POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Multi-fidelity training for data-driven thermal turbulence models

Supervisors:
Prof. Jacopo Serpieri
Dr. Matilde Fiore

Candidate:
Valerio Di Domenico

OCTOBER 2024

Abstract

In new-generation nuclear reactors, liquid metals are used to cool down the reacting core. Due to the opacity of liquid metals and their harsh operating conditions, a digital design approach based on CFD simulations is useful to study the thermal-hydraulics conditions.

In a recent study (Fiore, Koloszar, Fare, et al., 2022), a new data-driven thermal turbulence model based on Artificial Neural Networks (ANNs) was developed to improve the modelling of heavy liquid metals, characterized by very low Prandtl numbers. The model was trained with high-fidelity averaged DNS data for a wide range of Prandtl numbers (Pr=0.01-0.71) and several flows of academic interest. The model validation showed remarkable accuracy for simple flows inside and outside the training range; however, the current training database seems too limited to extend its application to simulate nuclear reactors, which are characterized by a variety of flow regimes and a wide range of Reynolds numbers.

Given the limited availability of high-fidelity data in nuclear engineering flows, a multi-fidelity modeling methodology is proposed. This approach leverages existing RANS models to provide reference conditions for scenarios where high-fidelity data is insufficient. With the use of ANNs and multi-fidelity modelling, this approach could capture the most important trends predicted by current RANS models in a wider range of flows and enrich the learned relationship with the knowledge provided by the high-fidelity data.

Contents

List of Figures

Chapter 1

Introduction

The first grid-connected nuclear power plant dates back to 1954, when, in the "Science City" of Obninsk, the APS-1 Obninsk started to produce electricity. From those early days, nuclear power has lived through varying levels of media attention and public acceptance. Nevertheless, the scientific research has gone on, exploring different design solutions to improve efficiency, sustainability and safety of nuclear reactors.
It is now standard to describe developments in terms of the following generations of technology (Nuttall, 2022):

- *Generation I*: early prototypes dating back to the 1950s and 1960s.
- *Generation II*: commercial power reactors from the 1970s and 1980s, mostly belonging to the Pressurised Water Reactor (PWR), Boiling Water Reactor (BWR) or Advanced Gas-cooled Reactor (AGR) families.
- *Generation III*: reactors matured in the late 1990s, belonging to the Advanced Light Water Reactors.
- *Generation IV*: reactors currently under research and development, with 8 clear technological goals to be achieved. Those 8 goals are concerned with the sustainability (with regards to pollutants and nuclear waste), safety, reliability, physical protection and economics of the reactors.

A critical area of such research is the design and analysis of the cooling system, which has a key role in both the power efficiency and the safety of the reactor.
One of the main elements in the cooling system is the *coolant*, meaning the fluid used to absorb the heat generated by the reactor and to generate power by either transferring it to a secondary circuit or directly moving a turbine (boiling water reactors). The common choice for nuclear reactor coolant has been, and still is, water, due to its large *availability* and *heat capacity*. Using water as the primary coolant comes with some drawbacks, such as the necessity to pressurize the reactor to raise the boiling point, thus raising safety, maintenance and technical challenges.

Another option, devised by E.Fermi and W.H. Zinn in 1944 at Los Alamos, is to use *liquid metals* as coolant. The usage of liquid metals allows better steam quality and an increase in electrical efficiency of around 30% when compared to light water reactors (Wydler, 2005). Low coolant pressure and high thermal inertia are two very important properties for liquid metals, as they act as passive safety measures. In fact, having high coolant pressure increases the chance of loss-of-cooling accidents that, mixed with low thermal inertia typical of gas/water cooled reactors, can lead to core melting. The liquid metals under consideration are sodium, lead and lead-bismuth; the Sodium Fast Reactor is the most mature of all the Generation IV technologies.

Despite their advantages, liquid metals still have some disadvantages such as the need for corrosion control systems and careful seismic design due to the higher density; however from a purely fluid dynamic point of view the main concerns are:

- Their *opacity* and *chemical composition* makes inspection and experimental campaigns more difficult, as special measurement techniques are necessary.
- They have *very low Prandtl numbers*, in the order of 10^{-2}.

This last point is a critical one for turbulence modeling and is a central matter in this project.

Before delving into the details of data-driven turbulence modeling, some fundamental notions regarding low-Pr turbulent phenomenology will be given in the following sections.

1.1 Low Prandtl number flows

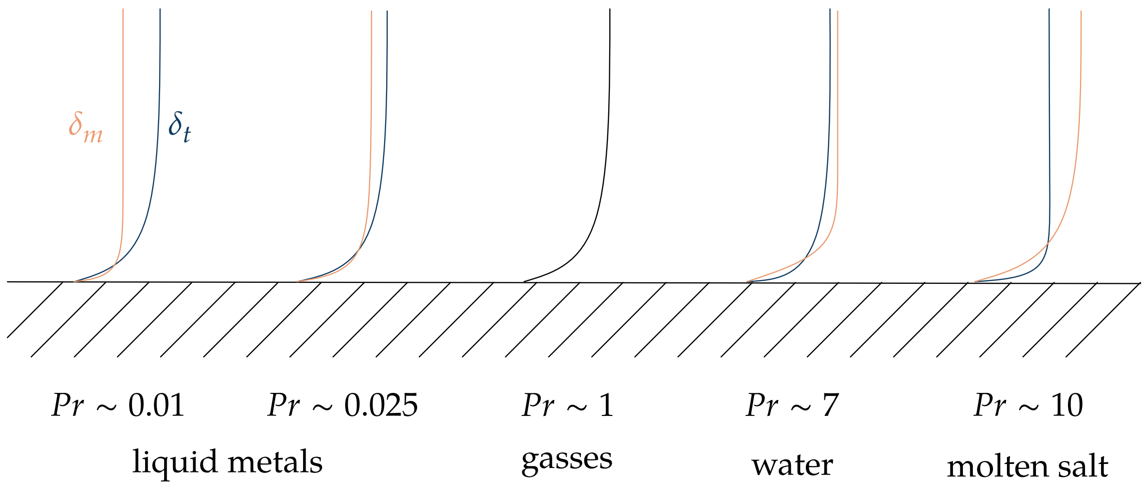


Figure 1.1: Typical momentum (δ_m) and thermal (δ_t) boundary layers in reactor applications. Reproduced from (Shams et al., 2020).

The Prandtl number Pr is a dimensionless quantity defined as the ratio of *momentum diffusivity* ν to *thermal diffusivity* α:

$$Pr = \frac{\nu}{\alpha} = \frac{c_p \mu}{k} \tag{1.1}$$

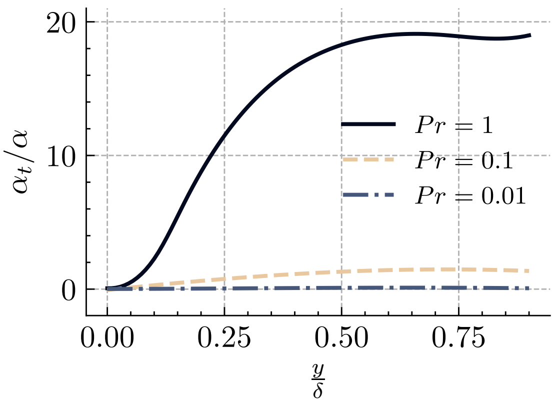


Figure 1.2: Turbulent heat diffusivity for the turbulent channel flow at $Re_\tau = 180$ for various Pr. Reproduced from (Bricteux et al., 2012).

The effect of the Prandtl number on the relationship between the thermal boundary layer δ_t and the momentum boundary layer δ_m is shown in Figure 1.1. For near unity Prandtl numbers the velocity and temperature fields are similar, with the two boundary layers having around the same thickness. Not only the velocity and temperature fields are geometrically similar, but their statistical features also are closely matched.

At lower Pr values, the molecular heat conduction is at least of the same order of magnitude than that transported by the turbulent structures and thus propagates easier into the bulk of the flow. This leads to a modification of the thermal boundary layer, that loses the similarity with the momentum boundary layer characteristic of near-unity Pr and sees a thickening of the conductive sublayer.

The *turbulent heat diffusivity* α_t can be defined as:

$$\alpha_t = \frac{-\overline{v\theta}}{d\overline{\theta}/dy} \tag{1.2}$$

where $\overline{v\theta}$ is the *turbulent heat flux*, with v representing the velocity fluctuation and θ the temperature one.

In Figure 1.2 the ratio α_t/α is computed from DNS data of a turbulent channel at $Re_\tau = 180$ for different Pr values. The figure shows that, for the lower Prandtl numbers, α_t/α decreases by an order of magnitude for the same y/δ coordinate, meaning that the heat flux is

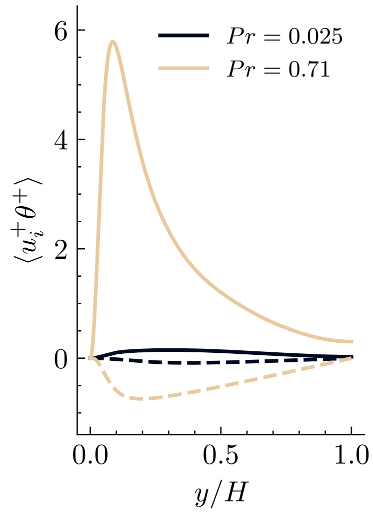


Figure 1.3: Streamwise (solid line) and wall normal (dashed line) components of the turbulent heat flux for the $Re_\tau = 180$ and different Prandtl number channel flow from (Kawamura et al., 2000).

dominated by its molecular part (Bricteux et al., 2012). A similar analysis carried out in (Grötzbach, 2013) showed how for $Pr = 0.025$ the ratio $\alpha_t/\alpha = 1$ for Reynolds numbers beyond $Re = 60000$, therefore coming to the conclusion that many applications end up in the range between conduction dominated and convection dominated turbulent heat transport.

A consequence of low Prandtl numbers is the smearing of the thermal gradients and the creation of large coherent structures, which introduce non-local effects; this leads to a significant modification of the turbulent heat flux components.

An example is shown in Figure 1.3, in which the streamwise and wall normal turbulent heat flux components are shown for two DNS of channel flows at $Re_\tau = 180\, Pr = 0.71$ and $Re_\tau = 180$, $Pr = 0.025$. In the $Pr = 0.025$ case it's clear how the difference in magnitude between the two components is much smaller than the case with $Pr = 0.71$ and they have comparable magnitude. At the higher Prandtl number, the streamwise component is clearly dominant.

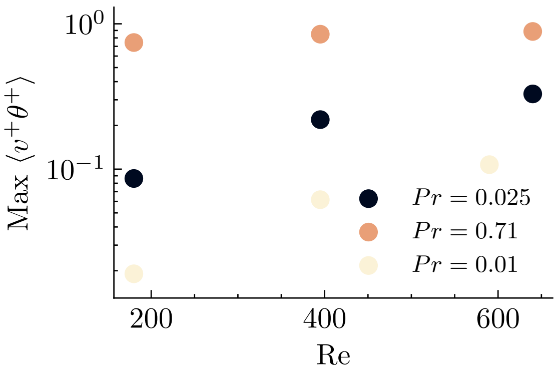


Figure 1.4: Maximum absolute value reached by the wall-normal turbulent heat flux for different flow conditions.

Another relevant feature of low Prandtl flows is the increased influence of the Reynolds number at constant Prandtl number. An example of this behavior is shown in Figure 1.4, where the maximum absolute value reached in different channel flows by the wall-normal turbulent heat flux component is shown. The figure shows how at lower Reynolds number, the peak turbulent heat flux is heavily influenced by the value of the Prandtl number; in other words the flows with lower Pr values incur in greater variations when the Reynolds number is lowered than their higher Pr counterpart. Conversely, when the Reynolds number is increased, the maximum values show less of a difference between each other.

The effect of the interplay between the Prandtl number and the Reynolds number can also be looked at from the energy spectra point of view. As shown in Figure 1.5, at $Pr \sim 1$ the velocity fluctuations energy spectra and the temperature fluctuation energy spectra are similar, and the effect of different Reynolds numbers is to shift the high wavenumber (small scales) part of the curve. Decreasing the Prandtl number has the effect of reducing the amplitude of the temperature fluctuations and dampening the fluctuations at small scales, shifting the whole curve towards bigger scales.

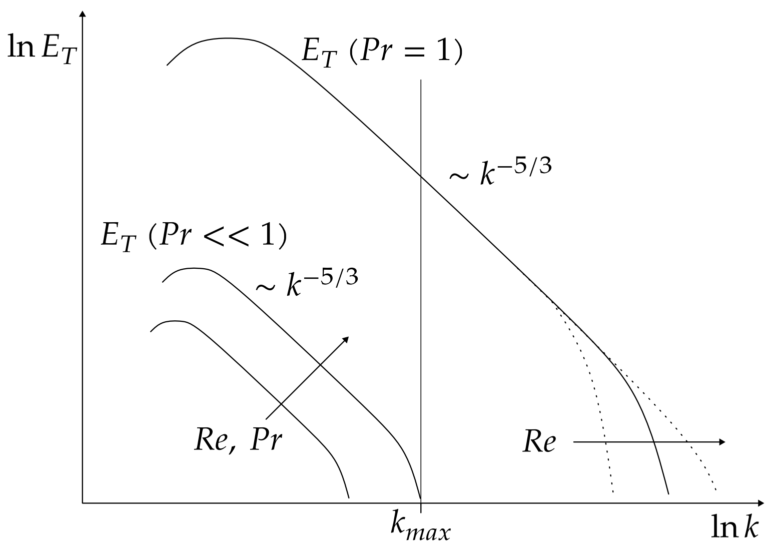


Figure 1.5: Three dimensional energy spectra $E_T(k)$ for temperature fluctuation in forced channel flows. Reproduced from (Grötzbach, 2013).

The phenomena get more complex when buoyancy influenced flows are considered, as the Prandtl number also influences the velocity field. This is particularly relevant for heavy liquid metals reactor, in which large amount of thermal energy is transported at moderate velocities. When buoyancy is not negligible, even for near unity Prandtl flows analogies between the thermal and momentum fields become unreliable.

1.2 Low Prandtl number turbulence modeling

The highlighted characteristics of low-Pr flows make their modeling extremely challenging, and a plethora of different numerical models have been put forth.
For a complete summary of the matter, the reader is referred to (Grötzbach, 2013), while here a brief description of the most relevant RANS (Reynolds Averaged Navier Stokes) modeling strategies will be given. Note that the high Reynolds numbers typical of reactor flows prevent the use of DNS (Direct Numerical Simulation) and LES (Large Eddy Simulation) approaches.
Using a RANS approach leads to the following expression for the temperature:

$$\frac{\partial T}{\partial t} + U_j \frac{\partial T}{\partial x_j} = \alpha_l \frac{\partial^2 T}{\partial x_j \partial x_j} - \frac{\partial \overline{u_j \theta}}{\partial x_j} \tag{1.3}$$

with T being the mean temperature, θ the temperature fluctuation, U the mean velocity, u the velocity fluctuation and α_l the thermal diffusivity.
The turbulent heat flux term $\overline{u_j \theta}$ needs to be modeled, and various strategies have been proposed with different hypothesis and mathematical structures. Two categories of models relevant to the current project are:

- *Eddy diffusivity models*
- *Algebraic models*

1.2.1 Eddy diffusivity models

The simplest framework is represented by *eddy diffusivity* models, that rely on the assumption of perfect alignment between the turbulent heat flux and the mean temperature gradient. This in practice means modeling the thermal turbulent diffusivity α_t as a scalar isotropic quantity:

$$\overline{u_i\theta} = -\alpha_t \frac{\partial T}{\partial x_i} \tag{1.4}$$

This structure naturally poses the question on how to model α_t.

The most direct approach is to use *zero equation models*, an example is expressing the turbulent thermal diffusivity as:

$$\alpha_t = \frac{\nu_t}{Pr_t} \tag{1.5}$$

where ν_t is the *eddy diffusivity* and Pr_t is the *turbulent Prandtl number* and takes the value $Pr_t \simeq 0.8 - 0.85$ for near unity Pr and $Pr_t \simeq 2$ for $Pr << 1$.

This kind of modeling shows several limitations. First and foremost, as discussed previously, the similarity between the thermal and momentum turbulent fields falls apart for low Pr values, which makes a local relationship between α_t and ν_t a not so valid assumption. Also, the increased sensitivity to the Reynolds number in the case of low Prandtl values makes the definition of a constant Pr_t value acceptable in a wide range of Re impossible (Fiore, n.d.). Furthermore, the assumption of a constant Pr_t has been proven quite restrictive (Grötzbach, 2013); this led to the creation of various correlations such as (Kays, 1994).
Another relevant drawback of simple gradient diffusion hypothesis is the assumption of alignment between the turbulent heat flux and the mean temperature gradient, which is only acceptable for parallel flows where the temperature profile is not affected by the streamwise turbulent heat flux. A SGDH model was tested in (Errico & Stalio, 2014) and (Errico & Stalio, 2015), where it was shown how the model wrongly predicted the heat flux in separation and reattachment regions.

A step up in complexity is represented by *one equation models* and *two equations models*, based on the idea of adding the dependence of α_t on the thermal statistics too and not limiting it to the momentum ones only. A relevant two equations model that will be used throughout this project was proposed in (Manservisi & Menghini, 2014), in which the authors express α_t as:

$$\alpha_t = C_\theta k \tau_{l\theta} \tag{1.6}$$

where $\tau_{l\theta}$ is a function of the time scale ratio $R = \frac{\varepsilon k_\theta}{k \varepsilon_\theta}$ and takes into account the corrections in the near-wall region. Temperature fluctuations k_θ, its dissipation ε_θ and its specific dissipation $\omega_\theta = \frac{\varepsilon_\theta}{k_\theta}$ are defined as:

$$k_\theta := \frac{1}{2}T'^2, \quad \varepsilon_\theta := \frac{v}{\Pr}\left\langle\left(\frac{\partial T'}{\partial x_i}\right)\left(\frac{\partial T'}{\partial x_i}\right)\right\rangle \quad \omega_0 = \frac{\varepsilon_0}{k_\theta} \tag{1.7}$$

and the following transport equations can be written:

$$\frac{\partial k_\theta}{\partial t} + U_i\frac{\partial k_\theta}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{k\theta}}\right)\frac{\partial k_\theta}{\partial x_i}\right] + P_\theta - \epsilon_\theta \tag{1.8}$$

$$\frac{\partial \epsilon_\theta}{\partial t} + U_i\frac{\partial \epsilon_\theta}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{\epsilon\theta}}\right)\frac{\partial \epsilon_\theta}{\partial x_i}\right] + \frac{\epsilon_\theta}{k_\theta}\left(C_{p1}P_\theta - C_{p2}\epsilon_\theta\right) + \frac{\epsilon_\theta}{k}\left(C_{p2}P_k - C_{d2}\epsilon\right) \tag{1.9}$$

The two equations add model coefficients, for which the authors give some base values. Similar models that aim to improve the numerical stability and accuracy have been proposed, using $k - \omega - k_\theta - \omega_\theta$ or $k - \Omega - k_\theta - \Omega_\theta$ transport equations and can be found in (Da Vià et al., 2016).

These models are an accuracy improvement over the zero-equation models in that they offer a more complex computation of the turbulent Prandtl number and they are compatible with standard isotropic momentum turbulence model like the $k - \varepsilon$ and $-\omega$. At the same time, they still suffer from the strong assumption of alignment between the turbulent heat flux and mean temperature gradient.

1.2.2 Explicit algebraic models

A class of models that overcomes the already mentioned alignment hypothesis is the *Explicit algebraic models* class. The assumption that those use is the Generalized Gradient Diffusion Hypothesis (GGDH), which models the turbulent heat flux as:

$$\overline{u_i\theta} = -D_{ij}\overline{\frac{\partial T}{\partial x_j}} \tag{1.10}$$

where D_{ij} is a *dispersion tensor* effectively replacing the scalar α_t. Various ways of modeling the dispersion tensor are explored by different models: for example strictly speaking GGDH models relate D_{ij} linearly to the Reynolds stresses, while Higher-Order GGDH involve higher order terms.

The Daly and Harlow model (Daly & Harlow, 1970) is a GGDH model that expresses D_{ij} as:

$$D_{ij} = C'\frac{k}{\varepsilon}\overline{u_iu_j} \tag{1.11}$$

thus introducing the the Reynolds stress model in the $\overline{u\theta}$ model; this makes the Daly and Harlow model quite sensitive to the correct representation of $\overline{u_iu_j}$. The model was tested along a SGDH model in low Pr flows in (Errico & Stalio, 2015), comparing to DNS data for both the magnitude and direction of $\overline{u\theta}$. In their paper, the authors showed how the GGDH model was an improvement over the SGDH model in that it predicted the direction of the heat fluxes, but at the same time it required the tuning of the model's constant C_θ when used for low Pr flows. An example of an higher order model is (Abe & Suga, 2001).

1.3 Current availability of high-fidelity data

The challenges posed by low Prandtl number flows make the availability of high fidelity data of utmost importance, both for the validation of existing modeling strategies and for the training and validation of new data-driven methods; those same challenges also make it more difficult to gather experimental and numerical results, so data availability is understandably a concern. In light of this, part of the Horizon 2020, EU SESAME and MYRTE ("Horizon 2020 - European Commission," n.d., "MYRRHA Research and Transmutation Endeavour | MYRTE Project | Fact Sheet | H2020," n.d.) projects has been the generation of experimental and numerical reference data (Shams et al., 2019). As a result, different cases were simulated without using turbulence models; these cases represent a wide array of phenomena like wall-bounded mixed convection, forced convection, separation bubbles in backward facing steps, impinging jets and free shear layers.

Besides the mentioned European projects, more high fidelity data for low Prandtl flows is available for simple geometries. Of particular relevance for this project are the databases (Kawamura et al., 2000), (Bricteux et al., 2012), (Tiselj et al., 2001) and (Alcántara-Ávila et al., 2018); these are all DNS of channel flows with different Reynolds and Prandtl numbers. To train the machine learning models for $\overline{u\theta}$ that will be illustrated in the following chapters, some statistics related to both the velocity and the temperature field are required, such as the turbulent dissipation rate ε and the dissipation rate of thermal fluctuations ε.

Table 1.1 summarises the available channel flow data, with the relevant Re_τ and Pr values. The column "Complete?" signals whether or not all the quantities necessary for the computation of the models' inputs are made available.

Table 1.1: Summary of the relevant datasets for the current project. The column "Complete?" signals whether or not all the quantities necessary for the computation of the model's invariants are available. Also note that the Reynolds and Prandtl are given as intervals, for more exact details consult the related resources.

	Re_τ	Pr	Complete?
Bricteux et al., 2012, Tiselj et al., 2001	180	0.01	Yes
	395	0.01	Yes
	590	0.01	Yes
Kawamura et al., 2000	180	0.025 - 0.71	Yes
	395	0.025 - 0.71	Yes
	640	0.025 -0.71	Yes
	1020	0.71	No
Alcántara-Ávila et al., 2018	500-2000	0.007-0.71	No

1.4 Motivation for the current project

Given the mentioned modeling challenges related to low Prandtl flows and the availability of data, the scientific community turned its attention towards data-driven thermal turbulence models. One of those is presented in (Fiore, Koloszar, Fare, et al., 2022) and (Fiore, Koloszar, Mendez, et al., 2022), which is a data-driven, physics constrained model for the turbulent heat flux. The model showed promising results, as in its first iteration it achieved good accuracy both on training data and validation data.

On the other hand, in its initial phase, it was trained using DNS data only, which severely limits the flow conditions it trains on. The limitations caused by this restricted training dataset will be shown in subsection 2.3.4. In order to overcome those limitations, the idea is to use *multi-fidelity modeling* to train the model on a wider array of flow cases, since lower fidelity datasets could be added to the training phase, covering flow conditions and geometries not compatible with DNS computing requirements.

This document details the development of the project, and is structured as follows:

- In Chapter 2 some notions about machine learning are introduced, along with the description of the single fidelity, data-driven, turbulent heat flux model. Both the capabilities and the limitation of the single fidelity model are shown, as they are the basis on which the development of the multi-fidelity model started.
- Chapter 3 introduces the concept of multi-fidelity modeling in general and then delves more in depth into multi-fidelity modeling via composite neural networks.
- In Chapter 4 the CFD databases used in the project are analyzed and compared, to understand the effects of turbulence modeling both on physical quantities and on the machine learning model's inputs.
- In Chapter 5 the process towards the development of the multi-fidelity model is traced, showing results and comparisons with its predecessor along the way.
- Chapter 6 summarises the conclusions and future works related to the project.

Chapter 2

Machine learning for fluid dynamics

In the following chapter a brief introduction to what machine learning is and why is it useful in fluid dynamics is given. Particular attention is dedicated to neural networks and how they work, as they are the main workhorse of this project. Then a recap of the already existing data driven turbulent heat flux model is made, analyzing its potential and limiting factors, which are the starting point of the current project.

2.1 Introduction

Citing Cherkassky and Mulier, 2007, *"A learning method is an algorithm that estimates an unknown mapping between a system's inputs and outputs from the available data, namely from known samples".*

From this definition alone the central role that *data* plays in the context of machine learning is quite clear. Historically speaking, fluid mechanics has dealt and is dealing with increasing amounts of data coming from experiments and numerical models, thanks to the big strides taken by high-performance computing architectures and experimental measuring systems (Brunton et al., 2020).

At the same time, many engineering fluid flows present relevant challenges in that they are strongly non-linear, non-stationary, highly dimensional and include multi-scale physics. A relevant example of said characteristics can be found in the numerical simulation of liquid metal reactors, in which the following difficulties exist among others:

- Multiple scales at which relevant physics phenomena happen. As an example the presence of the helical wires spacers in the reactor core strongly influences the thermal gradients and, due to their geometry, require detailed CAD models and CFD meshes of length scales much smaller than the system's dimension.

- Turbulence modeling. While obviously not unique to reactor flows, turbulence modeling poses additional challenges in liquid metal reactors due to their low Prandtl number, as shown in chapter 1.

- Prediction of time dependent sloshing phenomena in case of seismic activity is crucial due to the high density of liquid metals.

These features render incredibly computationally expensive or flat out impossible to obtain statistically converged numerical results. In common engineering tasks such as design and optimization, this computational cost is often not sustainable and faster, more efficient modeling is needed.

In this context, data-driven and machine learning methods thus aim to provide an alternative to complement existing modeling techniques and achieve the desired balance between accuracy and cost, be it computational, time-related or economical.

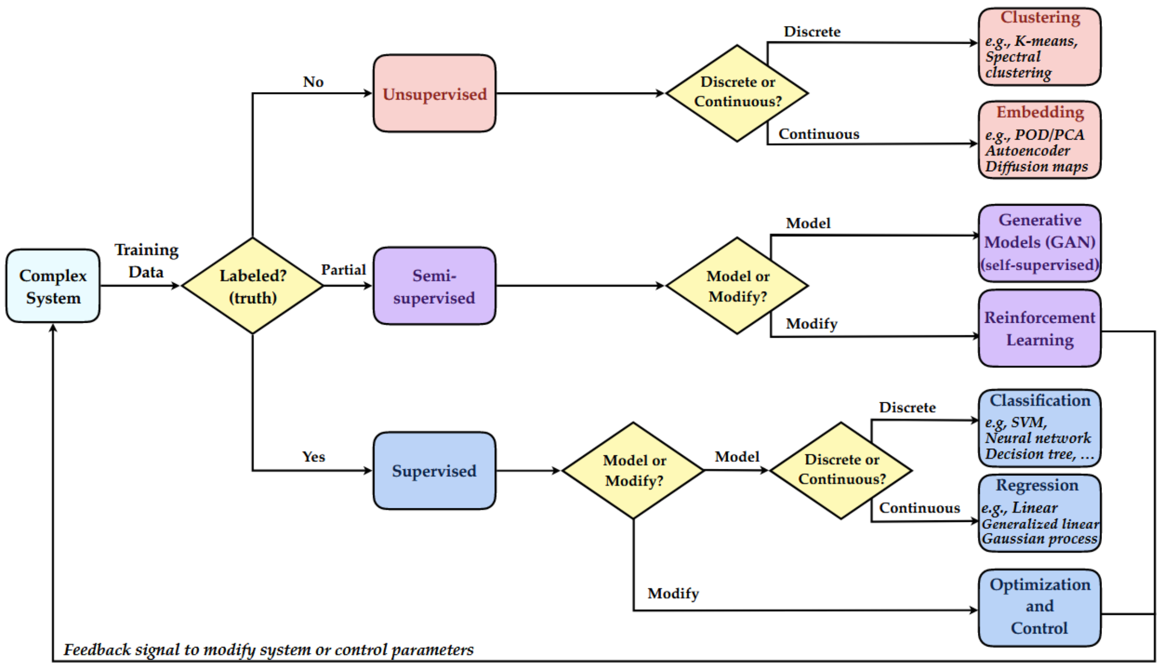


Figure 2.1: Schematic overview of various machine learning techniques. Taken from Mendez et al., 2023.

