NEW METHODOLOGIES FOR MACHINE LEARNING ASSISTED TURBULENCE BASED ON THE TRANSPORT EQUATIONS FOR THE REYNOLDS STRESS TENSOR AND THE REYNOLDS FORCE VECTOR

Matheus de Souza Santos Macedo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Mecânica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Mecânica.

Orientador: Roney Leon Thompson

Rio de Janeiro
Julho de 2020

NEW METHODOLOGIES FOR MACHINE LEARNING ASSISTED TURBULENCE BASED ON THE TRANSPORT EQUATIONS FOR THE REYNOLDS STRESS TENSOR AND THE REYNOLDS FORCE VECTOR

Matheus de Souza Santos Macedo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA MECÂNICA.

Orientador: Roney Leon Thompson

Aprovada por: Prof. Roney Leon Thompson
Prof. Angela Ourivio Nieckele
Prof. Daniel Alves Castello

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2020

Macedo, Matheus de Souza Santos

New Methodologies for Machine Learning Assisted Turbulence Based on the Transport Equations for the Reynolds Stress Tensor and the Reynolds Force Vector/Matheus de Souza Santos Macedo. – Rio de Janeiro: UFRJ/COPPE, 2020.

XIII, 102 p.: il.; 29, 7cm.

Orientador: Roney Leon Thompson

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Mecânica, 2020.

Referências Bibliográficas: p. 87 – 90.

1. Turbulence. 2. Machine Learning. 3. OpenFOAM. I. Thompson, Roney Leon. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Mecânica. III. Título.

Aos meus pais Eliane e Waldir, e
à minha irmã Nicole.

Agradecimentos

Primeiramente agradeço ao Povo Brasileiro pelo financiamento deste trabalho, não só pela estrutura da UFRJ e da Coppe por ele construída, como também pela bolsa que me sustentou ao longo destes 2 anos. Além disso, agradeço também pelo financiamento continuado depositado em mim desde meu ingresso na universidade pública. Este agradecimento é feito com o desejo de que um dia este caminho que eu tracei seja simples questão de escolha individual, quando todos tenham a estrutura social, familiar e financeira que me permitiu estar hoje aqui. Neste país isto é impossível para muitos de nós e infelizmente um quadro que não vislumbra melhora.

Agradeço especialmente aos meus pais e à minha irmã, pelo carinho, convívio, paciência e por tornarem possível mais esta etapa de minha formação. Além da estrutura familiar, é impossível a um mestrando, numa região metropolitana como a do Rio de Janeiro, sustentar-se sozinho apenas com os proventos de sua bolsa, quando a tem.

À minha namorada Laís a minha gratidão pela amizade, carinho, paciência e ajuda em todo o processo. Sua família, sempre presente apesar da distância geográfica, tem também os meus agradecimentos pelo suporte e carinho.

Agradeço ao meu amigo Eduardo, pelos fundamentais almoços (quase) semanais.

Ao meu orientador e amigo, professor Roney, agradeço muito por todo o conhecimento, disponibilidade e camaradagem ao longo deste período. Sou muito grato pela inovadora linha de pesquisa a mim apresentada por ele, na qual tive imenso prazer em trabalhar e na qual pretendo me aprofundar nos próximos anos. O meu agradecimento estende-se também aos colegas do grupo de pesquisa em Machine Learning e Turbulência, em especial ao Matheus Altomare, por toda ajuda e contribuição neste trabalho.

Agradeço à Universidade Federal do Rio de Janeiro e à Coppe, pela estrutura disponibilizada, em especial a todos os professores do PEM que muito me ensinaram neste período. Todos os servidores e funcionários da UFRJ, em especial os da secretaria do PEM, tem também a minha gratidão.

Ao CNPq, pela bolsa de mestrado que, infelizmente, cada vez mais é um privilégio e menos uma certeza de quem deseja seguir a pós-graduação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

NOVAS METODOLOGIAS PARA TURBULÊNCIA ASSISTIDA POR APRENDIZADO DE MÁQUINA BASEADA NAS EQUAÇÕES DE TRANSPORTE PARA O TENSOR E PARA O VETOR DE REYNOLDS

Matheus de Souza Santos Macedo

Julho/2020

Orientador: Roney Leon Thompson

Programa: Engenharia Mecânica

A antiga demanda por melhores modelos de turbulência e o custo computacional ainda proibitivo de simulações de alta-fidelidade em dinâmica dos fluidos, como DNS e LES, levaram ao crescente interesse em acoplar dados de simulações de alta-fidelidade com as populares, porém deficientes, simulações RANS, através de técnicas de Aprendizado de Máquina. Estas técnicas usam dados nobres como alvos para previsões de grandezas a serem propagadas pelas equações RANS. Muitos dos avanços recentes utilizaram o tensor de Reynolds como alvo destas correções. Mais recentemente, uma metodologia alternativa utilizou o divergente do tensor de Reynolds, denominado Vetor de Força de Reynolds, como alvo do Aprendizado de Máquina.

Uma nova estratégia é o uso de equações de transporte de grandezas turbulentas contendo termos fonte previstos por Aprendizado de Máquina. Neste contexto, duas novas metodologias foram propostas, uma delas utilizando a equação de transporte do tensor de Reynolds e outra utilizando uma equação para o Vetor de Força de Reynolds. A combinação destas equações com o balanço de momento e um acoplamento com a pressão formou dois modelos de turbulência com base em dados.

Redes Neurais foram treinadas utilizando dados DNS para prever os termos fonte de cada equação. Em seguida, os modelos foram usados para corrigir o escoamento turbulento em um duto de seção quadrada. Resultados razoáveis foram obtidos pelos dois modelos de turbulência, consistentemente recuperando o escoamento secundário no duto, que não existia nas simulações iniciais que utilizaram o modelo κ - ϵ. As duas metodologias foram também comparados com abordagens alternativas previamente apresentadas na literatura.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

NEW METHODOLOGIES FOR MACHINE LEARNING ASSISTED TURBULENCE BASED ON THE TRANSPORT EQUATIONS FOR THE REYNOLDS STRESS TENSOR AND THE REYNOLDS FORCE VECTOR

Matheus de Souza Santos Macedo

July/2020

Advisor: Roney Leon Thompson

Department: Mechanical Engineering

The long lasting demand for better turbulence models and the still prohibitively computational cost of high-fidelity fluid dynamics simulations, like DNS and LES, have led to a rising interest in coupling available high-fidelity datasets and popular, yet poor, RANS simulations through Machine Learning techniques. These techniques use noble sources as training targets for predicting quantities to be propagated by RANS equations. Many of the recent advances used the Reynolds stress tensor as the target for these corrections. More recently, an alternate methodology used the divergence of the Reynolds stress, denominated the Reynolds Force Vector, computed indirectly by manipulating mean momentum balance, as the target for the Machine Learning techniques.

An unexplored strategy in this effort is to use transport equations for turbulent quantities fueled by Machine Learning predicted source terms. In this context, two new methodologies were proposed, one using a transport equation for the Reynolds Stress and another one using a transport equation for the Reynolds Force Vector. The combination of these transport equations along with the momentum balance and a pressure coupling formed two data-driven turbulence models.

Neural Networks were trained using DNS data to predict the source terms of each equation. Subsequently, both proposed models were employed to correct the turbulent flow on a square-duct. Reasonable results were obtained by both data-driven turbulence models, consistently recovering the secondary flow on the duct, which was not present in the baseline simulations that used the κ - ϵ model. Results from both methods were compared with alternate strategies previously presented in literature.

Contents

List of Figures **x**

List of Tables **xiii**

1 Introduction **1**

2 Turbulence **5**

2.1 Mathematical Modelling of Turbulence 9

2.1.1 The Boussinesq Hypothesis and The $\kappa - \epsilon$ Model 11

3 Neural Networks **14**

3.1 The Multi-Layer Perceptron and Forward Propagation 16

3.2 Network Training and Back-Propagation 17

3.3 Hyper-Parameters 20

3.4 Train, Validation and Test Stages 21

4 Machine Learning Assisted Turbulence **23**

5 The Square-Duct Flow **33**

5.1 Baseline RANS Simulation of the Square-Duct Flow 34

6 Methodology **37**

6.1 The Adapted Reynolds Stress Tensor Transport Equation 37

6.2 The Reynolds Force Vector Transport Equation 38

6.2.1 The Modified RFVTE 40

6.3 The Data-Driven Turbulence Models 41

6.4 Neural Networks Setup 44

6.4.1 Prediction of $\hat{\mathbf{\Gamma}}$ and $\hat{\boldsymbol{\gamma}}$ 45

6.4.2 Prediction of $\mathbf{t}$ and $\mathbf{R}$ 52

6.4.3 Post-processing Predicted Quantities 53

7 Results **55**

7.1 A Priori Results 60

7.1.1 Prediction of $\mathbf{R}$ . . . 61
7.1.2 Prediction of $\mathbf{t}$ . . . 65
7.1.3 Prediction of $\hat{\mathbf{\Gamma}}$ . . . 68
7.1.4 Prediction of $\hat{\boldsymbol{\gamma}}$ . . . 73
7.2 A Posteriori Results . . . 76
7.2.1 Corrected $\mathbf{R}$ and $\mathbf{t}$ . . . 76
7.2.2 Recirculation . . . 78
7.2.3 Primary Flow . . . 81
7.3 Global Errors . . . 83

8 Conclusion **85**
8.1 Future Works . . . 86

References **87**

A Translating RFVTE From Indicial to Symbolic Notation **91**

B Omitted Plots **92**
B.1 $\mathbf{R}$ . . . 92
B.2 $\mathbf{t}$ . . . 95
B.3 $\hat{\mathbf{\Gamma}}$ and $\hat{\boldsymbol{\gamma}}$ . . . 97
B.4 $\mathbf{u}$ . . . 101

List of Figures

1.1 (a) Comparison between multiple RANS models (b) Comparison between multiple coefficient values in a same model 2

2.1 Transition on flow regime on inclined plate 6
2.2 Large and small eddies in a turbulent boundary layer over a flat plate 7
2.3 Flow past cylinder, two realizations of $u(x_0, t)$ and time average $\bar{u}(x_0)$ 9

3.1 Model of neuron used in Artificial Neural Networks, labeled as k . . . 15
3.2 Neural Network with 6 neurons and a single hidden layer 17
3.3 (a) $\sigma(\upsilon_k)$ (b) $\tanh(\upsilon_k)$ (c) $\mathrm{ReLU}(\upsilon_k)$ 21

4.1 Boussinesq hypothesis' performance versus a more complex constitutive relationship performance 24
4.2 Boussinesq hypothesis' perfomance versus other six more complex models 24
4.3 (a) Predicted versus true s value, associated to poor propagation, (b) Predicted versus true s values, associated to good propagation 26
4.4 (a) Square-duct predicted component $\mathbf{R}_{zz}$ profiles, (b) Periodic hills predicted component $\mathbf{R}_{xy}$ profiles 28
4.5 Predicted turbulent kinetic energy k profiles with trained using modified geometries 29
4.6 (a) u_y velocity component, (b) u_z velocity component 30
4.7 Comparison between RANS, DNS, and correction by NN predicted $\mathbf{t}$ 31
4.8 Global errors for RANS, $\mathbf{R}_{DNS}$, $\mathbf{t}_{DNS}$, predicted $\mathbf{R}$ and predicted $\mathbf{t}$. 32

5.1 Square-duct flow geometry and secondary flow 33
5.2 Baseline simulations' mesh 34
5.3 Boundary fields of the square-duct 35

6.1 Inaccurate $\mathbf{t}_{\mathrm{DNS}}$ regions 42
6.2 $\mathbf{t}_{\mathrm{DNS}}$ after pre-processing 43
6.3 (a) 84 inputs learning curve, (b) 22 inputs learning curve 47

6.4 Training datapoints weights 50
6.5 $\hat{\boldsymbol{\gamma}}$ neural networks training flow-chart 51
6.6 $\hat{\boldsymbol{\gamma}}$ neural networks testing flow-chart 52
6.7 Symmetric and mirrored components demonstration 54

7.1 u_x reconstructed by different methodologies ($Re = 2900$) 56
7.2 u_y reconstructed by different methodologies ($Re = 2900$) 57
7.3 u_z reconstructed by different methodologies ($Re = 2900$) 58
7.4 Reconstructed $\mathbf{R}$ components ($Re = 2900$) 59
7.5 Reconstructed $\mathbf{t}$ components ($Re = 2900$) 60
7.6 RANS, DNS and predicted R_{xx} for the test case of $Re = 2900$ 61
7.7 R_{xx} samples 62
7.8 RANS, DNS and predicted R_{xy} for the test case of $Re = 2900$ 62
7.9 R_{xy} samples 63
7.10 RANS, DNS and predicted R_{yy} for the test case of $Re = 2900$ 63
7.11 R_{yy} samples 64
7.12 RANS, DNS and predicted R_{yz} for the test case of $Re = 2900$ 64
7.13 R_{yz} samples 65
7.14 RANS, DNS and predicted t_x for the test case of $Re = 2900$ 66
7.15 t_x samples 66
7.16 RANS, DNS and predicted t_y for the test case of $Re = 2900$ 67
7.17 t_y samples 67
7.18 RANS, DNS and predicted $\hat{\Gamma}_{xx}$ for the test case of $Re = 2900$ 69
7.19 $\hat{\Gamma}_{xx}$ samples 69
7.20 RANS, DNS and predicted $\hat{\Gamma}_{xy}$ for the test case of $Re = 2900$ 70
7.21 $\hat{\Gamma}_{xy}$ samples 70
7.22 RANS, DNS and predicted $\hat{\Gamma}_{yy}$ for the test case of $Re = 2900$ 71
7.23 $\hat{\Gamma}_{yy}$ samples 71
7.24 RANS, DNS and predicted $\hat{\Gamma}_{yz}$ for the test case of $Re = 2900$ 72
7.25 $\hat{\Gamma}_{yz}$ samples 72
7.26 RANS, DNS and predicted $\hat{\gamma}_x$ for the test case of $Re = 2900$ 73
7.27 $\hat{\gamma}_x$ samples 74
7.28 RANS, DNS and predicted $\hat{\gamma}_y$ for the test case of $Re = 2900$ 74
7.29 $\hat{\gamma}_y$ samples 75
7.30 $\mathbf{t}_{\text{wall}}$ on $z = -1.0$ 76
7.31 DNS and corrected $\mathbf{R}$ components 77
7.32 DNS and corrected $\mathbf{t}$ components 78
7.33 u_y corrected by the four methodologies 79
7.34 Corrected u_y samples 80

7.35 Recirculation magnitude given by the three methodologies 81
7.36 u_x corrected by the three methodologies 82
7.37 Corrected u_x samples 83
7.38 Global error on recirculation $\overline{E_y}$ 84
7.39 Global error on main flow $\overline{E_x}$ 84

B.1 RANS, DNS and predicted R_{xz} for the test case of $Re = 2900$ 92
B.2 R_{xz} samples 93
B.3 RANS, DNS and predicted R_{zz} for the test case of $Re = 2900$ 93
B.4 R_{zz} samples 94
B.5 Corrected $\mathbf{R}$ 94
B.6 RANS, DNS and predicted t_z for the test case of $Re = 2900$ 95
B.7 t_z samples 96
B.8 Corrected t_z component 96
B.9 RANS, DNS and predicted $\hat{\Gamma}_{xz}$ for the test case of $Re = 2900$ 97
B.10 Γ_{xz} samples 98
B.11 RANS, DNS and predicted $\hat{\Gamma}_{zz}$ for the test case of $Re = 2900$ 98
B.12 Γ_{zz} samples 99
B.13 RANS, DNS and predicted $\hat{\gamma}_z$ for the test case of $Re = 2900$ 99
B.14 $\hat{\gamma}_z$ samples 100
B.15 u_z corrected by the four methodologies 101
B.16 Corrected u_z samples 102

List of Tables

2.1 $\kappa - \epsilon$ model's constants values 13

5.1 Simulated Re and corresponding kinematic viscosities 36

6.1 A priori inputs 46
6.2 Final list of inputs 47
6.3 Input and output matrices X and Y sizes 49
6.4 Input and output matrices X and Y sizes for the $\mathbf{t}_{\text{wall}}$ network 51
6.5 Summary of $\hat{\mathbf{\Gamma}}$, $\hat{\boldsymbol{\gamma}}$, $\mathbf{t}$ and $\mathbf{R}$ networks 52
6.6 Summary of $\mathbf{t}_{\text{wall}}$ network 53

Chapter 1

Introduction

The simulation of turbulent flows in computational fluid dynamics (CFD) has been a major issue for decades. Solving these problems via physically stringent simulations demand computational resources unavailable for most of the industry and researchers. They are also so time consuming that solving a single problem may take long enough to make them unviable for most applications.

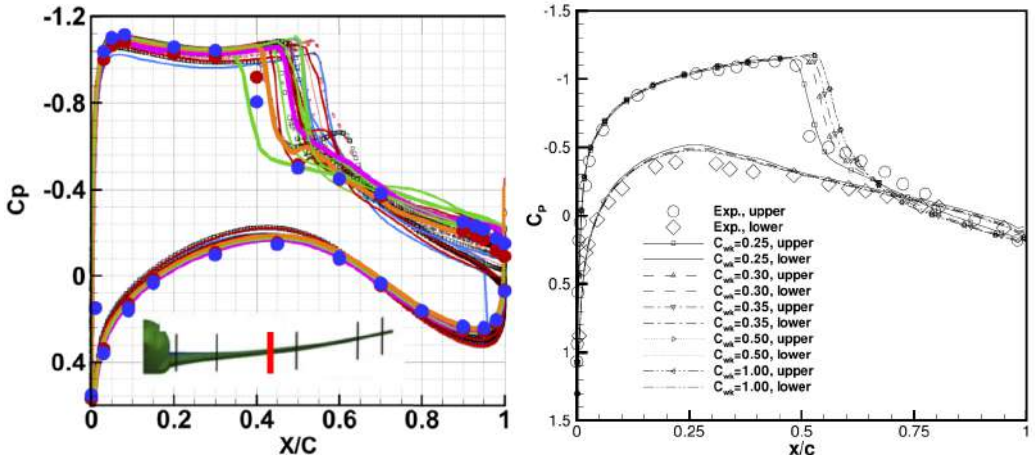
On this family of high-fidelity methods of fluid flow simulations the Direct Numerical Simulations (DNS) stand out as the most accurate available. Other less accurate but still physically rigorous alternative are the Large Eddy Simulations (LES). Since most industrial applications of CFD require numerous repetitions of simulations, and extensive variations in flow characteristics in the research and design processes, using DNS and even LES is not feasible.

A number of methods have been proposed to bypass this long lasting difficulty. Most popular ones are the Reynolds Averaged Navier-Stokes (RANS) models. These methods simplify the physics of fluid flows by filtering them through a time averaging procedure. Consequently, solving these flows numerically is much faster. Their stability and the possibility of simulating and re-simulating numerous cases in short periods of time are the reason behind their popularity, specially in industry.

However, due to their oversimplification of flow's phenomena, these models present a number of limitations that handicaps their results and that make them unreliable for many applications. DURAISAMY *et al.* (2019) lists four central sources of uncertainty in RANS models.

1. Uncertainties due to mathematical simplification of momentum equations
2. Uncertainties introduced by the choice of modelization of turbulent stresses
3. Uncertainties associated to the functional forms within a chosen model
4. Uncertainties in the calibrated coefficients inside the model's functions

Figure 1.1 illustrates the uncertainties associated with a number of different RANS models, and associated with different coefficients inside the same model. Panel (a) of figure 1.1 shows the pressure coefficients C_p profiles in a wing section of an aircraft, highlighted by the red line. Every distribution is provided by a different model. On panel (b) of figure 1.1 several C_p distributons on a same transonic flow over an NACA0012 airfoil are shown. The different curves are the result of varying a single parameter in an algebraic RANS model that has seven coefficients in total.



(a) (b)

Figure 1.1: (a) Comparison between multiple RANS models (b) Comparison between multiple coefficient values in a same model

Taken from XIAO e CINNELLA (2019) *apud* TINOCO *et al.* (2018) and CINNELLA (2016), respectively.

The necessity of making a compromise between immense time consumption associated to huge computational resources versus low accuracy predictions permeated with uncertainties, that most of the time are not quantifiable, has caused a growing interest in methods of improving RANS simulations. These methods rely on Machine Learning (ML) techniques to combine the predictions of high-fidelity sources like DNS, LES and experimental, with the popular yet poor RANS predictions.

TRACEY *et al.* (2015) developed one of the earliest works in this field. There, Neural Networks (NN) are used to predict the source term in the Spallart-Allmaras (SA) one-equation turbulence model. Although it does not use higher-fidelity sources of data, the authors demonstrate that NN are capable of satisfactorily predicting the desired quantities when aiming at the previously known SA analytical results.

Following, LING *et al.* (2016) uses NN combined with DNS and LES results to predict the anisotropic part $\mathbf{B}$ of the Reynolds Stress Tensor $\mathbf{R}$. Later, the predicted $\mathbf{B}$ was propagated through RANS momentum equations in order to improve the velocity fields. Reasonable results were achieved, indicating that approaches using

NN are indeed effective as a mean of correcting lower-fidelity simulations. LING *et al.* (2016) also demonstrates that $\mathbf{B}$ can be considered a suitable target for the ML method.

Other works have used the discrepancy between RANS and high-fidelity $\mathbf{R}$ as the target for the ML procedure. The predicted discrepancies are then used to correct the baseline Reynolds stress. WANG *et al.* (2017a) successfully used Random Forest models to correct the Reynolds stress via predicted discrepancies in a square-duct flow. WANG *et al.* (2017b) also applies Random Forests to predict the discrepancies, the adressed problems were once again the square-duct and also the flow over periodic-hills.

After THOMPSON *et al.* (2016) demonstrated that although DNS data is canonically considered the highest quality data available, some of their provided quantities cannot be as highly regarded as others. Namingly, the provided second statistical moments, that is, the components of $\mathbf{R}$, in many DNS databases are not as well converged numerically as the mean velocities. Therefore, as pointed by the authors, injecting these imprecise quantities into RANS environments may lead to propagation of these defects, resulting in different flow fields than those provided by the database. As a consequence, using the $\mathbf{R}$ from DNS simulations in any type o modelization, including ML, will surely impair the quality of results.

In order to avoid the uncertainties evidentiated in THOMPSON *et al.* (2016) a novel methodology is proposed by CRUZ *et al.* (2019). Because targeting the Reynolds stress with ML strategies might be inadequate, the author indirectly calculates the divergence of $\mathbf{R}$ through the mean velocity field and the mean momentum equations (MME).

CRUZ *et al.* (2019) denominates this quantity the Reynolds Force Vector, $\nabla \cdot \mathbf{R} \equiv \mathbf{r}$. By using $\mathbf{r}$ as the ML targeted value, the posteriori flow field resulting from the propagation of this quantity by the RANS simulations shows much better agreement with the DNS data than those provided by the use of $\mathbf{R}$ as a target for the ML technique.

Until the current moment, all efforts to correct RANS simulations via machine learning techniques have focused on predicting turbulent quantities to be directly injected into the momentum equations. The present work proposes two alternate methodologies to correct RANS flows, through the use of data-driven turbulence models that corrects both velocity field and the turbulent quantity.

The first model was constructed using an adapted Reynolds Stress Transport Equation (RSTE) along with the MME and a pressure coupling algorithm, forming a system of ten coupled partial differential equations (PDEs). For this purpose the canonical form of the RSTE was adapted into a data-driven equation, with a source term $\mathbf{\Gamma}$ responsible for fueling the proposed procedure. This source symmetric tensor

$\mathbf{\Gamma}$ is composed by all of the terms in the RSTE which require modeling.

The second turbulence model is a direct extension of the methodology utilized by CRUZ *et al.* (2019), employing a transport equation for the Reynolds Force Vector derived from the canonical RSTE. The deduced Reynolds Force Vector Transport Equation (RFVTE) also contains a source term, the vector $\boldsymbol{\gamma}$, which comprises all of the terms that require modeling or that are not direct functions of $\mathbf{r}$ itself. The set of seven PDEs formed by the RFVTE, the MME and a pressure equation also constitute a data-driven turbulence model.

Both proposed turbulence models were employed to correct RANS flows via the ML predicted quantities tensor $\mathbf{\Gamma}$ and vector $\boldsymbol{\gamma}$. To predict both source terms two groups of NN were trained for each methodology, using DNS results of a square-duct flow provided by PINELLI *et al.* (2010). The RFVTE methodology also requires the training of a third set of neural networks, used to predict the boundary conditions (BC) of $\mathbf{r}$. With the predicted $\mathbf{\Gamma}$, $\boldsymbol{\gamma}$ and the BC to be used in the RFVTE, RANS results were used as initial conditions (IC) and were thus corrected separately by the two set of partial differential equations.

What both proposed methodologies have in common is that the terms to be predicted by ML techniques are injected into a transport equation for the turbulent quantity that is subsequently injected in the momentum balance. This remained an unexplored approach to improve RANS velocity fields while also improving turbulent quantities.

Results of both methodologies were compared with the ones obtained by the prediction and subsequent injection of $\mathbf{R}$ and $\mathbf{r}$ into the same RANS simulations.

Chapter 2

Turbulence

The study of turbulent flows is of primary importance to engineering and most segments of industry, its applications impacts society as a whole. An in-depth knowledge of fluid mechanics is not necessary to realize that laminar flows are the exception and not the rule and very few observations are required to quickly confirm that turbulence is ubiquitous in nature and in human applications.

Laminar flow is characterized by the organized motion of parallel layers of fluid sliding over one another. In these, the dimensionless Reynolds Number (Re), described in equation 2.1 is below a certain threshold, which varies depending on circumstances such as experimental arrangement.

$$Re = \frac{UL}{\nu} \tag{2.1}$$

For example, according to KUNDU *et al.* (2016), when Osborne Reynolds did his experiments with dye in water flows through tubes in 1883, he noticed that the threshold across which the flow regime went from organized to chaotic was of around 3000. Still according to KUNDU *et al.* (2016) the critical Re for the change of flow regime on the boundary layer on a flat plate is around 10^6. Figure 2.1 illustrates the regime transition over a flat plate, in this case the plate is inclined of 1° and Reynolds number is of 10^6.

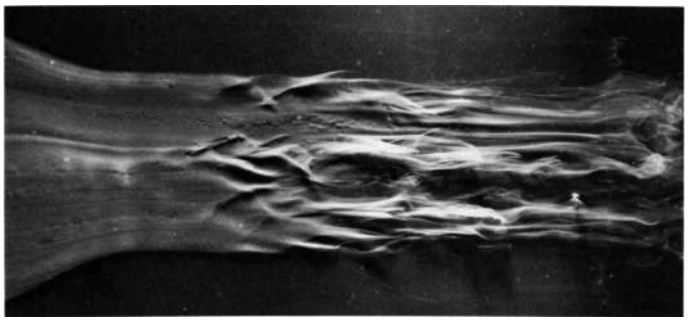


Figure 2.1: Transition on flow regime on inclined plate
Taken from VAN DYKE (1982)

As Re increases and the laminar threshold is crossed, perturbations experienced by the flow are no longer completely dissipated by the viscous effects. These perturbations propagate themselves throughout the flow, as this happens they grow and modify the flow behaviour.

A main aspect of turbulence is that it's inherently three-dimensional. According to WILCOX (2006) the velocities fluctuations in the flow are maintained by the intense stretching of its non-parallel vortex lines. This vortex stretching mechanism is nonexistent in two-dimensional problems. Although three-dimensionality always occurs, the flow perturbations that give rise to the turbulent regime may be two-dimensional (TENNEKES e LUMLEY, 1972). Depending on slightly different initial or boundary conditions, these perturbations may give rise to completely different flow aspects, in dynamical system's this characterizes chaos, which is another challenging aspect of turbulent flows.

Another extremely complex aspect of turbulence is its wide range of scales. In turbulent regime, mechanical energy is passed from the largest scales continuously to intermediate scales until the smallest ones are reached. The largest eddies in the turbulent flow are as big as the flow's characteristic length L (TENNEKES e LUMLEY, 1972), but it's in the smallest ones that the transferred kinetic energy is dissipated by molecular viscosity. Figure 2.2 illustrates the scales difference in a turbulent boundary layer, note how the length of the large eddies are of the same order of magnitude as the boundary layer thickness L_t.

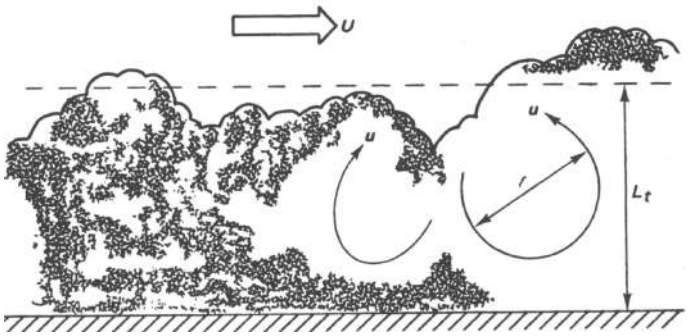


Figure 2.2: Large and small eddies in a turbulent boundary layer over a flat plate
Taken from TENNEKES e LUMLEY (1972)

This process of intense energy transfer throughout different sized structures is commonly denoted as energy cascade, it was first exposed by RICHARDSON (1922) in his book on numerical weather prediction. In the book the author states:

> We realize thus that: big whirls have little whirls that feed on their velocity, and little whirls have lesser whirls and so on to viscosity - in the molecular sense.

Even though inferring the length of the largest eddies in the flow is quite obvious, estimating the length of the lesser scales is not as simple. After RICHARDSON (1922) introduced the cascade process, the characteristic length of energy dissipation was the object of study of KOLMOGOROV (1941). In this seminal work the microscales of dissipation in turbulent flows were estimated through dimensional analysis. On the smallest scale, where dissipation occurs, inertial and viscous effects are equivalent, that is, Reynolds number is unitary. On equation 2.2 the subscript k indicates that the quantities refer to the Kolmogorov microscale.

$$Re_k = \frac{v_k l_k}{\nu} = 1 \tag{2.2}$$

Since turbulent dissipation (ϵ) is equivalent to the rate of supplied kinetic energy (k) its dimension must be as described by equation 2.3a.

$$\epsilon = -\frac{dk}{dt} = \left[\frac{m^2}{s^3}\right] \tag{2.3a}$$

$$\epsilon \sim \frac{v_k^3}{l_k} \tag{2.3b}$$

Taking into account that Re is unity, the characteristic velocity becomes $v_k = \nu/l_k$, substituting this in equation 2.3b results that the Kolmogorov microscale length is as described by equation 2.4.

$$l_k \sim \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}} \tag{2.4}$$

From the result obtained in equation 2.4 and from 2.2 the Kolmogorov microscale characteristic velocity (v_k) and time (t_k) can be estimated.

$$v_k \sim (\nu\epsilon)^{\frac{1}{4}} \tag{2.5a}$$

$$t_k \sim \left(\frac{\nu}{\epsilon}\right)^{\frac{1}{2}} \tag{2.5b}$$

Equation 2.6a brings the turbulent dissipation in terms of the flow's largest scales. Combining it with equation 2.3b leads to the conclusion that the ratio between largest and smallest scales is a function of the Reynolds number.

$$\epsilon \sim \frac{U^3}{L} \tag{2.6a}$$

$$\frac{L}{l_k} \sim L\left(\frac{\epsilon}{\nu^3}\right)^{\frac{1}{4}} \sim \left(\frac{L^3U^3}{\nu^3}\right)^{\frac{1}{4}} \tag{2.6b}$$

$$\frac{L}{l_k} \sim Re^{\frac{3}{4}} \tag{2.6c}$$

To illustrate the huge range of scales present in a same turbulent flow the the ratio of scales for the flow over the inclined flat plate on figure 2.1 can be calculated. On this case the Reynolds number is of 10^6, therefore the ratio L/l_k is of approximately 3.2×10^4. In a situation where a flat plate measures 1m, with the same Re the scale of the smallest eddy on the flow is of around 0.03mm.

On a Direct Numerical Simulation of the Navier-Stokes (NS) equations the mesh size needs to be of at least the same length as the Kolmogorov scale, since all the physics of the flow must be captured by the simulations. The example of flow over the flat plate demonstrates how difficult simulating even reasonably simple flow geometries can be.

Along with all the listed complexities of turbulent flows, the requirement of extremely refined meshes for solving the NS equations numerically, with the computational strength currently available, makes DNS simulations impracticable, with few exceptions. Taking into account that engineering applications require rapidly repeating simulations while varying conditions and geometry, additional mathematical modelling is required to simplify and make turbulence simulations feasible.

2.1 Mathematical Modelling of Turbulence

Further mathematical modelling of turbulence departs from the fact that it can be treated statistically. Knowing that turbulence is chaotic, as mentioned before, and that small variations in initial conditions can produce a completely different velocity field, DAVIDSON (2015) proposes an experiment with the flow around a cylinder. In this experiment the streamwise velocity $u(x,t)$ is measured at a location x_0 downstream of the cylinder. Independently of how many times this measurement is made along time, the results are always different.

However random $u(x_0,t)$ seems to be, its mean over a long enough period of time is always the same. This indicates that the velocity field can be decomposed on a time average plus a fluctuation, as described by equation 2.7. The time averaging over a period T of a given flow quantity $q(x,t)$ is described by equation 2.8, some useful properties of the time averaging procedure are described by equations 2.9.

$$u_i(x,t) = \bar{u}_i(x) + u_i'(x,t) \tag{2.7}$$

$$\bar{q}(x) = \lim_{T\to\infty} \frac{1}{T} \int_0^T q(x,t)dt \tag{2.8}$$

$$\overline{q_1 + q_2} = \bar{q_1} + \bar{q_2}, \quad \overline{\bar{q_1} q_2} = \bar{q_1}\bar{q_2}, \quad \overline{\frac{\partial q_1}{\partial s}} = \frac{\partial \bar{q_1}}{\partial s}, \quad \overline{q'} = 0 \tag{2.9}$$

Figure 2.3 depicts the flow past a cylinder as proposed by DAVIDSON (2015) and two measurement realizations for the streamwise velocity as well as the time averaged velocity.

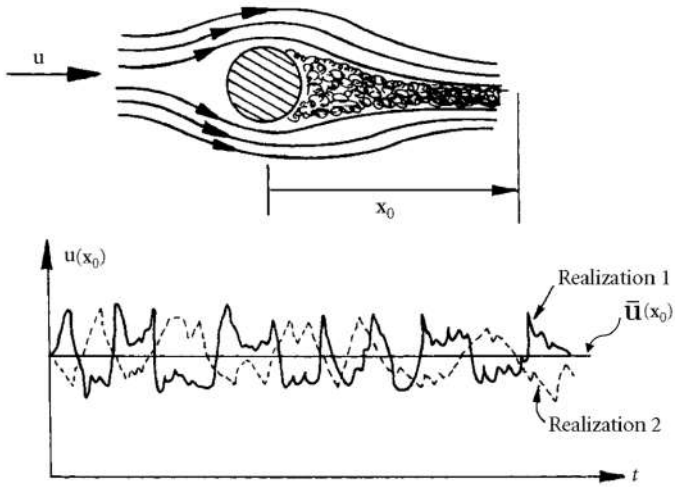


Figure 2.3: Flow past cylinder, two realizations of $u(x_0,t)$ and time average $\bar{u}(x_0)$
Adapted from DAVIDSON (2015)

Considering a flow with an incompressible fluid, the Navier-Stokes equation may be modified into its conservation form 2.10a. Mass conservation is described on equation 2.10b.

