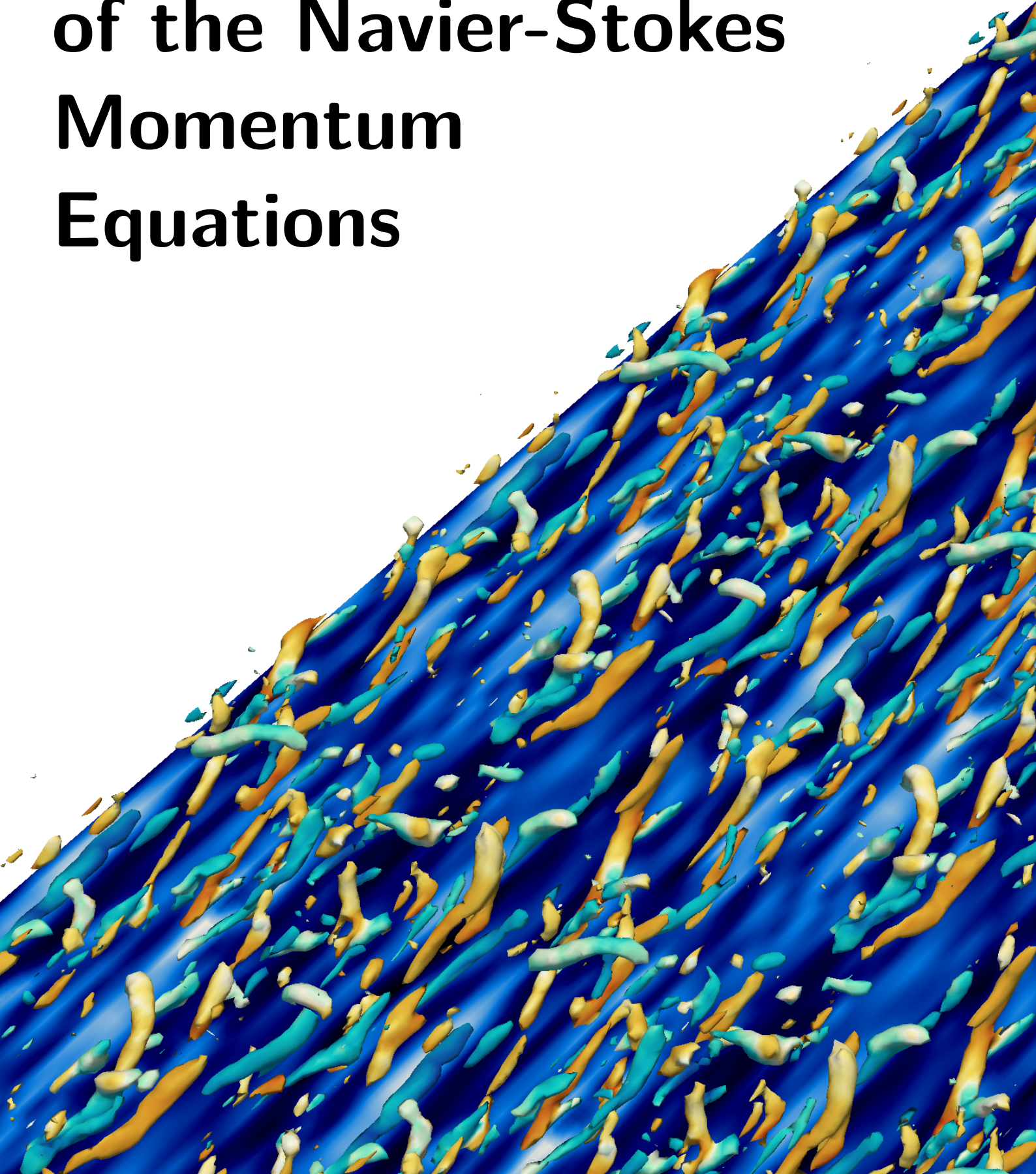


Data-Driven Closure Modelling of the Navier-Stokes Momentum Equations



Data-Driven Closure Modelling of the Navier-Stokes Momentum Equations

Master of Science Thesis

by

Oleksandr Krochak

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 30th September 2024.

Student number 5015529
Supervisor Dr. S.J. Hulshoff, TU Delft
Project Duration: February, 2024 - September, 2024



Delft University of Technology

Contents

List of symbols	xi
Acknowledgements	xv
Abstract	xvii
Summary	xix
1 Introduction	1
2 Methodology	7
2.1 Incompressible Navier-Stokes equations and problem definition	7
2.2 Galerkin formulation	8
2.3 Scale separation	9
2.4 Unresolved-scale Green’s function	11
2.5 Space and time discretization	12
2.6 Resolved-scale system	13
2.7 Data-driven modelling of the closure terms	14
2.8 Modeling workflow	15
2.9 Solver validation and new closure term definition	17
3 Stabilization strategies	21
3.1 Effect of H^1 projector on energy spectrum	21
3.2 Strategy 1: Filtering the ANN target	22
3.3 Strategy 2: Mini-batch sum loss	24
3.4 Strategy 3: Rotational data augmentation	25
4 Neural network closure model design	27
4.1 Preliminary network design	27
4.2 Mini-batch sum loss implementation	31
4.3 Sensitivity of model accuracy to model architecture	35
4.4 Sensitivity of model accuracy to input features	35
4.5 Sensitivity of model accuracy to the feature stencil	36
4.6 Chosen Network Designs	37
5 Application of data-driven closure to the complete Navier-Stokes equations	39

6	Evaluation of stabilization techniques and discretization effects	45
6.1	Effect of stabilization techniques	46
6.2	Effect of mesh refinement, CFL number and input stencil	48
7	Results	51
7.1	Instantaneous kinetic flow energy evolution	51
7.2	Flow evolution for the model M_{16}	53
7.3	Flow evolution for the model $M_{16}F$	56
7.4	Flow evolution for the model M_{32}	58
8	Conclusions	63
	Bibliography	67

List of Figures

2.1	Turbulent Channel Flow Geometry. White planes denote periodic boundary conditions, gray planes denote wall boundaries.	8
2.2	Scale decomposition and discretization.	10
2.3	Unresolved-scale Green's function for based on the H^1 seminorm projector (left) and L_2 projector (right) for a two-dimensional advection–diffusion problem. The global Péclet number is $Pe = 1000$. Adapted from [3], figs. 27-28.	12
2.4	Schematic of a multilayer perceptron neural network (Own Work). Black arrows denote the forward pass, blue arrows denote the backward pass of the training procedure.	16
2.5	Methodology flow chart.	17
2.6	Divergence of velocity field obtained from (interpolated) DNS, computed with the ParaView "Divergence" filter.	18
2.7	Root-mean-square error of the streamwise velocity obtained directly from DNS projection and indirectly from the perfect closure terms.	18
3.1	One-dimensional energy spectra from the DNS solution (black) and projections onto LES 32^3 mesh, \mathcal{P}_0 - nodal, $\mathcal{P}_1 - H_0^1$ projection.	22
3.2	Filtered and unfiltered LES streamwise energy spectra.	24
3.3	Rotational Data Augmentation visualized.	25
4.1	Neural network pipeline.	28
4.2	Three stencil options - default stencil 1, expanded stencil 2 and single element stencil 3. The labels (predictions) are closures associated to shape functions ϕ_i of the element highlighted by blue.	28
4.3	Bulk history of DNS flow. Validation data (48 snapshots), training data for 32^3 case (360 snapshots) and for 16^3 case (480 snapshots) highlighted.	30
4.4	Loss evolution (MSE loss in green, mini-batch sum loss in red).	32
4.5	Loss evolution, with model subject only to MSE loss.	32
4.6	Loss evolution, with model re-trained with mini-batch sum loss.	32
4.7	True closures from validation (unseen) data against ANN predictions of validation features, for a model subject to only MSE loss. Red line shows the perfect line of best fit, yellow line shows the actual line of best fit. Expected values (averages) of true and predicted closure terms in top left corners, Pearson's correlation coefficient in bottom right corners.	33

4.8	True closures from validation (unseen) data against ANN predictions of validation features, for a model re-trained with mini-batch sum loss. Red line shows the perfect line of best fit, yellow line shows the actual line of best fit. Expected values (averages) of true and predicted closure terms in top left corners, Pearson's correlation coefficient in bottom right corners.	34
5.1	Corrector pass convergence plot, using continuity closure provided by ANN model. .	40
5.2	Corrector pass convergence plot, using exact continuity closure with ML momentum closure terms.	40
5.3	Corrector pass convergence plot, using exact closure terms.	41
5.4	Pressure oscillations captured for model M1 ₁₆ (ML momentum closure, exact continuity closure) on the left, compared to projected DNS solution on the right. First LES time step.	41
6.1	Corrector pass convergence plot, constraining pressure to projected DNS pressure .	45
6.2	A posteriori results for variations of model M1 ₁₆ , with constrained pressure. Top: resolved specific kinetic energy. Middle: root mean square error (LES vs. PDNS) in resolved streamwise velocity. Bottom: correlation coefficient of the streamwise momentum closure prediction of the LES and PDNS.	47
6.3	A posteriori results for ANN models with varying stencil, CFL number, mesh refinement, with constrained pressure. Top: resolved specific kinetic energy. Middle: root mean square error (LES vs. PDNS) in resolved streamwise velocity. Bottom: correlation coefficient of the streamwise momentum closure prediction of the LES and PDNS. 48	48
7.1	Evolution of mean and turbulent kinetic energy for model M1 ₁₆ (red), compared against algebraic model (gray).	53
7.2	Stream wise resolved fluctuation \bar{u}'' energy spectra at varying wall distance at $t = 15\Delta t = 0.11$ for closure model M1 ₁₆ (red) and algebraic closure (gray), both with constrained pressure.	54
7.3	Mean flow and resolved components of Reynolds stresses at $t = 14\Delta t = 0.10$ for model M1 ₁₆ (red), compared against algebraic model (gray).	55
7.4	Stream wise resolved fluctuation \bar{u}'' energy spectra at varying wall distance at $t = 20\Delta t = 0.14$ for closure model M1 ₁₆ (red) and M1 ₁₆ F (yellow) both with constrained pressure.	56
7.5	Evolution of mean and turbulent kinetic energy for model M1 ₁₆ (red), compared against filtered version of the model (orange).	57
7.6	Evolution of mean and turbulent kinetic energy for model M1 ₃₂ (purple), compared against algebraic model (gray).	58

7.7	Stream wise resolved fluctuation \bar{u}' energy spectra at varying wall distance at $t = 3\Delta t$ for closure model M1 ₃₂ (purple) and algebraic closure (gray), both with constrained pressure. High wave number region zoomed-in on the right, with algebraic spectrum omitted for visibility.	59
7.8	Stream wise resolved fluctuation \bar{u}' energy spectra at varying wall distance at $t = 10\Delta t = 0.03$ for closure model M1 ₃₂ (purple) and algebraic closure (gray), both with constrained pressure.	59
7.9	Mean flow and components of resolved Reynolds stress at $t = 14\Delta t = 0.05$ for model M1 ₃₂ (purple), compared against algebraic model (gray).	60

List of Tables

3.1	Rotational data augmentations applied.	25
4.1	Effect of model architecture on a priori correlations.	35
4.2	Effect of input features on a priori correlations.	36
4.3	Effect of stencil width on a priori correlations. For space stencil, right is upstream, up/down is spanwise. For time stencil, right is in the past.	36
4.4	Selected Network Designs.	37

List of symbols

Abbreviations

Abbreviation	Definition
ANN	Artificial neural network
CFD	Computational fluid dynamics
CPI	Corrector-pass instability
CPU	Central processing unit
DNS	Direct numerical simulation
GPU	Graphics processing unit
LES	Large-eddy simulation
LFS	Lagged feature set
LTI	Long-term instability
MLP	Multilayer Perceptron
PDNS	Projected direct numerical simulation
RANS	Reynolds-averaged Navier-Stokes
RMS	Root mean squared
SGS	Sub-grid scale
TCF	Turbulent channel flow
VMS	Variational multiscale

Dimensionless numbers

Symbol	Description	Definition
Re_τ	Friction Reynolds number	$\frac{u_\tau \delta}{\nu} \equiv \frac{1}{\nu}$
CFL	Courant-Friedrichs-Lewy number	$\frac{u\Delta t}{\Delta x} + \frac{v\Delta t}{\Delta y} + \frac{w\Delta t}{\Delta z}$

Symbols

Symbol	Definition	Dimensions
x, y, z	Streamwise, wall-normal and spanwise coordinates	L
\mathbf{x}	Coordinate vector	L
\mathbf{u}	Exact velocity field	LT^{-1}
p	Exact kinematic pressure field	L^2T^{-2}
\mathbf{f}	Forcing term	LT^{-2}
Ω	Space domain	L^3
$\partial\Omega$	Boundary of the space domain	L^2
Q	Time-space domain	L^3T
Q_n	Time-space slab	L^3T
∂Q	Boundary of time-space slab domain	L^2T
\mathcal{L}_M	Navier-Stokes momentum differential operator	–
\mathcal{L}_C	Navier-Stokes continuity differential operator	–
\mathbb{L}	Linear form	–
\mathbb{B}_1	Bilinear form	–
\mathbb{B}_2	Trilinear form	–
\mathbf{w}	Momentum weighting function vector	LT^{-2}
q	Continuity weighting function scalar	T^{-1}
\mathbf{U}	Solution (velocity and pressure) vector	–
\mathbf{W}	Weighting function (momentum and continuity) vector	–
\mathcal{V}_s	Space of approximation functions	–
$\overline{\mathcal{V}}_s$	Coarse-scale solution space	–
\mathcal{V}'_s	Fine-scale solution space	–
$\bar{\mathbf{u}}$	Resolved (coarse-scale) velocity field	LT^{-1}
\bar{p}	Resolved (coarse-scale) pressure field	L^2T^{-2}
$\bar{\mathbf{U}}$	Resolved (coarse-scale) solution vector	–
$\tilde{\mathbf{U}}$	Coarse-scale solution in frequency-domain	–
$\tilde{\mathbf{U}}_f$	Filtered coarse-scale solution in frequency-domain	–
$\bar{\mathbf{U}}_f$	Filtered coarse-scale solution in physical domain	–
$\bar{\mathbf{W}}$	Resolved (coarse-scale) weighting function vector	–
$\bar{\mathbf{w}}$	Resolved momentum weighting function	LT^{-2}
\bar{q}	Resolved continuity weighting function	T^{-1}
\mathbf{u}'	Unresolved (fine-scale) velocity field	LT^{-1}
p'	Unresolved (fine-scale) pressure field	L^2T^{-2}
\mathbf{U}'	Unresolved (fine-scale) solution vector	–
\mathbf{W}'	Unresolved (fine-scale) weighting function vector	–
$N \cdot, \cdot $	Norm	–

Symbol	Definition	Dimensions
iden	Identity operator	–
$\text{proj}_N^{\overline{V}_s}$	Projection onto coarse-scale space	–
$\text{Res}_s(\overline{\mathbf{U}})$	Strong coarse-scale Navier-Stokes residual	LT^{-2}
g'	Fine-scale Green's function	–
g'_e	Element fine-scale Green's function	–
H^1	H^1 seminorm	–
L_2	L_2 norm	–
f'	Fine-scale component functional	–
\mathcal{F}'	Navier Stokes contribution fine-scale functional	–
Λ	Hat function	–
ϕ_i	i -th element shape function	–
\mathbf{a}_i	Solution coefficient vector	–
$\mathcal{L}_{\mathcal{M}'}$	Unresolved Navier-Stokes momentum operator	–
h_i	Neural network i -th input to a given layer	–
w_{ij}	Neural network weights matrix	–
b_i	Neural network bias vector	–
$f_{\text{activation}}$	Neural network activation function	–
\mathbf{y}	Neural network prediction vector	–
$\tilde{\mathbf{y}}$	Neural network true output vector	–
L	Neural network loss function	–
\mathbf{e}	Neural network error vector	–
N_{DNS}	Number of elements in DNS mesh in a given coordinate direction	–
N_{LES}	Number of elements in DNS mesh in a given coordinate direction	–
$\overline{\varphi}_q$	Resolved q -th element momentum shape function	LT^{-2}
$\overline{\psi}_q$	Resolved q -th element continuity shape function	T^{-1}
E_{uu}	Streamwise turbulent energy spectrum	L^2T^{-2}
κ_x	Streamwise wavenumber	L^{-1}
$\boldsymbol{\kappa}$	Wavenumber vector	L^{-1}
$\boldsymbol{\kappa}_{\text{Nyq}}$	Nyquist wavenumber vector	L^{-1}
$\mathcal{F}(\cdot)$	Fourier transform	–
$\mathcal{F}^{-1}(\cdot)$	Inverse Fourier transform	–
$w(\boldsymbol{\kappa})$	Filter scaling factor for wavenumber $\boldsymbol{\kappa}$	–
RMS	Root mean squared error	–
ρ	Correlation coefficient	–
k	Specific total instantaneous flow kinetic energy	$\text{L}^{-1}\text{T}^{-2}$
e	Specific total mean flow kinetic energy	$\text{L}^{-1}\text{T}^{-2}$

Symbol	Definition	Dimensions
\bar{R}_{ij}	Resolved Reynolds stress tensor	$L^{-1}T^{-2}$
$\text{ave}_{x,z}$	Homogeneous averaging operator	–
$\bar{\mathbf{u}}''$	Resolved LES velocity fluctuation	LT^{-1}
U_{bulk}	Bulk streamwise flow velocity	LT^{-1}
y^+	Non-dimensional wall distance	–

Acknowledgements

This thesis is the culmination of my five-year journey at the Delft University of Technology to become an aerospace engineer, and I would like to extend my deepest gratitude to people who have supported me along the way.

Firstly, I would like to thank my family, both in the Netherlands and in Ukraine, for their support and encouragement. In particular, I owe a deep bow to my mother Natalia, who always gave me profound advice in difficult situations and was always there for me. To Elinor, I thank you for your unwavering support and love in the last two years. And to my friend Francisco, with whom I've shared many afternoons in the High-Speed Laboratory, I wish you the best of luck in the future, and thank you for your company. To Andrea and Niklas, I thank you for being my amicable and reliable colleagues, and wish you the best in your academic careers. Ultimately, I would like to thank Dr. Hulshoff, my supervisor, for the fruitful discussions we've had, as well as for his readiness to address any problems I've faced and for making this challenging project possible.

Alex Krochak, September 2024

Abstract

An approach to data-driven closure modelling in the framework of variational multiscale method for large eddy simulation is presented. A turbulent channel flow at the friction Reynolds number of 180 is used as a case study. Challenges in the modelling linked to the continuity closure terms force the scope of the study to be narrowed to only momentum Navier-Stokes equations.

By using integrated forms of the resolved flow solution to train a multi-layer perceptron closure model, high a priori correlations with true closure terms can be achieved. Validation of the data-driven closure models a posteriori shows that the models are suitable for the computation of the velocity field, but fail to obtain a physical pressure solution. Accumulation of the model error leads to a decay of the kinetic energy of the mean flow, while the resolved turbulent fluctuations become excessively energized. Novel techniques such as data augmentation, mini-batch sum loss and filtering of the closure terms are presented and evaluated via large eddy simulations.

Summary

In this thesis, an approach to data-driven closure modelling in the framework of variational multiscale method for large eddy simulation is presented. Using a turbulent channel flow at $Re_\tau = 180$ as a case study, projections of flows obtained by direct numerical simulations are used as training data for a multi-layer perceptron closure model. By using integrated forms of Navier-Stokes components of the resolved flow, high a priori correlations with exact closure terms can be achieved through a practically viable neural network prediction. Chapter 1 presents the state-of-the-art for current attempts at data-driven modelling of closure terms for multiscale simulations of fluid flows. In chapter 2, a detailed description of variational multiscale method in fluid dynamics context is given, as well as preliminary discussion on multi-layer perceptron *modus operandi*. Afterward, in chapter 3, a few stabilization methods are proposed, that can possibly alleviate stability issues present in large-eddy simulations with data-driven closure. A detailed description of data processing and neural network training is provided in chapter 4, as well as additional considerations for data-driven modelling motivated by the complexity of fluid flow simulation. In chapter 5, the first application of data-driven closure modelling to LES simulation is presented, together with the issues in corrector pass loop convergence due to the continuity closure predicted by a machine learning model. Because of these issues, the scope of the problem under investigation was limited to the three-dimensional Navier-Stokes momentum equations, with the pressure solution obtained directly from the DNS projection. In chapter 6, a selection of simulations with data-driven closure is presented, together with a brief discussion of their stability and accuracy. Next, in chapter 7 a more thorough examination of flow stability and kinetic energy evolution is conducted for the selected closure models. Conclusions and answers to the research questions of this thesis, as well as recommendations for future work, are provided in the ultimate chapter 8.

1

Introduction

In this thesis, a non-standard approach to the idea of Large Eddy Simulation (LES) is taken - instead of conventional filtering techniques, explicit projection is defined to separate the resolved and non-resolved scales. This technique is called the variational multiscale method (VMS method), and was initially developed to gain further insight into stable computational approaches to general multiscale phenomena [1], i.e., physical phenomena where a variety of relevant physical scales are present. Later on, this approach was tailored specifically to LES of turbulent flows, where a separation is made between large resolved scales and small modelled scales of turbulent fluid motions [2]. From the 2000s to present time the method has gained maturity, and a comprehensive review of the VMS method and related mathematical concepts can be found in a work by Hughes et al. [3]. There are several reasons to consider VMS method instead of traditional LES. First and foremost, the former avoids the filtering-differentiation commutativity issue, which is described in depth in the contribution of Dunca et al [4]. Secondly, LES applied to wall-bounded flows necessitate *ad hoc* fixes such as Van Driest damping for non-dynamic eddy viscosity SGS models. Lastly, most SGS stress models do not allow for backscatter. The first applications of the VMS method to wall-bounded flows show increased accuracy compared to consistent LES application [2]. It should be noted that the first VMS simulations of turbulent flows employed primitive sub-grid scale (SGS) stress models, and it is expected that leveraging more sophisticated models would further augment the accuracy of the method.

The variational multiscale method builds upon the finite element method, which was originally developed to solve structural mechanics and elasticity problems in civil and aerospace engineering. Its mathematical approach makes it an optimal tool for such problems since the underlying physical equations for structural problems are often *linear* and *self-adjoint*. In such a case, the Galerkin procedure of subdivision of continuous domain into *finite elements* and converting the strong form of the partial differential equation into weak form is sufficient to arrive at an optimal solution. However, for fluid dynamics problems, the convective term $\nabla \cdot (\mathbf{u} \otimes \mathbf{u})$ is *non-linear*,

and straight-forward application of the Galerkin procedure results in an unstable solution [5]. For these reasons, the popularity of finite element method in structural mechanics did not spur fluid dynamics practitioners to adopt it. Instead, historically finite difference methods were used, and later on these were replaced by finite-volume formulations. Finite-volume method has an advantage of satisfying conservation laws over the discrete domain [6], which is an advantage in regard to the physical fidelity of the model. Finite-difference methods were the first attempt at continuous problems of fluid dynamics.

An important consequence of these historical developments is that closure models developed for the simulation of turbulent flows were developed and applied without paying much regard to the method of discretization. The Smagorinsky model, which is nowadays implemented in every finite-volume LES solver, was originally written for a finite-difference model [7]. In contrast, the variational multiscale framework allows for the unification of the discretization and sub-grid model errors into a single VMS model error, which is *only* dependent on the closure term model. This allows for more accurate and physical simulations of the turbulent flows. It is worth noting that the idea of variational multiscale method for turbulence modelling has some similarities with *implicit* LES, where discretization is carefully chosen in such a way that truncation error acts as sub-grid scale model [8]. At the same time, traditional closure models applied to large-eddy or Reynolds-averaged formulations of Navier-Stokes are known to be dissipative and improve the stability of the simulation at the cost of its accuracy [9, 10]. Thus, if one wishes to develop a new closure model for VMS formulations of Navier-Stokes equations, leveraging the orderly mathematical formulation for increase in accuracy, difficulties in *online* simulation stability are expected.

It is worth noting that this problem of stability is not unique to the VMS methodology. In fact, most researchers attempting to use neural networks as a means of RANS or LES closure modelling have to deal with stability and divergence of their simulations sooner or later. Vollant et al. [11] considered a problem of passive scalar mixing in a plane jet and found that a multi-objective optimization approach can lead to increased stability when compared to only ANN SGS closure estimation. Beck et al. [12] encounters high-frequency error accumulation with ANN closure for isotropic turbulence.

The precise mathematical framework of the VMS method shows that the unresolved component of the solution is driven by a *functional* of the strong residual of the resolved solution and the resolved solution itself [13]. The dependence of fine-scale error on the coarse-scale residual is governed by a fine-scale Green's function [1]. While this has been calculated analytically for a number of simple problems [14], exact evaluation for the Navier-Stokes equations is not available.

This functional is thus responsible for both truncation and closure errors, and finding a successful model for it can allow for significant gains in simulation fidelity. The working hypothesis is that multi-layer perceptron [15] (MLP) neural network is a suitable tool for the approximation of such a functional. Although MLP's and other types of neural networks have gained immense popularity in data processing, there is limited experience in applying them to numerical modelling of physical phenomena. One important difference is that typically the output of a neural network is not "recycled" back into it. For example, if a neural network is used to recognize text or categorize

images, it is only evaluated once and is not affected by an error in its prediction. However, if an ANN SGS model is inserted into an LES solver, ANN-induced errors will cause the solution vector to deviate. If inside the solution loop these errors accumulate, this will quickly lead to simulation divergence and crash. Thus, a key desirable property of such ANN closure model is that it must *dampen the errors present in the solution that were induced by an imperfect ANN prediction of the closure terms*.