Various families of algorithms can be identified and the most common distinction is made between three main categories, based on the amount of supervisory information available to the learning machine:

- Unsupervised learning
- Semi-supervised learning
- Supervised learning

2.1.1 Unsupervised learning

In the case of *unsupervised learning*, features get extracted from the data without the need of ground truth-label for the results. In other words the learning system receives input samples without the notion of the relative output. Typical unsupervised learning

problems include dimensionality reduction, quantization and clustering.
A common technique of unsupervised learning, the Principal Component Analysis, will be used in chapter 4 to assess the difference between two complex datasets of RANS and DNS simulations, thus a short introduction is given in the following section.

Principal Component Analysis

The Principal Component Analysis (PCA) is one of the oldest techniques for multivariate analysis, firstly presented by Pearson, 1901. At its core it is a *dimensionality reduction* tool that computes a transformation into a new set of orthogonal variables (Principal Components) that are a combination of the original variables, used when the dataset is characterized by several interrelated variables. The new coordinate system's components are ordered to capture the most variance within the first components, retaining the most information about the dataset in the first few variables. A complete reference on the PCA is (Jolliffe, 2010). Next, the computation of PCA will be shown following the procedure and notation found in (Brunton & Kutz, 2019).

The first step in computing the PCA is the construction of an $n \times m$ matrix $\mathbf{X}$, in which each row represents a measurement of all the features, or a "snapshot" of the system.

The second step is the computation of the mean of all rows $\bar{\mathbf{x}}$:

$$\bar{\mathbf{x}}_j = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_{ij} \tag{2.1}$$

This in turns allows the creation of the mean matrix $\bar{\mathbf{X}}$:

$$\bar{\mathbf{X}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \bar{\mathbf{x}} \tag{2.2}$$

The mean subtracted data matrix $\mathbf{B}$ is then:

$$\mathbf{B} = \mathbf{X} - \bar{\mathbf{X}} \tag{2.3}$$

Next, the matrix $\mathbf{C}$ is computed normalizing the covariance of $\mathbf{B}$ by $n-1$.

$$\mathbf{C} = \frac{1}{n-1}\mathbf{B}^T\mathbf{B} \tag{2.4}$$

Each entry c_{ij} of this new matrix quantifies the correlation between the two measured quantities i and j across all measurements.

The *eigenvectors* of $\mathbf{C}$ are called *principal components* and define a new set of coordinates in which the covariance matrix is diagonal.

$$\mathbf{CV} = \mathbf{VD} \quad \Longrightarrow \quad \mathbf{C} = \mathbf{VDV}^T \quad \Longrightarrow \quad \mathbf{D} = \mathbf{V}^T\mathbf{CV} \tag{2.5}$$

The elements λ_k of the diagonal matrix $\mathbf{D}$ are the variances of the data along the directions constructed by the columns of the eigenvector matrix $\mathbf{V}$, which correspond to the principal components. The element $\mathbf{v_{ij}}$ are commonly referred to as *loadings*.

2.1.2 Semi-supervised learning

Semi-supervised learning algorithms use both labeled and unlabeled datasets during the training phase. In fact, in many modern applications, it is either prohibitively expensive to build a labeled data set large enough to train learning algorithms or full data about the problem is not available. Popular methods belonging to this groups are Generative adversarial networks (GANs) and Reinforcement learning (RL).

2.1.3 Supervised learning

Methods belonging to the supervised learning category imply the availability of a *labeled dataset*, that means feeding the algorithm with both the training data and its label or "true value". These labels can either be *discrete* (e.g car/bicycle), in which case the task is called *classification*, or *continuous* (e.g. the temperature at a point in space), and the task is called *regression*. During the training a loss function is built, generally evaluating some measure of error between the prediction given an input and the relative label, and then gets minimized by acting on the learning machine's parameters. In this category of methods, the most popular is probably that of *neural networks*, which play a central role in the current project and will thus be analyzed in more details in the following section.

2.2 Neural networks

The term "neural network" has been used to describe a wide range of different models (Bishop, 2006), with varying degrees of resemblance to their biological namesake. In this context, we will consider them as *nonlinear function approximator*.

The unit element in a neural network is a *neuron*, that receives an input and passes it through an activation function to produce an output. Neurons are combined and arranged in different structures tailored to the learning task. Among the most common structures is the *feed-forward networks*, composed of *layers* of neurons producing an output that gets weighted and then sent as input to the next layer. The first layer of such networks is the *input layer*, while the last layer is the *output layer* that gives the *prediction*; this prediction is then used by the cost function to compute some form of error that gets minimized by acting onto the network parameters. A more formal expression of what a neural network will be now presented, following conventions and nomenclature from (Bishop, 2006).

Considering a layer with M neurons, M linear combinations of the D input variables $x_1, x_2, ..., x_D$ are constructed:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.6}$$

where $j = 1, ..., M$ and the superscript (1) signals that the parameters w belong to the first layer of the neural network. The parameters w_{ji} are called *weights* and the

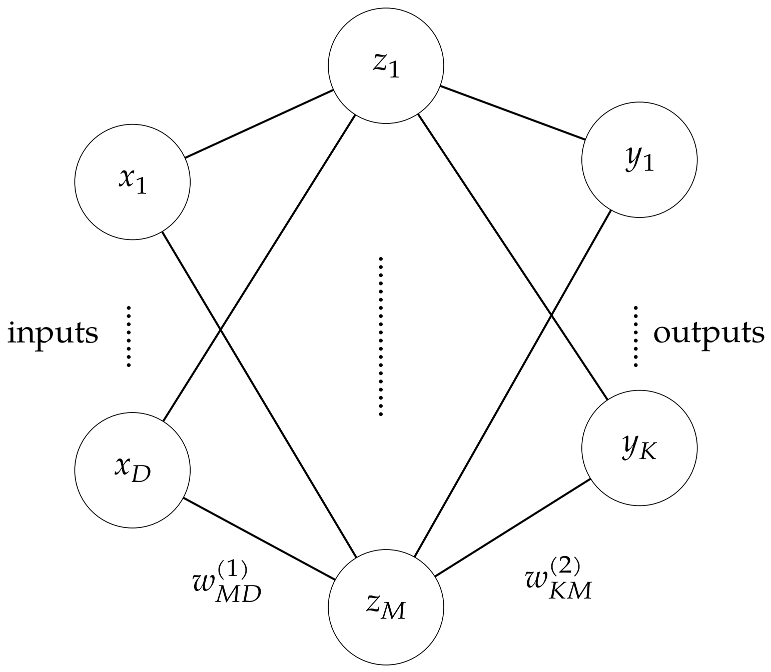


Figure 2.2: Neural network diagram, reproduced from Bishop, 2006. The variables are represented by nodes, while weights are represented by links and biases are omitted.

parameters w_{j0} are called *biases*; the quantity a_j is commonly referred to as *activation*. Each of the resulting activations undergoes a non-linear transformation represented by an *activation function* $h(\cdot)$ to obtain an *hidden unit* z_j:

$$z_j = h(a_j) \tag{2.7}$$

These hidden units are again linearly combined to give *output unit activations*:

$$a_k = \sum_{j=1}^{M} w^{(2)}_{kj} z_j + w^{(2)}_{k0} \tag{2.8}$$

where $k = 1, ..., K$ and K is the total number of outputs. A set of *outputs* y_k is given by applying an activation function to the output unit. Choices of activation function vary depending on the use case, the common ones are the ReLU, sigmoid and hyperbolic tangent functions. For regression problems, the output activation function is commonly chosen to be the identity function.

2.2.1 Neural network training

In the preceding section a neural network was built as a parametric model with parameters w_{ji} and w_{j0}, but no mention was made as to how to choose and optimize these weights and biases for a given task. The process of finding the "best" w_{ij} and w_{j0} to approximate a target t_n given the inputs x_n is called *training*. The mechanism that drives this process is the minimization of an error function $E(w)$, for this example taken to be the sum-of-squares error:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2 \tag{2.9}$$

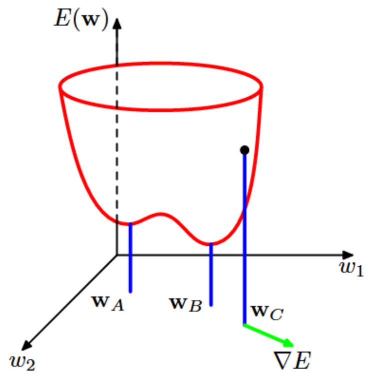


Figure 2.3: 3D space view of the error function $E(w)$ (Reproduced from Bishop, 2006). A local minimum is found in w_A and a global minimum in w_B.

In the case of a 2D input space, the error function can be represented as a 3D surface as in Figure 2.3. Note that, very much like in the figure, in common applications the error function is generally not convex in the weights' space, and thus presents both *local* and *global* minima, all of which correspond to the situation where:

$$\nabla E(\mathbf{w}) = 0 \tag{2.10}$$

Finding an analytical solution to this equation is generally not possible, and thus the usual procedure consists in using *numerical methods* of the form:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)} \tag{2.11}$$

in which an initial value $w^{(0)}$ is set and then updated at each iteration τ according to the update term $\Delta w^{(\tau)}$. An effective way to proceed is using information given by the gradient itself $\nabla E(w)$ to step into its negative direction, which is the direction of steepest descent. In general, these methods are called *gradient descent* methods, and take the form:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E\left(\mathbf{w}^{(\tau)}\right) \tag{2.12}$$

where the parameter η is referred to as *learning rate*.

In the context of the neural networks, the learning rate is one of the key *hyperparameters* that the user can act upon to influence both the training and the performance of the model. Its magnitude must be set to achieve a balance between the *stability* and the *speed* of the training. In general, choosing too large of a learning rate will result in *divergence*, meaning that the optimization is not able to minimize the loss function. Too small of a learning rate will instead result in either a very slow or ineffective training process. A simple numerical example of gradient descent is shown in Figure 2.4, where

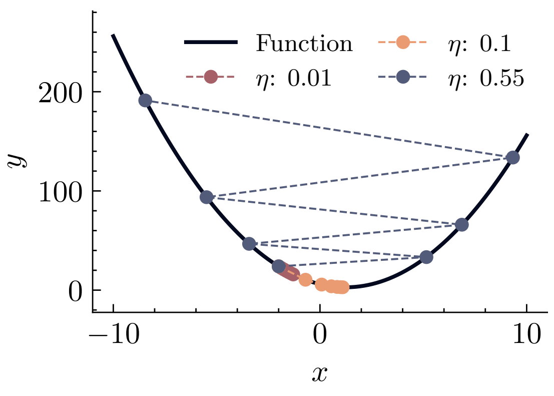


Figure 2.4: Example of gradient descent method on the function $y = 2x^2 - 5x + 6$, starting from $x = -2$.

the minimum of the function $y = 2x^2 - 5x + 6$ is to be found, starting the search from the point $x = -2$. The first 8 optimization steps of three different learning rates are shown. With the smallest step, the minimization proceeds in the right direction but does not make enough progress and stops before getting to the minimum. The intermediate step on the other hand is able to find the minimum within the 8 steps. The largest learning rate not only does not find the minimum, but diverges from the start, leading to a complete inaccurate solution.

In the numerical example, the function's gradient was easily computed symbolically; in the case of the error function $E(\mathbf{w})$ used in neural networks this would be unfeasible. The technique used to efficiently compute the derivatives is called *error backpropagation* or *backpropagation*.

2.3 Single fidelity data-driven turbulence model for the turbulent heat flux

During the 2000s, machine learning methods started to become popular in the CFD community to get over the limitations of classical turbulence models: for example in (Yarlanki et al., 2012) neural networks were used to estimate the $k - \varepsilon$ coefficients. A notable and relevant use of neural networks in turbulence model is (Ling et al., 2016), in which they are used to predict the Reynolds stresses anisotropy tensor while embedding the Galilean invariance mathematically into the network itself; the authors also introduce the concept of a *Tensor Basis Neural Network* (*TBNN*), which effectively embeds the rotational invariance by enforcing that the anisotropy tensor lies on a basis of isotropic tensors.

The starting point for the current project is the single fidelity, data-driven turbulent heat flux model developed in (Fiore, Koloszar, Fare, et al., 2022), which aims to model the dispersion tensor $\mathbf{D}$ (and thus $\overline{u_j\theta}$) using the high fidelity DNS datasets cited in the introduction as training data. For the full derivation of the model the reader is referred to the original paper, here only the main takeaways will be summarised.

2.3.1 Mathematical formulation

The derivation starts with a general functional relationship between tensors, scalars and vectors related to both the momentum and thermal field:

$$\overline{\mathbf{u}\theta} = f(\mathbf{b}, \mathbf{S}, \Omega, \nabla T, g, k, \varepsilon, k_\theta, \varepsilon_\theta, \alpha, \nu) \tag{2.13}$$

where:

$$\begin{aligned} b_{ij} &= \frac{\overline{u_i u_j}}{k} - \frac{2}{3}\delta_{ij} \\ S_{ij} &= \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \\ \Omega_{ij} &= \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right) \end{aligned} \tag{2.14}$$

The model is guaranteed to be Galilean invariant thanks to the inclusion of rotational and strain rates in Equation 2.13; it is also algebraic and explicit, meaning it can be expressed as:

$$\overline{u\theta} = -\mathbf{D}\nabla T \tag{2.15}$$

Where **D** is a *dispersion tensor.*

Adding the assumption that the anisotropic part of **D** depends on the anisotropy of the momentum field and not on the temperature distribution, and only considering forced convection flows:

$$\mathbf{D} = \mathcal{F}(\mathbf{b}, \mathbf{S}, \Omega, ||\nabla T||, k, \varepsilon, k_\theta, \varepsilon_\theta, \alpha, \nu) \tag{2.16}$$

The term **D** is modeled using a Tensor Basis Neural Network, as to embed two fundamental properties: the *rotational invariance* and the *consistency with the second law of thermodynamics*. This is achieved by expressing the dispersion tensor as:

$$\mathbf{D} = \left[\left(\mathbf{A} + \mathbf{A}^T\right)\left(\mathbf{A}^T + \mathbf{A}\right) + \frac{k}{\epsilon^{0.5}}\left(\mathbf{W} - \mathbf{W}^T\right)\right] \tag{2.17}$$

which enforces the real part of the eigenvalues of the symmetric part of **D** to be real, thus satisfying the second law of thermodynamics. The quantity $\frac{k}{\epsilon^{0.5}}$ ensures the dimensional consistency of the expression.

The tensors **A** and **W** are a combination of the *tensor basis* $\mathbf{T^i}$:

$$\mathbf{A} = \sum_{i=1}^{n} a_i \mathbf{T}^i, \quad \mathbf{W} = \sum_{i=1}^{n} w_i \mathbf{T}^i \tag{2.18}$$

The minimal tensor basis was derived and consists of the following set:

$$\mathbf{I}, \mathbf{b}, \mathbf{S}, \Omega, \mathbf{bS}, \mathbf{S}\Omega, \mathbf{b}\Omega, \mathbf{b}\Omega\mathbf{S} \tag{2.19}$$

The corresponding minimal basis of invariants is:

$$\{\mathbf{b_2}\}, \{\mathbf{b}^2\}, \{\mathbf{S}^2\}, \{\Omega^2\}, \{\mathbf{bS}\}, \{\mathbf{bS}\} \tag{2.20}$$

where $\mathbf{b_2}$ is the projection of $\mathbf{b}$ on the $x - y$ plane of the flow.

Using the Buckingham Theorem, it is possible to define 10 independent dimensionless groups, so that the coefficients in Equation 2.18 take the general form:

$$c_i = f_i(\pi_i, Re_t, Pr) \tag{2.21}$$

where the 8 π_i are:

- $\pi_1 = \frac{k^2}{\varepsilon^2}\{\mathbf{S^2}\}$
- $\pi_2 = \frac{k^2}{\varepsilon^2}\{\Omega^2\}$
- $\pi_3 = \{\mathbf{b^2}\}$
- $\pi_4 = \frac{k}{\varepsilon}\{\mathbf{bS}\}$
- $\pi_5 = \{\mathbf{b_2}\}$
- $\pi_6 = \{\mathbf{bS}\Omega\}$
- $\pi_7 = \frac{||\nabla T||}{\sqrt{k_\theta}}\frac{k^{3/2}}{\varepsilon}$
- $\pi_8 = R = \frac{k_\theta \varepsilon}{k \varepsilon_\theta}$

Note that the invariants π_i are expressed in dimensionless form so that the model can be used for arbitrary flow conditions; this means that the a_i and w_i coefficients in Equation 2.18 are dimensionless too. In turn, the tensors composing the tensors basis are multiplied and divided using k and ε to obtain a dimensional heat flux. The tensors used in the model then become:

- $\mathbf{T_1} = \frac{k}{\varepsilon^{1/2}}\mathbf{I}$
- $\mathbf{T_2} = \frac{k}{\varepsilon^{1/2}}\mathbf{b}$
- $\mathbf{T_3} = \frac{k^2}{\varepsilon^{3/2}}\mathbf{S}$
- $\mathbf{T_4} = \frac{k^2}{\varepsilon^{3/2}}\Omega$
- $\mathbf{T_5} = \frac{k^2}{\varepsilon^{3/2}}\mathbf{bS}$
- $\mathbf{T_6} = \frac{k^2}{\varepsilon^{3/2}}\mathbf{b}\Omega$
- $\mathbf{T_7} = \frac{k^3}{\varepsilon^{5/2}}\mathbf{S}\Omega$
- $\mathbf{T_8} = \frac{k^3}{\varepsilon^{5/2}}\mathbf{bS}\Omega$

2.3.2 Structure of the neural network

The neural network models the coefficient $\mathbf{Y} = [a_1, ..., a_8, w_1, ..., w_8]$ in Equation 2.18 from the input $\mathbf{X} = [\pi_1, ..., \pi_8, Re_t, Pr]$ using the weights W:

$$\mathbf{Y} = \mathcal{F}(\mathbf{X}, \mathcal{W}) \tag{2.22}$$

The inputs are normalized using minimum and maximum values in the dataset so that they fall into the $[-1, 1]$ interval; then they get fed into the ANN through two branches:

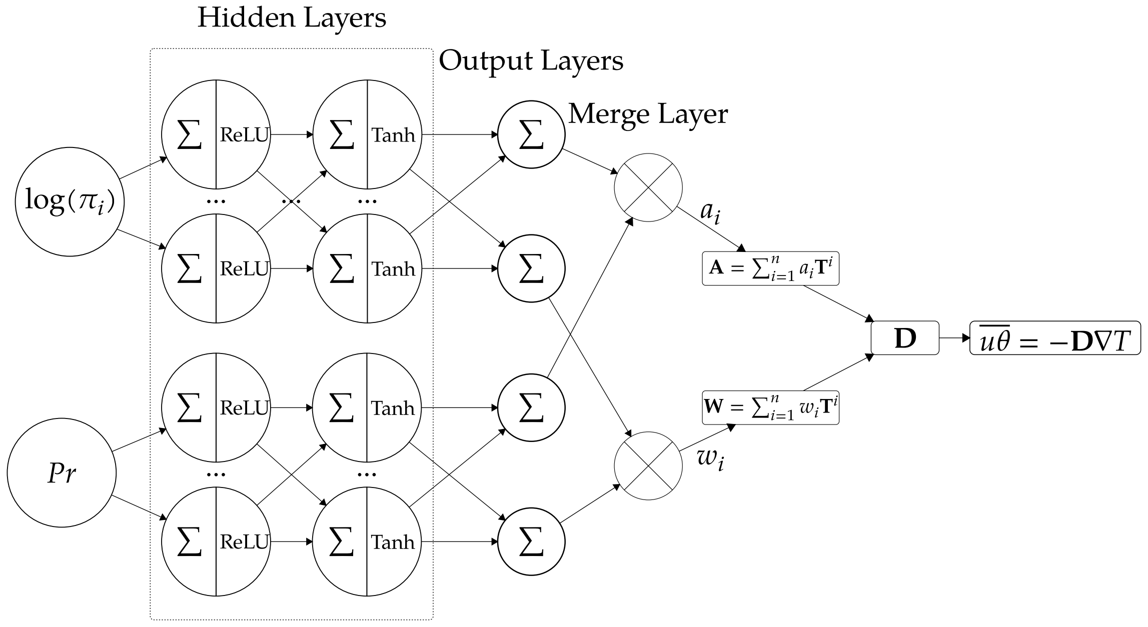

Hidden Layers
Output Layers
Merge Layer
$\log(\pi_i)$
Pr
Σ ReLU
Σ Tanh
a_i
w_i
$\mathbf{A} = \sum_{i=1}^{n} a_i \mathbf{T}^i$
$\mathbf{W} = \sum_{i=1}^{n} w_i \mathbf{T}^i$
$\mathbf{D}$
$\overline{u\theta} = -\mathbf{D}\nabla T$

Figure 2.5: Single fidelity neural network structure. The top branch has 6 hidden layers of 100 neurons each, while the bottom (Pr) branch has 2 layers of 100 neurons each. Only the last layers use the tanh activation function, while all the others use the *ReLU*.

- The first one receives all the input together and processes them through 6 layers of 100 neurons each. The first 5 layers use the *ReLU* activation function while the last layer uses the hyperbolic tangent. The inputs of this branch also pass through the function $\log(|x| + 1)$ to reduce their right-skewdness.
- The second branch only gets fed the Pr number, which goes through 2 layers of 100 neurons each; the first layer uses the *ReLU* activation function while the second uses the hyperbolic tangent.

These two branches then get multiplied by a merge layer out of which the 16 coefficients $[a_1, ..., a_8, w_1, ..., w_8]$ are computed. From there on the tensor **D** gets computed following Equation 2.18 and Equation 2.17.

2.3.3 Training process

The training process uses the Adam optimizer with a constant learning rate $\eta = 0.001$ to minimize the loss function $\mathcal{L}$, defined as:

$$\mathcal{L} = \frac{1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{3}\left(\hat{q}_{i,j} - q_{i,j}\right)^2\right) + \frac{\lambda}{N}\left(\sum_{i=1}^{N}\sum_{j,k=1}^{3}\left|\frac{\partial \hat{q}_{i,j}}{\partial x_k} - \frac{\partial q_{i,j}}{\partial x_k}\right|\Delta x_k\right) \tag{2.23}$$

In the loss function definition two terms can be identified:

- $\frac{1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{3}\left(\hat{q}_{i,j}-q_{i,j}\right)^2\right)$: this terms drives the optimization towards the minimization of the error in the turbulent heat flux prediction.
- $\frac{\lambda}{N}\left(\sum_{i=1}^{N}\sum_{j,k=1}^{3}\left|\frac{\partial \hat{q}_{i,j}}{\partial x_k}-\frac{\partial q_{i,j}}{\partial x_k}\right|\Delta x_k\right)$: this terms promotes a *smooth* solution. This is an important property, as the heat flux enters the energy equation through its divergence. The partial derivatives in this terms are computed using central differences. λ is a regularizing parameter, empirically set to 10.

The batches of data are constructed by sampling subpartitions of the domains of each simulation so that the spatial derivatives can be computed.

2.3.4 Performance

For a complete and in-depth analysis of the performance of the model on various flows, the reader is referred to the original paper (Fiore, Koloszar, Fare, et al., 2022). Adding onto what was found by the authors, in this section the model will also be used *a priori* using the RANS database created for this project, which will be further analyzed in chapter 4. Note that the model was re-trained using only high fidelity channel flow data, to achieve a fair comparison with the multi-fidelity models built for this project.

In the Re_τ window used for training, the single fidelity model proved to be accurate with both training and validation data, showing no apparent signs of overfitting. An example of validation data is shown in Figure 2.6, in which the prediction for

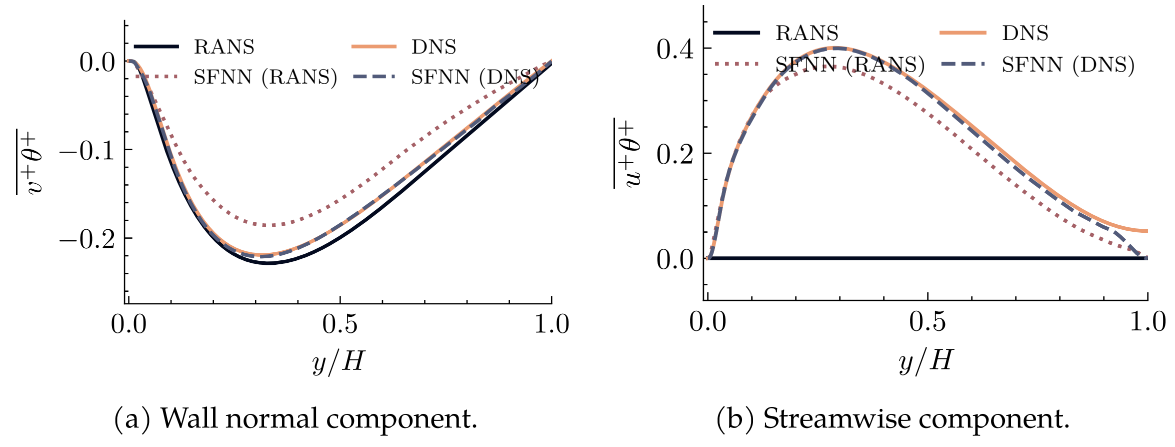
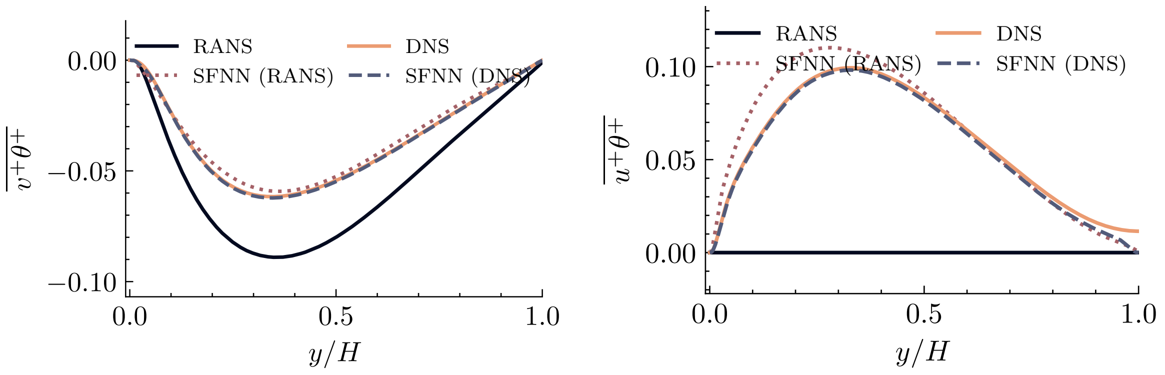


Figure 2.6: Turbulent heat flux prediction for the Single Fidelity Neural Network (SFNN) for $Re_\tau = 395$, $Pr = 0.025$. The solid lines show CFD results, while the dotted lines show the prediction made a priori using either the DNS or RANS-derived inputs and tensors.

$Re_\tau = 395$, $Pr = 0.025$ is shown. It can be seen that the SFNN matches very closely the DNS results when using DNS input data, and only marginally worsen when RANS data is used, underestimating the wall-normal component. Note that, even tough the

RANS simulation gives no information whatsoever about the streamwise component, the model is still able to output a somewhat accurate prediction.



(a) Wall normal component.

(b) Streamwise component.

