - \mathcal{P}(\mathbf{u}) \tag{3.22}$$

This is typically the H_0' projector but could be L2 or nodal. Now let us consider a homogeneous Dirichlet boundary condition as an example. Let $\mathcal{L}$ be any arbitrary operator, either linear or non-linear.

$$\mathcal{L}(\mathbf{u}) = f \quad \text{in} \quad \Omega \tag{3.23}$$

$$\mathbf{u} = 0 \quad \text{on} \quad \Gamma \tag{3.24}$$

Taking the Galerkin weak form of Equation 3.23 with weighting function $\mathbf{w}$ we get:

$$\left(\bar{\mathbf{w}}, \mathcal{L}(\bar{\mathbf{u}} + \mathbf{u}')\right)_\Omega = \left(\bar{\mathbf{w}}, \mathbf{f}\right)_\Omega \tag{3.25}$$

$$\left(\mathbf{w}', \mathcal{L}(\bar{\mathbf{u}} + \mathbf{u}')\right)_\Omega = \left(\mathbf{w}', \mathbf{f}\right)_\Omega \tag{3.26}$$

Where Equation 3.25 and Equation 3.26 are known as the resolved and unresolved scale equations. The unresolved scale equation can be further written as:

$$\left(\mathbf{w}', \mathcal{L}(\mathbf{u}') + \mathcal{N}(\overline{\mathbf{u}}, \mathbf{u}')\right)_\Omega = \left(\mathbf{w}', \mathbf{f} - \mathcal{L}(\overline{\mathbf{u}})\right)_\Omega = -\left(\mathbf{w}', \mathcal{R}(\overline{\mathbf{u}})\right)_\Omega \tag{3.27}$$

Where $\mathcal{N}$ is a non-linear operator and $\mathcal{R}(\overline{\mathbf{u}})$ is the residual of the residual scale velocity component.

The significance of Equation 3.27 is that it shows that the unresolved velocity term can be written in terms of the residual of the resolved scales. As discussed previously, this avoids creating commutative errors. This remains the most significant advantage of using VMM compared to conventional LES models.

In addition to this, Hughes (1995) also presents an analytical representation of $\mathbf{u}'$ in terms of intrinsic time scale τ and $\mathcal{R}(\overline{\mathbf{u}})$:

$$\mathbf{u}' = -\tau\mathcal{R}(\overline{\mathbf{u}}) \tag{3.28}$$

The intrinsic time scale τ is defined as:

$$\tau = \frac{1}{meas(\Omega^e)} \int_{\Omega^e} \int_{\Omega^e} g(x, y) d\Omega_x d\Omega_y \tag{3.29}$$

Where Ω^e and $meas(\Omega^e$ are the spatial limit and volume of the element, respectively. Later, Hughes *et al.* (1998) further applied the VMM framework to an abstract Dirichlet problem and examined ways to widen the applications of VMM.

3.3.1. VMM in LES

Bazilevs *et al.* (2007) introduced the concept of residual-based turbulence modelling. Firstly, a cutoff wavelength was defined to divide the energy spectrum into resolved and unresolved scales. The cutoff wavelength was assumed to be situated in the inertial sub-range of the energy spectrum. The unresolved fine scale equation was divided into parts; linear and non-linear. An infinite perturbation series expansion approximated the non-linear part, while the remaining linear part was ”inverted” with the help of a matrix greens function. Considering only first-order terms, the non-linear expansion was simplified into Equation 3.28, and $\tau\mathcal{R}(\overline{\mathbf{u}})$ was calculated for every element. Finally, the study showed that for the case of forced isotropic turbulence in channel flow (Re_τ=395), results produced by residual-based modelling were in close agreement with DNS data.

While the previous study by Bazilevs *et al.* (2007) inverted the linear part of the unresolved fine scale equation, Shakib *et al.* (1991) used the orthogonality of the resolved and unresolved spaces and formulated τ for a linear advection-diffusion operator as follows:

$$\tau_{linAD} = \left[\left(\frac{2a}{x}\right)^2 + \left(\frac{4v}{\Delta x^2}\right)^2\right] \tag{3.30}$$

It is important to note here that the Navier-Stokes equation requires more than one τ, out of which the formulation mentioned above is only for the linear advection part of the diffusion operator.

3.4. Formulation of the closure terms

3.4.1. Residual Based LES

The system of equations for the large scales are:

$$B\left(w^h, q^h; u^h + u', p^h + p'\right) = F\left(w^h, q^h\right) \quad \forall \left\{w^h, q^h\right\} \in \left[V^h\right]^3 \times Q^h \tag{3.31}$$

V^h is the space of velocity approximations and Q^h is the space of pressure approximations. The superscripts h and ’ indicate the large and small scales, respectively.

Substituting Equation 3.20 in Equation 3.31 gives us the following formulation:

$$\begin{aligned}\left(w^h, \frac{\partial u^h}{\partial t}\right) &- \left(\nabla w^h, u^h \otimes u^h\right) + \left(q^h, \nabla \cdot u^h\right)\\ &- \left(\nabla \cdot w^h, p^h\right) + \left(\nabla^s w^h, 2\nu\nabla^s u^h\right)\\ +\left(w^h, \frac{\partial u'}{\partial t}\right) &- \left(\nabla w^h, u^h \otimes u'\right) - \left(\nabla w^h, u' \otimes u^h\right)\\ &- \left(\nabla w^h, u' \otimes u'\right) + \left(q^h, \nabla \cdot u'\right)\\ &- \left(\nabla \cdot w^h, p'\right) + \left(\nabla^s w^h, 2\nu\nabla^s u'\right) = \left(w^h, f\right)\end{aligned} \tag{3.32}$$

The closure terms are marked blue to differentiate them from the large-scale terms. Equation 3.32 in the non-vector notation for the x,y,z momentum and continuity equations can be written as:

X-momentum equation

q is the weight function

$$\begin{aligned}(q^h, u_t^h) + (q^h, u^h u_x^h) + (q^h, v^h u_y^h) + (q^h, w^h u_z^h) + (q^h, p_x^h) - 2\nu(q^h, u_{xx}^h) - \nu(q^h, v_{xy}^h) - \nu(q^h, w_{xz}^h)\\ -\nu(q^h, u_{yy}^h) - \nu(q^h, u_{zz}^h) + (q^h, u_t') - (q_x^h, u^h u') - (q_x^h, \frac{u'u'}{2})\\ +\left[-(q_y^h, v^h u') - (q^h, v_y^h u') + (q^h, v' u_y^h) - (q_y^h, v'u') - (q^h, v_y' u')\right]\\ +\left[-(q_z^h, w^h u') - (q^h, w_z^h u') + (q^h, w' u_z^h) - (q_z^h, w'u') - (q^h, w_z' u')\right]\\ -(q_x^h, p') + 2\nu(q_x^h, u_x') + \nu(q_x^h, v_y') + \nu(q_x^h, w_z') + \nu(q_y^h, u_y') + \nu(q_z^h, u_z') = (q^h, f_x)\end{aligned} \tag{3.33}$$

Y-momentum equation

r is the weight function

$$\begin{aligned}(r^h, v_t^h) + (r^h, u^h v_x^h) + (r^h, v^h v_y^h) + (r^h, w^h v_z^h) + (r^h, p_y^h) - \nu(r^h, v_{xx}^h) - \nu(r^h, u_{yx}^h) - 2\nu(r^h, v_{yy}^h)\\ -\nu(r^h, w_{yz}^h) - \nu(r^h, v_{zz}^h) + (r^h, v_t') + \left[-(r_x^h, u^h v') - (r^h, u_x^h v') + (r^h, u' v_x^h) - (r_x^h, u'v') - (r^h, u_x' v')\right]\\ -(r_y^h, v^h v') - (r_y^h, \frac{v'v'}{2}) + \left[-(r_z^h, w^h v') - (r^h, w_z^h v') + (r^h, w' v_z^h) - (r_z^h, w'v') - (r^h, w_z' v')\right]\\ -(r_y^h, p') + \nu(r_x^h, v_x') + \nu(r_y^h, u_x') + 2\nu(r_y^h, v_y') + \nu(r_y^h, w_z') + \nu(r_z^h, v_z') = (r^h, f_y)\end{aligned} \tag{3.34}$$

Z-momentum equation

m is the weight function

$$\begin{aligned}(m^h, w_t^h) + (m^h, u^h w_x^h) + (m^h, v^h w_y^h) + (m^h, w^h w_z^h) + (m^h, p_z^h) - \nu(m^h, w_{xx}^h) - \nu(m^h, w_{yy}^h) - \nu(m^h, u_{zx}^h)\\ -\nu(m^h, v_{zy}^h) - 2\nu(m^h, w_{zz}^h) + (m^h, w_t') + \left[-(m_x^h, u^h w') - (m^h, u_x^h w') + (m^h, u' w_x^h) - (m_x^h, u'w') - (m^h, u_x' w')\right]\\ +\left[-(m_y^h, v^h w') - (m^h, v_y^h w') + (m^h, v' w_y^h) - (m_y^h, v'w') - (m^h, v_y' w')\right] - (m_z^h, w^h w') - (m_z^h, \frac{w'w'}{2})\\ -(m_z^h, p') + \nu(m_x^h, w_x') + \nu(m_y^h, w_y') + \nu(m_z^h, u_x') + \nu(m_z^h, v_y') + 2\nu(m_z^h, w_z') = (m^h, f_z)\end{aligned} \tag{3.35}$$

continuity equation

l is the weight function

$$(l^h, u_x^h) + (l^h, v_y^h) + (l^h, w_z^h) + (l^h, u_x') + (l^h, v_y') + (l^h, w_z') = 0 \tag{3.36}$$

The derivation of Equation 3.33-Equation 3.35 are given in Appendix A.

3.4.2. DNS reference data notation

The resolved and closure terms in Equation 3.33-Equation 3.36 can be grouped into:

1. Time derivative terms $(\mathrm{wU\_t})$
2. Convection terms (wConvU)
3. Pressure Gradient terms (wGradP)
4. Diffusion terms (wDifU)
5. Divergence / Continuity terms (wDivU)

The additional 'h' at the end of the terms will denote the resolved terms, while the unresolved terms will be denoted by '. Take the x-momentum equation (Equation 3.33) for consideration:

$$\begin{aligned}(q^h,u^h_t)+(q^h,u^hu^h_x)+(q^h,v^hu^h_y)+(q^h,w^hu^h_z)+(q^h,p^h_x)-2\nu(q^h,u^h_{xx})-\nu(q^h,v^h_{xy})-\nu(q^h,w^h_{xz})\\ -\nu(q^h,u^h_{yy})-\nu(q^h,u^h_{zz})+(q^h,u'_t)-(q^h_x,u^hu')-(q^h_x,\frac{u'u'}{2})\\ +\left[-(q^h_y,v^hu')-(q^h,v^h_yu')+(q^h,v'u^h_y)-(q^h_y,v'u')-(q^h,v'_yu')\right]\\ +\left[-(q^h_z,w^hu')-(q^h,w^h_zu')+(q^h,w'u^h_z)-(q^h_z,w'u')-(q^h,w'_zu')\right]\\ -(q^h_x,p')+2\nu(q^h_x,u'_x)+\nu(q^h_x,v'_y)+\nu(q^h_x,w'_z)+\nu(q^h_y,u'_y)+\nu(q^h_z,u'_z)=(q^h,f_x)\end{aligned}$$

The DNS data extraction code outputs the closure terms, which are then grouped as:

$$\begin{aligned}\underbrace{(q^h,u^h_t)}_{\mathrm{wUh\_t}}+\underbrace{\left((q^h,u^hu^h_x)+(q^h,v^hu^h_y)+(q^h,w^hu^h_z)\right)}_{\mathrm{wConvUh}}+\underbrace{(q^h,p^h_x)}_{\mathrm{wGradPh}}\\ +\underbrace{\left(-2\nu(q^h,u^h_{xx})-\nu(q^h,v^h_{xy})-\nu(q^h,w^h_{xz})-\nu(q^h,u^h_{yy})-\nu(q^h,u^h_{zz})\right)}_{\mathrm{wDifUh}}+\underbrace{(q^h,u'_t)}_{\mathrm{wU'\_t}}\\ +\underbrace{\left(-(q^h_x,u^hu')-(q^h_x,\frac{u'u'}{2})+\left[-(q^h_y,v^hu')-(q^h,v^h_yu')+(q^h,v'u^h_y)-(q^h_y,v'u')-(q^h,v'_yu')\right]\right)}_{\mathrm{wConvU'(1/2)}}\\ +\underbrace{\left[-(q^h_z,w^hu')-(q^h,w^h_zu')+(q^h,w'u^h_z)-(q^h_z,w'u')-(q^h,w'_zu')\right]}_{\mathrm{wConvU'(2/2)}}+\underbrace{\left(-(q^h_x,p')\right)}_{\mathrm{wGradP'}}\\ +\underbrace{\left(2\nu(q^h_x,u'_x)+\nu(q^h_x,v'_y)+\nu(q^h_x,w'_z)+\nu(q^h_y,u'_y)+\nu(q^h_z,u'_z)\right)}_{\mathrm{wDifU'}}=(q^h,f_x)\end{aligned} \tag{3.37}$$

This can be generalized to all three momentum equations (X, Y, and Z). The closure terms along an axis are summed up into one term. Hence, there will be only one term per equation, each in the X, Y, and Z directions.

$$\underbrace{(\mathrm{wUh\_t}+\mathrm{wConvUh}+\mathrm{wGradPh}+\mathrm{wDifUh})}_{\text{Resolved terms}}+\underbrace{(\mathrm{wU'\_t}+\mathrm{wConvU'}+\mathrm{wGradP'}+\mathrm{wDifU'})}_{\text{Closure term}}=0 \tag{3.38}$$

The continuity equation Equation 3.36 can be simplified and represented as:

$$\mathrm{wDivUh}+\mathrm{wDivU'}=0 \tag{3.39}$$

There will be a total of 4 $((1\times 3(X/Y/Z)+1))$ closure terms per weighting function. Through trilinear interpolation, each closure term is stored in the element as the weighted average of the values of the eight weighting functions. There are four closure terms and eight weighting functions in total. The weights of only one weighting function are considered for our present analysis. The network needs to predict the four closure terms as a function of the resolved scale terms: velocity (u,v,w), pressure and their spatiotemporal derivatives.