Initial investigations of ANN SGS models for the VMS formulation were performed on the model problem of one-dimensional turbulence governed by forced Burgers equation [16, 17, 18, 19]. A brief overview of the previous work at Delft University of Technology is presented here. Robijns [18] explored the application of data-driven closures to a variational multiscale framework by considering a one-dimensional Burgers equation, with either constant or sinusoidal forcing applied. It was found that MLP ANN can adequately describe the functional relationship between large-scale input features and fine-scale closure terms. Janssens [16] built upon work of Robijns by applying it to a more complex problem of atmospheric turbulence and general circulation models, still in an one-dimensional setting but with a forcing term obtained directly from large-eddy simulations of convective boundary layers. Although this confirmed the model's success in *offline* evaluations, two key instabilities were identified that destabilizes the simulation in an *online* setting. The first one is the so corrector-pass instability (CPI), which manifests during the iterative minimization of the coarse scale's weak residual in the Newton root-finding procedure. The introduction of ANN leads to additional spurious roots in the solution space that tarnish the convergence of the Newton loop and return non-physical local minima. The second, long-time, instability (LTI) arises from the poor ability of ANN to generalize - in an online setting, once model-induced errors begin to accumulate, the ANN is not able to cope with them and the solution vector diverges. Both these instabilities need to be resolved for the model success in online simulations. Pusuluri [19] explored potential solutions by augmenting the training data set with white or Gaussian noise. An increase in stability was found, at the cost of the closure model accuracy, and Gaussian noise was found to perform better than white noise augmentation. Rajampeta [17] further explored other stabilization strategies. In order to address CPI, a lagged feature set (LFS) strategy was proposed, in which the feature set for the ANN is evaluated from the previous time-step, effectively decoupling the ANN evaluation and corrector pass loop within the time-step. It is presumed that this negatively affects the ANN accuracy, as there is a lag introduced between the flow solution used to obtain the ANN features and the flow solution being updated within the corrector pass loop. This effect would be more profound at high Courant-Friedrichs-Lewy numbers. Another strategy that was proposed is backscatter limiting, which effectively disables backscatter from the fine scale to coarse scales. This had a positive effect on stability but eliminates one of the advantages of the VMS formulation. Successful large-eddy simulations of one-dimensional turbulent flows were obtained with backscatter limiting and lagged feature set strategies applied to ANN closure.

Bettini [20] was the first to build upon the work of aforementioned authors and apply it to a far more ambitious problem of modelling wall-bounded three-dimensional turbulence in a channel flow. In addition to using a deeper network architecture, they also utilized the LFS strategy. Bet-

tini used distinct networks depending on the element shape function used to weigh the solution. By implementing additional loss functions for force-free weak-residual error, a solver adherence penalty, and an energy transfer penalty, high correlations ($\rho > 0.90$) were achieved for the momentum and continuity closure terms. However, a large model architecture was adopted, with 8 varying MLP models, one for each element shape function. Each MLP model had 5 hidden layers with 600 neurons. Additional computational cost was introduced by using a large spatial features stencil, consisting of 5 elements for momentum and 8 for continuity. For each element, coarse-scale features over the element had to be integrated to pass them forward as model input.

It was discovered by Bettini that closure terms introduced by the ANN can strongly affect the high-frequency components of the solution. To address these possible instabilities, Bettini has developed an energy-biased training approach. It uses an augmented Lagrangian method to transform a constrained objective function into an unconstrained problem that can be readily tackled with the current training methodology. The objective function penalizes the ANN model when it predicts backscatter or dissipation values that exceeded the truth values obtained from projected DNS. This constraint was implemented on element level, which is significantly more restrictive than the global constraint, as now all the weak-forms $(\cdot, \cdot)_{E_n}$ for all the elements and shape functions have to satisfy the constraint, rather their net global sum $(\cdot, \cdot)_{\Omega}$. Additionally, a H^1 projector was used to map a DNS solution to an LES solution, as in this thesis. However, the efficacy of such an approach is not clear as no verified a posteriori evaluation has been carried out.

In this thesis, the primary subject of investigation will be the Navier-Stokes momentum equations, with the continuity equation omitted. This is because continuity closure has been discovered to be particularly challenging to model with the data-driven methods used in this study, and is explained in more detail in chapter 5.

The main objective of this thesis is to develop effective strategies for long-term stabilization of VMS simulations of incompressible Navier-Stokes momentum equations with data-driven closure, as well as reducing the computational cost associated with training and executing of MLP closure models. This leads to the following research objective:

Research Objective

To address the long-term instabilities and improve the practicality of neural network fine-scale (i.e., sub-grid scale) model in variational multiscale framework for incompressible Navier-Stokes momentum equations, through application to turbulent channel flow.

The following research questions have been formulated to achieve the objective:

- **RQ-1** - How is the stability of the variational multiscale flow solution affected by ANN closure?
 - **RQ-1a** How does the flow deviate from projected DNS solution when subject to ANN closure?
 - **RQ-1b** Can long-term instabilities (LTI's) be adequately addressed through additional filtering of the DNS solution during *offline* ANN training?

- **RQ-1c** Can LTI's be addressed through introduction of new accuracy metrics during *offline* network training?
- **RQ-2** - How can the training and run-time computational cost of the neural network be reduced?
 - **RQ-2a** What ANN features can be discarded without compromising accuracy?
 - **RQ-2b** Can the feature set stencil be compacted?
 - **RQ-2c** Can the training routine be optimized?
- **RQ-3** - How does the ANN model respond to numerical refinement?
 - **RQ-3a** What is the effect of mesh size on model accuracy and stability?
 - **RQ-3b** What is the effect of CFL number on model accuracy and stability?
- **RQ-4** - What is the effect of data augmentation on network accuracy and stability?

2

Methodology

In this chapter, the variational multiscale method and its formulation of scale separation are described. Specifically, section 2.1 describes the governing Navier-Stokes equations for an incompressible flow. Section 2.2 describes the Galerkin method formulation, while section 2.3 introduces the scale separation and LES approach. Next, section 2.4 explains the role of unresolved-scale Green's function, while section 2.5 describes the finite-element spatial and temporal discretization. Section 2.6 introduces the resolved-scale system in its final form. Section 2.7 provides some preliminaries and a brief introduction to data-driven modelling. Section 2.8 provides a high-level overview of the ANN closure model data curation, training and validation, and lastly section 2.9 describes the validation of the LES solver used and an alternative definition of the closure terms.

2.1. Incompressible Navier-Stokes equations and problem definition

The laws of motion for a continuum Newtonian fluid at sufficiently small Mach numbers can be described by the incompressible Navier-Stokes equations, viz:

$$\nabla \cdot \mathbf{u} = 0, \quad \mathbf{x} \in \Omega \quad (2.1)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla p - \nu \Delta \mathbf{u} = \mathbf{f} \quad \mathbf{x} \in \Omega \quad (2.2)$$

where $\mathbf{u}(\mathbf{x}, t) = [u \ v \ w]^T$ is the velocity vector, with $\mathbf{x}(x, y, z)$ being the coordinate vector and t the time coordinate, $p(\mathbf{x}, t)$ is the kinematic pressure (static pressure divided by fluid density), \otimes is the tensor product ($[u \otimes v]_{ij} = u_i v_j$), ν is constant positive kinematic viscosity, \mathbf{f} is the forcing term, and Δ is the Laplace operator. The streamwise direction and velocity are denoted by x , u , respectively, spanwise by z , w and wall-normal by y , v . Consider a channel domain $\Omega = L_x \times L_y \times L_z$, where friction velocity $u_\tau = \sqrt{\tau_{\text{wall}}/\rho}$ and channel half-height δ are chosen as characteristic velocity and length scales. The channel length is denoted by $L_x = 6.0\delta$, the span by $L_z = 4.0\delta$ and the height by $L_y = 2.0\delta$. From here onwards, the δ symbol is dropped for brevity in notation. The geometry is

shown in Figure 2.1. The spatial boundary $\partial\Omega$ consists of the 6 edge faces: two walls at $y = 0$ and $y = 2.0$, and two coupled pairs of periodic boundary conditions at $x = 0, x = 6.0$ and $z = 0, z = 4.0$.

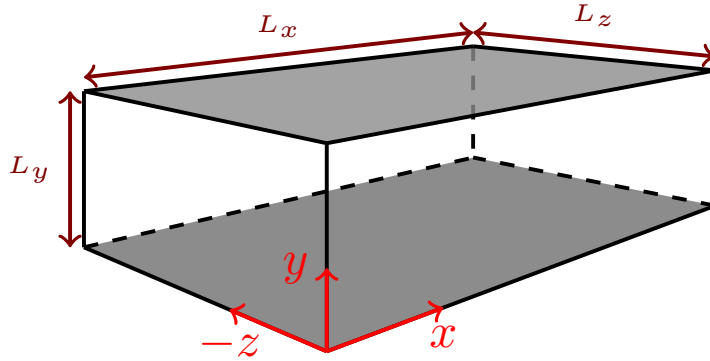


Figure 2.1: Turbulent Channel Flow Geometry. White planes denote periodic boundary conditions, gray planes denote wall boundaries.

Of interest is the time evolution of the flow variables in this domain, namely the solution vector $\mathbf{U} = [\mathbf{u}; p](\mathbf{x}, t)$.

2.2. Galerkin formulation

The continuous problem defined through equations 2.1 and 2.2 on Ω is cast into the integral formulation by following the Galerkin procedure as described in [21] and [13]. First, differential operators $\mathcal{L}_M, \mathcal{L}_C$ are introduced:

$$\mathcal{L}_M(\mathbf{u}, p) = \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla p - \nu \Delta \mathbf{u} \quad (2.3)$$

$$\mathcal{L}_C(\mathbf{u}) = \nabla \cdot \mathbf{u} \quad (2.4)$$

Then eqs. (2.1) and (2.2) can be written as:

$$\mathcal{L}_C(\mathbf{u}) = 0, \quad \mathbf{x} \in \Omega \quad (2.5)$$

$$\mathcal{L}_M(\mathbf{u}, p) - \mathbf{f} = \mathbf{0}, \quad \mathbf{x} \in \Omega \quad (2.6)$$

The space of solution functions is denoted by $\mathcal{V}_s(\Omega)$, and the space of weighting functions by $\mathcal{V}_w(\Omega)$. Please note that these spaces are infinite-dimensional, and the solution \mathbf{U} lives in the space \mathcal{V}_s , i.e., $\mathbf{U}(\mathbf{x}, t) \in \mathcal{V}_s$.

The objective of the continuous formulation is to find such $\mathbf{U} = [\mathbf{u}; p](\mathbf{x}, t)$ that the weak formulation of incompressible Navier Stokes is satisfied for all weighting (test) functions $\mathbf{W} = [\mathbf{w}; q] \in \mathcal{V}_w$, assuming an initial condition $\mathbf{U}_0 \in \Omega$ is provided:

$$\begin{aligned} (\mathbf{w}, \mathcal{L}_M(\mathbf{u}, p))_{\Omega} - (\mathbf{w}, \mathbf{f})_{\Omega} &= 0 \\ (q, \mathcal{L}_C(\mathbf{u}))_{\Omega} &= 0 \end{aligned} \quad (2.7)$$

Alternatively, equation 2.7 can be expressed in terms of the linear form \mathbb{L} , bilinear form \mathbb{B}_1 , trilinear form \mathbb{B}_2 , all defined as follows:

$$\begin{aligned}\mathbb{L}(\mathbf{W}) &= (\mathbf{w}, \mathbf{f})_{\Omega} \\ \mathbb{B}_1(\mathbf{W}, \mathbf{U}) &= \left(\mathbf{w}, \frac{\partial \mathbf{u}}{\partial t} + \nabla p - \nu \Delta \mathbf{u} \right)_{\Omega} + (q, \nabla \mathbf{u})_{\Omega} \\ \mathbb{B}_2(\mathbf{W}, \mathbf{U}, \mathbf{V}) &= (\mathbf{w}, (\mathbf{u} \cdot \nabla) \mathbf{v})_{\Omega} \\ \mathbb{B}_1(\mathbf{W}, \mathbf{U}) + \mathbb{B}_2(\mathbf{W}, \mathbf{U}, \mathbf{U}) - \mathbb{L}(\mathbf{W}) &= \mathbf{0}\end{aligned}\tag{2.8}$$

This formulation is used as it streamlines the decomposition of the solution space \mathcal{V}_s , which is the next item of discussion.

2.3. Scale separation

The key idea of the variational multiscale method lies in invoking the scale separation, or equivalently the separation of numerical approximation and error of the solution explicitly *ab initio*. The space of approximation functions $\mathcal{V}_s(\Omega)$ is separated into resolved-scale solution space $\overline{\mathcal{V}}_s$, which is finite-dimensional, and unresolved-scale solution space \mathcal{V}'_s , which is infinite-dimensional:

$$\mathcal{V}_s = \overline{\mathcal{V}}_s \oplus \mathcal{V}'_s\tag{2.9}$$

The resolved-scale solution space $\overline{\mathcal{V}}_s$ is defined through the discretization introduced in the finite element formulation. This space contains the numerically resolved resolved-scale solution $\overline{\mathbf{U}} = [\overline{\mathbf{u}}; \overline{p}]$, while the "error" in the solution $\mathbf{U}' = [\mathbf{u}'; p']$ lies in the unresolved-scale space \mathcal{V}'_s . The scale separation of $\overline{\mathbf{U}}$ and \mathbf{U}' is visualized for 1-dimensional scenario in fig. 2.2a. The separation of scales is also conducted for the weighting function space:

$$\mathcal{V}_w = \overline{\mathcal{V}}_w \oplus \mathcal{V}'_w,\tag{2.10}$$

with $\overline{\mathcal{V}}_w$ being the resolved-scale weighting function space, which is equivalent to $\overline{\mathcal{V}}_s$ in this thesis (as Bubnov-Galerkin approach is implemented), and \mathcal{V}'_w is the unresolved-scale weighting function space.

For an explicit definition of the resolved and unresolved solution a resolved-scale projector is defined, which finds a resolved solution $\overline{\mathbf{U}}$ as a minimum of a specified norm $N|\mathbf{U}, \overline{\mathbf{U}}|$:

$$\overline{\mathbf{U}} = \text{proj}_N^{\overline{\mathcal{V}}_s} \mathbf{U}\tag{2.11}$$

It should be noted that this projector also accounts for the spatial discretization of the resolved-scale solution $\overline{\mathbf{U}}$, as the basis functions used to reconstruct the solution live in space $\overline{\mathcal{V}}_s$. The choice

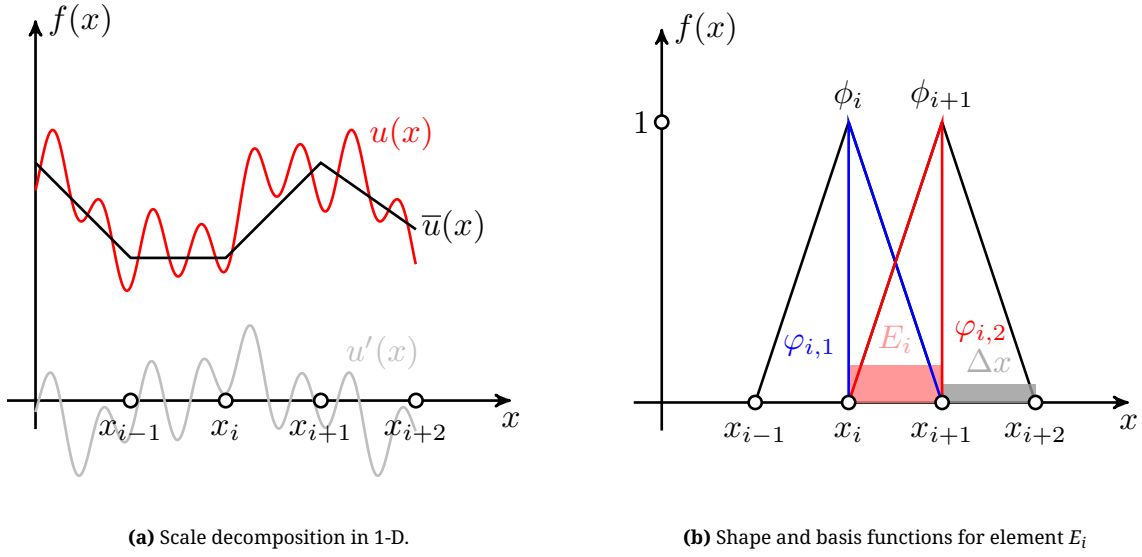


Figure 2.2: Scale decomposition and discretization.

of basis functions will be further explained in section 2.5.

$$\bar{\mathbf{W}} = \text{proj}_N^{\bar{\mathcal{V}}_s} \mathbf{W} \quad (2.12)$$

Resolved-scale projector also implicitly defines an unresolved-scale projector from the difference with the identity projector:

$$\begin{aligned} \mathbf{U}' &= \mathbf{U} - \bar{\mathbf{U}} = (\text{idem} - \text{proj}_N^{\bar{\mathcal{V}}_s}) \mathbf{U} \\ \mathbf{W}' &= \mathbf{W} - \bar{\mathbf{W}} = (\text{idem} - \text{proj}_N^{\bar{\mathcal{V}}_s}) \mathbf{W} \end{aligned}$$

Now that the discretization of the continuous problem has been introduced through the scale separation, separate resolved- and unresolved-scale equations can be defined from equation 2.8, through substitution of $\mathbf{U} = \bar{\mathbf{U}} + \mathbf{U}'$:

$$\mathbb{B}_1(\mathbf{W}, \bar{\mathbf{U}} + \mathbf{U}') + \mathbb{B}_2(\mathbf{W}, \bar{\mathbf{U}} + \mathbf{U}', \bar{\mathbf{U}} + \mathbf{U}') - \mathbb{L}(\mathbf{W}) = \mathbf{0} \quad (2.13)$$

Using the linearity of functional forms, eq. (2.13) is equivalent to:

$$\begin{aligned} &\mathbb{B}_1(\mathbf{W}, \bar{\mathbf{U}}) + \mathbb{B}_2(\mathbf{W}, \bar{\mathbf{U}}, \bar{\mathbf{U}}) - \mathbb{L}(\mathbf{W}) = \\ &- \mathbb{B}_1(\mathbf{W}, \mathbf{U}') - \mathbb{B}_2(\mathbf{W}, \bar{\mathbf{U}}, \mathbf{U}') - \mathbb{B}_2(\mathbf{W}, \mathbf{U}', \bar{\mathbf{U}}) - \mathbb{B}_2(\mathbf{W}, \mathbf{U}', \mathbf{U}') \end{aligned}$$

As this holds for any $\mathbf{W} \in \mathcal{V}_w$, separate systems of partial differential equations can be derived for resolved-scale quantities, with $\bar{\mathbf{W}} \in \bar{\mathcal{V}}_w \subset \mathcal{V}_w$, and for unresolved-scale quantities with $\mathbf{W}' \in \mathcal{V}'_w \subset \mathcal{V}_w$:

$$\mathbb{B}_1(\bar{\mathbf{W}}, \bar{\mathbf{U}} + \mathbf{U}') + \mathbb{B}_2(\bar{\mathbf{W}}, \bar{\mathbf{U}} + \mathbf{U}', \bar{\mathbf{U}} + \mathbf{U}') - \mathbb{L}(\bar{\mathbf{W}}) = \mathbf{0}$$

$$\mathbb{B}_1(\mathbf{W}', \bar{\mathbf{U}} + \mathbf{U}') + \mathbb{B}_2(\mathbf{W}', \bar{\mathbf{U}} + \mathbf{U}', \bar{\mathbf{U}} + \mathbf{U}') - \mathbb{L}(\mathbf{W}') = \mathbf{0}$$

Applying linearity and re-arranging:

$$\mathbb{B}_1(\bar{\mathbf{W}}, \bar{\mathbf{U}}) + \mathbb{B}_2(\bar{\mathbf{W}}, \bar{\mathbf{U}}, \bar{\mathbf{U}}) - \mathbb{L}(\bar{\mathbf{W}}) = -\mathbb{B}_1(\bar{\mathbf{W}}, \mathbf{U}') - \mathbb{B}_2(\bar{\mathbf{W}}, \bar{\mathbf{U}}, \mathbf{U}') - \mathbb{B}_2(\bar{\mathbf{W}}, \mathbf{U}', \bar{\mathbf{U}}) - \mathbb{B}_2(\bar{\mathbf{W}}, \mathbf{U}', \mathbf{U}') \quad (2.14)$$

$$\mathbb{B}_1(\mathbf{W}', \mathbf{U}') + \mathbb{B}_2(\mathbf{W}', \mathbf{U}', \mathbf{U}') - \mathbb{L}(\mathbf{W}') = -\mathbb{B}_1(\mathbf{W}', \bar{\mathbf{U}}) - \mathbb{B}_2(\mathbf{W}', \bar{\mathbf{U}}, \mathbf{U}') - \mathbb{B}_2(\mathbf{W}', \mathbf{U}', \bar{\mathbf{U}}) - \mathbb{B}_2(\mathbf{W}', \bar{\mathbf{U}}, \bar{\mathbf{U}}) \quad (2.15)$$

Equation 2.14 is an exact equation for the resolved-scales, while equation 2.15 is an exact equation for the unresolved-scale component of the solution.

2.4. Unresolved-scale Green's function

Equation 2.15 can be rewritten to show that the unresolved-scale component is forced by its strong residual of the resolved solution, i.e., $\text{Res}_s(\bar{\mathbf{U}}) = \mathcal{L}_M(\bar{\mathbf{u}}, \bar{p}) + \mathcal{L}_C(\bar{\mathbf{u}}) - \mathbf{f}$:

$$\mathbb{B}_1(\mathbf{W}', \mathbf{U}') + \mathbb{B}_2(\mathbf{W}', \mathbf{U}', \mathbf{U}') + \mathbb{B}_2(\mathbf{W}', \bar{\mathbf{U}}, \mathbf{U}') + \mathbb{B}_2(\mathbf{W}', \mathbf{U}', \bar{\mathbf{U}}) = -(\mathbf{W}', \text{Res}_s(\bar{\mathbf{U}})) \quad (2.16)$$

This means that the unresolved-scale component is determined by a functional f' , $\mathbf{U}' = f'(\bar{\mathbf{U}}, \text{Res}_s(\bar{\mathbf{U}}))$. This functional is closely related to the idea of unresolved-scale Green's function g' , see eq. 65 in [3]:

$$\mathbf{U}' = - \int_Q g' \text{Res}_s(\bar{\mathbf{U}}) dQ \quad (2.17)$$

For a detailed overview of the derivation of the unresolved-scale Green's function, reader is referred to work by Hughes et al. [3]. In the given problem, it is important to realize that neither the functional f' nor the unresolved-scale Green's function g' can be determined exactly. Limited insights can be gained from analytically derived unresolved-scale Green's functions for simpler partial differential equations. Hughes and Sangalli [14] have observed that locality of the unresolved-scale Green's function (in other words, how *spread out* is the influence of the strong residual $\text{Res}_s(\bar{\mathbf{U}})$ on the unresolved-scale component \mathbf{U}') strongly depends on the projector used to define the resolved-scale component in eq. (2.11). They have found that L_2 norm projection leads to strongly non-localized unresolved-scale Green's function, while H^1 projector leads to a more localized g' , see figure 2.3. The definition of this H^1 projector is given in equation 2.19. In short, it considers the L_2 norm of both the solution and its spatial gradient. The unresolved-scale Green's function, which is unknown for Navier-Stokes equations, is only helpful in the current context to provide limited insight onto the spatial dependence of \mathbf{U}' due to the strong residual of the resolved solution. In this thesis, the H^1 projector is used, due to the improved locality of \mathbf{U}' observed for two-dimensional problems, which *might* also be the case for the Navier-Stokes equations.

It should be noted that only integrated forms of \mathbf{U}' directly contribute to the evolution of the

resolved-scale solution, not \mathbf{U}' directly. The precise definition of these integrals is introduced in section 2.6. The spatial dependence of \mathbf{U}' is only discussed here to motivate the choice of projector used to define the coarse-scale solution.

$$\|v\|_{L_2(\Omega)} = \left(\int_{\Omega} |v|^2 d\Omega \right)^{1/2} = (v, v)_{\Omega}^{1/2} \quad (2.18)$$

$$\|v\|_{H^1(\Omega)} = \left(\|v\|_{L^2(\Omega)}^2 + \|\nabla v\|_{L^2(\Omega)}^2 \right)^{1/2} \quad (2.19)$$

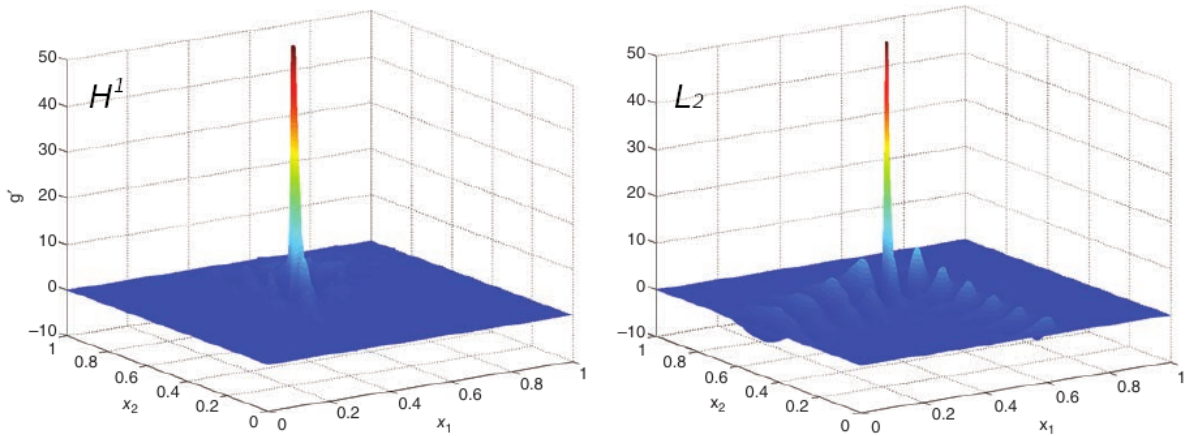


Figure 2.3: Unresolved-scale Green's function for based on the H^1 seminorm projector (left) and L_2 projector (right) for a two-dimensional advection–diffusion problem. The global Péclet number is $Pe = 1000$. Adapted from [3], figs. 27-28.