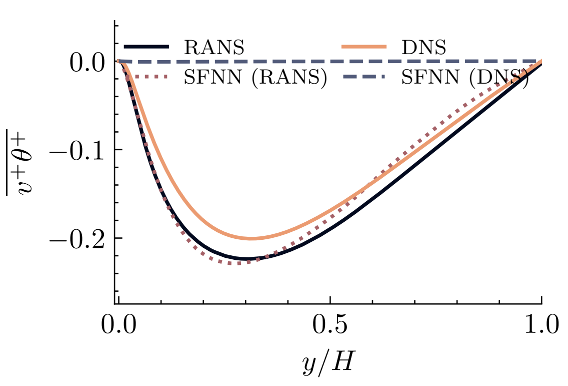
Figure 2.7: Turbulent heat flux prediction for the Single Fidelity Neural Network (SFNN) for $Re_\tau = 395$, $Pr = 0.01$. The solid lines show CFD results, while the dotted lines show the prediction made a priori using either the DNS or RANS-derived inputs and tensors.

Changing the Prandtl number to $Pr = 0.01$ does not alter in any negative way the model's accuracy, as shown in Figure 2.7. It's interesting to note that in this case the SFNN is perfectly able to "mend" the inaccuracy that characterizes the RANS simulation at lower Prandtl number, thus giving a prediction of the wall-normal turbulent heat flux closely matched to the DNS data (Figure 2.7a).

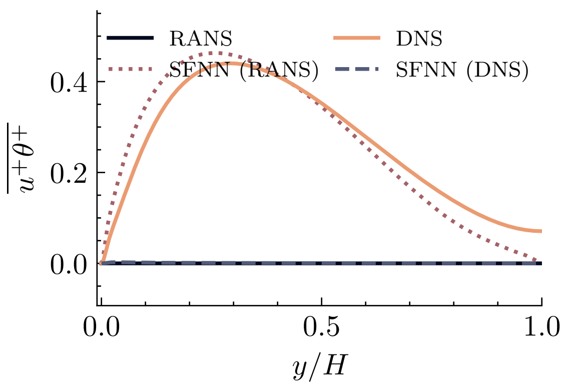
Moving onto higher Reynolds numbers, a distinct pattern starts to develop in the wall-normal turbulent heat flux prediction. Following the left panels in figures 2.8 and 2.9, the SFNN's prediction develops a "bump" (i.e. an overestimation) of the maximum in the wall-normal turbulent heat flux with respect to the RANS solution. Note that, as further analyzed in section 4.4, the error between RANS and DNS tends to decrease when the Reynolds and Prandtl number are increased; for this reason the RANS data was deemed to be adequate for qualitative comparison with the data-driven methods. Given the absence of RANS data for the streamwise component, it is difficult to gather conclusion on whether a similar behavior can be found in that component too.

This kind of behavior is hypothesized to be related to the fact that the model was trained on a database too narrow, that does not include flows with Reynolds number high enough to "inform" the neural network about the physical phenomena characteristic of typical nuclear reactor flows. As such, the single fidelity training database was deemed to be too limited and a different training framework necessary.

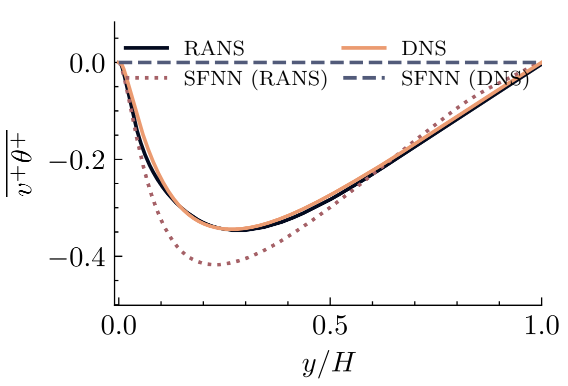
It must be noted that this is not the only limitation at play: in (Fiore et al., 2024) an in depth analysis of the impact of momentum modeling inconsistencies onto the thermal closure was carried out. Nevertheless, as will be shown in chapter 4, for the current project an accurate momentum modeling strategy was used for simple channel



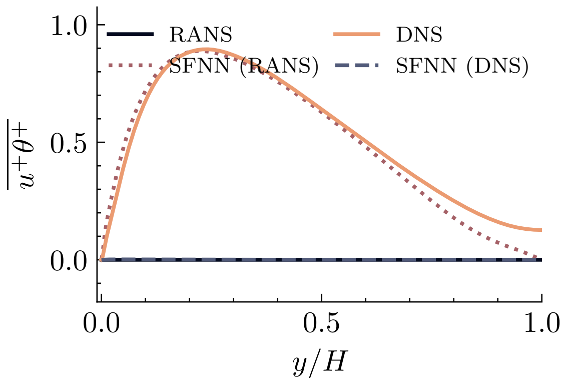
(a) Wall normal component. $Re_\tau = 1020$, $Pr = 0.01$



(b) Streamwise component. $Re_\tau = 1020$, $Pr = 0.01$



(c) Wall normal component. $Re_\tau = 2000$, $Pr = 0.01$



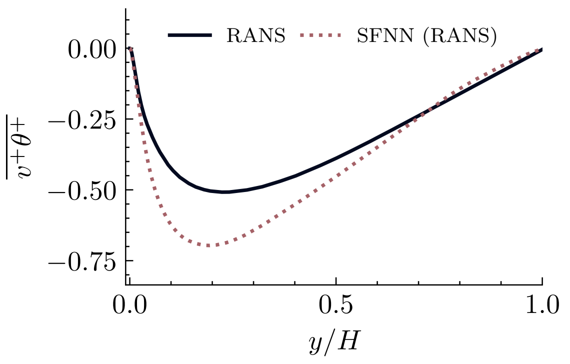
(d) Streamwise component. $Re_\tau = 2000$, $Pr = 0.01$

Figure 2.8: Turbulent heat flux prediction for the Single Fidelity Neural Network (SFNN) for higher Reynolds numbers. The solid lines show CFD results, while the dotted lines show the prediction made a priori using the RANS-derived inputs and tensors. In this case no SFNN data with DNS input is available, as explained in chapter 1.

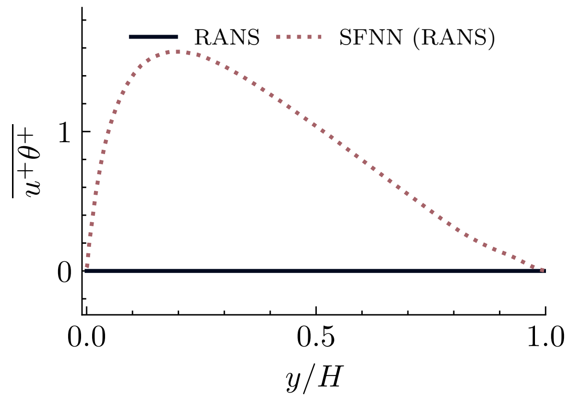
flows via the EBRSM, and thus the effect of modeling is secondary to that caused by the lack of high-Re data.

A promising field of study, which will be the main protagonist of this project, is *multi-fidelity modeling*, that relies on data of different origins (and degrees of accuracy) to "inform" the model about trends in regions where high fidelity data is not available.

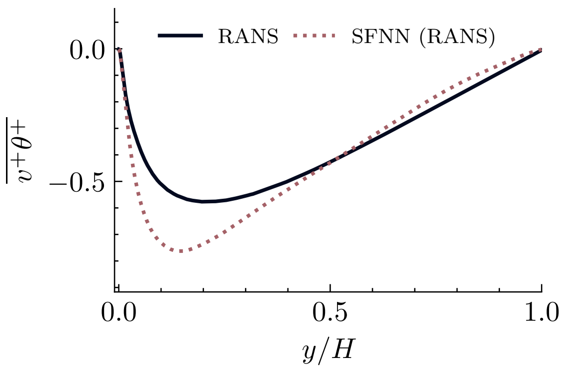
In chapter 3 a brief introduction to multi-fidelity modeling will be given, along with simple typical examples to show its potential and key parameters. In chapter 4 the first step towards the adaptation of said framework towards the heat flux modeling will be taken, in the form of comparing the key differences between simulations of different accuracy both in the physical quantities' space and in the model's input's space. At last, in chapter 5 several models' structures will be built to accommodate the multi-fidelity training and results will be reported.



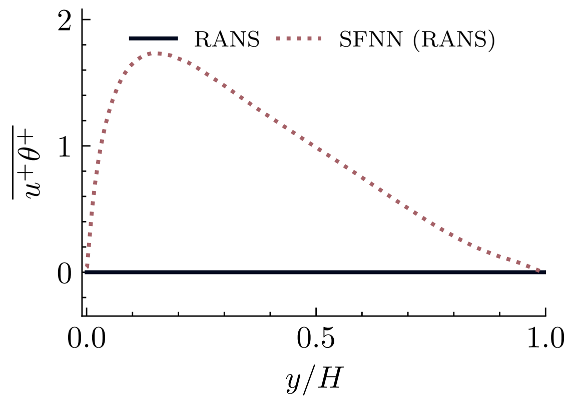
(a) Wall normal component. $Re_\tau = 4000$, $Pr = 0.01$



(b) Streamwise component. $Re_\tau = 4000$, $Pr = 0.01$



(c) Wall normal component. $Re_\tau 6000$, $Pr = 0.01$



(d) Streamwise component. $Re_\tau = 6000$, $Pr = 0.01$

Figure 2.9: Turbulent heat flux prediction for the Single Fidelity Neural Network (SFNN) for higher Reynolds numbers. The solid lines show CFD results, while the dotted lines show the prediction made a priori using the RANS-derived inputs and tensors. In this case no SFNN data with DNS input is available, as explained in chapter 1. Note that for these Reynolds numbers, no DNS data is available for either model usage or comparison.

Chapter 3

Multi-fidelity modeling

3.1 Introduction

Multi-fidelity modeling is a science branch tasked with discovering methods that allow us to learn relationships between data of different origins and qualities and leverage them to maximize the accuracy of a model's predictions. Data can be generally categorized based on *availability or cost* and *accuracy*:

- *High-fidelity* data better captures a phenomenon's features and behavior, but is harder (as in more costly, technically harder or time consuming) to obtain and thus less available. Common examples of high fidelity data are Direct Numerical Simulations (DNS) and experimental measurements.

- *Low-fidelity* data, on the other hand, is a lower quality approximation of the ground truth, but is cheaper and easier to obtain. Common examples include RANS simulations and reduced order models.

In the next section a very brief introduction will be given; for a more complete review on the subject, the reader is encouraged to consult the following review article (Fernández-Godino, 2023). In the context of this project, more attention will be given to *Multi-Fidelity Surrogate Models*, which integrate information from the low and high fidelity models to build a surrogate model, as opposed to Hierarchical Models that do not explicitly build a multi-fidelity surrogate model architecture. In particular, only *Deterministic Models*, which establish a single, fixed relationship between the input and output variables will be treated.

In recent years, multi-fidelity modeling has gathered some traction in aerodynamics and aeroacoustics: the improved accuracy with little added computational cost has proved a key element in shape optimization problems of airfoils (Liao et al., 2021) and wing-body configurations (X. Zhang et al., 2021), as well as in the design of acoustic metasurfaces (Wu et al., 2023).

3.1.1 Model's fidelity

The level of fidelity of a model is determined by three categories:

- *Modeling strategy*: this category is concerned with how the model *mathematically represents* the phenomenon and which assumptions, if any, are used to simplify the problem. An example of different modeling strategies is the different turbulence models used in CFD.
- *Accuracy*: this category refers to how the model discretizes the domain in space and/or time. The mesh resolution and the time-step are a feature of this category.
- *Source*: this category signals the introduction of experimental data, which is generally considered to be of the highest fidelity available.

At the current stage of the project no experimental data is used and numerical data differs in the turbulence modeling used and in the computing mesh; more details will be given in the dedicated section.

3.1.2 Relationship between data of different fidelity levels

The core of any multi-fidelity model is the relationship found between high and low fidelity data. With this in mind, different methods have been proposed in literature, and they can be grouped as follows:

- *Additive and multiplicative corrections*. This group of corrections aims to improve the low fidelity prediction by either an addition or a multiplication. In equations, they can be written as:

$$y_{HF} = y_{LF}(\mathrm{x}) + \delta(\mathrm{x}) \tag{3.1}$$

$$y_{HF} = \rho(\mathrm{x}) \cdot y_{LF}(\mathrm{x}) \tag{3.2}$$

- *Space mapping*. The space mapping strategy aims to align the low fidelity model to the high fidelity model in the parameter space. If in a design space the various fidelity are similar, the optimization can be carried out on the low fidelity, gaining in efficiency with a minimal loss in accuracy.
- *Comprehensive corrections*. This category mixes together the additive and multiplicative correction strategy.

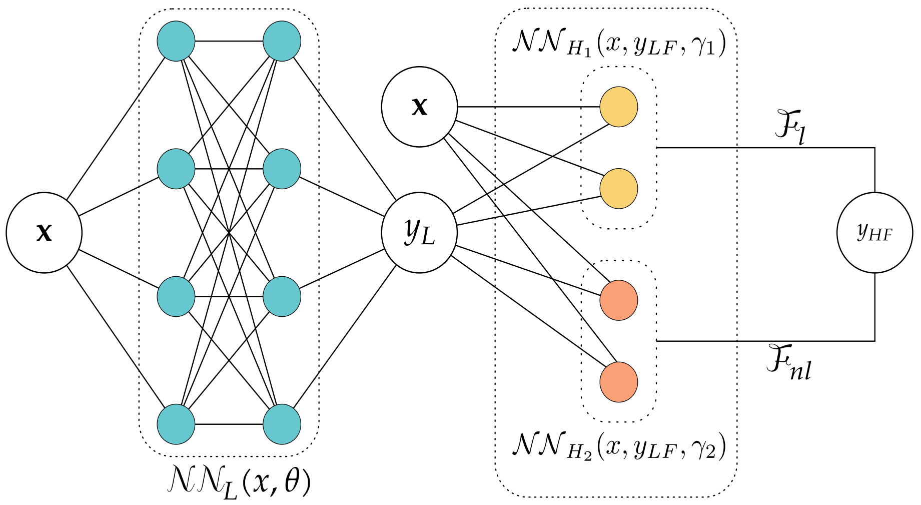
$$y_{LF} = \rho(\mathrm{x}) \cdot y_{LF}(\mathrm{x}) + \delta(\mathrm{x}) \tag{3.3}$$

Note that depending on the approach chosen, the data may come from more than 2 levels of fidelity. Either way, in the following only two levels will be considered.

3.2 Multi-fidelity modeling via composite neural network

One application of multi-fidelity surrogate modeling is found in (Meng & Karniadakis, 2020), in which the authors put forth a generalized auto-regressive scheme expressed as:

$$y_{HF} = \mathcal{F}(y_{LF}) + \delta(x) \tag{3.4}$$



$\mathcal{NN}_{H_1}(x, y_{LF}, \gamma_1)$
$\mathcal{F}_l$
x
x
y_L
y_{HF}
$\mathcal{F}_{nl}$
$\mathcal{NN}_{H_2}(x, y_{LF}, \gamma_2)$
$\mathcal{NN}_L(x, \theta)$

Figure 3.1: multi-fidelity neural network, as represented in (Meng & Karniadakis, 2020). $\mathcal{NN}_L$ is the low-fidelity neural network, while $\mathcal{NN}_{H_1}$ and $\mathcal{NN}_{H_2}$ are respectively the linear and non linear neural network. θ, γ_1 and γ_2 are network parameters.

where $\mathcal{F}(.)$ is an linear/nonlinear function mapping the low-fidelity data to the high fidelity level. $\mathcal{F}$ is then further decomposed in a linear and in a non linear part:

$$\mathcal{F} = \mathcal{F}_l + \mathcal{F}_{nl} \tag{3.5}$$

thus obtaining

$$y_H = \mathcal{F}_l(x, y_{LF}) + \mathcal{F}_{nl}(x, y_{LF}) \tag{3.6}$$

The two linear and non linear functions can be modeled with neural networks, using a structure like the one shown in Figure 3.1. Keeping the figure as reference, going from the left to the right, the components making up the composite neural network are:

- The input layer, taking as input the vector **x**.
- The low-fidelity neural network $\mathcal{NN}_L$. Its role is to approximate the low fidelity data given the input **x** and the network parameters θ.
- The low-fidelity output layer, in which y_L is obtained.
- The high fidelity section, in turn composed of the two neural networks $\mathcal{NN}_{H_1}$ and $\mathcal{NN}_{H_2}$. Notice how these two networks take as input both the low-fidelity prediction and the input **x**, outputting $\mathcal{F}_l$ and $\mathcal{F}_{nl}$. To represent the linear contribution, no activation function is used in $\mathcal{NN}_{H_1}$.
- The high fidelity output y_{HF}, given as a sum of the two high fidelity networks' outputs. This reproduces Equation 3.6.

The unknown parameters θ, γ_1 and γ_2 are learned by minimizing a loss function. A typical loss function in a multi-fidelity framework will have both contribution from the high and low fidelity parts, thus having a form similar to the following:

$$MSE = MSE_{y_L} + MSE_{y_H} + \lambda \sum \beta_i \tag{3.7}$$

in which the Mean Square Error *MSE* loss function for a given fidelity level i is used:

$$MSE_i = \frac{1}{N_i} \sum_{j=1}^{N_i} (|y_i^* - y_i|^2) \tag{3.8}$$

Note that the choice of *MSE* as the error function is purely symbolic here and can be varied according to the problem at hand, as will be clearer in the relevant section later on.

The term $\lambda \sum \beta_i$ represents the *regularization term*, which penalizes the magnitude of the parameters β_i of $\mathcal{NN}_{LF}$ and $\mathcal{NN}_{H_2}$.

3.3 Toy problems

To better understand all the factors at play in the creation and usage of the framework presented in the previous sections, some typical regression toy problems are replicated with the composite neural network structure. The problems are coded using Python as the programming language and PyTorch as the main machine learning package.

3.3.1 Forrester function

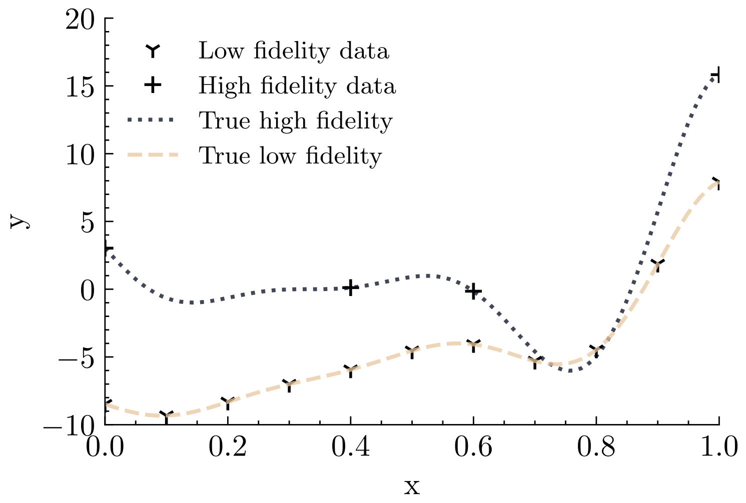


Figure 3.2: Forrester function and sampled data points.

The first toy problem is represented by the Forrester function from (Forrester et al., 2008), in which high and low fidelity data is given by the following functions:

$$y_{LF}(x) = A(6x-2)^2 \sin(12x-4) + B(x-0.5) + C, \quad x \in [0,1] \tag{3.9}$$
$$y_{HF}(x) = (6x-2)^2 \sin(12x-4) \tag{3.10}$$

with $A = 0.5$, $B = 10$ and $C = -5$. To keep consistency with literature, the same sampling points are used as the one used in Meng and Karniadakis, 2020, with 11 equally spaced low fidelity points $[x_{LF}, y_{LF}]$ and 4 high fidelity points $[x_{HF}, y_{HF}$ (shown in figure 3.2).

The neural network is structured as follows:

- $\mathcal{NN}_{LF}$ composed of 2 hidden layers with 20 neurons each, using the hyperbolic tangent as activation function.
- $\mathcal{NN}_{H_1}$ composed of no hidden layers.
- $\mathcal{NN}_{H_2}$ composed of 2 hidden layers with 10 neurons each, using the hyperbolic tangent as activation function.

Differing from the cited paper, here the Adam optimizer is used, as its implementation in PyTorch allows for different settings of learning rate and weight decay for each of the different parameters and networks. A working training configuration is found to be the one with:

- Adam optimizer, with weight decay on $\mathcal{NN}_{LF} = 0.001$ and $\mathcal{NN}_{H_2} = 0.01$.
- Cyclical learning rate with lower and upper bounds $[0.001, 0.01]$ and a triangular pattern with *cyclelength* $= 10000$. This is not crucial as the training finds an acceptable solution even with a constant learning rate, but using a cyclical LR helps the optimizer traverse saddle points faster (Smith, 2017).
- The loss function chosen is the MSE for both the training and validation data.

As can be seen from figure 3.3, the composite neural network is able to learn the main features from the low fidelity data and use them to enhance the high fidelity prediction. It is especially important to notice that the multi-fidelity framework is able to accurately predict the high fidelity function in the $x \in [0.65,\ 0.9]$ interval, where no high fidelity data is sampled for training.

In Figure 3.4a the output of the two high fidelity subnetworks is shown: the entirety of the high fidelity prediction is due to $\mathcal{NN}_{\mathcal{H}1}$, which is understandable when considering that the two fidelity levels are linearly correlated. $\mathcal{NN}_{\mathcal{H}2}$ gives no contribute whatsoever.

Figure 3.4b shows the single fidelity prediction, obtained by training on the high fidelity network on the high fidelity data only. It is clearly seen how the single fidelity is not capable of predicting relevant features of the true high fidelity function such as the main valley.

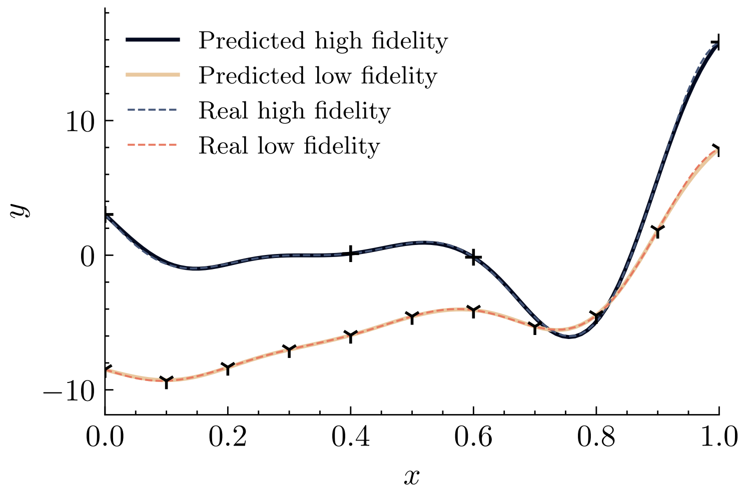
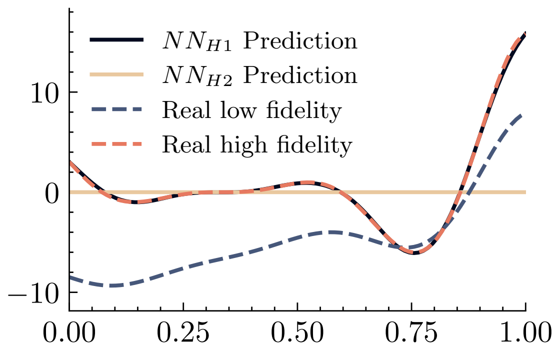
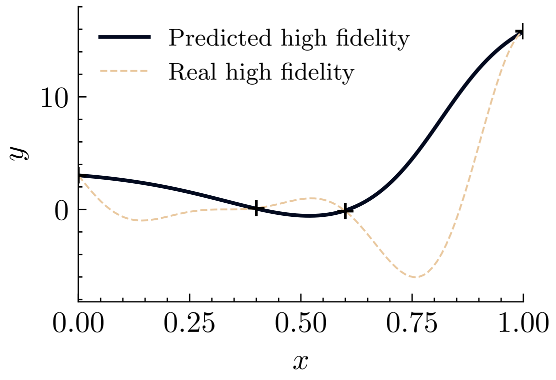


Figure 3.3: Multi-fidelity prediction of the Forrester function. The sampled points are also shown.



(a) Output of the two high fidelity sub-networks.



(b) Single fidelity prediction of the Forrester function using only high fidelity data on the Forrester function.

3.3.2 Forrester function with a discontinuity

A different toy function is the Forrester function with an added discontinuity, in which the low fidelity data can be expressed as:

$$y_{LF}(x) = \begin{cases} 0.5(6x-2)^2 \sin(12x-4) + 10(x-0.5) - 5 & 0 \le x \le 0.5 \\ 3 + 0.5(6x-2)^2 \sin(12x-4) + 10(x-0.5) - 5 & 0.5 < x \le 1 \end{cases} \tag{3.11}$$

and the high fidelity data can be written as:

$$y_{HF}(x) = \begin{cases} 2y_{LF}(x) - 20x + 20 & 0 \le x \le 0.5 \\ 4 + 2y_{LF}(x) - 20x + 20 & 0.5 < x \le 1 \end{cases} \tag{3.12}$$

For this test case, 38 low fidelity data points and 5 high fidelity data points are sampled from the above definitions; both the sampled data points and the high/low fidelity

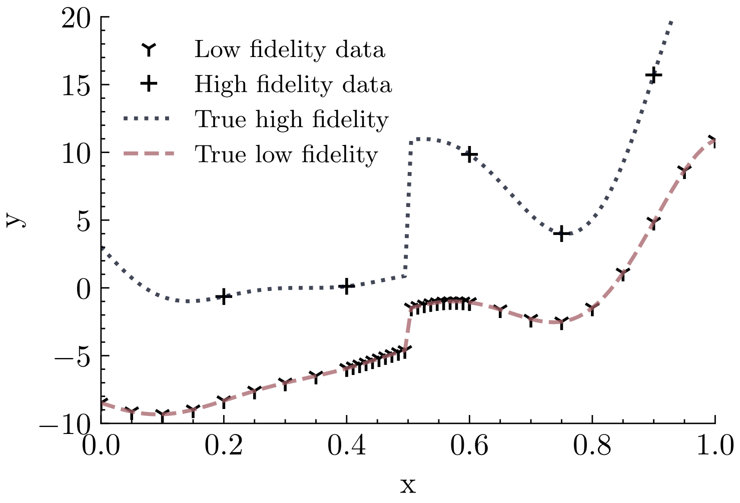


Figure 3.5: Data sampling for the discontinuous test function. The dotted and dashed lines represent the "ground truth" for the high and low fidelity functions respectively. Note how no high fidelity point is sampled in the $(0.4, 0.6)$ interval, giving no information whatsoever about the discontinuity.

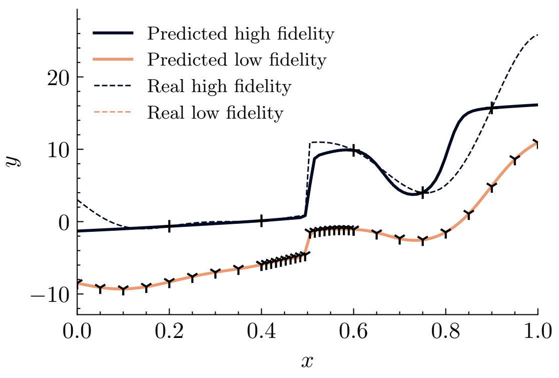
functions are shown in Figure 3.5. The high fidelity data is sampled as to not capture relevant information regarding the discontinuity, while the low fidelity data is sampled with an higher density in the discontinuity region.

Using the same network structure as for the first toy problem does not yield good results for this problem, as shown in Figure 3.6. While the low fidelity prediction is accurate, the high-fidelity prediction is off in the low and high x ranges, even though it vaguely captures the discontinuity.

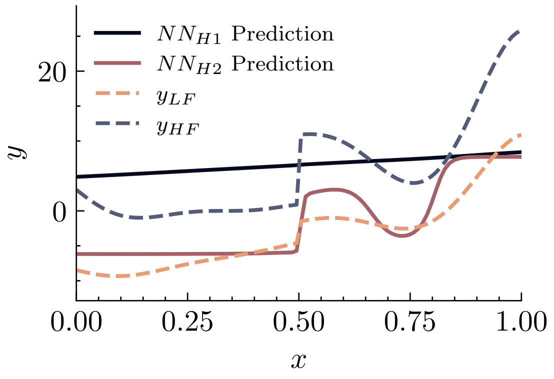
A much better performance is achieved switching to the following structure:

- $\mathcal{NN}_{\mathcal{LF}}$ composed of 2 hidden layers of 10 neurons each.
- $\mathcal{NN}_{\mathcal{H}1}$ composed of 1 hidden layer of 10 neuron with no activation function. Note that this is the same configuration used in the ArchivX version of (Meng & Karniadakis, 2020).
- $\mathcal{NN}_{\mathcal{H}2}$ composed of 2 hidden layers of 10 neurons each.
- A key element in this case is the regularization applied to each network. A working configuration is found to be the one with $\lambda_{LF} = 5^{-5}$, $\lambda_{NN_{H2}} = 5^{-3}$ and no regularization on the linear network. A multi-step learning rate scheduler was also used, pictured in Figure 3.8a.