4 Previous work at TU Delft

Work at TU Delft could be divided based on the approach taken. One approach was to train the neural networks to predict model coefficients for essential parameters. While the other was to use neural networks to find the underlying relation between resolved and unresolved scales. Durieux (2015) modelled the stabilization parameter τ (Hughes *et al.* (1998)) for the burgers and advection-diffusion equations using neural networks. Though a strategy for an effective loss measure was formulated, its applications were limited to the 1D Burgers equation. Beekman (2016) improved upon this and created a more generalized SGS model using neural networks and the open-source library OpenNN. Later, Kurian (2019) found the coefficients of the stabilization parameter τ for both subgrid scales and subgrid-scale closures using neural networks. The work discussed thus far constitutes the former approach, i.e. using neural networks to model coefficients.

The latter approach makes optimum use of the non-linear modelling capabilities of the ANNs. Robijns (2019) focused on the relation between the large resolved scales and the interaction terms. The neural network was well-trained with flows of different Reynold's numbers, time steps, and forcing but could not achieve stability. A 99% limiting scheme was implemented on the network outputs and managed to bring stability to the ANN. However, the underlying cause of instability was not determined.

Janssens (2019) tackled similar stability issues in the context of global climate simulations. The ANN was trained offline with a high-resolution data set but made predictions online. The approach used byJanssens (2019) to train the network offline and integrate it into an online simulation led to errors and instability. Upon examining the instabilities, it was seen that they could be categorized into two types. The first type was because the ANN added additional roots to the residual space, leading to convergence at the wrong estimates and, eventually, divergence. This instability was identified and called the Corrector Pass Instability (CPI). The second type of instability was due to an accumulation of errors and was called Long Term Instability (LTI).

Janssens (2019) attempted to train the ANN with the complete evolution history of the simulation. This method made the simulation stable yet led to an exponential increase in training data size and computational cost. In addition, a quadratic coefficient ANN model was tested, where the ANN makes predictions at the start and keeps them constant throughout the time step. CPI was eliminated, yet only at grid refinements close to DNS data. Several other techniques were tried, with only a slight improvement in stability.

The data enrichment approach for the CPI was extended to LTI by Pusuluri (2020), with the augmentation of the neural network's training data with the noise of different types. Gaussian noise worked best for the 1D Burgers equation, and it was explicitly added near the walls, as the magnitude of the interaction terms was highest near the walls. The study proved that adding noise improved stability, but its generalizability and applicability were unclear.

In contrast, Rajampeta (2021) considered strategies like Lagged Feature Set (LFS), Energy Transfer Regularization (ETR), three-scale discretization, and Backscatter Limiting (BL) to address Corrector

Pass Instability (CPI) and Long Term Instability (LTI) for a 1D model driven by turbulent channel forcing data. LFS aimed to decouple the ANN and the time march by removing terms in the feature set at the next time level (i.e. n+1). The ANN made predictions based on data from the previous time step. LFS efficiently eliminated CPI but not LTI. The following strategy, ETR, aimed to reduce the Long Term Instability. In ETR, an element-wise unresolved scales energy transfer (UET_{e_i}) was calculated, along with a corresponding Energy Transfer Loss. The energy transfer MSE loss was to be added with the MSE of ANN predictions to embed the ANN with more physical information and keep it more informed. ETR was implemented along with LFS, and the combination proved effective only on CPI. The third strategy of three-scale discretization made the simulation more robust but was ineffective with LTI. The last strategy was Backscatter Limiting (BL), and the backscatter was set to a limit of zero to keep the simulation stable. BL was a great enforcer of stability, irrespective of CPI or LTI. The only concern was that it affected the overall accuracy of the simulation. Hence, it was decided that though BL was effective in increasing the stability of the simulation, it should be tried in addition to the other strategies. Finally, LFS and BL were tried together, and the combination successfully brought stability and accuracy to the simulation.

The latest studies, such as the work done by Rajampeta (2021), have concentrated on a one-dimensional model of turbulent channel flow. However, extending this VMM-ANN method to a three-dimensional flow and determining the difficulty in replicating the closure terms would be highly beneficial. Therefore, the focus of the study would be to find the simplest ANN architecture which can accurately, i.e. with a correlation of approximately 99%, reproduce the closure terms.

5 Methodology

5.1. Main Test Case

The flow of interest is 3D turbulent channel flow at a Reynolds Number (Re_τ) of 180 along with a courant number of 0.1. The initial discretisation of interest is 32 elements per direction, which brings the total number of elements to $(32 \times 32 \times 32)$, i.e. 32768. There is one weighting function and four closure terms in total. We will discretise with piecewise linear functions for which 32^3 is a very coarse mesh. Therefore, it will rely heavily on the closure model to obtain good results. The geometry of the fluid domain of interest can be depicted as:

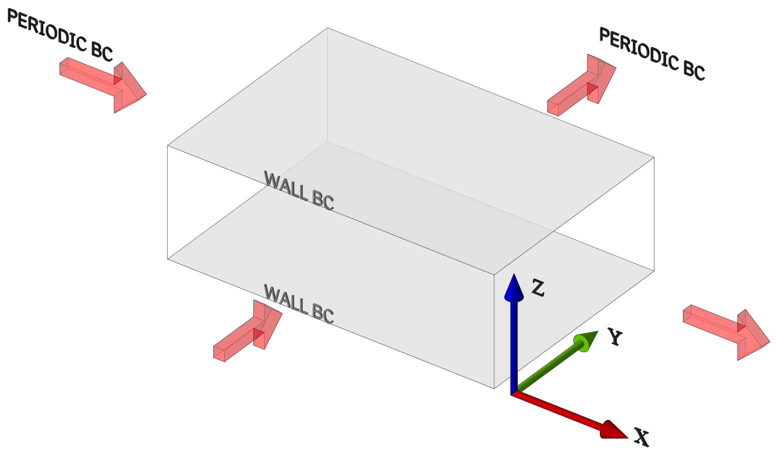


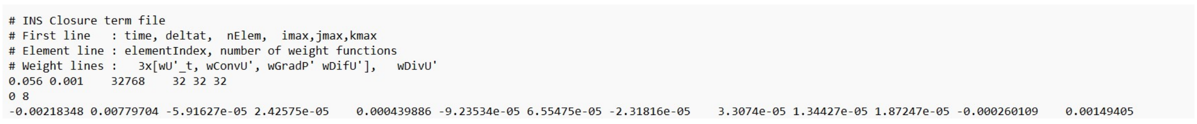
Figure 5.1: 3D Turbulent Channel Flow

As there are 32 elements in each direction, the total number of elements per XY plane is 1024 (32×32), and 32 of these planes are along the Z direction. Therefore, the X, Y, and Z directions are denoted i,j, and k, respectively, with numbering for the planes 0 to 31.

5.2. Training Data

5.2.1. Closure

The output is from a dedicated spectral/finite volume TCF DNS code from which $\bar{u}$=P(u), and the closure terms were calculated. The closure terms were computed as u'= $u_{DNS} - P(\bar{u})$, and four closure terms per element form the output. The output files were arranged such that each file represented one LES time step and contained element-wise information on the closure terms. The output read from these files was shaped into an array of sizes (32768, 4). The final shape of the output $(32768 \times n, 4)$ depends on the number of time steps/files (n) being considered for that particular run. Each output file contained information related to the LES time, time-step, discretisation, total number of elements, and dimensions, along with element-wise information, as can be seen below:



```
# INS Closure term file
# First line   : time, deltat,  nElem,  imax,jmax,kmax
# Element line : elementIndex, number of weight functions
# Weight lines :   3x[wU'_t, wConvU', wGradP' wDifU'],   wDivU'
0.056 0.001    32768    32 32 32
0 8
-0.00218348 0.00779704 -5.91627e-05 2.42575e-05    0.000439886 -9.23534e-05 6.55475e-05 -2.31816e-05    3.3074e-05 1.34427e-05 1.87247e-05 -0.000260109    0.00149405
```

Figure 5.2: Output File

In the picture above, the closure terms along an axis are summed up into one term. Hence, there will be only one term per equation, each in the X, Y, and Z directions.

5.2.2. Feature Set

The input of the ANN was provided in a total of five files per LES time step; velocity (U), time gradient (Ut), and the three spatial gradient files (Ux, Uy, and Uz). Each file contained the following element-wise contribution:

File	Contribution
Velocity file (U)	$\bar{u}$,$\bar{v}$,$\bar{w}$, and $\bar{p}$
Time gradient (Ut)	$\bar{u}_t$,$\bar{v}_t$,$\bar{w}_t$, and $\bar{p}_t$
Spatial gradient-X (Ux)	$\bar{u}_x$,$\bar{v}_x$,$\bar{w}_x$, and $\bar{p}_x$
Spatial gradient-Y (Uy)	$\bar{u}_y$,$\bar{v}_y$,$\bar{w}_y$, and $\bar{p}_y$
Spatial gradient-Z (Uz)	$\bar{u}_z$,$\bar{v}_z$,$\bar{w}_z$, and $\bar{p}_z$

The information from each file is read into an array of shapes (32768,4). The combined shape of the input array formed from information read from all five files is (32768,20), i.e. 20 features per element. Similar to the output, the final shape of the input $(32768 \times \mathrm{n}, 20)$ depends on the number of time steps/files (n) being considered for that particular run.

The input and output arrays were normalised immediately after reading the information from the files. At this point, the requirement from the network is to map the correlation between the four input features and the 20 closure terms. The number of time steps/files (n) will ultimately dictate the number of training examples $(32768 \times \mathrm{n})$ the network has to train on.

5.2.3. Stencil

A stencil was created which included the neighbours of the element of interest in its feature set. First, the immediate neighbours in all three spatial directions were considered, namely the elements from the left, right, top, bottom, front and back. The analysis details about the different possible combinations of stencils considered shall be seen later in subsection 6.3.3

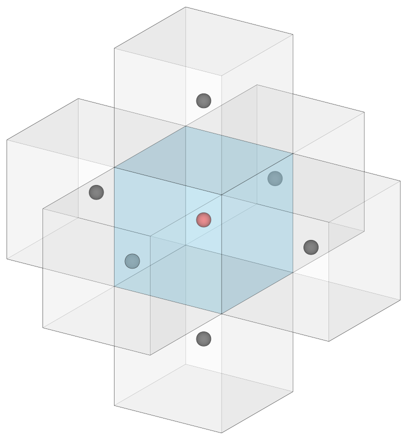


Figure 5.3: Visualisation of the element stencil

The element of interest is shown in blue in the figure above, with its centroid marked in red. The six neighbouring elements and their respective centroids are shown in grey and black, respectively. Including the neighbouring elements in the stencil effectively increases the number of input features to 140 (20×(6+1))

5.2.4. Boundary Conditions

The stencil described in the previous section gives rise to the challenge that elements at the boundary planes have stencil members that cross the bounds of the defined geometry. As we dealt with turbulent channel flow, a periodic boundary condition was applied in the i and j directions. In the k-direction, a wall condition was imposed by setting the bottom stencil members of the k=0 plane elements and the top stencil members of the k=31 plane elements to zero.

6 Results

The preceding chapters have presented the recent advancements and contributions of neural networks in turbulence, as discussed in chapter 2. The initial section covered the theory behind LES, VMM, and the input provided to the neural network. This was followed by a discussion of the previous work at TU Delft and the potential for expanding the framework to three dimensions, as outlined in chapter 4. In chapter 5, the study's primary objective was introduced, which involved examining turbulent channel flow and the methodology used to accomplish this. The primary aim of the analysis is to identify the requirements for a neural network to reproduce the closure terms with a correlation of approximately 99%, as well as to determine the smallest network that can achieve this goal.