2.5. Space and time discretization

Further steps need to be taken to reduce the equation 2.14 into a fully discrete system. The finite element method primarily consists of separating the continuous problem domain into a multitude of discrete finite elements, over which the solution is defined as a product of a scalar coefficient and a shape/basis function. While some VMS applications use a time-space finite-element method, in the current work, a semi-discrete approach is taken, where the finite-element method is only employed for the spatial discretization. For the time discretization of the resolved-scale equation, the time derivative term $\partial \bar{\mathbf{u}} / \partial t$ is computed using the backward finite-difference second-order accurate scheme, viz:

$$\left| \frac{\partial \bar{\mathbf{u}}}{\partial t} \right|_n = \frac{1}{2\Delta t} |3\bar{\mathbf{u}}_n - 4\bar{\mathbf{u}}_{n-1} + \bar{\mathbf{u}}_{n-2}| + \mathcal{O}(\Delta t^2), \quad (2.20)$$

where $\bar{\mathbf{u}}_n$ is the discretized resolved-scale solution of the velocity field at time $t = n\Delta t$. Additionally, the error term $\mathcal{O}(\Delta t^2)$ is also part of the closure term and is thus evaluated implicitly by the closure terms of the resolved system. It should be noted that the pressure field does not require a temporal discretization.

The spatial discretization, which is needed for the complete resolved solution $\bar{\mathbf{U}}$ (both pressure and velocity fields) is introduced as follows:

$$\bar{\mathbf{U}}_{t=t_n} = \sum_{i=1}^{N^3_{\text{elem}}} \mathbf{a}_i(t, \mathbf{x}_i) \phi_i(\mathbf{x}, \mathbf{x}_i), \quad (2.21)$$

with $\phi_i(\mathbf{x}, \mathbf{x}_i)$ being the basis function centered at \mathbf{x}_i , the node coordinates of element i . Trilinear spatial discretization is used, thus the *basis function* is defined as follows:

$$\phi_i(\mathbf{x}, \mathbf{x}_i) = \Lambda\left(\frac{x-x_i}{\Delta x}\right) \Lambda\left(\frac{y-y_i}{\Delta y}\right) \Lambda\left(\frac{z-z_i}{\Delta z}\right), \quad (2.22)$$

where $\Lambda(x) = \max(1 - |x|, 0)$ is a regular hat function, and Δx , Δy , Δz are the constant element width in x -, y - and z -directions, as a homogeneous LES mesh is used. The discretization can alternatively be defined through a combination of *element shape functions*, which do not span multiple elements but are contained within a single mesh element. This is advantageous during the assembly process of the discrete system. Considering a 1D case with only $\Lambda\left(\frac{x-x_i}{\Delta x}\right)$, both basis functions (ϕ_i, ϕ_{i+1}) and element shape functions ($\varphi_{i,1}, \varphi_{i,2}$) are shown in fig. 2.2b. For a three-dimensional case with homogeneous cuboid mesh elements, there will be 8 element shape functions per element, in total $8 \cdot N^3$ element shape functions for the whole mesh (assuming an equal number N of elements in x , y and z directions).

2.6. Resolved-scale system

As was mentioned previously in section 2.4, the LES closure problem in the VMS framework comprises the approximation of quantities that drive the resolved-scale equation 2.14, which is rewritten here with operator notation. Importantly, one is not interested in \mathbf{U}' itself, but rather the integrated forms of the unresolved-scale solution, which force the resolved solution $\bar{\mathbf{U}}$. These spatial integrals of \mathbf{U}' are denoted by \mathcal{F}' , and are defined as follows:

$$\mathcal{F}' = \mathcal{F}'(\bar{\mathbf{U}}, \text{Res}_s(\bar{\mathbf{U}})) = -\mathbb{B}_1(\bar{\mathbf{W}}, \mathbf{U}') - \underbrace{\mathbb{B}_2(\bar{\mathbf{W}}, \bar{\mathbf{U}}, \mathbf{U}') + \mathbb{B}_2(\bar{\mathbf{W}}, \mathbf{U}', \bar{\mathbf{U}})}_{\text{Cross stress}} - \underbrace{\mathbb{B}_2(\bar{\mathbf{W}}, \mathbf{U}', \mathbf{U}')}_{\text{Reynolds stress}} \quad (2.23)$$

Then the resolved-scale equation 2.14 can be rewritten as follows:

$$\boxed{\mathbb{B}_1(\bar{\mathbf{W}}, \bar{\mathbf{U}}) + \mathbb{B}_2(\bar{\mathbf{W}}, \bar{\mathbf{U}}, \bar{\mathbf{U}}) = \mathbb{L}(\bar{\mathbf{W}}) + \mathcal{F}'(\bar{\mathbf{U}}, \text{Res}_s(\bar{\mathbf{U}}))} \quad (2.24)$$

Alternatively, equation 2.24 can be rewritten with operator notation, introducing new $\mathcal{L}_{\mathcal{M}'}$ and $\mathcal{L}_{\mathcal{C}'}$ operators that represent the total closure of the momentum and continuity equations, respectively:

$$\mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p') = \mathcal{L}_{\mathcal{M}}(\mathbf{u}', p') + \nabla \cdot (\mathbf{u}' \otimes \bar{\mathbf{u}}) + \nabla \cdot (\bar{\mathbf{u}} \otimes \mathbf{u}')$$

$$\mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p') = \frac{\partial \mathbf{u}'}{\partial t} + \underbrace{\nabla \cdot (\mathbf{u}' \otimes \mathbf{u}')}_{\text{Reynolds term}} + \underbrace{\nabla \cdot (\mathbf{u}' \otimes \bar{\mathbf{u}}) + \nabla \cdot (\bar{\mathbf{u}} \otimes \mathbf{u}')}_{\text{Cross term}} + \nabla p' - \nu \Delta \mathbf{u}' \quad (2.25)$$

$$\mathcal{L}_{\mathcal{C}'}(\mathbf{u}') = \nabla \cdot \mathbf{u}' \quad (2.26)$$

This allows to rewrite equation 2.24 as separate resolved-scale momentum and continuity equations, viz.:

$$\begin{aligned} (\bar{\mathbf{w}}, \mathcal{L}_{\mathcal{M}}(\bar{\mathbf{u}}, \bar{p}))_Q &= (\bar{\mathbf{w}}, \mathbf{f})_Q - (\bar{\mathbf{w}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_Q \\ (\bar{q}, \mathcal{L}_{\mathcal{C}}(\bar{\mathbf{u}}))_Q &= -(\bar{q}, \mathcal{L}_{\mathcal{C}'}(\mathbf{u}'))_Q \end{aligned} \quad (2.27)$$

It should be noted that these equations are solved at every LES time step, with the time-derivative term obtained through equation 2.20.

2.7. Data-driven modelling of the closure terms

The approach adapted in this thesis is to model the $\mathcal{L}_{\mathcal{M}'}$ and $\mathcal{L}_{\mathcal{C}'}$ terms by using a multi-layer perceptron (MLP) architecture of a neural network with resolved-scale terms as input features. Neural networks have been recently used (usually with partial success) for a wide range of CFD-related problems, such as RANS or LES closure modelling, deconvolution operation in LES, dynamic coefficient procedures, and others [9]. As this is a relatively young effort, improvements in such approaches are expected, and they can potentially break the deadlock in turbulence modelling and lead to new physical insights. Even more recently, new efforts have been made to embed existing physical insight into the architecture of the neural network and its associated data curation [22]. For example, Ling et al [23] took advantage of an integrity basis to force the neural network to search for a RANS closure that was invariant to rotation. Raissi et al [24] added additional loss terms during the training of the network to further constrain it to obey physical principles, such as periodicity. Not surprisingly, such developments have led to training acceleration accuracy improvement.

The multilayer perceptron neural network architecture was chosen for the purposes of this thesis. It is the simplest architecture available, as it is in our interest to maintain the dependence of ANN performance to its specific parameters to the minimum, in order for our results to be generalizable to other problems. The first perceptron was developed by Rosenblatt in 1958 [25], which consisted of only one layer with a single scalar output. It was later quickly improved by other researchers, leading to the emergence of multilayer perceptron, the variant used in this study. A detailed overview of the perceptron model is given in [15], while a brief version will be given here for the present context.

A schematic of an MLP neural network is provided in figure 2.4. The network consists of nodes organized into several layers. The first layer is an input layer, where the resolved-scale input features vector \mathbf{x} is inserted, built up from relevant resolved-scale features. For example, $(\bar{\phi}_i, \bar{u})_{Q_n}$, $(\bar{\phi}_i, \bar{v})_{Q_n}$, $(\bar{\phi}_i, \bar{w})_{Q_n}$, $(\bar{\phi}_i, \bar{p})_{Q_n}$ (with i being the element shape function index) are all fitting choices,

but more features can be added. The nodes make up the heart of the network, and each node has an associated scalar weight and bias. First, these parameters are initialized from a random distribution, usually limited between 0 and 1. Afterward, they are adjusted during the training of the neural network with reference data. This happens in two passes, so-called forward and backward passes. During the forward pass, the MLP computes a prediction based off the currently stored node parameters (weights w_{ij} , biases b_i). This computation proceeds layer-by-layer. First, information passes from the input layer \mathbf{x} to the first hidden layer \mathbf{h} . Each entry of the vector \mathbf{x} then contributes to an input to the first hidden layer, viz:

$$h_{i, \text{input}} = \sum_{j=1}^{n_i} w_{ij}^1 x_j + b_i^1$$

This input to the hidden layer is fed through an activation function $f_{\text{activation}}$:

$$h_i = f_{\text{activation}}(h_{i, \text{input}})$$

The forward pass from the first hidden layer to the next hidden layer or output layer proceeds in the same way. In the end, the forward pass is finished with a prediction vector \mathbf{y} , which is solely determined by the input vector \mathbf{x} and all the weights and biases, or the current state of the MLP. After this stage, the error is computed through a loss function, which compares the prediction vector to the truth vector $\tilde{\mathbf{y}}$. The error vector $\mathbf{e} = L(\mathbf{y}, \tilde{\mathbf{y}})$ is then returned. This vector can be thought of as an alternative input vector \mathbf{x} , but for the backward pass. Thus, backward pass begins with an output layer and proceeds backwards, first to the ultimate hidden layer, then penultimate, until it reaches the first hidden layer. At each stage, the backpropagation algorithm adjusts the weights and biases of the MLP to reduce the net error. For more details on this procedure, please refer to section 4.4 in the book by Haykin [15]. In short, it takes advantage of the chain rule to compute how much each weight/bias scalar contribute to the error in the output vector and adjusts it accordingly. After both passes are finished for the complete dataset (all the pairs $\mathbf{x}, \tilde{\mathbf{y}}$), one *epoch* of training is said to be passed. The process is repeated until the loss function L returns a net error below a certain tolerance.

2.8. Modeling workflow

In this section, a high-level overview is provided of the data curation, training and validation. A flowchart is given in figure 2.5. First, DNS data of TCF at $Re_\tau = 180$ is generated with `cha.cpp` solver. This is a hybrid solver, specifically developed for an efficient simulation of a turbulent channel flow, which uses a spectral method in span- and streamwise directions, while a finite-volume method is used in the wall-normal direction. The DNS mesh contains $N_{x, \text{DNS}} = N_{z, \text{DNS}} = 128$ spectral modes, and $N_{y, \text{DNS}} = 164$ finite-volume elements.

The DNS solver is called within the main routine, where DNS snapshots at given time intervals (consistent with LES time step) are used to:

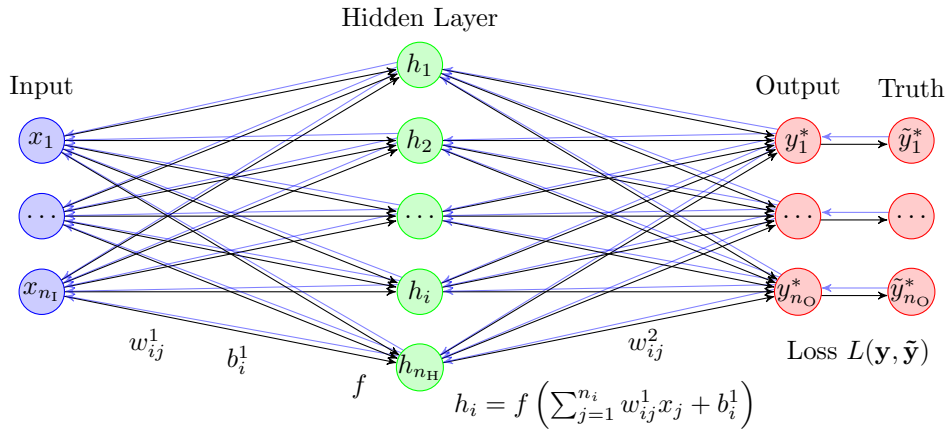


Figure 2.4: Schematic of a multilayer perceptron neural network (Own Work). Black arrows denote the forward pass, blue arrows denote the backward pass of the training procedure.

- Compute a "perfect" (LES) $\bar{\mathbf{U}}$ as the H^1 projection of DNS onto LES grid. The LES mesh is made up of $N_{x,LES} = N_{y,LES} = N_{z,LES} = 16$ (or 32) elements in three coordinate directions.
- Using both the DNS and $\bar{\mathbf{U}}$ solution, compute the "perfect" closure terms $\mathcal{L}_{\mathcal{M}'}$ and $\mathcal{L}_{\mathcal{C}'}$, according to equations 2.25 and 2.26.
- Also compute the feature quantities based on the LES solution. These will be used as inputs to the MLP.
- Save both the closure terms and features to file.

Afterward, a script named `readCT.py` is called to convert the raw ASCII closure and feature term files into binary NumPy arrays `pdata.npz`, which are more convenient to pre-process and manipulate.

At this point, the resolved-scale input features are assembled into the time-space stencil in `src/data.py` (explained in more detail in section 4.1). In short, `src/data.py` handles the complete transformation of snapshots of training data (defined per element and per time step) into complete feature vectors \mathbf{x} , which span across multiple time steps and elements. It should be noted that as in [26], in this thesis LFS approach is taken, meaning that features from the current and previous time steps are used to predict closure terms for the next time step.

Finally, the prepared feature-label pairs are either written to disk as NumPy files or are saved in RAM. The complete dataset is split into training and validation sets, and only the training set is available to the model during the training phase. Training examples are fed into the TensorFlow framework to train the parameters of the neural network. Once the MLP is sufficiently converged, the resulting model is compared against the validation set for the first indication of its performance. Afterward, a posteriori validation is completed through `insML.cpp`, where an ANN generated closure model is injected into the TCF solver and compared against the projected DNS.

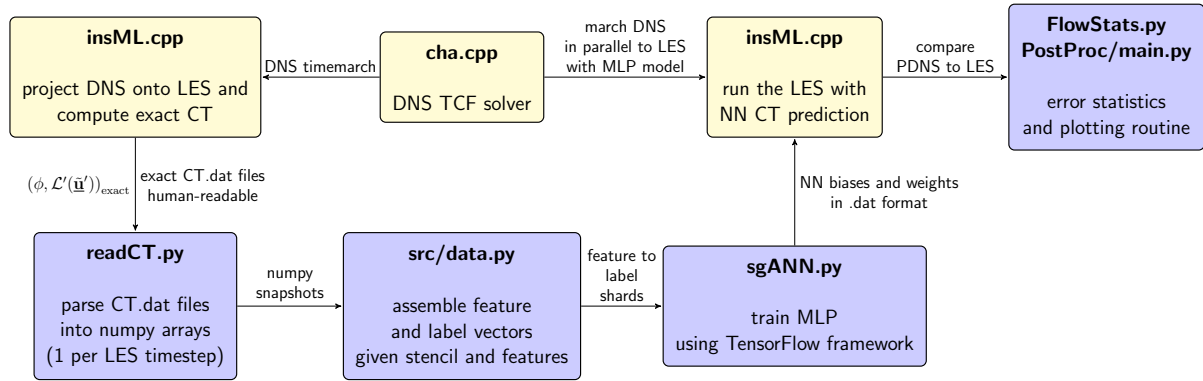


Figure 2.5: Methodology flow chart.

2.9. Solver validation and new closure term definition

To validate `insML.cpp` and confirm that DNS projection can be reconstructed when provided with the perfect closure terms, a validation study was carried out. The DNS was marched for 600 time steps (equivalent to 10 LES time steps), and LES projection of the DNS solution was carried out each LES time step. The LES mesh in this case was chosen to have 16^3 elements. The perfect closure terms, computed directly after subtracting the DNS from the DNS projection, were written to file. After the run was completed, these closure term files are used to reproduce the perfect DNS projection through an LES simulation.

Previously, the closure terms were defined based on the unresolved-scale velocity field, as follows. From the DNS velocity field, an exact solution vector \mathbf{U} is obtained. Then, the resolved velocity field is obtained as DNS projection, i.e., $\bar{\mathbf{U}} = \text{proj}_N^{\bar{\mathbf{V}}^s} \mathbf{U}$. Afterward, the unresolved-scale field is computed simply as $\mathbf{U}' = \mathbf{U} - \bar{\mathbf{U}}$. The closure terms are then computed based on equations 2.25 and 2.25.

While this seems an entirely reasonable approach, completely following the earlier discussion in section 2.3, it assumes that the DNS solution is exactly equivalent to the true solution, which is not always true in practice, unfortunately. In particular, it was found that the DNS solution doesn't satisfy the continuity equation sufficiently, see figure 2.6. The original DNS solution is obtained with a spectral method in span- and streamwise directions and with a finite volume method in the wall-normal direction. However, to compute the closure terms, grid points need to be defined, and the spectral solution needs to be interpolated in span- and streamwise directions, and this can introduce additional errors, in particular with the continuity terms.

Likely because of this, when it was attempted to march the LES with "exact" closure terms, the solution couldn't reach a sufficient tolerance in the corrector pass loop and stalled. This indicates that the Jacobian used in the corrector pass loop could not effectively handle the provided closure terms in their current form. To further investigate this issue, the pressure field was constrained by inserting the exact pressure (obtained as projection of DNS at the next time step). This solved the issue and corrector pass tolerance could be reached, further indicating that the continuity closure is the cause of the problem (due to strong coupling between the continuity equation and

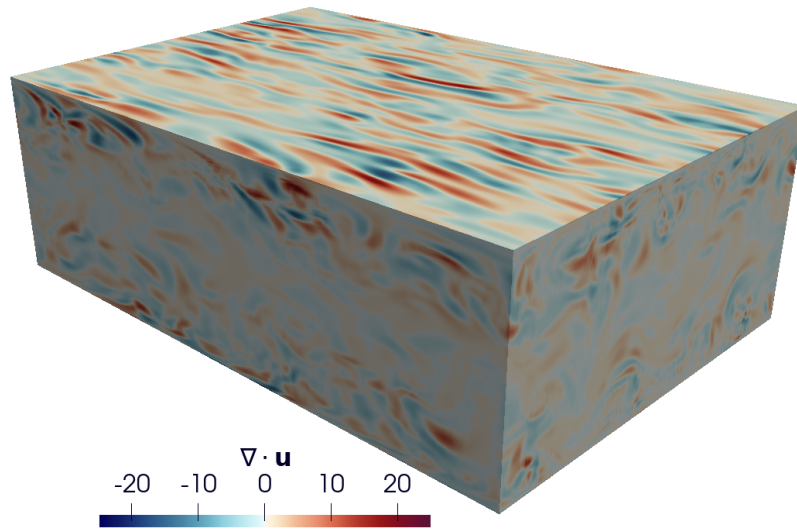


Figure 2.6: Divergence of velocity field obtained from (interpolated) DNS, computed with the ParaView "Divergence" filter.

the pressure field).

An alternative formulation for the closure terms was sought to alleviate this issue. It was found that "enforcing" the closure terms to satisfy equation 2.27 removes the need to insert exact pressure and also significantly reduces an error between the LES solution computed from exact closure terms and one obtained directly as projection of the DNS. As figure 2.7 shows, this drastically decreases the error between the obtained LES and computed DNS projection, in contrast to closure term computation obtained with the previously used formulation.

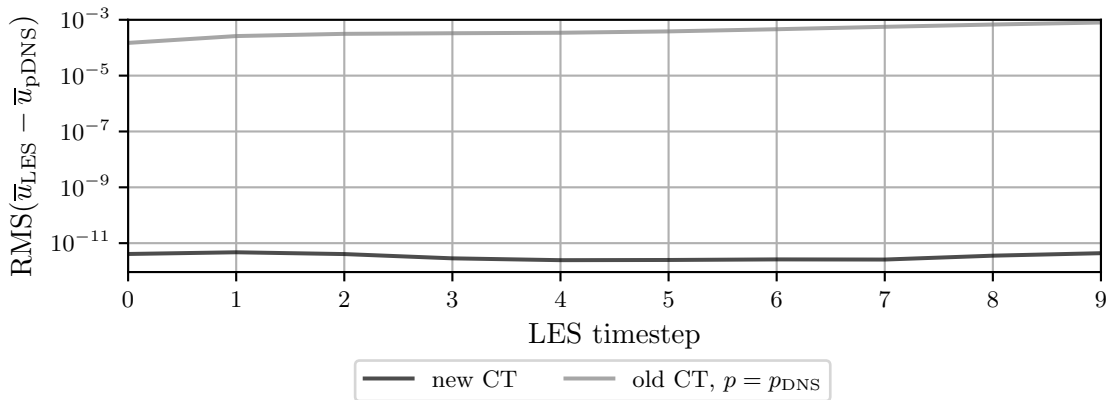


Figure 2.7: Root-mean-square error of the streamwise velocity obtained directly from DNS projection and indirectly from the perfect closure terms.

To conclude, the closure terms are defined in a new manner, as follows:

$$\begin{aligned}
 (\bar{\mathbf{w}}, \mathcal{L}_{\mathcal{M}}'(\bar{\mathbf{u}}, \bar{p}, \mathbf{u}', p'))_{\Omega} &= (\bar{\mathbf{w}}, \mathbf{f})_{\Omega} - (\bar{\mathbf{w}}, \mathcal{L}_{\mathcal{M}}(\bar{\mathbf{u}}, \bar{p}))_{\Omega} \\
 (\bar{q}, \mathcal{L}_{\mathcal{C}}'(\mathbf{u}'))_{\Omega} &= -(\bar{q}, \mathcal{L}_{\mathcal{C}}(\bar{\mathbf{u}}))_{\Omega},
 \end{aligned}$$

where the closure terms on the left-hand side are determined *solely* by the weak resolved-scale forms on the right-hand side. Implicitly, this leads to:

$$(\bar{q}, \mathcal{L}_C'(\mathbf{u}))_{\Omega} = (\bar{q}, \mathcal{L}_C'(\bar{\mathbf{u}} + \mathbf{u}'))_{\Omega} = 0, \quad (2.28)$$

by linearity of the continuity (divergence) operator. Thus, the new closure terms are defined to *enforce* the continuity of the complete velocity field solution \mathbf{u} .

This formulation circumvents the need to perform computationally expensive integration procedures over the unresolved-scale solution \mathbf{U}' and avoids the possible errors due to integration of weak forms. Additionally, it removes the need to explicitly define the unresolved-scale contributions to the closure terms, in particular the unresolved-scale time derivative term, $\partial\mathbf{u}'/\partial t$. This term is defined from varying time-instances for resolved-scale (LES or DNS projection) and unresolved-scale (DNS) simulations. While they both use the same second-order approximation for the time-derivative, DNS uses a much smaller time step size compared to LES (coarse-scale quantities). However, the newly introduced approach completely avoids this issue.

It should be noted that for a given element, there are three scalar closures for momentum, $\mathcal{L}_M'(\bar{\mathbf{u}}, \mathbf{u}', p')$ and one for continuity, $\mathcal{L}_C(\mathbf{u}')$. On an element level, each of these terms is weighted with the *element* shape functions $\bar{\psi}_q$ or $\bar{\varphi}_q = [\bar{\varphi}_{x,q} \quad \bar{\varphi}_{y,q} \quad \bar{\varphi}_{z,q}]$, of which there are 8 ($q = 1, \dots, 8$), corresponding to 8 vertices of an individual element. Thus, in total, the model needs to predict 32 closure scalars. However, element shape functions predictions are only implemented in the solver for the parallelization of the assembly procedure. Thus, as the final solution vector $\bar{\mathbf{U}} = [\bar{\mathbf{u}} \quad \bar{p}]$ is tested with test functions $\bar{\mathbf{w}}$ (equal to basis functions in our case of Galerkin method) that span multiple elements, the *sum* of these contributions over eight element shape functions is the determining contribution.

3

Stabilization strategies

In this chapter, stabilization strategies, their implementation and effects are described. Firstly, section 3.1 discusses the physical impact of H^1 and nodal projectors on the turbulent energy spectrum. Next, three stabilization strategies are proposed, which could lead to improved stability of the LES simulations with ANN closure. Section 3.2 introduces the idea of convolution of the training dataset with a filter to impose a prescribed energy spectrum. Section 3.3 introduces a new loss function that can improve the accuracy of the network when it has to carry out a large number of predictions simultaneously. Lastly, section 3.4 explains how training data can be augmented through a rotation of the coordinate axes to aid the generalization abilities of an ANN model.