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(u_i u_j\right) = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{2.10a}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.10b}$$

Expressing the velocity field and pressure as the sum of a mean and its fluctuating counterpart such as described by equation 2.7 and substituting them on NS equations, results

$$\frac{\partial}{\partial t}\left(\overline{u_i}+u_i'\right)+\frac{\partial}{\partial x_j}\left[(\overline{u_i}+u_i')(\overline{u_j}+u_j')\right] = -\frac{1}{\rho}\frac{\partial}{\partial x_i}\left(\bar{p}+p'\right)+\nu\frac{\partial^2}{\partial x_j \partial x_j}\left(\overline{u_i}+u_i'\right) \tag{2.11}$$

Expanding all the terms on equation 2.11 and taking a time average of the whole equation yields

$$\frac{\partial}{\partial t}\left(\overline{\overline{u_i}+u_i'}\right)+\frac{\partial}{\partial x_j}\left(\overline{\overline{u_j}u_i'+\overline{u_i}u_j'+\overline{u_i}\,\overline{u_j}+u_i'u_j'}\right) = -\frac{1}{\rho}\frac{\partial}{\partial x_i}\left(\overline{\bar{p}+p'}\right)+\nu\frac{\partial^2}{\partial x_j \partial x_j}\left(\overline{\overline{u_i}+u_i'}\right) \tag{2.12a}$$

$$\frac{\partial}{\partial x_j}\left(\bar{u}_i\bar{u}_j+\overline{u_i'u_j'}\right) = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i}+\nu\frac{\partial^2 \overline{u_i}}{\partial x_j \partial x_j} \tag{2.12b}$$

This process is known as Reynolds averaging. Rearranging the terms on the mean momentum equation 2.12b and carrying out the same operations analogously on the continuity equation 2.10b results in equations 2.13a and 2.13b, respectively. These are known as the Reynolds-averaged Navier-Stokes equations.

$$\frac{\partial \bar{u}_i\bar{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\nu\frac{\partial \overline{u_i}}{\partial x_j}-\overline{u_i'u_j'}\right) \tag{2.13a}$$

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \tag{2.13b}$$

Comparing the mean momentum equation 2.13a with its usual instantaneous form 2.10a two main differences are noted. First, the time derivative disappears on the RANS form, this makes sense since time averages covering a long period should not vary with time. In case it happened this would mean that the period T was not long enough. Second, an extra term shows up on the right side of the equation, this new term $-\overline{u_i'u_j'}$ has the dimension of [length^2/time^2] therefore it is interpreted as an additional stress. These new stresses are normally called specific Reynolds turbulent

stress tensor **R**. According to DAVIDSON (2015) the Reynolds stress tensor is not a conventional stress in the word's usual sense, in reality these velocity fluctuations represent the momentum fluxes induced by the mean flow.

When dealing with the usual forms of Navier-Stokes and mass conservation, there are 4 equations and 4 unknowns, in cartesian coordinates u, v, w and p, therefore a determined system. In these new mean equations there are still 4 equations and the 4 mean flow properties but there are also the 6 additional Reynolds tensor components, which turns the system of differential equations in an undetermined system. This inability to solve the RANS equations is widely known as the closure problem of turbulence. Solving any problem through RANS equations requires additional differential equations.

2.1.1 The Boussinesq Hypothesis and The $\kappa - \epsilon$ Model

One of the simplest intuitions one can have regarding the turbulent stress tensor **R** is that, like the deviatoric part of the specific stress tensor τ, it may be proportional to the strain rate tensor **S**. The proportionality parameter between τ and **S** is the molecular kinematic viscosity ν, in the turbulent stress correlation an eddy viscosity ν_t is proposed. Equation 2.14a and 2.14b expose the correlation between both stress tensors and the strain rate tensor.

$$\tau_{ij} = 2\nu S_{ij} = \nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.14a}$$

$$R_{ij} = 2\nu_t \overline{S_{ij}} = \nu_t\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) \tag{2.14b}$$

The turbulent stress tensor proposed in equation 2.14b has an intrinsic problem, its trace will be $-\overline{u_i' u_i'} = 2\nu_t \frac{\partial u_i}{\partial x_i}$. Since the right hand side coincides with the continuity equation this would mean that $\overline{u_i' u_i'} = 0$ which is not reasonable. The most common correction to this problem is shown in equation 2.15, where $\kappa = \frac{1}{2}\overline{u_i' u_i'}$ is the turbulent kinetic energy.

$$R_{ij} = \nu_t\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) - \frac{2}{3}\kappa\delta_{ij} \tag{2.15}$$

The first to propose this correlation between turbulent stresses and mean strain rate was BOUSSINESQ (1877), therefore the reason why it's commonly called Boussinesq's hypothesis. Due to the fact that ν_t has the dimension [length^2/time] the most simple solution to calculate ν_t involves de multiplication of a characteristic length and a velocity, this is commonly referred as Prandtl's mixing-length theory (DAVIDSON, 2015). This solution was also proposed by BOUSSINESQ (1877) after obtaining the expression for the turbulent stress tensor.

There are numerous ways of calculating the turbulent viscosity, known as turbulence models. Namely, the models generally are categorized in algebraic formulations like Prandtl's mixing-length, formulations that involve solving one differential equation, while others require solving two differential equations. There are also models that directly model the Reynolds stress tensor and therefore do not follow the Boussinesq's hypothesis.

The $\kappa - \epsilon$ Model

Two-equation models are probably the most employed with engineering purposes. According to WILCOX (2006) $\kappa - \epsilon$ model is as well known as Prandtl's mixing-length theory, and was the most used model by the end of the twentieth century. Although many other models have been popularized and new ones formulated, as of today it probably remains one of the most used ones. Although other previous works proposed similar ones, it was JONES e LAUNDER (1972) that established the fundamental basis to this portion of turbulence modeling. Later LAUNDER e SHARMA (1974) applied it to a flow around a spinning disc and settled the widely renowned and used *Standard* $\kappa - \epsilon$.

Transport equations for the turbulent kinetic energy κ and the turbulent dissipation rate ϵ are proposed, the eddy viscosity is then calculated using these two obtained quantities. Although the model has many limitations, MOHAMMADI e PIRONNEAU (1993) affirms it provides reasonable results due to its good range of generalization and cheap computational cost.

There are multiple available derivations in literature for turbulent kinetic energy's transport equation. MOHAMMADI e PIRONNEAU (1993) follows the same steps as JONES e LAUNDER (1972), through the manipulations of NS equation 2.10a and RANS equation 2.13a with further time averaging steps, the κ transport equation is obtained. On the other hand WILCOX (2006) derives it by first finding a differential equation that directly models $\mathbf{R}$ and then taking the tensorial equation's trace. In this work, for simplicity and conciseness, the κ equation 2.16 will not be derived and will be used directly as given by WILCOX (2006).

$$\frac{\partial \kappa}{\partial t} + \overline{u_j}\frac{\partial \kappa}{\partial x_j} = \nu_t \frac{\partial \overline{u_i}}{\partial x_j}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\kappa}\right)\frac{\partial \kappa}{\partial x_j}\right] - \epsilon \tag{2.16}$$

The dissipation rate equation 2.17, for the same reason as the κ, will not be derived in this work. The procedure carried out to obtain it can be checked at numerous sources like MOHAMMADI e PIRONNEAU (1993) and WILCOX (2006).

$$\frac{\partial \epsilon}{\partial t} + \overline{u_j}\frac{\partial \epsilon}{\partial x_j} = C_{\epsilon 1}\frac{\epsilon}{\kappa}\nu_t \frac{\partial \overline{u_i}}{\partial x_j}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right] - C_{\epsilon 2}\frac{\epsilon^2}{\kappa} \quad (2.17)$$

Once it's possible to calculate the scalars κ and ϵ, eddy viscosity can be written as a function of both, shown in equation 2.18.

$$\nu_t = C_\mu \frac{\kappa^2}{\epsilon} \quad (2.18)$$

The constants σ_κ, used in equation 2.16, σ_ϵ, $C_{\epsilon 1}$ $C_{\epsilon 2}$, used in 2.17, and C_μ used in the eddy viscosity formula 2.18 are calibrated using experimental data. Table 2.1 shows their values as proposed by LAUNDER e SPALDING (1974), different values for these constants may be encountered across literature. An example of such variations is that these on table 2.1 are different from those first proposed by JONES e LAUNDER (1972).

Table 2.1: $\kappa - \epsilon$ model's constants values

C_μ	$C_{\epsilon 1}$	$C_{\epsilon 2}$	σ_κ	σ_ϵ
0.09	1.44	1.92	1.0	1.3

Chapter 3

Neural Networks

Machine learning is a generic term to describe a series of mathematical manipulation of data by algorithms, in order to either group or map the input data into a desired output, or simply finding patterns across the inputs.

When mapping or grouping into targeted outputs the process is referred to as supervised learning, when no specific output is desired rather than better understanding the data through pattern recognition, the procedure is named unsupervised learning (DURAISAMY *et al.*, 2019). Other two categories of algorithms are cited by MARSLAND (2011), reinforcement learning, a hybrid between unsupervised and supervised, and evolutionary learning, alike biological evolution it uses the concept of output's fitness.

Naming these numerical operations as a learning process comes from the fact that, similarly to humans and animals in general, the algorithms enhance themselves through experience and repetition. Some popular machine learning techniques reviewed by KOTSIANTIS *et al.* (2006) include: Decision Trees, Neural Networks, Random Forests (RF), Bayesian Newtorks (BN) and Support Vector Machines (SVM).

Artificial Neural Networks were conceived loosely based on the nature of the human brain, an attempt to computationally model the brain architecture. According to BISHOP (2006), this ML category now comprises many different models. The plausibility of the biological analogy and how well these models capture it have been considerably exaggerated over time. This is not necessarily a downside of their functioning, on the contrary, the method has proven its wide utility and its popularity is on the rise.

The fundamental structure in the NN is the neuron, its first mathematical model was proposed by MCCULLOCH e PITTS (1943) and stablished the basis for the modern Artificial Neural Networks (ANN). Figure 3.1 depicts the typical modern modeled neuron.

This structure works by receiving vectorized m inputs $\mathbf{x} = [x_1, x_2, ..., x_m]$, each

input is multiplied by a corresponding weight, also organized in a vector $\mathbf{w}_k = [w_{k1}, w_{k2}, ..., w_{km}]$. Following, the product of all the multiplications are summed with a bias b_k, the result of the summation is passed through a function $\varphi(\upsilon_k)$. This function is called the activation function and its choice depends on the application of the NN.

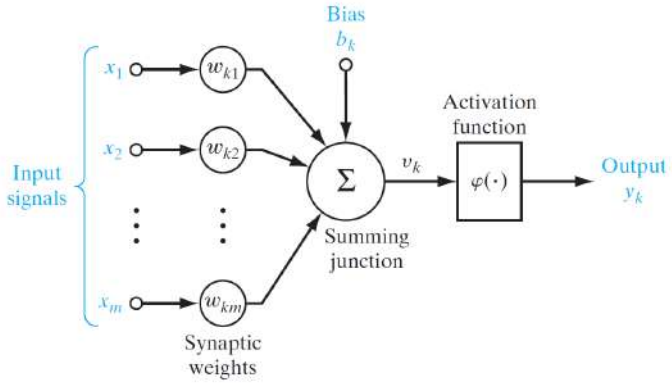


Figure 3.1: Model of neuron used in Artificial Neural Networks, labeled as k
Taken from HAYKIN (2009)

Equations 3.1 summarize the operations carried out on a single neuron. Note that the weights $\mathbf{w}_k$ and the bias b_k are functions of the neuron, in a NN with multiple neurons each one has its own set of $\mathbf{w}_k$ and b_k.

$$\upsilon_k = \mathbf{w}_k^T \cdot \mathbf{x} + b_k \tag{3.1a}$$

$$y_k = \varphi(\upsilon_k) \tag{3.1b}$$

The model presented by MCCULLOCH e PITTS (1943) has a main difference to the modern one depicted in figure 3.1, its activation function is a step function. If the inputed summation does not exceed a certain threshold this neuron's model output is zero, if the threshold is exceeded the output is one.

Further development to this type of modeling was presented by ROSENBLATT (1958), where what now is categorized as supervised learning was first presented (HAYKIN, 2009). In ROSENBLATT (1958) learning is first discussed as way to adjust the neuron's weights and bias in classification problems, this employment of a neuronal structure is named perceptron. Later ROSENBLATT (1961) demonstrated the *Principal Convergence Theorem*, proving that the perceptron is able to correctly classify linearly separable patterns through the proposed error correction algorithm.

According to HAYKIN (2009), following up ROSENBLATT (1961), a number

of works identified a series of limitations of the perceptron and disencouraged the further development and exploration of neural networks in general for nearly two decades. The most compromising shortcoming of the perceptron was the proven inability to correctly generalize its results based on the inputed learning data.

More recently it has been widely demonstrated that the use of many combined perceptrons, arranged into multiple layers can achieve results previously unobtainable thorugh the use of single or few combined perceptrons (HAYKIN (2009), BISHOP (2006), MARSLAND (2011)). This new structure of multiple perceptrons joined together in multiple connected layers is called the Multi-Layer Perceptron (MLP).

3.1 The Multi-Layer Perceptron and Forward Propagation

As mentioned before, the MLP is divided into layers, each of these composed by a number of neurons N. Layers are subdivided into three categories: input, hidden and output layers. The input layer does not perform any calculations nor has any neurons, it's a layer composed only by the inputs. The output layer has neurons and performs calculations, it is in fact the last layer and its results are the outputs of the NN.

Hidden layers are the intermediate ones, they can be multiple, ranging from a single hidden layer to hundreds of them, depending on the application and the programmer's choice. The number of layers on a NN is commonly denoted L. There are also networks with no hidden layers, that is, all the calculations are performed only in the output layer.

A network with multiple layers functions as follows, subscripts k denote the k-th neuron and superscripts (l) denote the l-th layer.

1. The first hidden layer $l = 1$ receives the inputs $\mathbf{x}$ from the input layer

   (a) Every neuron in this hidden layer receives all of the inputs and they're multiplied by the neuron's weights $\mathbf{w}_k^{(1)}$.

   (b) The results of these multiplications are then summed along with the bias $b_k^{(1)}$, forming the scalar $v_k^{(1)}$ as described by equation 3.1a.

   (c) The scalar $v_k^{(1)}$ is passed through the neuron's activation function $\varphi(v_k^{(1)})$, resulting in the neuron's output $\bar{y}_k^{(1)}$ as described by equation 3.1b.

   (d) The outputs of every neuron in the first hidden layer form the layer's output vector $\bar{\mathbf{y}}^{(1)} = [\bar{y}_1^{(1)}, \bar{y}_2^{(1)}, ..., \bar{y}_k^{(1)}]$.

2. The previous layer's outputs $\bar{\mathbf{y}}^{(1)}$ become the input for the following hidden layer $l = 2$. The same procedures are repeated in this layer until output $\bar{\mathbf{y}}^{(2)}$ is calculated.

3. For all subsequent hidden layers the procedure described in item 1 is repeated analogously until the output layer is reached

4. In the output layer the procedure is equally repeated, this time the layer's outputs configure the final MLP's output $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_k]$

This process is known as forward propagation (FP), note that each neuron processes all the inputs its layer receives from the previous one, this configures what is commonly called a fully connected neural network. Figure 3.2 shows an example of a NN with four inputs, one hidden layer with three neurons and three neurons in the output layer. For clarity and better understanding, figure 3.2 does not show a fully connected NN, since including all the connections depicted by the arrows would make it polluted.

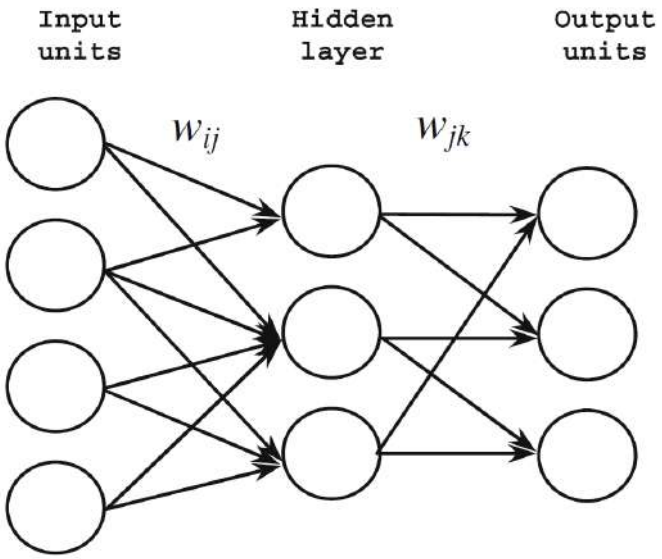


Figure 3.2: Neural Network with 6 neurons and a single hidden layer
Taken from KOTSIANTIS *et al.* (2006)

3.2 Network Training and Back-Propagation

The error correction algorithms proposed in ROSENBLATT (1961) require direct error measurements between outputs $\hat{\mathbf{y}}$ and target values $\mathbf{y}$, this is feasible when using single perceptrons or few ones organized into a single layer, since their outputs are the network's outputs.

This is not possible when dealing with a Multi-Layer Perceptron because one can only measure the error of the output layer. There are no targeted values for the hidden layers' outputs. According to MARSLAND (2011) this is the reason why intermediate layers are called hidden, because their oututs are not usually seen and there are no metrics to measure their error, there is in fact no associated error.

The solution to circumvent the impossibility of adjusting the NN parameters through direct error corrections is called Back-Propagation (BP), or Back Propagation of Error (MARSLAND, 2011). This procedure uses an error function to be minimized throughout subsequent updates of the weights and biases of the network. These adjusts utilize the function's gradient with respect to the network's parameters.

This error measurement is usually called loss function, and will be denoted by $J(\hat{\mathbf{y}}, \mathbf{y})$. Since the gradient is computed with respect to the the network's parameters, this implies that the derivatives of the chosen activation functions are known and continuous.

A widely used loss function is the sum of the squares error, as shown in equation 3.2, where n is the number of samples.

$$J(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{2}\sum_{k=1}^{n}(\hat{\mathbf{y}} - \mathbf{y})^2 \tag{3.2}$$

The back-propagation scheme known as gradient descent, or stochastic gradient descent (SGD), as described by BISHOP (2006) is summarized below. Subscripts k and j denote respectively the k-th neuron and the j-th input received by this neuron. Superscript (l) denotes the l-th layer.

For conciseness the involved quatities, *e.g.* the gradient's components and others, will be given in formulas, their deductions are explained in detail in multiple sources like BISHOP (2006), HAYKIN (2009) and MARSLAND (2011).

1. The gradient of the loss function ∇J with respect to all weights $\mathbf{w}_k^{(l)}$ and biases $b_k^{(l)}$, as shown in equation 3.3, is computed.

$$\nabla J = \left[\frac{\partial J}{\partial w_{11}^{(1)}}, \frac{\partial J}{\partial w_{12}^{(1)}}, ..., \frac{\partial J}{\partial w_{1j}^{(1)}}, \frac{\partial J}{\partial b_1^{(1)}}, ..., \frac{\partial J}{\partial w_{\mathrm{kj}}^{(l)}}, \frac{\partial J}{\partial b_k^{(l)}}\right] \tag{3.3}$$

   (a) A delta function $\delta_k^{(l)}$, presented in equation 3.4, is defined to simplify the calculation of the derivatives.

$$\delta_k^{(l)} = \begin{cases} \varphi'(v_k^{(l)})(\hat{y}_k - y_k), & \text{if } l = L \\ \varphi'(v_k^{(l)})\sum_{b=1}^{N}(\delta_b^{(l+1)} w_{\mathrm{bk}}^{(l+1)}), & \text{if } 1 \leq l < L \end{cases} \tag{3.4}$$

(b) The derivatives with respect to the weights are calculated following the formula in equation 3.5. The quantity $\varphi'(v_k^{(l)})$ is the derivative of the activation function with respect to $v_k^{(l)}$.

$$\frac{\partial J}{\partial w_{kj}^{(l)}} = \bar{y}_j^{(l-1)} \delta_k^{(l)} \tag{3.5}$$

(c) The derivatives of J with respect to the biases are calculated via equation 3.6.

$$\frac{\partial J}{\partial b_k^{(l)}} = \delta_k^{(l)} \tag{3.6}$$

2. After all derivatives have been computed, the weights $w_{kj}^{(l)}$ and biases $b_k^{(l)}$ can be updated via the following equations 3.7a and 3.7b, respectively.

$$w_{kj}^{(l)} = w_{kj}^{(l)} - \alpha \frac{\partial J}{\partial w_{kj}^{(l)}} \tag{3.7a}$$

$$b_k^{(l)} = b_k^{(l)} - \alpha \frac{\partial J}{\partial b_k^{(l)}} \tag{3.7b}$$

It can be noted in equation 3.4 that the delta function is recursive. In order to calculate $\delta_k^{(l)}$ in a layer l the function's value in the following layer $\delta_k^{(l+1)}$ is required. Therefore, the algorithm must begin in the output layer and advance towards the first hidden layer. This characteristic is the reason why the scheme is designated back-propagation of error.

The negative signs in the update rules in equations 3.7 indicate that the parameters will be updated in the direction of the greatest decrease in the function, therefore minimizing it.

The parameter α is a learning rate and works the same way as under-relaxation does in the numeric solution of differential equations. Ranging from zero to unity, this parameter defines if the network's learning speed will be increased, in case of higher values, or decreased. The way it affects the training depends on the application, in some a higher α may take $\mathbf{w}$ and b values to local minimum of J that would not be reached by a lower value. In other cases higher α may simply impossibilitate the network to settle in any minimum at all.

There are alternate optimization techniques based on gradient descent, an example is the *Adam* algorithm introduced in KINGMA e BA (2014). *Adam* uses moving averages of the gradient and the gradient squared to adapt the learning rate throughout the training. These averages are known as moments, and their use have been proved beneficial for the training stage.

3.3 Hyper-Parameters

During this entire chapter many of the aspects of the Neural Network are listed as an application dependent choice. So far these parameters were the number of layers L, the number of neurons per layer N, the activation function $\varphi(v)$ and the learning rate α. They are defined as the hyper-parameters of a network, in order to avoid confusion with the parameters $\mathbf{w}$ and b.

The appropriate choice of hyper-parameters is fundamental to the network in a way that small differences might lead to completely different results. KOTSIANTIS *et al.* (2006) highlights that the appropriate choice of L and N can be troubling, excessive number of neurons can impair results by overfitting them, that is, the network works exceptionally well in the training steps but performs poorly on other applications. While picking less neurons than necessary will lead to poor generalization capability and bad results also.

Other common hyper-parameters are the number of epochs and batch sizes. According to MARSLAND (2011) an Epoch is defined as a complete FP and BP performed over all the training dataset. The duration of an epoch, depends on how many data points will be inputed in the NN before their associated errors are back-propagated. This amount of data points passed in forward propagation before the backward propagation is computed is defined as a batch size, or mini-batch size.

The batch size defines the duration of an epoch. A batch size can vary from a single data point to an entire dataset. Smaller batch-sizes make training slower since a lot more updates will be necessary in the same epoch. Consequently, many more epochs will occur. For example, if a given dataset's batches are composed by a tenth of its data points, a single epoch would be composed of ten parameters updates.

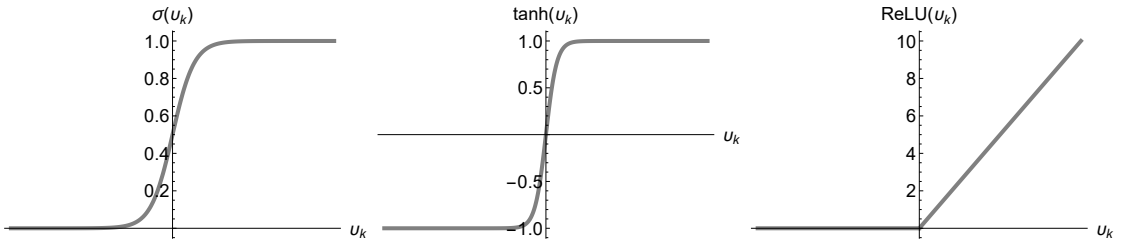
Dividing training into epochs is important since one of the simplest ways to determine when to interrupt training is to set a maximum number of epochs.

As mentioned before, the activation functions must meet two requirements, they must be continuous as well as their derivatives. Popular activation functions are the sigmoid function $\sigma(x)$, which ranges from $[0, 1]$, the hyperbolic tangent function $\tanh(x)$, which ranges from $[-1, 1]$, and the ReLU function, ranging from $[0, \infty]$. These three functions are presented in equations 3.8 and their behaviour is displayed at figure 3.3.

$$\sigma(v_k) = \frac{1}{1 + e^{-v_k}} \tag{3.8a}$$

$$tanh(v_k) = \frac{e^{v_k} - e^{-v_k}}{e^{v_k} + e^{-v_k}} \tag{3.8b}$$

$$\mathrm{ReLU}(v_k) = \max(0, v_k) \tag{3.8c}$$



(a) (b) (c)

Figure 3.3: (a) $\sigma(v_k)$ (b) $\tanh(v_k)$ (c) $\mathrm{ReLU}(v_k)$

Choosing the best hyper-parameters can take considerable time, turning into a heuristic choice that is dependent on the programmer's experience, size of the available training dataset, desired speed of network training and complexity of the problem. The nature of this search can be so intricate that this procedure often described as hyper-parameter tuning.

3.4 Train, Validation and Test Stages

One of the most common issues in the use of NN occurs when the network memorizes the training dataset instead of correctly assimilating the data in order to generalize its aspects. When this happens the network makes poor predictions to any other unknown data inputed into it. This configures overfitting of the network (BISHOP, 2006).

To avoid this major problem it is usual to split the available dataset into three different groups, the train, validation and test datasets. The training stage uses the correlated dataset to perform back-propagation and adjust all of the parameters. While training is in progress, the validation dataset is used by the network not to update any of the parameters, but to check what is the value of the loss function when using this new data.

With this new measurement it is possible to verify how both training and validation losses are evolving throughout learning. Usually both values decrease alogside, with the training loss ideally being a little lower than the validation one. After some time in the learning process the validation loss will either stop decreasing and reach a plateau or even start increasing, while the training loss will remain decreasing continuously. This behavior indicates that the network is overfitting and that training should be interrupted. Therefore, using a validation set is a useful manner to efficiently know when to stop training.

The test dataset, in opposition to te validation and train datasets, is never seen by the network. It is interesting to keep a part of the available data to evaluate the fitness of the trained network. Since the results associated to this dataset are known it is possible to evaluate how well this network may perform on never before seen data.

Keeping a separate unevaluated dataset is also useful from the perspective that it allows the evaluation of multiple trained networks. Knowing how well the network performs on this testing data makes it possible to pick the best one among the multiple networks. This is important since every network has some randomness within it, due to the fact that before training their parameters are initialized with random values.

There is no rule to splitting the available dataset between train, validation and test, it depends on how much data one possesses. It is a good practice to split in reasonable percentages that give the network enough data to correctly learn while also efficiently validate and test it.

Chapter 4

Machine Learning Assisted Turbulence

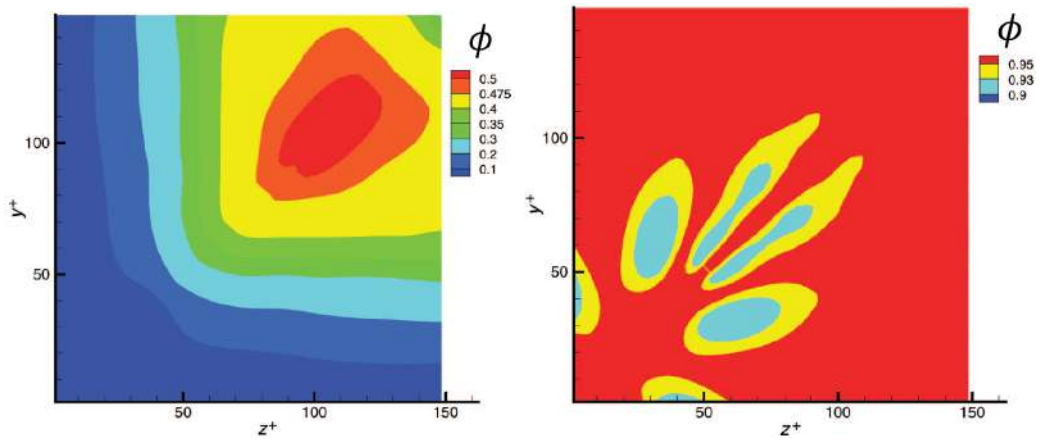
Although RANS simulations have been proved very useful, their results are still limited. The Boussinesq hypothesis is not suitable for a number of scenarios, namingly flows with mean streamline curvature, secondary flows, sudden changes in mean strain-rate (WILCOX, 2006), adverse pressure gradients and flows with separation and reattachment (THOMPSON *et al.*, 2019).

In THOMPSON *et al.* (2010) other constitutive relations for the deviatoric part $\mathbf{B}$ of the Reynolds stress are evaluated. A metric for measuring the importance of a second order tensor with respect to another is used to evaluate the linear eddy-viscosity assumption and other more complex constitutive relations. This is measured by an index ϕ proposed by THOMPSON (2008). The index ϕ is shown in equation 4.1, it varies from zero to unity, with zero indicating that both tensors are totally non-proportional and one indicating that they keep proportionality.

In equation 4.1 $\mathbf{R_{mod}}$ is a modeled Reynolds stress and $\mathbf{R_{HF}}$ is a high-fidelity one.

$$\phi = 1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\|\mathbf{R_{mod}}\|}{\|\mathbf{R_{HF}}\|}\right) \tag{4.1}$$

Figure 4.1 shows two scenarios of ϕ in a flow on a square-duct, there DNS is used as the high-fidelity data. In panel (a) of figure 4.1 the usual linear eddy-viscosity assumption is evaluated, panel (b) depicts the performance index ϕ for the model $\mathbf{B} = \alpha_0\mathbf{I} + \alpha_1\mathbf{S} + \alpha_2\mathbf{S}^2 + \beta\mathbf{P}$ proposed by THOMPSON *et al.* (2010), where α and β are scalar coefficients. The tensor $\mathbf{P}$ is the nonpersistence-of-straining tensor (THOMPSON, 2008), which indicates how persistent is the local strain in the flow. As the tensor's intensity grows the less persistent is the straining.



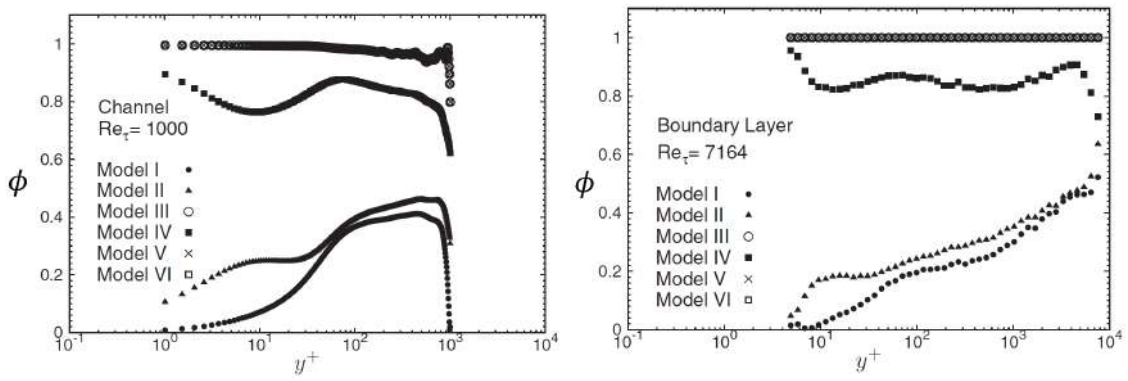
(a) $\mathbf{B} = \alpha_1 \mathbf{S}$ (b) $\mathbf{B} = \alpha_0 \mathbf{I} + \alpha_1 \mathbf{S} + \alpha_2 \mathbf{S}^2 + \beta \mathbf{P}$

Figure 4.1: Boussinesq hypothesis' performance versus a more complex constitutive relationship performance

Adapted from THOMPSON *et al.* (2010)

Other consitutive relations between mean kinematic quantities and the turbulent stress tensor were studied by NIECKELE *et al.* (2016). There, DNS and experimental data were employed to evaluate the proposed models' performances. The tensorial basis used to constitute $\mathbf{R}$ was based on the mean strain-rate tensor $\mathbf{S}$ and the nonpersistence-of-straining tensor $\mathbf{P}$ (THOMPSON, 2008). A total of six models are evaluated by NIECKELE *et al.* (2016) and their performances were also measured by the index ϕ, in equation 4.1.

Figure 4.2 depicts ϕ for the six models in a plane channel flow and a boundary layer flow. The index ϕ demonstrated in figure 4.2 used DNS data as the high-fidelity source, while in the boundary layer flow experimental data was used.



(a) DNS plane channel flow (b) Experimental boundary layer flow

Figure 4.2: Boussinesq hypothesis' perfomance versus other six more complex models

Adapted from NIECKELE *et al.* (2016)

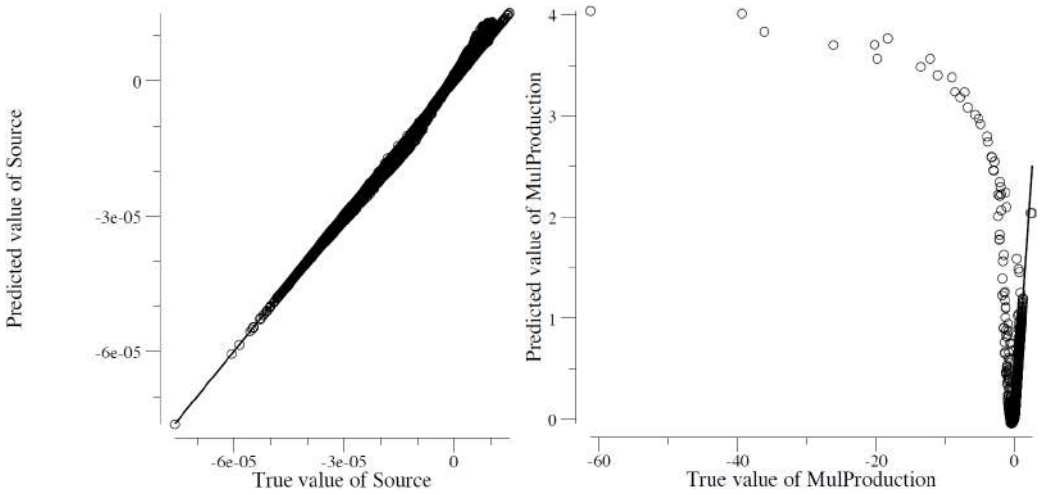
In figure 4.2 the Boussinesq linear eddy-viscosity hypothesis is categorized as

model I, while the model $\mathbf{B} = \alpha_0\mathbf{I} + \alpha_1\mathbf{S} + \alpha_2\mathbf{S}^2 + \beta\mathbf{P}$ proposed by THOMPSON *et al.* (2010) is categorized as model III. The other depicted models are different combinations of tensors $\mathbf{S}$, $\mathbf{S}^2$, $\mathbf{P}$, $\mathbf{P}^2$ and $\mathbf{I}$.

The limitations exposed by THOMPSON *et al.* (2010) and NIECKELE *et al.* (2016), illustrated by figures 4.1 and 4.2, combined with the difficulty in formulating new models that are both stable and computationally cheap, have encouraged the development of alternative strategies to simulate turbulent flows. A promising alternative has been the application of Machine Learning to improve the results of RANS simulations, based on corresponding results from high-fidelity sources, like DNS and LES. The ML strategies, in general, build a functional relationship between mean flow quantities from low-fidelity simulations and a desired quantity from a high-fidelity simulation.

In this novel context, TRACEY *et al.* (2015) was one of the first works to explore the capabilities of predicting turbulent flows' quantities through ML. The authors constructed a two hidden-layer neural network, with 50 neurons per layer, to predict the source term of the Spallart-Allmaras one-equation turbulence model. The SA model consists of a transport equation for the turbulent viscosity ν_t. The source term s included in the model equation, according to TRACEY *et al.* (2015), was built based on modelers experience and calibrated over simple flows. High-fidelity data was not used to train the NN, it instead aimed at a known analytic SA source terms to investigate the NN capacity of correctly predicting a known quantity with a given set of inputs. Promising results were obtained using a reasonable range of different flows. The predicted source terms were, in many cases, capable of successfully being injected into CFD solvers and further propagated to velocity fields and other quantities of interest (QoI), *e.g.* skin-friction coefficient C_f. The use of multiple targeted outputs, input features, loss functions and training datasets in TRACEY *et al.* (2015) evidentiated that the achieved results are highly dependent on the adequate choice of these configurations. It is also highlighted by the authors that predicting reasonably accurate quantities does not guarantee that reasonable results will be achieved when these are injected into the CFD environment and propagated by the PDEs. The inverse is also true, sometimes poor predictions do lead to good posteriori flow results. This obviously is highly dependent on the CFD environment in question.

Figure 4.3 compares predicted versus analytical results of two different source terms. According to TRACEY *et al.* (2015), the almost identical results in panel (a) lead to bad flow solutions, while the deviating result in panel (b) has almost no consequences when propagated. This outlines the importance of coupling ML results with the CFD environment. The quality of results can not solely be judged by direct comparison to true values of yet to be propagated quantities.