With this structure the prediction is much more accurate, as shown in Figure 3.7, especially in the $x > 0.8$ region. Notably, the model correctly predicts the discontinuity even though no high fidelity data is available. An interesting difference with respect to the previous toy problem is that in this case the $\mathcal{NN}_{\mathcal{H}2}$ output is non zero and gives



(a) Discontinuous function prediction.



(b) Linear and non linear functions predicted by the neural networks.

Figure 3.6: Results for the second toy problem using the same neural network as the one used for the first toy problem.

a small contribution to the final result. Still, $\mathcal{NN}_{\mathcal{H}1}$ heavily resembles the true high fidelity function and the other contribution is almost constant in each of the windows before and after the jump.

Another interesting feature that the multi-fidelity model is able to capture is the slight increase for increasingly low x values close to zero. In fact no high fidelity data were sampled in the $x < 0.2$ region, but the model manages to reproduce the correct trend regardless.

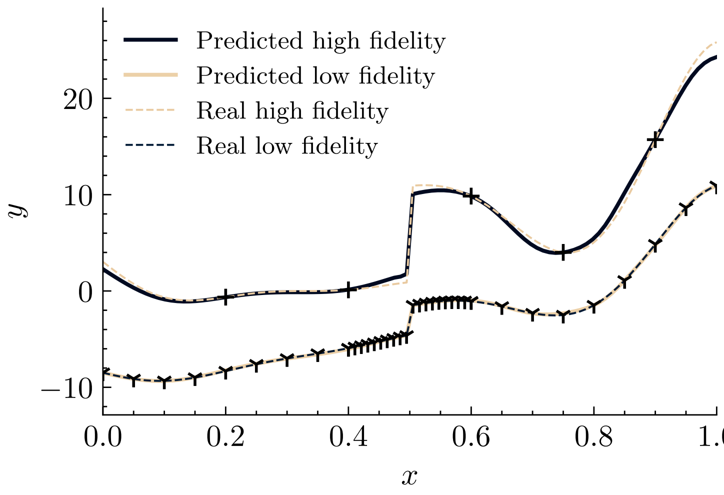
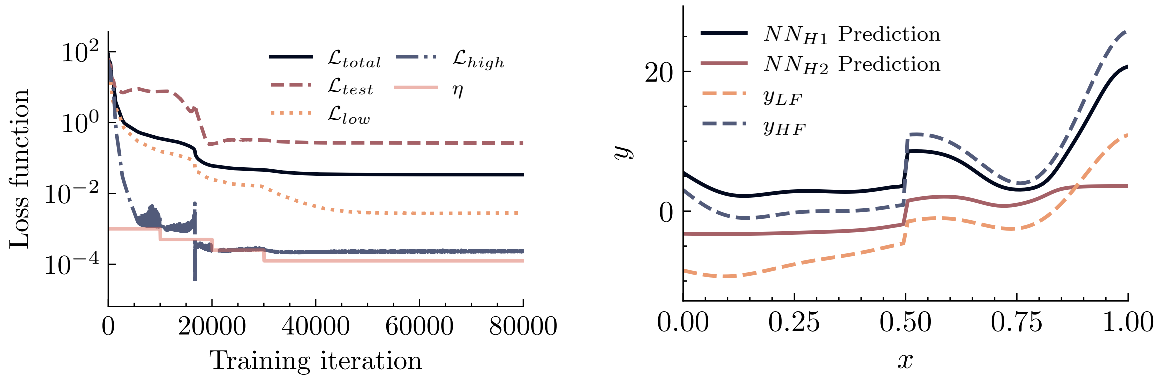


Figure 3.7: Discontinuous function prediction with the multi-fidelity strategy.



(a) Loss functions $\mathcal{L}$ and learning rate η.

(b) $\mathcal{NN}_{\mathcal{H}1}$ and $\mathcal{NN}_{\mathcal{H}2}$ output.

Figure 3.8: Training statistics, $\mathcal{NN}_{\mathcal{H}1}$ and $\mathcal{NN}_{\mathcal{H}2}$ contribution in the discontinuous function prediction with the multi-fidelity strategy.

3.3.3 Continuous function with nonlinear correlation

The last toy problem is a case of continuous function with nonlinear correlation. The two functions are given by:

$$y_{LF}(x) = \sin(8\pi x) \tag{3.13}$$

$$y_{HF}(x) = (x - \sqrt{2})y_{LF}^2(x) \tag{3.14}$$

The fact that, for some x values, the low fidelity is not only wrong in magnitude but also qualitatively wrong with respect to the high fidelity, makes this a case of *adversarial training*. This can clearly be seen in Figure 3.9, where the odd valleys of the high fidelity correspond to peaks of the low fidelity.

For this case 51 $[x_{LF}, y_{LF}]$ and 14 $[x_{HF}, y_{HF}]$ points were sampled and the first structure was used, meaning:

- $\mathcal{NN}_{\mathcal{LF}}$ with a 4 × 20 structure.
- $\mathcal{NN}_{\mathcal{H}1}$ with no hidden layers.
- $\mathcal{NN}_{\mathcal{H}2}$ with a 2 × 10 structure.

Once again the frameworks proves to be robust and able to discover the correlation between the two data origins. In Figure 3.11 it is clear how the model correctly captures the high fidelity function even though only one or two points per valley are available and the low fidelity function suggests an opposite trend. This time the "feature analysis" in Figure 3.10b starts to lose a clear pattern.

For completeness' sake a single, high fidelity fitting with the same y_{HF} data points is shown in Figure 3.12, which makes clear how the high fidelity alone is not capable of extracting the true trend of the data.

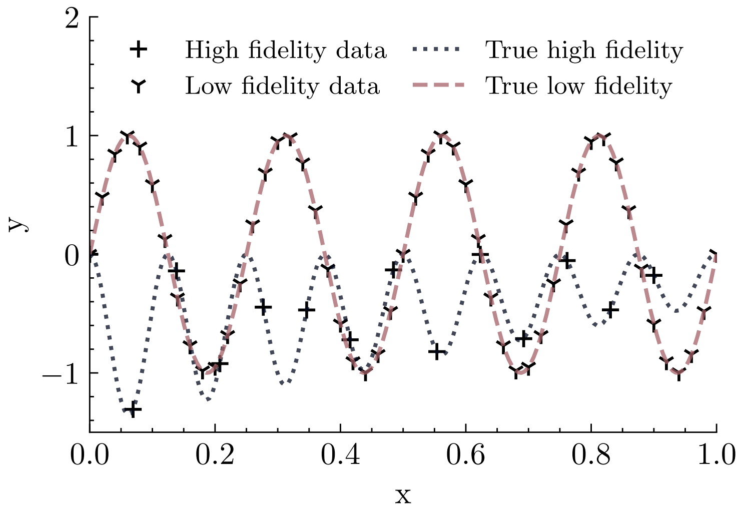
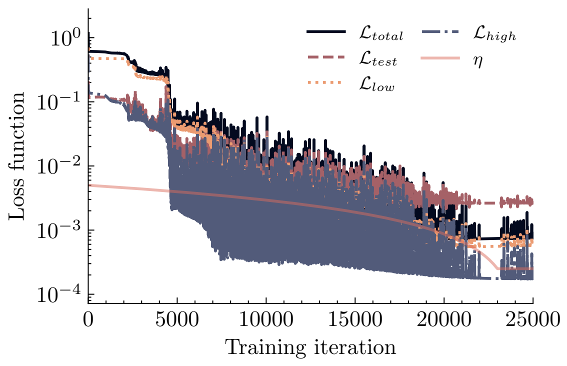
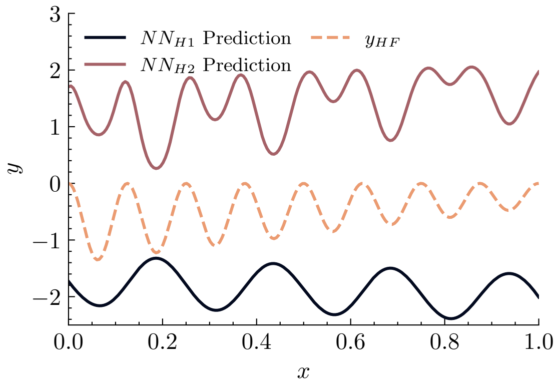


Figure 3.9: Third toy function. Note that the low fidelity is *adversarial*, meaning that in some regions it's quantitatively and qualitatively wrong with respect to the high fidelity data.



(a) Loss functions. $\mathcal{L}_{total}$



(b) $NN_{\mathcal{H}1}$ and $NN_{\mathcal{H}2}$ output.

Figure 3.10: Training statistics, $NN_{\mathcal{H}1}$ and $NN_{\mathcal{H}2}$ contribution in the continuous function prediction with the multi-fidelity strategy.

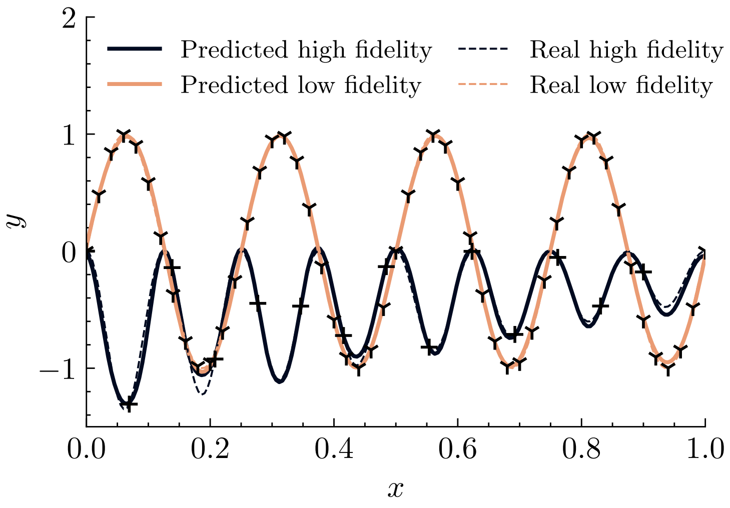


Figure 3.11: multi-fidelity fitting of the continuous function with non linear correlation.

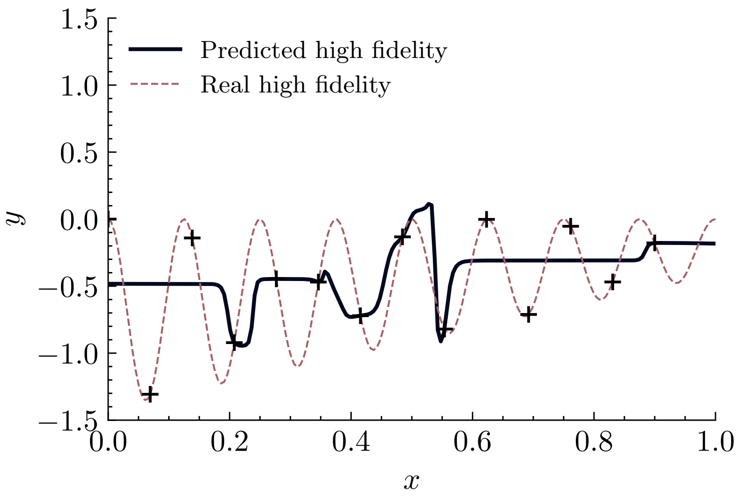


Figure 3.12: Single fidelity fitting of the continuous function with non linear correlation.

3.4 Towards multi-fidelity turbulent heat flux modeling

In this chapter the multi-fidelity neural network framework was used to solve simple test problems. Even though in the original paper it was applied to more complex test cases, such as a 20D function, there still is a complexity step up when considering the turbulent heat flux modeling. This step up is given mainly by the need to enforce first principles such as the Galilean and rotational invariance, but the scarcity of data combined with the high dimensional input play their part too.

Nevertheless, the adoption of a multi-fidelity training strategy could still provide with an increase in accuracy outside of the training range of the original model and provide a valid alternative to existing modeling solutions. The main focus will thus be on the ability of the model to reproduce *physical trends* and not necessarily exact values.

In chapter 4 the implementation will start with an analysis and comparison of the high and low fidelity dataset, while in chapter 5 some multi-fidelity models for the turbulent heat flux and their results will be shown.

Chapter 4

RANS/DNS input comparison

4.1 Introduction

In this chapter the databases used in the training and validation of the multi-fidelity thermal turbulence model will be analyzed and compared. The general flow characteristics such as flow geometry, fluid properties and turbulence modeling strategies will be shown first. Then we will move onto the comparison of physical quantities and model's invariants. The last step will be a Principal Component Analysis of the channel flow, to further explain how the DNS and RANS approaches differ from each other.

4.2 Channel flows

4.2.1 RANS channel database description

Re	*Pr*
180	0.01, 0.025
395	0.01, 0.025
640	0.01, 0.025
1020	0.01, 0.025
2000	0.01, 0.025
4000	0.01, 0.025
6000	0.01, 0.025

Table 4.1: RANS channel database.

The RANS channel data used was created ad-hoc for this project, and is made up of 14 simulations with different combinations of Re_τ and Pr numbers, as shown in table Table 4.1. The simulations were carried out using the OpenFOAM suite, using the Elliptic Blending Reynolds Stress Model (EBRSM) (Manceau & Hanjalić, 2002; Manceau, 2015) and a custom implementation of the Manservisi model (Manservisi & Menghini, 2014) for the turbulent heat transfer. The geometry is that of a fully developed 2D channel flow with constant wall heat flux q_w and the following parameters:

- $\nu = 1.74 \cdot 10^-7 \text{ m}^2/\text{s}$
- $C_p = 145.75 \text{ m}^2/\text{s}^2\text{ K}$
- $\alpha = 1.74 \cdot 10^-5 \text{ m}^2/\text{s}$
- $\beta = 0.01 \text{ K}^-1$
- $\kappa = 26.22 \text{ kg m/s}^3\text{ K}$
- $T_{ref} = 278 \text{ K}$
- $\rho = 10340 \text{ kg/m}^3$
- $q_w = 36 \cdot 10^4 \text{ kg/s}^3$

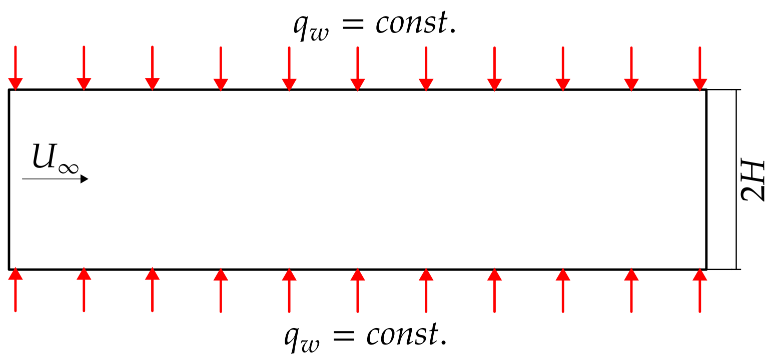


Figure 4.1: RANS channel flow geometry and boundary conditions.

To be consistent with the DNS literature, the adimensionalization of the values is carried out using standard wall values like u_τ and ν for the velocity and viscosity, while the *friction temperature* is defined as $t_\tau = \frac{q_w}{\rho c_p u_\tau}$.

The OpenFOAM output consists in several text files that contains the cell values for various fields sampled along the channel height at its midpoint. The files are then read with Python and all the necessary fields (invariants, tensors and post-processing data) are computed after the simulation has run.

4.2.2 DNS channel database description

Re_τ	Pr
180	$0.01[T]$, $0.025[T]$, $0.05[T]$, $0.1[T]$, $0.2[T]$, $0.4[T]$, $0.6[T]$, $0.71[T]$
395	$0.01[T/V]$, $0.025[T/V]$, $0.71[T]$,
590	$0.01[T]$
640	$0.025[T]$, $0.71[T]$
1000	$0.01[V]$
1020	$0.71[N]$
2000	$0.0100[V]$

Table 4.2: DNS channel database simulations and whether they were used for training $[T]$ or validation $[V]$. The $Re_\tau = 395$ cases are labeled as [T/V] because each training was repeated including and excluding them so that the model's robustness could be checked.

The DNS data for the channel flow was taken from (Bricteux et al., 2012), (Kawamura et al., 2000) and Alcántara-Ávila et al., 2018, even tough for the training of the model only the first two database cited were (partially) used, since the last one does not have all of the quantities needed for the computation of the invariants and was thus only used during the validation. The $Re_\tau = 1020$ case did not have its full turbulent statistics neither, and was thus used for turbulent heat flux comparison only. All the DNS data used for the training or validation of the model is shown in table Table 4.2. Note that in (Kawamura et al., 2000) 4 different cases were run, which were either Couette flows or Poiseuille flows with either constant q_w or constant wall-temperature difference ΔT_w; for the task at hand only the Poiseuille flows with constant q_w were used.

Both the DNS and the RANS data were interpolated on the same grid, made up of 1000 equally spaced grid elements across the channel height.

4.3 Impinging jet

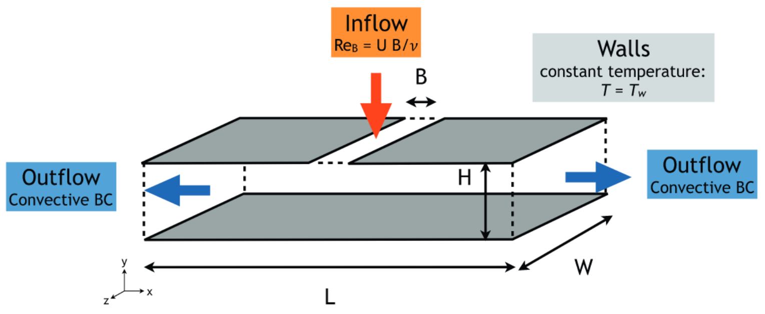


Figure 4.2: Computational setup for the impinging jet. Taken from (Duponcheel & Bartosiewicz, 2021).

Another flow considered will be the impinging jet, a configuration in which a hot jet is blown via a slit in the top wall of a channel onto a lower, cooler wall. Impinging jets are flows of interest because they are a canonical case of wall interaction and a common and efficient cooling strategy (Duponcheel & Bartosiewicz, 2021). Temperature and velocity fields for a RANS simulation of an impinging jet at $Re = 5700$, $Pr = 0.01$ are shown in Figure 4.4 and 4.3 respectively.

The Reynolds number can be defined as a function of the jet mean velocity U and width B: $Re = \frac{UB}{\nu}$. The hot jet coming out of the slit is isothermal with temperature $T = T_j$, while the wall are kept at temperature $T = T_w$. The temperature difference $\Delta T = T_j - T_w$ is used as reference temperature.

An important geometric property is the aspect ratio $AR = \frac{H}{B}$, given by the ratio between the height H and the jet width B. Higher AR values represents cases in which the jet is closer to a free jet, but they also require longer domains so that the boundary conditions have negligible effects. In the following, jets with $AR = 2$ will be considered.

In the current project, impinging jets will be used at the end to assess the performance of the model and to enlarge the low fidelity training database.

4.3.1 DNS impinging database description

Since the DNS database from (Duponcheel & Bartosiewicz, 2021) did not have enough statistics to be used for training, it was only used in the validation phase as a benchmark.

In particular, the flow case with $Re_B = 5700$ and $Pr = 0.01$ was chosen, with the inlet being fully turbulent.

4.3.2 RANS impinging database description

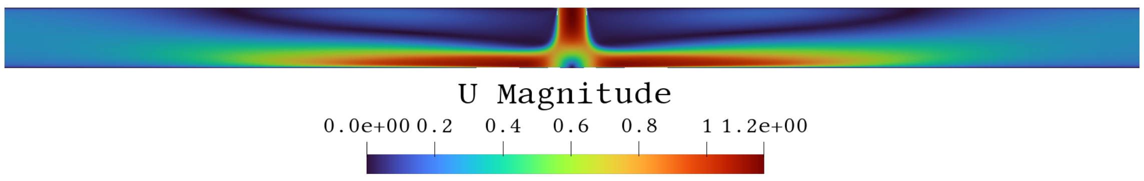


Figure 4.3: Velocity magnitude in the RANS impinging jet flow with $Re = 5700$, $Pr = 0.01$.

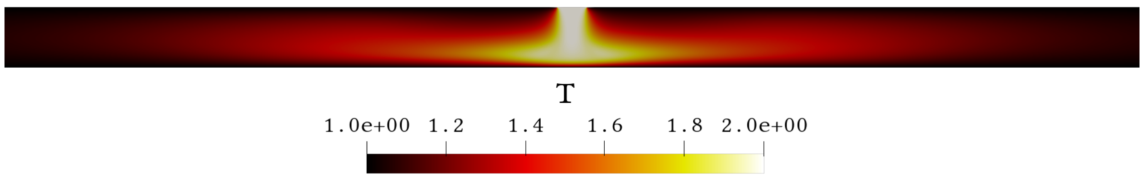


Figure 4.4: Normalized temperature field in the RANS impinging jet flow with $Re = 5700$, $Pr = 0.01$.

The same flow as in the DNS case was replicated using a RANS approach, to be able to enlarge the RANS training set. The simulation was carried out in OpenFOAM, using an EBRSM model (Manceau, 2015) for the Reynolds Stresses and the Manservisi (Manservisi & Menghini, 2014) model for the heat transfer. In Figure 4.3 and Figure 4.4 the velocity and temperature fields respectively are represented. The jet velocity is $U_j = 1$, and the temperatures are $T_j = 2$ and $T_w = 1$. Before passing the data to the data-driven models all the dimensional data are adimensionalized using either the ΔT or the U_j, and all of the fluid properties are defined as functions of those values to have a fully parametric model:

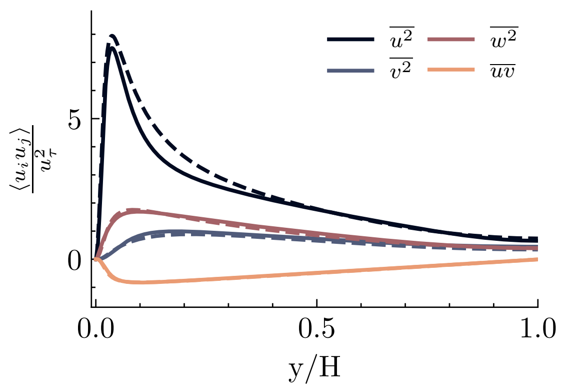
- $\nu = \frac{U_j B}{Re}$
- $\alpha = \frac{\nu}{Pr}$
- $\rho = const. = 1$
- $C_p = 1$

- $\kappa = \alpha \rho C_p$
- $\mu = \frac{U_j B \rho}{Re}$

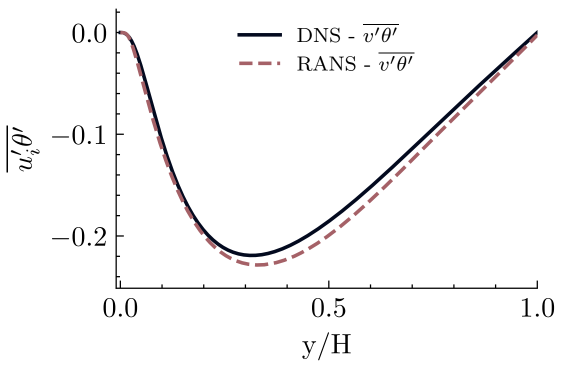
Buoyancy effects are neglected and the fluid composition is considered constant. In subsection 4.4.2 the DNS and RANS results will be compared.

4.4 Physical quantities comparison

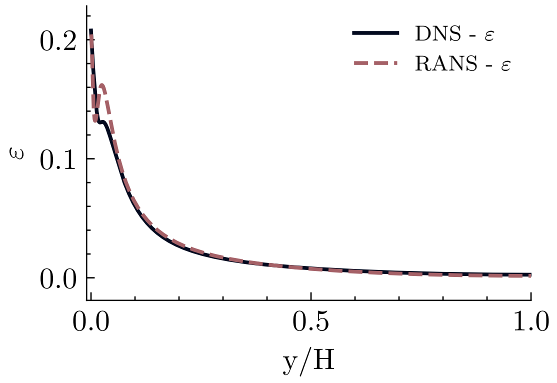
4.4.1 Channel flow



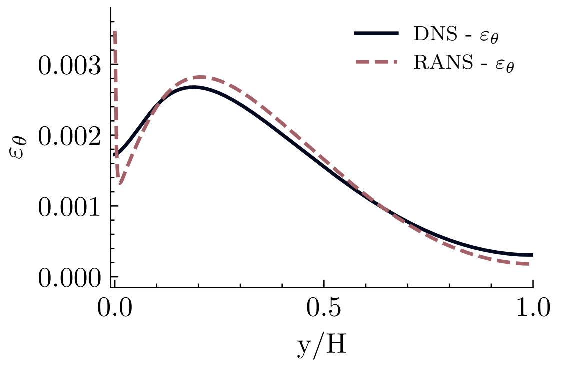
(a) Reynolds stresses.



(b) Wall-normal turbulent heat flux.



(c) Turbulent dissipation rate.



(d) Thermal dissipation rate.

Figure 4.5: Various turbulent quantities for the channel flow along the channel height for $Re_\tau = 395$ and $Pr = 0.025$.

The first step in assessing the difference between the DNS approach and the RANS approach was to compare physical quantities such as average fields and turbulent statistics. For the majority of the turbulent quantities, in particular the momentum-related ones, a good agreement is shown between the DNS and the RANS dataset, as shown in panel Figure 4.5. An accurate prediction of the Reynolds stress tensor is key, as discussed in (Fiore, Koloszar, Fare, et al., 2022), and thus the choice of an EBRSM is justified since the $\overline{u_i u_j}$ components are well captured even at low Reynolds, as shown in Figure 4.5a. The only relative inaccuracy in the RANS prediction is shown close to the wall for the ε and ε_θ values, in which small valleys are noticeable in the lower fidelity prediction. Also note that the chosen thermal turbulence model *does not predict any stream-wise turbulent heat flux*, or any non temperature gradient-aligned turbulent heat flux for that matter.

The biggest difference is observed on the k_θ prediction, shown in Figure 4.6; for this particular flow the peak k_θ suffers a percentage error of -18.79%.

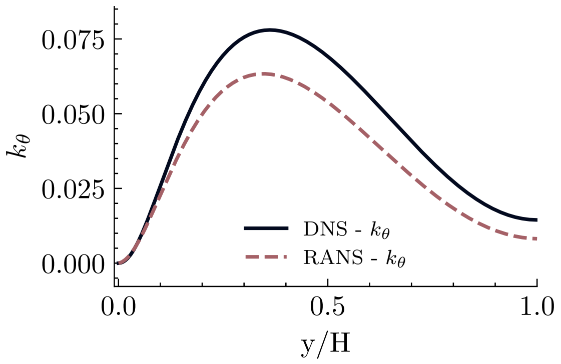


Figure 4.6: Thermal variance k_θ along the channel height for $Re = 395$ and $Pr = 0.025$.

The RANS modeling is also inaccurate in predicting the turbulent heat flux and the entity of the error heavily depends on the flow conditions. To quantify such error, the peak relative error was computed for every matching DNS-RANS conditions as follows:

$$peak\ relative\ error = \frac{\min(DNS) - \min(RANS)}{\min(DNS)} \tag{4.1}$$

The result is shown in Figure 4.7, which makes clear that in general the prediction made by the turbulence model becomes more and more accurate as the Re and Pr increase. As mentioned, this result is used to justify the usage of RANS data as baseline for comparison at high Re_τ, where no DNS data is available.

4.4.2 Impinging jet

While in the case of the channel flow it is reasonable to expect the RANS modeling to be relatively accurate, flows with separation and recirculation zones such as the impinging jet pose more of a challenge. To compare the two strategies, data is sampled along the channel height at $x/B = 1, 2, ..., 9$; those locations are shown overlaid onto the streamlines in Figure 4.8.