3.1. Effect of H^1 projector on energy spectrum

As was mentioned previously, H^1 projector was chosen in this thesis due to a possible advantage it may provide in terms of the locality of the fine-scale Green's function. However, no further analysis was carried out on what such projection implies in terms of the turbulence characteristics. Although it seems to be beneficial to use H^1 projection from a mathematical point of view, it can have undesirable effects from a physical standpoint. Referring to equation 2.19, it is straightforward to see that involvement of the gradient in the projector is likely to skew the turbulence properties. This is evidenced by the shift in the energy spectra, shown in figure 3.1. Using an H^1 projector leads to increased energy content away from the wall and skewed energy spectrum in the first element at the wall ($y^+ = 20$) when compared to real DNS. This is not the case for a nodal projector, which returns an energy spectrum that lies much closer to the DNS energy spectrum. While both nodal and H^1 spectra can be reproduced without issue if the completely exact closure terms are provided, it is possible that the H^1 energy spectrum, with its relatively increased high-wavenumber energy content, might be more sensitive to errors in the closure terms induced by the machine learning model. The errors in the closure term predictions are also likely to be of high-frequency (small wavelength), as the model prediction is based on *local* input features and

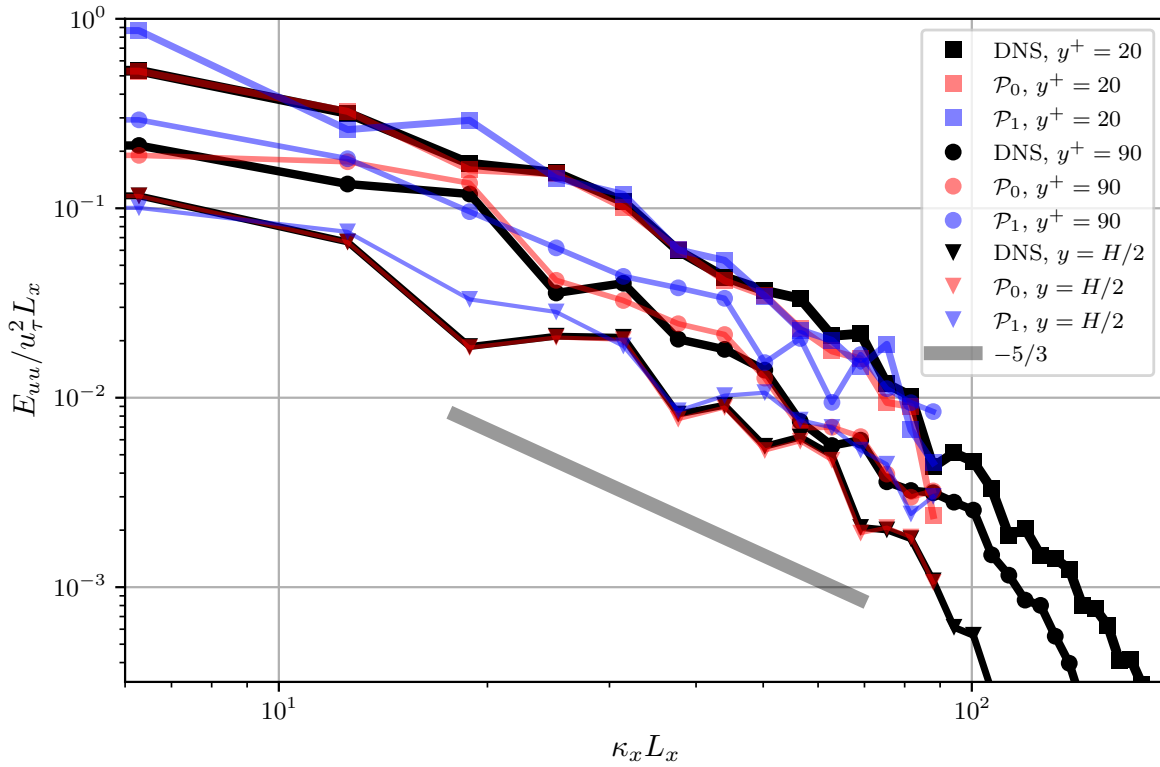


Figure 3.1: One-dimensional energy spectra from the DNS solution (black) and projections onto LES 32^3 mesh, \mathcal{P}_0 - nodal, $\mathcal{P}_1 - H_0^1$ projection.

are provided for each element shape function specifically.

3.2. Strategy 1: Filtering the ANN target

There are several ways to introduce physical insights into the design of a neural network. One of them is through careful selection and manipulation of the desired output, i.e., the truth vector that the machine learning model must predict based on the given output. As was shown in the previous section, due to inaccuracies in this approximation of \mathcal{F}' , the resolved scales vector $\bar{\mathbf{U}}$ accumulates energy at the finest scales. Thus, a potentially effective strategy would be to somehow direct the network to decrease the amount of energy that is transferred to the smallest resolved scales of $\bar{\mathbf{U}}$ through the forcing of \mathcal{F}' . This is not as straightforward as tuning some coefficients in eddy-viscosity models, since there are many more degrees of freedom in the ANN that are not easily interpretable. However, one can "teach" the network to decrease the energy transfer by introducing a filter for the LES field used for target closure term computation. In other words, now the closure terms are defined through DNS solution \mathbf{U} and Fourier filtered projected solution $\bar{\mathbf{U}}_f$. The network will then be rewarded for dampening high frequency modes, which can potentially be an effective approach to tackle high-frequency numerical instabilities introduced due to ANN error.

Filtering Approach

A filtering function was implemented inside the training data generation routine in `insML.cpp`. Simply put, it takes as an input the resolved-scale solution vector $\bar{\mathbf{U}}$, performs a three-dimensional Fourier transform to arrive at $\tilde{\bar{\mathbf{U}}}$, then performs multiplication with the filter weight $w(\boldsymbol{\kappa})$ to produce filtered Fourier modes $\tilde{\bar{\mathbf{U}}}_f$. Finally, these Fourier modes are used to reconstruct the physical LES solution $\bar{\mathbf{U}}_f$. This LES solution is consequently used to calculate the exact filtered closure terms through comparison with DNS projection onto LES mesh, i.e. vector $\bar{\mathbf{U}}$ is substituted by $\bar{\mathbf{U}}_f$ in equation 2.23, leading to new \mathcal{F}'_f defined in equation 3.2.

$$\begin{aligned}\mathbf{x} &= \begin{bmatrix} x & y & z \end{bmatrix}^T \\ \boldsymbol{\kappa} &= \begin{bmatrix} \kappa_x & \kappa_y & \kappa_z \end{bmatrix}^T = 2\pi \begin{bmatrix} \frac{i}{L_x} & \frac{j}{L_y} & \frac{k}{L_z} \end{bmatrix}^T \\ \mathcal{F}(f(\mathbf{x})) &= \int_{\Omega} f(\mathbf{x}) \exp(-\boldsymbol{\kappa} \cdot \mathbf{x}) d\mathbf{x} = \tilde{f}(\boldsymbol{\kappa}) \\ \mathcal{F}^{-1}(\tilde{f}(\boldsymbol{\kappa})) &= \int_{\Omega_{\boldsymbol{\kappa}}} \tilde{f}(\boldsymbol{\kappa}) \exp(\boldsymbol{\kappa} \cdot \mathbf{x}) d\boldsymbol{\kappa} = f(\mathbf{x}) \\ \bar{\mathbf{U}}_f &= \mathcal{F}^{-1}(w(\boldsymbol{\kappa}) \mathcal{F}(\bar{\mathbf{U}})),\end{aligned}\tag{3.1}$$

$$\mathcal{F}'_f = -\mathbb{B}_1(\bar{\mathbf{W}}, \mathbf{U}') - \mathbb{B}_2(\bar{\mathbf{W}}, \bar{\mathbf{U}}_f, \mathbf{U}') - \mathbb{B}_2(\bar{\mathbf{W}}, \mathbf{U}', \bar{\mathbf{U}}_f) - \mathbb{B}_2(\bar{\mathbf{W}}, \mathbf{U}', \mathbf{U}')\tag{3.2}$$

where $\boldsymbol{\kappa}$ is the element wavenumber, computed from i, j, k indices of the element at location \mathbf{x} . The filter weight is defined based on the spectral distance from the so-called Nyquist wavenumber, which is the ultimate wavenumber $\boldsymbol{\kappa}$ that can be resolved by the mesh. For a 16^3 mesh, this Nyquist wavenumber corresponds to:

$$\boldsymbol{\kappa}_{\text{Nyq}} = 2\pi \begin{bmatrix} \frac{8}{L_x} & \frac{8}{L_y} & \frac{8}{L_z} \end{bmatrix}^T\tag{3.3}$$

If this spectral distance is below a given cut-off distance $\Delta_{c/o}$, then the filter weight is given as:

$$\begin{aligned}d &= \|\boldsymbol{\kappa} - \boldsymbol{\kappa}_{\text{Nyq}}\| = \left((\kappa_x - \kappa_{x,\text{Nyq}})^2 + (\kappa_y - \kappa_{y,\text{Nyq}})^2 + (\kappa_z - \kappa_{z,\text{Nyq}})^2 \right)^{1/2} \\ w(\boldsymbol{\kappa}) &= 1, \quad d > \Delta_{c/o} \\ w(\boldsymbol{\kappa}) &= \min \left[\frac{s \cdot d}{\Delta_{c/o}}, 1 \right] = [0, 1], \quad \|\boldsymbol{\kappa} - \boldsymbol{\kappa}_{\text{Nyq}}\| \leq \Delta_{c/o}\end{aligned}$$

Here, ramp slope s and filter width $\Delta_{c/o}$ are manually chosen. This ramp filter is designed to leave the large spatial modes unaffected, while dampening the small spatial modes, which are likely to be inaccurate as they are close to the resolution limit of the LES mesh and suffer the most from closure term errors. By providing filtered closure terms as the goal to the neural network, we teach the network to dampen small scale flow fluctuations.

The effect of filtering on the streamwise energy spectrum is shown in figure 3.2 for a 32^3 case,

with $s = 0.09$ and $\Delta_{c/0} = 15.0$. It should be noted that this stabilization strategy allows for an arbitrary filter choice.

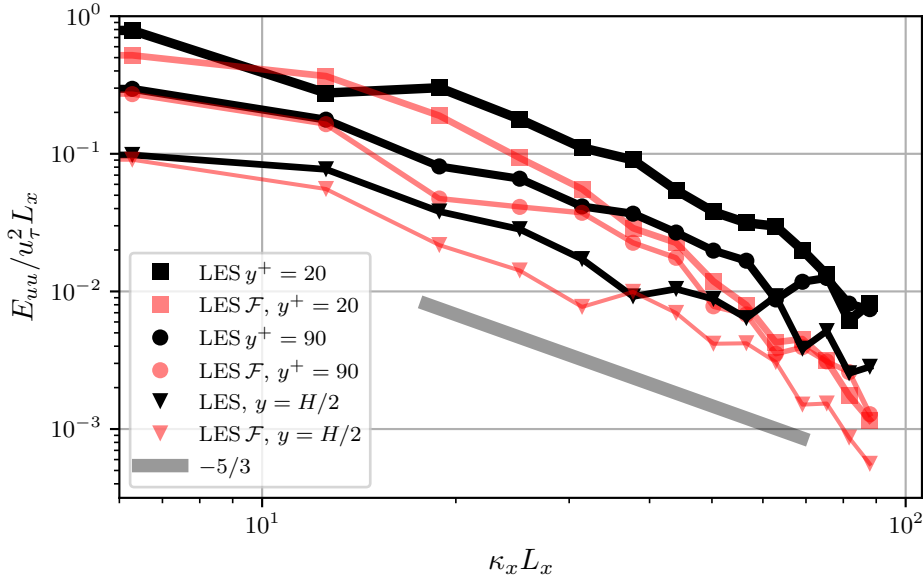


Figure 3.2: Filtered and unfiltered LES streamwise energy spectra.

3.3. Strategy 2: Mini-batch sum loss

Mean squared error loss is the most popular loss mechanism used for ANN regression models. However, this approach only computes the difference in the prediction on the element level, which can lead to systematic biases in the ANN predictions if the model is repeatedly executed many times (such as predicting closures on a CFD mesh with thousands of degrees of freedom). Very small systematic errors in ANN predictions can lead to nonphysical kinetic energy input or drain into/from the flow, as the error is compounded with the number of elements and with each time step. A proposed solution to alleviate this problem is a so-called batch sum loss, defined as follows. For a given (mini-)batch prediction matrix Y and truth matrix \tilde{Y} , both of size $N_b \times N_o$, (number of examples in a batch times the number of outputs in the output vector) the batch sum loss is:

$$L_{\Sigma}(Y, \tilde{Y}) = \sum_j^{N_o} \left[\sum_i^{N_b} Y_{ij} - \sum_i^{N_b} \tilde{Y}_{ij} \right]^2 \quad (3.4)$$

Of course, if only this loss is used, the optimizer will force the individual predictions off their truth values, as only the sum will contribute. Thus, the combination of L_{MSE} and mini-batch sum loss is used. For a mini-batch, it is computed in a straightforward way:

$$L_{\text{MSE}}(Y, \tilde{Y}) = \sum_{ij}^{N_b \cdot N_o} [Y_{ij} - \tilde{Y}_{ij}]^2 \quad (3.5)$$

The actual loss subject to minimization is then:

$$L(Y, \tilde{Y}) = \sum_j^{N_0} \left(\sum_i^{N_b} [Y_{ij} - \tilde{Y}_{ij}]^2 + \left[\sum_i^{N_b} Y_{ij} - \sum_i^{N_b} \tilde{Y}_{ij} \right]^2 \right) \quad (3.6)$$

3.4. Strategy 3: Rotational data augmentation

As the orientation of the coordinate axis is purely arbitrary and primarily chosen based on convention, a way to augment the training dataset is by rotating the coordinate axis and thus rotating the feature and label vectors as well. This way, the model is trained to be invariant to rotation of the coordinate axis, which is an important physical consideration. The original coordinate axes x, y, z can be rotated by yaw angle α about z -axis, pitch angle β about y -axis and roll angle γ about the x -axis, by multiplying with the rotation matrix R , defined as follows [27]:

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \quad (3.7)$$

Three rotational data augmentations are applied, described in table 3.1 and figure 3.3. The purpose of the first rotation is to ensure that the network can predict accurately if the feature vectors orientation is completely changed in opposite direction). Rotations 2 and 3 are to account for misalignment of the feature vectors with the mean flow direction and directions of homogeneity of the flow. It thus enforces new constraints on the model. The addition of this data augmentation should make the network more general (as it exposes it to new data), and hopefully more robust.

Table 3.1: Rotational data augmentations applied.

Rotation	Yaw Angle (deg)	Pitch Angle (deg)	Roll angle (deg)
Original	0	0	0
Rotation 1	180	90	0
Rotation 2	45	45	45
Rotation 3	-45	-45	-45

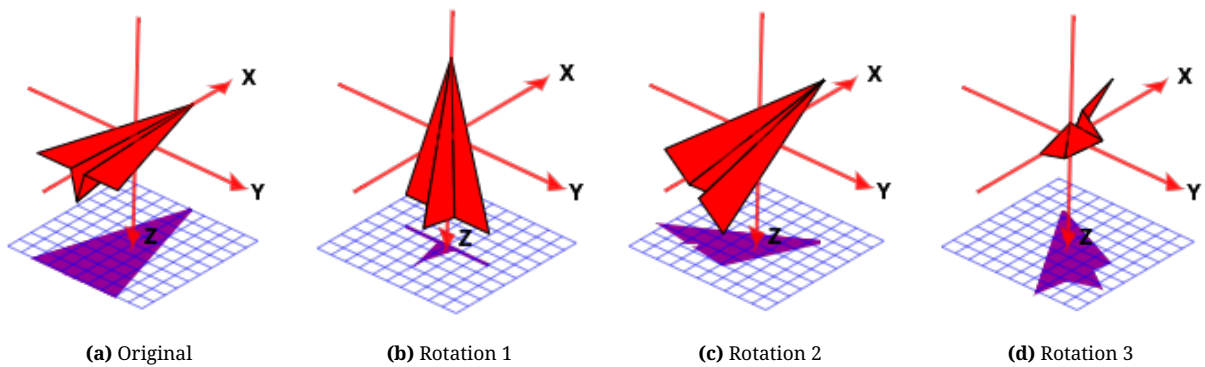


Figure 3.3: Rotational Data Augmentation visualized.

4

Neural network closure model design

In this chapter, a detailed description is given of the design of ANN closure model, a priori validation results and the final selection of ANN models. Specifically, in section 4.1 the description of preliminary ANN closure model design is given. In section 4.2, the training approach for minimization of mini-batch sum loss is described. Lastly, three sensitivity studies were carried out. In section 4.3 the effect of varying ANN architecture on a priori correlations with true terms is discussed. Similarly, sections 4.4 and 4.5 discuss the effect of varying the input features and feature stencil on a priori correlations. The final selection of ANN closure models is provided in section 4.6.

4.1. Preliminary network design

An initial network design was developed for an explorative study, mostly inspired by previous work of A. Bettini [26]. In this section, a description is given of the training approach, which is visualized as a flowchart in figure 4.1. Additionally, values chosen for network hyperparameters are explained.

Several steps were taken in order to reduce network size and complexity. The model now consists of 1 network, which outputs weighted momentum and continuity closures associated with $\bar{\varphi}_q$ and $\bar{\psi}_q$ element shape functions in the element. This model is trained on the data for all eight element shape functions, and uses coarse-scale features as input. Referring to equation 2.25:

$$(\bar{\varphi}_q, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \bar{p}, \mathbf{u}', p'))_e = \text{ANN} \left(\left(\frac{\bar{\varphi}_q}{\partial t} \right)_e, \dots \right) \quad (4.1)$$

The same model is re-used for all 8 element shape functions. This significantly decreases the total model training time and computational resources, as well as increasing the total amount of data that is being fed into the model (as now data for $q = 1, \dots, 8$ is provided). Additionally, this training approach is more generalizable, as it doesn't pay attention to arbitrary indexing of the shape

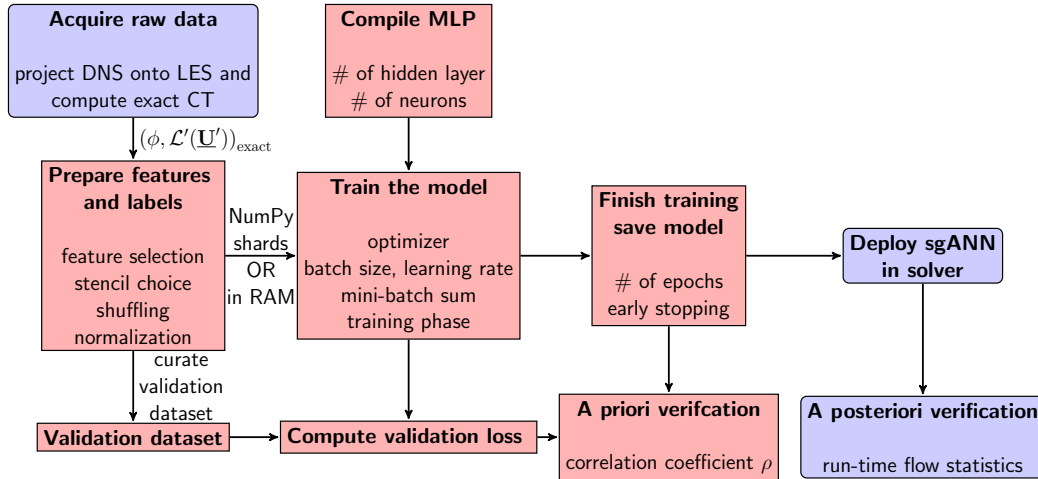


Figure 4.1: Neural network pipeline.

functions. Previously, 8 networks were used (one per shape function), which added to model complexity and increased training time, although the accuracy was likely higher as well.

By stencil hereafter it is meant the time-space domain from which features are taken to predict the closure for a given element. The most compact stencil thus spans the element itself at the current time step. Larger stencils provide more information and *may* improve model accuracy, up until a given point. The fine-scale Green's functions that were derived for simpler PDE's (see figure 2.3) were used as inspiration for the spatial stencil. Three options for the stencil selection are provided, visualized in figure 4.2. For the first one, only one element upstream in x and one from the previous time step is taken, in addition to the element itself, which leads to 4 elements in total. This stencil is used for most initial designs. Stencil 2 is significantly extended, and includes the original element, one upstream and one downstream element, as well as one lateral element to each spanwise side. Thus, 16 elements in total over two time steps. Stencil 3 includes only the element itself at current and previous time step.

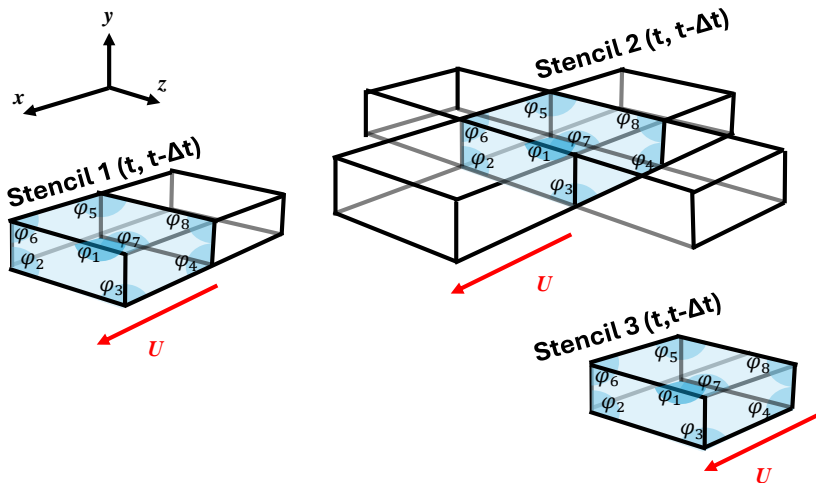


Figure 4.2: Three stencil options - default stencil 1, expanded stencil 2 and single element stencil 3. The labels (predictions) are closures associated to shape functions ϕ_i of the element highlighted by blue.

Bettini [26] suggests using coarse-scale integrated sums/features, which can be succinctly expressed using equations 2.3 and 2.4 as $(\bar{\varphi}_q, \mathcal{L}_M(\bar{\mathbf{u}}, \bar{p}))$ or $(\bar{\varphi}_q, \mathcal{L}_C(\bar{\mathbf{u}}))$. Using these input features leads to very high a priori correlations with exact terms, and for this reason they are also adopted in this thesis. As features averaged coarse-scale terms that are found on the left-hand side of momentum equation 2.2, as well as resolved divergence and basis coefficients of velocity amplitudes \mathbf{a}_i , (equation 2.21) are adopted. Each term is weighted with the given element shape function $\bar{\varphi}_q$ for momentum terms and velocity basis coefficients and $\bar{\psi}_q$ for continuity. At the end of the feature vector, the wall-normal coordinate is inserted. The feature vector \mathbf{x} is given in equation 4.2.

The true labels (perfect predictions) are momentum and continuity closures per shape function, for assembly purposes (4 in total). The output vector \mathbf{y} is given in equation 4.2.

$$\mathbf{x} = \underbrace{\begin{bmatrix} \left(\bar{\varphi}_q, \frac{\partial \bar{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x}, t) \\ \left(\bar{\varphi}_q, \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}})\right)_e(\mathbf{x}, t) \\ \left(\bar{\varphi}_q, \nabla \bar{p}\right)_e(\mathbf{x}, t) \\ \left(\bar{\varphi}_q, -\nu \Delta \mathbf{u}\right)_e(\mathbf{x}, t) \\ \mathbf{a}_i(\mathbf{x}, t) \\ \left(\bar{\psi}_q, \nabla \cdot \bar{\mathbf{u}}\right)_e(\mathbf{x}, t) \\ \vdots \\ \left(\bar{\varphi}_q, \frac{\partial \bar{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x} - \Delta \mathbf{x}, t) \\ \vdots \\ \left(\bar{\varphi}_q, \frac{\partial \bar{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x}, t - \Delta t) \\ \vdots \\ \left(\bar{\varphi}_q, \frac{\partial \bar{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x} - \Delta \mathbf{x}, t - \Delta t) \\ \vdots \\ \left(\bar{\psi}_q, \nabla \cdot \bar{\mathbf{u}}\right)_e(\mathbf{x} - \Delta \mathbf{x}, t - \Delta t) \\ z \end{bmatrix}}_{65 \text{ features}} \quad \mathbf{\hat{y}} = \underbrace{\begin{bmatrix} \left(\bar{\varphi}_q, \mathcal{L}_M'\right)_e(\mathbf{x}, t + \Delta t) \\ \left(\bar{\psi}_q, \mathcal{L}_C(\mathbf{u}')\right)_e(\mathbf{x}, t + \Delta t) \end{bmatrix}}_{4 \text{ labels}} \quad (4.2)$$

To accelerate the training and evaluation of the initial network designs, LES mesh of 16^3 and 32^3 elements were used. As shown in figure 4.3, a time-sequence of 48 snapshots of the closure terms and features was used for the validation dataset, starting from the initial condition at $t = 0.0$ up to $t = 0.3456$. For the training dataset (on a 16^3 mesh) 480 snapshots were recorded from $t = 1.0$ up until $t = 4.455$, while for 32^3 mesh this had to be decreased to 360 snapshots, from $t = 1.0$ up until $t = 3.592$, due to large file sizes (the memory requirement is increased by factor of 8 if the mesh is refined by factor of 2).

There is a time separation of 90 LES time steps between the data used for training and validation for both 16^3 and 32^3 cases. Additionally, the validation dataset relates directly to a posteriori

validation, as there the networks are evaluated together with solver coupling, also starting from the initial condition at $t = 0$. It is thus expected that for the first time step the model will show the same correlations when tested a posteriori as when tested a priori.

During data pre-processing, input features were normalized to the range of $[0, 1]$. The network architecture comprised an input layer with 65 entries of the feature vector (corresponding to stencil version 1 and a complete feature set), a varying number of hidden layers with a constant number of neurons in each, and an output layer of 4 closure predictions. The rectified linear unit activation function was used for the hidden layers and the linear activation function was used for the output layer.

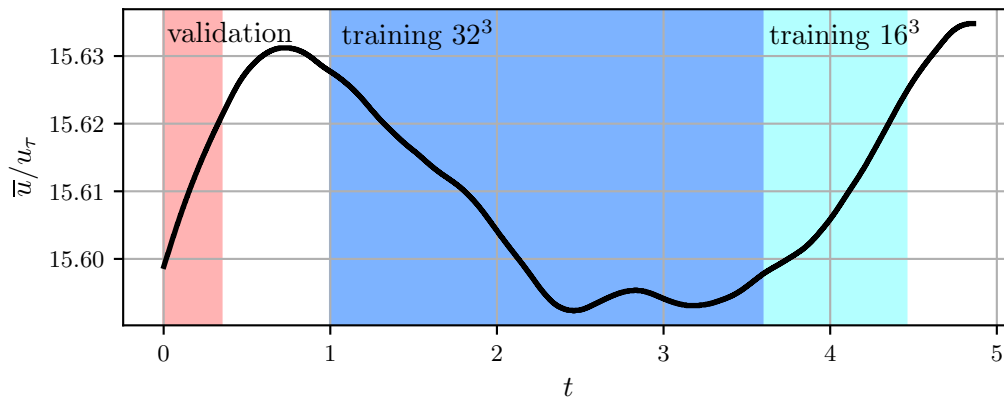


Figure 4.3: Bulk history of DNS flow. Validation data (48 snapshots), training data for 32^3 case (360 snapshots) and for 16^3 case (480 snapshots) highlighted.