(a) (b)

Figure 4.3: (a) Predicted versus true s value, associated to poor propagation, (b) Predicted versus true s values, associated to good propagation
Taken from TRACEY *et al.* (2015)

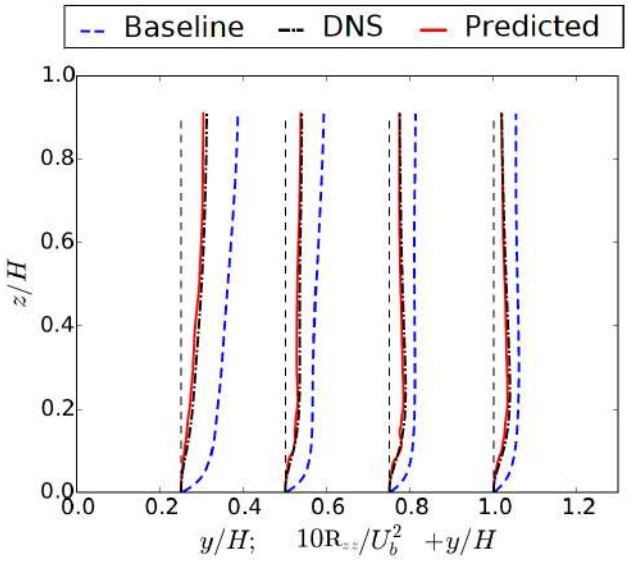
In an attempt to identify regions where the linear eddy-viscosity assumption does not perform well LING e TEMPLETON (2015) also employed ML techniques. Three metrics were selected to identify these problematic regions in a set of different flows, and were first evaluated by direct comparison between RANS and DNS or LES. Following, three different ML algorithms were used to predict these regions, each technique predicted the RANS performance on the three uncertainty metrics. The ML techniques used were the Support Vector Machines, Decision Trees and Random Forests (RF). Predicted regions where Boussinesq hypothesis fails are compared with the previously known results. Results demonstrate that better predictions were made by the RF technique. Since very different class of flows were used to train and test the algorithm, a promising aspect of LING e TEMPLETON (2015) is that it indicates that ML is able to generalize the learned data to flows with considerable different characteristics. It also expands the possibilities of using ML to quantify uncertainties in RANS simulations.

More recently, LING *et al.* (2016) applied deep Neural Networks with embedded Galilean invariance to correct RANS simulations. Being Galilean invariant guarantees that if the coordinates axes are rotated, inputed and outputed quantities will also be, assuring that same flows with different axes orientations will provide the same predictions by the NN. Deep Neural Networks differentiate themselves from usual NN by their large number of hidden layers, in LING *et al.* (2016) 8 hidden-layers with thirty neurons each are used. There, the inputs were an invariant normalized tensor basis, based on the mean strain-rate tensor $\mathbf{S}$ and the mean rotation rate tensor $\mathbf{W}$. The NN is trained to output the normalized deviatoric part

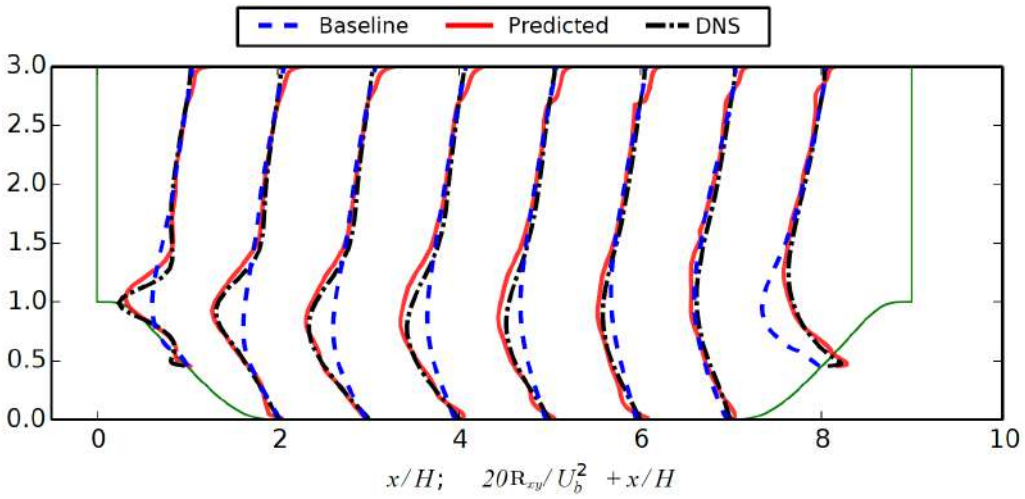
of the Reynolds stress, which is then propagated to a new velocity field, through the mean momentum equations. LING *et al.* (2016) also used a Bayesian optimization technique to tune the hyperparameters α, L and N, this procedue ensured that the built networks achieved more accurate results. Another interesting aspect is that data from DNS and LES simulations from 9 entirely different flows are used to train the network, while it is tested on a square-duct and wavy wall flows. This implementation by LING *et al.* (2016) provided reasonable results when compared to other procedures, a comparison is made with a NN that did not have any embedded invariance, with a linear eddy-viscosity model and a quadratic eddy-viscosity model results. It successfully predicted the secondary motion on a square duct and some recirculation on a wavy-wall channel flow. The quadratic model and non-invariant NN although capable of predicting these phenomena to some extent, have very limited results. This demonstrates that the invariant NN is capable of providing better results than usual models, although they are still not quantitatively comparable to DNS.

Another strategy is employed by WANG *et al.* (2017b) to reconsruct the turbulent stresses. Using RF trained with DNS data the authors predicted the discrepancies between DNS and RANS Reynolds stresses $\Delta\mathbf{R}$. The discrepancies are computed after the tensors have been projected to six quantities. These quantities are Galilean invariant, and represent the magnitude, shape and orientation of the tensor. This new procedure is tested on the square-duct flow and on the flow over periodic hills. The flow over periodic hills is specially interesting to this purpose because it presents large separation regions. These two flows datasets are not mixed together, that is, separate random forest regressors are trained for the different flow geometries. Chosen input features are a set of nine non-dimensional, physically representative quantities. The square-duct RF is trained using different Reynolds numbers, while two separate RF are trained to predict the periodic hills $\Delta\mathbf{R}$, one trained on the same geometry with differents Re and other trained on slightly different hills geometries with different Re as well. Obtained results are good for both cases where training is conducted only on multiple Re cases, the predicted Reynolds stresses in these scenarios are much closer to the high-fidelity test data than the baseline RANS simulations were.

Figure 4.4 shows the comparison between baseline RANS, DNS and predicted components of $\mathbf{R}$ for the different Re training scenario.



(a)



(b)

Figure 4.4: (a) Square-duct predicted component $\mathbf{R}_{zz}$ profiles, (b) Periodic hills predicted component $\mathbf{R}_{xy}$ profiles

Adapted from WANG *et al.* (2017b)

Figure 4.5 illustrates that the RF predictions for slightly different hills geometries in the periodic hills flow is not as good as the results in figure 4.4, obtained via training only on alternate Re.

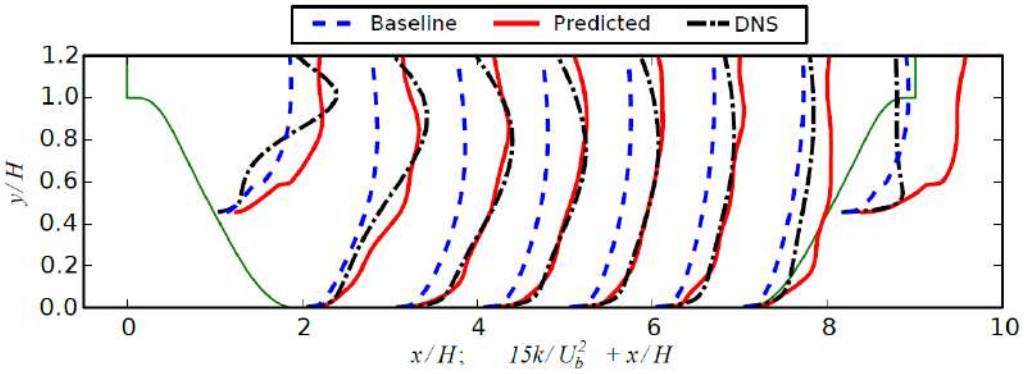


Figure 4.5: Predicted turbulent kinetic energy k profiles with trained using modified geometries

Taken from WANG *et al.* (2017b)

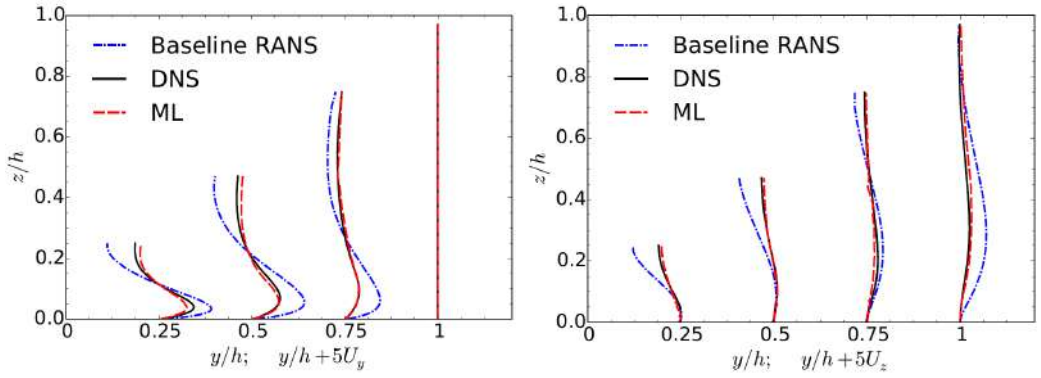
According to WANG *et al.* (2017b) although the results in figure 4.5 are not very good on the whole domain, the methodology is proven efficient since the main interest was predicting the recirculating region. In figure 4.5 the recirculating zone ranges from the profiles in coordinates 2 to 6. The author suggests that the inadequacy on other coordinates of figure 4.5 are due to the lack of data quality in the upper channel region. Moreover, the employed approach suggests that training ML on sets of data from similar geometries might be a better strategy than using wide datasets from flows that are too different from each other. A shortcoming of WANG *et al.* (2017b) is that the predicted stresses are not propagated to the mean flow by the momentum equations. According to the authors it is due to the discrepancies in predicted quantities, especially the ones shown in some regions of figure 4.5.

In this context of employing representations of the discrepancy of $\mathbf{R}$ in the ML framework, WU *et al.* (2017) explored the suitability in representing $\Delta\mathbf{R}$ eigenvectors through Euler angles, relative Euler angles and the unit quaternion. Euler angles describe the current orientation of a given coordinate set with respect to a global coordinate set. If the coordinate axes are respectively the low and high-fidelity Reynolds stress eigenvectors, the Euler angles are said relative, and are invariant. If the Euler angles of both low and high-fidelity $\mathbf{R}$ are correlated to a global coordinate system, this is said an absolute Euler angle and it is not invariant. Unit quaternions are a four component vector that compactly representate a rotation of an angle θ about an axis n, independently of a global coordinate system, therefore invariant. According to WU *et al.* (2017) a suitable representation for ML predictions must be smooth in space and invariant. According to the authors, absolute Euler angles are coordinate dependent and discrepancy-based Euler angles are not spatially smooth. Therefore the most suitable representation of eigenvectors discrepancy to be predicted by ML is provided by the unit quaternion, which is corroborated by

the ML predictions obtained for both unit quaternion and relative Euler angles. As in WANG *et al.* (2017b) predictions are not propagated to mean velocity fields.

WU *et al.* (2018) also used RF on the square-duct and periodic-hills flows. By predicting an optimal eddy-viscosity ν_t and the part of $\mathbf{R}$ that is nonlinear with $\mathbf{S}$. An optimal $\mathbf{R}$ is reconstructed and subsequently propagated to the mean flow field. The baseline RANS simulations were provided by direct Reynolds Stress Tensor Modeling (RSTM). Contrarily to the linear eddy-viscosity models like the κ-ϵ, the RSTM models do predict an exaggerated secondary flow in the square-duct. The applied methodology did succeed on improving the baseline RSTM results. In the periodic hills simulation, ML predictions also led to better mean-flow than the one provided by the baseline RANS simulations, but were not as good as the square-duct results.

Figure 4.6 demonstrates the corrections in the secondary flow performed in WU *et al.* (2018). Although the baseline RSTM does predict recirculation, the ML predictions led the velocity profiles much closer to the DNS data.



(a) (b)

Figure 4.6: (a) u_y velocity component, (b) u_z velocity component
Taken from WU *et al.* (2018)

The problem involving the full reconstruction of the turbulent stresses by the ML algorithms is adressed by CRUZ *et al.* (2019). Since the stresses provenient from the DNS results might not be fully converged like the mean flow $\bar{\mathbf{u}}$, as evidentiated by THOMPSON *et al.* (2016), an alternative path to train ML techniques based on high-fidelity datasets was proposed. Since the mean velocity field of DNS sources do not contain the convergence problems highlighted in THOMPSON *et al.* (2016), an indirect calculation of the Reynolds Force Vector through the manipulation of the mean momentum equations is used to avoid the unconverged DNS $\mathbf{R}$. This procedure is demonstrated in equations 4.2.

$$\mathbf{r} = \nabla \cdot \mathbf{R} \tag{4.2a}$$

$$\mathbf{r} = \bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}} + \frac{1}{\rho} \nabla p - \nu \nabla^2 \bar{\mathbf{u}} \tag{4.2b}$$

Since the DNS square-duct dataset provided by PINELLI *et al.* (2010) does not provide pressure fields, a modified RFV used by CRUZ *et al.* (2019) is demonstrated in equation 4.3.

$$\mathbf{t} = \mathbf{r} - \frac{1}{\rho} \nabla p \tag{4.3}$$

The chosen ML method was a two hidden layers NN, with 100 neurons each, which was trained using four different Re square-duct flows, and used to predict the vector $\mathbf{t}$. An extended tensor basis was inputed into the network, based on the RANS $\mathbf{S}$, $\mathbf{R}$ and $\mathbf{P}$. A vector basis consisting on the divergence of each tensor in the tensor basis was also used as input. In CRUZ *et al.* (2019) only Euclidean invariance was ensured, this means that the rotation and translation of test dataset's coordinate axes would require the training of an alternate network with the same alteration in training dataset coordinates. After completing training, test predictions were injected into the RANS MME in order to correct the mean flow field. Reasonably accurate results were demonstrated, most notably the null secondary flow of RANS simulations is succesfully corrected and showed great agreement with the DNS data, as can be seen in figure 4.7.

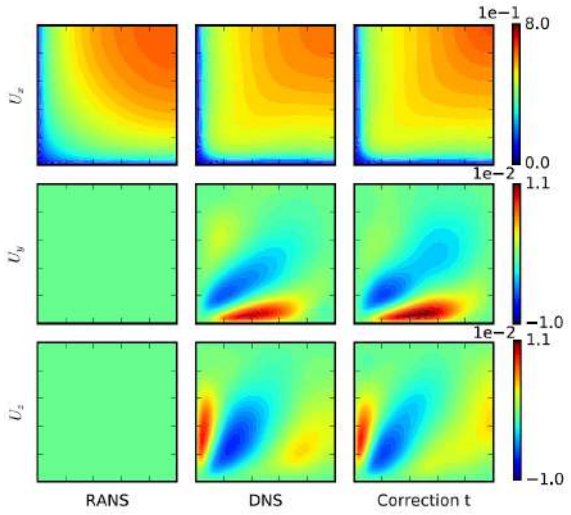


Figure 4.7: Comparison between RANS, DNS, and correction by NN predicted $\mathbf{t}$
Adapted from CRUZ *et al.* (2019)

For comparison, an alternate NN using $\mathbf{R}$ as the target is built to evaluate both methodologies. Figure 4.8 demonstrates that the global error measured by equations 4.4 for the three velocity components are much smaller when using the

RFV methdology. The $\mathbf{R}_{DNS}$ and $\mathbf{t}_{DNS}$ errors correspond to the direct injection of both DNS quantities into the MME, therefore they served as an upper perfomance boundary estimate.

$$E_i = \frac{\sqrt{(\bar{u}_{i,\mathrm{DNS}} - \bar{u}_{i,\theta})^2}}{U_{\mathrm{bulk}}},\ i = x,\ y,\ z;\ \theta = t,\ R \tag{4.4a}$$

$$\overline{E_i} = \frac{1}{A}\int_A E_i dA,\ i = x,\ y,\ z \tag{4.4b}$$

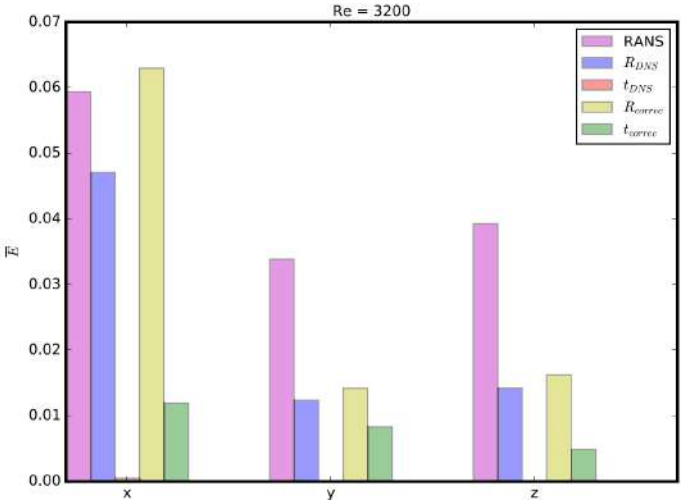


Figure 4.8: Global errors for RANS, $\mathbf{R}_{DNS}$, $\mathbf{t}_{DNS}$, predicted $\mathbf{R}$ and predicted $\mathbf{t}$
Taken from CRUZ *et al.* (2019)

Chapter 5

The Square-Duct Flow

The square-duct flow is of main interest due to the recirculation present in the near wall region. This phenomena occurs mainly because of the imbalance between the normal stresses in the cross sectional plane (XIAO *et al.* (2016), WANG *et al.* (2017b), WU *et al.* (2018)). Figure 5.1 depicts the flow and its secondary motion, in it the x direction is the direction of the main flow. The normal stresses whose imbalance drives the secondary motion are $\mathbf{R}_{yy}$ and $\mathbf{R}_{zz}$.

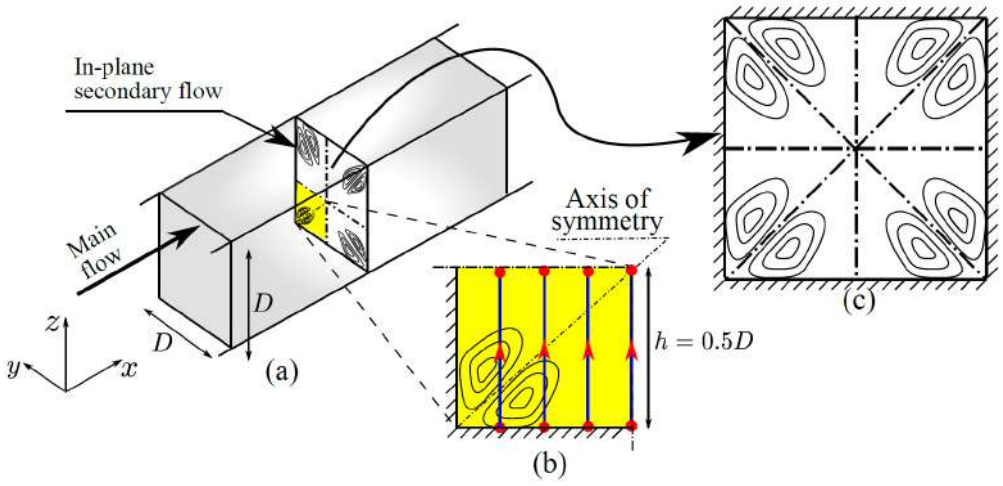


Figure 5.1: Square-duct flow geometry and secondary flow
Adapted from WANG *et al.* (2017b)

Linear-eddy viscosity models fail to capture this imbalance, making this a concern in the development of data-driven efforts of correcting mean flows and correlated quantities of interest.

5.1 Baseline RANS Simulation of the Square-Duct Flow

The baseline simulations of a square-duct were performed to serve both as inputs and to be corrected by neural network predictions. For this purpose, the selected RANS model was the Standard κ - ϵ with coefficients from LAUNDER e SPALDING (1974), shown in table 2.1. In order to reduce computational costs, only one quadrant of the duct was simulated, taking advantage of the flow's symmetry with respect to the center lines in the cross section.

For comparison between methodologies, the setup was based on CRUZ *et al.* (2019). The sole difference is that in CRUZ *et al.* (2019) ten sections of the square-duct were simulated, each section with 1600 cells, resulting in a total of 16000. In this work, the ten sections centroids were clustered in a single section with 125×125 cell centers, resulting in 15625 cells, covering a computational domain of $1\text{m} \times 1\text{m} \times 1\text{m}$. This mesh configuration ensures that the number of cells used to train and test the neural networks are close to the utilized in CRUZ *et al.* (2019), and also guarantees that y^+ is smaller than 0.6 on the duct's walls. The employed mesh is depicted in figure 5.2, it gets coarser as it gets farther away from the duct's wall. Elements length growth ratio in the wall-normal directions is of 2. That means that following on the same wall, the last cell's length will be the double of the first cell located in the duct's corner. This expansion ratio is imposed on both wall-normal directions, y and z in this context.

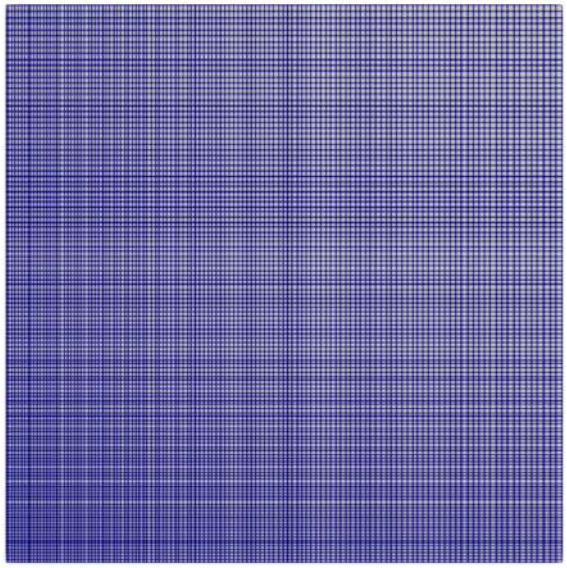


Figure 5.2: Baseline simulations' mesh

Simulations were conducted on the open source software OpenFOAM-4.x (OF).

OF is an extended framework on *C++* programming language, consisting of libraries, solvers and executables for pre and post-processing in fluid mechanics applications. It also enables user developed environments and solvers for personalized simulations.

The *simpleFoam* application was used, a finite volume solver for incompressible, steady-state flows, using the SIMPLE method of coupling the pressure field and the continuity equation (PATANKAR e SPALDING, 1972). The boundary fields of the simulated domain are shown in figure 5.3. The boundary conditions on the fixed walls were: (a) no-slip condition, (b) zero pressure gradient, (c) null turbulent kinetic energy κ, (d) *epsilonWallFunction* for turbulent dissipation ϵ, implemented on OF and derived from the canonical law of the wall. Symmetry boundary conditions are imposed on the symmetry planes and periodic boundary conditions are imposed on the inlet and outlet.

Initial conditions were: (a) uniform velocity field $\bar{\mathbf{u}} = (4.81958 \times 10^{-1}, 0, 0)$ [m/s], (b) null pressure field, (c) uniform $\kappa = 1.09456 \times 10^{-3}$ [m^2/s^2], (d) uniform $\epsilon = 4.28838 \times 10^{-5}$ [m^2/s^3].

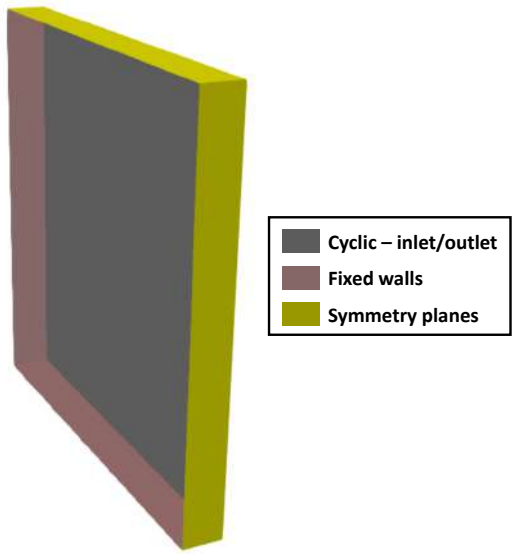


Figure 5.3: Boundary fields of the square-duct

An area averaged value for the velocity components is imposed, this bulk velocity is of $U_{\text{bulk}} = 4.81958 \times 10^{-1}$ [m/s] in the main flow direction x and null on the secondary directions y and z. This is done in OF by implementing a *momentumSource* condition on the internal field. On each iteration, the main velocity component u is averaged over the cross sectional area, if the averaged value is different from the imposed U_{bulk}, an artificial pressure gradient in the main flow's direction $\frac{\partial p}{\partial x}$ is calculated and inserted it in the momentum equations to assure that the averaged value is the bulk velocity.

Six different Re, with respect to U_{bulk} and half the duct's wall length $h = 1$, were simulated by altering ν. Used values of ν are given in table 5.1.

Table 5.1: Simulated Re and corresponding kinematic viscosities

Re	$\nu[\text{m}^2/\text{s}]$
2200	2.1858×10^{-4}
2400	2.0082×10^{-4}
2600	1.8537×10^{-4}
2900	1.6619×10^{-4}
3200	1.5061×10^{-4}
3500	1.3770×10^{-4}

The discretization schemes of the simulations were:

1. Advective terms: Gaussian integration with upwind differencing
2. Diffusive terms: Gaussian integration with corrected linear interpolation
3. Gradient terms: Gaussian integration with linear interpolation
4. Interpolation scheme: Linear interpolation

The Gaussian integration indicates that the values at cell centers are interpolated to and summed at cell faces. Linear differencing means that the values at the cell faces will be a linear interpolation of the two cells separated by the current face. Upwind differencing indicates that the cell face values assumes the value of the following cell in the positive direction.

Once discretized, the linearized pressure equation was solved using the Preconditioned Conjugate Gradient (PCG) solver, with Diagonal incomplete-Cholesky (DIC) preconditioner, for symmetric matrices. The linearized systems of equations for $\bar{\mathbf{u}}$, κ and ϵ were solved using the smooth solver coupled with the symmetric Gauss-Seidel smoother.

Residuals tolerance was of 1×10^{-7} and under-relaxation factors for all solved fields was of 0.3. Relative tolerance between consecutive iterations was of 1×10^{-2}, meaning that the linear systems solvers stopped iterating when solving a same pseudo time-step after their final residual had decreased by two orders of magnitude. Imposing a relative tolerance is considerably effective in reducing computational time, especially when many simulations need to be carried. This is not mandatory in OpenFOAM solvers, where one can simply impose that the residuals tolerance used as the simulation stopping criterion is also employed on pseudo time-steps solutions.

Chapter 6

Methodology

In this chapter the data-driven turbulence models based on the Reynolds Stress Tensor Transport Equation (RSTE) and the Reynolds Force Vector Transport Equation (RFVTE) are introduced and explained. Both equations were used as the means of injecting NN predicted data into the RANS environment for correcting its mean flow field. The two coupled set of differential equations for $\bar{\mathbf{u}}$, p and $\mathbf{R}$ or $\mathbf{r}$ constitute new turbulence models supplied by data provided by ML schemes.

First, the turbulence model that solves the Reynolds Stress tensor $\mathbf{R}$ and it's numerical implementation in OpenFOAM are explained. Second, the deduction of the RFVTE is carried in detail and is subsequently adjusted to the case where the modified Reynolds Force Vector $\mathbf{t}$ is to be used instead of $\mathbf{r}$. This modification is done in order to enable that all involved quantities are calculated by the sole manipulation of the DNS velocity field $\mathbf{u}_{\mathrm{DNS}}$. Afterwards, the numerical solver of the RVTE turbulence model implemented in OpenFOAM is explained.

The CFD segment of the present work is followed by the Machine Learning segment, where the details of the Neural Networks assembling and training stages are explained. Four alternative methodologies to be compared in chapter 7 are detailed. These are: the prediction of the RSTE source term $\hat{\mathbf{\Gamma}}$, the modified RFVTE boundary conditions and its source term $\hat{\boldsymbol{\gamma}}$, the prediction of the modified RFV $\mathbf{t}$ and the Reynolds stress tensor $\mathbf{R}$. These four constitute alternative ways of correcting RANS flows through ML predicted quantities, and their perfomances are evaluated and compared.

6.1 The Adapted Reynolds Stress Tensor Transport Equation

The $\mathbf{R}$ transport equation can be deducted by multiplying the Navier-Stokes equations by the fluctuating velocities, decomposing the velocities into an time av-

erage and a fluctuation and further time averaging the products. The first step of this procedure is shown in equation 6.1b, where $\mathcal{N}(u_i)$ is the Navier-Stokes operator. Further mathematical steps of the deduction of equation 6.2 will be skipped to maintain conciseness.

$$\mathcal{N}(u_i) = \frac{\partial u_i}{\partial t} + u_k \frac{\partial u_i}{\partial x_k} + \frac{1}{\rho}\frac{\partial p}{\partial x_i} - \nu \frac{\partial^2 u_i}{\partial x_k \partial x_k} = 0 \tag{6.1a}$$

$$\overline{u_i' \mathcal{N}(\bar{u}_j + u_j') + u_j' \mathcal{N}(\bar{u}_i + u_i')} = 0 \tag{6.1b}$$

Equation 6.2 presents the RSTE after all calculations in equation 6.1b are carried (WILCOX, 2006).

$$\mathbf{u} \cdot \nabla \mathbf{R} = -\nabla^T \mathbf{u} \cdot \mathbf{R} - \mathbf{R} \cdot \nabla \mathbf{u} + \nu \nabla^2 \mathbf{R} + \mathbf{\Gamma} \tag{6.2}$$

Because the purpose of the present methodology is to deduce an equation containing a term to be used as the target of a NN, all the terms that require modeling *e.g.* the triple velocity fluctuation products, pressure fluctuations and pressure-strain correlation, were grouped into a single tensor $\mathbf{\Gamma}$, as done in THOMPSON *et al.* (2019). In equation 6.2 and subsequent equations the overbar in the mean velocities is dropped for simplicity.

To preserve numerical stability of the proposed method, two adjustments were made to equation 6.2. The terms $-\nabla^T \mathbf{u} \cdot \mathbf{R} - \mathbf{R} \cdot \nabla \mathbf{u}$, canonically referred to as the production term of the RSTE, were incorporated to the source term of the equation. Also, a turbulent vicosity ν_t was introduced into the diffusive term, resulting in equation 6.3a. The source term of the proposed RSTE model becomes the symmetric tensor $\hat{\mathbf{\Gamma}}$, described in equation 6.3b.

$$\mathbf{u} \cdot \nabla \mathbf{R} = \nabla \cdot ((\nu + \nu_t) \nabla \mathbf{R}) + \hat{\mathbf{\Gamma}} \tag{6.3a}$$

$$\hat{\mathbf{\Gamma}} = \mathbf{\Gamma} - \nabla^T \mathbf{u} \cdot \mathbf{R} - \mathbf{R} \cdot \nabla \mathbf{u} - \nabla \cdot (\nu_t \nabla \mathbf{R}) \tag{6.3b}$$

Modifying equation 6.2 into equation 6.3a was necessary to guarantee not only numeric stability but also the robustness of the proposed methodology, ensuring that most NN predictions would be able to correct the RANS flow field.