6.1. Baseline Network

The entire code written to train and evaluate the network was run on a GPU for faster performance. Tensorflow and Keras of version 2.10 were used in our analysis. The overall specifications considered for the network can be seen below:

6.1.1. Network Hyperparameters

The structure and hyperparameters were taken from Rajampeta (2021).

Category	Specification
Number of Hidden Layers	4
Number of Neurons	256
Batch Size	32
Activation Function	ReLU
Optimizer	Nadam

The number of input and output units of the network are:

Input	140
Output	4

6.1.2. Network Structure

The number of neurons for each hidden layer is kept constant. The structure chosen for the ANN is:

Layer	Neurons	Activation
Input Layer	140	ReLU
Layer 1	256	ReLU
Layer 2	256	ReLU
Layer 3	256	ReLU
Layer 4	256	ReLU
Output Layer	104	ReLU

6.1.3. Training and Validation

The initial number of time steps considered is one. The total number of samples provided to the network is 32768. 25% of these samples have been split to be used as test data. Out of the 75% data that is kept for training, 15% is used to validate the data. Normalisation was performed over the entire dataset. The entire time step was used to compute the normalisation parameters, such as mean and standard deviation, which were then used to scale the input features.

The data is normalized as a whole

Category	Specification
Number of Time Step(s)	1
Number of Training Samples	32768
Test Split	0.25
Validation Split	0.15
Learning Rate	0.001
Epochs	100

The training and validation losses and the network's performance on unseen test data are seen below.

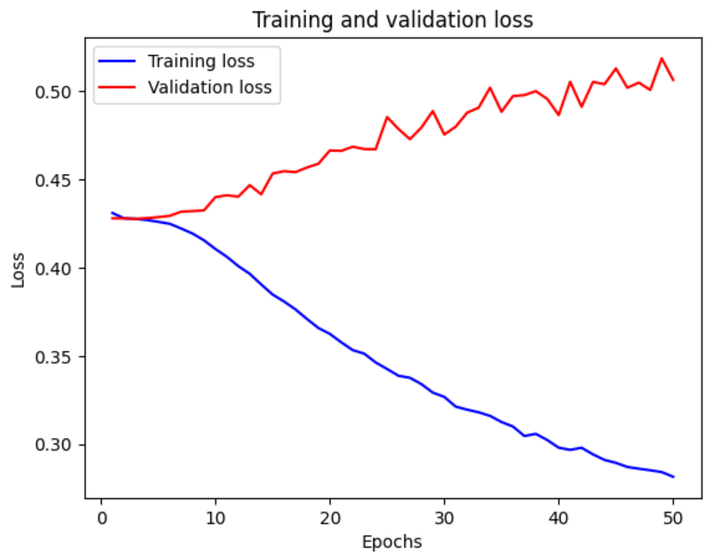
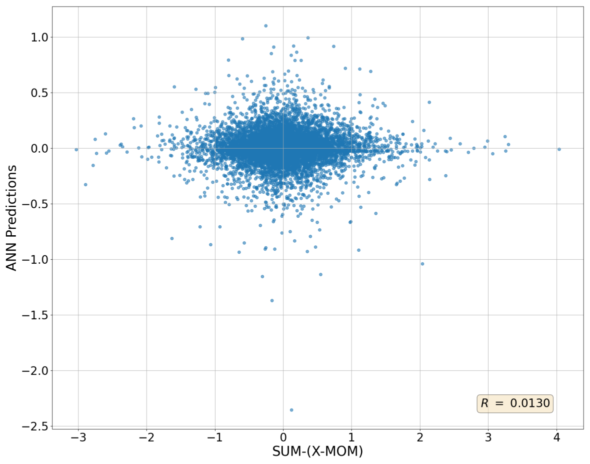
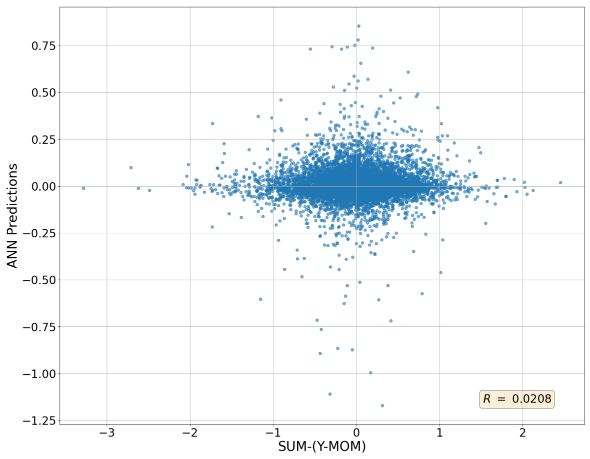


Figure 6.1: Training and Validation loss (Baseline Network)

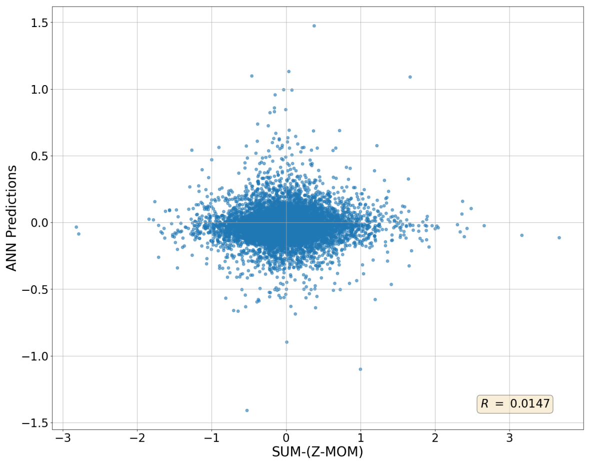


(a) X

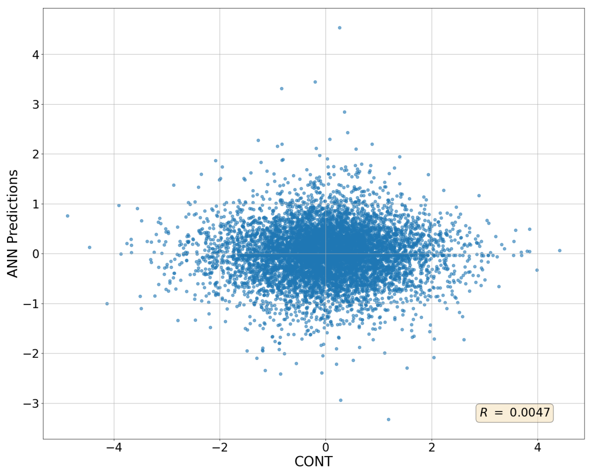


(b) Y

Figure 6.2: X and Y Momentum Terms (Baseline Network)



(a) Z



(b) Cont.

Figure 6.3: Z Momentum and Continuity Terms (Baseline Network)

Train Loss	Test Loss	X-Mom (Sum.)	Y-Mom (Sum.)	Z-Mom (Sum.)	Cont.
0.2816	0.5064	0.0130	0.0208	0.0147	0.0047

The training loss kept decreasing while the validation loss increased. The massive gap between graphs and fluctuations in the validation plot indicate instability in the training process and the occurrence of overfitting. As the amount of training data is limited relative to the complexity of the model, the network has memorised the training data and performed poorly on new, unseen test data.

The low correlation for the closure terms indicated that the ANN predicts the same repeated answer for different inputs. The ANN is not accurately capturing the relationship between the input and output variables. The first step to tackle overfitting and instability in the training process was to increase the training data.

6.2. Increasing the training data

In the previous section, overfitting was observed in the neural network training due to insufficient training data. Only one time step or 32768 training samples was considered. In this section, double the training data, 65536 samples were considered for training. The training and validation losses and the network's performance on unseen test data are seen below.

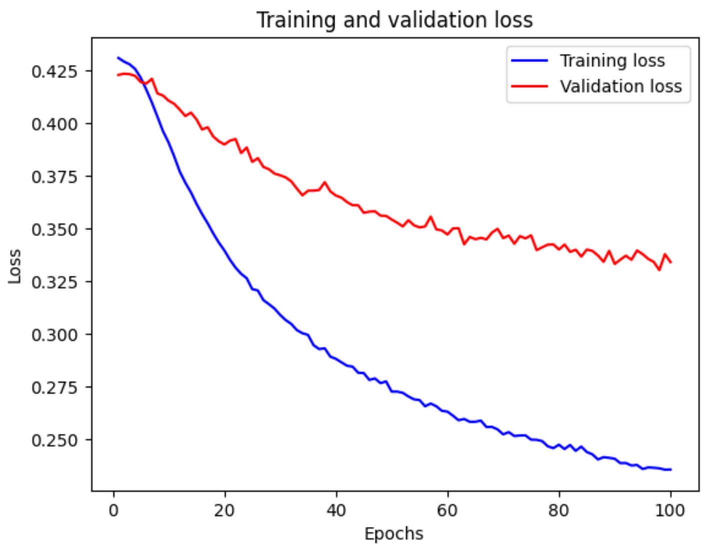
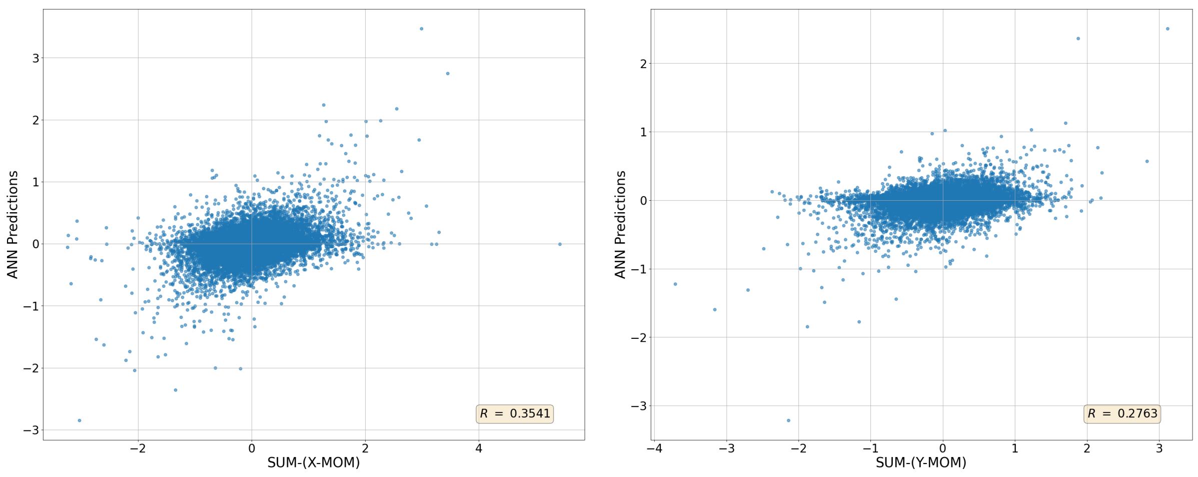


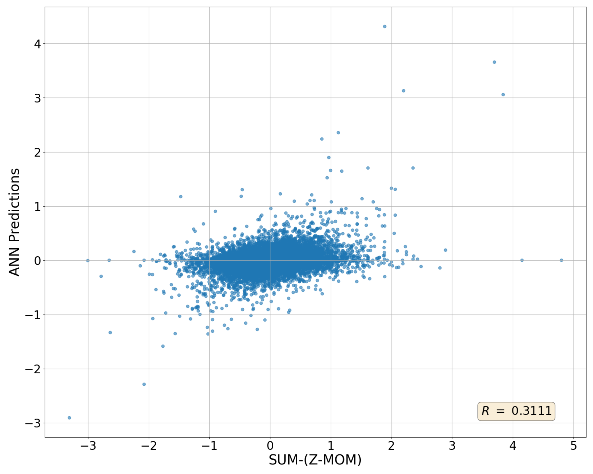
Figure 6.4: Training and Validation loss (Baseline – 2 Time Steps)



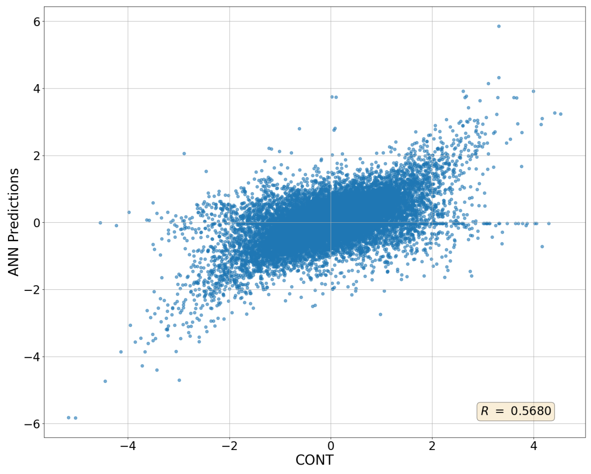
(a) X

(b) Y

Figure 6.5: X and Y Momentum Terms (Baseline – 2 Time Steps)



(a) Z



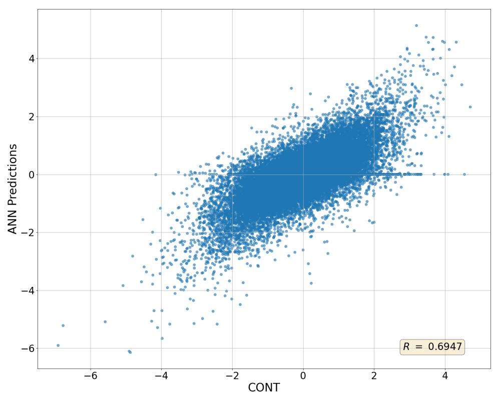
(b) Cont.