The Adam optimizer with the learning rate of 10^{-4} was used to converge the neural network. The loss L was computed as the mean-squared error:

$$L = L_{\text{MSE}}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{N_y} \sum_1^{N_y} (y_i - \tilde{y}_i)^2 \quad (4.3)$$

To avoid overfitting, an early stopping criterion was used. This Keras library feature monitors the validation loss and stops training early once it stops improving. The patience parameter was set to 5, meaning that the validation loss is monitored for over the last 5 epochs, and if no improvement in the loss metric greater than a custom tolerance `min_delta` is found, the training is stopped. For the 16^3 model this value was set to 10^{-6} , while for the 32^3 model it was lowered to 10^{-8} , to account for the decreased magnitude of the closure terms and consequently the loss function values themselves.

The batch size for the training and validation was chosen to be $N_b = 10^4$. It was found that using larger batches utilizes the GPU more effectively and decreases the training time per epoch, while smaller batch sizes improve overall convergence (decrease in the loss metric gained over 1 epoch). A batch size of 10^4 examples has been found to provide optimal performance in terms of network convergence and total training time.

4.2. Mini-batch sum loss implementation

One of the stabilization measures aimed to improve the quality of ANN model prediction is the mini-batch sum loss, previously introduced in section 3.3. At first, both the standard mean-squared error and new mini-batch sum loss were used together. However, it was discovered through observing the loss plot in figure 4.4, that the mini-batch sum loss doesn't show good convergence when used directly together with the mean-squared error.

An improved and more controlled approach is to first subject the network only to the mean squared error loss function and train it until it reaches a sufficiently good correlation and the loss function plateaus. This can be considered an initial phase of model training. The resulting loss function is shown in figure 4.5, and there is a considerable improvement in convergence. The resulting correlation plots are given in figure 4.7, where a probability density function of true output vectors and ANN predictions is shown for each scalar term in the closure vector.

The Pearson correlation coefficient is also used as a measure of ANN model accuracy a priori:

$$\rho_{X,Y} = \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\mathbb{E}[X^2] - (\mathbb{E}[X])^2} \sqrt{\mathbb{E}[Y^2] - (\mathbb{E}[Y])^2}} \quad (4.4)$$

Additionally, expected value (average) of the predictions is computed as

$$E_{\text{truth}} = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \tilde{y}_i \quad E_{\text{pred}} = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} y_i,$$

with \tilde{y}_i and y_i being the i -th entry in true output and model prediction vectors, respectively.

Clearly, the model achieves quite high correlation coefficients between the predicted and true values of the closure terms. However, comparing the expected value (average) of its predictions to the true expected value (obtained from the exact validation data) shows that it has difficulty correctly predicting the average of its predictions. Thus, deviations can be expected in integral flow quantities such as flow kinetic energy from the DNS results. The most significant error is in the continuity term - despite having (almost) perfect correlation, the model's errors still lead to high values of E_{pred} when comparing it to its truth counterparts.

After the model is converged using the MSE loss, the model weights are reloaded, and a new training loop is commenced. Here, only the mini-batch sum loss is active, but the optimizer learning rate is decreased drastically from 10^{-4} to 10^{-9} . It was found through experimentation that mini-batch sum loss is extremely sensitive to the model's weights and biases, and very small updates are needed. The model is then trained until the mini-batch sum loss stops improving and converges. The resulting loss is shown in figure 4.6, and the correlations are provided in figure 4.8. Clearly, even though the model is not aware of the mean-squared loss during the second training phase, the MSE loss is not negatively impacted, and neither are the closure term correlations. This is due to the extremely low learning rate. However, some improvement can be observed in the accuracy of model predictions when considering the averages, comparing figures 4.7 and 4.8.

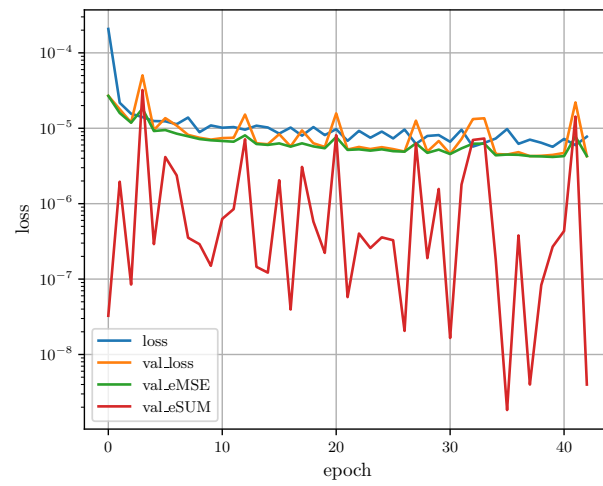


Figure 4.4: Loss evolution (MSE loss in green, mini-batch sum loss in red).

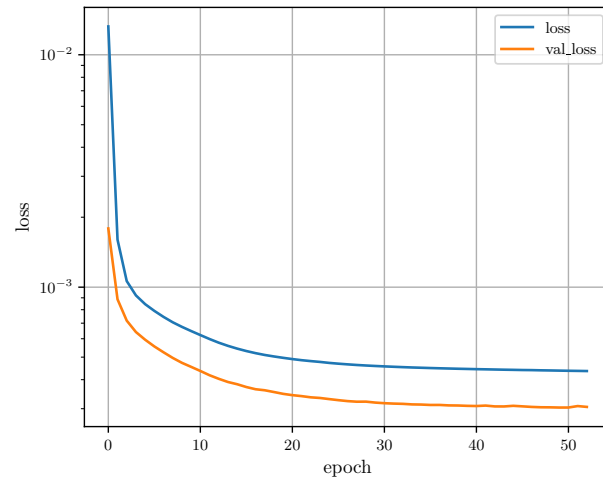


Figure 4.5: Loss evolution, with model subject only to MSE loss.

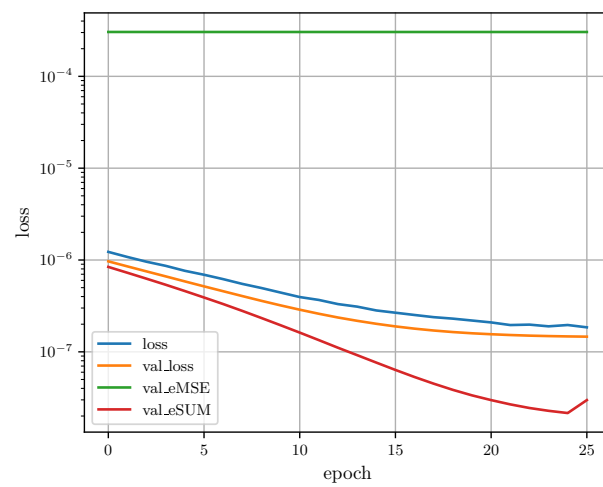


Figure 4.6: Loss evolution, with model re-trained with mini-batch sum loss.

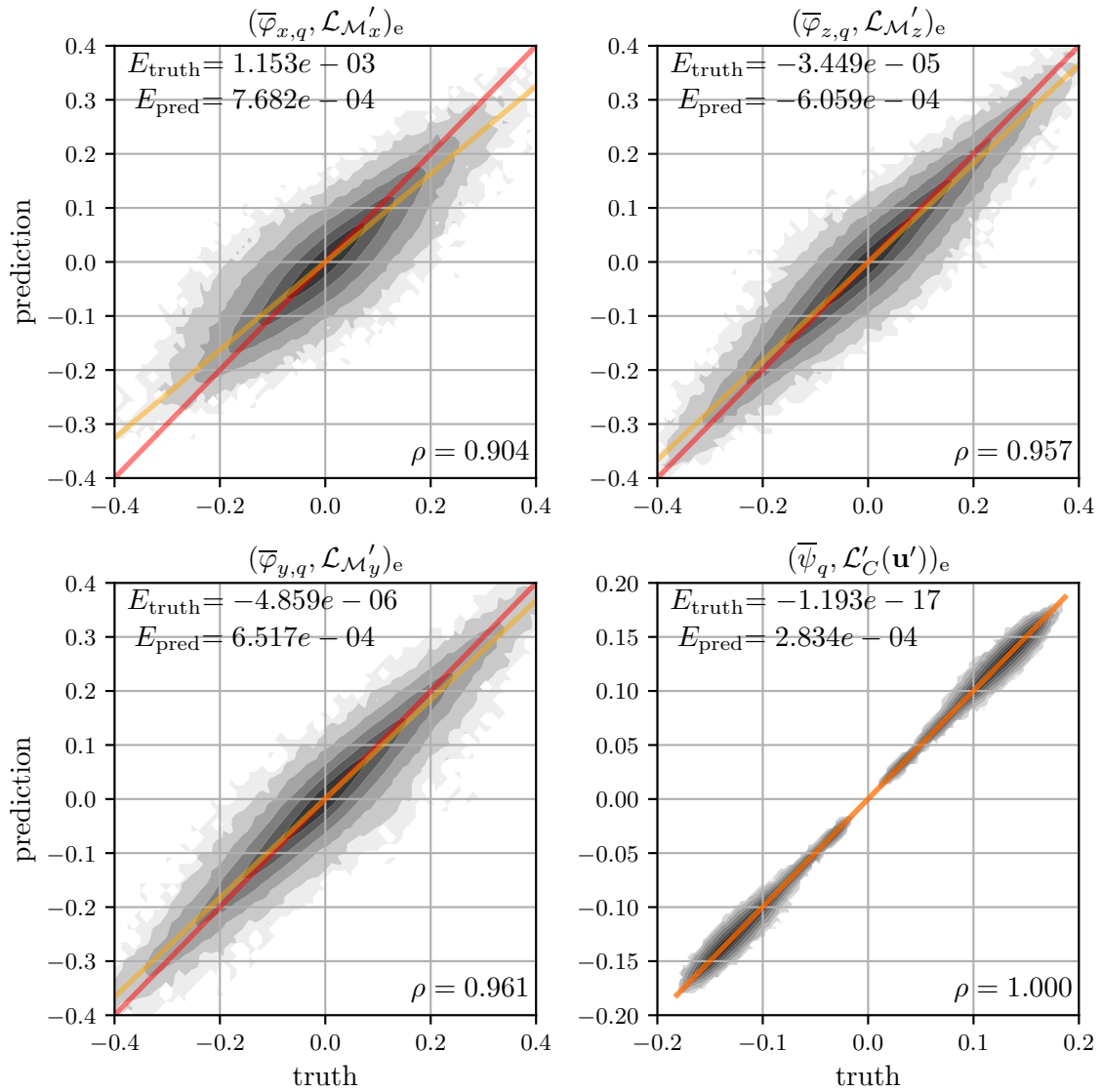


Figure 4.7: True closures from validation (unseen) data against ANN predictions of validation features, for a model subject to only MSE loss. Red line shows the perfect line of best fit, yellow line shows the actual line of best fit. Expected values (averages) of true and predicted closure terms in top left corners, Pearson's correlation coefficient in bottom right corners.

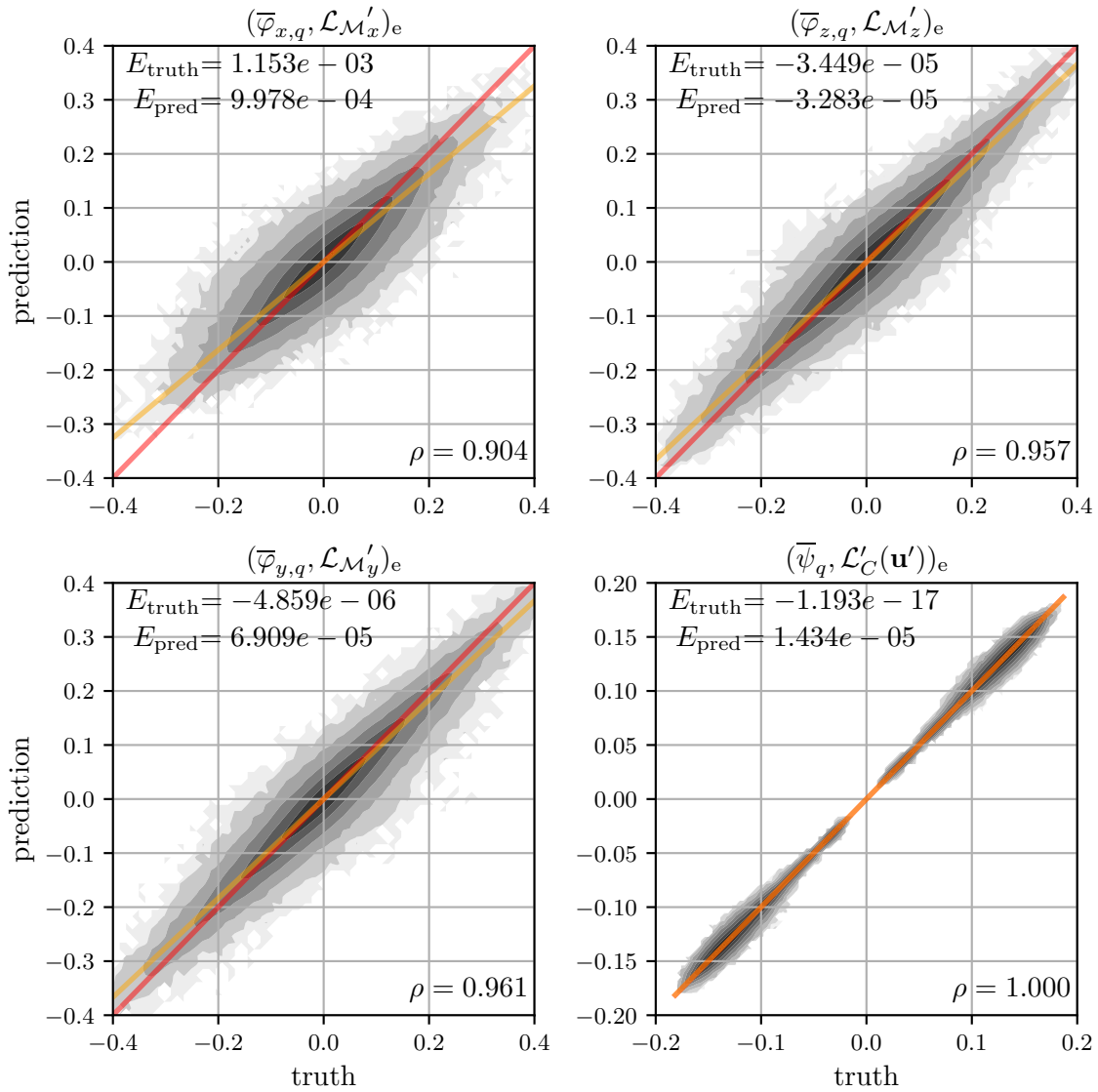


Figure 4.8: True closures from validation (unseen) data against ANN predictions of validation features, for a model re-trained with mini-batch sum loss. Red line shows the perfect line of best fit, yellow line shows the actual line of best fit. Expected values (averages) of true and predicted closure terms in top left corners, Pearson's correlation coefficient in bottom right corners.

4.3. Sensitivity of model accuracy to model architecture

To determine the best model architecture in terms of model accuracy and computational cost associated with the model training and execution, a range of models was trained, with varying architecture. To measure the accuracy, correlations of predictions with true labels, computed on the validation dataset, are used. In all cases, only MSE loss was used. There was no significant difference in the observed Pearson correlation coefficients with or without additional mini-batch sum loss training, which only has an influence on E_{pred} term. This is also the case for sections 4.4 and 4.5. An example of correlation plot for a model with one hidden layer with 100 neurons is given in figure 4.7. Tested model architectures together with resulting correlation coefficients are given in figure 4.1. Evidently, there is a gain in accuracy when comparing perceptron model with zero hidden layer to one hidden layer models, in particular with 100 or 200 neurons in the hidden layer. However, not much improvement is found with a larger model architecture beyond one hidden layer with 100 neurons. For this reason, this will be the adopted architecture from here onwards.

Table 4.1: Effect of model architecture on a priori correlations.

N_{layers}	N_{neurons}	ρ_{M_x}	ρ_{M_y}	ρ_{M_z}	ρ_C
0	0	0.820	0.930	0.920	0.999
1	30	0.879	0.948	0.940	0.999
1	100	0.904	0.961	0.957	0.999
1	200	0.911	0.963	0.959	0.999
2	30	0.894	0.947	0.954	0.999
2	100	0.911	0.965	0.960	0.999
2	200	0.916	0.966	0.962	0.999

4.4. Sensitivity of model accuracy to input features

Another aspect of the model is what coarse-scale features are to be used to predict the closure terms. A sensitivity study was thus carried out to determine the relative importance of the coarse-scale terms that are found on the left-hand side of momentum equation 2.2, the resolved divergence, and basis coefficients of velocity amplitude \mathbf{a}_i . Utilizing model architecture with 1 hidden layer with 100 neurons, a series of models was trained. One model was trained with all the features of equation 4.2. Then six other models were trained, each with one of the features missing. The results are provided in table 4.2. Evidently, the coarse-scale time-derivative and convective terms are the most important for the momentum closure, while unsurprisingly, the resolved divergence term is needed for high-correlation predictions of unresolved divergence. The pressure gradient term, diffusion term and basis coefficients have little effect on the a priori accuracy. They were omitted to minimize the input data to the model and accelerate the solver, as less integration procedures have to be carried out over the coarse scale solution.













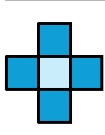

Table 4.2: Effect of input features on a priori correlations.

Features \mathbf{x}	ρ_{M_x}	ρ_{M_y}	ρ_{M_z}	ρ_C
complete	0.904	0.961	0.957	0.999
no $\left(\overline{\varphi}_q, \frac{\partial \overline{\mathbf{u}}}{\partial t}\right)_e$	0.721	0.779	0.764	0.999
no $\left(\overline{\varphi}_q, \nabla \cdot (\overline{\mathbf{u}} \otimes \overline{\mathbf{u}})\right)_e$	0.812	0.915	0.896	0.999
no $\left(\overline{\varphi}_q, \nabla \overline{p}\right)_e$	0.901	0.961	0.956	0.999
no $\left(\overline{\varphi}_q, -\nu \Delta \mathbf{u}\right)_e$	0.894	0.962	0.956	0.999
no $\left(\overline{\psi}_q, \nabla \cdot \overline{\mathbf{u}}\right)_e$	0.863	0.947	0.940	0.891
no \mathbf{a}_i	0.894	0.956	0.952	0.999
$\left(\overline{\varphi}_q, \frac{\partial \overline{\mathbf{u}}}{\partial t}\right)_e, \left(\overline{\varphi}_q, \nabla \cdot (\overline{\mathbf{u}} \otimes \overline{\mathbf{u}})\right)_e, \left(\overline{\psi}_q, \nabla \cdot \overline{\mathbf{u}}\right)_e$	0.884	0.958	0.954	0.999

4.5. Sensitivity of model accuracy to the feature stencil

Last, but not least, the model accuracy was evaluated with respect to the width of the input stencil in the physical and temporal domains. Again, a model architecture with 1 hidden layer of 100 neurons was used, with $\left(\overline{\varphi}_q, \frac{\partial \overline{\mathbf{u}}}{\partial t}\right)_e, \left(\overline{\varphi}_q, \nabla \cdot (\overline{\mathbf{u}} \otimes \overline{\mathbf{u}})\right)_e, \left(\overline{\psi}_q, \nabla \cdot \overline{\mathbf{u}}\right)_e$ as input features (for each time and space element). The results are presented in table 4.3, where the light blue denotes the element for which the closure is predicted, and the dark blue denotes its neighbors (in space or time). Stencils D, G and B correspond to stencils 1, 2 and 3 in figure 4.2.

Table 4.3: Effect of stencil width on a priori correlations. For space stencil, right is upstream, up/down is spanwise. For time stencil, right is in the past.

Name	Space stencil	Time stencil	ρ_{M_x}	ρ_{M_y}	ρ_{M_z}	ρ_C
A			0.350	0.347	0.313	0.999
B (3)			0.783	0.875	0.858	0.999
C			0.808	0.907	0.891	0.999
D (1)			0.904	0.957	0.961	0.999
E			0.888	0.961	0.954	0.999
F			0.901	0.965	0.959	0.999
G (2)			0.886	0.957	0.955	0.999

It is evident that only including the element itself at $t = t^n$ (stencil A) is not sufficient and provides very poor correlations. Including coarse-scale information from the previous time-step enhances the quality of predictions (comparing stencils A and B). Including one element upstream further improves the correlations for all three momentum components (comparing B and D). It should be noted that adding any other elements in space, as well as including information at $t = t^{n-2}$ doesn't significantly improve model accuracy. Thus, the most promising stencil is still stencil D (1), which includes features at current and previous time step ($t = t^n$ and $t = t^{n-1}$) and element itself with its

upstream neighbor ($x = x_i, x = x_{i-1}$). Another interesting stencil is stencil B (3), as it still provides reasonable correlations without including other elements.

4.6. Chosen Network Designs

In this section, an overview of finalized ANN models is provided. An emphasis was placed on network simplicity, to decrease the training and execution cost. Two input feature stencils will be tested - stencil 1 and 3. Only the most relevant features were chosen, namely resolved time-derivative term $\left(\bar{\varphi}_q, \frac{\partial \bar{\mathbf{u}}}{\partial t}\right)_e$, resolved convective term $\left(\bar{\varphi}_q, \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}})\right)_e$, and resolved divergence term $\left(\bar{\psi}_q, \nabla \cdot \bar{\mathbf{u}}\right)_e$. A single model was trained over all the element shape functions $\bar{\varphi}_q$ and $\bar{\psi}_q$, and re-used 8 times during solver deployment to predict the unresolved closure terms weighted with a given element shape function, based off the resolved terms weighted with the same element shape function. All the models are trained in two phases as described in section 4.2, except for model M1₁₆MSE, which only experienced the first training phase, in order to quantify the impact of mini-batch sum training loss. As accuracy metrics, correlation coefficients for the x - and y -momentum terms were used, as well as percentage error calculated in the prediction of average closure terms:

$$\epsilon(E_{\mathcal{M}_x}) = \frac{E_{\text{pred}, \mathcal{M}_x} - E_{\text{truth}, \mathcal{M}_x}}{E_{\text{truth}, \mathcal{M}_x}} \quad (4.5)$$

The model M1₁₆ was also tested with both a filtered (M1₁₆F) and a rotationally augmented (M1₁₆A) training dataset. To investigate the effect of mesh refinement, both 16³ and 32³ LES meshes are used, with default model configuration for stencil 1 and 3. The 32³ models are also trained at varying CFL numbers (i.e., LES time steps) of the training dataset - 0.5 and 1.0.

Table 4.4: Selected Network Designs.

Name	Mesh	CFL	$\rho_{\mathcal{M}_x}$	$\rho_{\mathcal{M}_y}$	$\epsilon(E_{\mathcal{M}_x})$	Note
M1 ₁₆	16 ³	0.5	0.884	0.958	19 %	
M1 ₁₆ F	16 ³	0.5	0.823	0.938	19 %	Filtered
M1 ₁₆ A	16 ³	0.5	0.848	0.946	71 %	Augmented
M1 ₁₆ MSE	16 ³	0.5	0.884	0.958	73 %	only MSE loss
M3 ₁₆	16 ³	0.5	0.783	0.875	24 %	
M1 ₃₂ CFL0.5	32 ³	0.5	0.922	0.978	9 %	
M3 ₃₂ CFL0.5	32 ³	0.5	0.887	0.959	25 %	
M1 ₃₂ CFL1.0	32 ³	1.0	0.889	0.943	9 %	
M3 ₃₂ CFL1.0	32 ³	1.0	0.685	0.845	14 %	

All the chosen network configurations were tested a posteriori. The feature and labels vectors are the same for all M1 models (for M3 models all $\mathbf{x} - \Delta \mathbf{x}$ entries are omitted) and are given below:

$$\mathbf{x} = \underbrace{\begin{bmatrix} \left(\overline{\varphi}_q, \frac{\partial \overline{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x}, t) \\ \left(\overline{\varphi}_q, \nabla \cdot (\overline{\mathbf{u}} \otimes \overline{\mathbf{u}})\right)_e(\mathbf{x}, t) \\ \left(\overline{\psi}_q, \nabla \cdot \overline{\mathbf{u}}\right)_e(\mathbf{x}, t) \\ \vdots \\ \left(\overline{\varphi}_q, \frac{\partial \overline{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x} - \Delta \mathbf{x}, t) \\ \vdots \\ \left(\overline{\varphi}_q, \frac{\partial \overline{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x}, t - \Delta t) \\ \vdots \\ \left(\overline{\varphi}_q, \frac{\partial \overline{\mathbf{u}}}{\partial t}\right)_e(\mathbf{x} - \Delta \mathbf{x}, t - \Delta t) \\ \vdots \\ \left(\overline{\psi}_q, \nabla \cdot \overline{\mathbf{u}}\right)_e(\mathbf{x} - \Delta \mathbf{x}, t - \Delta t) \\ z \end{bmatrix}}_{29 \text{ features}}$$

$$\tilde{\mathbf{y}} = \underbrace{\begin{bmatrix} \left(\overline{\varphi}_q, \mathcal{L}_{M'}\right)_e(\mathbf{x}, t + \Delta t) \\ \left(\overline{\psi}_q, \mathcal{L}_C(\mathbf{u}')\right)_e(\mathbf{x}, t + \Delta t) \end{bmatrix}}_{4 \text{ labels}}$$

5

Application of data-driven closure to the complete Navier-Stokes equations

In this chapter, the first attempt at large-eddy simulation with data-driven modelling of the complete Navier-Stokes closure is presented. As was briefly mentioned previously in section 4.2, despite the ANN models achieving good a priori correlations for the continuity closure term, a large relative error is present when considering the averages of model predictions over many examples. While deploying the model into the solver, this led to issues in the corrector pass convergence, which are discussed in more detail below. The impact of machine learning continuity closure predictions on the solver convergence is explained, and an equation for resolved pressure is derived, illustrating close coupling between ANN closure and the resolved pressure term.

For all the large-eddy simulations discussed in this and consequent chapters, a finite-difference Jacobian matrix was used for the corrector pass loop. This Jacobian matrix was also augmented with the terms from the algebraic model (see Akkerman et al [28]) for the continuity and pressure entries, in order to assist with corrector pass convergence.