In Figure 4.9 the DNS and RANS Reynolds stresses and $\overline{v\theta}$ are shown, sampled at $x/B = 1$. The first difference when compared with the channel flow is that the Reynolds stresses prediction by the RANS is much less accurate, especially the $\overline{uu}$ component. This inaccuracy is pronounced close to the lower wall and in the region between the wall jet and the recirculation bubble. The turbulent heat flux component

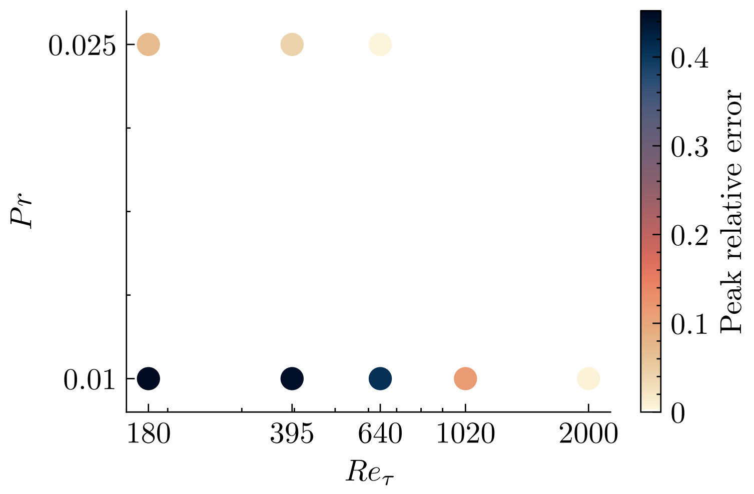


Figure 4.7: Peak relative error Equation 4.1 on the turbulent heat flux for the overlapping DNS/RANS channel simulations. Note that for $Pr = 0.01$ the DNS data is actually computed with $Re_\tau = 590$, but here is compared to the RANS simulation with $Re_\tau = 640$.

$\overline{v\theta}$ is majorly underestimated by the RANS model for coordinates $2y/H < 0.3$, while it is in reasonable agreement with the DNS data further from the lower wall.

Further away from the jet, the RANS prediction gets more accurate both considering the momentum modeling (Figure 4.10a) and the turbulent heat flux prediction (Figure 4.10b). At the $x/B = 5$ coordinate, the Reynolds stress predicted by the EBRSM model are more accurate than at $x/B = 1$, even though the $\overline{uu}$ component is still wrongly computed near the lower wall. The turbulent heat flux maximum and minimum are well predicted in magnitude; the RANS model also manages to correctly predict the location of the minimum, while placing the maximum closer to the lower wall than the DNS does. For a more comprehensive analysis of different turbulence modeling strategies on the same flow, the reader is referred to (De Santis et al., 2019).

Overall, the RANS model is still able to capture the general behavior in most of the domain, even though it is clear how much more challenging it is when compared to a simpler flow like the channel one. This brief analysis of a more complex configuration, should make clear how fundamental it is to train and validate any data driven model against different flows of interest.

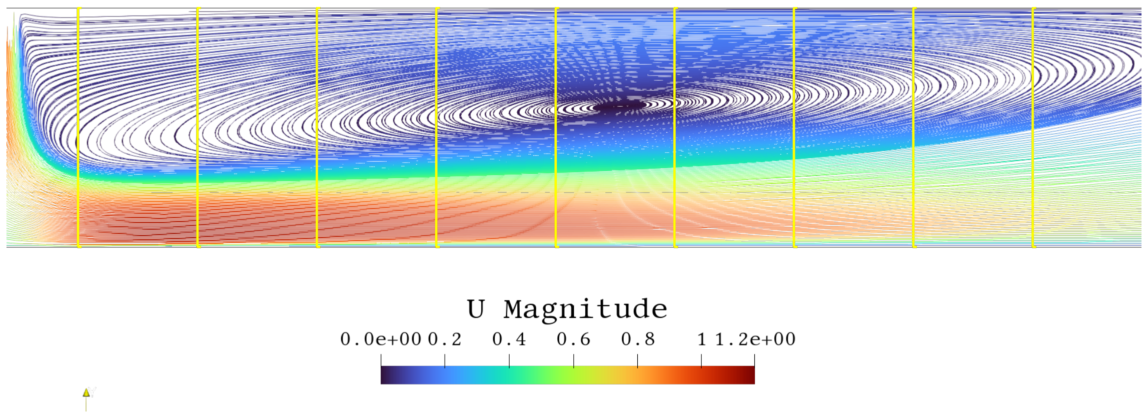
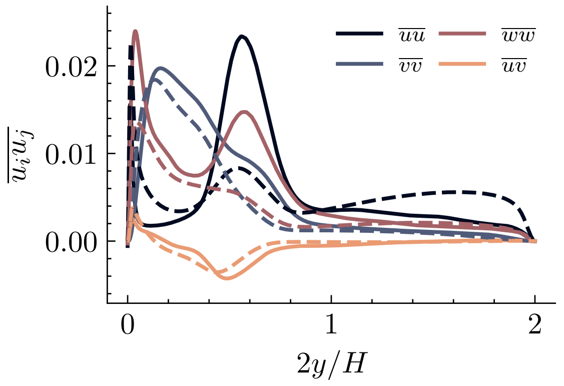
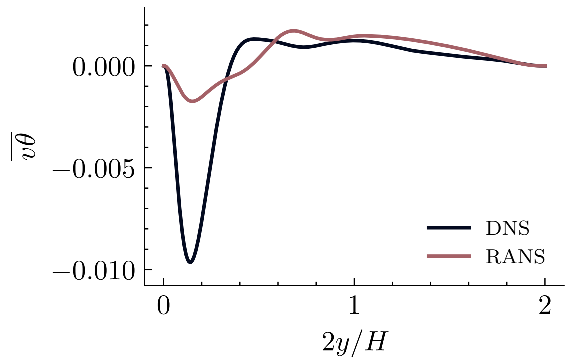


Figure 4.8: Impinging jet sampling locations, shown with the yellow lines on top of the velocity streamlines. Only the right half of the domain is shown.

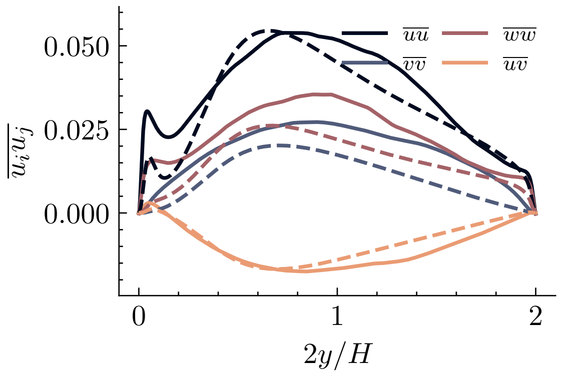


(a) Reynolds stresses. The DNS data is represented with the solid lines, the RANS data with the dashed lines.

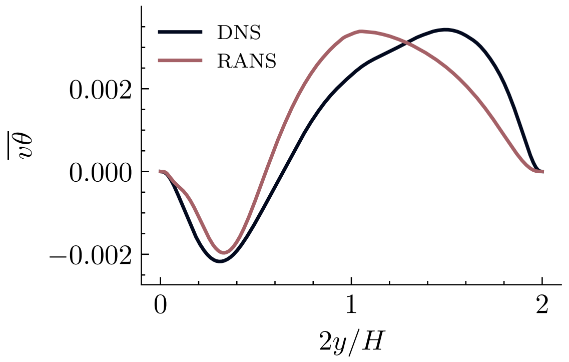


(b) $\overline{v\theta}$ turbulent heat flux component.

Figure 4.9: Reynolds stresses (left) and turbulent heat flux (right) in the impinging jet with $Re = 5700,\ Pr = 0.01$ at $x/B = 1$.



(a) Reynolds stresses. The DNS data is represented with the solid lines, the RANS data with the dashed lines.



(b) $\overline{v\theta}$ turbulent heat flux component.

Figure 4.10: Reynolds stresses (left) and turbulent heat flux (right) in the impinging jet with $Re = 5700,\ Pr = 0.01$ at $x/B = 6$.

4.5 Model's invariants comparison

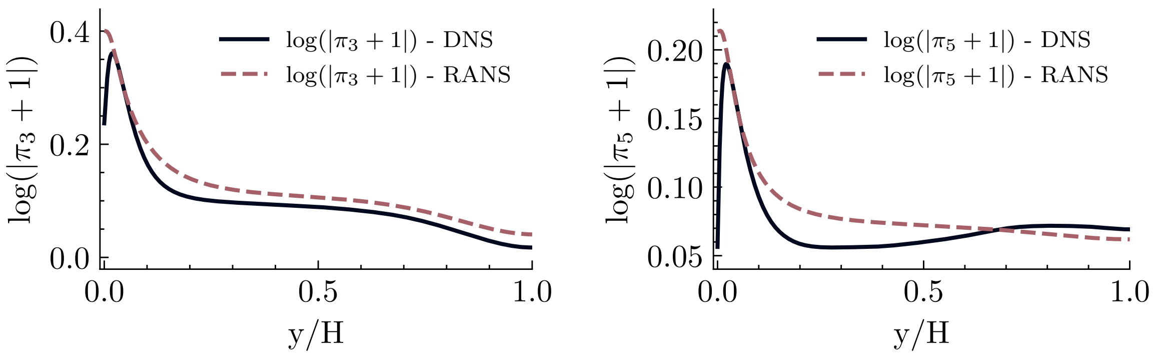


Figure 4.11: **b**-only related invariants along the channel height for $Re = 395$ and $Pr = 0.025$. Here they are plotted after undergoing the $\log(|\pi_i + 1|)$ transformation, meaning that these are the true neural network inputs.

Moving on to the analysis of the actual model's inputs (i.e. the π_i invariants together with Re_t), the same general trend already shown for the physical quantities in the channel flows is observed. The momentum-related invariants which do not involve **b** alone (shown in Figure 4.12) are similar both in trend and in values between the two datasets. This similarity tends to degrade when lower Pr and Re are considered, like what has already been noted about the heat flux prediction.

A more pronounced difference is instead noted when observing invariants which involve either **b** or $\mathbf{b_2}$ only (as far as the tensor basis is concerned). Looking at Figure 4.11 not only the gap all along the channel height is bigger, but also the trend very close to the wall predicted by the RANS is off, as both invariants reach their maximum at the wall in the RANS case, while they do not in the DNS case. This difference is attributed to a combination of the minor differences in the prediction of both the turbulent kinetic energy and the Reynolds stresses. It's interesting to note that π_4, while still being related to **b** via the $\{\mathbf{bS}\}$ term, does not show the same difference close to the wall. Overall, the momentum-related invariants are similar between RANS and DNS simulation, which is expected since an EBRSM model was used for the Reynolds stresses.

The major difference among the invariants is observed in the behavior of π_8 along the channel height, which gets underestimated by the RANS modeling. Given the definition of $\pi_8 = \frac{k_\theta \varepsilon}{k \varepsilon_\theta}$ and the aforementioned error in the k_θ prediction made by the thermal turbulence model, one can attribute the difference in π_8 to the RANS thermal modeling strategy.

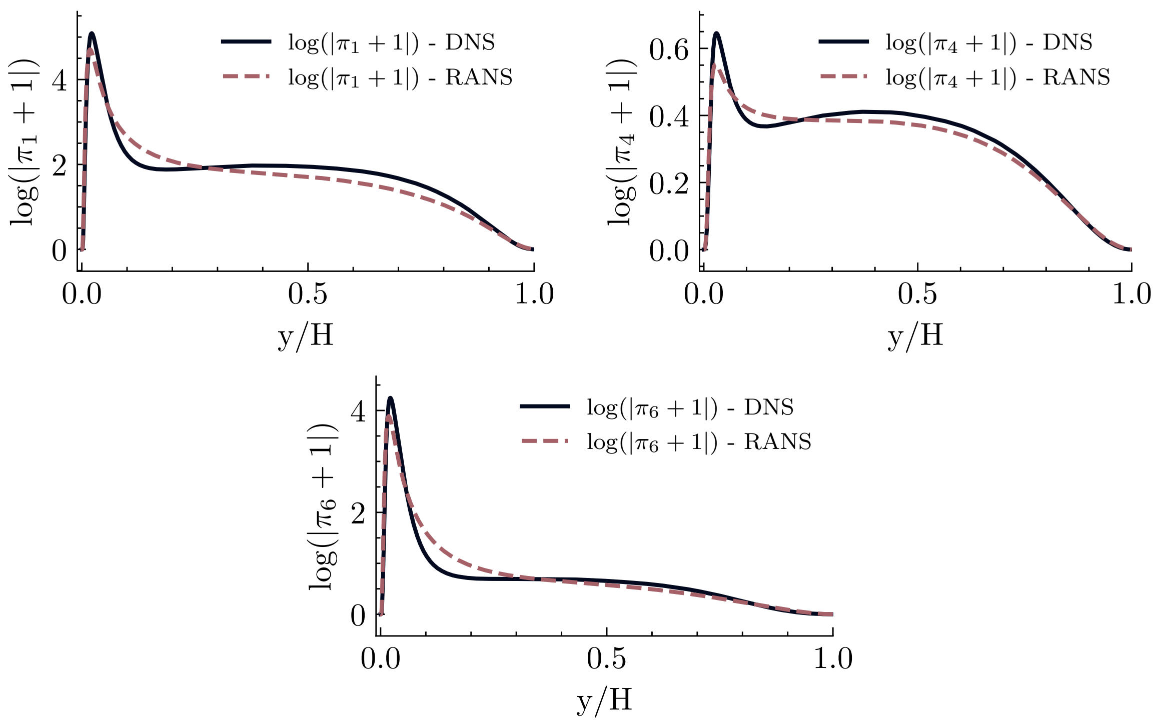


Figure 4.12: Non **b**-only related invariants along the channel height for $Re = 395$ and $Pr = 0.025$. Here they are plotted after undergoing the $\log(|\pi_i + 1|)$ transformation, meaning that these are the true neural network inputs. Also π_2 is not shown since it is identical to π_1 for channel flows.

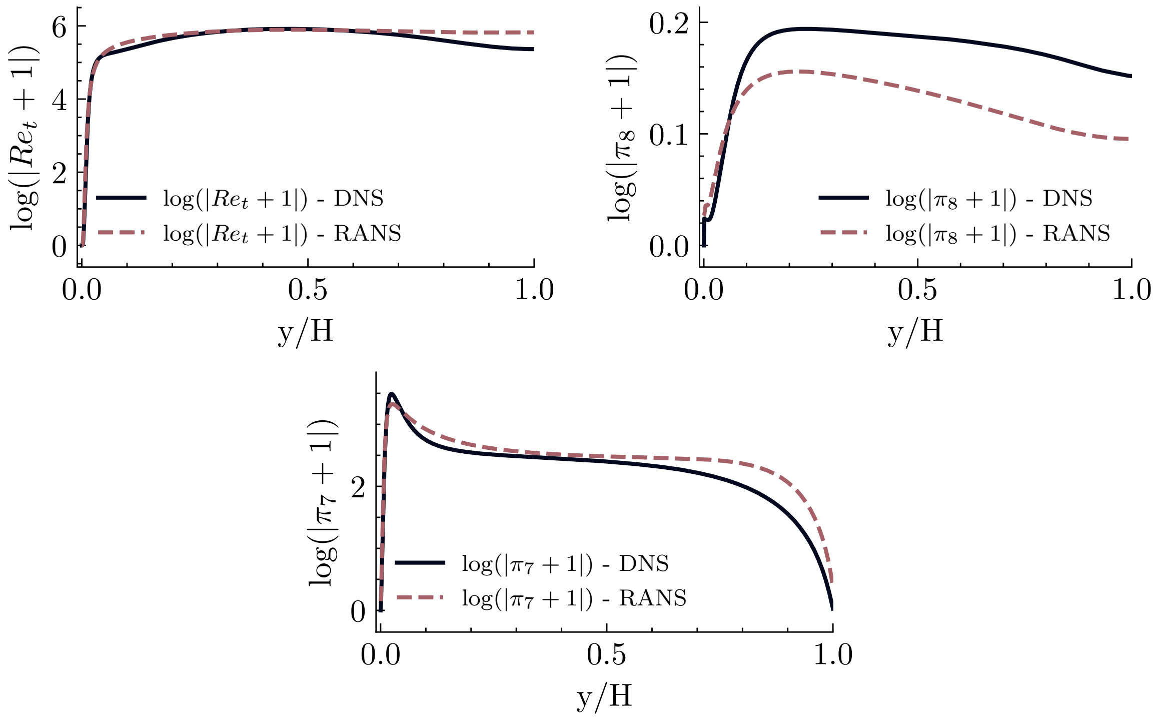


Figure 4.13: Thermal-related invariants along the channel height for $Re = 395$ and $Pr = 0.025$. Here they are plotted after undergoing the $\log(|\pi_i + 1|)$ transformation, meaning that these are the true neural network inputs.

4.6 Principal Component Analysis

In this project, the PCA was used to further understand the difference between the RANS and DNS datasets, especially in the model inputs' space. Hence, the PCA was computed on the DNS data, then the RANS data was projected on the new coordinate system; the scikit Python package was used to handle both the data pre-processing and the actual PCA computation.

4.6.1 Components' analysis

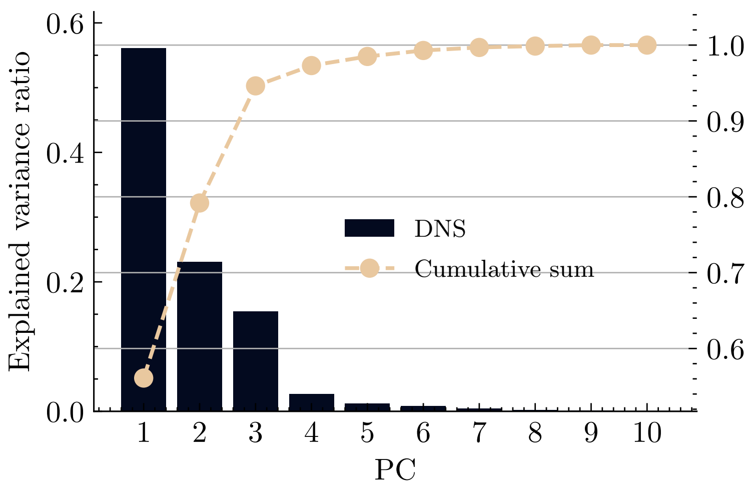


Figure 4.14: Ratio of explained variance for each component and cumulative sum of the variance over the components.

The explained variance ratio computed for each component is shown in Figure 4.14. It is clearly seen that the first component is quite dominant, as it explains around 55% of the variance in the DNS dataset; also it shows how just the first three components are able to encompass more than 90% of the variance and were thus selected for a more in depth analysis.

Figure 4.15 shows the composition of the first three components in terms of the original variables (i.e. the invariants); the PCA groups the invariants as follows:

- Component 1 is influenced mostly by the momentum-only invariants $\pi_1, ..., \pi_6$ and on a smaller scale by π_7.
- Component 2 on the other hand is influenced for the biggest part by thermal-related invariants such as $Pr,\ R$ and π_7.
- Component 3 is dominated by the effect of the Re_t and, in a smaller fashion, by the invariants closely related to the Reynolds stresses such as $\pi_3,\ \pi_4$ and π_5.

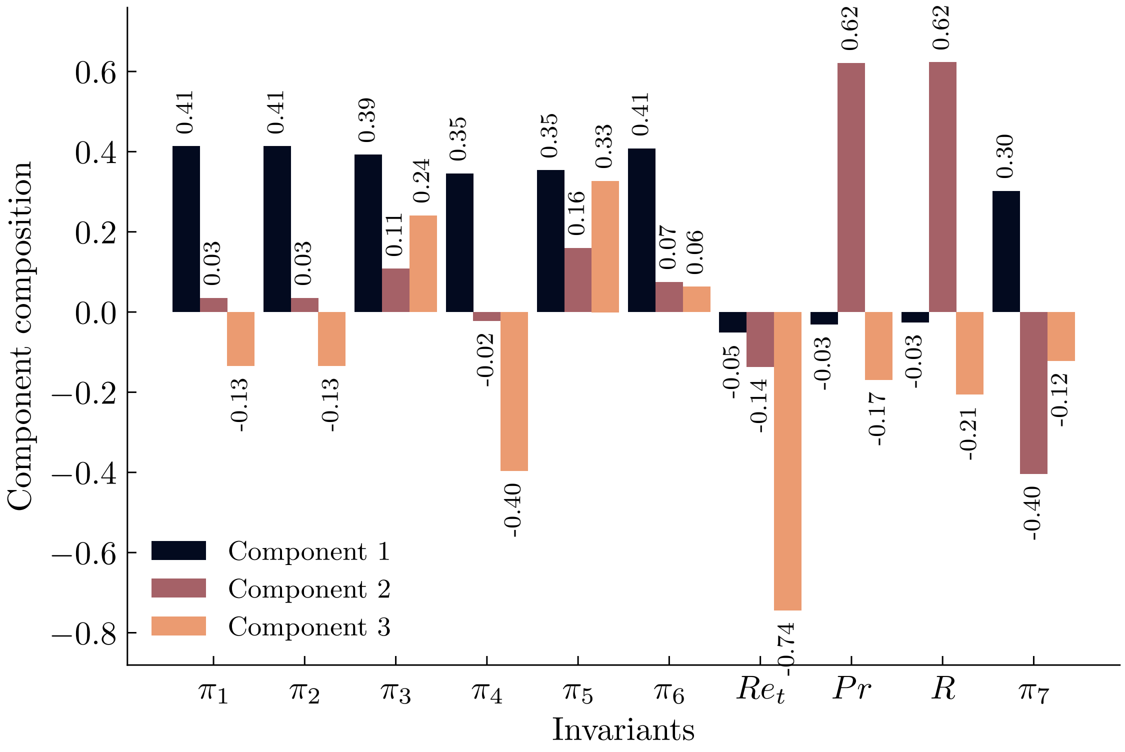
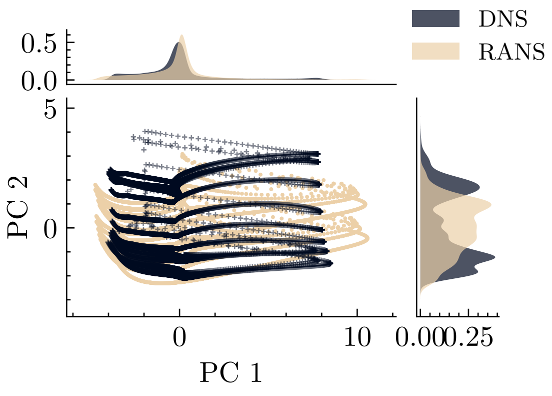
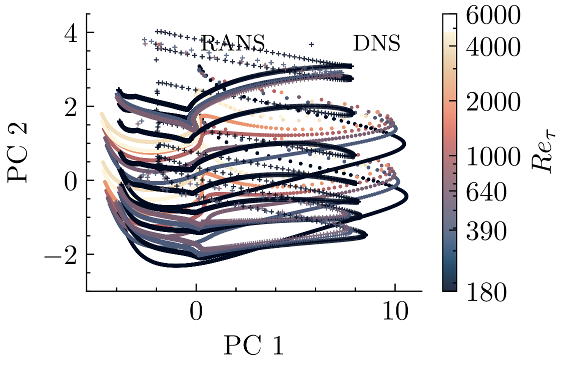


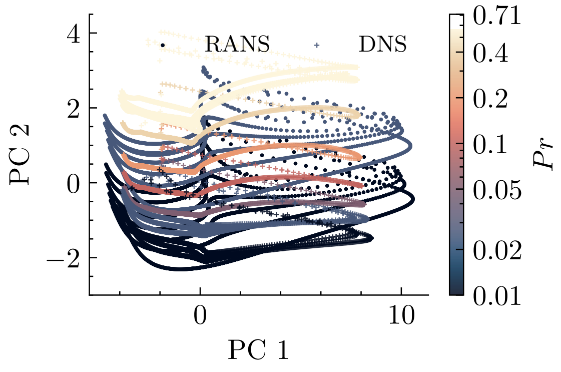
Figure 4.15: First three principal components and their loading in terms of the original variables.



(a)



(b)



(c)

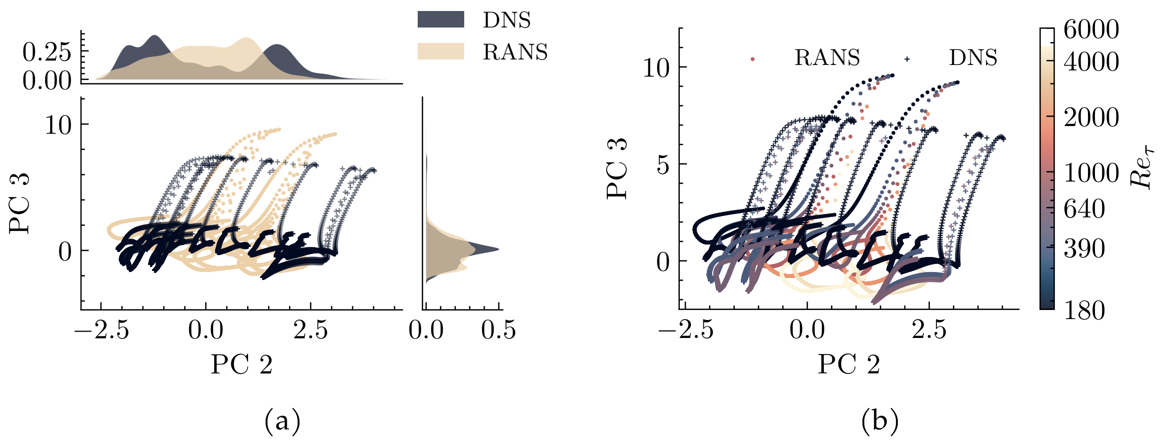
Figure 4.16: RANS/DNS datasets in the PC1-PC2 space. Figure 4.16a on the left also shows the density of the data along the two variables.

Having computed the transformation onto the DNS dataset, it was then applied to the RANS data to evaluate the differences in the Principal Components' space.

In Figure 4.16 the data is projected onto the axis given by PC1 and PC2: along PC1 the RANS data expands the range achieved by the DNS data, which is in part a consequence of the higher *Re* spectrum achieved by the RANS simulations. In fact, as shown in Figure 4.16b, the top end of Re_τ range is mostly on the left, past the minimum values achieved by the DNS data. As seen in Figure 4.16a, the distribution is similar, except for the fact that RANS data has an higher peak, is shifted slightly to higher PC1 values and has a longer tail to the left.

Along PC2 on the other hand, the DNS dataset covers a larger range of values, which is mostly a consequence of the higher range of *Pr* numbers explored with the DNS dataset. This hypothesis is confirmed by Figure 4.16c, in which data is colored by its *Pr* number: here clear "banding" can be observed along PC2, as both RANS and DNS data get ordered along the axis by their Prandtl number. From the same figure it's also clear how the DNS data with higher *Pr* expands the range for positive PC2 values. No clear influence of the Re_τ can be seen on PC2, as justified by the composition shown in

Figure 4.15. PC2's distribution differs between the two fidelity databases, as the DNS data presents two separated peaks while the RANS data shows a flatter distribution with only one larger peak.



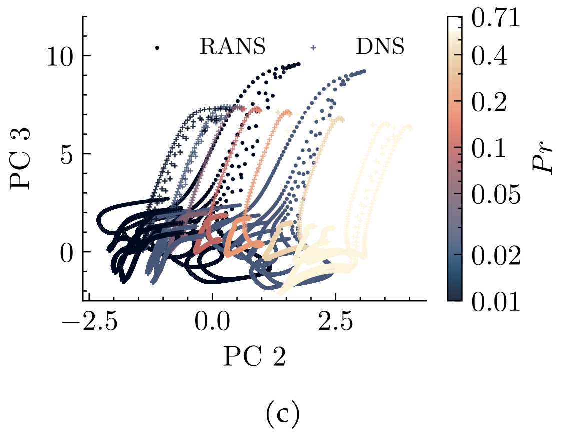


Figure 4.17: RANS/DNS datasets in the PC2-PC3 space. Figure 4.17a on the left also shows the distribution of the data along the two variables.

In Figure 4.17a the data is projected onto PC2-PC3 space. PC3's distribution is fairly similar between the two databases, even tough the RANS' one is less peaky and the valley between the two peaks is much less pronounced.