Figure 6.6: Z Momentum and Continuity Terms (Baseline – 2 Time Steps)

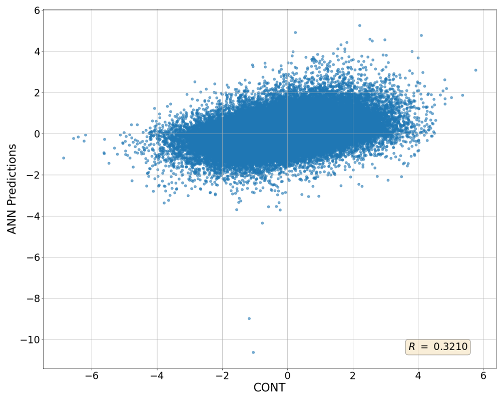
Train Loss	Test Loss	X-Mom (Sum.)	Y-Mom (Sum.)	Z-Mom (Sum.)	Cont.
0.2356	0.3341	0.3541	0.2763	0.3111	0.5680

No. of Time Steps	Train Loss	Test Loss	X-Mom (Sum.)	Y-Mom (Sum.)	Z-Mom (Sum.)	Cont.
4	0.5245	1.0590	0.8327	0.8222	0.8134	0.6947
5	0.4222	0.8974	0.8728	0.8471	0.8484	0.7039
6	0.4657	0.8974	0.8711	0.8495	0.8537	0.6827
8	0.5708	0.9608	0.8630	0.8388	0.8402	0.6176
21	1.0162	1.3368	0.8161	0.7557	0.7717	0.3210

It was observed that an increase in training data failed to increase the correlation of the network after a certain number of time steps. Giving the network access to more information did not help it capture the underlying patterns in the data. The decrease in network correlation with increased training data can be seen below:

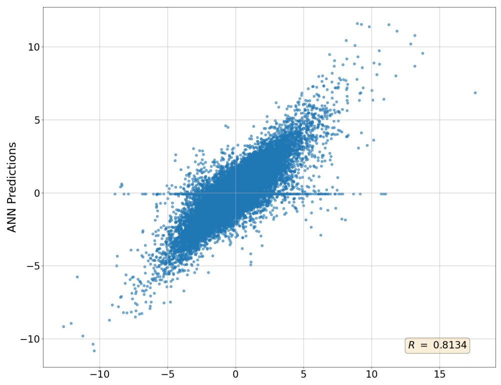


(a) 4 Time Steps

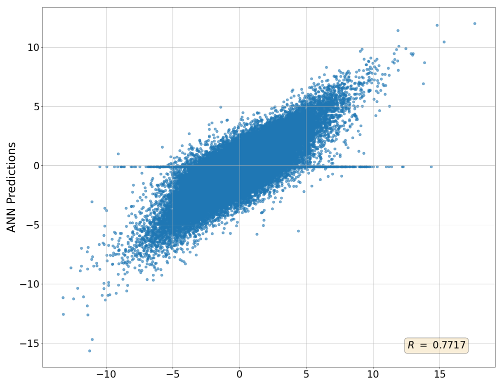


(b) 21 Time Steps

Figure 6.7: Continuity Terms (Baseline Network)



(a) 4 Time Steps



(b) 21 Time Steps

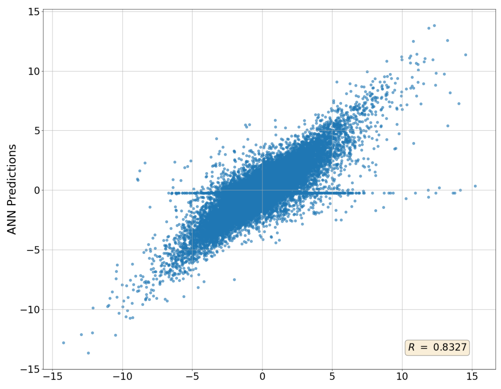
Figure 6.8: Z Momentum Terms (Baseline Network)

It was observed in the correlation graphs given above that most points in the distribution were approximately 5-10 standard deviations away from the mean value of the data. Furthermore, many points could be seen 10-15 standard deviations away. This meant a discrepancy in the normalisation of the data, propagated, and led to incorrect predictions.

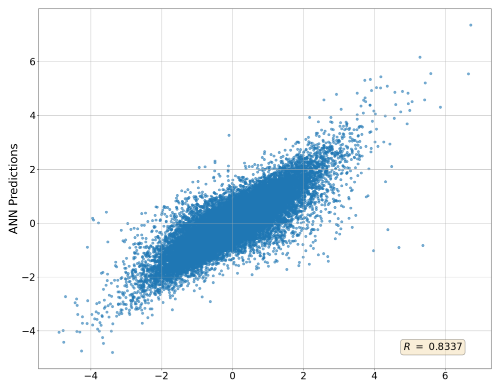
For the baseline network, normalisation was performed over the entire dataset. When data were normalised, the entire time step was used to compute the normalisation parameters, such as mean and standard deviation, which were then used to scale the input features. This led to the loss of spatial information and the introduction of biases into the model.

In turbulent channel flow, the elements that are located along a plane are similar due to homogeneity and isotropy. Each plane has its own unique mean and standard deviation, and normalising each plane individually can help preserve the spatial information in the data. Normalising data as a whole leads to inconsistent normalisation across different planes. The normalisation discrepancy was fixed by normalising the dataset plane by plane and using an automated normalisation function **StandardScalar()**.

SS()	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
Before	0.5245	1.0590	0.8327	0.8222	0.8134	0.6947
After	0.1230	0.3055	0.8337	0.8276	0.8401	0.8521

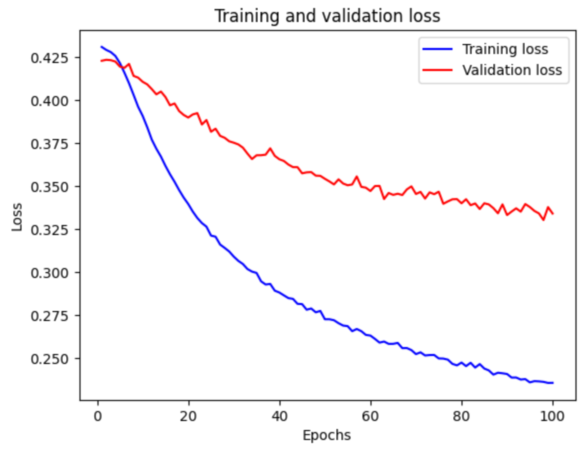


(a) Normalization – Whole

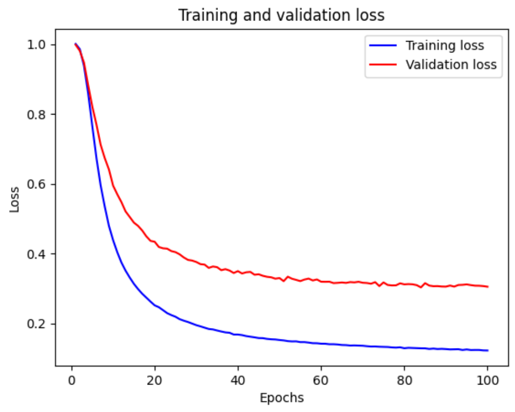


(b) Plane-by-Plane Normalization

Figure 6.9: Fixing Normalization (Baseline Network – 4 Time Steps)



(a) Normalization – Whole



(b) Plane-by-Plane Normalization

Figure 6.10: Change in Loss graph (Baseline Network – 4 Time Steps)

Change in the normalisation procedure led to a visible decrease in the presence of ANN prediction points equal to zero (zero points). Along with zero points, the difference between the losses and the number of standard deviations of the data distribution also decreased. Proper normalisation ensured that each input feature was scaled to a similar range or distribution, thus reducing the variability in the correlation coefficients and noise and outliers.

6.2.1. Reflection

This section implemented the ANN used by Rajampeta (2021) for 3D turbulent channel flow. First, it showed deficiencies in the training process due to overfitting, leading to poor performance on new test data. Then, the training data size was increased to reduce overfitting, which proved successful as the ANN's performance improved, and the gap between the training and validation loss decreased. However, further increasing the training data did not improve the ANN's accuracy, indicating that the network could not capture the underlying patterns in the data. A discrepancy was then identified in the normalisation of the data, which led to the loss of spatial information and biases in the model. This was rectified by normalising the data plane by plane using an automated normalisation function, and the ANN's performance significantly improved. The results demonstrate the effectiveness of increasing the training data to reduce overfitting and the importance of preserving spatial information during the normalisation process. The following section will further concentrate on increasing the network predictions' correlation.

6.3. Phase II: Post-Normalization Correction

6.3.1. Neuron-Hidden Layer Analysis

In the previous section, improvements were realised, but the training data size was varied, and its relation to the network's performance was analysed. The results showed that directly adopting the structure and hyper-parameters used in Rajampeta (2021) is not sufficient for the current analysis. This necessitates further analysis to find the network capable of reproducing the 3D turbulent channel flow closure terms. Therefore, this section studies the impact of the network's structure on its accuracy.

In Figure 6.10, the difference in training and validation loss had decreased, but not completely. Many hidden layers increased the model's complexity and made it prone to overfitting. To combat overfitting, the number of hidden layers was reduced to a minimum (one or two layers), and the number of neurons was varied. The number of time steps considered for training remained at 4.

Single Hidden Layer

The structure of the network was limited to one hidden layer. However, a single hidden layer could still capture the essential features of the input data and learn relevant patterns without the added complexity and potential overfitting of deeper networks.

Neurons	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
1000	0.1100	0.3004	0.8458	0.8399	0.845	0.8514
1500	0.0825	0.2510	0.87	0.8649	0.871	0.8786
2000	0.0727	0.2234	0.8833	0.8834	0.8868	0.8924
3000	0.0644	0.2035	0.8965	0.8925	0.8968	0.9038
4500	0.0604	0.1918	0.904	0.8956	0.9051	0.9081
6000	0.0584	0.1822	0.9075	0.9025	0.9108	0.9132
9000	0.0575	0.1784	0.9108	0.9045	0.9119	0.9177
15000	**0.0573**	**0.1700**	**0.9135**	**0.9087**	**0.9153**	**0.9191**
20000	0.0576	0.1680	0.9153	0.9084	0.9152	0.9174

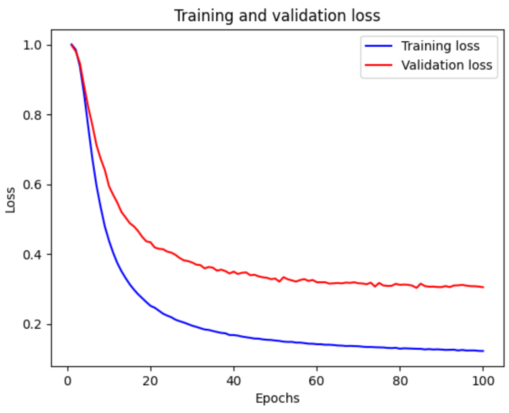
Increasing the number of neurons improved the network's ability to capture complex patterns in the input data. A significant number of neurons allowed the network to identify more features and combinations, leading to better correlations between the input and output. There was a considerable increase in correlation until 15000 neurons, after which the network performance decreased slightly. Adding too many neurons led to overfitting, as seen by the decrease in performance from 15000 neurons to 20000. As the structure of the network was not optimised, increased network complexity led to a degradation in performance. A hidden layer was added to reduce the number of neurons while retaining the network's performance.

Two Hidden Layers

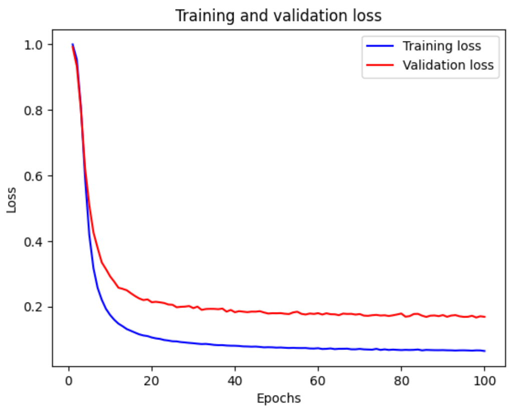
The peak performance of the single hidden-layer neural network was kept as the benchmark for the double-layer network. The result of the analysis can be seen below.