6.2 The Reynolds Force Vector Transport Equation

Because the Reynolds Force Vector is defined as $\nabla \cdot \mathbf{R} \equiv \mathbf{r}$, the divergence of all terms in the RSTE are taken. For that, equation 6.2 is presented in indicial notation in equation 6.4.

$$u_k \frac{\partial R_{\mathrm{ij}}}{\partial x_k} = -R_{\mathrm{jk}} \frac{\partial u_i}{\partial x_k} - R_{\mathrm{ik}} \frac{\partial u_j}{\partial x_k} + \nu \frac{\partial^2 R_{\mathrm{ij}}}{\partial x_k \partial x_k} + \Gamma_{\mathrm{ij}} \tag{6.4}$$

First, the divergence of the convective term is taken in equaton 6.5.

$$\begin{aligned} \frac{\partial}{\partial x_m}\left(u_k \frac{\partial R_{\mathrm{ij}}}{\partial x_k}\right) e_m \cdot e_i e_j &= \left[\frac{\partial u_k}{\partial x_m}\frac{\partial R_{\mathrm{ij}}}{\partial x_k}\delta_{\mathrm{im}} + u_k \frac{\partial}{\partial x_k}\left(\frac{\partial R_{\mathrm{ij}}}{\partial x_m}\right)\delta_{\mathrm{im}}\right] e_j \\ &= \left[\frac{\partial u_k}{\partial x_i}\frac{\partial R_{\mathrm{ij}}}{\partial x_k} + u_k \frac{\partial}{\partial x_k}\left(\frac{\partial R_{\mathrm{ji}}}{\partial x_i}\right)\right] e_j \end{aligned} \tag{6.5}$$

Then, the first production term, in equation 6.6.

$$\begin{aligned} \frac{\partial}{\partial x_m}\left(-R_{\mathrm{jk}} \frac{\partial u_i}{\partial x_k}\right) e_m \cdot e_i e_j &= -\left[\frac{\partial R_{\mathrm{jk}}}{\partial x_m}\frac{\partial u_i}{\partial x_k}\delta_{\mathrm{im}} + R_{\mathrm{jk}} \frac{\partial}{\partial x_m}\left(\frac{\partial u_i}{\partial x_k}\right)\delta_{\mathrm{im}}\right] e_j \\ &= -\left[\frac{\partial R_{\mathrm{jk}}}{\partial x_i}\frac{\partial u_i}{\partial x_k} + \frac{\partial}{\partial x_k}\left(\frac{\partial u_i}{\partial x_i}\right) R_{\mathrm{kj}}\right] e_j \end{aligned} \tag{6.6}$$

Following, the second production term's divergence is calculated in equation 6.7.

$$\begin{aligned} \frac{\partial}{\partial x_m}\left(-R_{\mathrm{ik}} \frac{\partial u_j}{\partial x_k}\right) e_m \cdot e_i e_j &= -\left[\frac{\partial R_{\mathrm{ik}}}{\partial x_m}\frac{\partial u_j}{\partial x_k}\delta_{\mathrm{im}} + R_{\mathrm{ik}} \frac{\partial}{\partial x_m}\left(\frac{\partial u_j}{\partial x_k}\right)\delta_{\mathrm{im}}\right] e_j \\ &= -\left[\frac{\partial R_{\mathrm{ik}}}{\partial x_i}\frac{\partial u_j}{\partial x_k} + R_{\mathrm{ki}} \frac{\partial}{\partial x_i}\left(\frac{\partial u_j}{\partial x_k}\right)\right] e_j \end{aligned} \tag{6.7}$$

Fourth, the divergence of the diffusive term, equation 6.8.

$$\begin{aligned} \frac{\partial}{\partial x_m}\left(\nu \frac{\partial^2 R_{\mathrm{ij}}}{\partial x_k \partial x_k}\right) e_m \cdot e_i e_j &= \nu \frac{\partial}{\partial x_m}\left(\frac{\partial^2 R_{\mathrm{ij}}}{\partial x_k \partial x_k}\right)\delta_{\mathrm{im}} e_j \\ &= \nu \frac{\partial^2}{\partial x_k \partial x_k}\left(\frac{\partial R_{\mathrm{ji}}}{\partial x_i}\right) e_j \end{aligned} \tag{6.8}$$

Lastly, the divergence of the tensor $\mathbf{\Gamma}$ in equation 6.9.

$$\frac{\partial \Gamma_{\mathrm{ij}}}{\partial x_m} e_m \cdot e_i e_j = \frac{\partial \Gamma_{\mathrm{ij}}}{\partial x_i} e_j \tag{6.9}$$

Joining all the terms from equations 6.5 until 6.9 leads to Reynolds Force Vector Transport Equation 6.10, in indicial notation.

$$\frac{\partial u_k}{\partial x_i}\frac{\partial R_{\mathrm{ij}}}{\partial x_k} + u_k \frac{\partial}{\partial x_k}\left(\frac{\partial R_{\mathrm{ij}}}{\partial x_i}\right) = -\frac{\partial R_{\mathrm{jk}}}{\partial x_i}\frac{\partial u_i}{\partial x_k} - \frac{\partial R_{\mathrm{ik}}}{\partial x_i}\frac{\partial u_j}{\partial x_k} - R_{\mathrm{ki}}\frac{\partial}{\partial x_i}\left(\frac{\partial u_j}{\partial x_k}\right) + \nu \frac{\partial^2}{\partial x_k \partial x_k}\left(\frac{\partial R_{\mathrm{ij}}}{\partial x_i}\right) + \frac{\partial \Gamma_{\mathrm{ij}}}{\partial x_i} \tag{6.10}$$

Equation 6.10 translates back into the symbolic equation 6.11. Further math-

ematical details of translating equation 6.10 back into symbolic notation can be checked in appendix A.

$$\mathbf{u} \cdot \nabla(\nabla \cdot \mathbf{R}) = -\nabla^T \mathbf{u} \cdot (\nabla \cdot \mathbf{R}) + \nu \nabla^2 (\nabla \cdot \mathbf{R}) + \boldsymbol{\gamma} \tag{6.11}$$

Where the vector $\boldsymbol{\gamma}$ consists of all the terms that require modeling and that are not a direct function of $\nabla \cdot \mathbf{R}$. Equation 6.12 shows all terms within $\boldsymbol{\gamma}$, in it : is the double product between a third and a second order tensors.

$$\boldsymbol{\gamma} = -2\nabla^T \mathbf{u} : \nabla \mathbf{R} - \mathbf{R} : \nabla(\nabla \mathbf{u}) + \nabla \cdot \boldsymbol{\Gamma} \tag{6.12}$$

Substituting the definition of the Reynolds force vector $\mathbf{r} \equiv \nabla \cdot \mathbf{R}$ into equation 6.11 results in equation 6.13.

$$\mathbf{u} \cdot \nabla \mathbf{r} = -\nabla^T \mathbf{u} \cdot \mathbf{r} + \nu \nabla^2 \mathbf{r} + \boldsymbol{\gamma} \tag{6.13}$$

6.2.1 The Modified RFVTE

Owing to the defficiencies in the DNS provided $\mathbf{R}$ exposed in THOMPSON *et al.* (2016), and their limitation in providing reasonable predictions in the ML context evidentiaded by CRUZ *et al.* (2019), the present methodology also calculates $\mathbf{r}$ indirectly, through the manipulation of the mean momentum equations, as shown in equation 6.14.

$$\mathbf{r} = \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p - \nu \nabla^2 \mathbf{u} \tag{6.14}$$

The present work also uses the DNS square-duct data provided by PINELLI *et al.* (2010), as in CRUZ *et al.* (2019). Since no mean pressure fields are provided in this dataset, an modified RFV $\mathbf{t} = \mathbf{r} - \frac{1}{\rho} \nabla p$ was used, therefore, equation 6.14 becomes equation 6.15.

$$\mathbf{t} = \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} \tag{6.15}$$

Because the modified RFV $\mathbf{t}$ is used, a modification of the RFVTE is necessary. Substituting the definition $\mathbf{r} = \mathbf{t} + \frac{1}{\rho} \nabla p$ into equation 6.13 results:

$$\mathbf{u} \cdot \nabla(\mathbf{t} + \frac{1}{\rho} \nabla p) = -\nabla^T \mathbf{u} \cdot (\mathbf{t} + \frac{1}{\rho} \nabla p) + \nu \nabla^2 (\mathbf{t} + \frac{1}{\rho} \nabla p) + \boldsymbol{\gamma} \tag{6.16a}$$

$$\mathbf{u} \cdot \nabla \mathbf{t} + \mathbf{u} \cdot \nabla(\frac{1}{\rho} \nabla p) = -\nabla^T \mathbf{u} \cdot \mathbf{t} - \nabla^T \mathbf{u} \cdot (\frac{1}{\rho} \nabla p) + \nu \nabla^2 \mathbf{t} + \nu \nabla^2 (\frac{1}{\rho} \nabla p) + \boldsymbol{\gamma} \tag{6.16b}$$

$$\mathbf{u}\cdot\nabla\mathbf{t} = -\nabla^T\mathbf{u}\cdot\mathbf{t} + \nu\nabla^2\mathbf{t} + \underbrace{\boldsymbol{\gamma} - \mathbf{u}\cdot\nabla(\frac{1}{\rho}\nabla p) - \nabla^T\mathbf{u}\cdot(\frac{1}{\rho}\nabla p) + \nu\nabla^2(\frac{1}{\rho}\nabla p)}_{\tilde{\gamma}} \tag{6.16c}$$

$$\mathbf{u}\cdot\nabla\mathbf{t} = -\nabla^T\mathbf{u}\cdot\mathbf{t} + \nu\nabla^2\mathbf{t} + \tilde{\boldsymbol{\gamma}} \tag{6.16d}$$

Therefore, the RFVTE is adapted to the modified Reynolds Force Vector and the terms that require modeling and are not a direct function of $\mathbf{t}$ are grouped into the source term $\boldsymbol{\gamma}_p$, as shown in equation 6.17

$$\boldsymbol{\gamma}_p = \boldsymbol{\gamma} - \mathbf{u}\cdot\nabla(\frac{1}{\rho}\nabla p) - \nabla^T\mathbf{u}\cdot(\frac{1}{\rho}\nabla p) + \nu\nabla^2(\frac{1}{\rho}\nabla p) \tag{6.17}$$

As done in section 6.1 in the RSTE, the proposed transport equation 6.16d requires further adaptions to preserve the equation's numerical stability. Analogously as in the RSTE, the velocity gradient term in 6.16d is incorporated into the source term, along with a turbulent viscosity into the diffusive term. Resulting in the final data-driven transport equation for the modified Reynolds Force Vector 6.18a. The source term of 6.18a is presented in equation 6.18b.

$$\mathbf{u}\cdot\nabla\mathbf{t} = \nabla\cdot((\nu+\nu_t)\nabla\mathbf{t}) + \hat{\boldsymbol{\gamma}} \tag{6.18a}$$

$$\hat{\boldsymbol{\gamma}} = \boldsymbol{\gamma}_p - \nabla^T\mathbf{u}\cdot\mathbf{t} - \nabla\cdot(\nu_t\nabla\mathbf{t}) \tag{6.18b}$$

6.3 The Data-Driven Turbulence Models

Both presented source terms $\hat{\boldsymbol{\Gamma}}$ and $\hat{\boldsymbol{\gamma}}$ were employed as NN targets, responsible for correcting the RANS flow fields. For this purpose, the DNS $\hat{\boldsymbol{\Gamma}}_{\text{DNS}}$ and $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$ were calculated indirectly, through the manipulation of equations 6.3a and 6.18a, respectively, resulting in the formulas 6.19a and 6.19b. On both equations 6.19a and 6.19b the employed turbulent viscosity ν_t were taken from the baseline RANS simulations.

$$\hat{\boldsymbol{\Gamma}}_{\text{DNS}} = \mathbf{u}_{\text{DNS}}\cdot\nabla\mathbf{R}_{\text{DNS}} - \nabla\cdot((\nu+\nu_t)\nabla\mathbf{R}_{\text{DNS}}) \tag{6.19a}$$

$$\hat{\boldsymbol{\gamma}}_{\text{DNS}} = \mathbf{u}_{\text{DNS}}\cdot\nabla\mathbf{t}_{\text{DNS}} - \nabla\cdot((\nu+\nu_t)\nabla\mathbf{t}_{\text{DNS}}) \tag{6.19b}$$

To calculate $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$ via equation 6.19b a pre-processing in $\mathbf{t}_{\text{DNS}}$ was required. The calculation of $\mathbf{t}_{\text{DNS}}$ through the manipulation of the momentum balance results in

a field with improper values on the symmetry plane on the center line of the duct. This probably occurs due to the low resolution of the available mesh from the DNS simulations of the square-duct by PINELLI *et al.* (2010).

Although relatively small, these regions grow in size and magnitude when inputed into equation 6.19b and harm the $\hat{\boldsymbol{\gamma}}_{\mathrm{DNS}}$ fields, their usage in the RFVTE and consequently the training of the NN responsible for predicting $\hat{\boldsymbol{\gamma}}$.

The pre-processing consists on simply erasing the inaccurate regions and replacing them by an extrapolation of the remaining field. The extrapolation was done by using a $2D$ cubic spline provided by the *RectBivariateSpline* command of the *SciPy* Python library (VIRTANEN *et al.*, 2020). *SciPy* is a scientific package containing numerous mathematical algorithms and functions. Figure 6.1 demonstrates the x component of $\mathbf{t}_{\mathrm{DNS}}$ with the inaccurate region highlighted.

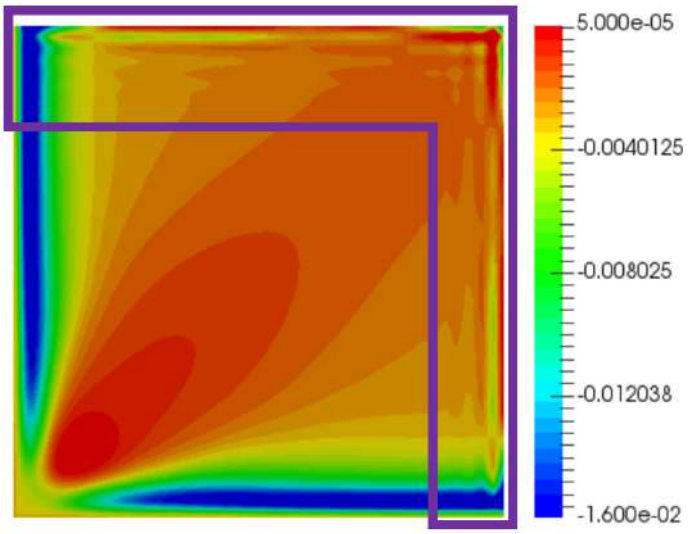


Figure 6.1: Inaccurate $\mathbf{t}_{\mathrm{DNS}}$ regions

Figure 6.2 demonstrates the x component of $\mathbf{t}_{\mathrm{DNS}}$ after the pre-processing step is performed.

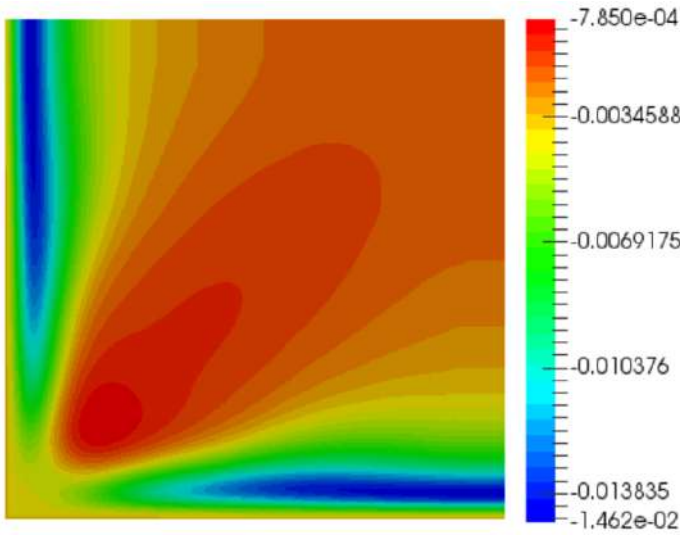


Figure 6.2: $\mathbf{t}_{\mathrm{DNS}}$ after pre-processing

Differently from the vector $\hat{\boldsymbol{\gamma}}$, the indirect calculation of the source term $\hat{\boldsymbol{\Gamma}}$ of the RSTE methodology did not require any pre-processing.

Two data-driven turbulence models are proposed using the presented Reynolds Stress Tensor Transport Equation 6.3a and the modified Reynolds Force Vector Transport Equation 6.18a. Using the tensor $\hat{\boldsymbol{\Gamma}}$ and the vector $\hat{\boldsymbol{\gamma}}$ as a source terms, both the RSTE and the RFVTE were used to correct the mean flow provided by the κ - ϵ simulations described in chapter 5. This was done by coupling the novel transport equations with the mean momentum equations and the pressure equation from the SIMPLE method by PATANKAR e SPALDING (1972). The SIMPLE method uses the continuity equation to build an equation for the pressure field, while still imposing the conservation of mass to the flow field.

With these schemes, two sets of coupled partial differential equations were formed. The first one, using the RSTE, consisted on ten coupled PDEs, solving the three components of the velocity vector $\mathbf{u} = [u, v, w]$, the six components of the symmetric tensor $\mathbf{R} = [R_{xx}, R_{xy}, R_{xz}, R_{yy}, R_{yz}, R_{zz}]$ and the pressure p. The second data-driven turbulence model solves the three velocity components, the three components of the modified RFV $\mathbf{t} = [t_x, t_y, t_z s]$ and the pressure field.

Both algorithms were implemented in OpenFOAM-4.x. Implementation was done in two separate turbulence models, this type of implementation enables the use of the software's preexisting solvers. Both turbulence models are available on the GitHub platform at: `https://github.com/mthsmcd/DataDrivenTurbulenceModels/`.

The chosen OpenFOAM's solver was *simpleFoam*, the same used in the κ - ϵ simulations detailed in chapter 5. Once the source terms $\hat{\boldsymbol{\Gamma}}$ or $\hat{\boldsymbol{\gamma}}$ are provided the

numerical solution procedure works as follows,

1. Update $\mathbf{u}$ through mean momentum equations $\mathbf{u} \cdot \nabla \mathbf{u} = \nu \nabla^2 \mathbf{u} - \nabla \cdot \mathbf{R}$ or $\mathbf{u} \cdot \nabla \mathbf{u} = \nu \nabla^2 \mathbf{u} + \mathbf{t}$

2. Impose continuity to new mean velocity field and update p

3. Update field $\mathbf{R}$ or $\mathbf{t}$ through their respective transport equations $\mathbf{u} \cdot \nabla \mathbf{R} = \nabla \cdot ((\nu + \nu_t) \nabla \mathbf{R}) + \hat{\mathbf{\Gamma}}$ or $\mathbf{u} \cdot \nabla \mathbf{t} = \nabla \cdot ((\nu + \nu_t) \nabla \mathbf{t}) + \hat{\boldsymbol{\gamma}}$ using the updated $\mathbf{u}$

The boundary conditions (BC) for $\mathbf{u}$ and p are the same as the ones used in the κ - ϵ simulations, detailed in chapter 5. The tensor $\mathbf{R}$ is null at the wall. For the vector $\mathbf{t}$, unlike the first and second-type boundary conditions of $\mathbf{u}$ and p respectively, the BC are non-canonical. This difficulty imposes the necessity of also using ML procedures to predict the boundary conditions for the RFV $\mathbf{t}_{\text{wall}}$, which is also done in the present work by the use of NN.

To correct the RANS flow, the presented numerical procedure used $\mathbf{u}_{\text{RANS}}$ and p_{RANS} as initial conditions (IC), and the NN predicted $\hat{\mathbf{\Gamma}}_{\text{NN}}$ or $\hat{\boldsymbol{\gamma}}_{\text{NN}}$ and $\mathbf{t}_{\text{wall,NN}}$ values. The initial conditions of $\mathbf{R}$ and $\mathbf{t}$ however are not their corresponding RANS fields. Although possible to use the κ - ϵ fields $\mathbf{R}$ or $\mathbf{t}$ as initial conditions, the procedure achieved better results when departing from null IC on both cases. Numerical convergence also occurred reasonably faster.

6.4 Neural Networks Setup

In order to fuel the presented turbulence model, three different sets of Neural Networks were built to predict the source terms $\hat{\mathbf{\Gamma}}_{\text{NN}}$ and $\hat{\boldsymbol{\gamma}}_{\text{NN}}$, and the boundary field $\mathbf{t}_{\text{wall,NN}}$. The DNS calculated fields $\hat{\mathbf{\Gamma}}_{\text{DNS}}$, $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$ and $\mathbf{t}_{\text{wall,DNS}}$ were used to train the networks and to evaluate it. Other two sets of NN were built to predict respectively $\mathbf{t}$ and $\mathbf{R}$, which are trained by using the same setup of the $\hat{\mathbf{\Gamma}}$ and $\hat{\boldsymbol{\gamma}}$ networks. This was done to compare the four methodologies' capabilities.

The DNS database utilized was provided by PINELLI *et al.* (2010) and post-processed by RANGEL (2019). The post-processing consisted on enhancing the time averages of the velocity and the Reynolds stress components by further averaging the eight square-duct's octants provided in the DNS dataset, taking advantage of the symmetry between these regions. The symmetry between the eight octants is illustrated in figure 5.1. This average increment enhances the dataset, especially the problematic $\mathbf{R}$, whose numerical convergence is not as well observed in the original dataset, reducing the disparities when using $\mathbf{R}$ as ML targets that will correct the RANS flow field.

All NN were built using the *Keras* Python library for Machine Learning (CHOLLET *et al.*, 2015). The *Keras* library is widely used and provides an extensive framework for ML purposes, with its flexibility and wide range of available pre-programmed fully customizable models and configurations, *Keras* makes the development of artificial intelligence codes much simpler.

6.4.1 Prediction of $\hat{\mathbf{\Gamma}}$ and $\hat{\boldsymbol{\gamma}}$

Neural Network Architecture and Hyper-Parameters Configuration

The neural networks responsible for predicting the source terms $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$ and $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$ were built with two hidden layers each and 100 neurons per layer. The number of neurons in the output layer varied with the network's target. The $\hat{\mathbf{\Gamma}}$ predicting networks' output layer contained six neurons, while the $\hat{\boldsymbol{\gamma}}$ output layers contained three. Each neuron in the output layer is assigned to each of the target's components. Multiple architectures were evaluated, ranging from single layer networks to deep-learning networks with more than 10 layers and various neurons per layer configurations. The best fit was found to be the two layer, 100 neurons networks.

In the output layer the activation fucntions $\varphi(v)$ used were linear functions. In the hidden layers the $\varphi(v)$ functions were the hyperbolic tangent function, equation 3.8b, whose behavior is depicted in figure 3.3 (b). Other activation functions like the sigmoid and ReLU were tested, but results from $\tanh(v)$ were considerably better.

The weights and biases of the network were updated after each iteration accordingly to an adaptative learning rate α. At the beginning of training α was of 1×10^{-3}, it was reduced by a factor of 0.6 whenever 5 consecutive epochs resulted in no decrease of the loss function on the validation dataset. Training was interrupted when 20 consecutive epochs resulted in no improvement of the validation loss. The training batch-size was of 32 data points, which guaranteed a reasonable training speed whithout harming the resulting networks predicting abilities. The used loss-function was the mean-squared error function, and the learning optimizer was the *Adam* algorithm (KINGMA e BA, 2014).

Input Selection

The selection of the network's inputs departed from the ones used in CRUZ *et al.* (2019). At first, the inputs for the $\hat{\mathbf{\Gamma}}$ networks consisted on the components of the eight symmetric tensors and associated divergents used in CRUZ *et al.* (2019), plus the symmetric tensor $\hat{\mathbf{\Gamma}}$ and its divergent, resulting in 81 features. As for the $\hat{\boldsymbol{\gamma}}$ networks, the same tensorial basis was used, with the addition of the non-symmetric tensor $\nabla\hat{\boldsymbol{\gamma}}$ and the vector $\hat{\boldsymbol{\gamma}}$, resulting in a total of 84 features. These *a priori* inputs are listed in table 6.1.

Table 6.1: A priori inputs

Tensors	**Vectors**
$\mathbf{S}$	$\nabla \cdot \mathbf{S}$
$\mathbf{P}$	$\nabla \cdot \mathbf{P}$
$\mathbf{S}^2$	$\nabla \cdot \mathbf{S}^2$
$\mathbf{P}^2$	$\nabla \cdot \mathbf{P}^2$
$\mathbf{S} \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{S}$	$\nabla \cdot (\mathbf{S} \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{S})$
$\mathbf{S}^2 \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{S}^2$	$\nabla \cdot (\mathbf{S}^2 \cdot \mathbf{P} + \mathbf{P} \cdot \mathbf{S}^2)$
$\mathbf{P}^2 \cdot \mathbf{S} + \mathbf{S} \cdot \mathbf{P}^2$	$\nabla \cdot (\mathbf{P}^2 \cdot \mathbf{S} + \mathbf{S} \cdot \mathbf{P}^2)$
$\mathbf{R}$	$\mathbf{r}$
$\hat{\mathbf{\Gamma}}$ or $\nabla\hat{\gamma}$	$\nabla \cdot \hat{\mathbf{\Gamma}}$ or $\hat{\gamma}$

An analysis on each of the tensors and vectors components magnitudes' revealed that some of them are orders of magnitude higher than others in some regions of the flow geometry. This discrepancy was found to greatly impair the networks' predicting capabilities. Therefore, all components of tensors or vectors whose average magnitudes were more than 2 orders of magnitude higher than the others were excluded.

Some other inputs were composed mostly by null components in the whole domain *e.g.* tensor $\mathbf{S}$ components S_{xx}, S_{yy}, S_{zz} and S_{yz}. When inputed into the NN these variables served as noise to be taken into account by the successive operations done in the network. Members of this group were also excluded from the inputs list.

Another special case is the tensor $\mathbf{S}\cdot\mathbf{P}+\mathbf{P}\cdot\mathbf{S}$ and its divergent, whose components on the κ - ϵ context were mostly null on the duct's geometry, but on some very scarce regions presented notably high values, constituting a cosiderably discontinuous field. This also served as additional noise when inputed into the networks, and impaired the networks performance, being also excluded.

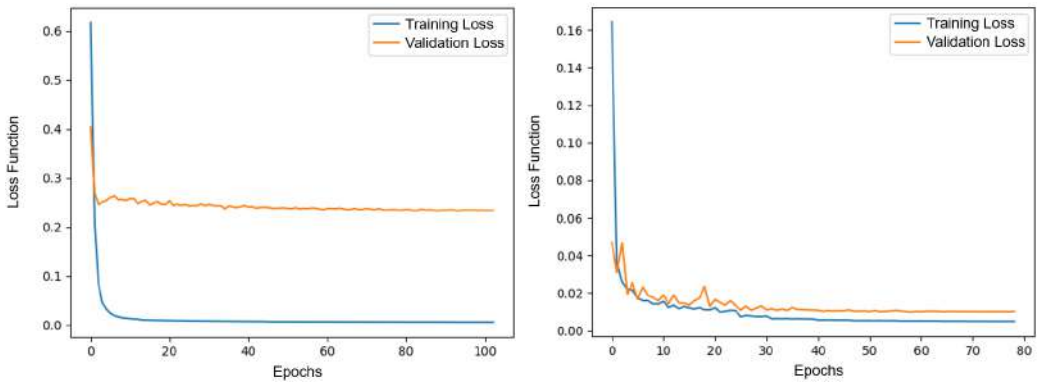
After filtering out the three problematic groups within table 6.1, the $\hat{\mathbf{\Gamma}}$ networks' inputs were reduced from 81 to 27 and the $\hat{\gamma}$ inputs' went from the *a priori* 84 inputs to a total of 22. The remaining features for both cases are presented on table 6.2

Table 6.2: Final list of inputs

Tensors Components	Vectors Components
$\mathbf{S}$: $[S_{xy}, S_{xz}]$	$\nabla \cdot \mathbf{S}$: $(\nabla \cdot \mathbf{S})_x$
$\mathbf{P}$: $[P_{xx}, P_{yy}, P_{yz}, P_{zz}]$	-
$\mathbf{S}^2$: $[S^2_{xx}, S^2_{yy}, S^2_{yz}, S^2_{zz}]$	-
$\mathbf{R}$: $[R_{xx}, R_{xy}, R_{xz}, R_{yy}, R_{zz}]$	$\mathbf{r}$: $[r_x, r_y, r_z]$
$\nabla\hat{\boldsymbol{\gamma}}$: $[\nabla\hat{\boldsymbol{\gamma}}_{yx}, \nabla\hat{\boldsymbol{\gamma}}_{zx}]$	$\hat{\boldsymbol{\gamma}}$: $[\hat{\gamma}_x]$
or	or
$\hat{\boldsymbol{\Gamma}}$: $[\hat{\boldsymbol{\Gamma}}_{xx}, \hat{\boldsymbol{\Gamma}}_{xy}, \hat{\boldsymbol{\Gamma}}_{xz}, \hat{\boldsymbol{\Gamma}}_{yy}, \hat{\boldsymbol{\Gamma}}_{zz}]$	$\nabla \cdot \hat{\boldsymbol{\Gamma}}$: $[(\nabla \cdot \hat{\boldsymbol{\Gamma}})_x, (\nabla \cdot \hat{\boldsymbol{\Gamma}})_y, (\nabla \cdot \hat{\boldsymbol{\Gamma}})_z]$

The inputs reduction is beneficial not only because the impairing and noisy components are excluded, but also because the considerable number of *a priori* inputs increased the number of parameters to be trained in the networks, since the first layer would have more weights to be adjusted. Lesser parameters to be adjusted when using limited amounts of training data, like the present work's case, is advantageous to the network.

Figure 6.3 demonstrates the difference between the evolution of training and validation losses when using the *a priori* 84 inputs of the $\hat{\boldsymbol{\gamma}}$ and the *a posteriori* 22 inputs. The proximity between the training and validation curves indicates that the network is assimilating and generalizing the training data to the validation dataset much better when using the reduced number of inputs. The same behaviour depicted in figure 6.3 occurred in the $\hat{\boldsymbol{\Gamma}}$ networks training.



(a) (b)

Figure 6.3: (a) 84 inputs learning curve, (b) 22 inputs learning curve

Input and Output Normalization

Before being inserted into the network, the inputs are combined in an input matrix X. In X, each column represents one of the components in table 6.2 and each line refers to a centroid in the simulation mesh, that is, matrix X in the $\hat{\boldsymbol{\gamma}}$

networks is a 15625×22 matrix. Every column of the input matrix X must be normalized following the formula in equation 6.20, where μ is the mean and σ is the standard deviation of the column's values. This normalization ensures that the values of each column will be adjusted to a normal distribution with zero mean and unit variance. This is a standard procedure in neural network training and use in general, it must also be done to the testing input matrix X_{test} and validation input matrix X_{val}.

$$X_{\text{column}} = \frac{X_{\text{column}} - \mu_{\text{column}}}{\sigma_{\text{column}}} \tag{6.20}$$

The output matrix Y organization follows the same logic of the matrix X, lines correspond to cells of the simulations and columns represent each of the target's components. In the $\hat{\mathbf{\Gamma}}$ networks, matrix Y was a 15625×6 matrix. The normalization procedure in equation 6.20 must also be done to the output training and validation matrix Y_{train} and Y_{val}. Following the training stage, the outputed Y_{test} will also be normalized, requiring the rescaling through the inverse procedure, shown in equation 6.21

$$Y_{\text{column}} = Y_{\text{column}} \times \sigma_{\text{column}} + \mu_{\text{column}} \tag{6.21}$$

Train, Test and Validation Splitting

A cross-validation step was performed to separate the 6 available Re simulations into train, validation and test datasets. Using four Re to compose the training group, one Re for validation and one for testing creates a scenario of 30 possible combinations of the three groups. The cross-validation ensures that the best fit is chosen based on the performance of the network on the test data of each of the 30 possibilities. To find out the combination that lead to the better results, all 30 possibilites were evaluated, 5 times each, resulting in 150 separately trained networks for each of the targets. The error metric employed to select the network that performed the better on the predicted Y_{test} is shown in equation 6.22, where the term $\text{rms}(\hat{\boldsymbol{\gamma}}_{\text{test}}, \hat{\boldsymbol{\gamma}}_{\text{DNS}})$ is the root mean-squared error, and the term $\max(\text{mean}(|\hat{\boldsymbol{\gamma}}_{\text{DNS}}|))$ is the maximum value among local means of the absolute values of the three components of $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$.

$$S(\hat{\boldsymbol{\gamma}}_{\text{test}}, \hat{\boldsymbol{\gamma}}_{\text{DNS}}) = \frac{\text{rms}(\hat{\boldsymbol{\gamma}}_{\text{test}}, \hat{\boldsymbol{\gamma}}_{\text{DNS}})}{\max(\text{mean}(|\hat{\boldsymbol{\gamma}}_{\text{DNS}}|))} \tag{6.22}$$

For both $\hat{\mathbf{\Gamma}}$ and $\hat{\boldsymbol{\gamma}}$ networks, better results were achieved by using $Re = 2400$ as the validation dataset, $Re = 2900$ as the testing dataset and $Re = [2200, 2600, 3200, 3500]$ as the training dataset. Train, test and validation input

and output matrix sizes are exposed in table 6.3. Since every Re contains 15625 computational cells, the training X and Y matrices have 4 times the number of lines of the validation and test matrices.

Table 6.3: Input and output matrices X and Y sizes

	X		Y	
	Lines	**Columns**	**Lines**	**Columns**
Train	62500	27 or 22	62500	6 or 3
Validation	15625	27 or 22	15625	6 or 3
Test	15625	27 or 22	15625	6 or 3
Total	93750	27 or 22	93750	6 or 3

After cross-validation was performed, 100 neural networks were trained for each of the targets $\hat{\mathbf{\Gamma}}$ and $\hat{\gamma}$ using the configuration pointed by the cross-validation as the best setup. The error metric in equation 6.22 was used again to select the best network in the 100 trained.

Penalization of Loss Function

The $\hat{\gamma}$ field in the square-duct is mostly restrained to the near wall regions. Outside this vicinity the values decrease greatly and are almost null. To help the network correctly assimilate that some of the cell centers it has to predict are more important than others, a weight can be imposed on some data points. This was only done in the training of the $\hat{\gamma}$ networks, the weights in the mesh elemenst within a 8 cell distance to the wall were assigned a weight of 5, while the remaining cells were assigned a weight of 1. Other values of wall distances and weight values were also evaluated.

The weights penalize the loss function on the signalized datapoints. The loss function in equation 3.2 becomes the function 6.23, where ϕ_k is the weight of a given datapoint k.

$$J(\hat{\gamma}_{\mathrm{NN}}, \hat{\gamma}_{\mathrm{DNS}}) = \frac{1}{2}\sum_{k=1}^{n} \phi_k(\hat{\gamma}_{k,\mathrm{NN}} - \hat{\gamma}_{k,\mathrm{DNS}})^2 \tag{6.23}$$

Distribution of loss function's weights can be checked at figure 6.4.

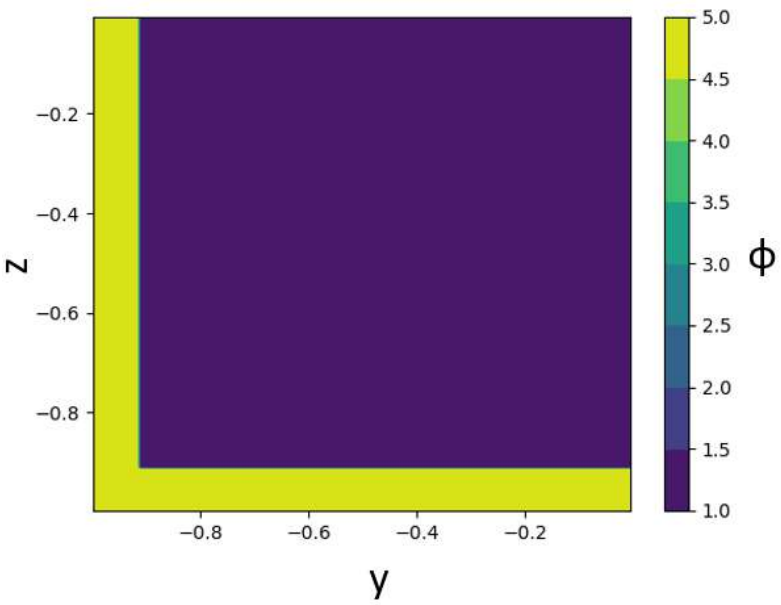


Figure 6.4: Training datapoints weights

Prediction of BC's for t

The network for predicting the boundary conditions of the modified RFVTE consisted of a single hidden layer with 50 neurons. Once again, numerous architectures were evaluated before selecting one. Output layer had again three neurons, and linear activation functions. Activation functions of the hidden layer were also the hyperbolic tangent function and an adaptative learning rate was also used, in this scenario starting at 5×10^{-3} and being reduced by a factor of 0.8 whenever 20 consecutive updates resulted in no decrease on validation loss.

The starting α is considerably high when taking into account the network size, but testing various starting values indicated that higher values, although not capable of minimizing the loss function by themselves, when reduced after some epochs, reached better local minima than if starting at smaller α. This did not occur when training the $\hat{\gamma}$ network, where starting with higher α impaired the networks capabilities.

The training batch-size was of 5 data points and the training segment was interrupted after 300 consecutive epochs did not improve validation loss. Loss function and training optimizer were the same as the ones used in the $\hat{\gamma}$ network. The best fit network was selected using the same error metric of equation 6.22 among 100 trained networks.

Most of the differences in the configuration of the $\hat{\gamma}$ and $\mathbf{t}_{\text{wall}}$ networks was due to the difference in the amount of available data points. In the $\hat{\gamma}$, all mesh cells were used to train and predict the target. In the $\mathbf{t}_{\text{wall}}$ network only wall cells can

be utilized, with a mesh of 125×125 points, this results in only 250 data points for each Reynolds number simulation.

The table 6.4 summarizes the input and output matrices sizes for the $\mathbf{t}_{\text{wall}}$ network.

Table 6.4: Input and output matrices X and Y sizes for the $\mathbf{t}_{\text{wall}}$ network

	X		Y	
	Lines	**Columns**	**Lines**	**Columns**
Train	1000	22	1000	3
Validation	250	22	250	3
Test	250	22	250	3
Total	1500	22	1500	3

Figure 6.5 summarizes the presented training procedure of the $\hat{\boldsymbol{\gamma}}$ neural networks. The procedure in the $\hat{\boldsymbol{\Gamma}}$ networks is analogous. Figure 6.6 illustrates the testing of the $\hat{\boldsymbol{\gamma}}$ NN and the further propagation of the predicted $\hat{\boldsymbol{\gamma}}_{\text{NN}}$ into the computational fluids dynamics solver *simpleFoam* using the developed turbulence model.

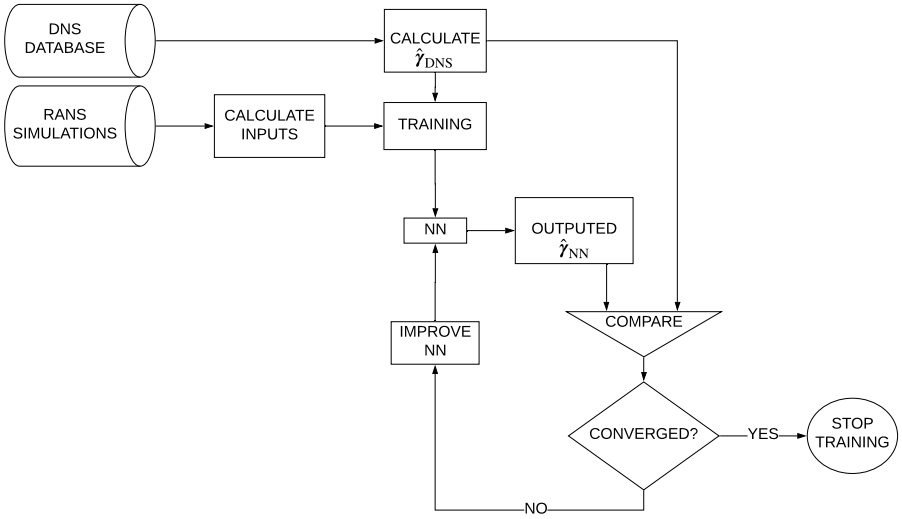


Figure 6.5: $\hat{\boldsymbol{\gamma}}$ neural networks training flow-chart

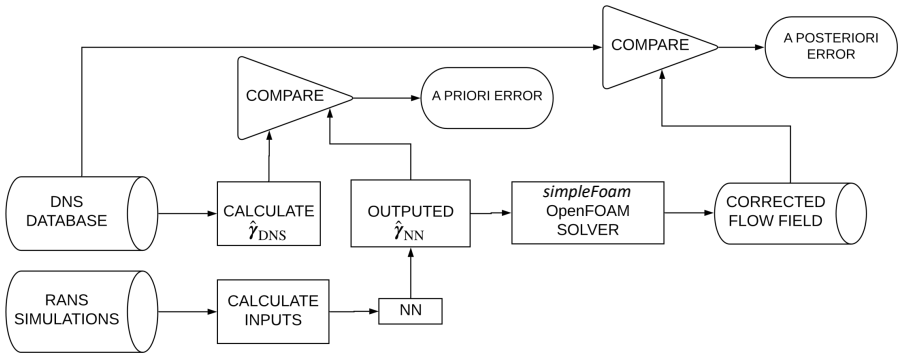


Figure 6.6: $\hat{\gamma}$ neural networks testing flow-chart

6.4.2 Prediction of t and R

Two alternative sets of networks were trained to predict the six components of the tensor $\mathbf{R}$ and the three components of the vector $\mathbf{t}$ respectively. Networks' architectures, adaptative learning rates, activation functions, loss function and optimizer were the same as the ones used in the $\hat{\mathbf{\Gamma}}$ and $\hat{\gamma}$ networks. Inputs differed only by the exclusion of the components of $\hat{\mathbf{\Gamma}}$, $\nabla \cdot \hat{\mathbf{\Gamma}}$, $\nabla\hat{\gamma}$ and $\hat{\gamma}$ listed in table 6.2.