An interesting observation can be made regarding the relationship between PC3 and Re_τ: as seen from Figure 4.17b, the correlation is clear and inverse, but the main difference (and the range extension by the RANS data) does not come from the top end of the Re_τ range but rather from the bottom (or top range of PC3). While the inverse relation with PC3 is justified by PC3 composition, the reason behind the behavior for high values of PC3 might be related to the increasing inaccuracy in the momentum-related quantities by the RANS modeling which accompanies a decrease in *Re*.

Overall, the PCA confirmed that the most critical combination for RANS modeling is low Reynolds numbers combined with low Prandtl values, mimicking the results given by the analysis of the physical quantities. The PCA has also shown how adding the RANS dataset will expand the training database not only in terms of Re_τ values, but also in terms of model's inputs.

Chapter 5

Multi-fidelity model for the turbulent heat flux

5.1 Introduction

In the following chapter, the development of several data-driven, multi-fidelity models for the turbulent heat flux will be discussed. The mathematical and physical constraints will be explained, as well as the training details.

In the previous chapters all the elements necessary to the creation of a multi-fidelity strategy have been analyzed, such as the single fidelity limitations and the differences between data of different origins. In the creation of the multi-fidelity model, it is of paramount importance to maintain the *physical nature* of the single fidelity model by ensuring that the result is rotational and Galilean invariant and satisfies the already mentioned decomposition to abide by the second law of thermodynamics. At the same time the implementation of a multi-fidelity framework inevitably adds complexity to the model and its training, so several options have been explored and the most promising ones will be explained in what follows.

5.2 Requirements and constraints

In general, unless some hard or soft constraint are imposed, there is no guarantee that a structure like the one shown in chapter 3 is consistent with the physics of the problem, thus the implementation of the multi-fidelity strategy necessitates the addition of some constraints to the multi-fidelity framework. The main constraints are given by:

- *Rotational invariance* with respect to the coordinate system and *Galilean invariance*. In chapter 2 it has been shown how the rotational invariance was enforced in the single fidelity model using the Tensor Representation theory. It is clear that such invariance must be a feature of the multi-fidelity model too, and as such the dispersion tensor cannot undergo any transformation violating this property. This severely limits the way the neural network structure can be formulated, as for example a framework like the one used in chapter 3 could not be used as is, since the low fidelity prediction passes through both linear and non-linear networks.

The Galilean invariance is guaranteed in the same way as it was done for the single fidelity model in chapter 2.

- *Consistency with the second law of thermodynamics*. This is again a central point in keeping the model *physics enforced*, and the same methodology shown in chapter 2 will be used in the following models. This means that any dispersion tensor modeled will be expressed as:

$$\mathbf{D} = \left[\left(\mathbf{A} + \mathbf{A}^T\right)\left(\mathbf{A}^T + \mathbf{A}\right) + \frac{k}{\epsilon^{0.5}}\left(\mathbf{W} - \mathbf{W}^T\right) \right] \tag{5.1}$$

  with $\mathbf{A}$ and $\mathbf{W}$ obtained as combination of the basis tensors $\mathbf{T^i}$ via the neural network modeled coefficients a_i, w_i:

$$\mathbf{A} = \sum_{i=1}^{n} a_i \mathbf{T}^i, \quad \mathbf{W} = \sum_{i=1}^{n} w_i \mathbf{T}^i \tag{5.2}$$

- *Correct physical dimensions*. This requirement deals with the model giving dimensionally consistent output during usage. During the training phase, all the quantities are adimensionalized using the relevant flow values; on the other hand during CFD usage the model takes as inputs whichever form of the data that the solver gives it. To keep dimensional consistency, each of the structures shown in the following will have small expedients applied to the structure itself. These ranges from the standard multiplication/division by k and ε shown in Equation 5.1 to the use of adimensionalized tensors $\mathbf{T^i}$ in the high fidelity networks in model S7.

5.2.1 Loss function

Another difference when compared to the Single Fidelity Neural Network (SFNN) is the wider possibilities on how to compute the loss function. The basic form, already discussed in chapter 2, is:

$$\mathcal{L} = \frac{1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{3}\left(\hat{q}_{i,j} - q_{i,j}\right)^2\right) + \frac{\lambda}{N}\left(\sum_{i=1}^{N}\sum_{j,k=1}^{3}\left|\frac{\partial \hat{q}_{i,j}}{\partial x_k} - \frac{\partial q_{i,j}}{\partial x_k}\right|\Delta x_k\right) \tag{5.3}$$

In a multi-fidelity environment, other terms with the exact same form can be added. For example, given a flow (Re_τ and Pr) for which both DNS and RANS inputs are available, two *difference terms* (or *mixed terms*) can be added measuring the difference in both the low and high fidelity outputs. The idea behind these terms is to drive the optimization towards learning the relationship between RANS and DNS since, due to the earlier discussion about the rotational invariance, the low fidelity prediction cannot pass through the high fidelity network, hindering the model's capability to learn said relationship.

With those terms added, the loss function can have up to 4 terms:

$$\mathcal{L}_{total} = \mathcal{L}_{RANS} + \mathcal{L}_{DNS} + \mathcal{L}_{mix-lf} + \mathcal{L}_{mix-hf} \tag{5.4}$$

where:

- $\mathcal{L}_{RANS}$ measures the error between the RANS result and the low fidelity prediction given the low fidelity input. Note that for this term only the wall-normal component is computed.
- $\mathcal{L}_{DNS}$ measures the error between the DNS result and the high fidelity prediction given the high fidelity input.
- $\mathcal{L}_{mix-lf}$ measures the error between the low fidelity predictions given the RANS and DNS inputs for the same flow condition (if available).
- $\mathcal{L}_{mix-hf}$ measures the error between the high fidelity predictions given the RANS and DNS inputs for the same flow condition (if available).

In the following sections the effect of including the additional terms $\mathcal{L}_{mix-lf}$ and $\mathcal{L}_{mix-hf}$ will be shown, both on the turbulent heat flux prediction and on the training performance.

5.2.2 Training

The optimization algorithm chosen was Adam, as for the original single fidelity model; in this project a weight decay factor of $1 \cdot 10^{-3}$ was added to reduce the possibility of overfitting, as suggested as good practice in (Goodfellow et al., 2016). Since no memory limit was hit, the batch size was chosen to be equal to the entire dataset so that the mixed terms could always be computed.

When the 2D impinging jet data was introduced, only the RANS data could be used for training due to the lack of necessary DNS statistics. The 2D RANS data was interpolated on a 1000×1000 grid, and subpartitions of said grid were fed to the network during the training phase. This was done so that the data at $x/B = 1, ..., 9$ could be removed from the training dataset and kept for the validation. Each model was trained for 20000 total epochs with a scheduled learning rate (more details in the next subsection).

Since the multi-fidelity model comes with an increase in complexity and computing time, as a nice to have the entire model is made GPU-compatible in order to speed up computation. To compare GPU vs. CPU training a sample multi-fidelity model with 14000 neurons was built and run on a 2019 laptop with an Intel(R) Core(TM) i7-9750H CPU with a maximum frequency clock of 2.6GHz and a 4GB NVIDIA GeForce GTX 1650 GPU. With this machine, the GPU training had an average of 0.0526s per epoch, while the CPU training took 0.249s per epoch on average, which translates to around a 78.9% training time decrease.

Having noted the available data characteristics in section 1.3 and wanting to evaluate the performance of the framework when given RANS inputs, the validation loss is computed for DNS cases $Re_\tau = 395$, $Re_\tau = 1020$, $Pr = 0.01$, $Re_\tau = 2000$, $Pr = 0.01$ and $Re_\tau = 1000$, $Pr = 0.02$. This is done in order to maximize the DNS data since the $Re_\tau = 1020, 2000$ could not be used for training. Different trainings including also the $Re_\tau = 395$ flows in the training dataset were also carried out to test the robustness of the model and the effect of the mixed terms; their results match well with the others unless specified otherwise.

5.2.3 Uncertainty quantification

To better interpret the model, the *uncertainty quantification* algorithm "Gaussian Stochastic Weight Algorithm" (Maddox et al., 2019) was implemented into the training and post-processing of the model. The SWAG algorithm was implemented along with a cyclic learning rate as in (Izmailov et al., 2019) to allow the optimizer to explore the parameters' space.

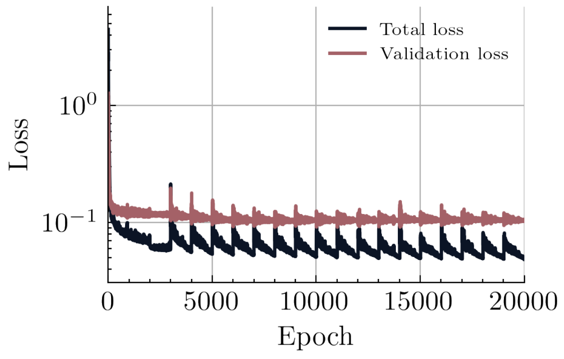


Figure 5.1: Typical training and validation loss functions when coupled with the SWAG algorithm. In this case the training starts with 3000 iterations to get to an initial solution. From there on, the SWAG algorithm engages by setting the learning rate to $1 \cdot 10^{-3}$ and linearly decreasing it to $2.5 \cdot 10^{-4}$ in 1000 epochs. At the end of each cycle (lowest learning rate point), the model's weights are sampled.

With reference to Figure 5.1, the training proceeds with a learning rate η of $5 \cdot 10^{-4}$ without sampling until epoch 2000; η then gets halved reaching an initial solution at epoch number 3000. From there on, the SWAG algorithm engages, by pushing the learning rate to $1 \cdot 10^{-}3$ and then linearly decreasing it back down to $2.5 \cdot 10^{-4}$ in the span of 1000 epochs. At the end of each cycle the network parameters are sampled and the mean and a low-rank diagonal approximation of the covariance are computed. After the training, the model is averaged by sampling 100 times the weights from the distribution, thus allowing to compute both the mean and the standard deviation of the network output. While the mean is used to compare the accuracy of the model when compared to baseline data, the standard deviation and the uncertainty intervals are useful to determine the convergence and stability of the training process.

5.3 GGD sum model (S1)

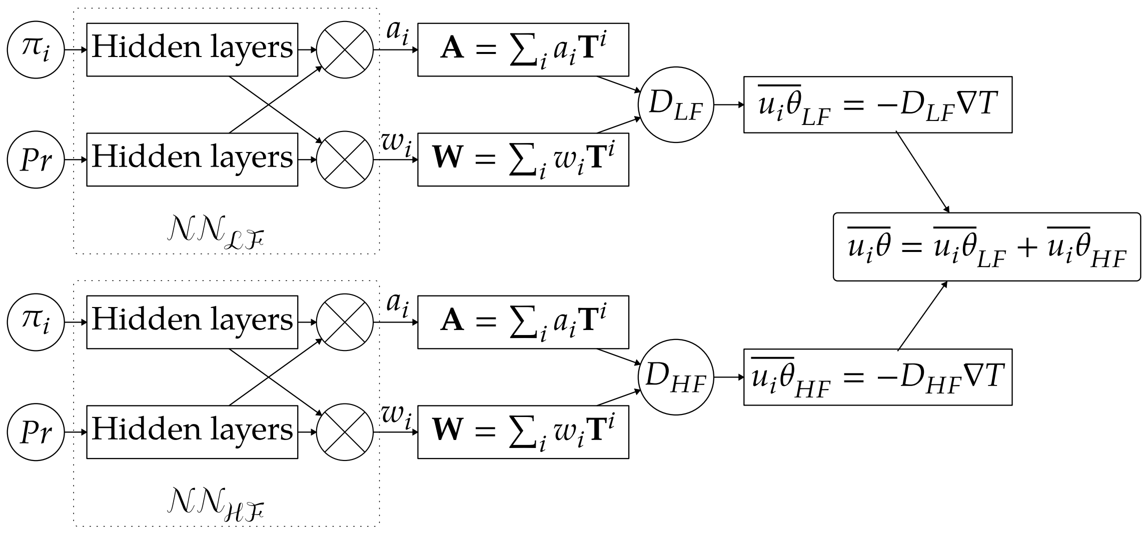

π_i
Hidden layers
Pr
Hidden layers
a_i
w_i
$\mathbf{A} = \sum_i a_i \mathbf{T}^i$
$\mathbf{W} = \sum_i w_i \mathbf{T}^i$
D_{LF}
$\overline{u_i\theta}_{LF} = -D_{LF}\nabla T$
$\mathcal{NN}_{\mathcal{LF}}$
$\overline{u_i\theta} = \overline{u_i\theta}_{LF} + \overline{u_i\theta}_{HF}$
π_i
Hidden layers
Pr
Hidden layers
a_i
w_i
$\mathbf{A} = \sum_i a_i \mathbf{T}^i$
$\mathbf{W} = \sum_i w_i \mathbf{T}^i$
D_{HF}
$\overline{u_i\theta}_{HF} = -D_{HF}\nabla T$
$\mathcal{NN}_{\mathcal{HF}}$

Figure 5.2: Multi Fidelity structure S1. The hidden layers of the "π_i" branches have 6 layers of 100 neurons each, while the "Pr" layers have 2 layers of 50 neurons each. The activation function is ReLU except for the last layer in each branch, that uses a Tanh activation function.

The most immediate implementation of a multi-fidelity model was constructed expressing the turbulent heat flux as a sum of a low and high fidelity terms built using the exact same structure:

$$\overline{u\theta} = -D_{LF}\nabla T - D_{HF}\nabla T \tag{5.5}$$

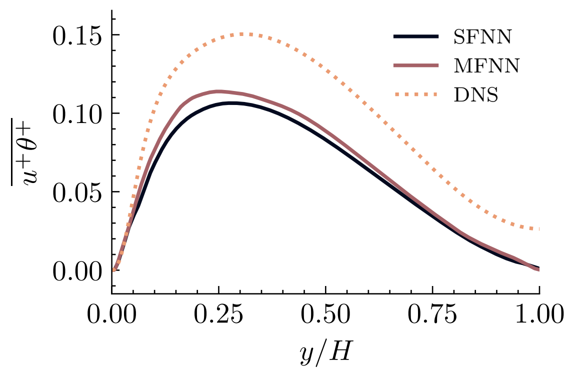
where the two dispersion terms D_{LF} and D_{HF} follow Equation 5.1. This approach was chosen as a first iteration as its implementation is relatively easy, since it is made up of two neural networks (one for each D) identical to the one already used in the single fidelity model. The full structure is shown in Figure 5.2.

At the same time this model adds a layer of complexity to the low fidelity part: it in fact represents the low fidelity contribution as a Generalized Gradient Diffusion term making use of a dispersion term, while the data it trains on is the result of a Simple Gradient Diffusion model, effectively going from a scalar α_t to a full tensor D_{LF}.

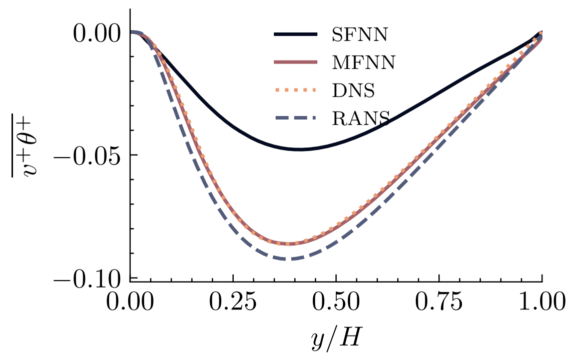
5.3.1 Results

Starting from the lower end of the Reynolds' range a first difference with the single fidelity model can be observed.

In panel 5.3 results for the (training) data point $Re_\tau = 180,\ Pr = 0.025$ can be seen for both the SFNN and the MFNN (S1). In Figure 5.3a the streamwise component is shown. As expected, the difference between the two training modes is minimal, since the RANS data adds no information about the streamwise component; both models underestimate the heat-flux when given RANS inputs.



(a) Streamwise $\overline{u\theta}$ component.



(b) Wall-normal $\overline{v\theta}$ component.

Figure 5.3: S1 turbulent heat flux prediction for the $Re_\tau = 180$, $Pr = 0.025$ channel flow for RANS inputs. Full loss function used in training.

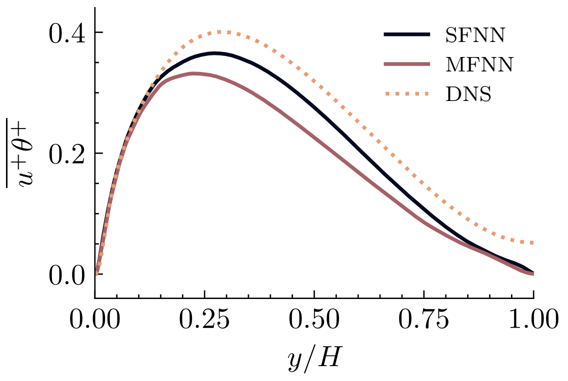
In Figure 5.3b the wall-normal component is represented, and a big difference is noted between the two models. In this case the multi-fidelity model is able to accurately track the DNS result, much improving the single fidelity model when given the same input data. This result however, must be taken with extreme caution: this is in fact one of the training points and, on top of that, the difference between the RANS and DNS is minimal. Nevertheless, the improvement over the SFNN model is still clear, as this was a training flow for the SFNN too.

Moving onto the $Re_\tau = 395$ cases shown in Figure 5.4 and 5.5, once again one can note the very little difference when comparing the two models over the streamwise component. On the right side of the figures though, the wall-normal component shows how the MFNN is basically trying to fit the RANS data, with no regard for the high fidelity data. Note that those results are from a training process in which the $Re_\tau = 395$ flows are excluded from the training dataset.

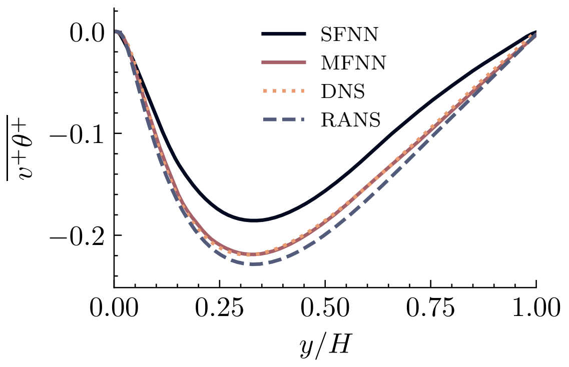
As expected, for the cases in which both fidelity levels are available as training data, the addition of mixed terms to the loss function helps a little in driving the MFNN prediction closer to the DNS data. To show this, the same model was trained using a loss function made of only the first two terms in Equation 5.4, so that the low fidelity neural network would have no direct information about the DNS data.

The results of the two different training methods are shown in Figure 5.6, where the flow $Re_\tau = 180$, $Pr = 0.01$ is computed. In Figure 5.6a, no mixed terms were considered. In this case, it's clear how the MFNN resorts to fitting the RANS data. In Figure 5.6b, the full loss function was used; the effect of the mixed terms is clear in that it closes the gap between the high fidelity data and the model's prediction.

While adding the mixed terms did not significantly influence the model's ability to predict the wall-normal component outside of the overlapping DNS/RANS cases, the one major effect it had was moving the MFNN streamwise component closer and closer to that predicted by the SFNN. This is a slight improvement in accuracy as in most cases the SFNN was more accurate when predicting $\overline{v\theta}$, with the caveat that past

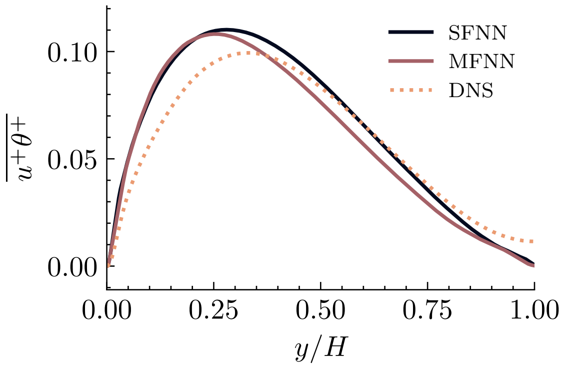


(a) Streamwise $\overline{u\theta}$ component.

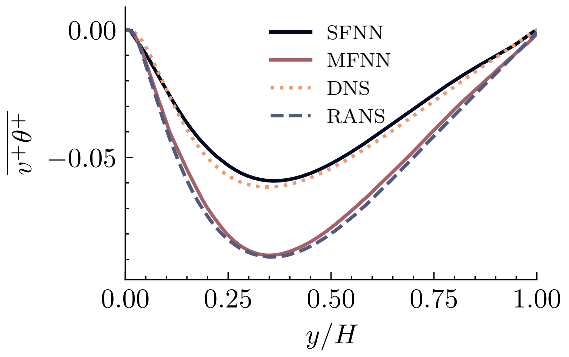


(b) Wall-normal $\overline{v\theta}$ component.

Figure 5.4: S1 turbulent heat flux prediction for the $Re_\tau = 395,\ Pr = 0.025$ channel flows for RANS inputs.



(a) Streamwise $\overline{u\theta}$ component.



(b) Wall-normal $\overline{v\theta}$ component.

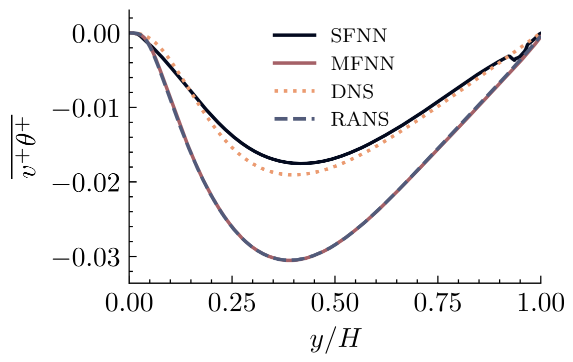
Figure 5.5: S1 turbulent heat flux prediction for the $Re_\tau = 395,\ Pr = 0.01$ channel flows for RANS inputs. Full loss function used in training.

$Re_\tau = 2000$ no data is available at all, making it impossible to express a fair judgment.

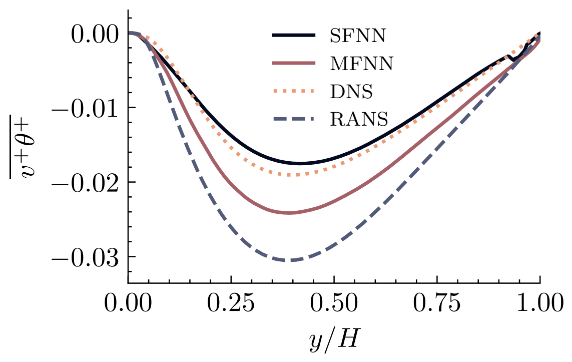
An example is shown in Figure 5.7, where the channel flow with $Re_\tau = 1020,\ Pr = 0.01$ is shown. In this case, whether the mixed terms are considered or not, the difference in the $\overline{v\theta}$ prediction is small and negligible. In the bottom row (Figure 5.7c and 5.7d) the streamwise component is shown, and the effect of the mixed terms is visible. The same results are true for greater Reynolds numbers, which is furthermore justified by the absence of DNS data to begin with.

The final comparison is done for flows outside of the original single fidelity training range, for which $Re_\tau \geq 2000$. Since no major differences are noted when varying the Prandtl number, only results for $Pr = 0.01$ will be shown in the following.

Starting from the flow case $Re_\tau = 2000,\ Pr = 0.01$, shown in Figure 5.8, a good improvement can be seen in the wall-normal component prediction when compared to the single fidelity model. Even at this relatively low Re_τ value, it is possible to notice how the multi-fidelity model avoids the bump in which the single fidelity model incurs.



(a) No mixed terms used during training.



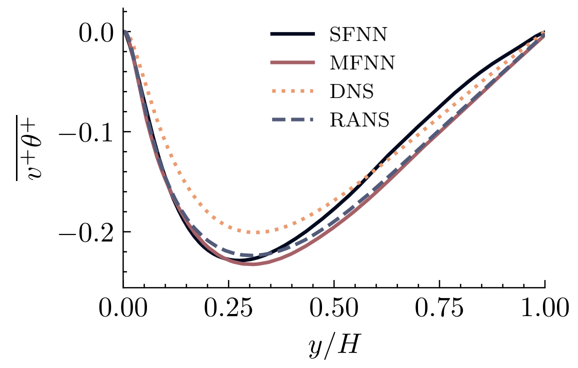
(b) Mixed terms used during training.

Figure 5.6: Effect of the loss function terms on S1's turbulent heat flux prediction for the $Re_\tau = 180,\ Pr = 0.01$.

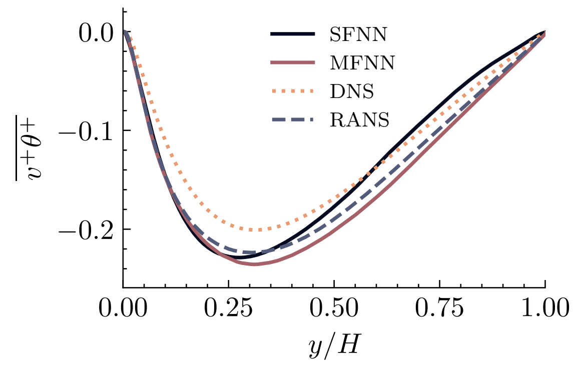
As will be noted in the following, this is a trend that the multi-fidelity model maintains even in the top end of the Re_τ spectrum.

On the other hand, the streamwise component suffers and underestimates slightly the turbulent heat flux along the whole channel height in the MFNN model; as already noted no increase in accuracy is expected anyway in said component since the RANS data has no $\overline{u\theta}$ component.

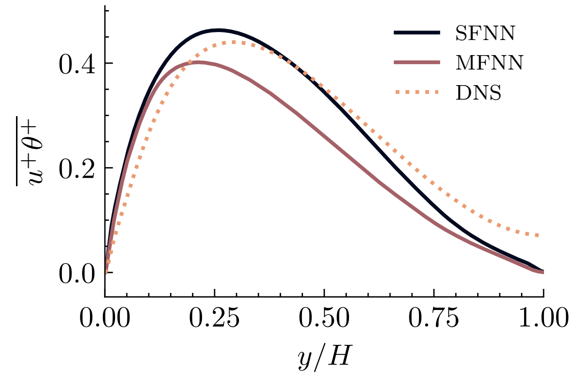
Results for flow conditions outside of the DNS database are shown in Figure 5.9. In light of the discussion in section 4.4 and chapter 1, for higher Reynolds number the RANS data is assumed to be increasingly correct and thus used as a baseline for qualitative comparison. Looking at the left of panel 5.9, the multi-fidelity model appears to reduce the chronic overestimation of the single fidelity model, closely matching the RANS data. This does not apply to the streamwise component (right side of the panel), for which however no quantitative or qualitative data is available and thus no definitive conclusion can be made.



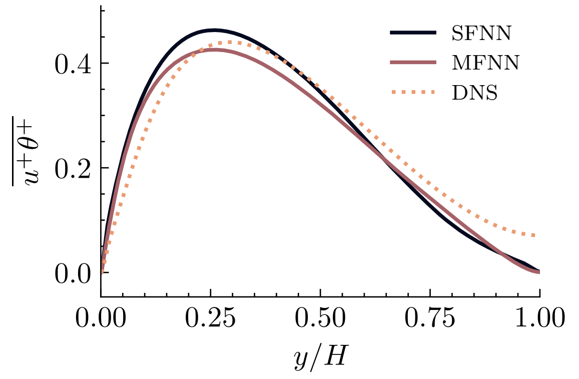
(a) $\overline{v\theta}$ prediction using the complete loss function.



(b) $\overline{v\theta}$ prediction using no mixed terms in the loss function.

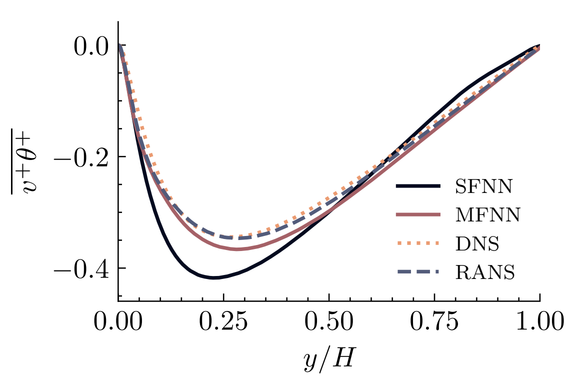


(c) $\overline{u\theta}$ prediction using the complete loss function.