Neurons	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
500	0.0794	0.2248	0.8826	0.8792	0.8832	0.891
1000	0.0637	0.1759	0.9108	0.906	0.9104	0.9174
2000	**0.0653**	**0.1694**	**0.9153**	**0.9085**	**0.9134**	**0.9214**
3000	0.0692	0.1702	0.9129	0.9068	0.9161	0.9207

In the case of the double-layer network, there was a considerable increase in correlation until 2000 neurons, after which the network performance decreased slightly. The highest correlation achieved by both networks was the same. The peak performance of the single-layer network was matched by the double-layer network, with a significantly lower amount of neurons being required. This meant that the added complexity of the significantly high number of neurons of the single-layer network did not contribute to the network's performance. The 13000 additional neurons did not provide information that the 2000 neurons could not capture in two layers. Thus the model with fewer neurons was selected due to its lower complexity. The performance of the network before and after the analysis is compared below:

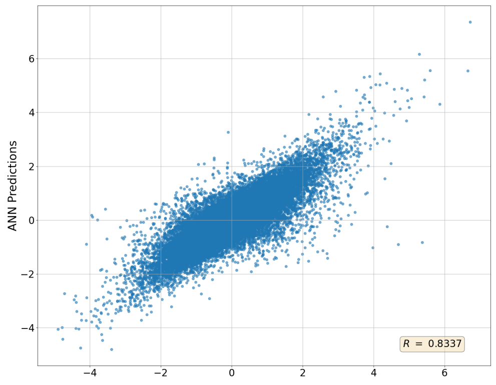


(a) Four Hidden Layers, 256 Neurons (Baseline Network)

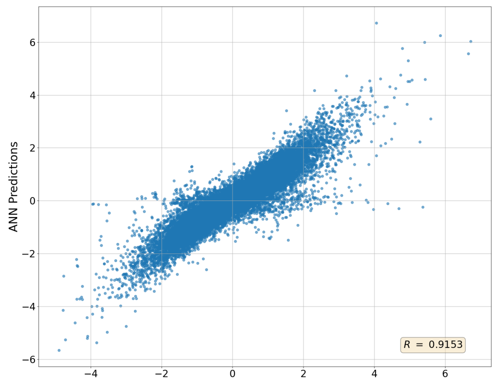


(b) Two Hidden Layers, 2000 Neurons

Figure 6.11: Effect of network structure optimization: Training and Validation Loss – 4 Time Steps



(a) Four Hidden Layers, 256 Neurons (Baseline Network)



(b) Two Hidden Layers, 2000 Neurons

Figure 6.12: Effect of network structure optimization: X-Momentum Terms – 4 Time Steps

Optimising the network structure led to a visible decrease in the difference between the losses and hence overfitting.

6.3.2. Feature Set (NO U, Ut)

The feature set affects the ability of the network to learn and generalise patterns in data. Optimising identifies features relevant to the problem, while irrelevant or redundant features that add to the noise in the data are removed, making the model more interpretable.

Feature Removed	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
-	0.0653	0.1694	0.9153	0.9085	0.9134	0.9214
U	0.0562	0.1648	0.9176	0.9129	0.9193	0.9231
U, Ut	0.0459	0.1511	0.9246	0.9192	0.9257	0.9285

Removing velocity from the feature set increased the prediction accuracy, so it appeared to be a redundant feature. However, the feature set already contained information related to the velocity of the elements through the spatial gradients of velocity. When different features in the input data are highly correlated, removing one improves the network's performance. Redundant features providing the same information to the neural network can otherwise lead to overfitting.

It was also seen that removing the time derivative from the feature set increased prediction accuracy. The analysis was carried out on a flow with a low courant number indicating that the time step used in the simulation was relatively small compared to grid spacing and flow velocity. This implied that unresolved spatial components of the solution would be more relevant than unresolved temporal components. Thus, a small Δt was considered to minimise the influence of the specific time march chosen. This may also reduce the relevance of Ut to the prediction of the closure terms. The feature set is thus comprised of only spatial gradient terms to streamline the required features and make the model more interpretable.

6.3.3. Stencil Studies

The choice of stencil is crucial because it determines the set of input features to which the neural network has access. In Figure 5.3, only the immediate (I) neighbouring elements were initially included in the stencil. This section investigates the effect of a change in the stencil. A stencil with extra elements along either side of the x-direction and alternatively extending the stencil to include two immediate neighbours (6×2 elements in stencil) instead of one (6 elements) was tested.

Stencil Change	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
X-direction (X + 1)	0.0634	0.1699	0.9122	0.9109	0.9166	0.9157
I + 1	0.0779	0.1852	0.9059	0.9003	0.9068	0.9126

Including extra elements in the x-direction or doubling the stencil does not have any noticeable impact on the correlations of the network. This means the additional stencil points could not provide any new relevant information that could improve the network's ability to predict the outcome. The additional stencil points are highly correlated with the existing stencil points. The network is already optimised and has reached its peak performance with the current stencil size. Hence the initial stencil with six elements is continued.

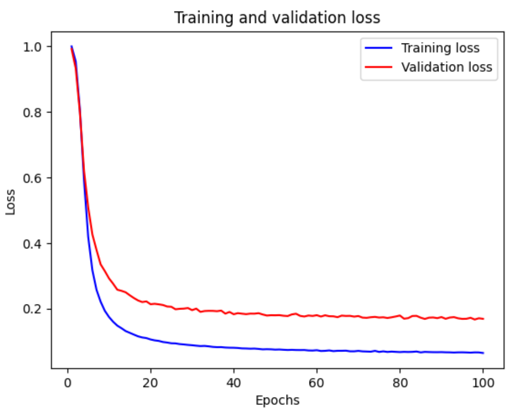
6.3.4. Batch Size Variation

This part of the analysis deals with the batch size of the neural network or the number of training examples utilised in one pass of the network. In neural network training, the data is typically divided into batches, and the network processes each batch before updating its parameters. Therefore, the choice of batch size is a crucial hyper-parameter that can affect the training process and the model's performance. The batch size was initially increased from 32 to 250. Subsequent variations in batch size can be seen below:

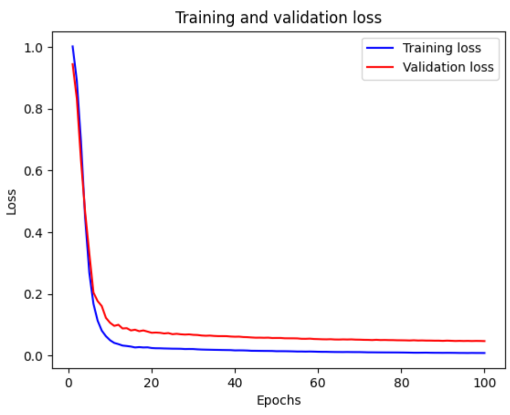
Batch Size	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
250	0.0125	0.0509	0.9746	0.9748	0.9752	0.9766
500	0.0094	0.0469	0.9767	0.9766	0.9768	0.9789
750	**0.0082**	**0.0469**	**0.9759**	**0.9774**	**0.977**	**0.9787**
1000	0.0080	0.0489	0.9758	0.9763	0.9766	0.978

Raising the batch size in a neural network resulted in the enhancement of the network's performance. Once the batch size of the network was increased from 32 to 250, the network's performance improved drastically. The large batch meant that each update to the weights and biases of the network was based on a larger sample of the training data, which provided a more accurate estimate of the actual gradient. This reduced the gradient estimate's variance and helped reduce noise in the optimisation process. In addition, a larger batch size allows the model to take more steps toward the actual gradient before updating the weights. This results in faster convergence during training and reduced computational expense.

However, a larger batch size also means the model cannot generalise to new data. This is because the batch provides a snapshot of the training data, and a larger batch size means that the model sees fewer unique examples during each training step. Therefore, the network's performance is maximum at the batch size of 750. The training and validation loss graph, along with the correlation of the network for a batch size of 750, are compared below with the network performance after structure optimisation at the beginning of this section:

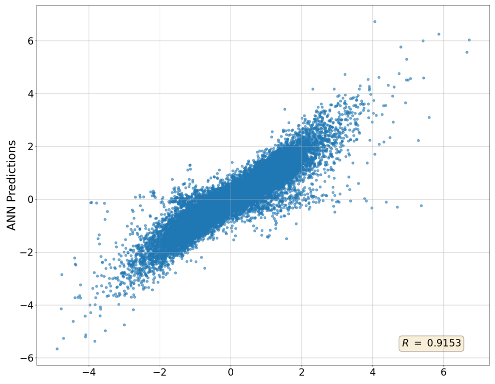


(a) Batch Size – 32

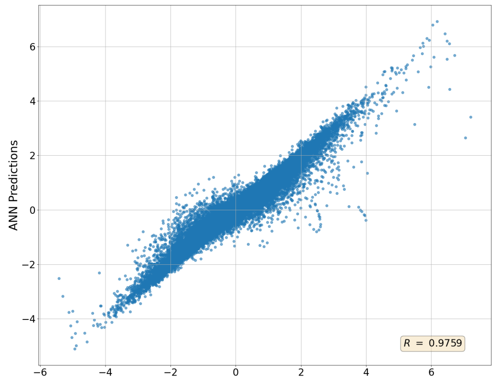


(b) Batch Size – 750

Figure 6.13: Effect of batch size variation: Training and Validation Loss – 4 Time Steps, Two Hidden Layers, 2000 Neurons



(a) Batch Size – 32



(b) Batch Size – 750

Figure 6.14: Effect of batch size variation: X-Momentum Terms – 4 Time Steps, Two Hidden Layers, 2000 Neurons

An increase in the batch size led to a visible decrease in the difference between losses, and hence overfitting. With larger batch sizes, the gradient updates became more stable and less affected by the noise in individual samples, which can help the network to learn more robust representations of the input data.

6.3.5. Reflection

The section started with a study analysing the impact of a neural network's structure on its accuracy. A single hidden layer network was first tested, which could capture the essential features of input data without the added complexity and potential overfitting of deeper networks. The study showed that increasing the number of neurons improved the network's ability to capture complex patterns, with a considerable increase in correlation until 15000 neurons. However, adding too many neurons led to overfitting, which reduced performance (seen from 15000 neurons to 20000). Increased network complexity led to a degradation in performance. Therefore, a hidden layer was added to reduce the number of neurons and retain the performance of the network. The double-layer network's peak performance was kept as a benchmark for comparison to the single-layer network, which required fewer neurons. The analysis showed that the highest correlation achieved by both networks was the same. Thus the model with the lower number of neurons because the added complexity of the single-layer network did not contribute to its performance.

Next, feature set optimisation was carried out, and it was seen that removing velocity from the feature set increased the prediction accuracy because it was a redundant feature. The feature set already contained information related to the velocity of the elements. Additionally, it was found that the time derivative feature was irrelevant to the feature set. Therefore, the feature set comprised only spatial gradient terms to streamline the required features and make the model more interpretable.

Next, the effect of a change in the stencil used for the neural network was investigated. First, a stencil with extra elements along either side of the x-direction and extending the stencil to include two immediate neighbours were tried. However, the analysis showed that doubling the stencil size or including extra elements in the x-direction had no noticeable impact on the network's correlations. Hence, the initial stencil had six elements continued.

Lastly, it was seen that increasing the batch size from 32 to 250 decreased the difference between training and validation losses, indicating less overfitting. In addition, the study showed that increasing the batch size allowed the model to generalise better by limiting the model's ability to fit noise in the data.

6.4. Phase III: Post-Batch Size Correction

The previous section discussed the importance of optimising a neural network's structure, feature set, stencil size, and batch size to reduce overfitting and improve performance. The correlation network predictions are nearing the target of 99%. Hence, this section focuses on understanding the minimum network needed to achieve this benchmark computationally inexpensively.

6.4.1. Neuron-Hidden Layer Analysis

The present network has two hidden layers and 2000 neurons. The number of neurons and hidden layers was varied to understand the computational expense of the network structure.

Two Hidden Layers

The network structure was limited to two hidden layers, and the number of neurons was varied. It was observed that the performance of the network plateaued at around 2000 neurons. Decreasing the number of neurons beyond 2000 decreased the network's performance. A decrease in the number of neurons in a network reduces the complexity, making it easier and faster to train. However, it also reduced the representational capacity of the network, making it less able to capture complex patterns in the data. Conversely, configurations with neurons greater than 2000 came at the cost of network complexity yet did not give any significant increase in correlation. Finally, the structure with 2000 neurons was selected as it was the right balance between performance and network complexity.