Train, test and validation groups were also the same in order to compare predictions and corrected flow fields of the four methodologies. For the injection of both $\mathbf{R}$ and $\mathbf{t}$ in the κ - ϵ environment, other two OpenFOAM turbulence models were programmed and used with the *simpleFoam* solver.

Table 6.5 summarizes the networks configuration used for the prediction of $\hat{\mathbf{\Gamma}}$, $\hat{\gamma}$, $\mathbf{t}$ and $\mathbf{R}$. Table 6.6 summarizes the network configuration used for the prediction of the boundary conditions $\mathbf{t}_{\text{wall}}$ to be used on the RFVTE, along with the predicted $\hat{\gamma}$.

Table 6.5: Summary of $\hat{\mathbf{\Gamma}}$, $\hat{\gamma}$, $\mathbf{t}$ and $\mathbf{R}$ networks

Number of layers	2
Neurons/layer	100
$\varphi(v)$	$\tanh(v)$
Batch-size	32
Starting α	1×10^{-3}
α **decay rate**	0.6
Number of epochs to reduce α	5
Number of epochs to stop training	20

Table 6.6: Summary of $\mathbf{t}_{\text{wall}}$ network

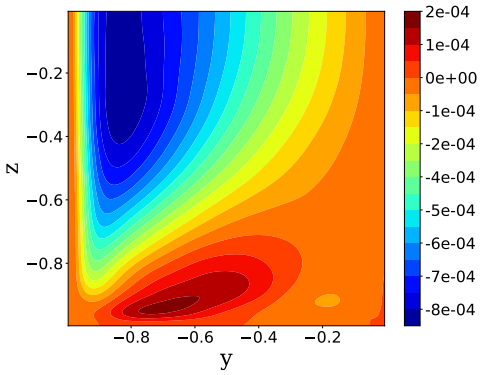
Number of layers	1
Neurons/layer	50
$\varphi(v)$	$\tanh(v)$
Batch-size	5
Starting α	5×10^{-3}
α **decay rate**	0.8
Number of epochs to reduce α	20
Number of epochs to stop training	300

6.4.3 Post-processing Predicted Quantities

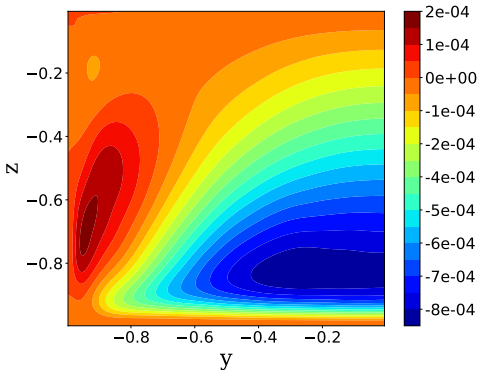
The NN outputs $\hat{\mathbf{\Gamma}}$, $\hat{\boldsymbol{\gamma}}$, $\mathbf{t}_{\text{wall}}$, $\mathbf{t}$ and $\mathbf{R}$ were subjected to a post-processing stage after being predicted. This post-processing guaranteed the symmetry in the predicted components, when it existed. It also guaranteed that different predicted components that should be mirrored with relation to one another are indeed. This is beneficial in two aspects, first, in general it considerably reduces the discrepancy between the DNS and the NN values. Second, it helps numeric convergence and leads to better corrected results when injected into the RANS environment.

These symmetries are imposed by averaging the symmetric parts. For example, Γ_{xx}, $\hat{\gamma}_x$, t_x and R_{xx} components are symmetric with respect to the diagonal line dividing a quadrant in half. An example of components that are mirrored with relation to one another are $\hat{\gamma}_y$ and $\hat{\gamma}_z$. Components Γ_{yy} and Γ_{zz}, Γ_{xy} and Γ_{xz}, R_{yy} and R_{zz}, R_{xy} and R_{xz}, t_y and t_z also contain the same values, whose distribution in space is mirrored.

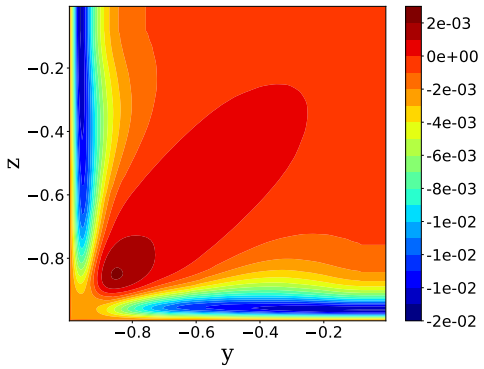
Plots of symmetric components in figure 6.7 demonstrate this characteristic, present in many of the square-ducts quantities.



(a) DNS R_{xy} component



(b) DNS R_{xz} component



(c) DNS t_x component

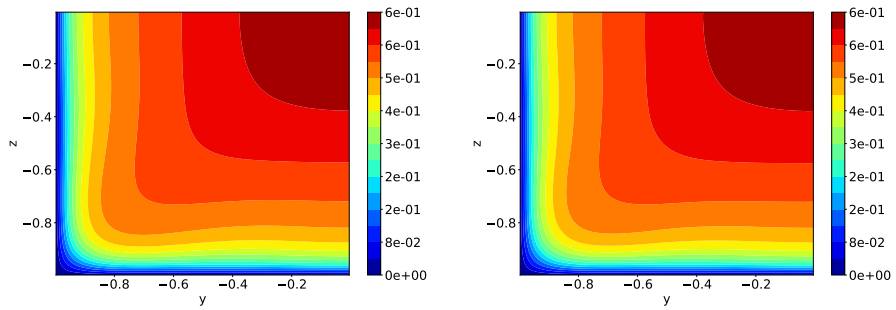
Figure 6.7: Symmetric and mirrored components demonstration

Chapter 7

Results

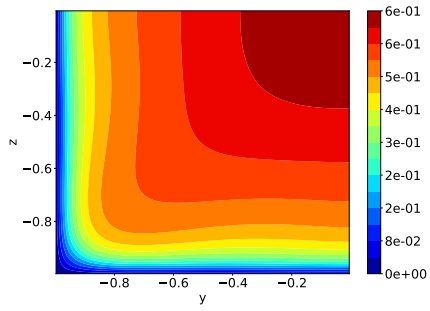
Before training the networks and evaluating their predictions, the four methodologies capabilites were evaluated by directly injecting the DNS values of the NN targets into the κ - ϵ environment. Since the best NN performance would be the scenario in which it perfectly predicted the DNS targets, it is important to know beforehand how accurate the reconstructed $\mathbf{u}$ can be in this case. This also serves as an upper performance boundary for the presented methodologies.

The reconstructed fields were obtained using the improved dataset post-processed by RANGEL (2019), all NN were also trained using it. Figure 7.1 compares the main flow u_x reconstructed by $\mathbf{R}_{\text{DNS}}$, $\mathbf{t}_{\text{DNS}}$, $\hat{\mathbf{\Gamma}}_{\text{DNS}}$ and $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$ with the DNS u_x. Reconstructed secondary flows components u_y and u_z are shown in figures 7.2 and 7.3 respectively. Reconstructed $\mathbf{R}$ and $\mathbf{t}$ are depicted in figures 7.4 and 7.5 respectively. All depicted fields correspond to the $Re = 2900$ simulation.

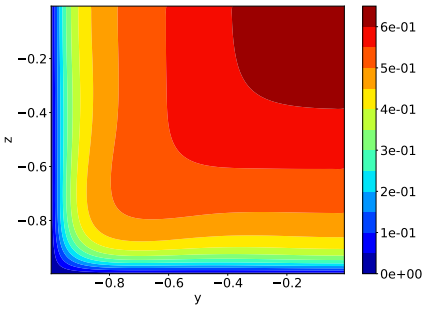


(a) DNS

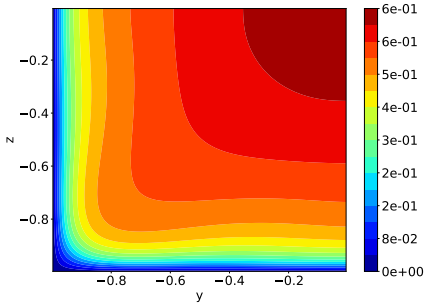
(b) $\mathbf{R}_{\text{DNS}}$



(c) $\mathbf{t}_{\text{DNS}}$

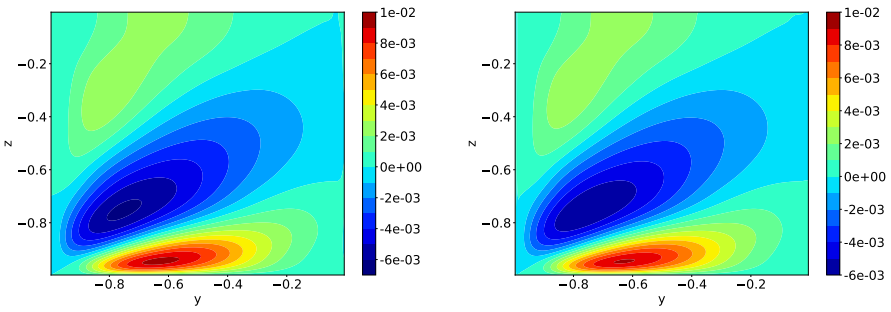


(d) $\hat{\mathbf{\Gamma}}_{\text{DNS}}$



(e) $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$

Figure 7.1: u_x reconstructed by different methodologies ($Re = 2900$)



(a) DNS

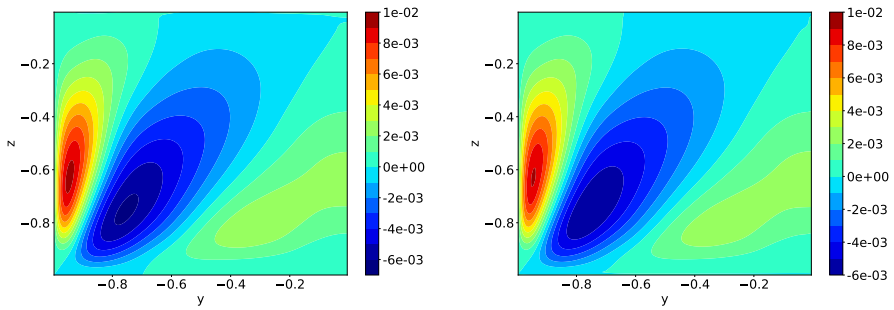
(b) $\mathbf{R}_{\mathrm{DNS}}$

(c) $\mathbf{t}_{\mathrm{DNS}}$

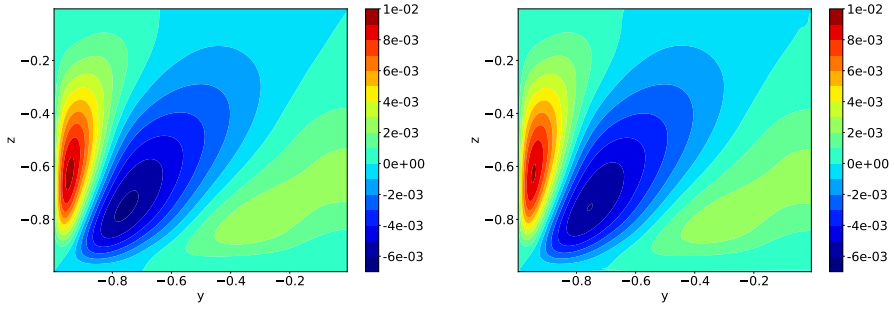
(d) $\hat{\mathbf{\Gamma}}_{\mathrm{DNS}}$

(e) $\hat{\boldsymbol{\gamma}}_{\mathrm{DNS}}$

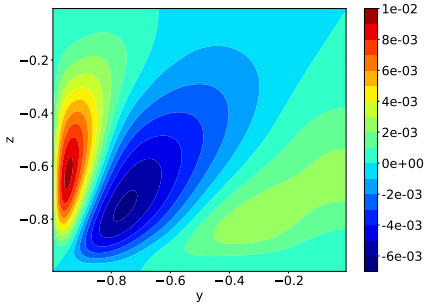
Figure 7.2: u_y reconstructed by different methodologies ($Re = 2900$)



(a) DNS (b) $\mathbf{R}_{\mathrm{DNS}}$

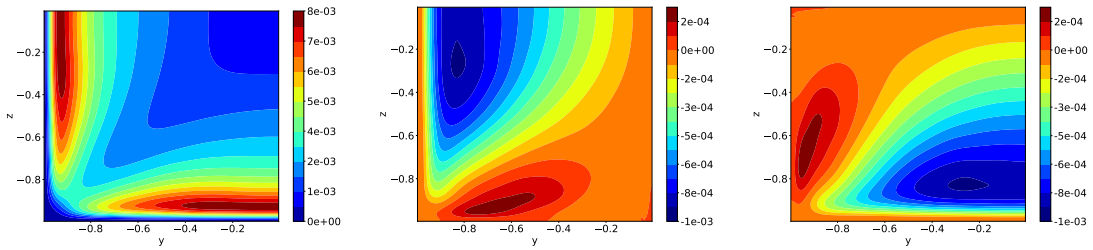


(c) $\mathbf{t}_{\mathrm{DNS}}$ (d) $\hat{\mathbf{\Gamma}}_{\mathrm{DNS}}$

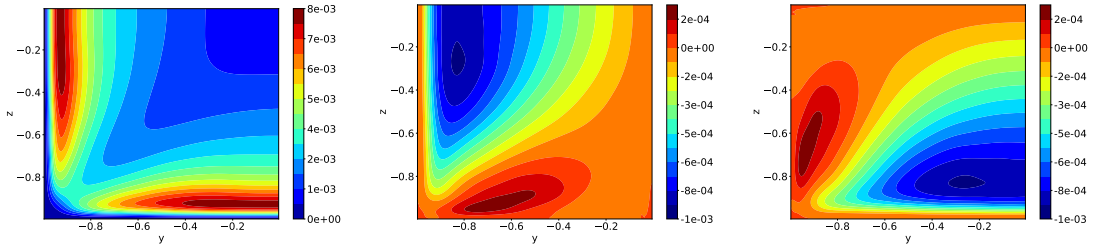


(e) $\hat{\boldsymbol{\gamma}}_{\mathrm{DNS}}$

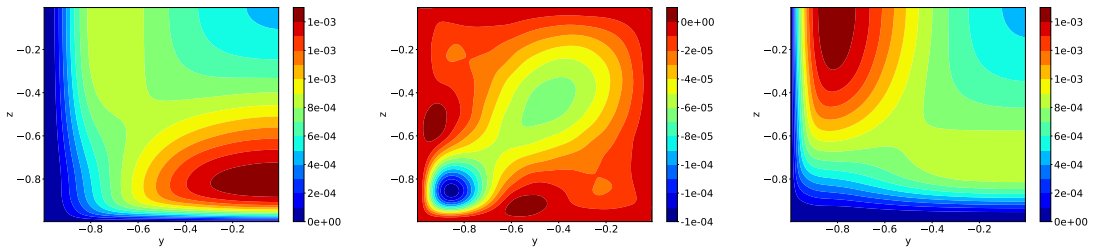
Figure 7.3: u_z reconstructed by different methodologies ($Re = 2900$)



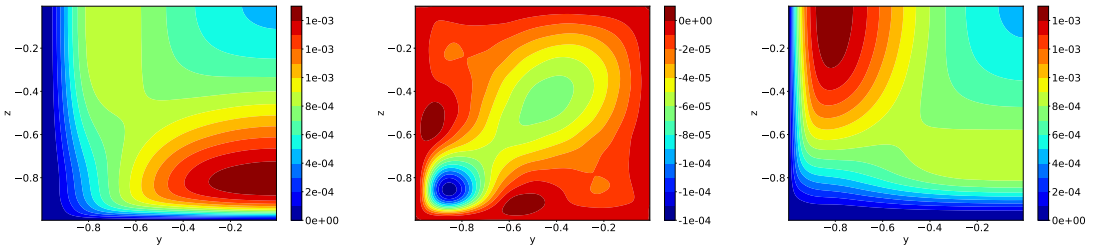
(a) DNS R_{xx} (b) DNS R_{xy} (c) DNS R_{xz}



(d) R_{xx} reconstructed by $\hat{\mathbf{\Gamma}}_{\text{DNS}}$ (e) R_{xy} reconstructed by $\hat{\mathbf{\Gamma}}_{\text{DNS}}$ (f) R_{xz} reconstructed by $\hat{\mathbf{\Gamma}}_{\text{DNS}}$

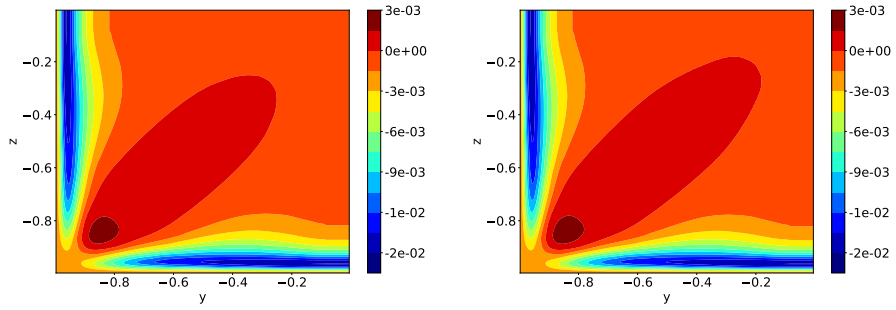


(g) DNS R_{yy} (h) DNS R_{yz} (i) DNS R_{zz}



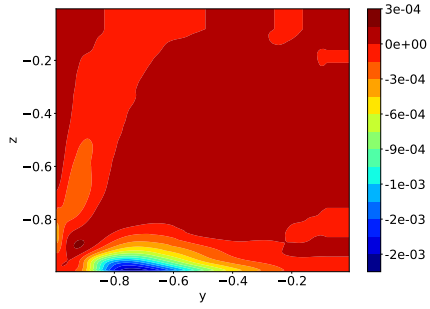
(j) R_{yy} reconstructed by $\hat{\mathbf{\Gamma}}_{\text{DNS}}$ (k) R_{yz} reconstructed by $\hat{\mathbf{\Gamma}}_{\text{DNS}}$ (l) R_{xx} reconstructed by $\hat{\mathbf{\Gamma}}_{\text{DNS}}$

Figure 7.4: Reconstructed $\mathbf{R}$ components ($Re = 2900$)

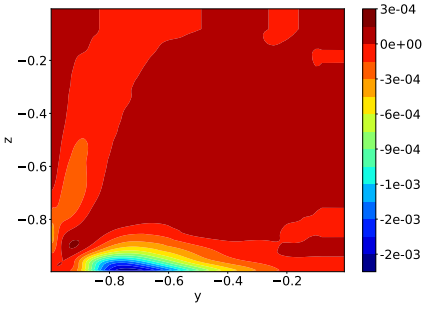


(a) DNS t_x

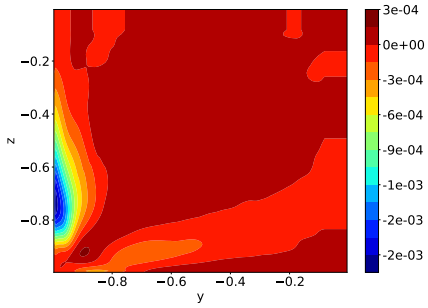
(b) Reconstructed t_x



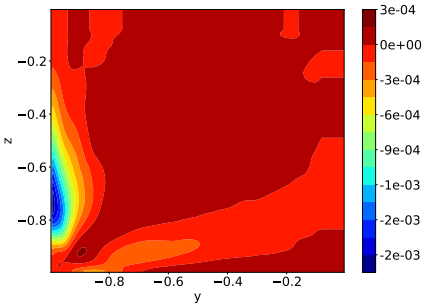
(c) DNS t_y



(d) Reconstructed t_y



(e) DNS t_z



(f) Reconstructed t_z

Figure 7.5: Reconstructed $\mathbf{t}$ components ($Re = 2900$)

7.1 A Priori Results

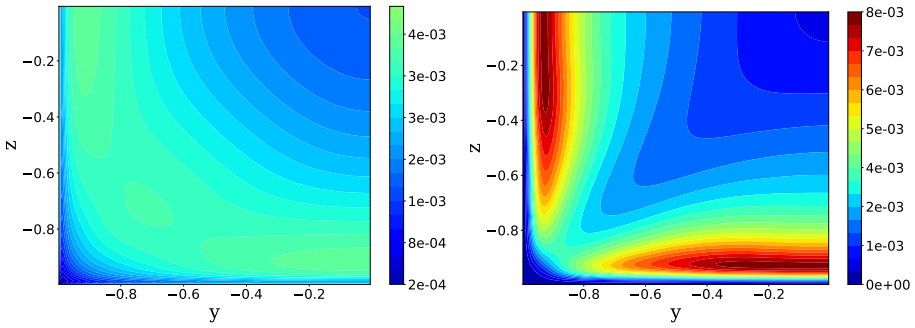
The *a priori* results consisted on the NN predictions before being injected into the CFD environments and being propagated by it. First, the predicted $\mathbf{R}_{\text{NN}}$ is shown, following, $\mathbf{t}_{\text{NN}}$ is demonstrated, then the proposed RSTE source term $\hat{\mathbf{\Gamma}}_{\text{NN}}$ is presented and lastly the proposed the RFVTE source term $\hat{\boldsymbol{\gamma}}_{\text{NN}}$ and the boundary conditions $\mathbf{t}_{\text{wall,NN}}$ are depicted. All results correspond to the test dataset of $Re = 2900$.

7.1.1 Prediction of R

Four components of predicted $\mathbf{R}_{\text{NN}}$ are demonstrated in contour plots on figures 7.6, 7.8, 7.10 and 7.12. Samples on four different locations of the square-duct's domain are shown in figures 7.7, 7.9, 7.11 and 7.13. Results are compared with the baseline RANS and the high-fidelity DNS tensor.

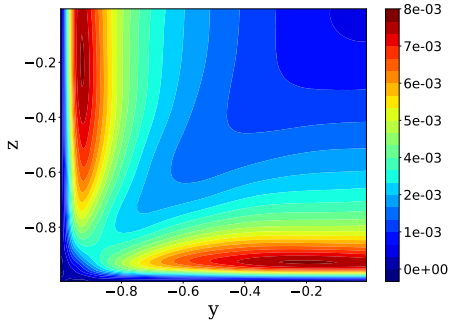
The omitted components R_{xz} and R_{zz} are not shown because their values are, respectively, components R_{xy} and R_{yy} mirrored. This characteristic is ensured in the predicted quantities by the averaging post-processing step mentioned in chapter 6. The omitted plots can be checked in the appendix B.

The post-processing also ensures that predicted components R_{xx} and R_{yz} are exactly symmetric with respect to the duct's quadrant diagonal. The trained neural networks are not capable of providing perfectly symmetric results because their parameters are updated after a batch of training points are passed through it. In the case that the training batch-size was of 1 data point per update, it would ensure exact symmetry, but this greatly increases computational costs.



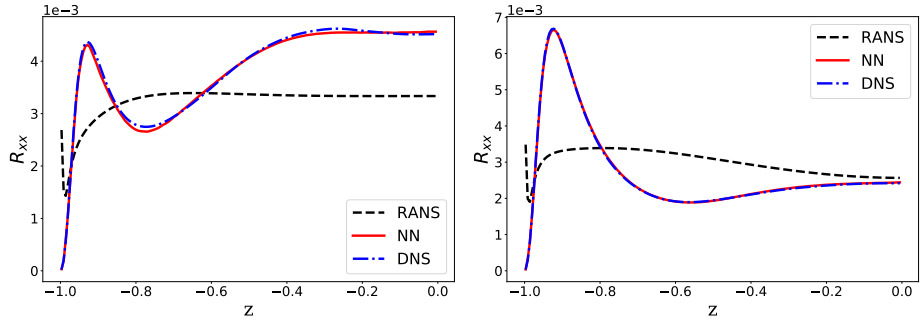
(a) R_{xx} RANS

(b) R_{xx} DNS

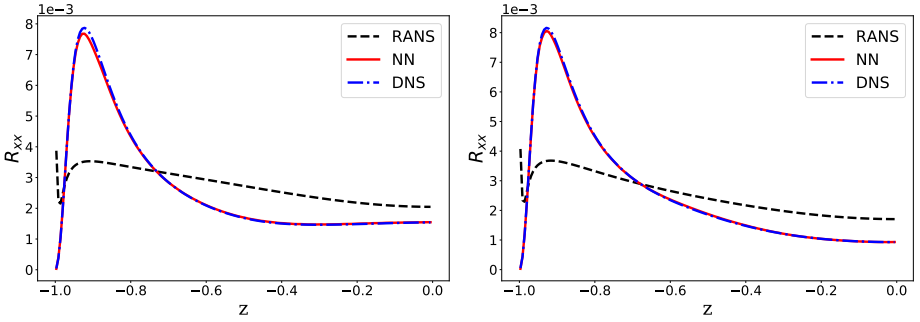


(c) R_{xx} predicted by NN

Figure 7.6: RANS, DNS and predicted R_{xx} for the test case of $Re = 2900$

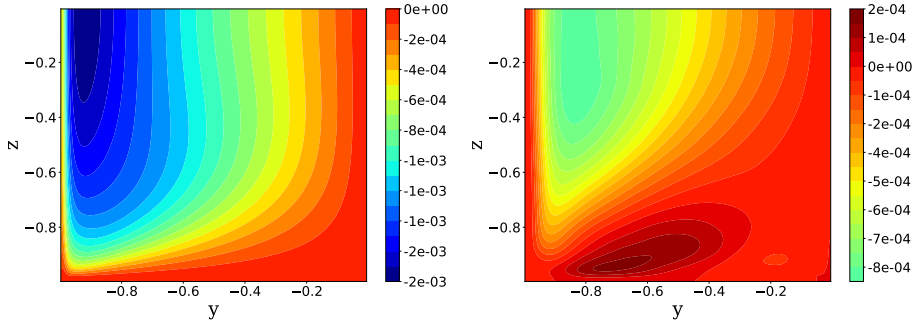


(a) $y = -0.8$ (b) $y = -0.6$

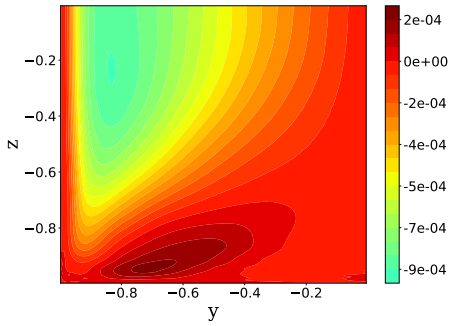


(c) $y = -0.4$ (d) $y = -0.2$

Figure 7.7: R_{xx} samples

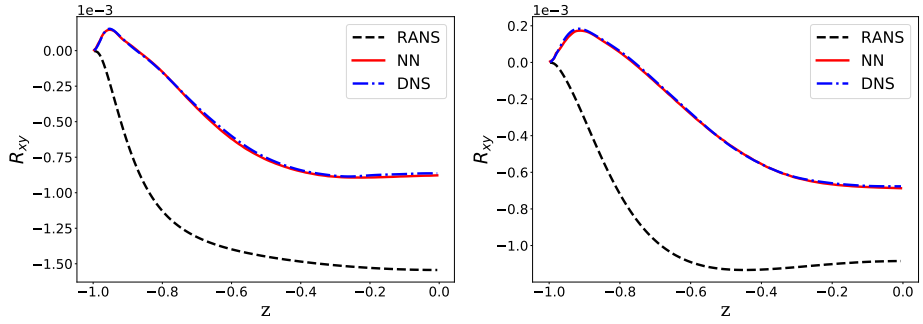


(a) R_{xy} RANS (b) R_{xy} DNS



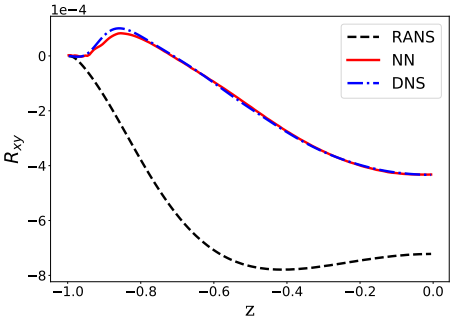
(c) R_{xy} predicted by NN

Figure 7.8: RANS, DNS and predicted R_{xy} for the test case of $Re = 2900$

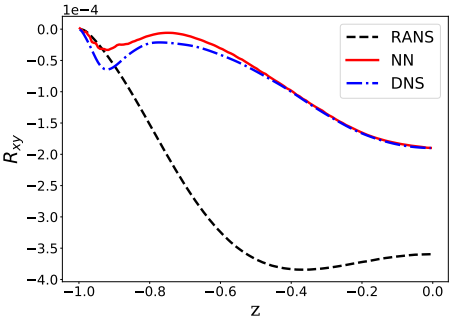


(a) $y = -0.8$

(b) $y = -0.6$

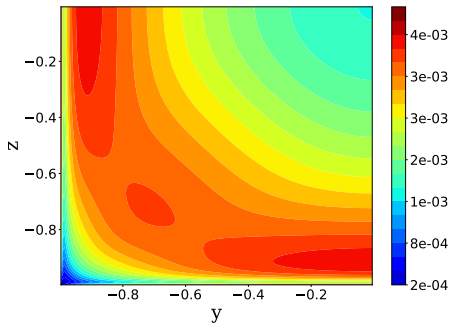


(c) $y = -0.4$

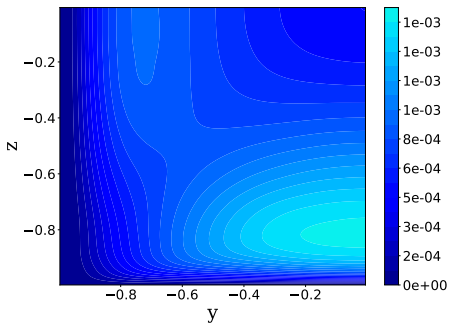


(d) $y = -0.2$

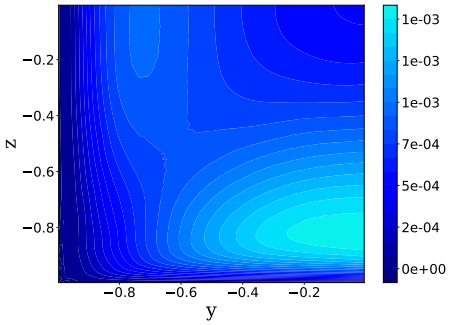
Figure 7.9: R_{xy} samples



(a) R_{yy} RANS

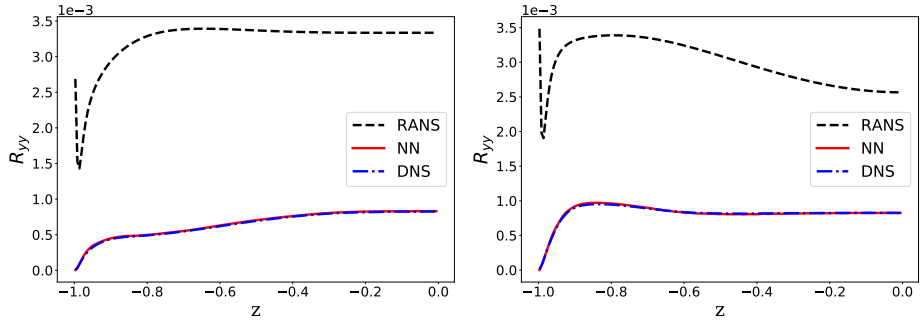


(b) R_{yy} DNS

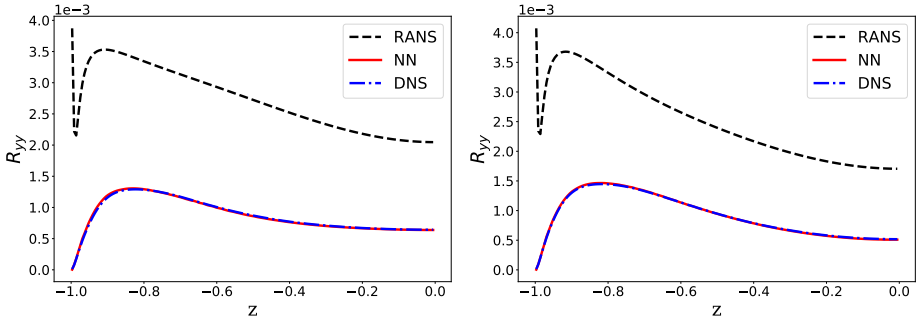


(c) R_{yy} predicted by NN

Figure 7.10: RANS, DNS and predicted R_{yy} for the test case of $Re = 2900$

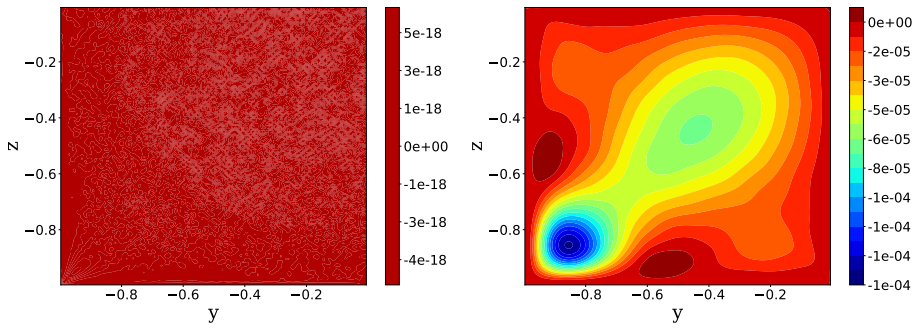


(a) $y = -0.8$ (b) $y = -0.6$

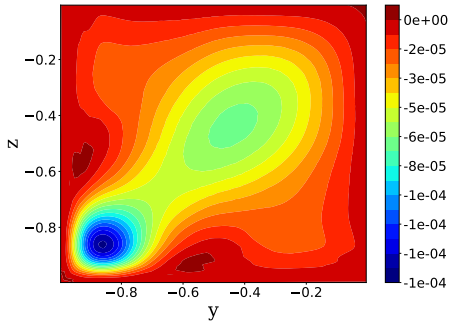


(c) $y = -0.4$ (d) $y = -0.2$

Figure 7.11: R_{yy} samples

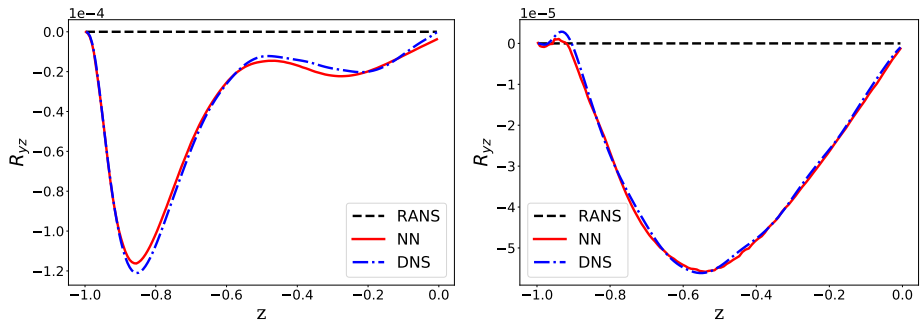


(a) R_{yz} RANS (b) R_{yz} DNS



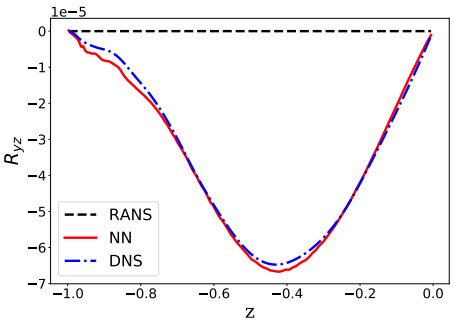
(c) R_{yz} predicted by NN

Figure 7.12: RANS, DNS and predicted R_{yz} for the test case of $Re = 2900$

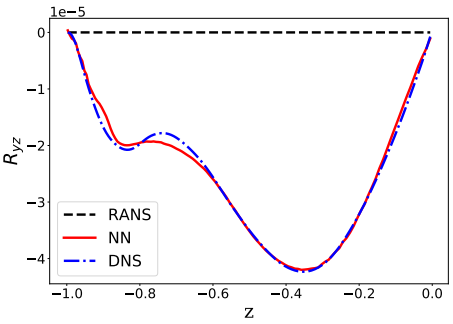


(a) $y = -0.8$

(b) $y = -0.6$



(c) $y = -0.4$



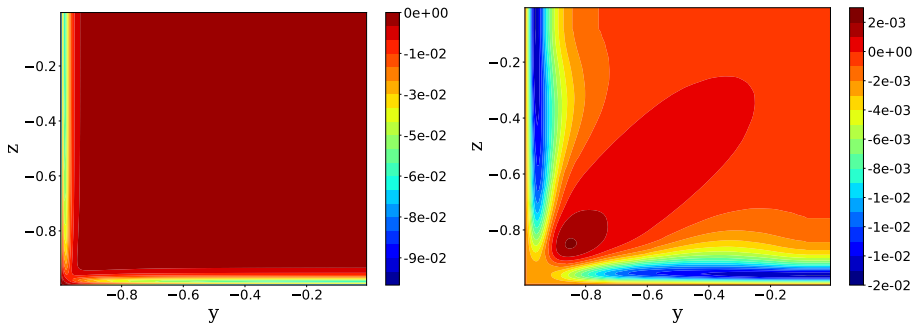
(d) $y = -0.2$

Figure 7.13: R_{yz} samples

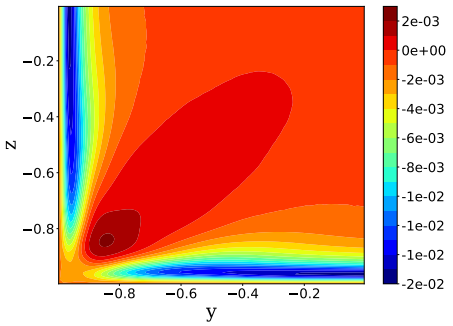
7.1.2 Prediction of t

Two out of three components of the predicted $\mathbf{t}_{\mathrm{NN}}$ are shown in comparison with RANS and DNS values in figures 7.14 and 7.16. Samples on four y coordinates are shown in figures 7.15 and 7.17.