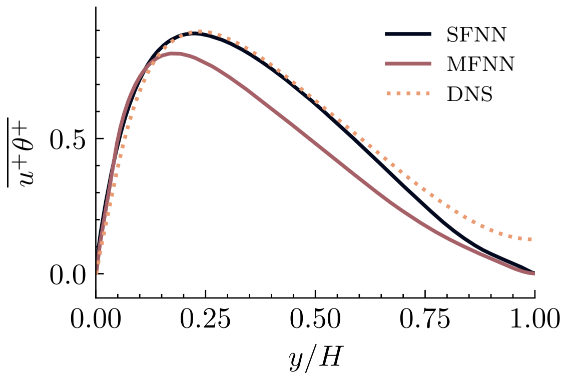


(d) $\overline{u\theta}$ prediction using no mixed terms in the loss function.

Figure 5.7: Effect of added loss function terms at $Re_\tau = 1020$, $Pr = 0.01$. This is a validation case and whether the mixed terms are considered or not does not make a noticeable difference on the wall-normal component prediction.

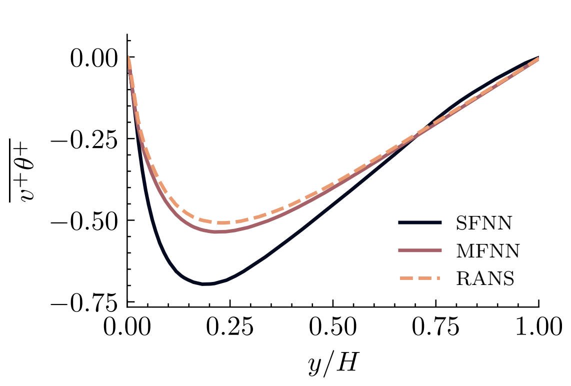


(a) Wall normal component $\overline{v\theta}$.

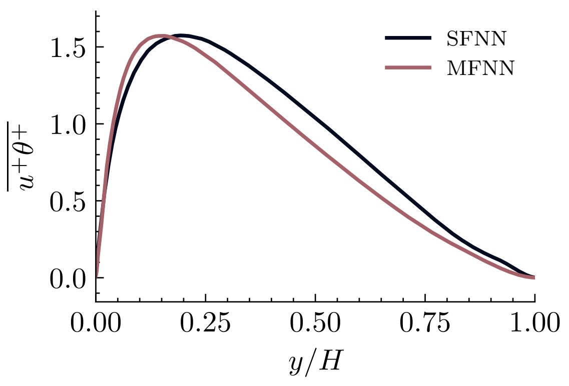


(b) Streamwise component $\overline{u\theta}$.

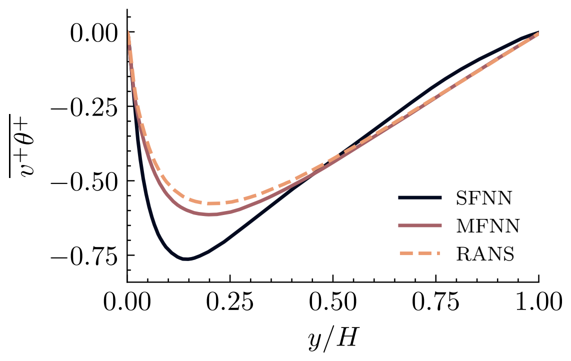
Figure 5.8: Comparison of turbulent heat flux prediction between Single Fidelity Neural Network (SFNN) and Multi Fidelity Neural Network (MFNN-S1) for $Re_\tau = 2000$, $Pr = 0.01$. Note that this is the highest Re_τ for which DNS data is available. The full loss function was used in this case.



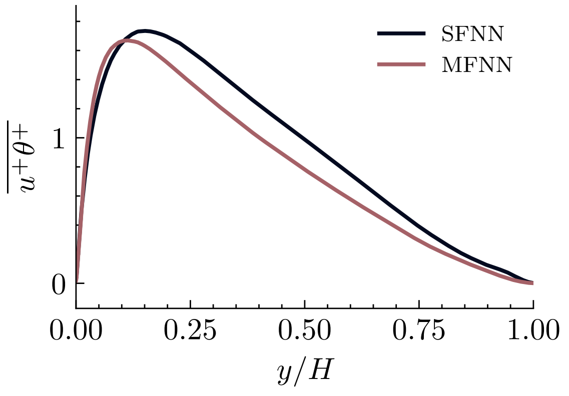
(a) Wall normal component $\overline{v\theta}$. Re_τ = 4000, $Pr = 0.01$



(b) Streamwise component $\overline{u\theta}$. Re_τ = 4000, $Pr = 0.01$



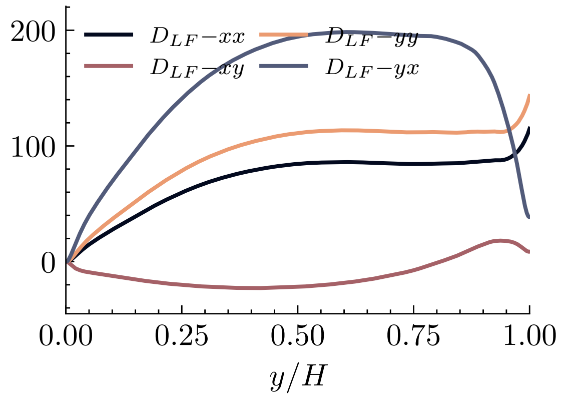
(c) Wall normal component $\overline{v\theta}$. Re_τ = 6000, $Pr = 0.01$



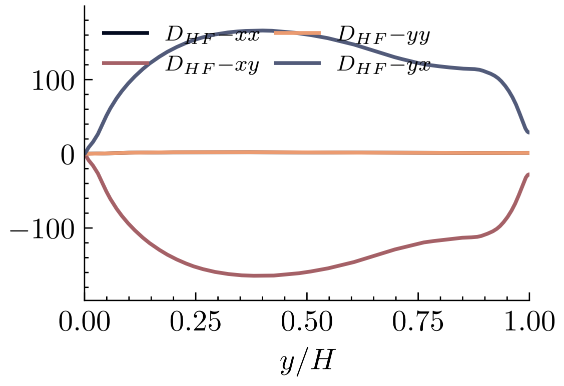
(d) Streamwise component $\overline{u\theta}$. Re_τ = 6000, $Pr = 0.01$

Figure 5.9: Comparison of turbulent heat flux prediction between Single Fidelity Neural Network (SFNN) and Multi Fidelity Neural Network (MFNN-S1) for higher Reynolds numbers. For the selected Re_τ no DNS data is available.

Further insight into the model can be gained by looking at the dispersion tensors D_{LF} and D_{HF}.



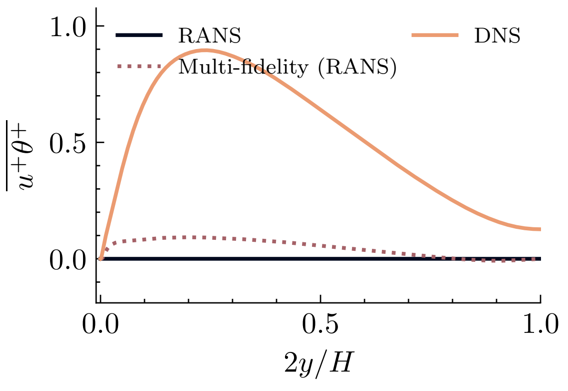
(a) D_{LF} for the $Re_\tau = 2000,\ Pr = 0.01$ case.



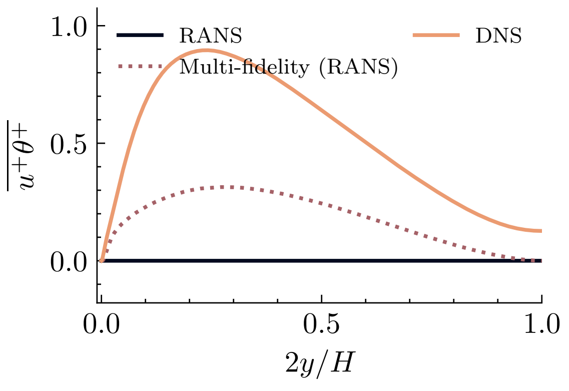
(b) D_{HF} for the $Re_\tau = 2000,\ Pr = 0.01$ case.

Figure 5.10: D_{LF} (left) and D_{HF} (right) components for the $Re_\tau = 2000,\ Pr = 0.01$ case. The full loss function is used.

In Figure 5.10a the low fidelity dispersion tensor D_{LF} is shown for the $Re_\tau = 2000,\ Pr = 0.01$ flow. It can be noted that with this model the optimization process drives towards fully building the low fidelity tensor D_{LF} to resemble the single fidelity tensor (shown in Fiore, Koloszar, Fare, et al., 2022), while enforcing the D_{HF} tensor to be skew-symmetric with small values on the main diagonal. This pattern is persistent regardless of the chosen Reynolds and Prandtl numbers and whether validation or training data are used.



(a) Full loss function.



(b) Loss function with only the direct terms.

Figure 5.11: Low fidelity streamwise turbulent heat flux component prediction with the two different loss functions. $Re_\tau = 2000,\ Pr = 0.01$.

In terms of turbulent heat flux prediction, this D_{LF} structure means that the low fidelity part of the network still tries to output a streamwise component $\overline{u\theta}$, even though it has no direct information from the RANS data, only from the mixed loss terms. The result is an inaccurate prediction of the streamwise component, shown for both loss functions in Figure 5.11. In both cases the low fidelity outputs something resembling to the streamwise component but with no accuracy either qualitatively or quantitatively.

The effect of the mixed loss terms can also be looked at from the training convergence point of view via the tools provided by the SWAG algorithm.

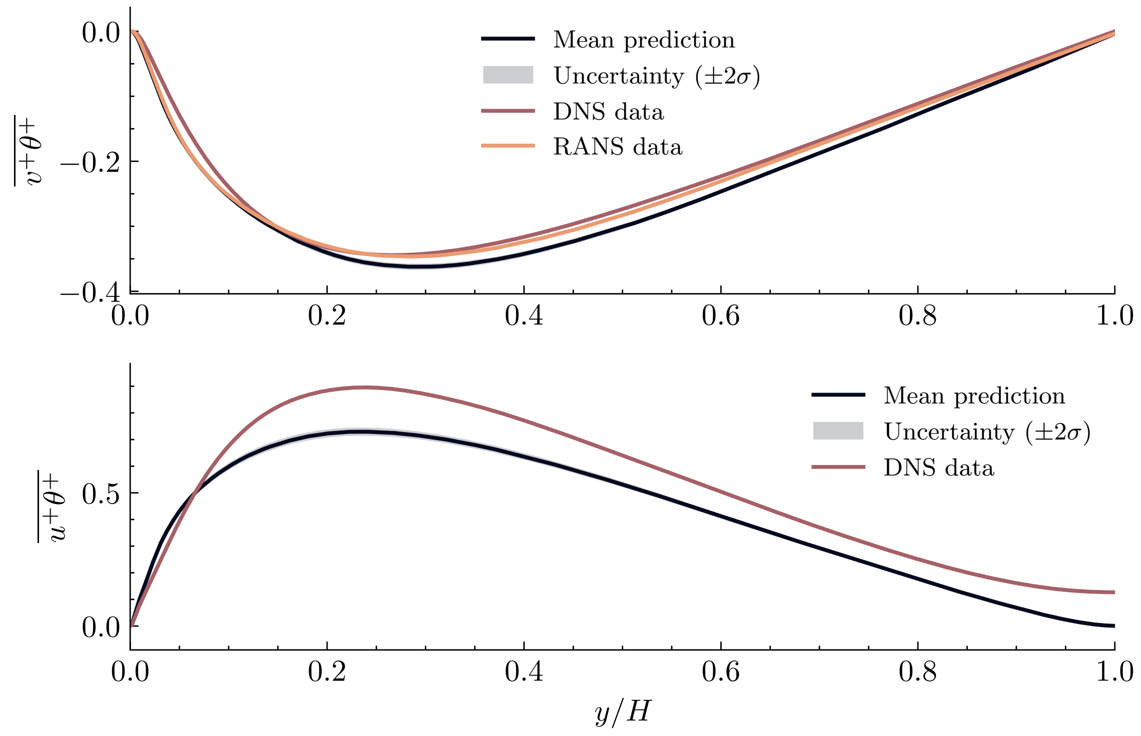
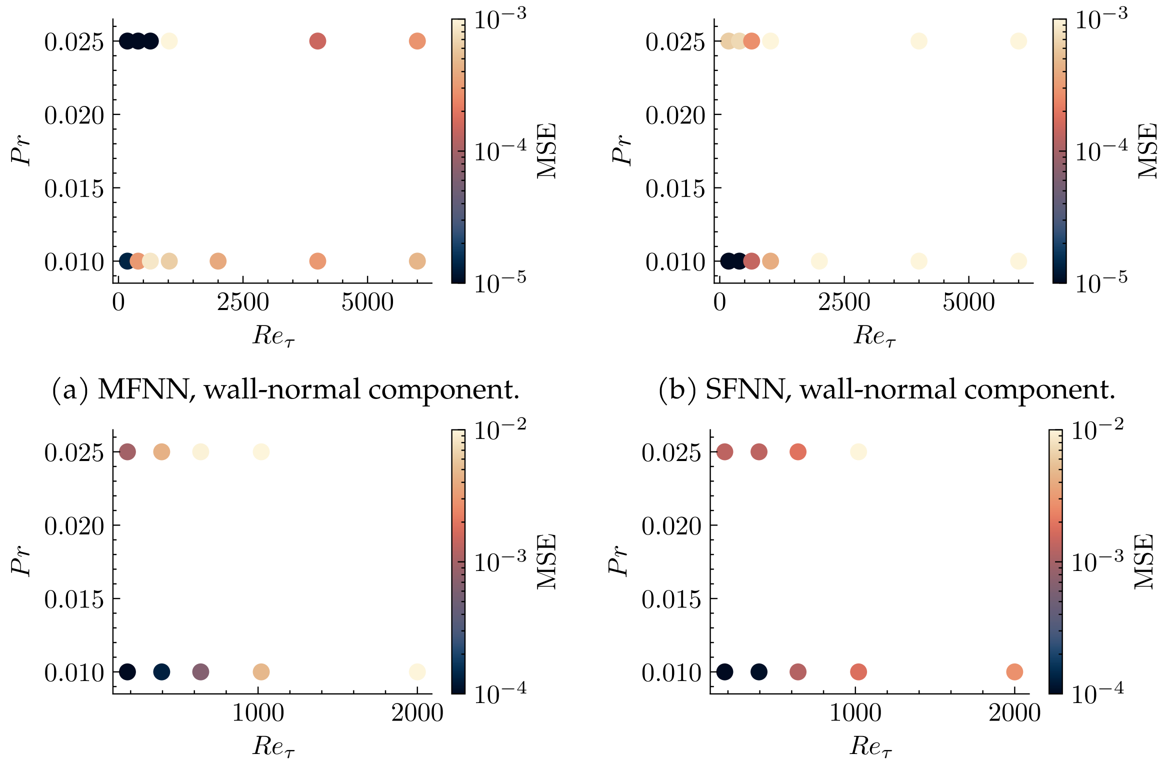


Figure 5.12: Uncertainty quantification for the loss function without the mixed terms. $Re_\tau = 2000$, $Pr = 0.01$ flow.

In figure 5.12 the results of the SWAG algorithm for the $Re_\tau = 2000$, $Pr = 0.01$ flow are shown. The training proved to be very stable and precise, achieving small standard deviation with the sampled models. Also note that the inclusion of the mixed terms into the loss function did not alter in any way the stability of the training, and the uncertainty intervals turned out to be virtually indistinguishable from those in Figure 5.12 and will therefore not be reported.

Final remarks

A plot of the mean squared error for all the cases available is shown in Figure 5.13. To produce those results, the MSE was computed using DNS data as ground truth where available and RANS data was used elsewhere. These plots show how, with the exception of a couple of low-Reynolds and low-Prandtl cases, the MFNN model



(a) MFNN, wall-normal component. (b) SFNN, wall-normal component.

(c) MFNN, streamwise component. (d) SFNN, streamwise component.

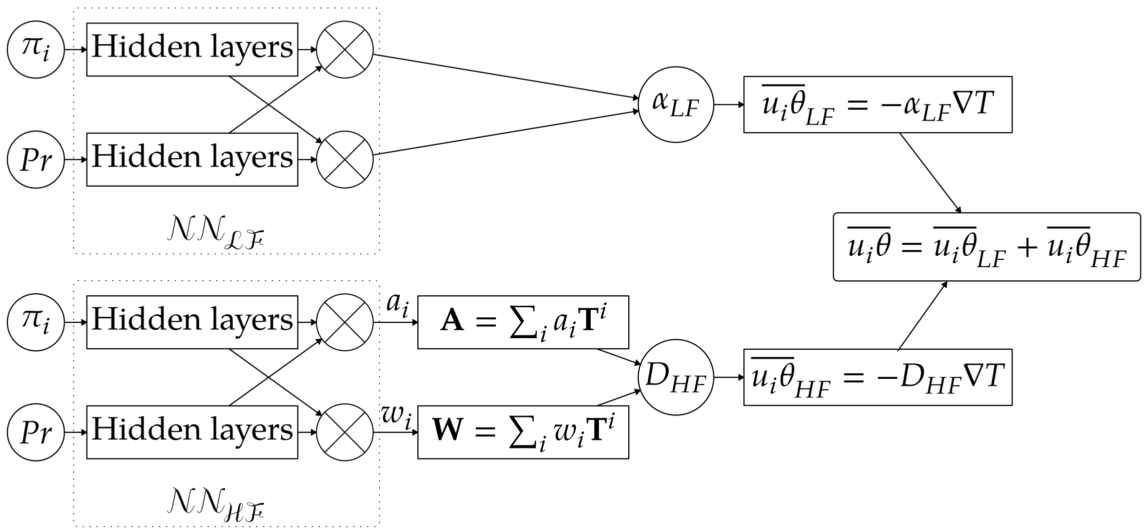
Figure 5.13: Mean squared error comparison between the SFNN and the MFNN (S1). To compute the MSE, DNS data was used as ground truth where available, RANS data was used elsewhere.

achieves lower error in the wall normal prediction, especially at higher Re_τ values. It's worth noting that the model performs worse where the RANS prediction is considerably far from that of DNS.
The MFNN model instead performs slightly worse than the single fidelity when the streamwise component is considered.

Overall, the model's first iteration shows the promising ability to follow trends indicated by RANS data for regions where no DNS data is available. No improvement was made on the prediction of the streamwise component, but this was somewhat expected as the low fidelity data added no information on that front.

The main drawback is that the model is almost exactly replicating the RANS data for the $\overline{v\theta}$ component, instead of trying to mimic the DNS values. This is partly attributed to the fact that, when compared to the original multi-fidelity framework presented in chapter 3, this multi-fidelity model had to be modified to enforce the mentioned first principles into the model. This meant altering the way in which the low fidelity prediction and high fidelity networks interact, as explained in section 5.2. While this is an open problem, what can be improved is the model interpretability: an avenue to do so is the *active* strategy of altering the network structure.

5.4 SGD + GGD additive model (S5)



π_i
Hidden layers
Pr
Hidden layers
$\mathcal{NN}_{\mathcal{LF}}$
α_{LF}
$\overline{u_i\theta}_{LF} = -\alpha_{LF}\nabla T$
$\overline{u_i\theta} = \overline{u_i\theta}_{LF} + \overline{u_i\theta}_{HF}$
π_i
Hidden layers
Pr
Hidden layers
a_i
w_i
$\mathbf{A} = \sum_i a_i \mathbf{T}^i$
$\mathbf{W} = \sum_i w_i \mathbf{T}^i$
D_{HF}
$\overline{u_i\theta}_{HF} = -D_{HF}\nabla T$
$\mathcal{NN}_{\mathcal{HF}}$

Figure 5.14: Multi Fidelity structure S5. The hidden layers of the "π_i" branches have 6 layers of 100 neurons each, while the "Pr" layers have 2 layers of 50 neurons each. The activation function is ReLU except for the last layer in each branch, that uses a Tanh activation function.

From the results given by the first structure tested, reported at the end of section 5.3, the idea of a simpler model was originated. The main area of improvement was deemed to be the low fidelity network, since in the first iteration it gave a full dispersion tensor D_{LF} as output even though the low fidelity data were results from Simple Gradient Diffusion models. To simplify the model and make it more consistent with its training data, the natural step is to modify the output given by $\mathcal{NN}_{\mathcal{LF}}$ and make it a scalar α_{LF} instead of a full tensor. The final turbulent heat flux prediction then becomes:

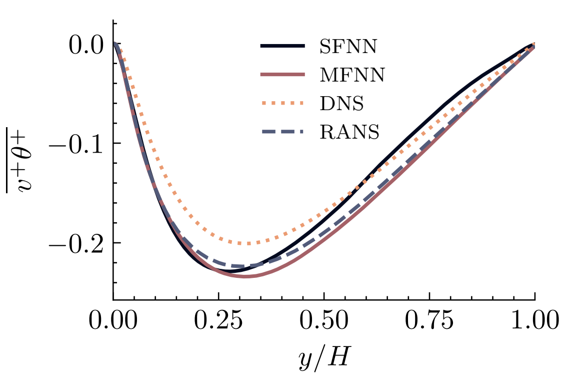
$$\overline{u\theta} = -\alpha_{LF}\nabla T - D_{HF}\nabla T \tag{5.6}$$

The resulting neural network is schematically shown in Figure 5.14; all of the hyperparameters related to network structure and training are unchanged.

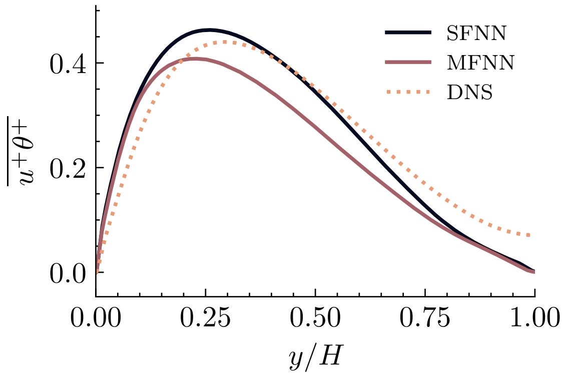
5.4.1 Results

Most of the results obtained with the S1 model regarding the high fidelity prediction are valid for the S5 structure too, meaning it is also capable of reproducing trends learned from the RANS data at higher Reynolds. Even with this simplified low fidelity network, the model still shows the tendency to disregard the high fidelity data for what concerns the wall normal component.

An example of such behavior is shown in Figure 5.15a, where the wall-normal $\overline{v\theta}$ prediction is plotted; the prediction is very similar to the one made by (S1) model,



(a) Wall-normal $\overline{v\theta}$ component.



(b) Streamwise $\overline{u\theta}$ component.

Figure 5.15: S5 model's performance for $Re_\tau = 1020,\ Pr = 0.01$.

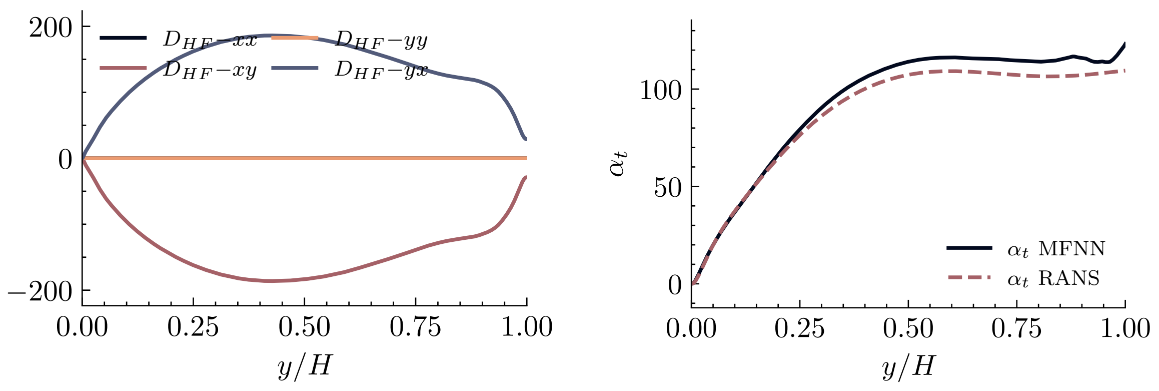
pictured in Figure 5.7a. In both cases the MFNN model seems to follow the RANS data more than the DNS. Even with this structure, the streamwise component (shown in Figure 5.15b) does not really improve in terms of quality over the single fidelity prediction.

The improvement that this particular structure allows is more towards model interpretability, which is a key area in machine learning research. According to the classification in (Y. Zhang et al., 2021), changing the network architecture is an *active* approach in the first dimension towards network interpretability.

There is now a clearer definition of what the low fidelity is: much like the original framework explored in chapter 3, with the S5 structure the low fidelity network outputs is closer to the low-fidelity data than before. As noted in subsubsection 5.3.1, with the S1 structure, the optimizer tends to build the low fidelity dispersion tensor D_{LF} as to represent all the components of $\overline{u\theta}$. This leads to an unrealistic and non-physically related low-fidelity streamwise component.

In Figure 5.16 the outputs of the two neural networks of the S5 model are shown. With this structure, the optimizer uses the low fidelity network to predict the wall-normal turbulent heat flux via the α_t value, while shaping the D_{HF} tensor to model the streamwise component. For the flow case $Re_\tau = 180,\ Pr = 0.01$, a very similar phenomenon to what happens with the S1 model is noted: during training, the optimizer "feeds back" high fidelity information onto the low fidelity network. This is clear by the fact that the entire prediction of $\overline{v\theta}$ is assigned to the low fidelity network, leading to a different α_t value than that computed via the RANS method; this phenomenon is particularly clear for flows in which the corresponding DNS data is very different from the RANS data.

A clear example of this "feedback" is shown in Figure 5.19, where both α_{LF} (right) and the high fidelity $\overline{v\theta}$ (left) are shown. For this flow case, one can note how the low fidelity network underestimates α_t (with respect to the RANS value), in an attempt to match the DNS $\overline{v\theta}$. This behavior is different from the simple test cases in chapter 3, where the low fidelity network accurately predicted the low fidelity data; at the same



(a) High fidelity tensor D_{HF}.

(b) α_t prediction comparison with the α_t in the RANS simulation.

Figure 5.16: S5 model's output for $Re_\tau = 2000$, $Pr = 0.01$. α_t MFNN on the right is what in Figure 5.14 was α_{LF}

time it is not necessarily a negative one: due to the enforcement of first principles, the exact linear and non-linear correction could not be learned in the same way as in the original multi-fidelity framework.

The "feedback" phenomenon goes away when mixed terms are not added to the loss function. This is clear when looking at Figure 5.20, where results are shown for $Re_\tau = 180$, $Pr = 0.01$ (even though the same considerations are valid for all the other flow cases). In this case the optimizer finds the best solution for $\mathcal{NN}_{\mathcal{LF}}$ to fit the low fidelity data, while being incapable of correcting it with the high fidelity contribution. The result is in essence a model that always outputs a RANS-like wall normal component without learning from high fidelity data, and then adds a streamwise component which is somewhat comparable in accuracy to the single fidelity model.

Adding or not the mixed terms to the loss function does not alter the training stability in the S5 structure, as was found out when analyzing the S1 model. To prove this assessment, the same data shown in Figure 5.12 was computed for the S5 model and reported in Figure 5.17. Once again very small standard deviation is achieved when sampling the model, and this is shown by the narrow uncertainty intervals.