Neurons	Train Loss	Test Loss	X-Mom (Sum.)	Y-Mom (Sum.)	Z-Mom (Sum.)	Cont.
100	0.6117	0.7499	0.4696	0.4416	0.5223	0.6061
500	0.0513	0.1700	0.9148	0.9115	0.9143	0.9177
1000	0.0130	0.0688	0.9646	0.9651	0.966	0.9682
1500	0.0090	0.0505	0.9737	0.9741	0.9751	0.9766
2000	**0.0082**	**0.0469**	**0 .9759**	**0.9774**	**0.977**	**0.9787**
2500	0.0078	0.0443	0.9776	0.9782	0.9781	0.9793
3000	0.0077	0.0443	0.9783	0.9781	0.9789	0.9799

Three Hidden Layers

The network's performance with two hidden layers and 2000 neurons was kept as a benchmark for the following analysis. With three hidden layers, the network performance converged at around 2000 neurons. The network's performance was very similar to that of two hidden layers but at an increased complexity. This indicated that the network architecture with three hidden layers was not optimised for the given task. The extra hidden layer did not provide any additional helpful information to the network to benefit from the added complexity. Increasing the hidden layers also comes with the risk of overfitting. Hence the network with 2000 neurons and two hidden layers was chosen for the subsequent analyses.

Neurons	Train Loss	Test Loss	X-Mom (Sum.)	Y-Mom (Sum.)	Z-Mom (Sum.)	Cont.
100	0.5231	0.6866	0.5395	0.5217	0.5775	0.6369
500	0.0213	0.1070	0.9443	0.9431	0.9453	0.9492
1000	0.0094	0.0592	0.9698	0.9704	0.9706	0.9726
1500	0.0074	0.0499	0.9739	0.9756	0.9758	0.9776
2000	0.0066	0. 438	0.9784	0.9787	0.9795	0.9805
2500	0.0062	0.0414	0.9792	0.9796	0.9796	0.9816

6.4.2. Data-Set Size Variation

Previously, the structure of the network and its complexity were considered to find the minimum network that would produce the target correlation above 99%. Now the amount of training data required to produce the desired correlation is analysed.

No. of Time Steps	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
8	0.0082	0.0469	0.9759	0.9774	0.977	0.9787
9	0.0082	0.0427	0.9796	0.9792	0.9798	0.9808
10	0.0077	0.0387	0.9814	0.9804	0.9806	0.9814
11	0.0078	0.0369	0.9822	0.9813	0.9813	0.9821
12	0.0079	0.0349	0.9826	0.9825	0.9821	0.9834
13	0.0079	0.033	0.984	0.9831	0.9833	0.9846
14	0.0076	0.0307	0.9849	0.9843	0.9847	0.9856
15	0.0076	0.0291	0.9854	0.9846	0.9854	0.986
16	0.0078	0.0292	0.9858	0.985	0.9853	0.9864
17	0.0078	0.0287	0.9861	0.9856	0.9861	0.9865
18	0.0078	0.0274	0.9864	0.9859	0.9864	0.987
19	0.0079	0.0273	0.9864	0.985	0.9866	0.9868
20	**0.0082**	**0.0269**	**0.9868**	**0.9859**	**0.9864**	**0.9872**

An increase in training data did not improve the network's performance by a considerable margin. The network has reached the saturation of learning. It has learnt all the essential features and patterns in the data and can no longer improve its performance significantly with additional data. As a result, the network's accuracy plateaued, even with the addition of more training data. The highest performance of the network observed with 20 time steps can be seen below:

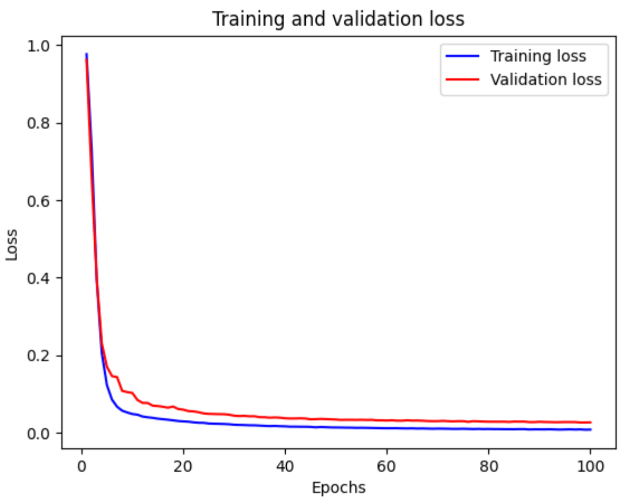
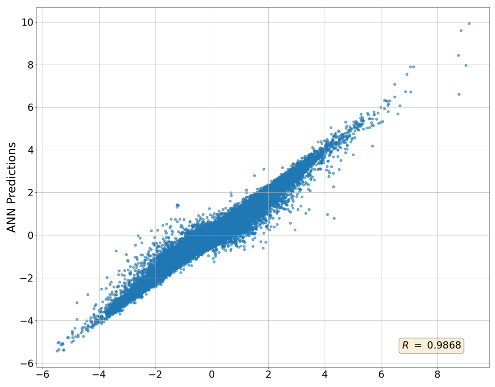
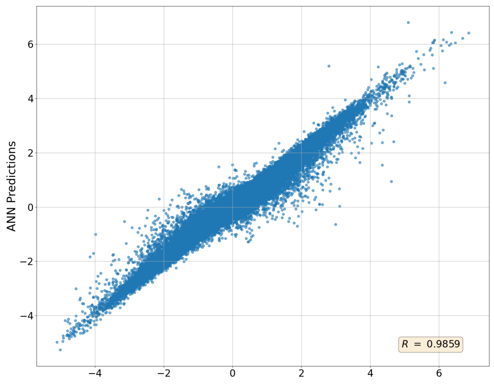


Figure 6.15: Training and Validation loss – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size–750

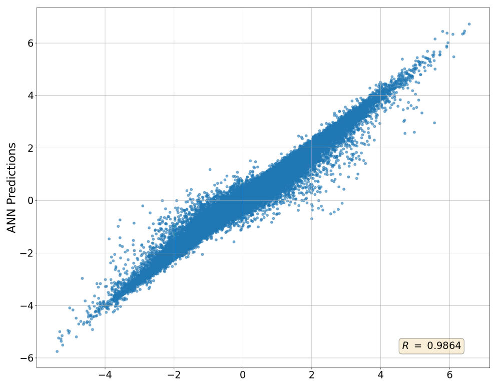


(a) X

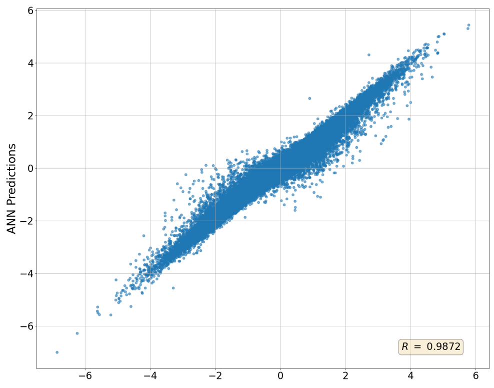


(b) Y

Figure 6.16: X and Y Momentum Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size–750



(a) Z



(b) Cont.

Figure 6.17: Z Momentum and Continuity Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size–750

An increase in the training data size led to a visible decrease in the difference between losses and hence overfitting. The network observed saturation of learning. This can be tackled by increasing the number of training epochs. Hence, a study on the correlation between the number of training epochs and network performance will also be conducted.

6.4.3. Epochs

An increase in epochs does increase the correlation of the network predictions. As the network was not overfitting, the increase in epochs led to an increase in the correlation between the predicted and actual output. In addition, an increase in epochs allowed the model to continue learning and refining its weights and biases to capture the underlying patterns in the data better. As the model got closer to the optimal set of weights and biases, the correlation between the predicted and actual output increased. Moreover, the target correlation of 99% was achieved at 300 epochs.

Epochs	Train Loss	Test Loss	X-Mom (Avg.)	Y-Mom (Avg.)	Z-Mom (Avg.)	Cont.
100	0.0089	0.0242	0.988	0.9868	0.9874	0.9884
300	0.005	0.0186	0.9909	0.99	0.9906	0.9911
400	0.0046	0.0175	0.9913	0.9903	0.9909	0.9915
1000	0.0039	0.0166	0.9917	0.9909	0.9918	0.9921

The training and validation loss graphs, along with the network correlations, are seen below:

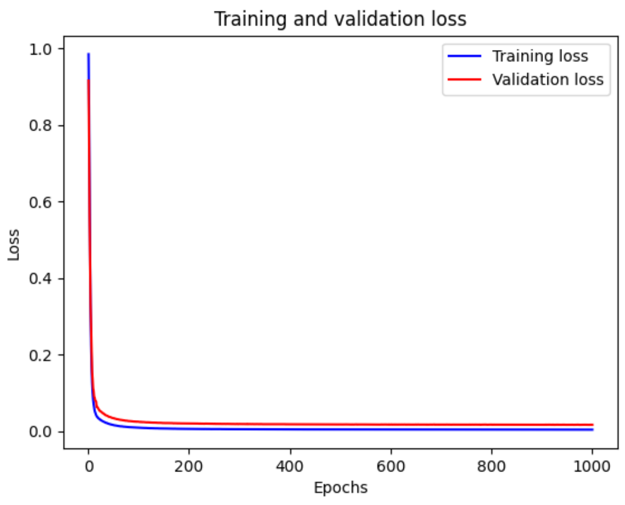
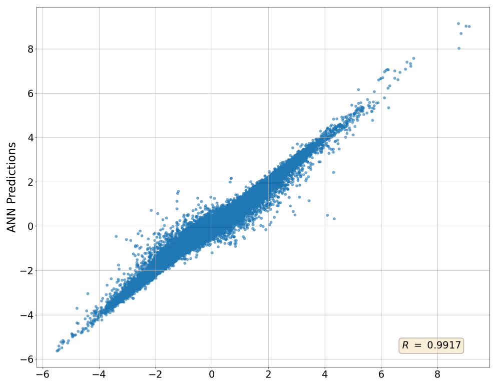
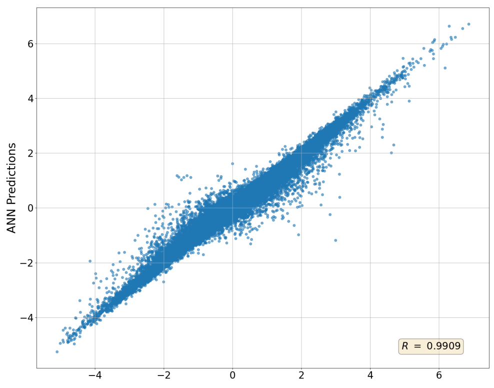


Figure 6.18: Training and Validation loss – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size – 750, – 1000 Epochs

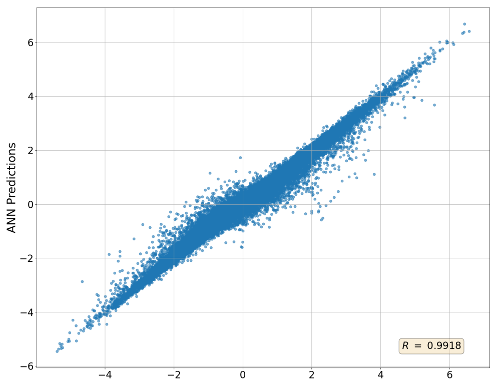


(a) X

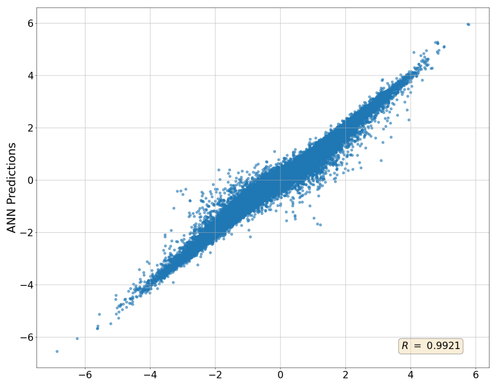


(b) Y

Figure 6.19: X and Y Momentum Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size – 750, – 1000 Epochs



(a) Z



(b) Cont.

Figure 6.20: Z Momentum and Continuity Terms – 20 Time Steps, Two Hidden Layers, 2000 Neurons, Batch Size – 750, – 1000 Epochs

6.4.4. Reflection

This analysis section discussed the third phase of neural network optimisation, i.e., post-batch size correction, and focused on understanding the minimum network required to achieve a benchmark of 99% correlation. The number of neurons and hidden layers of the present network were varied to find the minimum network structure required. It was observed that the network's performance with two hidden layers plateaued at 2000 neurons, making it the right balance between performance and computational complexity. Further, the network's performance with three hidden layers was similar to that of two hidden layers but at an increased complexity. Hence, the network with 2000 neurons and two hidden layers was chosen for subsequent analyses.

Next, the amount of training data required to produce the desired correlation was analysed. An increase in training data did not improve the network's performance by a considerable margin. The network was observed to have reached saturation of learning, having learned all the essential features and patterns in the data. Thus, the network's accuracy plateaued, even with the addition of more training data. Therefore, increasing the number of training epochs was the solution to tackle the saturation of learning.