Analogously to components R_{xy} and R_{xz}, components t_z and t_y are mirrored with respect to each other, therefore t_z plots are omitted from the current chapter for conciseness, but can be checked in appendix B.

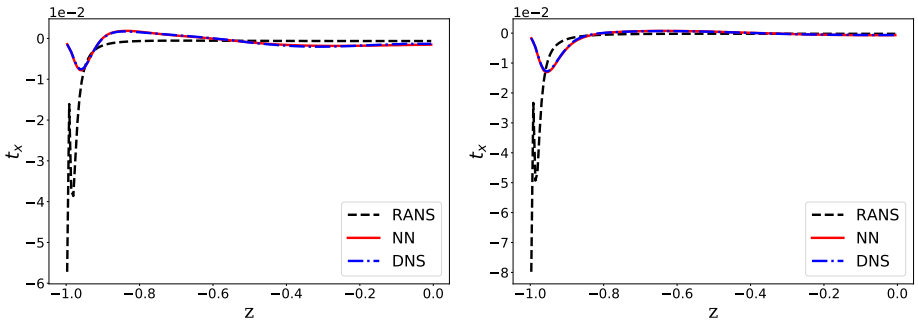


(a) t_x RANS (b) t_x DNS

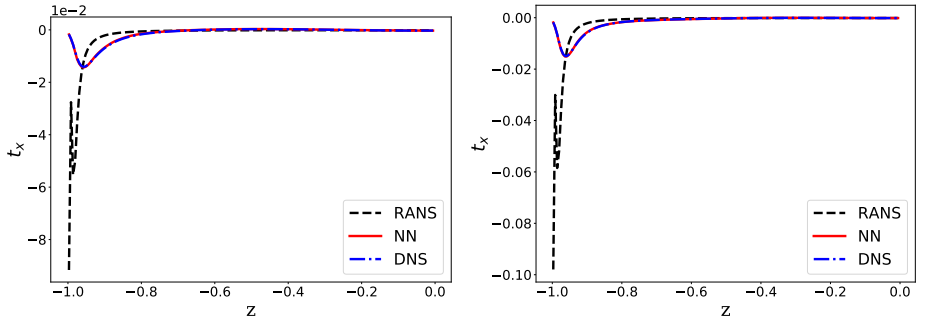


(c) t_x predicted by NN

Figure 7.14: RANS, DNS and predicted t_x for the test case of $Re = 2900$

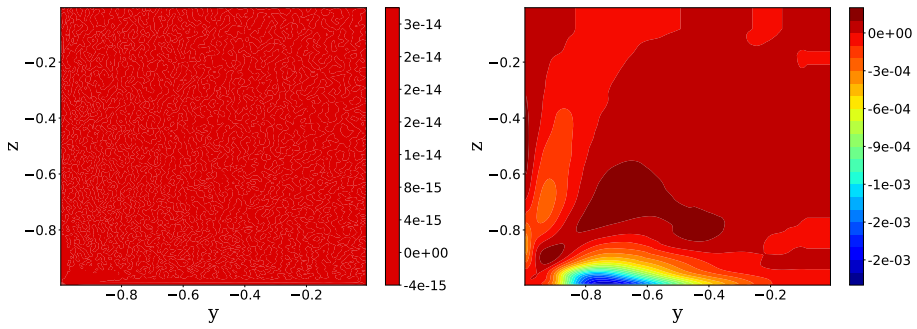


(a) $y = -0.8$ (b) $y = -0.6$

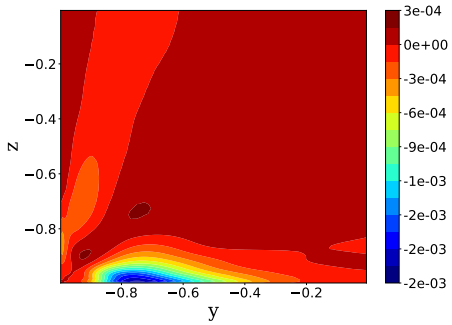


(c) $y = -0.4$ (d) $y = -0.2$

Figure 7.15: t_x samples

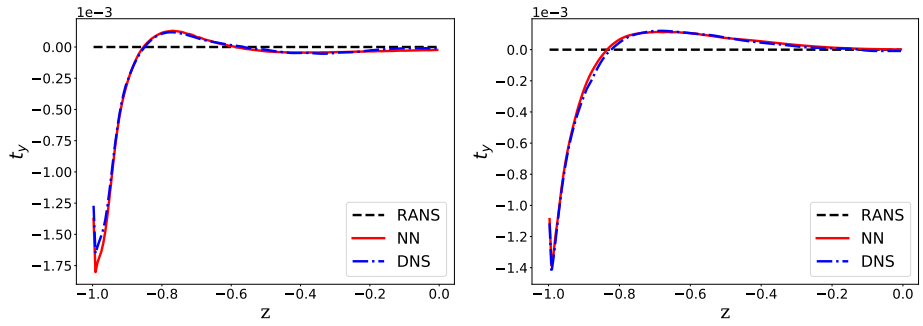


(a) t_y RANS (b) t_y DNS

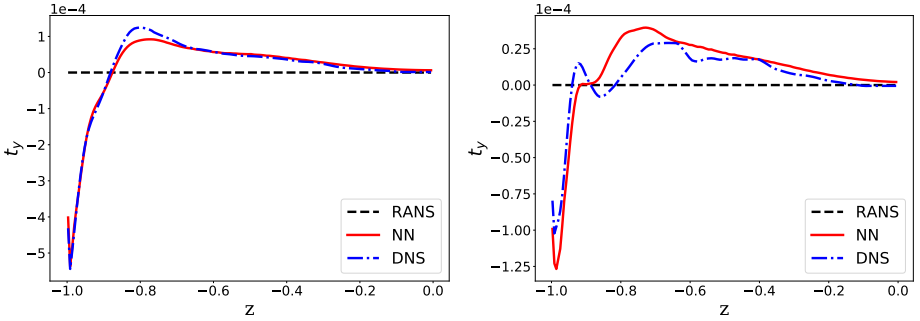


(c) t_y predicted by NN

Figure 7.16: RANS, DNS and predicted t_y for the test case of $Re = 2900$



(a) $y = -0.8$ (b) $y = -0.6$



(c) $y = -0.4$ (d) $y = -0.2$

Figure 7.17: t_y samples

As can be checked in figures 7.16 (c) and 7.17 (d), the NN has a moderate difficulty in predicting the modified RFV components in the duct's center region.

7.1.3 Prediction of $\hat{\mathbf{\Gamma}}$

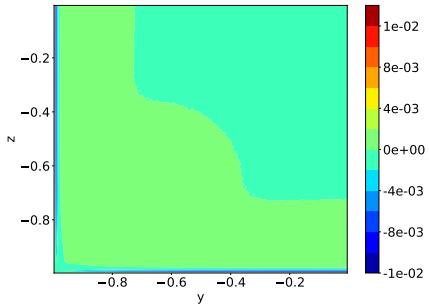
For the same reasos as the tensor $\mathbf{R}$, only four out of six components of the source term $\hat{\mathbf{\Gamma}}$ are presented, omitted plots can also be checked on appendix B.

Figures 7.18, 7.20, 7.22 and 7.24 presents the predicted $\hat{\Gamma}_{xx}$, $\hat{\Gamma}_{xy}$, $\hat{\Gamma}_{yy}$ and $\hat{\Gamma}_{yz}$ respectively. Each of the four components are also compared with RANS and DNS fields. The same four components are shown in more detail on four different locations on the square-duct quadrant in figures 7.19, 7.21, 7.23 and 7.25.

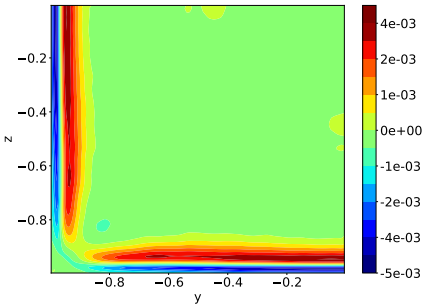
Figures 7.21 and 7.25 draw attention to the discontinuities in the DNS fields. This beahviour probably occurs due to the low density of DNS points provided in the employed database. For the $Re = [2200, 2400, 2600]$, a 49×49 mesh is given in the duct's quadrant, totalizing only 2401 points. In the other three Re cases, a mesh of 65×65 points is provided, totalizing 4225 points. Obviously this is not the mesh utilized by PINELLI *et al.* (2010), but rather a small sample of the DNS dataset. In order to be used in the present work, the given meshes were interpolated to the 125×125 mesh by using a cubic spline.

Since calculating $\hat{\mathbf{\Gamma}}$ involves up to second order derivatives on the $\mathbf{R}$ DNS field, doing so in a such a low density mesh is probably the cause of the evidenced discontinuities. Another probable cause is the statistical uncertainties in high-fidelity turbulent stresses exposed by THOMPSON *et al.* (2016) and CRUZ *et al.* (2019).

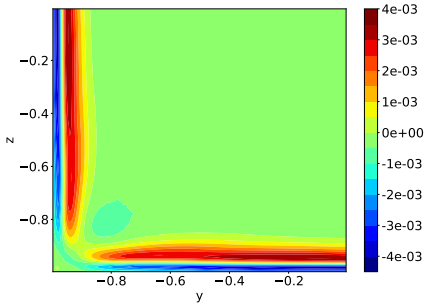
Despite this aspect of the $\hat{\mathbf{\Gamma}}_{\mathrm{DNS}}$ field and their effects on the networks' training, NN do achieve reasonable results. The smooth nature of the employed activation function $\varphi(v) = \tanh(v)$ ensures that all predictions will also be smooth across the computational domain.



(a) $\hat{\Gamma}_{xx}$ RANS

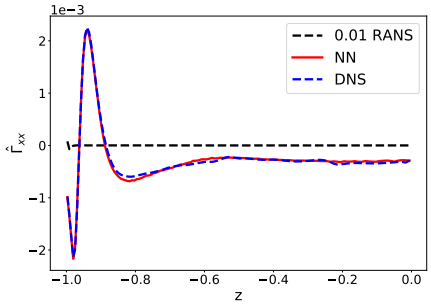


(b) $\hat{\Gamma}_{xx}$ DNS

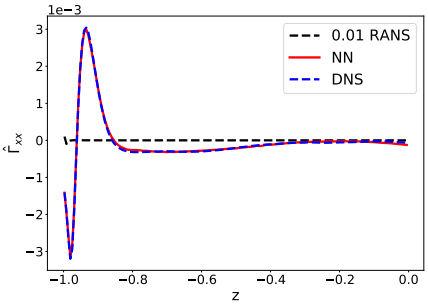


(c) $\hat{\Gamma}_{xx}$ predicted by NN

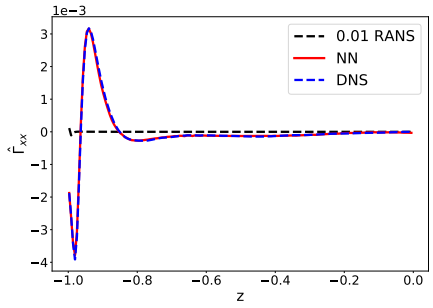
Figure 7.18: RANS, DNS and predicted $\hat{\Gamma}_{xx}$ for the test case of $Re = 2900$



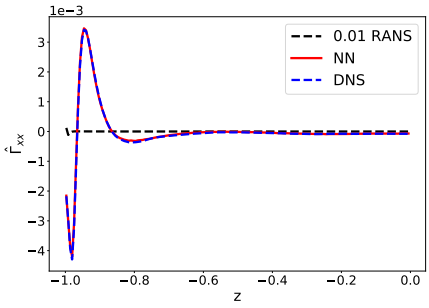
(a) $y = -0.8$



(b) $y = -0.6$

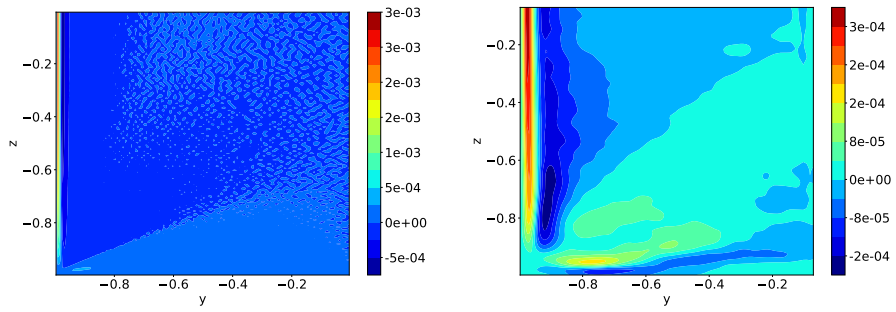


(c) $y = -0.4$

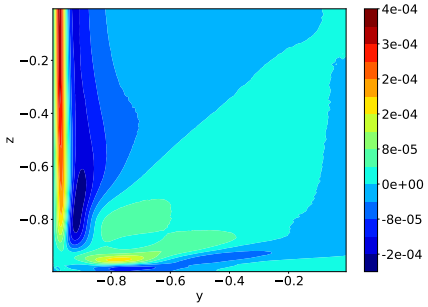


(d) $y = -0.2$

Figure 7.19: $\hat{\Gamma}_{xx}$ samples

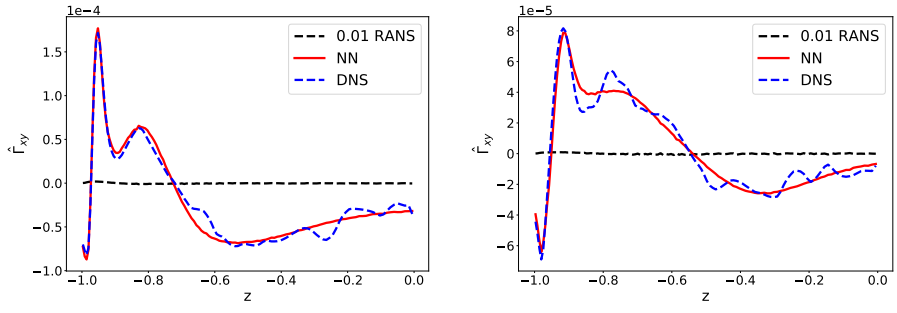


(a) $\hat{\Gamma}_{xy}$ RANS (b) $\hat{\Gamma}_{xy}$ DNS



(c) $\hat{\Gamma}_{xy}$ predicted by NN

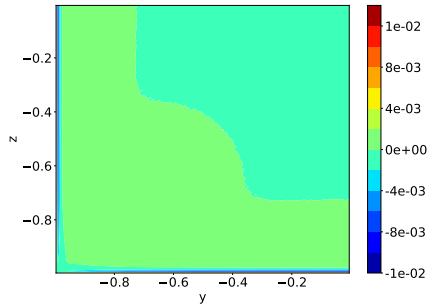
Figure 7.20: RANS, DNS and predicted $\hat{\Gamma}_{xy}$ for the test case of $Re = 2900$



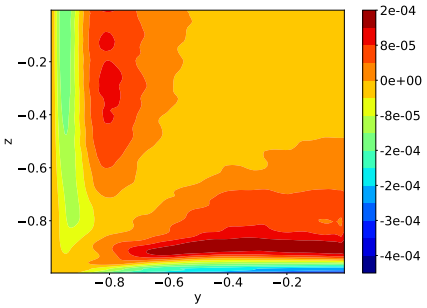
(a) $y = -0.8$ (b) $y = -0.6$

(c) $y = -0.4$ (d) $y = -0.2$

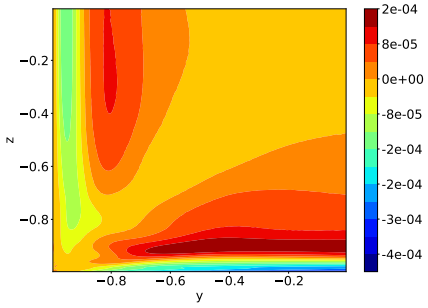
Figure 7.21: $\hat{\Gamma}_{xy}$ samples



(a) $\hat{\Gamma}_{yy}$ RANS

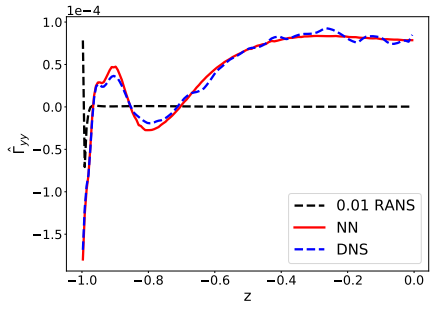


(b) $\hat{\Gamma}_{yy}$ DNS

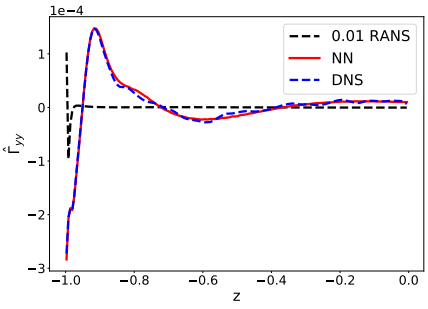


(c) $\hat{\Gamma}_{yy}$ predicted by NN

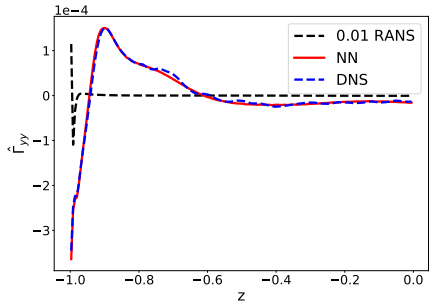
Figure 7.22: RANS, DNS and predicted $\hat{\Gamma}_{yy}$ for the test case of $Re = 2900$



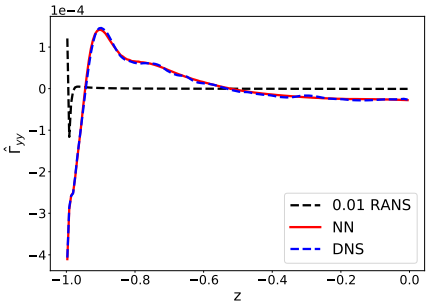
(a) $y = -0.8$



(b) $y = -0.6$

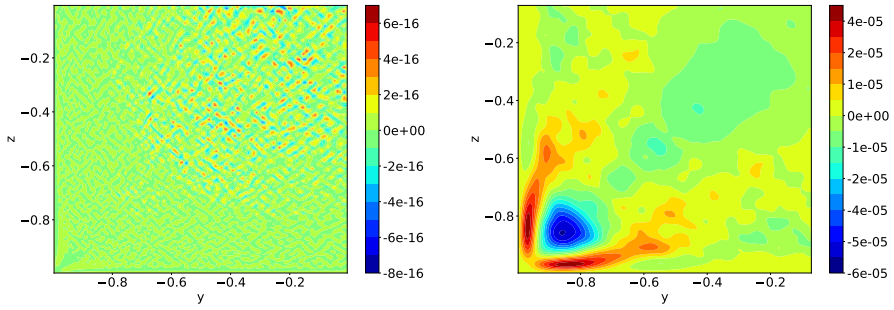


(c) $y = -0.4$



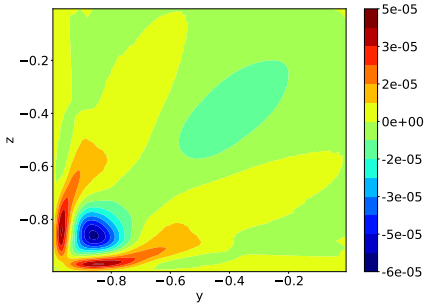
(d) $y = -0.2$

Figure 7.23: $\hat{\Gamma}_{yy}$ samples



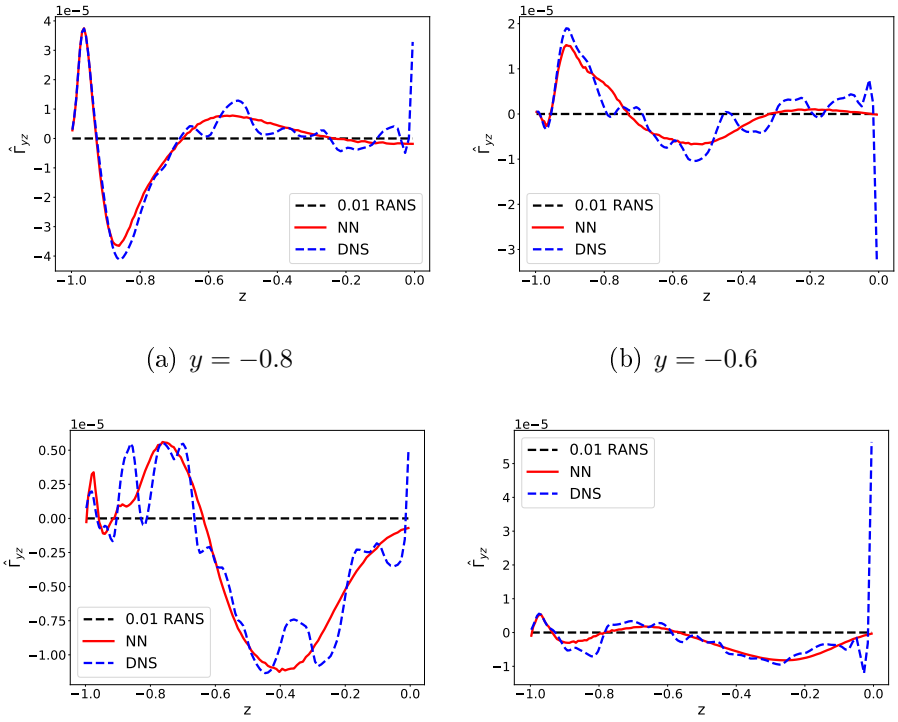
(a) $\hat{\Gamma}_{yz}$ RANS

(b) $\hat{\Gamma}_{yz}$ DNS



(c) $\hat{\Gamma}_{yz}$ predicted by NN

Figure 7.24: RANS, DNS and predicted $\hat{\Gamma}_{yz}$ for the test case of $Re = 2900$



(a) $y = -0.8$

(b) $y = -0.6$

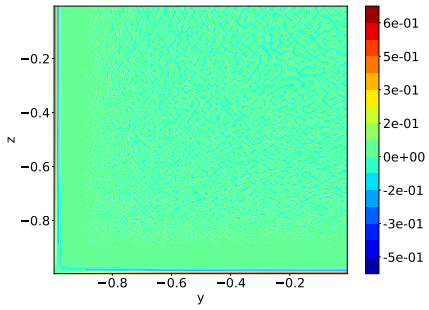
(c) $y = -0.4$

(d) $y = -0.2$

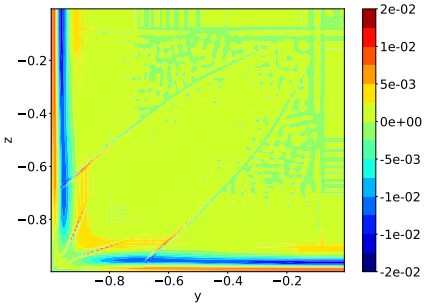
Figure 7.25: $\hat{\Gamma}_{yz}$ samples

7.1.4 Prediction of $\hat{\gamma}$

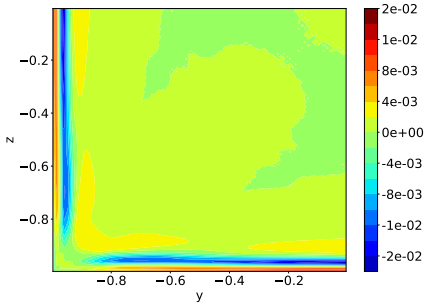
NN results for the source term $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$ components are shown and also compared with RANS and DNS values in figures 7.26 and 7.28. As in the vector $\mathbf{t}$, the components y and z of the vector $\hat{\boldsymbol{\gamma}}$ are mirrored versions of each other, therefore the plots for component $\hat{\gamma}_z$ can also be checked in appendix B. Samples of $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$ on four different coordinates are shown in figures 7.27 and 7.29.



(a) $\hat{\gamma}_x$ RANS

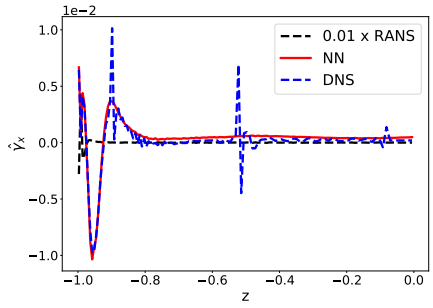


(b) $\hat{\gamma}_x$ DNS

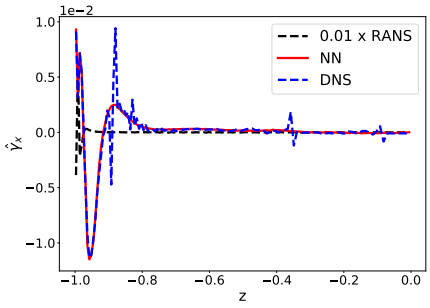


(c) $\hat{\gamma}_x$ predicted by NN

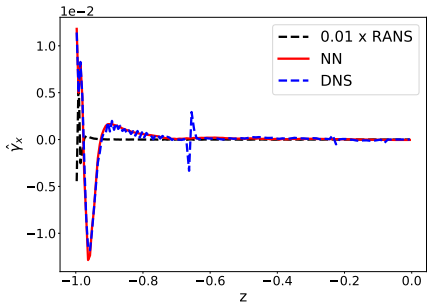
Figure 7.26: RANS, DNS and predicted $\hat{\gamma}_x$ for the test case of $Re = 2900$



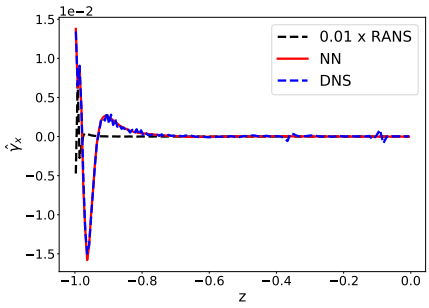
(a) $y = -0.8$



(b) $y = -0.6$

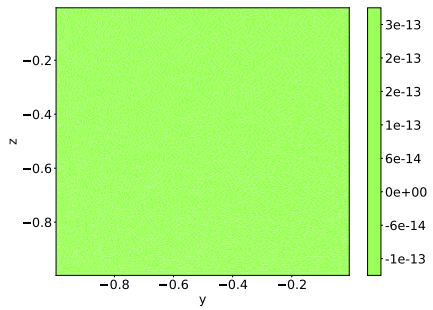


(c) $y = -0.4$

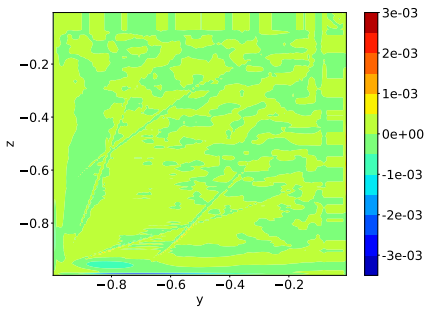


(d) $y = -0.2$

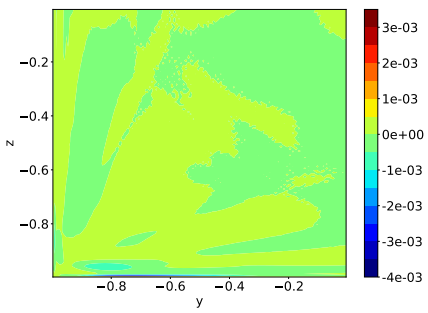
Figure 7.27: $\hat{\gamma}_x$ samples



(a) $\hat{\gamma}_y$ RANS

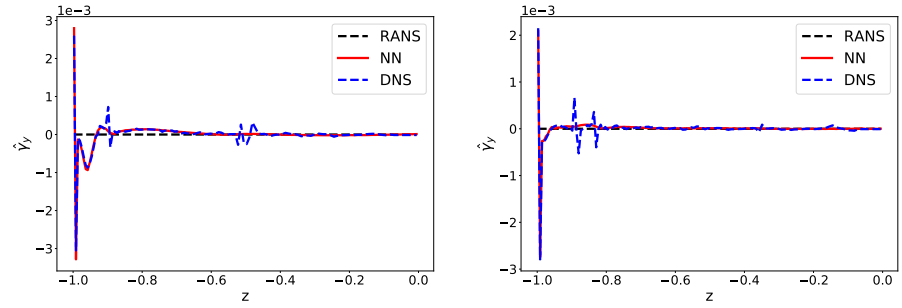


(b) $\hat{\gamma}_y$ DNS



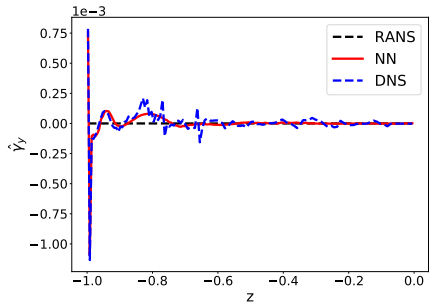
(c) $\hat{\gamma}_y$ predicted by NN

Figure 7.28: RANS, DNS and predicted $\hat{\gamma}_y$ for the test case of $Re = 2900$

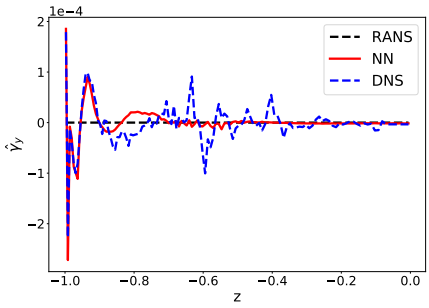


(a) $y = -0.8$

(b) $y = -0.6$



(c) $y = -0.4$



(d) $y = -0.2$

Figure 7.29: $\hat{\gamma}_y$ samples

Similarly to the source term $\hat{\mathbf{\Gamma}}$, figures 7.26 (b), 7.27 and 7.29 also demonstrate moderate discontinuities in the DNS $\hat{\boldsymbol{\gamma}}$ field. The reason behind this behaviour is probably the same as discussed in the previous subsection 7.1.3. Although $\hat{\boldsymbol{\gamma}}$ is calculated by only manipulating the DNS velocity field, avoiding the problems in $\mathbf{R}_{\text{DNS}}$, its calculation involves up to fourth order derivatives on $\mathbf{u}_{\text{DNS}}$. Doing this amount of operations on a mesh without a proper spatial resolution is probably the reason behind this defficiency on $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$.

Boundary conditions of t

The predicted boundary values $\mathbf{t}_{\text{wall}}$ are shown in figure 7.30. Only the values for the lower wall, $z = -1.0$, are shown. The x component's values on the upper wall, where $y = -1.0$, are the same. The values of component y in the upper wall are the same as those of component z in the lower wall and vice versa, they were also subjected to the averaging post-processing described in chapter 6.

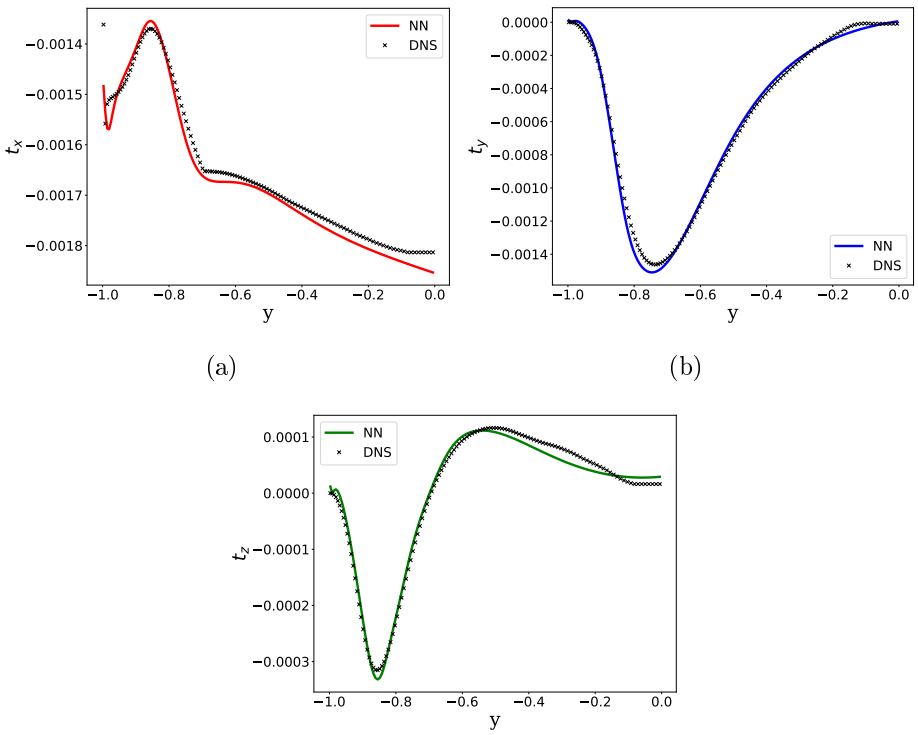


Figure 7.30: $\mathbf{t}_{\text{wall}}$ on $z = -1.0$

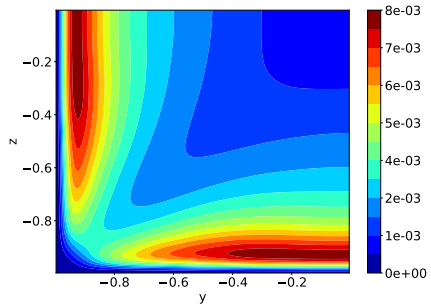
7.2 A Posteriori Results

Corrected turbulent quantities $\mathbf{R}$ and $\mathbf{t}$ are presented at first. Following, the corrected velocity fields $\mathbf{u}$ are shown. Since RANS simulations which use the Boussinesq hypothesis are unable to provide the recirculation in the square-duct and, therefore, the main concern of the machine learning employment in this problem is to correct this defficiency, the recirculation results are presented first. Subsequently, the results for the propagated main flows are presented.