Figure 5.1 shows the convergence history for the corrector pass loop at the first LES time step, when using the model $M1_{16}$ with a complete machine-learning closure, i.e., one closure term for each momentum component and one for continuity, four in total. While the residuals for the velocity components u, v, w are steadily decreasing, the pressure residual is stalled. The corrector pass procedure is ineffective at improving the accuracy of the pressure field. This is likely related to the poor ability of machine learning model to predict the continuity closure term $(\overline{\psi}_q, \mathcal{L}_C')_e$ on average, as can be seen in the top left corner of the continuity prediction in figure 4.8.

This issue can be alleviated by replacing the continuity closure prediction with either an algebraic model prediction or an exact continuity closure term, if available. The corrector pass convergence plot is shown for the latter, i.e., exact continuity closure, in figure 5.2. Now the corrector pass loop can reach a sufficient tolerance and displays a linear curve, even though many iterations

are still needed (≈ 100). For comparison, the corrector pass convergence history is shown in figure 5.3 for a simulation with exact closure terms obtained from a DNS projection. Also then, a similar number of corrector pass iterations are needed, although the v residual is decreasing much more rapidly.

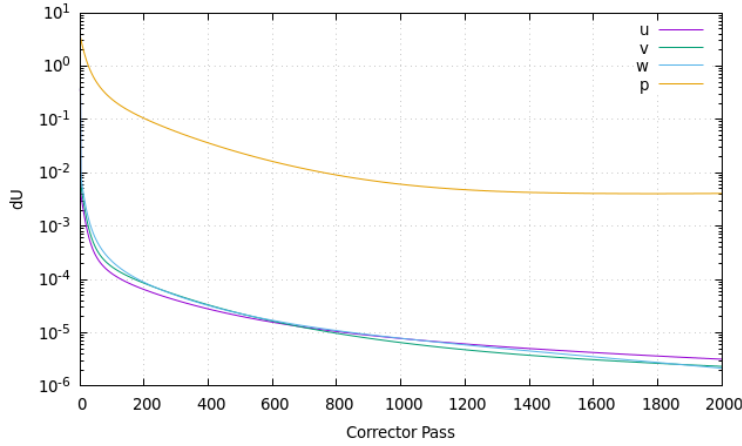


Figure 5.1: Corrector pass convergence plot, using continuity closure provided by ANN model.

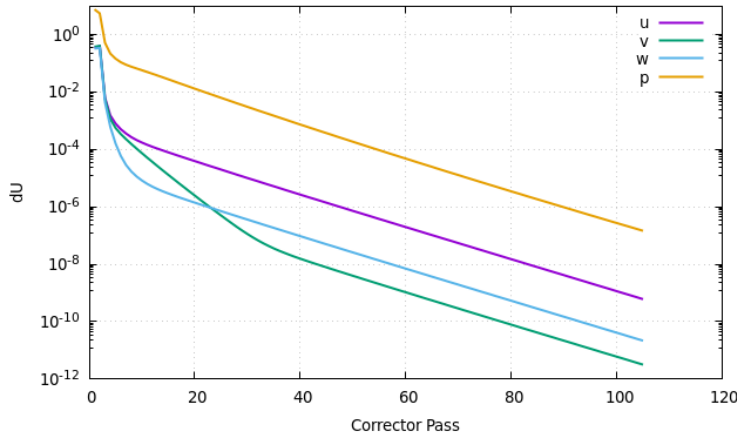


Figure 5.2: Corrector pass convergence plot, using exact continuity closure with ML momentum closure terms.

However, even when replacing the ML continuity closure term prediction by an exact or algebraic model alternative, large errors were observed in the LES pressure field, immediately after the first time step. This indicates that also machine learning predictions for the momentum terms lead to a spurious pressure solution field. However, the associated velocity field is not strongly affected. The spurious pressure field is compared to the proper pressure solution obtained from DNS projection in figure 5.4. As can be seen, the magnitude of the pressure is completely off, already after the first LES time step. In addition, the pressure field is modulated by high frequency pressure fluctuations.

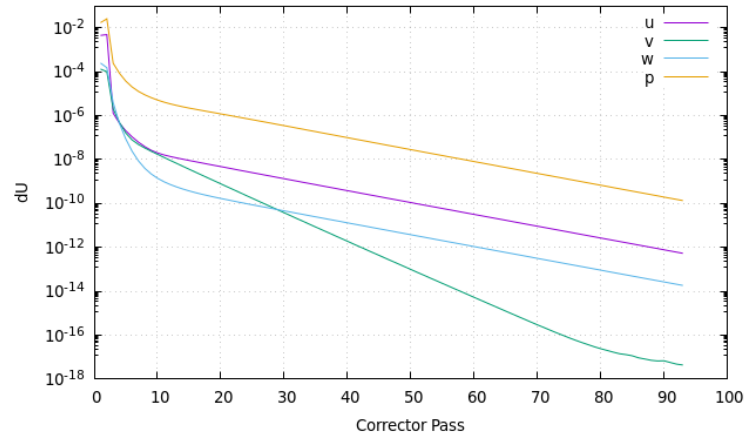


Figure 5.3: Corrector pass convergence plot, using exact closure terms.

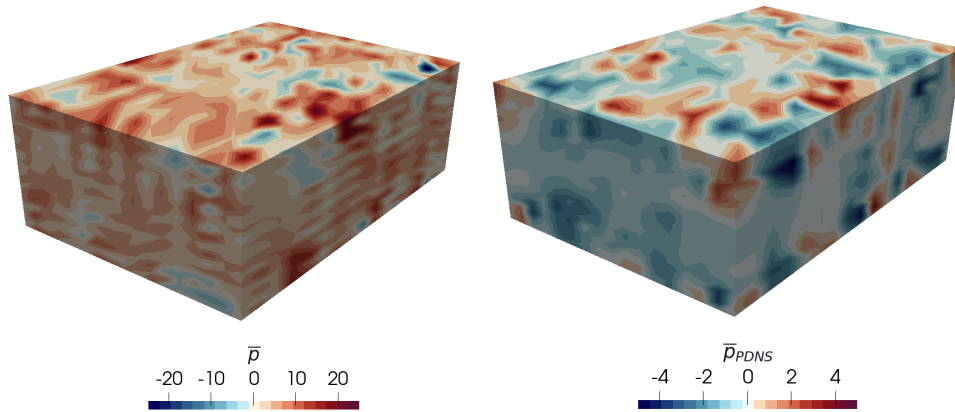


Figure 5.4: Pressure oscillations captured for model $M1_{16}$ (ML momentum closure, exact continuity closure) on the left, compared to projected DNS solution on the right. First LES time step.

Pressure Poisson equation

To further elucidate the link between momentum and continuity closure terms and a pressure field, consider equation 2.2:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla p - \nu \Delta \mathbf{u} = \mathbf{f}, \quad \mathbf{x} \in \Omega$$

Applying the divergence operator on both sides:

$$\frac{\partial (\nabla \cdot \mathbf{u})}{\partial t} + \nabla \cdot (\nabla \cdot (\mathbf{u} \otimes \mathbf{u})) + \Delta p - \nu \nabla \cdot (\Delta \mathbf{u}) = \nabla \cdot \mathbf{f}$$

As the forcing field is constant, it is divergence free. Using the vector identity $\Delta \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times (\nabla \times \mathbf{u})$, and taking advantage of the fact that divergence of a curl is zero, leads to $\nabla \cdot (\Delta \mathbf{u}) = \nabla \cdot (\nabla(\nabla \cdot \mathbf{u}))$.

Then:

$$\frac{\partial (\nabla \cdot \mathbf{u})}{\partial t} + \nabla \cdot (\nabla \cdot (\mathbf{u} \otimes \mathbf{u})) + \Delta p - \nu \nabla \cdot (\nabla(\nabla \cdot \mathbf{u})) = 0$$

Rearranging:

$$\Delta p = -\nabla \cdot (\nabla \cdot (\mathbf{u} \otimes \mathbf{u})) - \left(\frac{\partial}{\partial t} - \nu \nabla \cdot \nabla \right) (\nabla \cdot \mathbf{u}), \quad \mathbf{x} \in \Omega \quad (5.1)$$

Introducing the scale separation: $\mathbf{u} = \bar{\mathbf{u}} + \mathbf{u}'$, $p = \bar{p} + p'$ and isolating $\Delta \bar{p}$ on the left-hand side:

$$\begin{aligned} \Delta \bar{p} = & \left(-\frac{\partial}{\partial t} + \nu \nabla \cdot \nabla \right) (\nabla \cdot \bar{\mathbf{u}}) + \left(-\frac{\partial}{\partial t} + \nu \nabla \cdot \nabla \right) (\nabla \cdot \mathbf{u}') \\ & - \nabla \cdot (\nabla p' + \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) + \nabla \cdot (\mathbf{u}' \otimes \mathbf{u}') + \nabla \cdot (\bar{\mathbf{u}} \otimes \mathbf{u}') + \nabla \cdot (\mathbf{u}' \otimes \bar{\mathbf{u}})) \end{aligned} \quad (5.2)$$

If \mathbf{u}' term is exact, by equation 2.1 the first two terms will sum to zero ($\nabla \cdot \mathbf{u} = \nabla \cdot \bar{\mathbf{u}} + \nabla \cdot \mathbf{u}' = 0$). More specifically, if the machine learning approximation for the continuity closure term \mathcal{L}_C is replaced by its exact counterpart, then the resolved pressure is only influenced by unresolved pressure and unresolved convective terms, which all are part of the momentum closure $\mathcal{L}_{\mathcal{M}'}$:

$$\Delta \bar{p} = -\nabla \cdot (\nabla p' + \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) + \nabla \cdot (\mathbf{u}' \otimes \mathbf{u}') + \nabla \cdot (\bar{\mathbf{u}} \otimes \mathbf{u}') + \nabla \cdot (\mathbf{u}' \otimes \bar{\mathbf{u}})) \quad (5.3)$$

Using equation 2.25:

$$\Delta \bar{p} = -\nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) - \nabla \cdot (\mathcal{L}_{\mathcal{M}'}(\mathbf{u}', \bar{\mathbf{u}}, p')) + \nabla \cdot \left(\frac{\partial \mathbf{u}'}{\partial t} \right) - \nu \Delta \mathbf{u}'$$

$$\boxed{\Delta \bar{p} = -\nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) - \nabla \cdot (\mathcal{L}_{\mathcal{M}'}(\mathbf{u}', \bar{\mathbf{u}}, p')_{\text{ANN}}) + \left(\frac{\partial}{\partial t} - \nu \nabla \cdot \nabla \right) (\mathcal{L}_C(\mathbf{u}')_{\text{exact}})} \quad (5.4)$$

Once again, equation 5.4 is only valid if the exact continuity closure term is used. This equation shows that pressure is directly influenced by ANN predictions in the momentum terms. This contrasts with the standard momentum equations, i.e., first equation of the system in equation 2.27, where the ANN predictions instead influence the time derivative term $\partial \bar{\mathbf{u}} / \partial t$. Thus, for a velocity field, starting from a projected DNS initial condition, several time steps are needed to deviate from the evolving projected DNS solution due to errors induced by ANN momentum closure terms predictions. In a solution to a pressure Poisson equation 5.4, a local error deviation will be propagated globally and instantaneously, and the projected DNS initial condition for pressure has no influence on the pressure solution in the following time steps.

To limit the effects of erroneous pressure fields on the velocity field, in all the large-eddy simulations considered hereafter, the exact pressure solution is injected from projected DNS into the LES system. This way, complete ANN closure term predictions can be used, and the effect of errors in the closure terms is only limited to the velocity field. The corrector pass history for the first time step of the pressure-constrained system is given in figure 6.1.

Remarkably, the previously discussed continuity closures and the resulting pressure field have a very profound effect on the number of corrector pass iterations with the adopted Jacobian matrix. While running with a full ANN closure, even 2000 corrector passes only lead to pressure residual of $O(10^{-2})$, see figure 5.1. If the exact closure for continuity is inserted instead, in just

over 100 corrector passes, a pressure residual of $\mathcal{O}(10^{-6})$ can be achieved, see figure 5.2. With the exact pressure obtained from DNS projection, velocity residuals of $\mathcal{O}(10^{-12})$ can be reached with less than 10 corrector passes, see figure 6.1.

6

Evaluation of stabilization techniques and discretization effects

In this chapter, the results from LES simulations with ANN model closure are shown, including a description of post-processing and a discussion on solution stability. The ANN model performance is compared against algebraic model as introduced by Akkerman et al [28]. In section 6.1, the effect of stabilization techniques, such as data augmentation, closure terms filtering and mini-batch sum loss, is discussed. Next, in section 6.2, the impact of varying the CFL number, mesh refinement and ANN stencil on solution stability is presented.

As was mentioned previously, in all the LES simulations considered in this and the next chapter, pressure solution was inserted from the DNS projection, while the momentum equations are solved. This led to acceleration of the corrector pass loop, shown in figure 6.1, and lower simulation times.

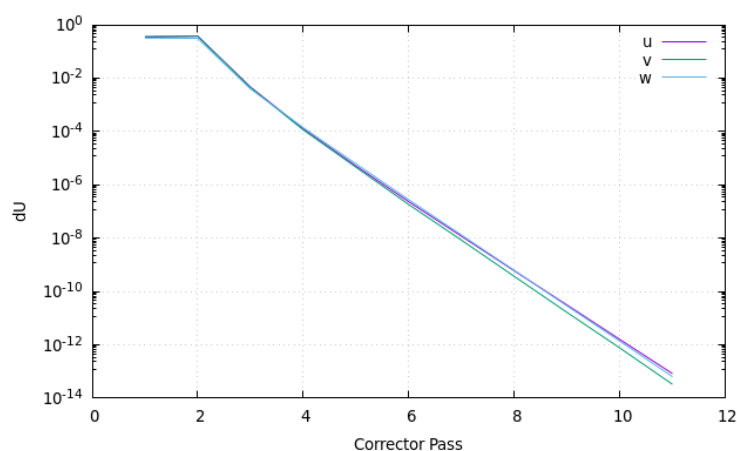


Figure 6.1: Corrector pass convergence plot, constraining pressure to projected DNS pressure

To quantify the stability of the flow solution and the model error introduced through the closure terms, three metrics are used. Firstly, specific resolved instantaneous energy in the flow is computed as:

$$k = \frac{1}{2V(\Omega)} \int_{\Omega} \bar{\mathbf{u}} \cdot \bar{\mathbf{u}} d\Omega \quad (6.1)$$

A numerical flow solution is said to be *stable* if the value of k in the domain is bounded, or $\partial k / \partial t \leq 0$. This is a necessary condition to allow for longer simulation times but doesn't guarantee that the solution is physical.

Secondly, as an indirect metric of closure model error, the root mean square error between streamwise velocities of LES and PDNS is computed as:

$$\text{RMS}(\bar{u}_{\text{LES}}, \bar{u}_{\text{PDNS}}) = \sqrt{\frac{\sum_N (\bar{u}_{\text{LES}} - \bar{u}_{\text{PDNS}})^2}{Nu_{\tau}^2}}, \quad (6.2)$$

with N being the number of elements in the mesh. Lastly, the Pearson correlation coefficient (defined in equation 4.4) is used to directly compare the true closure term for streamwise momentum $\mathcal{L}_{M_x, \text{exact}}$ with the ANN approximation \mathcal{L}_{M_x} . This term specifically is picked as the mean flow is oriented in streamwise direction. It should be noted that both of these last two metrics will grow in time, due to the chaotic nature of the turbulence. Thus, a zero/low correlation coefficient doesn't necessarily mean that the closure model is inaccurate, as it can also be due to the flow solution developing in a different, but still physical manner when comparing to the DNS projection. However, comparing the correlation coefficients and RMS errors across different closure models for the same simulation setup with the same initial condition can provide some insights into the effect of model parameters, especially immediately after the simulation start. A more advanced discussion with the flow statistics computed through spatial averaging is presented in chapter 7 for a few selected models.

6.1. Effect of stabilization techniques

The ANN closure models from table 4.4 are first tested with an initial condition at $t = 0.0$, corresponding to the beginning of the validation dataset, see figure 4.3 for reference. They are immediately exposed to an unseen flow solution. Additionally, model M1₁₆ is also tested starting from the initial condition at $t = 1.0$, corresponding to the beginning of training dataset. This way, generalization ability of the model can be evaluated.

Figure 6.2 shows the a posteriori performance, quantified by the metrics mentioned above, of these models on the 16³ element LES mesh at CFL = 0.5. It should be noted that the correlation evolution, shown on the bottom plot of figure 6.2, is expected to approximate a power law such as:

$$\rho(t) = \rho_{\text{a priori}}^{t/\Delta t} \quad (6.3)$$

This is because at such coarse resolution, the closure terms directly influence the resolved scales used as input features for the machine learning model. Thus, the reader should not be alarmed by

the decrease in the correlation coefficient.

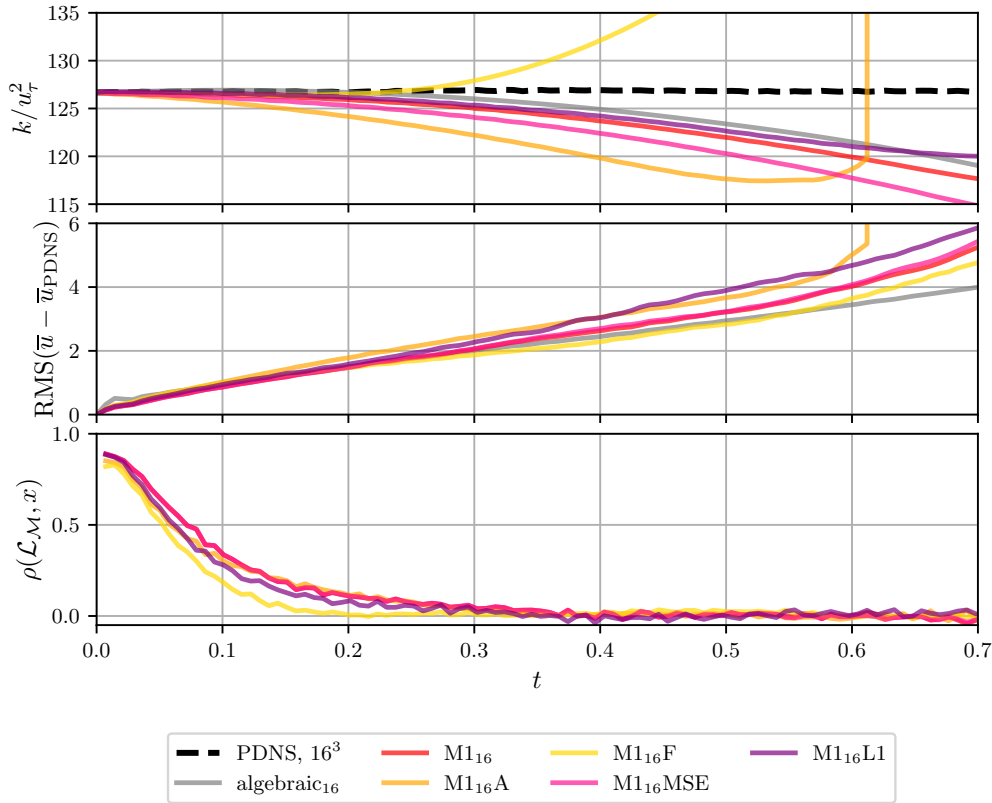


Figure 6.2: A posteriori results for variations of model M_{16} , with constrained pressure. Top: resolved specific kinetic energy. Middle: root mean square error (LES vs. PDNS) in resolved streamwise velocity. Bottom: correlation coefficient of the streamwise momentum closure prediction of the LES and PDNS.

Notably, there are no stark differences between the flow statistics of model M_{16} when initialized from $t = 0.0$ or $t = 1.0$, leading to a conclusion that the network can generalize beyond its training data to unseen turbulent channel flow. Expanding the training dataset by simply using more snapshots of DNS projections is not likely to improve the network performance.

Examining the performance of models M_{16} and M_{16A} , shows that the data augmentation does not lead to an improved flow solution. It is likely that augmenting the training dataset will only show improvement in the validation test case if the mean flow direction is misaligned with the coordinate axes, which was not the case currently.

Looking at the models M_{16} and M_{16F} , it is clear that enabling filtering leads to different kinetic energy evolution compared to other cases, hinting that the flow evolves in a different manner. In the beginning, just like all the other models, $\partial k/\partial t < 0$, but at $t = 0.2$ the slope changes and excessive kinetic energy is gained. Interestingly, the RMS error of the M_{16F} model is lower than all other models when considering later development of the flow, meaning that excessive kinetic energy gain (i.e., solution instability) does not directly manifest itself in the RMS error evolution.

Neither M_{16F} nor M_{16A} provide a better performance as is, when comparing to all the other models, as they both fail to satisfy the stability constraint. Comparing model M_{16} and M_{16} MSE

(with and without mini-batch sum loss respectively) shows no effect on the correlation or RMS error evolution, but a difference can be observed in flow kinetic energy evolution, where inclusion of mini-batch sum loss leads to a slightly better match with projected DNS flow kinetic energy. Even though the improvement is marginal, it leads to an important conclusion that training with additional metrics such as mini-batch sum loss during ANN training can be used to affect the a posteriori stability of the closure model.

6.2. Effect of mesh refinement, CFL number and input stencil

In this section, ANN models with varying input stencils (M_{16} vs. M_{32}), different mesh refinement (16^3 vs. 32^3) and CFL numbers (0.5 vs 1.0) are compared in terms of the previously mentioned three metrics. The results are shown in figure 6.3.

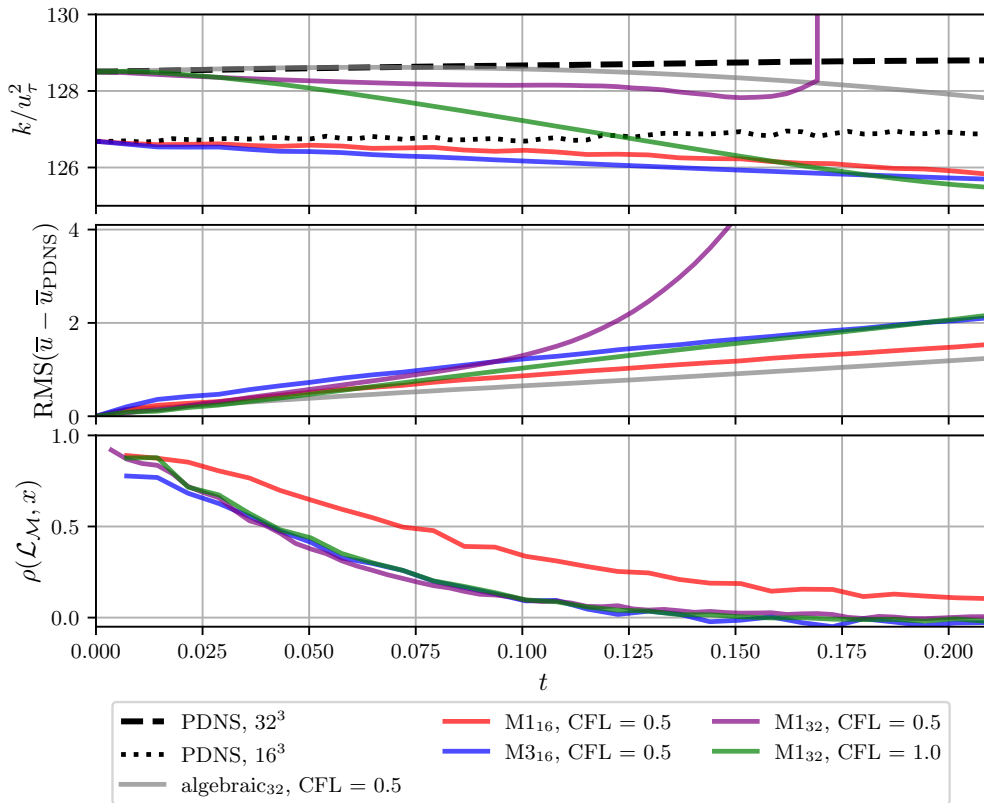


Figure 6.3: A posteriori results for ANN models with varying stencil, CFL number, mesh refinement, with constrained pressure. Top: resolved specific kinetic energy. Middle: root mean square error (LES vs. PDNS) in resolved streamwise velocity. Bottom: correlation coefficient of the streamwise momentum closure prediction of the LES and PDNS.

Surprisingly, the coarse resolution model M_{16} performs the best across all the three metrics. When comparing input stencil 1 to 3 (see figure 4.2), a considerable drop in correlation is observed, as well a greater RMS error and kinetic energy deviation. Nonetheless, the two models show very similar trends. The most surprising result is that increasing the resolution of the mesh does not have a clear benefit on stability or accuracy of the model. Comparing model M_{16} and M_{32} , M_{16} maintains higher correlations for considerably longer, and the M_{32} RMS error curve transitions

into an exponential already at $t > 0.10$, signalling divergence. This model instability can also be aggravated by the fact that the $M1_{32}$ model handles many more closure terms than the $M1_{16}$ model, and there could be more opportunities for significant errors. Lastly, increasing the CFL number at mesh resolution of 32^3 doesn't show a strong effect on correlations, but strongly degrades the ability of the model to maintain the correct kinetic energy already after several time steps. It should be noted that the models have a varying LES time step: models $M1_{16}$, $M3_{16}$, $M1_{32}$ (CFL = 1.0) are all running with $\Delta t = 0.0072$, while model $M1_{32}$ (CFL = 0.5) is running at $\Delta t = 0.0036$. Thus, over the same time period this model experiences twice as many ANN inputs, and has more opportunities for significant error, which can explain why the RMS metric of model $M1_{32}$ (CFL = 0.5) exceeds that of $M1_{32}$ (CFL = 1.0). To summarize, coarse mesh size and low CFL number seem to be stabilizing factors for ANN closure models.

7

Results

In this chapter, a closer view is presented of the machine learning closure model error and its impact on the flow solution for models M1₁₆, M1₁₆F and M1₃₂. In section 7.1, new quantities necessary for a detailed discussion of simulation stability are introduced. Afterward, in section 7.2, evolution of kinetic energy for model M1₁₆ is examined in more detail. In section 7.3 the effect of Fourier filtering is discussed in depth through comparison of models M1₁₆ and M1₁₆F. Lastly, in section 7.4 the flow evolution for model M1₃₂ is discussed.