Finally, an increase in the number of epochs increased the correlation of the network predictions. As the network was not overfitting, an increase in the number of epochs led to an increase in the correlation between the predicted and actual output. Hence, a study on the correlation between the number of training epochs and network performance was also conducted. Overall, the analysis emphasises the importance of optimising a neural network's structure, training data size, and training epoch size to reduce overfitting and improve performance.

7 Conclusions

The analysis focused on building a simple neural network to predict the interaction terms for 3D turbulent channel flow. Previous work by Rajampeta (2021) focused on one-dimensional flow. Hence, it was essential to ascertain the accuracy of the methodology in three dimensions. A multi-layer perception (MLP) architecture was used to try the methodology. The feature set comprised basic macroscopic flow properties like velocity, pressure, and spatial-temporal gradients. In addition, the correlation between the ANN predictions and actual DNS data was used as a parameter to measure the neural network's performance.

Initial results using the same architecture used by Rajampeta (2021) showed that overfitting was a significant issue with the ANN, leading to poor performance on new test data. To address this, the size of the training data set was increased, which improved the ANN's performance and decreased the gap between the training and validation loss. However, further increasing the training data set did not improve the accuracy of the ANN, indicating that the network could not capture the underlying patterns in the data. In addition, a discrepancy with the normalisation of the data used in the ANN was identified, which led to the loss of spatial information and biases in the model. This was rectified by normalising the data plane by plane using an automated normalisation function, significantly improving the ANN's performance.

Next, the impact of the neural network's structure on its accuracy was studied. It was found that a single hidden layer network could capture the essential features of input data without the added complexity and potential overfitting of deeper networks. Increasing the number of neurons improved the network's ability to capture complex patterns, but too many neurons led to overfitting, reducing performance. Therefore, a hidden layer was added to reduce the number of neurons and retain the network's performance. Furthermore, optimisation of the feature set was also carried out. It found that removing redundant features and focusing on spatial gradient terms increased the prediction accuracy and made the model more interpretable. Next, changing the stencil used in the neural network found that increasing the stencil size had no noticeable impact on the network's correlations. Increasing the batch size from 32 to 250 decreased the difference between training and validation losses, indicating less overfitting and better generalisation. Finally, it was seen that the network's performance plateaued with two hidden layers and 2000 neurons, making it the right balance between performance and complexity.

The amount of training data required to achieve the desired correlation was analysed. It was seen that an increase in training data did not improve the network's performance significantly, and the network had reached saturation of learning, having learned all the essential features and patterns in the data. Therefore, increasing the number of training epochs was the solution to tackle the saturation of learning. Finally, an increase in the number of epochs led to an increase in the correlation between the predicted and actual output.

Overall, the study shows the importance of optimising the ANN's structure, training data size, and training epoch size to reduce overfitting and improve performance computationally inexpensively. In addition, the study highlights the importance of preserving spatial information during the normalisation process.

7.1. Research Questions and Answers

The research questions posed in section 1.6 are now answered:

1. Can the neural network (structure and hyper-parameters) used in 1D turbulent channel flow (Rajampeta (2021)) be adapted to three dimensions?

   It was seen in section 6.1 that directly adopting the structure and hyper-parameters used in Rajampeta (2021) is insufficient for the current analysis. Furthermore, the ANN used by Rajampeta (2021) applied to 3D turbulent channel flow showed overfitting, leading to poor performance in new test data. This necessitated an analysis of the network capable of reproducing the closure terms for 3D turbulent channel flow.

2. How does the network structure affect the ANN's accuracy?

   2.1 What is the effect of widening the network?

   In subsection 6.3.1, the number of hidden layers was kept to one, and the number of neurons was increased. A single hidden layer could still capture the critical features of the input data and learn relevant patterns without the added complexity and potential overfitting of deeper networks. However, increasing the number of neurons improved the network's ability to capture complex patterns in the input data. In addition, a more significant number of neurons allowed the network to identify more features and combinations, leading to better correlations between the input and output.

   2.2 What is the effect of deepening the network?

   In subsection 6.3.1, a hidden layer was added to reduce the number of neurons while retaining the performance of the network. The inclusion of another hidden layer did help reduce the number of neurons necessary to achieve peak performance.

3. Can the basic, macroscopic properties of the flow (velocity, pressure, and their gradients) be used as features to find the subgrid-scale terms?

   Basic macroscopic properties such as velocity, pressure and spatiotemporal gradients were used as the network features and produced excellent correlations.

   3.1 Which of the features are most important for high correlation?

   A small Δt was considered to minimise the influence of the specific time march chosen. Velocity in the feature set turned out to be a redundant feature. The feature set already contained information related to the velocity of the elements through the spatial gradients of velocity.

4. What is the simplest ANN structure which can accurately (99%) reproduce the closure terms?

   The ANN performs well in the present analysis at a low courant number. The number of time steps required is 20, i.e. 600000 training samples.

   Final Architecture

Layer	Neurons	Activation
Input Layer	140	–
Layer 1	2000	ReLU
Layer 2	2000	ReLU
Output	4	ReLU

In subsection 6.4.3, it was seen that the network took 300 epochs at 9/10s per epoch and an evaluation time of 9s to produce the closure term with a correlation of 99%.

7.2. Recommendations

Since we focused on low CFL, the effects of the time discretisation were not significant. Further studies on this subject would be recommended to increase CFL number first and analyse whether MLP architecture is sufficient for the analysis. Otherwise, network architectures with more inherent temporal capabilities, such as a Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) type RNNs, are recommended as they selectively remember or forget previous inputs to improve performance on sequence tasks. A more significant amount of data and a more complex architecture may be required to process the data.

Using the Grid-Search algorithm to choose the best-performing architecture efficiently is recommended. As explained earlier, the network's feature set comprises essential macroscopic properties such as velocity, pressure, and gradients. This enabled us to get a primitive understanding of the most critical parameters required. However, further analysis can be done with more specific terms of the variational formulation, such as the tri-linear and bi-linear terms, to investigate the correlations and interactions of the resolved, unresolved terms.

A
Derivation of the closure terms

The x,y,and z components of Equation 3.14 in the convective form (retaining the symmetric velocity gradient) can be written as:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}+\frac{\partial p}{\partial x}-\nu[2\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}+\frac{\partial^2 v}{\partial x\partial y}+\frac{\partial^2 w}{\partial x\partial z}]=f_x \tag{A.1}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}+\frac{\partial p}{\partial y}-\nu[\frac{\partial^2 v}{\partial x^2}+2\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}+\frac{\partial^2 u}{\partial x\partial y}+\frac{\partial^2 w}{\partial y\partial z}]=f_y \tag{A.2}$$

$$\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}+\frac{\partial p}{\partial z}-\nu[\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+2\frac{\partial^2 w}{\partial z^2}+\frac{\partial^2 u}{\partial x\partial z}+\frac{\partial^2 v}{\partial y\partial z}]=f_z \tag{A.3}$$

Let us consider the x-component for our derivation:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}+\frac{\partial p}{\partial x}-\nu[2\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}+\frac{\partial^2 v}{\partial x\partial y}+\frac{\partial^2 w}{\partial x\partial z}]=f_x \tag{A.4}$$

With reference to Equation 3.31, the weak form of the x-momentum equation above can be written as:

$$(q,u_t)+(q,uu_x)+(q,vu_y)+(q,wu_z)+(q,p_x)-2\nu\,(q,u_{xx})-\nu\,(q,u_{yy})-\nu\,(q,u_{zz})-\nu\,(q,v_{xy})-\nu\,(q,w_{xz})=(q,f_x) \tag{A.5}$$

$$\begin{aligned}\underbrace{\big(q,(u^h_t+u'_t)\big)}_{1}+\underbrace{\big(q,(u^h+u')(u^h_x+u'_x)\big)}_{2}+\underbrace{\big(q,(v^h+v')(u^h_y+u'_y)\big)}_{3}+\underbrace{\big(q,(w^h+w')(u^h_z+u'_z)\big)}_{4}+\underbrace{\big(q,(p^h_x+p'_x)\big)}_{5}\\-2\nu\underbrace{\big(q,(u^h_{xx}+u'_{xx})\big)}_{6}-\nu\underbrace{\big(q,(u^h_{yy}+u'_{yy})\big)}_{7}-\nu\underbrace{\big(q,(u^h_{zz}+u'_{zz})\big)}_{8}\\-\nu\underbrace{\big(q,(v^h_{xy}+v'_{xy})\big)}_{9}-\nu\underbrace{\big(q,(w^h_{xz}+w'_{xz})\big)}_{10}=\underbrace{(q^h,f_x)}_{11}\end{aligned} \tag{A.6}$$

The term-wise expansion of the terms is given. For the sake of brevity, the h and ’ symbols denoting the scale of the terms have been omitted during the derivation:

1. $(q, u_t^h) + (q, u_t')$
2. $(q, u^h u_x^h) + (q, u^h u_x') + (q, u' u_x^h) + (q, u' u_x')$

   Applying integration by parts (IBP) we get: $(q, u^h u_x^h) - (q_x, u^h u') - \left(q_x, \frac{u'u'}{2}\right)$
3. $(q, v^h u_y^h) + (q, v^h u_y') + (q, v' u_y^h) + (q, v' u_y')$

   IBP: $(q, v^h u_y^h) - (q_y, v^h u') - (q, v_y^h u') + (q, v' u_y^h) - (q_y, v' u') - (q, v_y' u')$
4. $(q, w^h u_z^h) + (q, w^h u_z') + (q, w' u_z^h) + (q, w' u_z')$

   IBP: $(q, w^h u_z^h) - (q_z, w^h u') - (q, w_z^h u') + (q, w' u_z^h) - (q_z, w' u') - (q, w_z' u')$
5. $(q, p_x^h) + (q, p_x')$, IBP: $(q, p_x^h) - (q_x, p')$

   Similarly...
6. $-2\nu(q, u_{xx}^h) + 2\nu(q_x, u_x')$
7. $-\nu(q, u_{yy}^h) + \nu(q_y, u_y')$
8. $-\nu(q, u_{zz}^h) + \nu(q_z, u_z')$
9. $-\nu(q, v_{xy}^h) + \nu(q_x, v_y')$
10. $-\nu(q, w_{xz}^h) + \nu(q_x, w_z')$

Finally we have:

x-momentum equation

q is the weight constant

$$\begin{aligned}
&(q^h, u_t^h) + (q^h, u^h u_x^h) + (q^h, v^h u_y^h) + (q^h, w^h u_z^h) + (q^h, p_x^h) - 2\nu(q^h, u_{xx}^h) - \nu(q^h, v_{xy}^h) - \nu(q^h, w_{xz}^h) \\
&\quad -\nu(q^h, u_{yy}^h) - \nu(q^h, u_{zz}^h) + (q^h, u_t') - (q_x^h, u^h u') - \left(q_x^h, \frac{u'u'}{2}\right) \\
&\quad + \left[-(q_y^h, v^h u') - (q^h, v_y^h u') + (q^h, v' u_y^h) - (q_y^h, v' u') - (q^h, v_y' u')\right] \\
&\quad + \left[-(q_z^h, w^h u') - (q^h, w_z^h u') + (q^h, w' u_z^h) - (q_z^h, w' u') - (q^h, w_z' u')\right] \\
&\quad -(q_x^h, p') + 2\nu(q_x^h, u_x') + \nu(q_x^h, v_y') + \nu(q_x^h, w_z') + \nu(q_y^h, u_y') + \nu(q_z^h, u_z') = (q^h, f_x)
\end{aligned} \tag{A.7}$$

y-momentum equation

r is the weight constant

$$\begin{aligned}
&(r^h, v_t^h) + (r^h, u^h v_x^h) + (r^h, v^h v_y^h) + (r^h, w^h v_z^h) + (r^h, p_y^h) - \nu(r^h, v_{xx}^h) - \nu(r^h, u_{yx}^h) - 2\nu(r^h, v_{yy}^h) \\
&\quad -\nu(r^h, w_{yz}^h) - \nu(r^h, v_{zz}^h) + (r^h, v_t') + \left[-(r_x^h, u^h v') - (r^h, u_x^h v') + (r^h, u' v_x^h) - (r_x^h, u' v') - (r^h, u_x' v')\right] \\
&\quad -(r_y^h, v^h v') - \left(r_y^h, \frac{v'v'}{2}\right) + \left[-(r_z^h, w^h v') - (r^h, w_z^h v') + (r^h, w' v_z^h) - (r_z^h, w' v') - (r^h, w_z' v')\right] \\
&\quad -(r_y^h, p') + \nu(r_x^h, v_x') + \nu(r_y^h, u_x') + 2\nu(r_y^h, v_y') + \nu(r_y^h, w_z') + \nu(r_z^h, v_z') = (r^h, f_y)
\end{aligned} \tag{A.8}$$

z-momentum equation

m is the weight constant

$$(m^h, w_t^h) + (m^h, u^h w_x^h) + (m^h, v^h w_y^h) + (m^h, w^h w_z^h) + (m^h, p_z^h) - \nu(m^h, w_{xx}^h) - \nu(m^h, w_{yy}^h) - \nu(m^h, u_{zx}^h)$$
$$-\nu(m^h, v_{zy}^h) - 2\nu(m^h, w_{zz}^h) + (m^h, w_t') + \left[-(m_x^h, u^h w') - (m^h, u_x^h w') + (m^h, u' w_x^h) - (m_x^h, u' w') - (m^h, u_x' w')\right]$$
$$+\left[-(m_y^h, v^h w') - (m^h, v_y^h w') + (m^h, v' w_y^h) - (m_y^h, v' w') - (m^h, v_y' w')\right] - (m_z^h, w^h w') - (m_z^h, \frac{w' w'}{2})$$
$$-(m_z^h, p') + \nu(m_x^h, w_x') + \nu(m_y^h, w_y') + \nu(m_z^h, u_x') + \nu(m_z^h, v_y') + 2\nu(m_z^h, w_z') = (m^h, f_z) \quad (A.9)$$

continuity equation

l is the weight constant

$$(l^h, u_x^h) + (l^h, v_y^h) + (l^h, w_z^h) + (l^h, u_x') + (l^h, v_y') + (l^h, w_z') = 0 \quad (A.10)$$