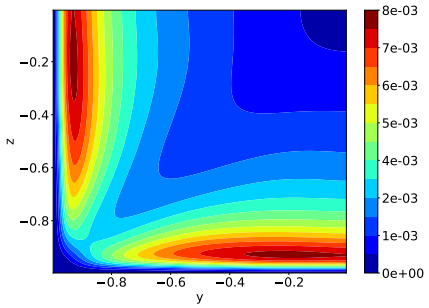
Propagated pressure fields were also calculated by the developed turbulence models, however, their results are omitted since they were null in all contexts, with exception to the $\mathbf{R}$ and $\hat{\mathbf{\Gamma}}$ corrections, where a portion of the deviatoric part of $\mathbf{R}$ gets incorporated into the pressure field. Since the RANS pressure field is null and no DNS p is provided, no true correction to p is possible and the pressure equation serves solely to impose continuity to the corrected flow fields.

7.2.1 Corrected R and t

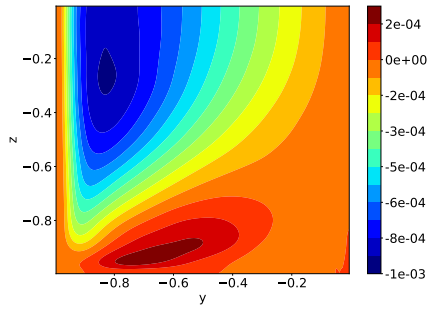
Corrected values of R_{xx}, R_{xy}, R_{yy} and R_{yz} obtained by the propagation of $\hat{\mathbf{\Gamma}}_{\text{NN}}$ in the RSTE turbulence model are demonstrated in figure 7.31. R_{xz} and R_{zz} fields can be checked in appendix B.



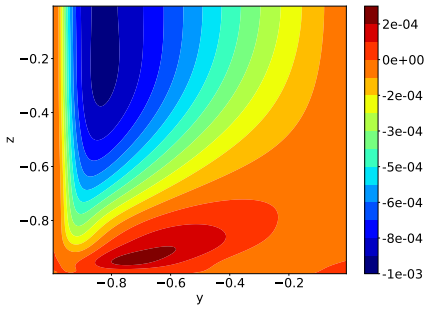
(a) R_{xx} DNS



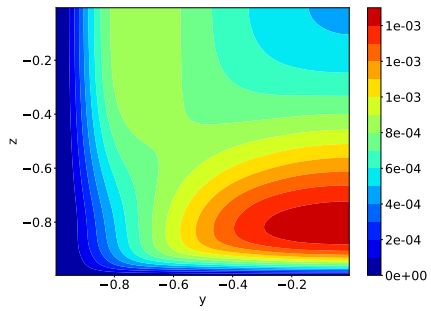
(b) R_{xx} corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$



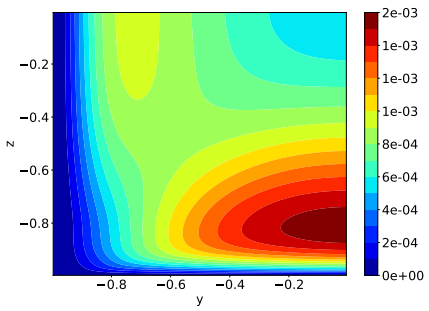
(c) R_{xy} DNS



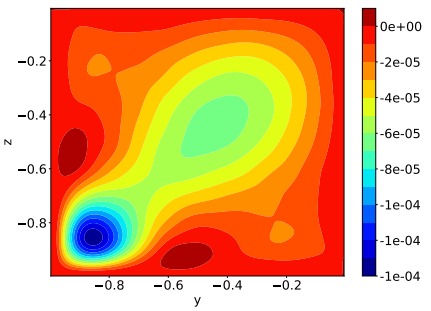
(d) R_{xy} corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$



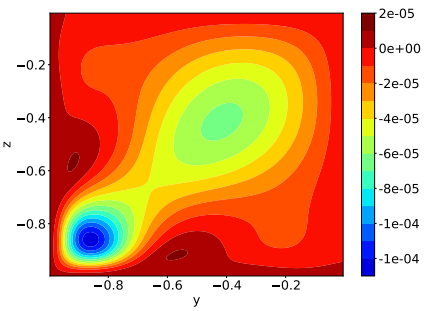
(e) R_{yy} DNS



(f) R_{yy} corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$



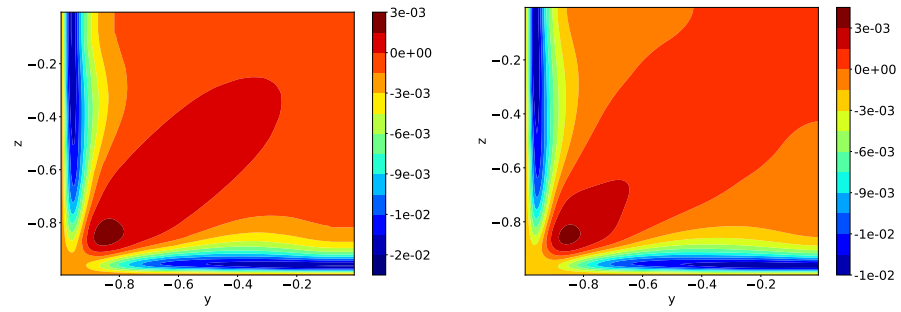
(g) R_{yz} DNS



(h) R_{yz} corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$

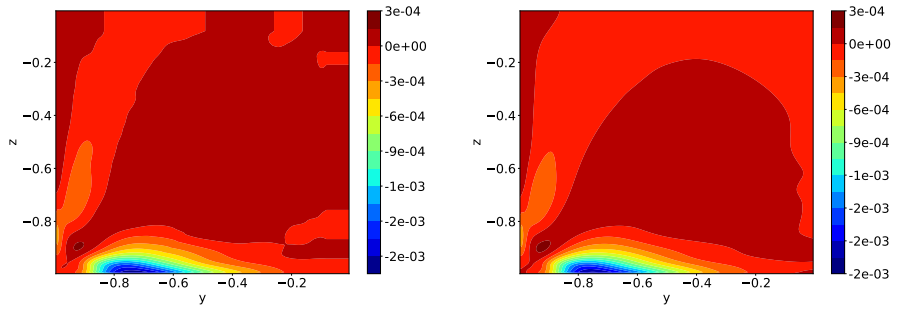
Figure 7.31: DNS and corrected $\mathbf{R}$ components

The corrected x and y components of the modified RFV $\mathbf{t}$ are exposed in comparison with DNS values on figure 7.32.



(a) t_x DNS

(b) t_x corrected by $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$



(c) t_y DNS

(d) t_y corrected by $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$

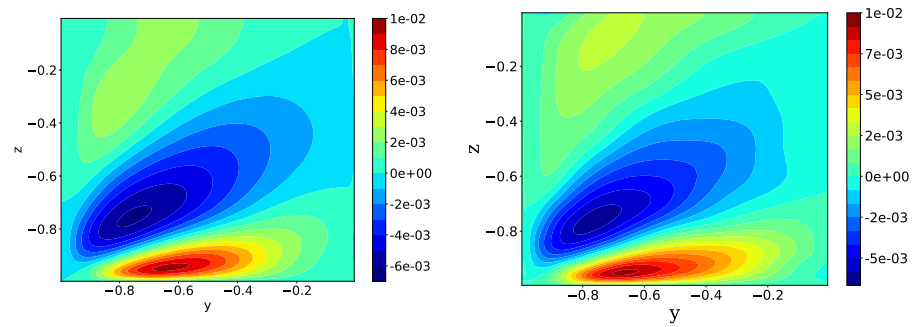
Figure 7.32: DNS and corrected $\mathbf{t}$ components

7.2.2 Recirculation

All four methodologies are able to correct the null recirculation obtained by the RANS simulations. This demonstrates the efficiency of the two methodologies previously presented in literature and also the efficiency of the proposed methodologies, based on the tensor $\hat{\mathbf{\Gamma}}$ and the vector $\hat{\boldsymbol{\gamma}}$ turbulence models.

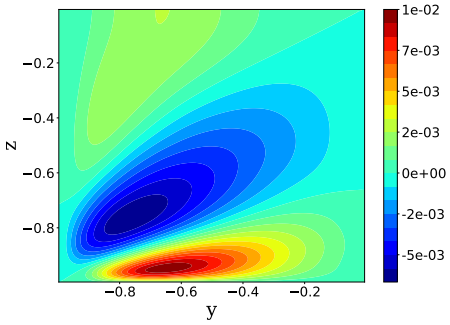
The corrected secondary flow's component u_y is shown in comparison to DNS values in figure 7.33. Corrected u_z components can be checked in appendix B. As exposed in figure 7.3, u_z is symmetric to u_y, this is guaranteed by the implemeted post-processing on NN predictions. In case the post-processing was not implemented, u_y and u_z would present small but visible discrepancies. RANS u_y and u_z are omitted because they are null, as previously noted.

Samples comparing the three methodologies with RANS and DNS values of u_y are shown in figure 7.34.

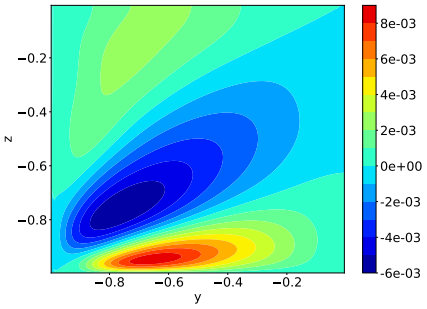


(a) DNS

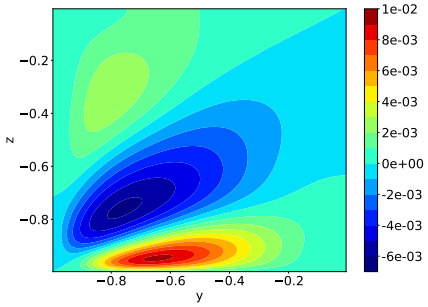
(b) Corrected by $\mathbf{R}_{\mathrm{NN}}$



(c) Corrected by $\mathbf{t}_{\mathrm{NN}}$

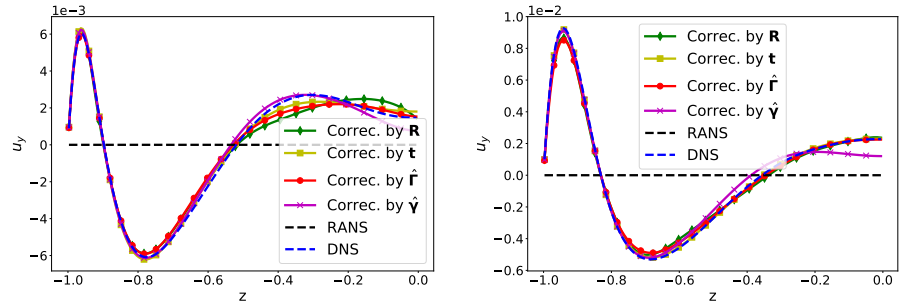


(d) Corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$



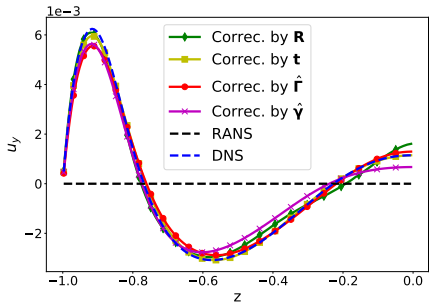
(e) Corrected by $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$

Figure 7.33: u_y corrected by the four methodologies

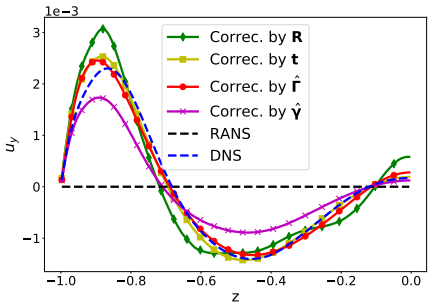


(a) $y = -0.8$

(b) $y = -0.6$



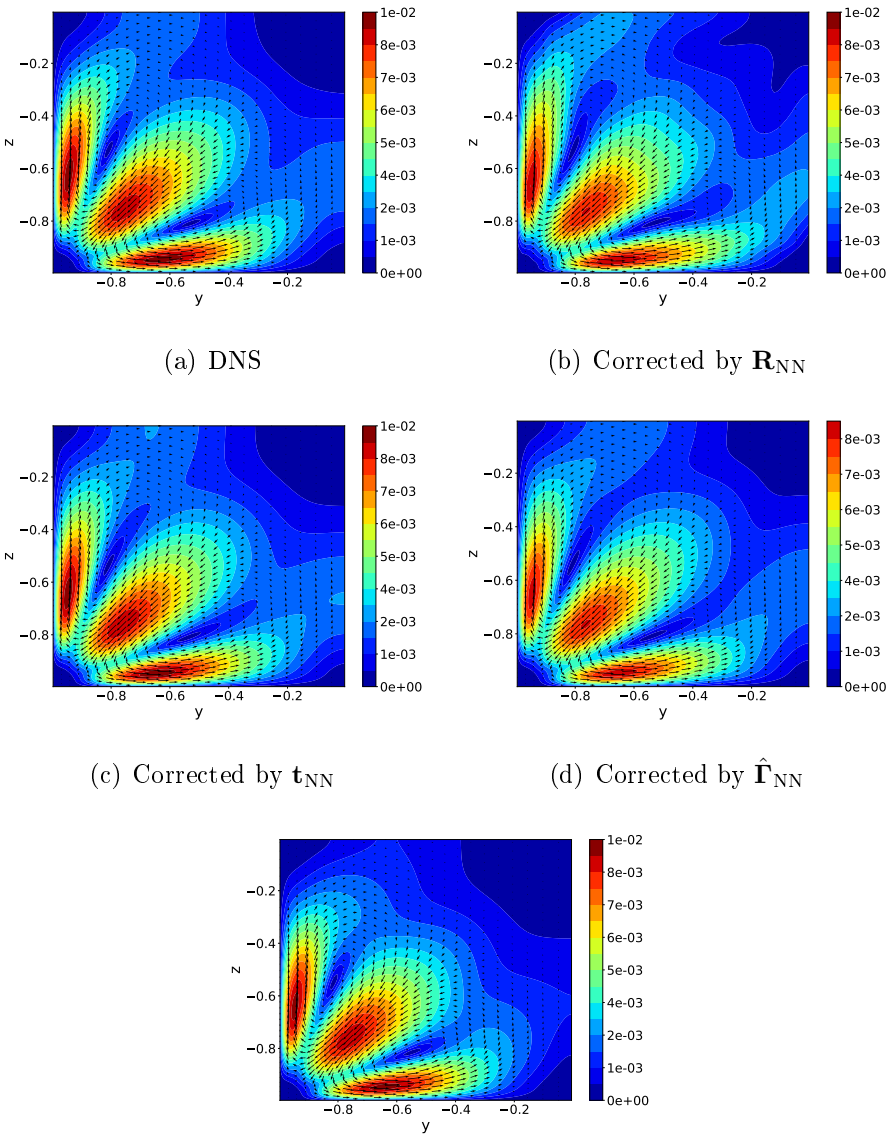
(c) $y = -0.4$



(d) $y = -0.2$

Figure 7.34: Corrected u_y samples

In general, the corrections that are closer to DNS values on most of the computational domain, as can observed in figure 7.34, are the $\mathbf{t}$ and $\hat{\mathbf{\Gamma}}$ methodologies. The magnitude of the recirculations corrected by all four approaches are depicted in figure 7.35.



(a) DNS

(b) Corrected by $\mathbf{R}_{\mathrm{NN}}$

(c) Corrected by $\mathbf{t}_{\mathrm{NN}}$

(d) Corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$

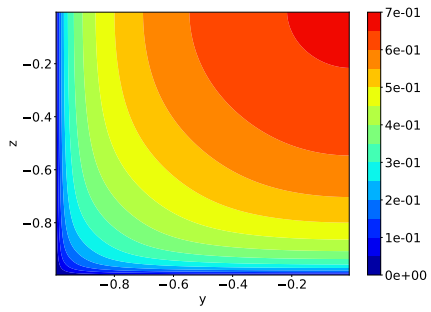
(e) Corrected by $\hat{\gamma}_{\mathrm{NN}}$

Figure 7.35: Recirculation magnitude given by the three methodologies

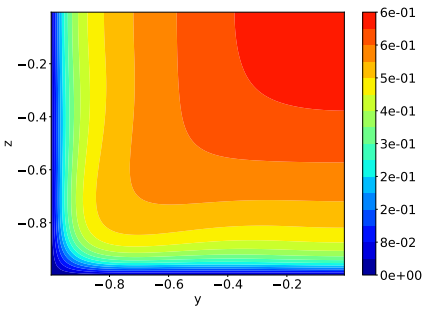
Figure 7.35 demonstrates that the discrepancy between alternative methodologies can only be noted by small differences in the recirculation shapes, especially near the duct's center. Once again, figure 7.35 demonstrates that $\mathbf{t}$ and $\hat{\mathbf{\Gamma}}$ corrections are closer to DNS than the $\mathbf{R}$ and $\hat{\gamma}$ corrections.

7.2.3 Primary Flow

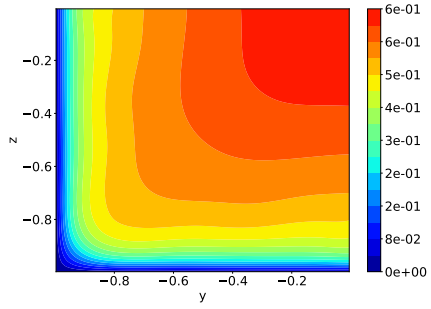
Corrected primary flow results are presented in figure 7.36 with comparison to the baseline RANS and the DNS u_x. All four approaches efficiently correct the primary flow. Most notably, $\mathbf{R}$ and $\hat{\gamma}$ corrections performances are poorer than the other two.



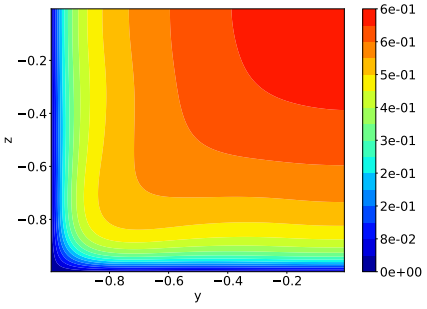
(a) RANS



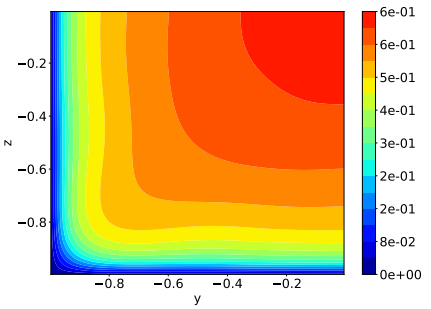
(b) DNS



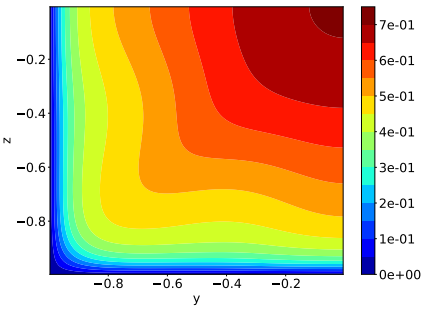
(c) Corrected by $\mathbf{R}_{\mathrm{NN}}$



(d) Corrected by $\mathbf{t}_{\mathrm{NN}}$

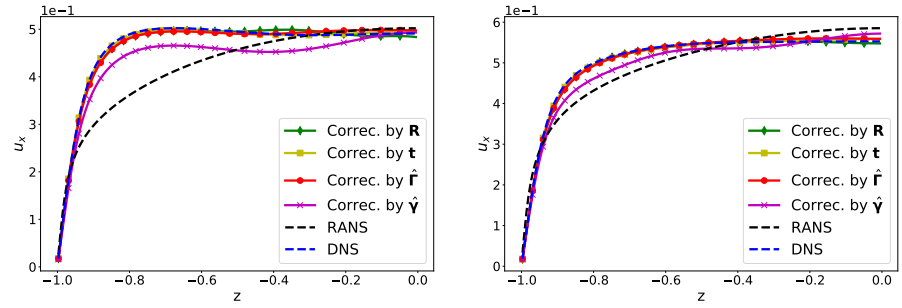


(e) Corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$

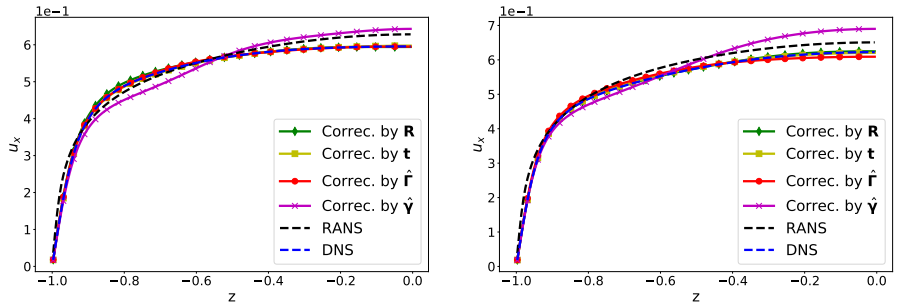


(f) Corrected by $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$

Figure 7.36: u_x corrected by the three methodologies



(a) $y = -0.8$ (b) $y = -0.6$



(c) $y = -0.4$ (d) $y = -0.2$

Figure 7.37: Corrected u_x samples

Although an improved **R** database is employed, figure 7.36 (c) demonstrates that its corrections still are not as close to DNS as the **t** corrections. $\hat{\mathbf{\Gamma}}_{\text{NN}}$ corrected u_x is consistently close to the DNS field across the whole duct's domain. Corrections by $\hat{\boldsymbol{\gamma}}$ deviate substantially more than other ones, particularly in the duct's center, where figure 7.37 seem farther away from the DNS than the baseline RANS.

7.3 Global Errors

To better compare the $\hat{\mathbf{\Gamma}}$ and $\hat{\boldsymbol{\gamma}}$ methodologies with the ones previously employed in the literature, a global error metric employed by CRUZ *et al.* (2019) is used to evaluate the achieved results in face of their respective upper boundaries. The global error formulas are given in equations 4.4a and 4.4b, they consist on an area averaging of the absolute local errors, normalized by the bulk velocity U_{bulk}.

Figure 7.38 shows the global error comparison for the component y of the velocity field. Although the upper performance boundary of $\hat{\boldsymbol{\gamma}}$ is comparable to **t** its correction on the secondary flow had the worse performance among all evaluated methodologies. This is probably due to the nature of the $\hat{\boldsymbol{\gamma}}_{\text{DNS}}$ used in training, as evidenced by figures 7.27 and 7.29.

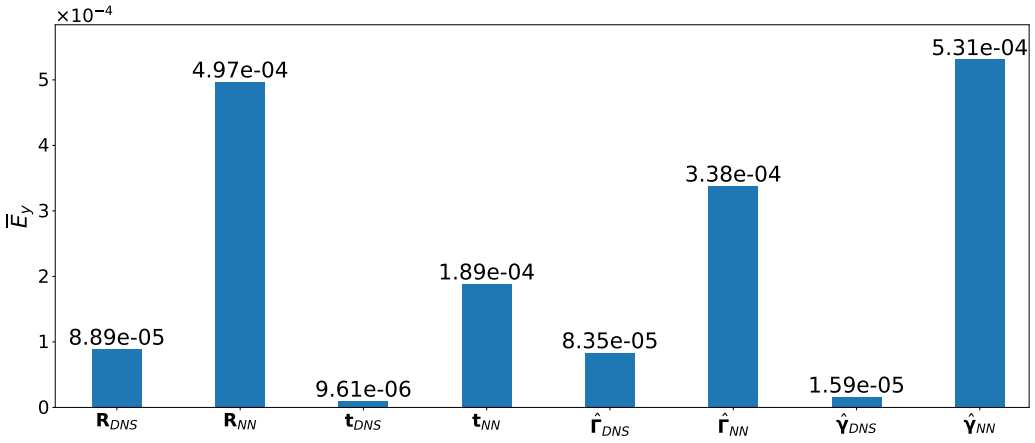


Figure 7.38: Global error on recirculation $\overline{E_y}$

The better performance of $\hat{\mathbf{\Gamma}}_{\mathrm{DNS}}$ and $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$ in comparison with $\mathbf{R}_{\mathrm{DNS}}$ and $\mathbf{R}_{\mathrm{NN}}$ can be explained by the fact that $\hat{\mathbf{\Gamma}}$ is calculated using both velocity field's data and $\mathbf{R}$ data. Therefore, using $\hat{\mathbf{\Gamma}}$ can be considered as an inverse problem, where the velocity and the Reynolds stress fields are used in conjunction to improve the Reynolds stress itself, through the use of the proposed RSTE 6.3a and its source term $\hat{\mathbf{\Gamma}}$.

The primary flow's global errors are shown in figure 7.39. With respect to the primary flow, the $\hat{\boldsymbol{\gamma}}$ corrections' defficiencies are even more considerable. The other three procedures performances are reasonably similar, with the $\mathbf{t}$ methodology achieving the most accurate results.

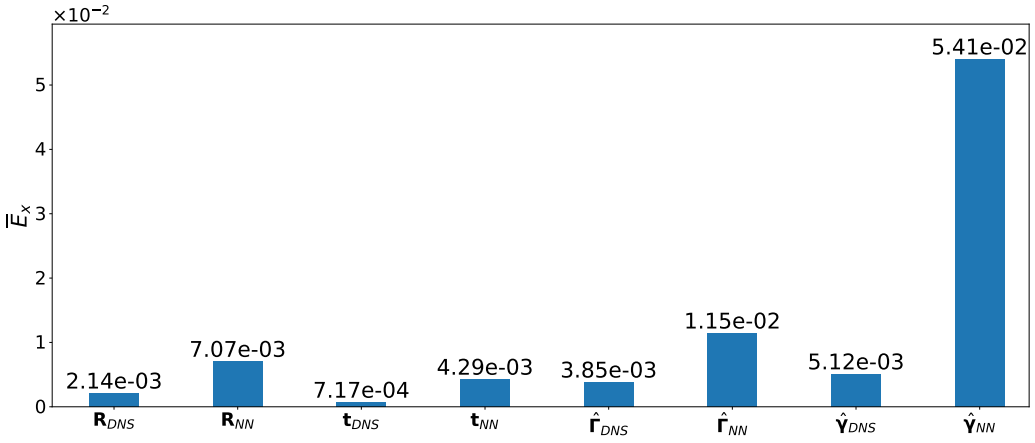


Figure 7.39: Global error on main flow $\overline{E_x}$

Chapter 8

Conclusion

Two new data-driven turbulence models based on transport equations for the Reynolds stress and the Reynolds Force Vector were proposed. The objective of both models was to correct RANS simulations through the use of machine learning predictions. Following, the proposed methodologies were evaluated with the use of Neural Networks as the ML scheme to provide the source terms of the RSTE $\hat{\mathbf{\Gamma}}$ and the modified RFVTE $\hat{\boldsymbol{\gamma}}$. These terms were resposible for correcting the RANS simulated square-duct flow.

The proposed NN inputs led to good predictions by the networks on the five targeted quantities, namingly $\mathbf{R}$, $\mathbf{t}$, $\hat{\mathbf{\Gamma}}$, $\hat{\boldsymbol{\gamma}}$ and $\mathbf{t}_{\text{wall}}$. Also, the applied input selection criteria proved itself efficient in reducing the number of inputs while increasing the network's performance on all of the five sets of networks.

The novel methodologies did succeed in correcting the RANS simulations associated turbulent quantities $\mathbf{R}$ and $\mathbf{t}$, as well as the velocity fields $\mathbf{u}$. As of today, this is the first effort to correct not only the velocity field but also the Reynolds stress and its divergence.

The tensor $\hat{\mathbf{\Gamma}}$ corrections achieved better results on the secondary flow than the most usual strategy, which uses the tensor $\mathbf{R}$ as the ML target. However, its recirculation results were not as accurate as the $\mathbf{t}$ methodology by CRUZ *et al.* (2019). On the primary flow, the $\hat{\mathbf{\Gamma}}$ correction's accuracy was lower than both $\mathbf{R}$ and $\mathbf{t}$ corrections.

The vector $\hat{\boldsymbol{\gamma}}$ performance, on both primary and secondary flows, was the poorer of all compared methodologies. This is in contrast with the expectation that the use of $\hat{\boldsymbol{\gamma}}$, which only requires velocity data to be computed, would perform better than $\hat{\mathbf{\Gamma}}$ and $\mathbf{R}$.

Further investigations on both methodologies need to be carried on different setups and problems. The use of different datasets, especially ones with a higher density of points in the numerical mesh, could lead to substantially different results.

8.1 Future Works

To better evaluate the efficiency of the methodologies in face of the other alternatives, and also the effects of using scarce meshes, both methods should be applied to datasets with higher density meshes. Future works should also address other geometries of interest in the machine learning and turbulence contexts, like the periodic-hill problem, and other flows that constitute a challenge to turbulence modeling.

Possible future research also includes coupling the present methodology with other sources of high-fidelity data, like LES simulations. The use of alternate ML techniques to predict the source terms $\hat{\mathbf{\Gamma}}$ and $\hat{\gamma}$ can also be an interesting extension of the present work.

The transport of the Reynolds stress tensor or of the Reynolds force vector have a source term that needs to be modelled. An alternative approach would be to treat the source term as an stochastic variable and the PDE associated with this term as a physically informed approach to propagate the uncertainty of this quantity to other quantities of interest, such as the velocity field.

References

BISHOP, C. M., 2006, *Pattern recognition and machine learning*. Springer.

BOUSSINESQ, J., 1877, *Essai sur la théorie des eaux courantes*. Impr. nationale.

CHOLLET, F., OTHERS, 2015. "Keras". `https://keras.io`.

CINNELLA, P., 2016, "Review of Uncertainty Quantification in Turbulence Modelling to Date", *SIAM Uncertainty Quantification Conference*, (04). doi: 10.13140/RG.2.1.4512.5523.

CRUZ, M. A., THOMPSON, R. L., SAMPAIO, L. E., et al., 2019, "The use of the Reynolds force vector in a physics informed machine learning approach for predictive turbulence modeling", *Computers & Fluids*, v. 192, pp. 104258.

DAVIDSON, P., 2015, *Turbulence: an introduction for scientists and engineers*. Oxford University Press.

DURAISAMY, K., IACCARINO, G., XIAO, H., 2019, "Turbulence Modeling in the Age of Data", v. 51, n. 1, pp. 357–377. ISSN: 0066-4189, 1545-4479. doi: 10.1146/annurev-fluid-010518-040547. Disponível em: <`https://www.annualreviews.org/doi/10.1146/annurev-fluid-010518-040547`>.

HAYKIN, S., 2009, *Neural networks and learning machines*. New York: Prentice Hall.

JONES, W., LAUNDER, B. E., 1972, "The prediction of laminarization with a two-equation model of turbulence", *International journal of heat and mass transfer*, v. 15, n. 2, pp. 301–314.

KINGMA, D. P., BA, J., 2014, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*.

KOLMOGOROV, A., 1941, "The local structure of isotropic turbulence in an incompressible viscous fluid". In: *Dokl. Akad. Nauk SSSR*, v. 30, pp. 301–305.

KOTSIANTIS, S. B., ZAHARAKIS, I. D., PINTELAS, P. E., 2006, "Machine learning: a review of classification and combining techniques", *Artificial Intelligence Review*, v. 26, n. 3, pp. 159–190.

KUNDU, P. K., COHEN, I. M., DOWLING, D. R., 2016, *Fluid Mechanics*. Academic Press.

LAUNDER, B. E., SHARMA, B., 1974, "Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc", *Letters in heat and mass transfer*, v. 1, n. 2, pp. 131–137.

LAUNDER, B. E., SPALDING, D. B., 1974, "The numerical computation of turbulent flows", *Computer methods in applied mechanics and engineering*, pp. 269–289.

LING, J., TEMPLETON, J., 2015, "Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty", *Physics of Fluids*, v. 27, n. 8, pp. 085103.

LING, J., KURZAWSKI, A., TEMPLETON, J., 2016, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance", *Journal of Fluid Mechanics*, v. 807, pp. 155–166.

MARSLAND, S., 2011, *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC.

MCCULLOCH, W. S., PITTS, W., 1943, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, v. 5, n. 4, pp. 115–133.

MOHAMMADI, B., PIRONNEAU, O., 1993, *Analysis of the k-epsilon turbulence model*. Editions Masson.

NIECKELE, A. O., THOMPSON, R. L., MOMPEAN, G., 2016, "Anisotropic Reynolds stress tensor representation in shear flows using DNS and experimental data", *Journal of Turbulence*, v. 17, n. 6, pp. 602–632.

PATANKAR, S. V., SPALDING, D. B., 1972, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows", *International Journal of Heat and Mass Transfer*, v. 15.

PINELLI, A., UHLMANN, M., SEKIMOTO, A., et al., 2010, "Reynolds number dependence of mean flow structure in square duct turbulence", *Journal of fluid mechanics*, v. 644, pp. 107–122.

RANGEL, V. B., 2019, *Influência do Tratamento da Base de Dados DNS na Aplicação de Técninas de Aprendizagem de Máquina para Melhorar Acurácia de Simulações RANS*. Tese de Mestrado, Universidade Federal do Rio de Janeiro.

RICHARDSON, L. F., 1922, *Weather prediction by numerical process*. Cambridge University Press.

ROSENBLATT, F., 1958, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, v. 65, n. 6, pp. 386.

ROSENBLATT, F., 1961, *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Relatório técnico, Cornell Aeronautical Lab Inc Buffalo NY.

TENNEKES, H., LUMLEY, J. L., 1972, *A first course in turbulence*. MIT press.

THOMPSON, R. L., 2008, "Some perspectives on the dynamic history of a material element", *International Journal of Engineering Science*, v. 46, n. 3, pp. 224–249.

THOMPSON, R. L., MOMPEAN, G., THAIS, L., 2010, "A methodology to quantify the nonlinearity of the Reynolds stress tensor", *Journal of Turbulence*, , n. 11, pp. N33.

THOMPSON, R. L., SAMPAIO, L. E. B., DE BRAGANÇA ALVES, F. A., et al., 2016, "A methodology to evaluate statistical errors in DNS data of plane channel flows", *Computers & Fluids*, v. 130, pp. 1–7.

THOMPSON, R. L., MISHRA, A. A., IACCARINO, G., et al., 2019, "Eigenvector perturbation methodology for uncertainty quantification of turbulence models", *Physical Review Fluids*, v. 4, n. 4, pp. 044603.

TINOCO, E. N., BRODERSEN, O. P., KEYE, S., et al., 2018, "Summary Data from the Sixth AIAA CFD Drag Prediction Workshop: CRM Cases", *Journal of Aircraft*, v. 55, n. 4, pp. 1352–1379. doi: 10.2514/1.C034409. Disponível em: <https://doi.org/10.2514/1.C034409>.

TRACEY, B. D., DURAISAMY, K., ALONSO, J. J., 2015, "A machine learning strategy to assist turbulence model development". In: *53rd AIAA aerospace sciences meeting*, p. 1287.

VAN DYKE, M., 1982, *An album of fluid motion*. Parabolic Press Stanford.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T. E., et al., 2020, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nature Methods*, v. 17, pp. 261–272. doi: https://doi.org/10.1038/s41592-019-0686-2.

WANG, J.-X., WU, J., LING, J., et al., 2017a, "A comprehensive physics-informed machine learning framework for predictive turbulence modeling", *arXiv preprint arXiv:1701.07102*.

WANG, J.-X., WU, J.-L., XIAO, H., 2017b, "Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data", *Phys. Rev. Fluids*, v. 2 (Mar), pp. 034603. doi: 10.1103/PhysRevFluids.2.034603. Disponível em: <`https://link.aps.org/doi/10.1103/PhysRevFluids.2.034603`>.

WILCOX, D. C., 2006, *Turbulence modeling for CFD*, v. 3. DCW industries.

WU, J.-L., SUN, R., LAIZET, S., et al., 2017, "Representation of Reynolds stress Perturbations with Application in Mmachine-Learning-Assisted Turbulence Modeling", *arXiv preprint arXiv:1709.05683*.

WU, J.-L., XIAO, H., PATERSON, E., 2018, "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework", *Phys. Rev. Fluids*, v. 3 (Jul), pp. 074602. doi: 10.1103/PhysRevFluids.3.074602. Disponível em: <`https://link.aps.org/doi/10.1103/PhysRevFluids.3.074602`>.