7.1. Instantaneous kinetic flow energy evolution

In this section, a more detailed investigation is carried out into the evolution of the flow subject to machine learning closure models. Namely, instead of using only specific resolved instantaneous kinetic energy k as the metric, is split up into its mean and fluctuating components. First, the mean resolved velocity is defined as spanwise and streamwise average of resolved velocity field, viz:

$$\text{ave}_{x,z}(\bar{\mathbf{u}}) = \frac{1}{N_x \cdot N_z} \sum_{N_x \cdot N_z} \bar{\mathbf{u}} \quad (7.1)$$

Next, resolved fluctuating velocity is given as:

$$\bar{\mathbf{u}}'' = \bar{\mathbf{u}} - \text{ave}_{x,z}(\bar{\mathbf{u}}) \quad (7.2)$$

Then, the resolved Reynolds stress tensor \bar{R}_{ij} can be defined as follows:

$$\bar{R}_{ij} = \text{ave}_{x,z}(\bar{u}_i'' \bar{u}_j''), \quad (7.3)$$

with $\bar{u}_1'' = \bar{u}''$, $\bar{u}_2'' = \bar{v}''$ and $\bar{u}_3'' = \bar{w}''$.

With these new quantities introduced, the resolved kinetic energy can be split into various parts. It is composed of turbulent specific resolved kinetic energy $\frac{1}{2}\bar{R}_{ii}$ and (spatial) mean specific

resolved kinetic energy e , defined as follows:

$$e = \frac{1}{2V(\Omega)} \int_{\Omega} \text{ave}_{x,z}(\bar{\mathbf{u}}) \cdot \text{ave}_{x,z}(\bar{\mathbf{u}}) d\Omega \quad (7.4)$$

In a projected DNS solution, no mean (average) flow is expected in wall-normal or spanwise directions, i.e., $\text{ave}_{x,z}(\bar{v}) = 0$, $\text{ave}_{x,z}(\bar{w}) = 0$. Even though LES runs with ANN closure can introduce slight deviations, they are negligible compared to the streamwise average flow $\text{ave}_{x,z}(\bar{u})$. Equation 7.5 shows that the instantaneous kinetic flow energy is composed of 4 contributions.

$$k = e + \frac{1}{2}\bar{R}_{ii} \approx \text{ave}_{x,z}(\bar{u})^2 + \frac{1}{2}\bar{R}_{11} + \frac{1}{2}\bar{R}_{22} + \frac{1}{2}\bar{R}_{33} \quad (7.5)$$

An evolution equation for resolved instantaneous flow kinetic energy can be found by replacing the weighting function \bar{w} with $\bar{\mathbf{u}}$ in equation 2.27:

$$(\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}}(\bar{\mathbf{u}}, \bar{p}))_{\Omega} = (\bar{\mathbf{u}}, \mathbf{f})_{\Omega} - (\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_{\Omega} \quad (7.6)$$

Expanding the momentum operator $\mathcal{L}_{\mathcal{M}}$:

$$\frac{\partial (\bar{\mathbf{u}} \cdot \bar{\mathbf{u}})_{\Omega}}{\partial t} + (\bar{\mathbf{u}} \cdot \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}))_{\Omega} + (\bar{\mathbf{u}} \cdot \nabla \bar{p})_{\Omega} - (\bar{\mathbf{u}} \cdot \nu \Delta \bar{\mathbf{u}})_{\Omega} = (\bar{\mathbf{u}}, \mathbf{f})_{\Omega} - (\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_{\Omega} \quad (7.7)$$

Considering that the forcing term \mathbf{f} is constant, it can be taken outside the integral. This leads to:

$$2 \frac{\partial k}{\partial t} = f_x (\bar{u})_{\Omega} - (\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_{\Omega} - (\bar{\mathbf{u}} \cdot \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}))_{\Omega} + (\bar{\mathbf{u}} \cdot \nu \Delta \bar{\mathbf{u}})_{\Omega} - (\bar{\mathbf{u}} \cdot \nabla \bar{p})_{\Omega} \quad (7.8)$$

Dividing by twice the domain volume $2V(\Omega)$:

$$\boxed{\frac{\partial k}{\partial t} = \frac{f_x U_{\text{bulk}}}{2} - \frac{(\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_{\Omega}}{2V(\Omega)} - \frac{(\bar{\mathbf{u}} \cdot \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}))_{\Omega}}{2V(\Omega)} + \frac{(\bar{\mathbf{u}} \cdot \nu \Delta \bar{\mathbf{u}})_{\Omega}}{2V(\Omega)} - \frac{(\bar{\mathbf{u}} \cdot \nabla \bar{p})_{\Omega}}{2V(\Omega)}} \quad (7.9)$$

With U_{bulk} being the bulk flow velocity. Equation 7.9 explicitly shows the effect of momentum closure term on the evolution of instantaneous specific kinetic energy. Considering that the ANN models have shown a tendency to (on average) overestimate the closure term $\mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p')$, as is shown in table 4.4 with positive values of $\epsilon(E_{\mathcal{M}_x})$, this leads to an initial decay of k for all the networks. Thus, $\epsilon(E_{\mathcal{M}_x}) \geq 0$ can be considered the a priori approximation for the stability constraint $\partial k / \partial t \leq 0$. To be more precise,

$$\boxed{(\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_{\text{pred}} \geq (\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_{\text{truth}}} \quad (7.10)$$

is the condition that should be enforced in the next iterations of the model to guarantee a posteriori stability of k .

7.2. Flow evolution for the model M1₁₆

The components of k described in equation 7.5 are computed for the LES simulation with closure model M1₁₆ and are shown in figure 7.1. The top subplot is showing the instantaneous specific kinetic energy, the second from top is showing the evolution of the largest turbulent contribution $\frac{1}{2}\overline{R}_{11}$, the third is showing less significant spanwise and wall-normal contributions, and the last plot on the bottom is showing the mean flow kinetic energy.

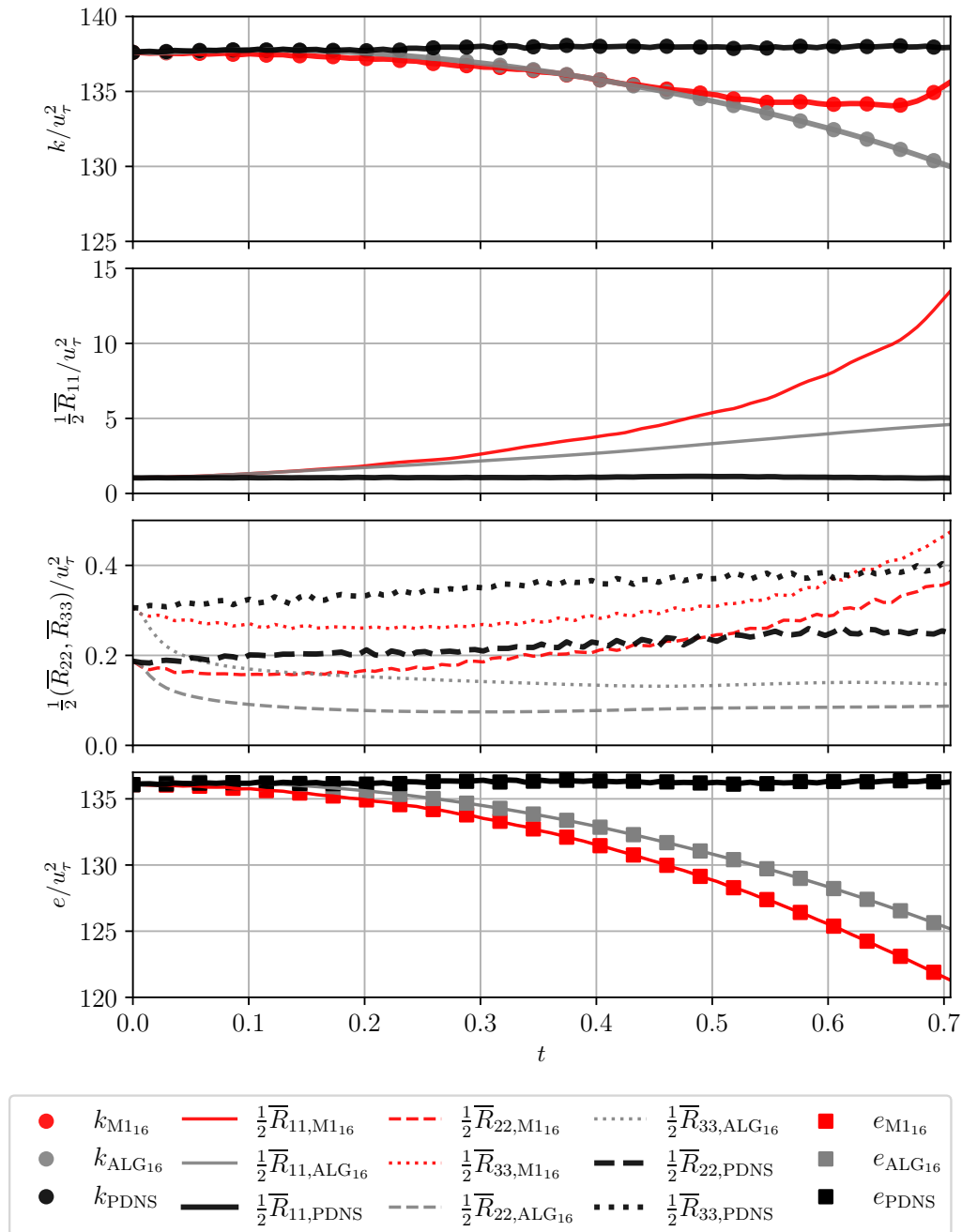


Figure 7.1: Evolution of mean and turbulent kinetic energy for model M1₁₆ (red), compared against algebraic model (gray).

Please note that the initial computation of k , presented in chapter 6, was carried out during runtime with higher number of integration points. The values of k and its components presented in this chapter have been computed via a post-processing routine, where only the element-averaged solution has been saved, leading to slightly different values of k , when comparing to figures 6.2 and 6.3.

There are three important parts in the flow system: mean flow, whose energy is e , resolved turbulent flow, whose energy is mostly $\frac{1}{2}\bar{R}_{11}$, and unresolved turbulence (fine-scale), whose energy is unknown, but which takes energy from the resolved flow via the modelled term $(\bar{\mathbf{u}}, \mathcal{L}_{\mathcal{M}'}(\bar{\mathbf{u}}, \mathbf{u}', p'))_{\Omega}$

Evidently, the closure term has a strong influence on the mean flow, immediately draining energy from it. At the same time, turbulent resolved energy components \bar{R}_{22} and \bar{R}_{33} show a good match with PDNS values, but the largest component \bar{R}_{11} grows linearly until $t = 0.3$, then grows exponentially beyond that point, completely deviating from projected DNS solution. The strong effect of the closure model on the mean flow can be explained by an extremely coarse resolution of the mesh, meaning that there is no sufficient separation between the mean flow, large resolved scales and modelled fine scales. The fact that the algebraic model shows a poor match with the projected DNS solution is an indication that this model also cannot handle the coarse 16^3 mesh.

Figure 7.2 shows streamwise energy spectra of the solution at $t = 0.11$, at varying wall distance. Please note that this and the following spectra are computed based on the resolved fluctuating velocity \mathbf{u}'' .

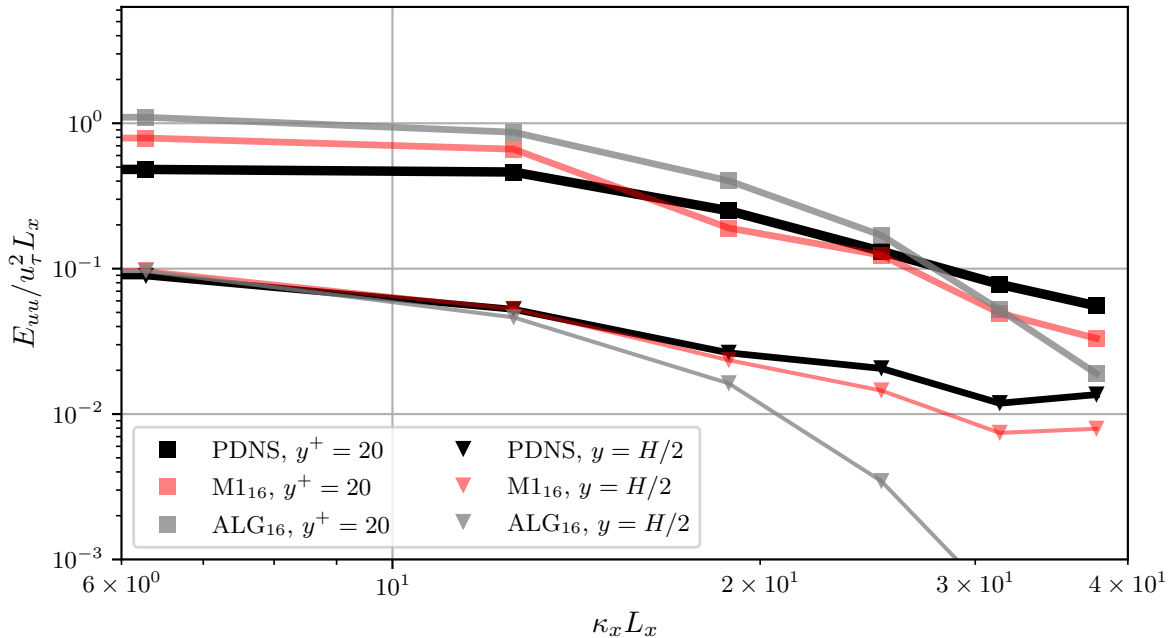


Figure 7.2: Stream wise resolved fluctuation \bar{u}'' energy spectra at varying wall distance at $t = 15\Delta t = 0.11$ for closure model M1₁₆ (red) and algebraic closure (gray), both with constrained pressure.

In general, the ANN model matches reasonably with the DNS projection, and fine resolved scales are found below the DNS projection, as would be expected given an excessively dissipative model

(due to a larger average magnitude of closure predictions). However, the problem lies in the largest resolved scales, in particular in the first element near the wall ($y^+ \approx 20$). These seem to gain energy, meanwhile the mean flow is losing energy (figure 7.1, bottom plot, $t = 0.11$).

To further examine the influence of the wall-normal coordinate, the streamwise mean velocity profile and resolved Reynolds stress components \bar{R}_{11} , \bar{R}_{12} are shown in figure 7.3 at $t = 0.10$. Both algebraic and M1₁₆ models show a good match for the mean flow profile and for the most of the \bar{R}_{11} profile. However, \bar{R}_{11} is overestimated strongly at the first element at $y^+ \approx 20$, although M1₁₆ model is more accurate. Similarly, a better match is found with the M1₁₆ model for the \bar{R}_{12} profile, but still a strong error is present in the first element near the wall.

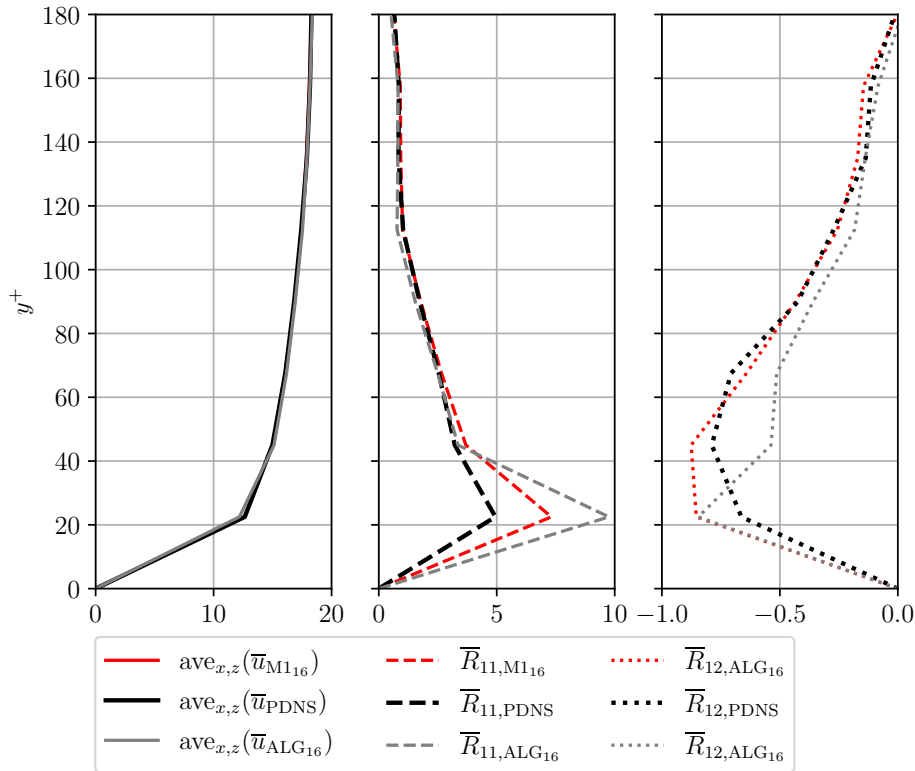


Figure 7.3: Mean flow and resolved components of Reynolds stresses at $t = 14\Delta t = 0.10$ for model M1₁₆ (red), compared against algebraic model (gray).

The model M1₁₆ immediately leads to decrease of mean flow kinetic energy and increase of large resolved turbulent energy when compared to projected DNS. This is likely an important contributing factor in the fast decrease of accuracy of closure predictions, as the network has only been exposed to projected DNS snapshots. It is thus likely to be strongly dependent on the magnitudes of the largest resolved turbulent flow scales. These evolved large resolved scales in the present large-eddy simulation are drastically different from the large resolved scales of DNS projection, due to the increase in resolved turbulent kinetic energy.

The largest deviations from projected DNS solutions are observed in the first element at the wall, meaning that near-wall closure predictions need to be prioritized during network training. This can be achieved by using a loss function weighted by the wall-normal distance, with higher

weights given to elements near the wall.

7.3. Flow evolution for the model M1₁₆F

Next, the effect of filtering on the flow evolution and streamwise turbulent energy spectra is discussed by the means of comparing models M1₁₆ and M1₁₆F. Their streamwise energy spectra are compared in figure 7.4.

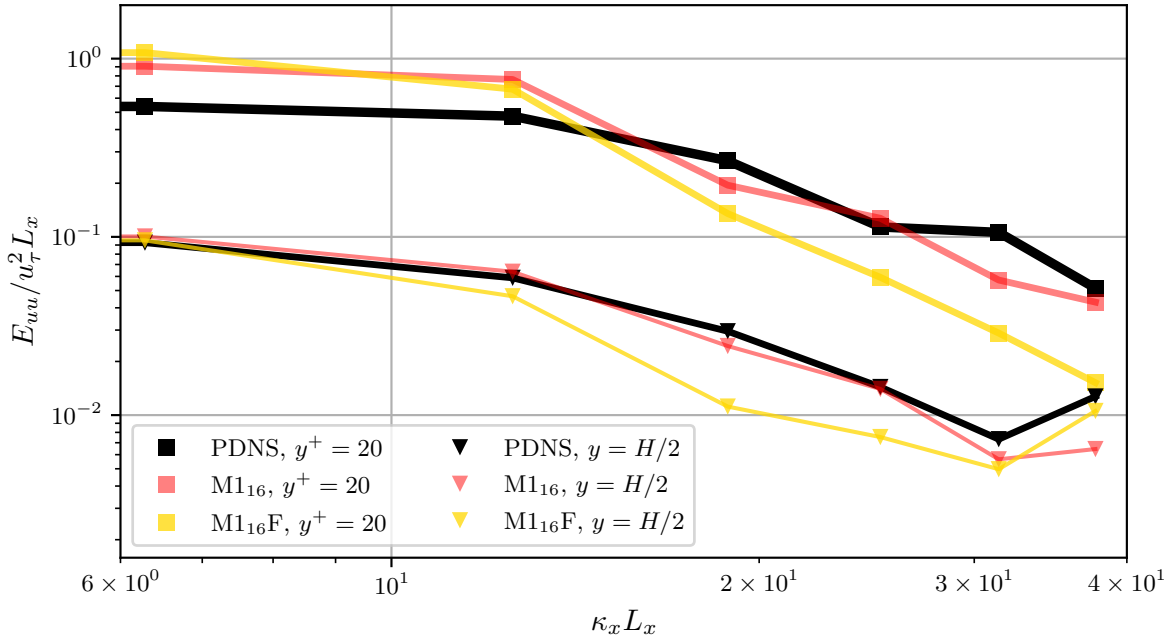


Figure 7.4: Stream wise resolved fluctuation \bar{u}'' energy spectra at varying wall distance at $t = 20\Delta t = 0.14$ for closure model M1₁₆ (red) and M1₁₆F (yellow) both with constrained pressure.

Clearly, the filtering of the training data has the expected influence on the energy spectra, even after several time steps. At the same time, a slight accumulation of energy can be observed at the end of the $y = H/2$ spectrum. The components of resolved instantaneous kinetic energy for the two models are compared in figure 7.5. For the filtered model, a good match with the projected DNS solution is observed until $t = 0.2$, after which the resolved Reynolds tensor components \bar{R}_{11} , \bar{R}_{22} , and \bar{R}_{33} start to diverge. It is unclear why the filtered model gains excessive energy in both the resolved turbulent energy $\frac{1}{2}\bar{R}_{ii}$ and mean-flow kinetic energy e . In conclusion, filtering of the training data can be used to modify the resulting spectra of the resolved flow, but other more critical stability issues need to be addressed first to make filtering a viable technique over longer run-times. Additionally, the current choice of the filter is likely not optimal and required further consideration.

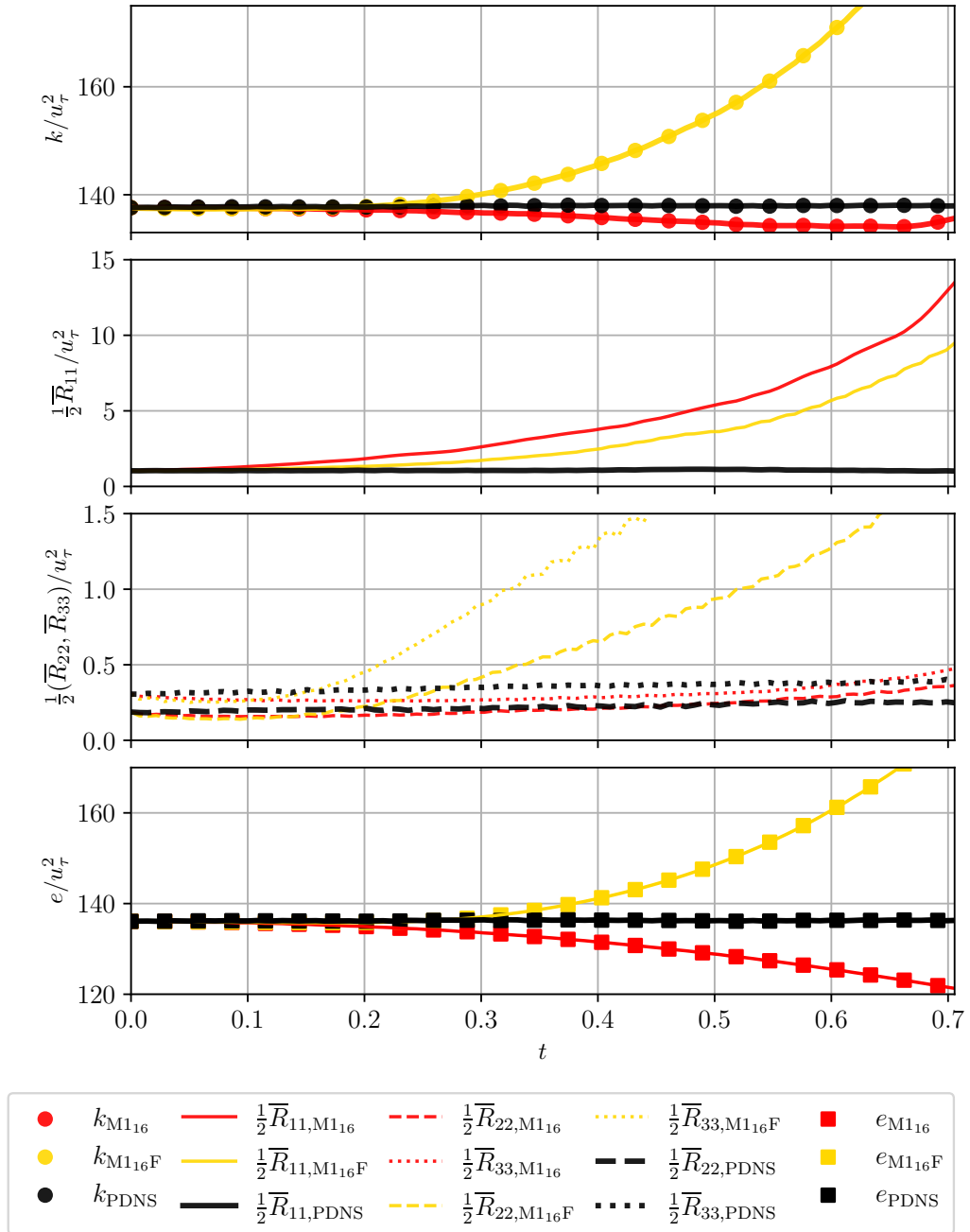


Figure 7.5: Evolution of mean and turbulent kinetic energy for model M1₁₆ (red), compared against filtered version of the model (orange).

7.4. Flow evolution for the model M1₃₂

In this section, the flow evolution of model M1₃₂ is described. Although a gain in stability and accuracy would be expected with a gain in mesh resolution, this has not been the case. Models used on a finer mesh led to solution divergence much sooner, both in terms of simulation time and number of LES time steps. Analyzing the resolved-flow kinetic energy evolution for model M1₃₂, shown in figure 7.6, and comparing it to model M1₁₆ in figure 7.1, leads to the similar conclusions, and the underlying stability issues are likely the same regardless of the mesh refinement.