Bibliography

1. Adrian, R. J., Christensen, K. T. & Liu, Z. C. Analysis and interpretation of instantaneous turbulent velocity fields. *Experiments in fluids,* 275–290. https://doi.org/10.1007/s003489900087 (2000).
2. Aggarwal, C. C. *Neural Networks and Deep Learning* 3. ISBN: 978-3-319-94463-0. https://doi.org/10.1007/978-3-319-94463-0 (Springer Cham, 2018).
3. Baliga, B. R. & Patankar, S. V. A control volume finite-element method for two-dimensional fluid flow and heat transfer. *Numerical Heat Transfer,* 245–261. https://doi.org/10.1080/01495728308963086 (1983).
4. Bardina, J., Ferziger, J. H. & Reynolds, W. C. IMPROVED SUBGRID-SCALE MODELS FOR LARGE-EDDY SIMULATION. *AIAA Paper.* www.scopus.com (1980).
5. Bazilevs, Y. *et al.* Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering* **197,** 173–201. https://doi.org/10.1016/j.cma.2007.07.016 (2007).
6. Beekman, M. Using artificial neural networks as unresolved-scale models for the Burgers equation: an investigation of their suitability and robustness. *TU Delft* **MA thesis.** http://resolver.tudelft.nl/uuid:e0188c65-531b-451b-ac1b-14fb04ca8ba8 (2016).
7. Besson, O. & Laydi, M. R. Some estimates for the anisotropic Navier-Stokes equations and for the hydrostatic approximation. *ESAIM: Mathematical Modelling and Numerical Analysis,* 855–865. https://doi.org/10.1006/gmip.1995.1012 (1992).
8. Breuer, M., Lakehal, D. & Rodi, W. Flow around a surface mounted cubical obstacle: comparison of LES and RANS-results. *In Computation of Three-Dimensional Complex Flows: Proceedings of the IMACS-COST Conference on Computational Fluid Dynamics Lausanne,* 22–30. https://doi.org/10.1007/978-3-322-89838-8_4 (1995).
9. Chassagnon, G., Vakalopolou, M., Paragios, N. & Revel, M. Deep learning: definition and perspectives for thoracic imaging. *European radiology* **30,** 2021–2030. https://doi.org/10.1007/s00330-019-06564-3 (2020).
10. Duraisamy, K., Zhang, Z. & Singh, A. New approaches in turbulence and transition modeling using data-driven techniques, 1284. https://doi.org/10.1007/10.2514/6.2015-1284 (2015).
11. Duraisamy, K. & Durbin, P. Transition modeling using data driven approaches. *Center for Turbulence Research: Summer Program 2014,* 427. https://www.researchgate.net/publication/319451256 (2014).
12. Durieux, T. Exploring the use of artificial neural network based subgrid scale models in a variational multiscale formulation. *TU Delft* **MA thesis.** http://resolver.tudelft.nl/uuid:de29e705-7426-4ebf-a843-2261b72e99dd (2015).
13. Feng, J., Acton, M., Baglietto, E., Kraus, A. R. & Merzari, E. On the relevance of turbulent structures resolution for cross-flow in a helical-coil tube bundle. *Annals of Nuclear Energy,* 107298. https://doi.org/10.1016/j.anucene.2019.107298 (2020).
14. Ge, X., Arolla, S. & Durbin, P. A bypass transition model based on the intermittency function. *Flow, Turbulence and Combustion* **93,** 37–61. https://doi.org/10.1007/s10494-014-9533-9 (2014).
15. Georgiadis, N. J. & DeBonis, J. R. Navier–Stokes analysis methods for turbulent jet flows with application to aircraft exhaust nozzles. *Progress in Aerospace Sciences,* 377–418. https://doi.org/10.1016/j.paerosci.2006.12.001 (2006).

16. Germano, M., Piomelli, U., Moin, P. & Cabot, W. H. A dynamic subgrid□scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics,* 1760–1765. https://doi.org/10.1063/1.857955 (1991).

17. Ghalichi, F. *et al.* Low Reynolds number turbulence modelling of blood flow in arterial stenoses. *Biorheology,* 281–294. https://doi.org/10.1016/S0006-355X(99)80011-0 (1998).

18. Guan, Y., Chattopadhyay, A., Subel, A. & Hassanzadeh, P. Stable a posteriori LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher Re via transfer learning. *Journal of Computational Physics* **458.** https://doi.org/10.1016/j.jcp.2022.111090 (2022).

19. Hughes, T. J. R. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering* **127,** 387–401. https://doi.org/10.1016/0045-7825(95)00844-9 (1995).

20. Hughes, T. J. R., Feijóo, G. R., Mazzei, L. & Quincy, J. The variational multiscale method - A paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering* **166,** 3–24. https://doi.org/10.1016/S0045-7825(98)00079-6 (1998).

21. Janssens, M. Machine Learning of Atmospheric Turbulence in a Variational Multiscale Model. *TU Delft* **MA thesis.** http://resolver.tudelft.nl/uuid:bd090309-305e-4c04-93b7-64f1b79df8d4 (2019).

22. Jindal, S. K., Banerjee, S., Patra, R. & Paul, A. in *Brain Tumor MRI Image Segmentation Using Deep Learning Techniques* (ed Chaki, J.) 135–161 (Academic Press, 2022). ISBN: 978-0-323-91171-9. https://doi.org/10.1016/B978-0-323-91171-9.00008-9.

23. Klein, M. An attempt to assess the quality of large eddy simulations in the context of implicit filtering. *Flow, Turbulence and Combustion,* 131–147. https://doi.org/10.1007/s10494-005-8581-6 (2005).

24. Kurian, N. Subgrid-Scale model using Artificial Neural Networks for Wall-bounded Turbulent flows. *TU Delft* **MA thesis.** http://resolver.tudelft.nl/uuid:e6b50795-6d2c-45fa-b308-9e5bd8ff5a44 (2019).

25. Kurz, M. & Beck, A. A MACHINE LEARNING FRAMEWORK FOR LES CLOSURE TERMS. *Electronic Transactions on Numerical Analysis* **56,** 117–137. https://doi.org/10.1553/ETNA_VOL56S117 (2021).

26. Lee, C., Kim, J., Babcock, D. & Goodman, R. Application of neural networks to turbulence control for drag reduction. *Physics of Fluids* **9,** 1740–1747. https://doi.org/10.1063/1.869290 (1997).

27. Meng, X., Li, Z., Zhang, D. & Karniadakis, G. E. PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Computer Methods in Applied Mechanics and Engineering* **370.** https://doi.org/10.1016/j.cma.2020.113250 (2020).

28. Milano, M. & Koumoutsakos, P. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics* **182,** 1–26. https://doi.org/10.1007/10.1006/jcph.2002.7146 (2002).

29. Noll, W. Derivation of the fundamental equations of continuum thermodynamics from statistical mechanics. https://doi.org/10.48550/arXiv.0810.0337 (2009).

30. Parish, E. J. & Duraisamy, K. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics* **305,** 758–774. ISSN: 0021-9991. https://www.sciencedirect.com/science/article/pii/S0021999115007524 (2016).

31. Prof.Dr.ir. S. Hickel. CFD for Aerospace Engineers (AE4202). *Technische Universiteit Delft (TU Delft)* (2020).

32. Pusuluri, A. Noise-augmented offline training of ANN unresolved scale models. *TU Delft* **MA thesis.** http://resolver.tudelft.nl/uuid:03b345f6-464a-4dd2-ac50-4fa9c880c839 (2020).

33. Raissi, M., Perdikaris, P. & Karniadakis, G. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. https://doi.org/10.48550/arXiv.1711.10561 (2017).

34. Rajampeta, V. Stabilization strategies for a variational multiscale method coupled with an artificial neural network. *TU Delft* **MA thesis.** https://repository.tudelft.nl/islandora/object/uuid:2114d9a4-7333-47bd-9793-77721a8303b0?collection=education (2021).

35. Robijns, M. A Machine Learning Approach to Unresolved-Scale Modeling for Burgers' Equation. *TU Delft* **MA thesis.** http://resolver.tudelft.nl/uuid:d79a48f4-de0b-45f0-b9ac-60e5455c85f3 (2019).

36. Sarghini, F., Felice, G. & Santini, S. Neural networks based subgrid scale modeling in large eddy simulations. *Computers and Fluids* **32.** https://doi.org/10.1016/S0045-7930(01)00098-6 (2003).

37. Shakib, F., Hughes, T. J. R. & Johan, Z. A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering* **89,** 141–219. https://doi.org/10.1016/0045-7825(91)90041-4 (1991).

38. Singh, A. P., Medida, S. & Duraisamy, K. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal* **55,** 2215–2227. https://doi.org/10.2514/1.J055595 (2017).

39. Smagorinsky, J. General circulation experiments with the primitive equations. *Monthly weather review,* 99–164. https://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2 (1963).

40. Sreenivasan, K. Turbulent Flow. *In Encyclopedia of science and technology,* 696–700. https://doi.org/10.1036/1097-8542.716800 (1991).

41. Stolz, S., Adams, N. A. & Kleiser, L. *The Approximate Deconvolution Model Applied to LES of Turbulent Channel Flow* in *Direct and Large-Eddy Simulation III* (eds Voke, P. R., Sandham, N. D. & Kleiser, L.) (Springer Netherlands, Dordrecht, 1999), 163–174. ISBN: 978-94-015-9285-7. https://doi.org/10.1007/978-94-015-9285-7_14.

42. Talstra, H. Large-scale turbulence structures in shallow separating flows. *TU Delft.* http://resolver.tudelft.nl/uuid:922e297b-19b9-4399-808d-23e837401a52 (2011).

43. Tracey, B., Duraisamy, K. & Alonso, J. J. *A machine learning strategy to assist turbulence model development* in *53rd AIAA Aerospace Sciences Meeting* (2015). https://doi.org/10.2514/6.2015-1287.

44. Vasilyev, O. V. & Lund, T. S. A general theory of discrete filtering for LES in complex geometry. *Annual Research Briefs,* 67–82. https://core.ac.uk/download/pdf/42767951.pdf#page=71 (1997).

45. Vollant, A., Balarac, G. & Corre, C. Subgrid-scale scalar flux modelling based on optimal estimation theory and machine-learning procedures. *Journal of Turbulence* **18,** 854–878. https://doi.org/10.1080/14685248.2017.1334907 (2017).

46. Vreman, B., Geurts, B. & Kuerten, H. A priori tests of large eddy simulation of the compressible plane mixing layer. *Journal of engineering mathematics,* 299–327. https://doi.org/10.1007/BF00042759 (1995).

47. Völker, S., Moser, R. D. & Venugopal, P. Optimal large eddy simulation of turbulent channel flow based on direct numerical simulation statistical data. *Physics of Fluids* **14,** 3675–3691. https://doi.org/10.1063/1.1503803 (2002).

48. Wilcox, D. *Turbulence Modeling for CFD Turbulence Modeling for CFD* **v. 1.** ISBN: 9781928729082. https://books.google.nl/books?id=tFNNPgAACAAJ (DCW Industries, 1998).

49. Yuan, Z., Wang, Y., Xie, C. & Wang, J. Deconvolutional artificial-neural-network framework for subfilter-scale models of compressible turbulence. *Acta Mechanica Sinica/Lixue Xuebao.* https://doi.org/10.1007/s10409-021-01150-7 (2022).

50. Zhiyin, Y. Large-eddy simulation: Past, present and the future. Chinese Journal of Aeronautics. *Annals of Nuclear Energy,* 11–24. https://doi.org/10.1016/j.cja.2014.12.007 (2015).