XIAO, H., CINNELLA, P., 2019, "Quantification of model uncertainty in RANS simulations: A review", *Progress in Aerospace Sciences*.

XIAO, H., WU, J.-L., WANG, J.-X., et al., 2016, "Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach", *Journal of Computational Physics*, v. 324, pp. 115–136.

Appendix A

Translating RFVTE From Indicial to Symbolic Notation

Departing from the RFVTE in indicial notation in equation A.1

$$\underbrace{\frac{\partial u_k}{\partial x_i}\frac{\partial R_{\mathrm{ij}}}{\partial x_k}}_{\nabla^T\mathbf{u}:\nabla\mathbf{R}} + \underbrace{u_k\frac{\partial}{\partial x_k}\left(\frac{\partial R_{\mathrm{ji}}}{\partial x_i}\right)}_{\mathbf{u}\cdot\nabla(\nabla\cdot\mathbf{R})} = -\underbrace{\frac{\partial R_{\mathrm{jk}}}{\partial x_i}\frac{\partial u_i}{\partial x_k}}_{\nabla^T\mathbf{u}:\nabla\mathbf{R}} - \underbrace{\frac{\partial R_{\mathrm{ik}}}{\partial x_i}\frac{\partial u_j}{\partial x_k}}_{\nabla^T\mathbf{u}\cdot(\nabla\cdot\mathbf{R})} - \underbrace{R_{\mathrm{ki}}\frac{\partial}{\partial x_i}\left(\frac{\partial u_j}{\partial x_k}\right)}_{\mathbf{R}:\nabla(\nabla\mathbf{u})} + \underbrace{\nu\frac{\partial^2}{\partial x_k\partial x_k}\left(\frac{\partial R_{\mathrm{ij}}}{\partial x_i}\right)}_{\nu\nabla^2(\nabla\cdot\mathbf{R})} + \frac{\partial\Gamma_{\mathrm{ij}}}{\partial x_i} \tag{A.1}$$

Term by term, from left to right:

$$\nabla^T\mathbf{u}:\nabla\mathbf{R} = \frac{\partial u_p}{\partial x_q}\frac{\partial R_{\mathrm{jk}}}{\partial x_i}e_p\cdot e_ie_q\cdot e_je_k = \frac{\partial u_i}{\partial x_j}\frac{\partial R_{\mathrm{jk}}}{\partial x_i}e_k = \frac{\partial R_{\mathrm{jk}}}{\partial x_i}\frac{\partial u_i}{\partial x_k}e_j \tag{A.2a}$$

$$\mathbf{u}\cdot\nabla(\nabla\cdot\mathbf{R}) = u_k\frac{\partial}{\partial x_k}\left[\frac{\partial R_{\mathrm{ij}}}{\partial x_i}\right]e_j \tag{A.2b}$$

$$\nabla^T\mathbf{u}\cdot(\nabla\cdot\mathbf{R}) = \frac{\partial u_j}{\partial x_k}\frac{\partial R_{\mathrm{im}}}{\partial x_i}e_j(e_k\cdot e_m) = \frac{\partial u_j}{\partial x_k}\frac{\partial R_{\mathrm{im}}}{\partial x_i}\delta_{\mathrm{mk}}e_j = \frac{\partial R_{\mathrm{ik}}}{\partial x_i}\frac{\partial u_j}{\partial x_k}e_j \tag{A.2c}$$

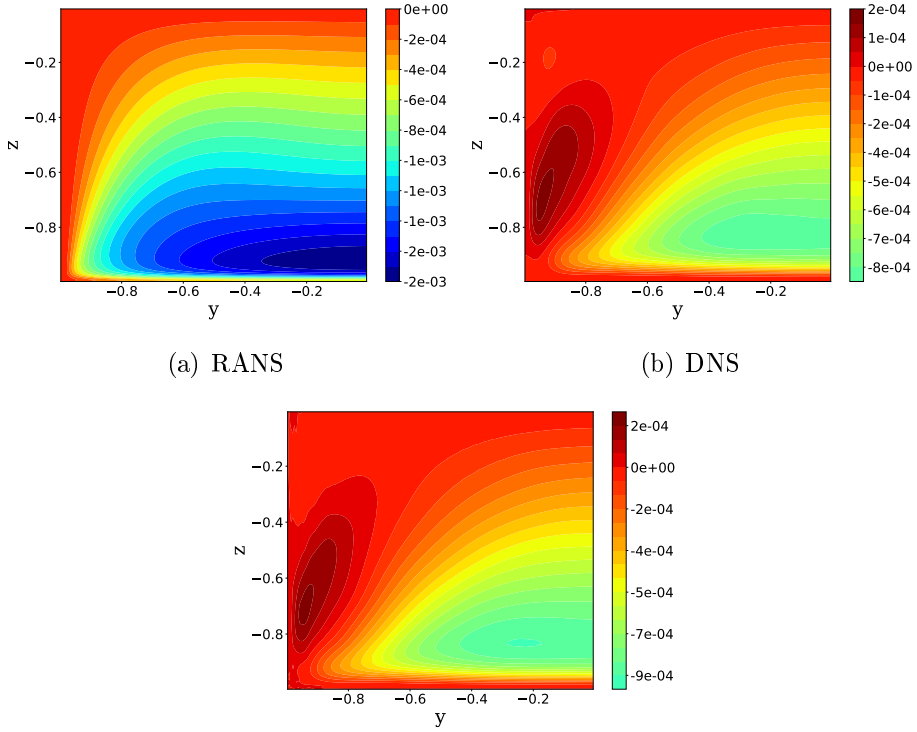
$$\mathbf{R}:\nabla(\nabla\mathbf{u}) = R_{\mathrm{ik}}e_ie_k:\frac{\partial^2u_j}{\partial x_m\partial x_n}e_me_ne_j = R_{\mathrm{ik}}\frac{\partial^2u_j}{\partial x_m\partial x_n}e_i\cdot e_me_k\cdot e_ne_j = R_{\mathrm{ki}}\frac{\partial}{\partial x_i}\left(\frac{\partial u_j}{\partial x_k}\right)e_j \tag{A.2d}$$

$$\nu\nabla^2(\nabla\cdot\mathbf{R}) = \nu\frac{\partial}{\partial x_k\partial x_k}\left[\frac{\partial R_{\mathrm{ij}}}{\partial x_i}\right]e_j \tag{A.2e}$$

Appendix B

Omitted Plots

B.1 R

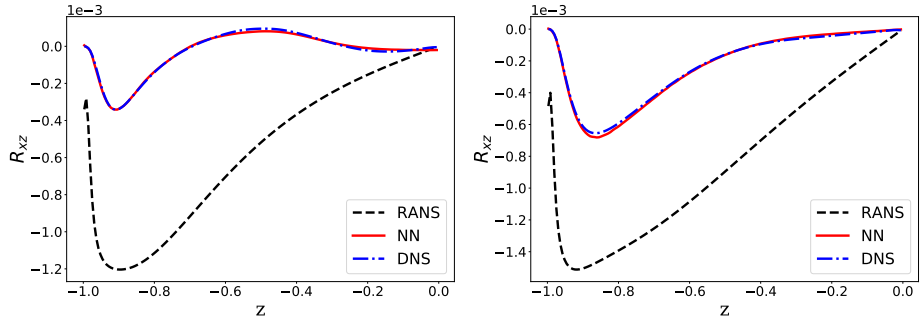


(a) RANS

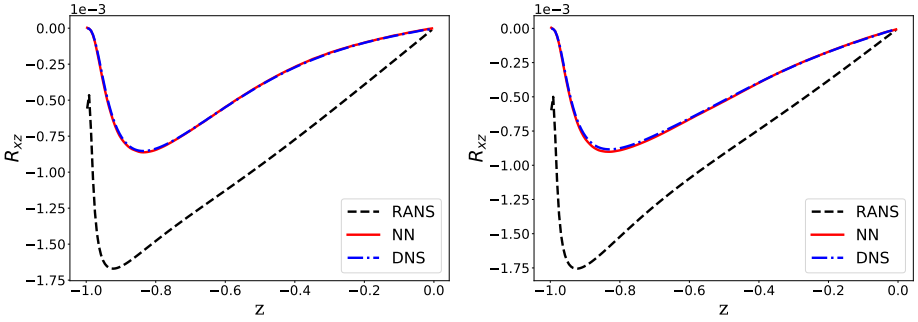
(b) DNS

(c) NN predicted

Figure B.1: RANS, DNS and predicted R_{xz} for the test case of $Re = 2900$

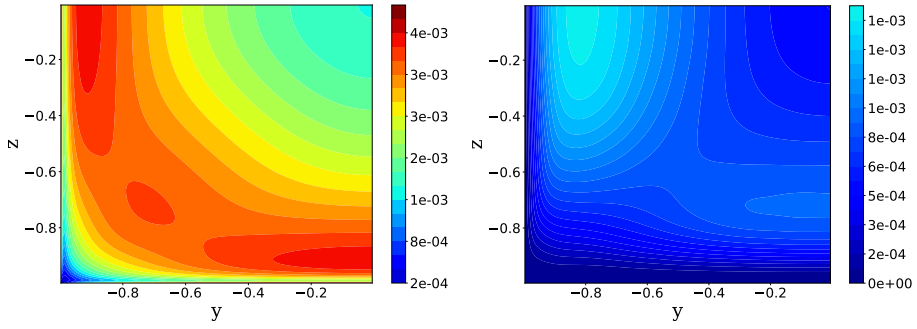


(a) $z = -0.8$ (b) $z = -0.6$

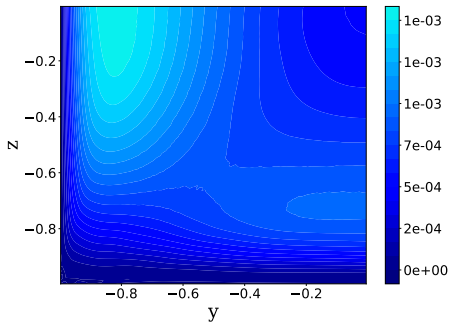


(c) $z = -0.4$ (d) $z = -0.2$

Figure B.2: R_{xz} samples

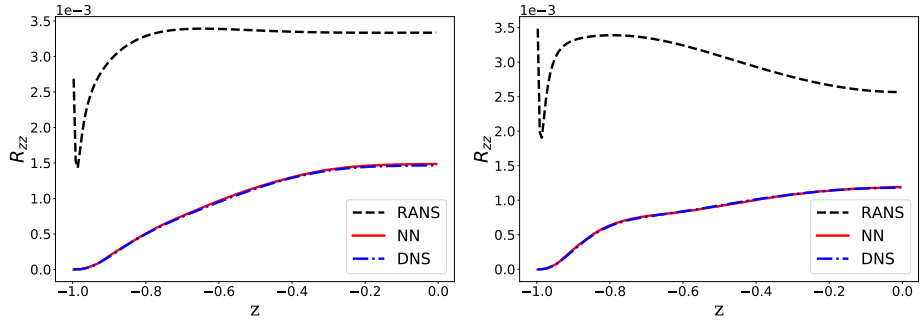


(a) RANS (b) DNS

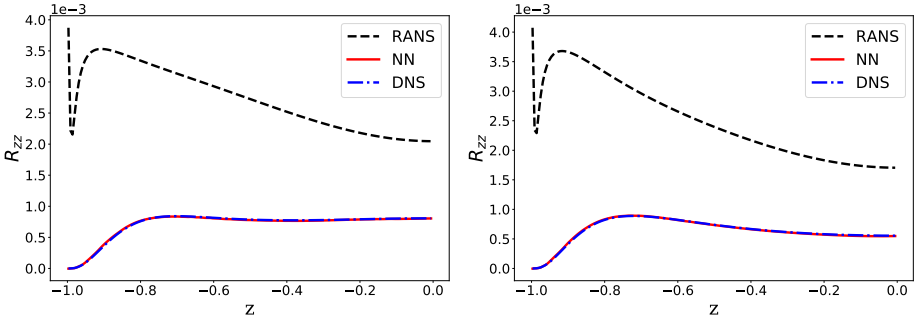


(c) NN predicted

Figure B.3: RANS, DNS and predicted R_{zz} for the test case of $Re = 2900$

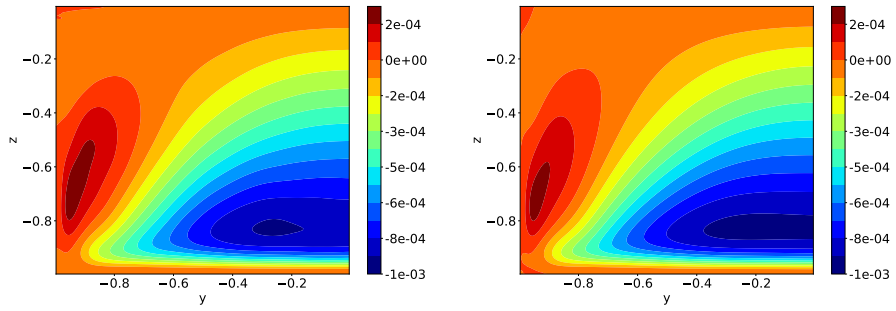


(a) $y = -0.8$ (b) $y = -0.6$

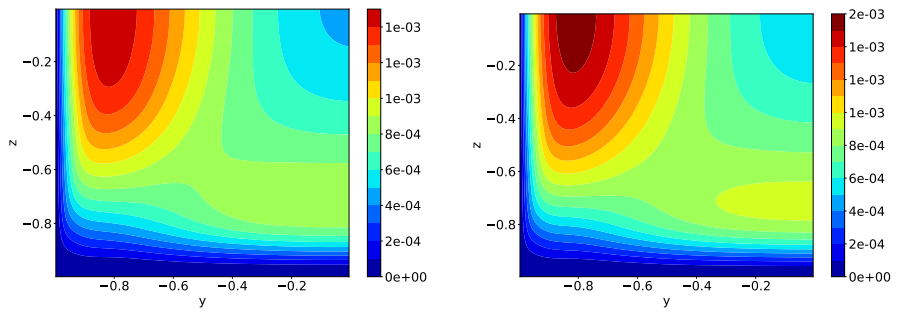


(c) $y = -0.4$ (d) $y = -0.2$

Figure B.4: R_{zz} samples



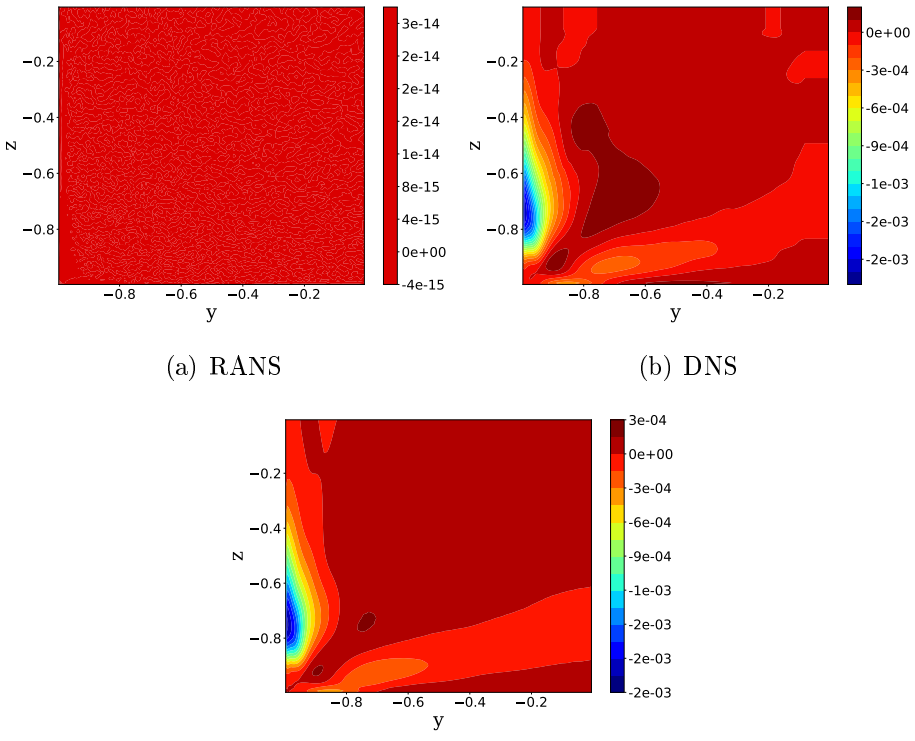
(a) R_{xz} DNS (b) R_{xz} corrected by $\hat{\Gamma}_{\mathrm{NN}}$



(c) R_{zz} DNS (d) R_{zz} corrected by $\hat{\Gamma}_{\mathrm{NN}}$

Figure B.5: Corrected $\mathbf{R}$

B.2 t

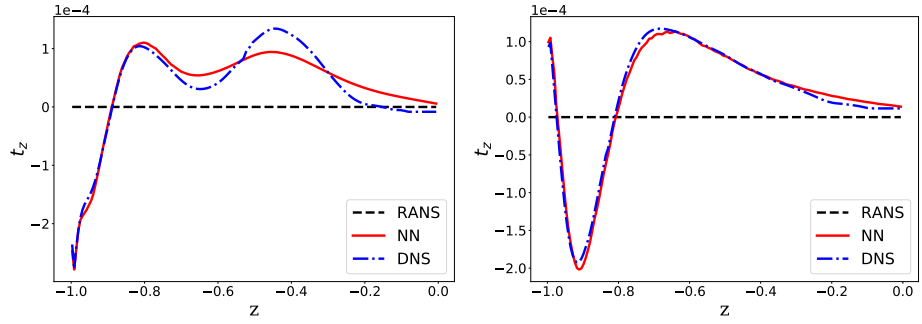


(a) RANS

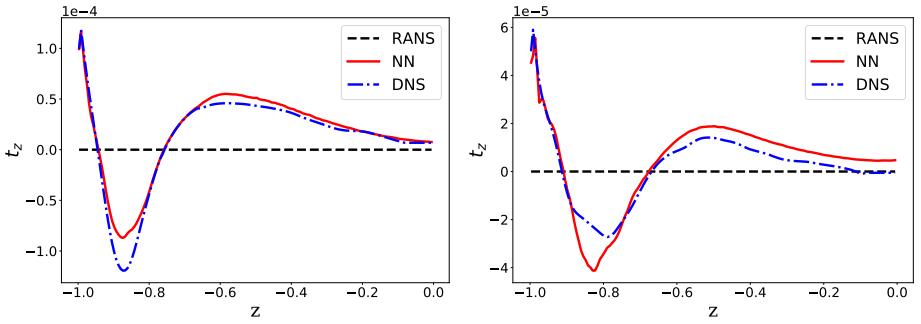
(b) DNS

(c) NN predicted

Figure B.6: RANS, DNS and predicted t_z for the test case of $Re = 2900$

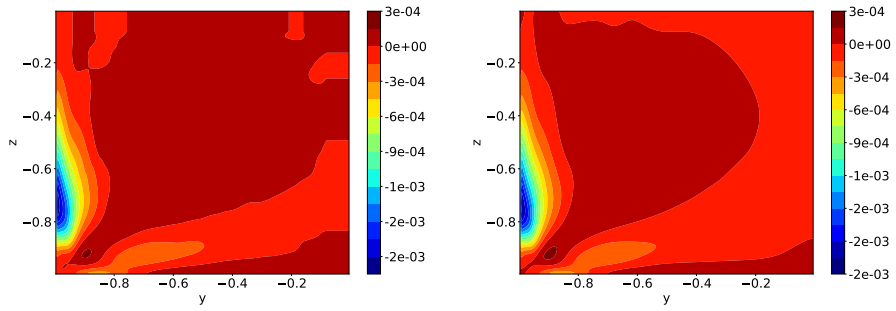


(a) $y = -0.8$ (b) $y = -0.6$



(c) $y = -0.4$ (d) $y = -0.2$

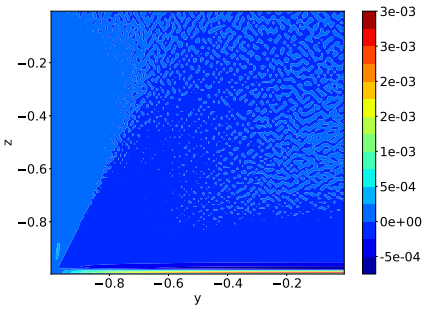
Figure B.7: t_z samples



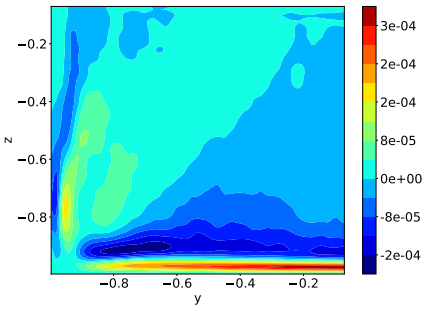
(a) DNS (b) Corrected by $\hat{\gamma}_{\mathrm{NN}}$

Figure B.8: Corrected t_z component

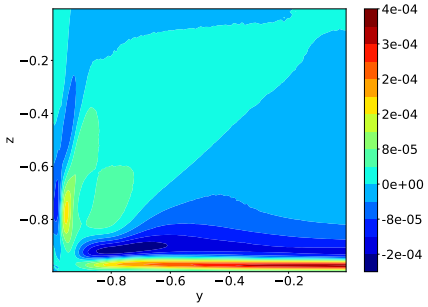
B.3 $\hat{\Gamma}$ and $\hat{\gamma}$



(a) $\hat{\Gamma}_{xz}$ RANS

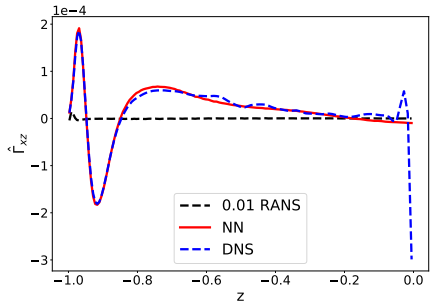


(b) $\hat{\Gamma}_{xz}$ DNS

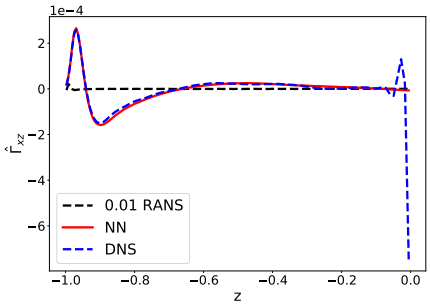


(c) $\hat{\Gamma}_{xz}$ predicted by NN

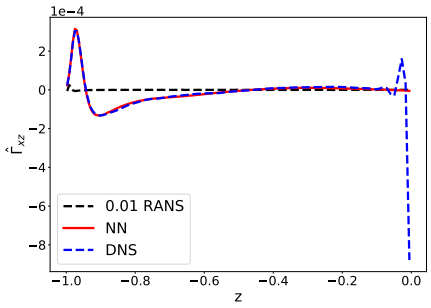
Figure B.9: RANS, DNS and predicted $\hat{\Gamma}_{xz}$ for the test case of $Re = 2900$



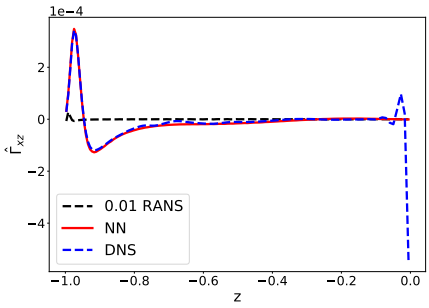
(a) $y = -0.8$



(b) $y = -0.6$

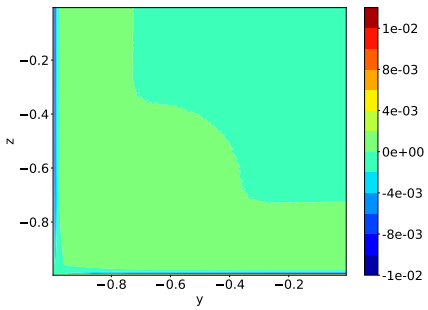


(c) $y = -0.4$

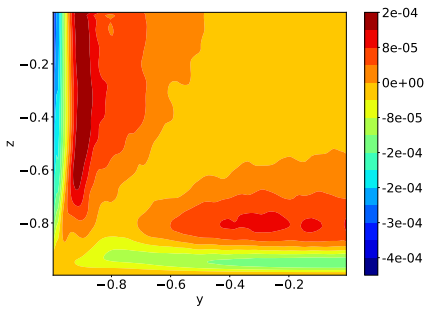


(d) $y = -0.2$

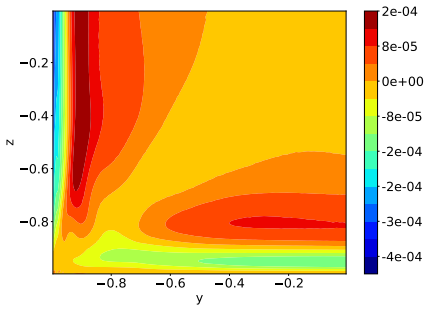
Figure B.10: Γ_{xz} samples



(a) $\hat{\Gamma}_{zz}$ RANS

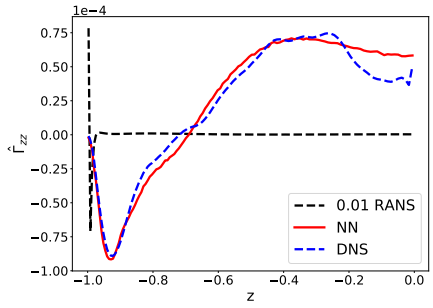


(b) $\hat{\Gamma}_{zz}$ DNS

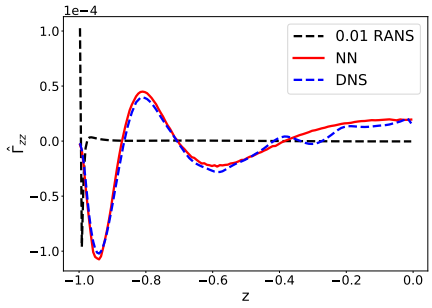


(c) $\hat{\Gamma}_{zz}$ predicted by NN

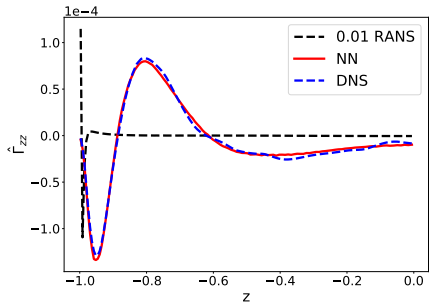
Figure B.11: RANS, DNS and predicted $\hat{\Gamma}_{zz}$ for the test case of $Re = 2900$



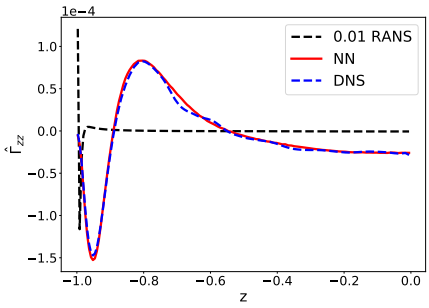
(a) $y = -0.8$



(b) $y = -0.6$

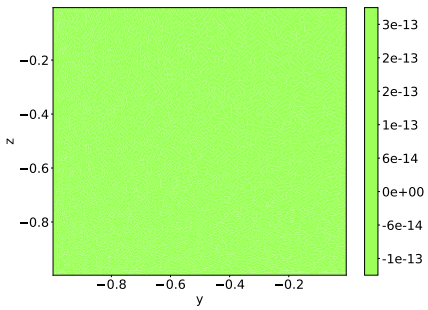


(c) $y = -0.4$

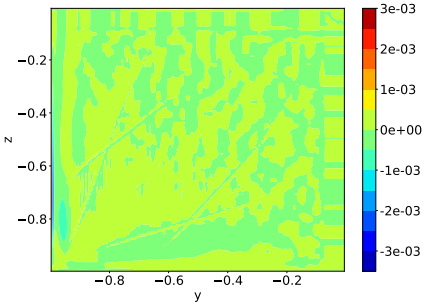


(d) $y = -0.2$

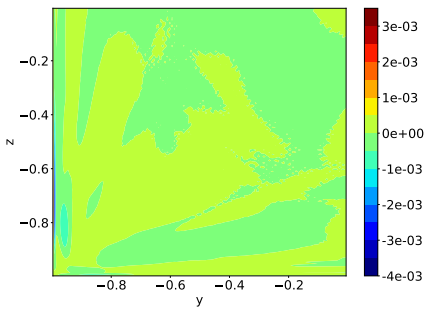
Figure B.12: Γ_{zz} samples



(a) RANS

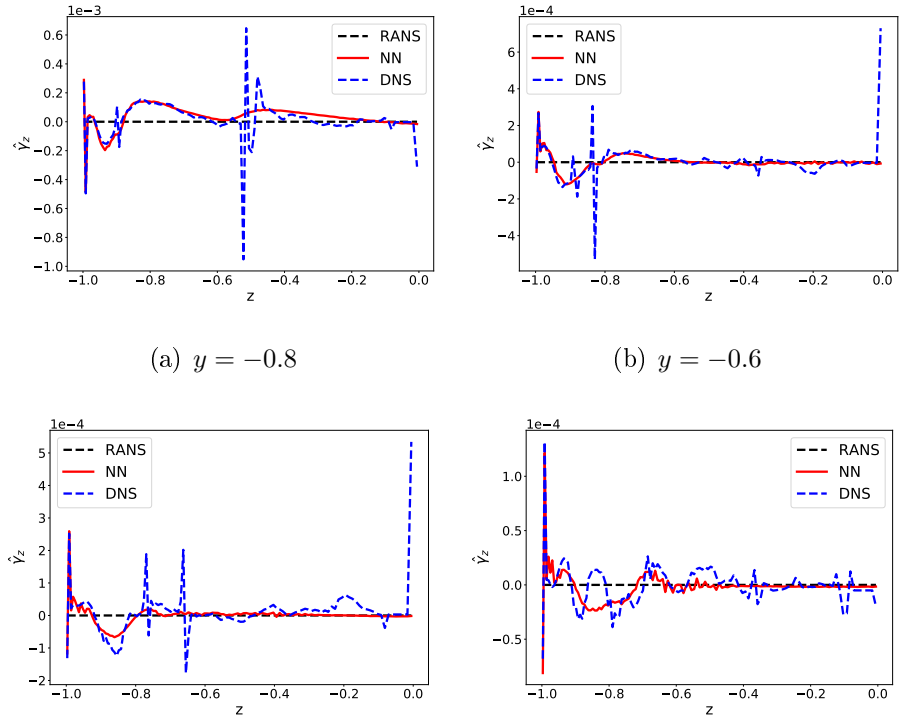


(b) DNS



(c) NN predicted

Figure B.13: RANS, DNS and predicted $\hat{\gamma}_z$ for the test case of $Re = 2900$

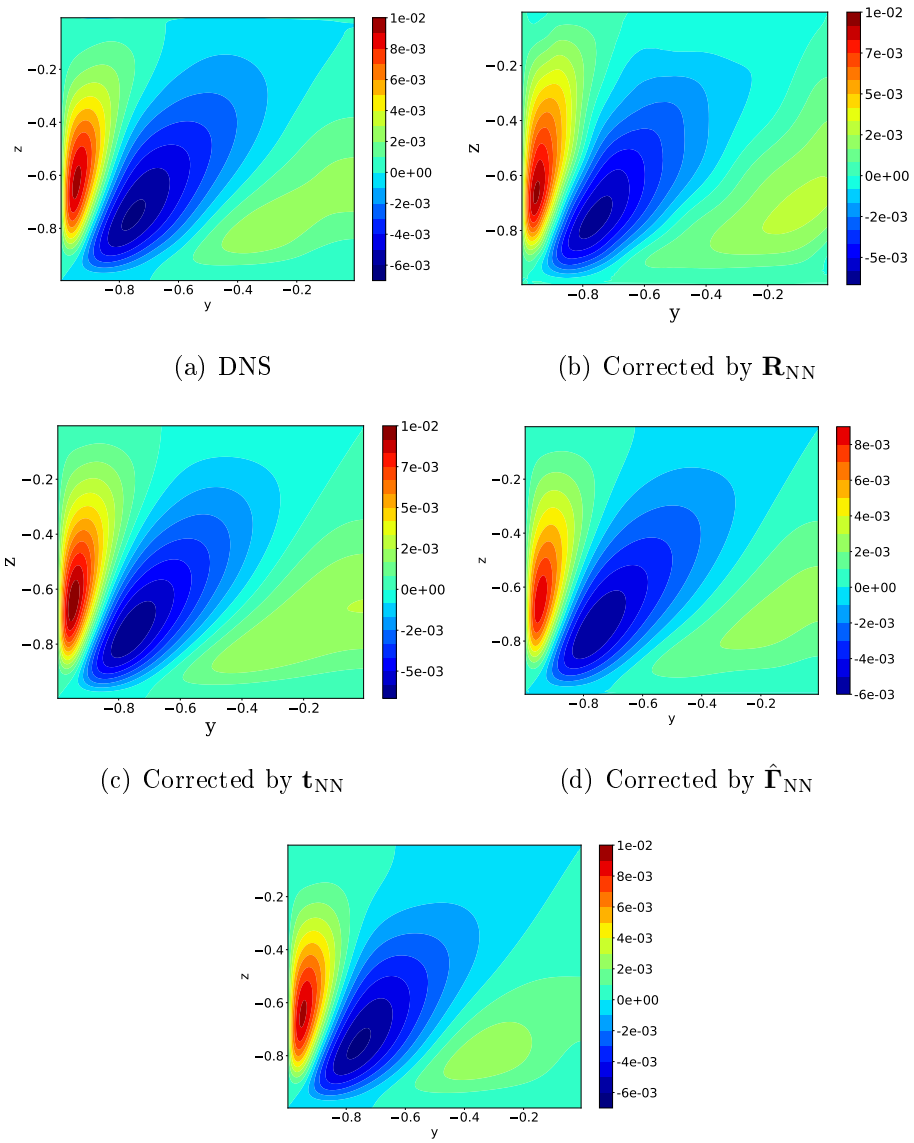


(a) $y = -0.8$ (b) $y = -0.6$

(c) $y = -0.4$ (d) $y = -0.2$

Figure B.14: $\hat{\gamma}_z$ samples

B.4 u



(a) DNS

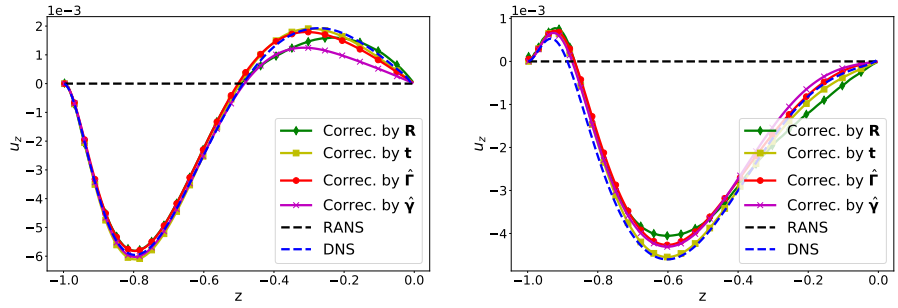
(b) Corrected by $\mathbf{R}_{\mathrm{NN}}$

(c) Corrected by $\mathbf{t}_{\mathrm{NN}}$

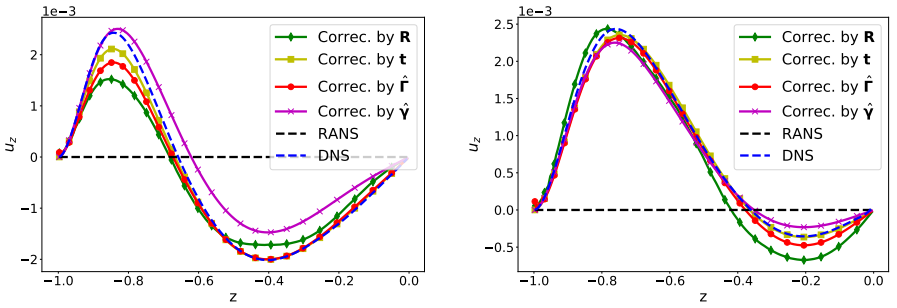
(d) Corrected by $\hat{\mathbf{\Gamma}}_{\mathrm{NN}}$

(e) Corrected by $\hat{\boldsymbol{\gamma}}_{\mathrm{NN}}$

Figure B.15: u_z corrected by the four methodologies



(a) $y = -0.8$ (b) $y = -0.6$



(c) $y = -0.4$ (d) $y = -0.2$

Figure B.16: Corrected u_z samples