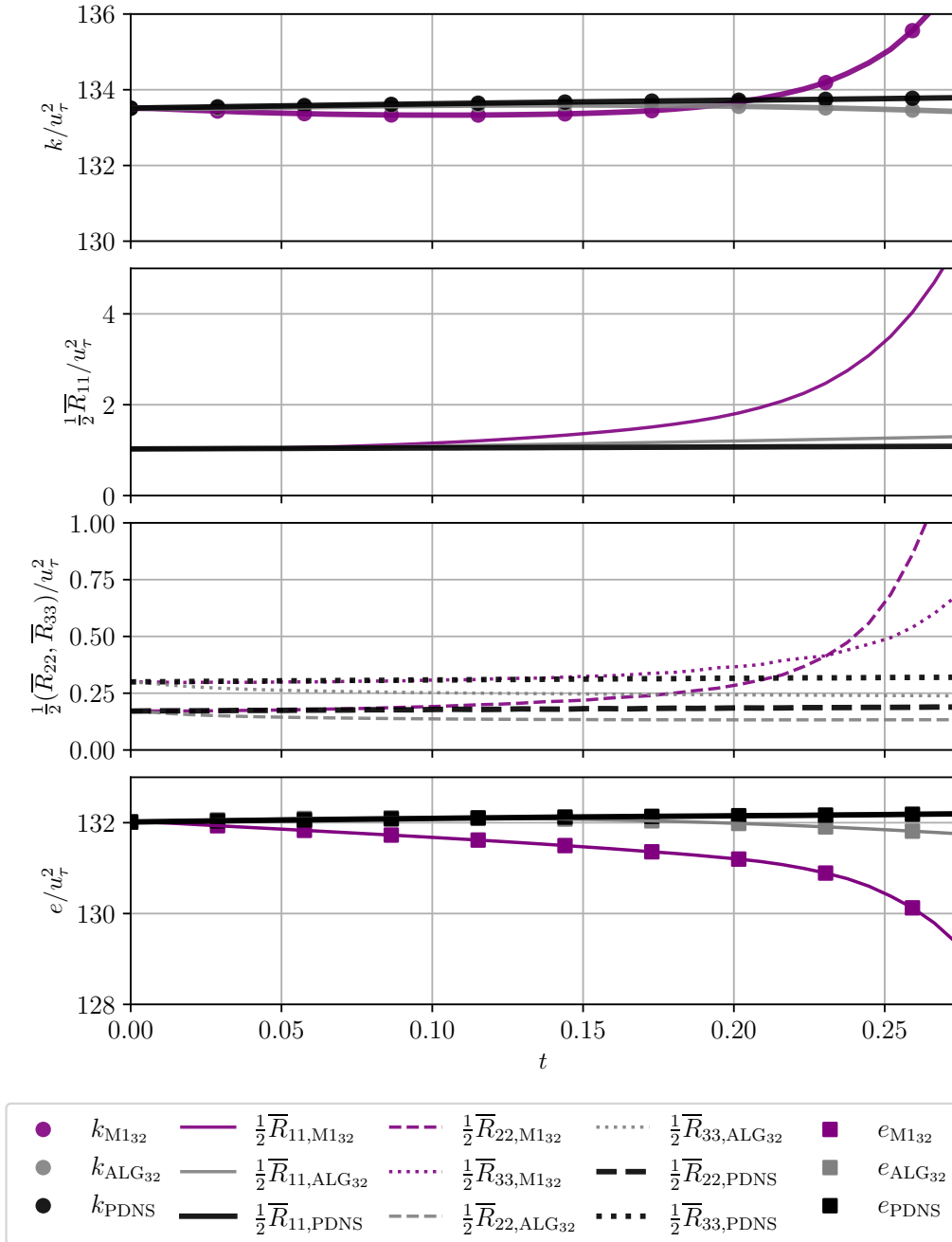


Figure 7.6: Evolution of mean and turbulent kinetic energy for model M1₃₂ (purple), compared against algebraic model (gray).

In both cases, quite soon after initialization, the resolved turbulent kinetic energy components \overline{R}_{ii} start to grow, while the mean-flow energy is decaying. At first, up until to $t = 0.20$ they deviate from the DNS projection linearly, but afterwards exponentially, which coincides with the complete loss of correlation accuracy. However, when observing the streamwise energy spectra, shown in figures 7.7 and 7.8 for the 32^3 case and in figure 7.2 for the 16^3 case, the effect of increased mesh refinement is seen. Initially, at $t = 3\Delta t$, the M1₃₂ model is excessively dissipative near the wall, but provides extra energy in the center of the channel. Afterward, the model begins to significantly drain energy from the mean flow, but instead of delivering it to the largest turbulent scales, following the idea of turbulent energy cascade, the smallest resolved scales are excessively energized. The cause of this is not clear.

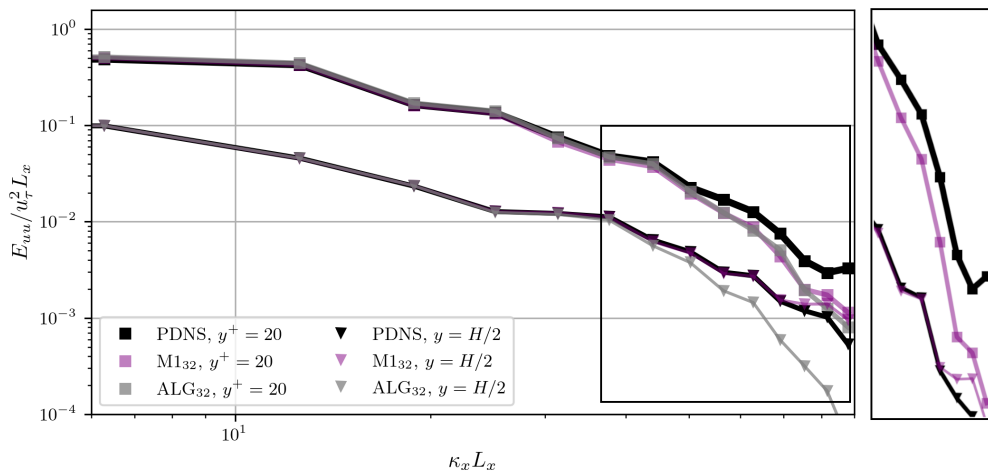


Figure 7.7: Stream wise resolved fluctuation $\overline{u''}$ energy spectra at varying wall distance at $t = 3\Delta t$ for closure model M1₃₂ (purple) and algebraic closure (gray), both with constrained pressure. High wave number region zoomed-in on the right, with algebraic spectrum omitted for visibility.

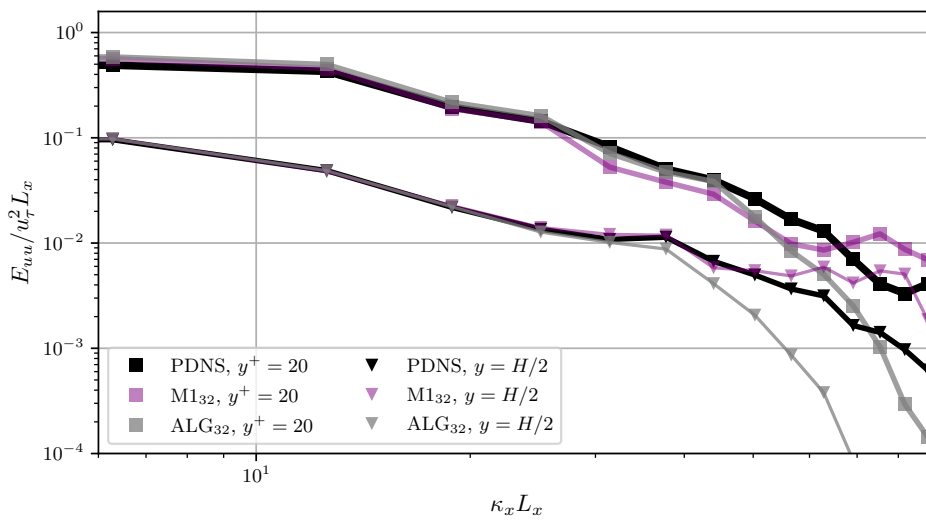


Figure 7.8: Stream wise resolved fluctuation $\overline{u''}$ energy spectra at varying wall distance at $t = 10\Delta t = 0.03$ for closure model M1₃₂ (purple) and algebraic closure (gray), both with constrained pressure.

The mean flow profiles and Reynolds stress components are shown in figure 7.9.

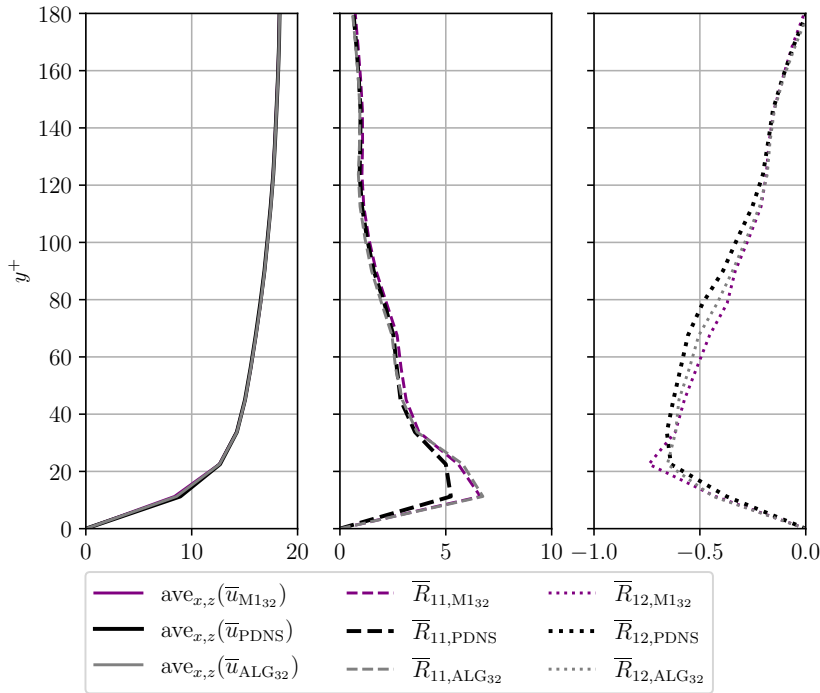


Figure 7.9: Mean flow and components of resolved Reynolds stress at $t = 14\Delta t = 0.05$ for model M1₃₂ (purple), compared against algebraic model (gray).

Again, deviations are strongest near the wall, particularly for \bar{R}_{11} . A much better match is observed for \bar{R}_{12} for the algebraic model, but not for model M1₃₂. It should also be noted that profiles shown for coarse case M1₁₆ in figure 7.3 are actually further in time from the initial condition than for M1₃₂, due to a larger time-step, again highlighting seemingly improved stability at the lower mesh refinement.

The cause for a more rapid breakdown of the flow with M1₃₂ model compared to M1₁₆ is likely the increased separation of turbulent modes in spectral space. Both models provide excessive energy to the turbulent fluctuations, but in the 16^3 case all the turbulent spectral modes gain a relatively equal amount of energy, while in the 32^3 case the smallest resolved modes are energized considerably more than the large resolved modes, which is more destabilizing to the flow solution as no energy cascade is observed. However, it is still not clear why a decrease in the mean flow energy in the 32^3 case does not lead to increased energy of largest turbulent modes. In order to further investigate this issue, a turbulent kinetic energy budget needs to be evaluated.

To conclude, in all the networks tested a posteriori, a significant and systematic deviation from the projected DNS solution is observed in the first element near the wall. This deviation leads to an increase in streamwise resolved turbulent energy \bar{R}_{11} near the wall, and may be linked to the drain of mean flow energy. While in the 16^3 case, this additional energy is transferred across all the resolved turbulent modes, in the 32^3 case it is mostly transferred to high frequency wavenumber modes. An improved accuracy in the ANN model predictions for near-wall elements is necessary to alleviate this issue. Filtering of the training dataset can be used to manipulate the resolved

energy spectra in simulations with ANN closure, but the effectiveness of this technique is still not clear. At the same time, a rapid degradation of ANN closure accuracy has been observed for all the models. Expanding the training dataset with features that are representative of ANN error statistics is likely a good strategy to address this degradation.

8

Conclusions

In this thesis, a data-driven approach to modelling the closure terms in variational multi-scale framework has been presented. By using integral forms of coarse-scale features, very high a priori correlations can be achieved for momentum closure terms, even with a relatively compact feature stencil of two elements and two time steps. However, a posteriori evaluations of LES simulations equipped with ANN closure show that the model cannot maintain high correlation accuracy and provides a poor match of the resolved Reynolds stress components in the first element near the wall. An excessive gain in turbulent energy is observed, while simultaneously a decay in mean flow energy is found.

RQ-1. How is the stability of the variational multiscale flow solution affected by introduction of ANN closure?

Using the current ANN closure models leads to the steep decrease in the correlation of modelled and exact momentum terms after 10–20 LES time steps.

RQ-1a. How does the flow deviate from projected DNS solution when subject to ANN closure?

With all the models tested a posteriori, significant deviations have been observed in the near-wall elements, hinting that the ANN models fail to correctly estimate the turbulence production in this region. After some simulation time, a deviation becomes so significant that the difference between resolved turbulent energy of DNS projection and LES grows exponentially, eventually leading to solution instability. At the same time, the mean flow energy is observed to be decreasing, suggesting that the energy is transferred from the mean flow into turbulent fluctuations, but this has not been explicitly confirmed.

RQ-1b. Can long-term instabilities (LTI's) be adequately addressed through additional filtering of the DNS solution during offline ANN training?

Filtering the DNS projection to modify the closure terms has been shown to be effective in manipulation of the run-time turbulent spectra of LES simulations. However, it is not clear whether this technique is advantageous for simulation stability. The accumulation of turbulent energy is likely linked to erroneous predictions in the near wall element layer(s), possibly leading to over-estimation of turbulent energy production, and filtering technique has only been observed to change how excess turbulent energy is distributed among turbulent modes. A more detailed evaluation of filtering technique on multiple cases is needed to fully answer this sub-question.

RQ-1c. Can LTI's be addressed through introduction of new accuracy metrics during offline network training?

It has been confirmed that achieving high a priori correlations doesn't guarantee flow stability in an a posteriori setting, as the model accuracy quickly degrades when it is faced with inputs corrupted by closure term error. Instead, to improve a posteriori solver performance, new a priori metrics need to be identified and implemented during the model training, to account for adverse effects of erroneous closure terms. One such metric has been identified to be the mini-batch sum loss, which punishes the model deviations from the truth values in terms of averages over large batches of predictions. Inclusion of this metric has shown an advantage in terms of conservation of total kinetic energy within the flow, without any adverse effects on closure term accuracy. Thus, it has been shown that a suitable accuracy metric can be beneficial for *online* model stability, but further steps are needed to achieve full model stability.

RQ-2. How can the training and run-time computational cost of the neural network be reduced?

The training cost of the model has been significantly reduced by adopting a universal model strategy, which is unaware of the index of element shape function used for integration. This way, the training dataset can be expanded by a factor of 8, which is beneficial for model generalization, but only 1 model needs to be trained. In addition, the network architecture has been radically downsized from 5 hidden layers of 600 neurons each to a network with a single hidden layer of 100 neurons. Still correlations of approximately 0.90 for streamwise momentum term are observed, 0.95 for span- and wall-normal momentum closures, and over 0.99 for continuity closure.

RQ-2a. What input features can be discarded without compromising the accuracy?

By performing a small sensitivity study, three most relevant and important coarse-scale feature terms have been identified. These are the time-derivative term $\left(\bar{\varphi}_q, \frac{\partial \bar{\mathbf{u}}}{\partial t}\right)_e$, resolved convection term $\left(\bar{\varphi}_q, \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}})\right)_e$ and resolved divergence term $\left(\bar{\psi}_q, \nabla \cdot \bar{\mathbf{u}}\right)_e$. Pressure gradient and diffusion terms contribute very little to higher correlation coefficients with exact closure terms.

RQ-2b. Can the feature set stencil be compacted without compromising network accuracy?

The feature set has been compacted in comparison to the stencil used by A. Bettini [20]. The standard stencil includes the original element together with its upstream neighbor, with information from the current and previous time steps. Including more elements or having a longer flow history has not led to increased accuracy in closure term correlation, and thus no accuracy losses are expected by using a compact stencil.

RQ-2c. Can the training routine be optimized?

Besides reducing the simplicity and cost of training methodology by adopting a single network formulation, two options are now available for network training:

- Training with full dataset in RAM - taking advantage of large RAM of work stations used (64 GB), a training dataset of up to 30 GB can be loaded into RAM (accounting for the generous memory allocation by Linux kernel). Once the dataset is loaded, the training is accelerated, as the data can be shuffled in RAM much faster.
- Another option is pre-processing the training data into NumPy shard files, each of up to 5 GB. While previously binary TensorFlow files are used, these files can not be opened or used for anything else besides training in TensorFlow framework. Meanwhile, NumPy files can be directly used for post-processing, data augmentation such as rotation of coordinate axes, etc.

Using the first option has shown a massive acceleration in model training, leads to a more than 10x speed-up.

RQ-3. How does the ANN model respond to numerical refinement?

The ANN model shows the most promising results at 16^3 mesh resolution with a CFL number of 0.5. For this specific case, a better match with DNS projection has been observed for the first 15 LES time steps with an ANN model than with the algebraic model.

RQ-3a. What is the effect of mesh size on model accuracy and stability?

The error accumulation induced by the ANN model is sensitive to the mesh refinement. For 16^3 LES mesh case, the model inserted excessive kinetic energy in the near wall region, without strong preference for high- or low wave number regions. While for 32^3 mesh case, high wave number energy accumulation has been observed. Filtering the training data-set can be used to manipulate the energy spectra induced by the model, but no improvement in solution stability has been observed for 16^3 the case, likely as this technique does not address the cause of excess turbulent kinetic energy production.

Comparing LES mesh of 16^3 elements to 32^3 elements, an improvement in simulation stability and accuracy over longer simulation times have been observed for a coarser mesh.

RQ-3b. What is the effect of CFL number on model accuracy and stability?

The effect of CFL number was investigated by comparing two 32^3 cases at CFL number of 0.5 and 1.0. While the correlations of closure terms and RMS error of streamwise velocity field are very similar (accounting for longer time steps), running the model at CFL number of 1.0 is detrimental to evolution of instantaneous flow kinetic energy. Therefore, it is recommended to keep the CFL number at 0.5 or below.

RQ-4. What is the effect of data augmentation on network accuracy and stability?

No improvements in stability or accuracy have been observed through the augmentation of training dataset by a rotation of coordinate axes. It is likely that this approach will only show improvement if the test case experiences a mean flow direction misaligned with the coordinate axes, which was not the case for the current validation tests.

Recommendations

There are several recommendations and research avenues that can lead to potential improvement in data-driven closure modelling:

- **Turbulent kinetic energy budget evaluation.** The turbulent energy budget should be computed to further explain the accumulation of turbulent kinetic energy and decay of mean flow kinetic energy. The most relevant terms, such as the energy transfer term from the mean flow into turbulent kinetic energy, can be evaluated during the training and introduced as a new loss function, to penalize the ANN model from significant deviations in energy transfer.
- **Training dataset diversification.** The training dataset of the model should be expanded by including input features that are corrupted with ANN-induced error. Including snapshots of simulations with accumulation of ANN error during training would help the network to maintain accurate closure term predictions even when faced with its own error in coarse-scale features. Alternatively, projected DNS flow fields can be filtered by using already developed methodology to "simulate" the spectra contaminated with ANN error, or the training dataset can be directly augmented with Gaussian noise, as was done in 1-D simulation by [19].
- **Adjusted training approach.** Given that the most significant errors have been observed at the near-wall element, the model has to be tuned more aggressively in the near-wall region. This can be achieved by using a varying factor loss function, that leads to higher loss values near the walls and decreases in the center of the channel, i.e. $L_z(y, \tilde{y}, z) = f(z) \cdot L(y, \tilde{y})$
- **Further investigation into the resolved pressure equation.** A cause for divergence of the pressure solution is still not clear, but is likely linked to $\nabla \cdot (\mathcal{L}_{\mathcal{M}'}(\mathbf{u}', \bar{\mathbf{u}}, p')_{ANN})$ and $(\frac{\partial}{\partial t} \nu \nabla \cdot \nabla) (\mathcal{L}_C(\mathbf{u}')_{exact})$ terms. A Fourier transform of eq. (5.4) can perhaps explain the high-frequency pressure fluctuation observed when using ANN model momentum predictions.

Bibliography

- [1] Thomas J R Hughes'. *Computer methods in applied mechanics and engineering Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods**. Tech. rep. 1995, pp. 387–401.
- [2] Thomas J.R. Hughes, Assad A. Oberai, and Luca Mazzei. “Large eddy simulation of turbulent channel flows by the variational multiscale method”. In: *Physics of Fluids* 13.6 (2001), pp. 1784–1799. ISSN: 10706631. DOI: 10.1063/1.1367868.
- [3] Thomas J. R. Hughes, Guglielmo Scovazzi, and Leopoldo P. Franca. “Multiscale and Stabilized Methods”. In: *Encyclopedia of Computational Mechanics Second Edition*. Wiley, Nov. 2017, pp. 1–64. DOI: 10.1002/9781119176817.ecm2051.
- [4] A Dunca, V John, and W J Layton. *The Commutation Error of the Space Averaged Navier-Stokes Equations on a Bounded Domain*. 2004, pp. 53–78.
- [5] O.C. Zienkiewicz. *The Finite Element Method Vol 3: Fluid Dynamics*. 5th. Vol. 3. Elsevier, 2000.
- [6] F Moukalled, L Mangani, and M Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Tech. rep. 2016. URL: <http://www.springer.com/series/5980>.
- [7] Joseph Smagorinsky. “General Circulation Experiments with the Primitive Equations”. In: *Monthly Weather Review* 91.3 (1963).
- [8] N. A. Adams and S. Hickel. “Implicit Large-Eddy Simulation: Theory and Application”. In: *Springer Proceedings in Physics*. Vol. 132. Springer Science and Business Media Deutschland GmbH, 2009, pp. 743–750. ISBN: 9783642030840. DOI: 10.1007/978-3-642-03085-7_{_}180.
- [9] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. “Annual Review of Fluid Mechanics Turbulence Modeling in the Age of Data”. In: *Annu. Rev. Fluid Mech* 51 (2019), pp. 357–377. DOI: 10.1146/annurev-fluid-010518. URL: <https://doi.org/10.1146/annurev-fluid-010518->.
- [10] Massimo Germano et al. “A dynamic subgrid-scale eddy viscosity model”. In: *Physics of Fluids A* 3.7 (1991), pp. 1760–1765. ISSN: 08998213. DOI: 10.1063/1.857955.
- [11] A. Volland, G. Balarac, and C. Corre. “Subgrid-scale scalar flux modelling based on optimal estimation theory and machine-learning procedures”. In: *Journal of Turbulence* 18.9 (Sept. 2017), pp. 854–878. ISSN: 14685248. DOI: 10.1080/14685248.2017.1334907.
- [12] Andrea Beck, David Flad, and Claus Dieter Munz. “Deep neural networks for data-driven LES closure models”. In: *Journal of Computational Physics* 398 (Dec. 2019). ISSN: 10902716. DOI: 10.1016/j.jcp.2019.108910.

- [13] Y. Bazilevs et al. “Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows”. In: *Computer Methods in Applied Mechanics and Engineering* 197.1-4 (Dec. 2007), pp. 173–201. ISSN: 00457825. DOI: 10.1016/j.cma.2007.07.016.
- [14] T. J.R. Hughes and G. Sangalli. “Variational multiscale analysis: The fine-scale green’s function, projection, optimization, localization, and stabilized methods”. In: *SIAM Journal on Numerical Analysis* 45.2 (2007), pp. 539–557. ISSN: 00361429. DOI: 10.1137/050645646.
- [15] Simon S. Haykin and Simon S. Haykin. *Neural networks and learning machines*. Prentice Hall/Pearson, 2009, p. 906. ISBN: 9780131471399.
- [16] Martin Janssens. “Machine Learning of Atmospheric Turbulence in a Variational Multiscale Model”. PhD thesis. Delft: Delft University of Technology, 2019. URL: [http://repository.tudelft.nl/..](http://repository.tudelft.nl/)
- [17] V S K Rajampeta. “Master Thesis Stabilization Strategies for a Variational Multiscale Method coupled with an Artificial Neural Network”. PhD thesis. Delft: Delft University of Technology, 2021.
- [18] Michel Robijns. “A Machine Learning Approach to Unresolved-Scale Modeling for Burgers’ Equation”. PhD thesis. Delft: Delft University of Technology.
- [19] Abhinand Pusuluri. “Noise-augmented offline training of ANN unresolved scale models”. PhD thesis. Delft: Delft University of Technology, 2020. URL: [http://repository.tudelft.nl/..](http://repository.tudelft.nl/)
- [20] Andrea Bettini. *Energy-Conservative Data-Driven Modelling for the two-scale Navier-Stokes Equations*. Tech. rep. Delft: Delft University of Technology, 2023. URL: [http://repository.tudelft.nl/..](http://repository.tudelft.nl/)
- [21] O.C. Zienkiewicz. *The Finite Element Method Vol 1: The Basis*. 5th. Vol. 1. Elsevier, 2000, pp. 39–45.
- [22] Ricardo Vinuesa and Steven L. Brunton. “Enhancing computational fluid dynamics with machine learning”. In: *Nature Computational Science* 2.6 (June 2022), pp. 358–366. ISSN: 26628457. DOI: 10.1038/s43588-022-00264-7.
- [23] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance”. In: *Journal of Fluid Mechanics* 807 (Nov. 2016), pp. 155–166. ISSN: 14697645. DOI: 10.1017/jfm.2016.615.
- [24] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. ISSN: 10902716. DOI: 10.1016/j.jcp.2018.10.045.
- [25] F Rosenblatt. “The Perceptron: A probabilistic model for information storage and organization of the brain”. In: *Psychological Review* 65.6 (1958), pp. 19–27.

-
- [26] Andrea Bettini and Steven Hulshoff. *Data-driven modelling in the two-scale incompressible Navier-Stokes equations*. Tech. rep. 2024.
- [27] H. Goldstein. "The Euler Angles" and "Euler Angles in Alternate Conventions." 4-4 and Appendix B ". In: *Classical Mechanics*. Reading MA: Addison-Wesley, 1980.
- [28] I. Akkerman et al. "The role of continuity in residual-based variational multiscale modeling of turbulence". In: *Computational Mechanics* 41.3 (2008), pp. 371–378. ISSN: 01787675. DOI: 10.1007/s00466-007-0193-7.