Overall the S5 structure improves some aspects of the S1 structure, mainly the model's interpretability, while still suffering from some of the same deficiencies, in particular the partial inability to correct the low fidelity prediction using higher fidelity data. Part of this is attributed to the already mentioned impossibility of completely replicating the framework proposed in (Meng & Karniadakis, 2020).
The omission of the mixed terms guides the model into predicting near-RANS values for the wall normal component, and then adding a streamwise component similar to that predicted by the SFNN. Adding those terms instead forces the model into a more accurate prediction of the turbulent heat flux for the overlapping DNS/RANS cases, but has little effect otherwise. A difference between the S1 and S5 models is that, in the case of the S5 model, adding or not the mixed terms does not lead to a noticeably

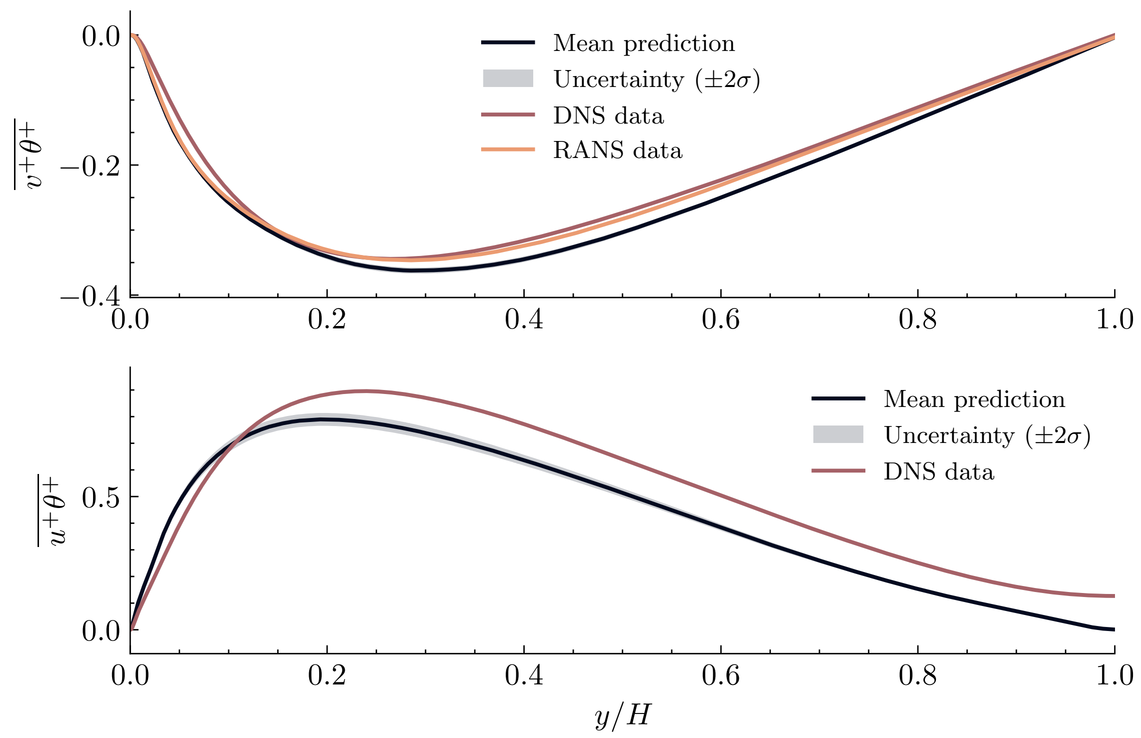
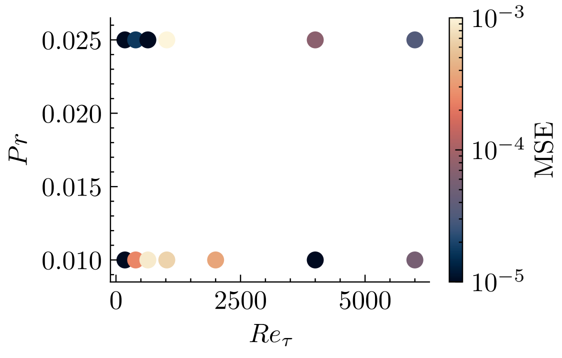
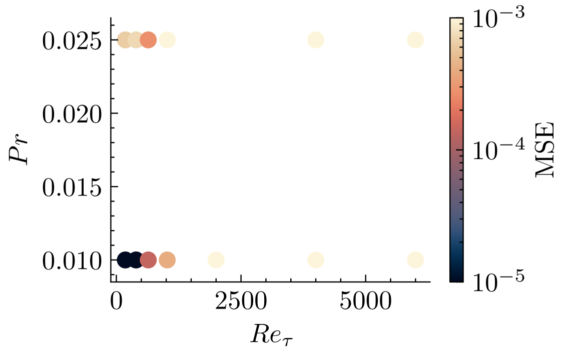


Figure 5.17: S5 model's uncertainty quantification for the loss function with no mixed terms. $Re_\tau = 2000,\ Pr = 0.01$ flow.

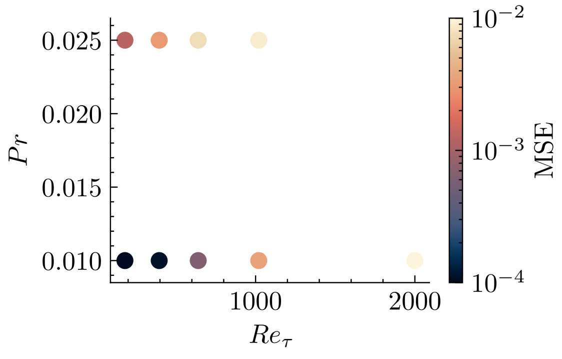
different $\overline{u\theta}$ prediction.



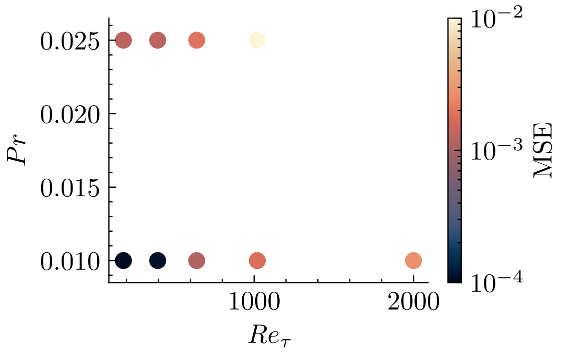
(a) MFNN, wall-normal component.



(b) SFNN, wall-normal component.

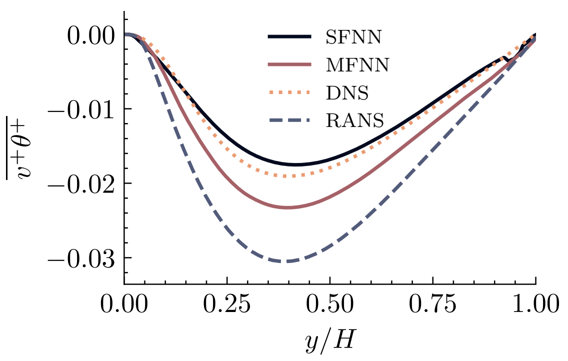


(c) MFNN, streamwise component.

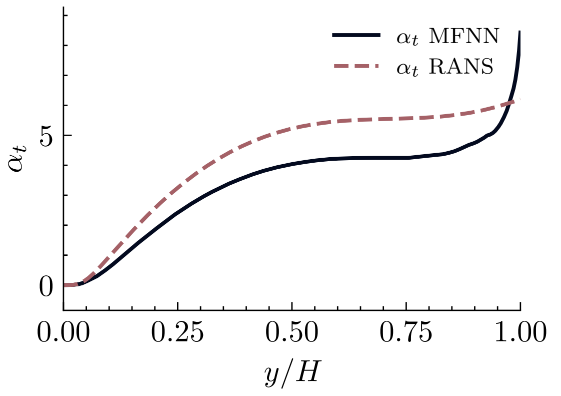


(d) SFNN, streamwise component.

Figure 5.18: Mean squared error comparison between the SFNN and the MFNN (S5). To compute the MSE, DNS data was used as ground truth where available, RANS elsewhere.

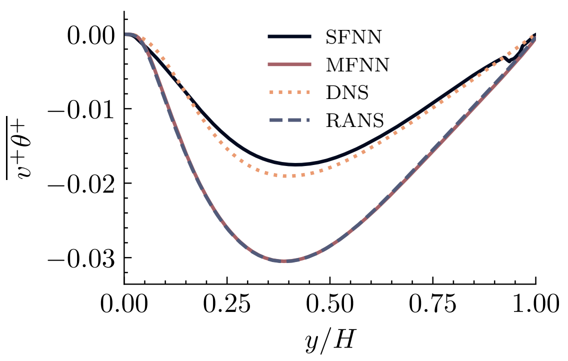


(a) Wall-normal $\overline{u\theta}$ component.

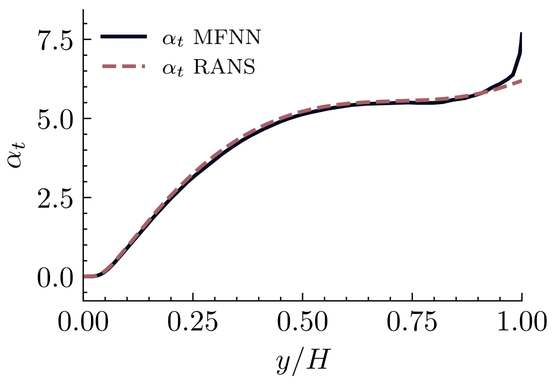


(b) α_t prediction comparison with the α_t in the RANS simulation.

Figure 5.19: S5 "feedback" phenomenon for $Re_\tau = 180,\ Pr = 0.01$. Notice how the model "adjusts" the low-fidelity prediction not to match the low fidelity data but to best match the high fidelity one.



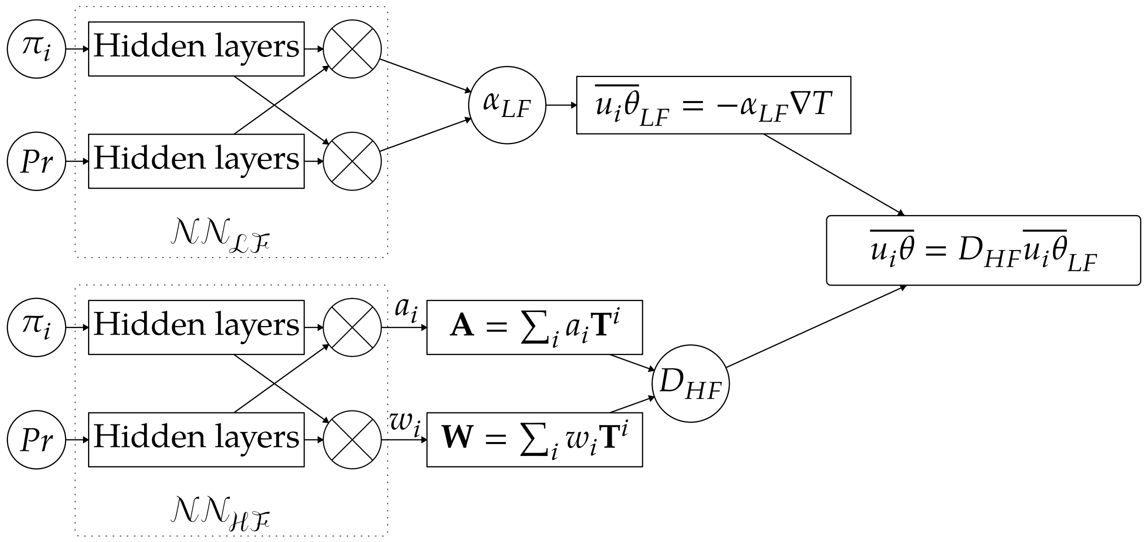
(a) Wall-normal turbulent heat flux prediction by the S5 model with no mixed terms in the loss function. $Re_\tau = 180,\ Pr = 0.01$.



(b) α_t prediction comparison with the $alpha_t$ in the RANS simulation.

Figure 5.20: Absence of "feedback" phenomenon for $Re_\tau = 180,\ Pr = 0.01$ in the S5 model. Notice how the model completely disregards the high fidelity data. These results are obtained using no mixed terms in the loss function.

5.5 SGD + GGD multiplicative model (S7)



π_i
Hidden layers
Pr
Hidden layers
$\mathcal{NN}_{\mathcal{LF}}$
α_{LF}
$\overline{u_i\theta}_{LF} = -\alpha_{LF}\nabla T$
$\overline{u_i\theta} = D_{HF}\overline{u_i\theta}_{LF}$
π_i
Hidden layers
Pr
Hidden layers
a_i
$\mathbf{A} = \sum_i a_i \mathbf{T}^i$
w_i
$\mathbf{W} = \sum_i w_i \mathbf{T}^i$
D_{HF}
$\mathcal{NN}_{\mathcal{HF}}$

Figure 5.21: Multi Fidelity structure S7. The hidden layers of the "π_i" branches have 6 layers of 100 neurons each, while the "Pr" layers have 2 layers of 50 neurons each. The activation function is ReLU except for the last layer in each branch, that uses a Tanh activation function.

The last model presented has the same general ideas of the S5 structure, except that in this case the final prediction is given by the product between the high fidelity tensor D_{HF} and the low fidelity prediction:

$$\overline{u\theta} = D_{HF}\overline{u\theta}_{LF} \tag{5.7}$$

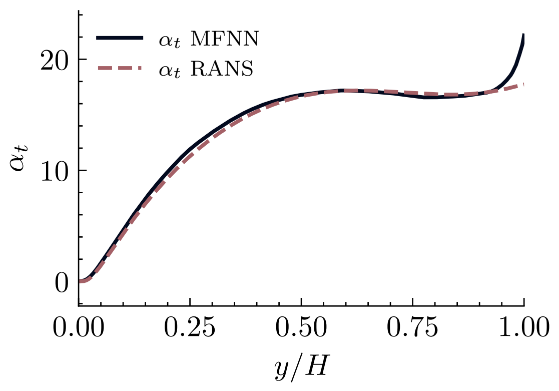
To achieve dimensional consistency the D_{HF} tensor is made made adimensional by using adimensional tensors T^i in its definition.

5.5.1 Results

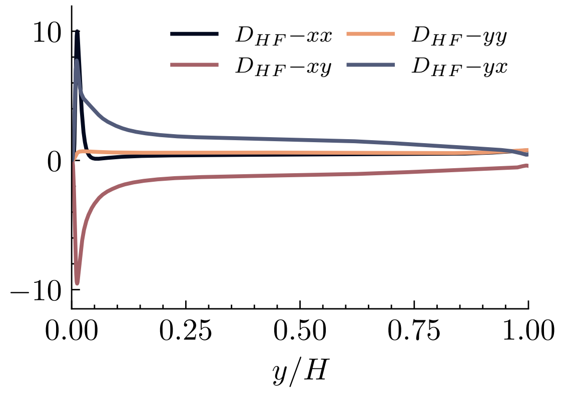
The S7 structure achieves radically different results from those of S1 and S5. In fact, when trained without the mixed terms, the model correctly predicts the low fidelity $\overline{v\theta}$ while still being able to apply corrections to it via the high fidelity network.

This is shown in Figure 5.22, where both the low fidelity (in the form of α_t) and the high fidelity (in the form of D_{HF}) prediction are shown. As opposed to the results of subsection 5.4.1, this model is able to both achieve a good low fidelity prediction of α_t, while still being able to correct it using D_{HF} to best match the high fidelity data. Moreover, this results is carried over even in the validation range, as demonstrated by Figure 5.23b.

The promising characteristic of this structure is that the ability to correct low fidelity data at low Reynolds, does not take away the fact that the model is still capable of

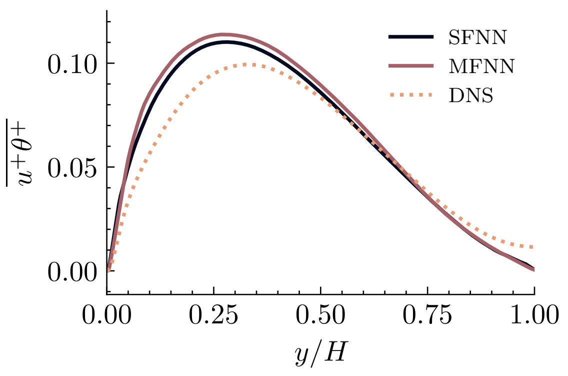


(a) α_t prediction.

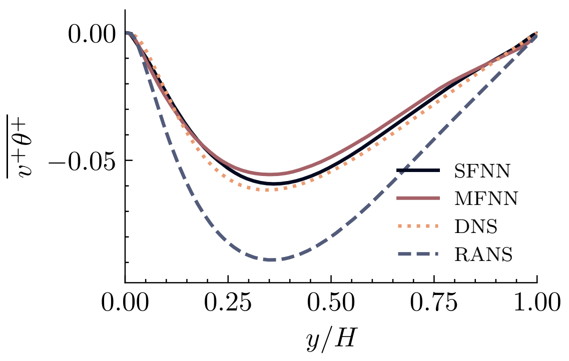


(b) D_{HF} tensor.

Figure 5.22: S7 predictions for $Re_\tau = 395,\ Pr = 0.01$ using the complete loss function. Notice how the model is simultaneously achieving a good prediction of the low fidelity α_t and still being able to correct the low fidelity prediction.



(a) Wall-normal component $\overline{v\theta}$ at $Re_\tau = 395,\ Pr = 0.01$.

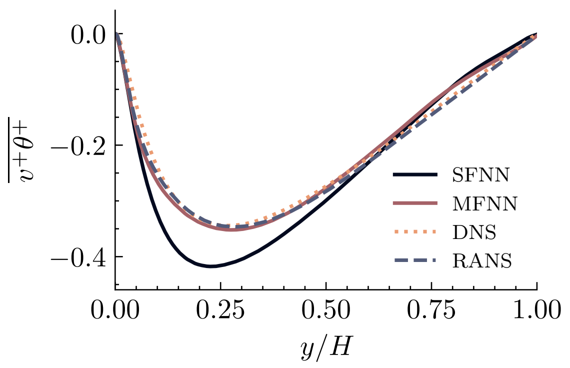


(b) Wall-normal component $\overline{v\theta}$ at $Re_\tau = 395,\ Pr = 0.01$.

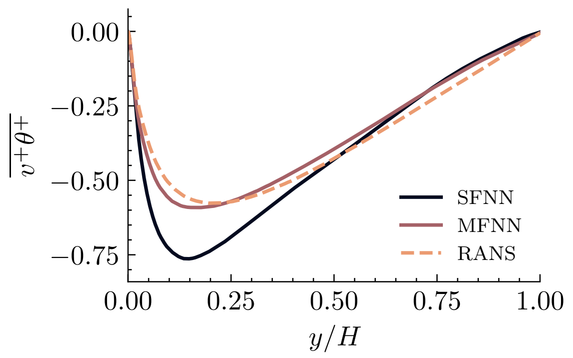
Figure 5.23: S7 predicted wall-normal component $\overline{v\theta}$ for the $Re_\tau = 395,\ Pr = 0.01$ case.

learning the high Reynolds trends from the RANS data. This capability is shown in Figure 5.24. In both cases the MFNN model still follows quite well the DNS (in the case of $Re_\tau = 2000$) and RANS data, avoiding the typical SFNN bump. Even this result must be taken with caution: the model is in fact mostly veering towards the SFNN results at low Reynolds numbers. This is not a problem per se, but it must be acknowledged that the SFNN results are not necessarily accurate when used with RANS modeling. Therefore a possible area of future work is implementing the robustness-based training strategies explored in (Fiore et al., 2024; Saccaggi, 2023) into the multi-fidelity training. It's also interesting to note how this structure applies minimal changes to the SFNN predicted streamwise component until $Re_\tau = 2000$, while for $Re_\tau > 2000$ it constantly overestimates the SFNN prediction.

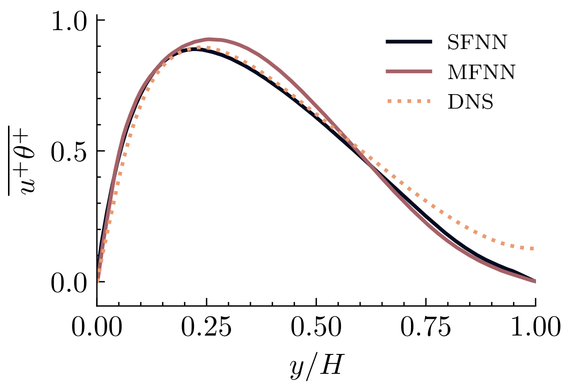
Even with the S7 structure, the training was very stable regardless of the loss function used, an example is shown in Figure 5.25; adding the mixed loss terms to the S7 training,



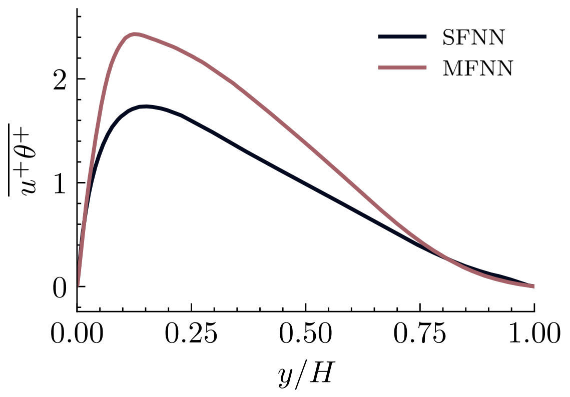
(a) Wall normal component $\overline{v\theta}$.
$Re_\tau = 2000,\ Pr = 0.01$.



(b) Wall normal component $\overline{v\theta}$.
$Re_\tau = 6000,\ Pr = 0.01$.



(c) Streamwise component $\overline{u\theta}$.
$Re_\tau = 2000,\ Pr = 0.01$.



(d) Streamwise component $\overline{u\theta}$.
$Re_\tau = 6000,\ Pr = 0.01$.

Figure 5.24: S7 predictions for high Re_τ and low Pr.

pushes the model towards the RANS prediction, as shown in Figure 5.26.

Overall, the S7 structure is the most promising among the ones tested. In fact the S1 and S5 structure can be crudely described as RANS approximators for the wall-normal component that add a streamwise component closely related to the SFNN one on top. S7 on the other hand proved to be a bit more flexible, being capable of mimicking the SFNN for flows inside the high fidelity range, while reverting to follow low fidelity patterns outside.

The main areas that need to be worked on remain how to improve the accuracy for very low Prandtl and Reynolds flows and how to improve the prediction of the streamwise component. As mentioned, for the first point a viable solution might be the path already explored with the single fidelity model in (Fiore et al., 2024; Saccaggi, 2023), that would increase the robustness of the network to sub-optimal RANS turbulence modeling. The second point could be tackled using more than two levels of fidelity: this would allow a wider range of models and geometries to be included in the training, adding representative results for the streamwise component at higher Reynolds numbers.

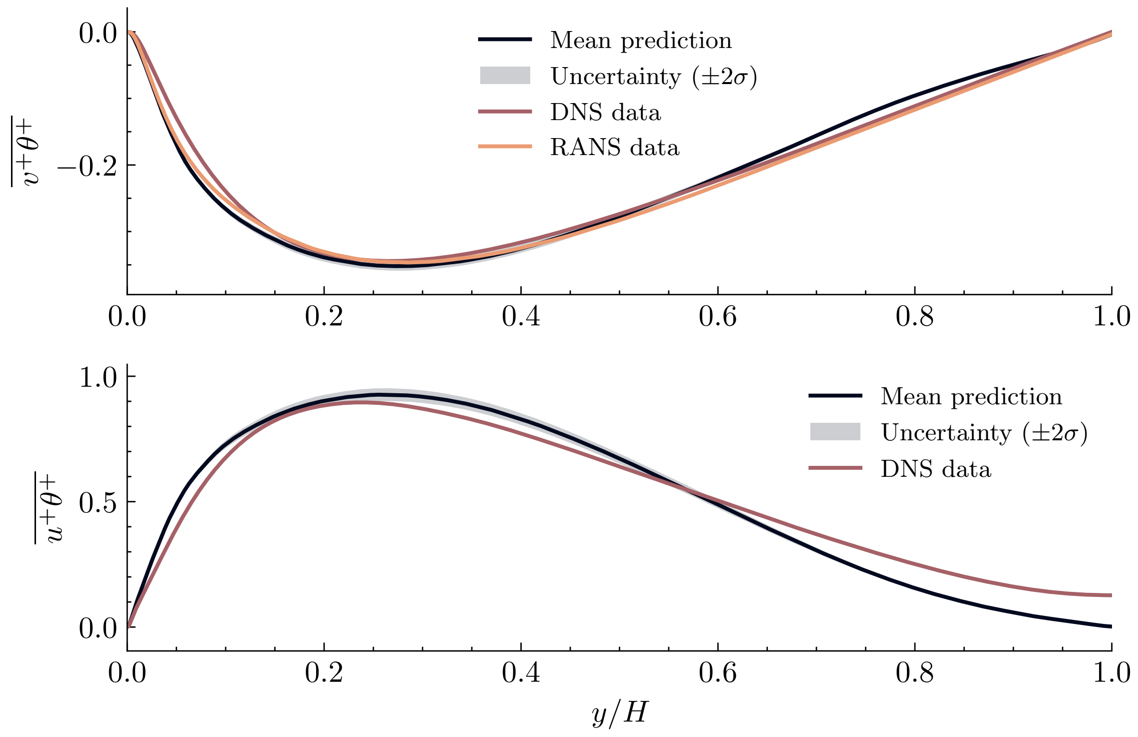


Figure 5.25: Uncertainty quantification for the S7 model. $Re_\tau = 2000,\ Pr = 0.01$

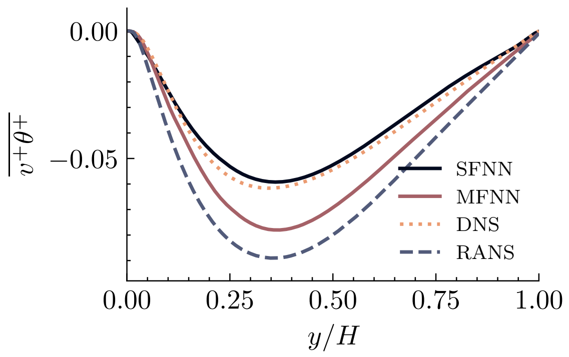
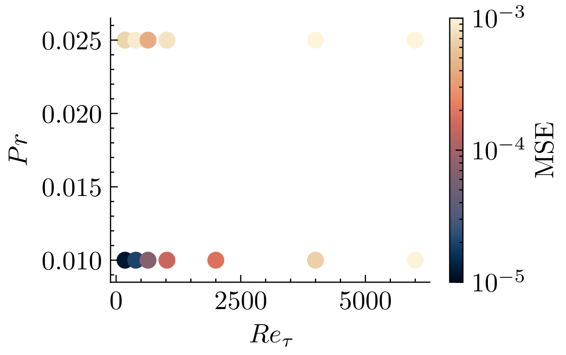
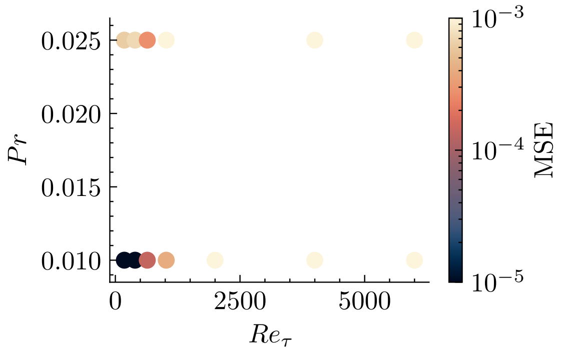


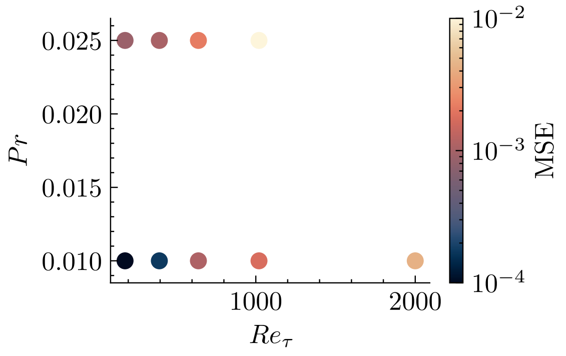
Figure 5.26: S7 prediction for the $Re_\tau = 395,\ Pr = 0.01$ case. Model trained using the mixed terms.



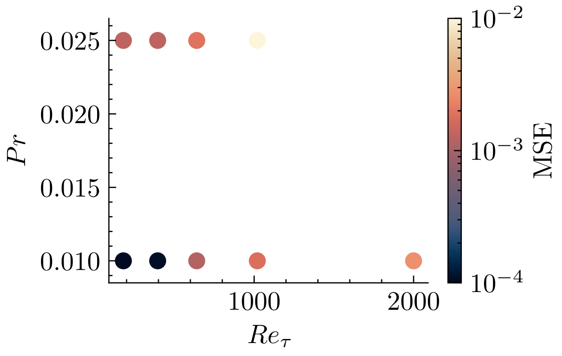
(a) MFNN, wall-normal component.



(b) SFNN, wall-normal component.



(c) MFNN, streamwise component.

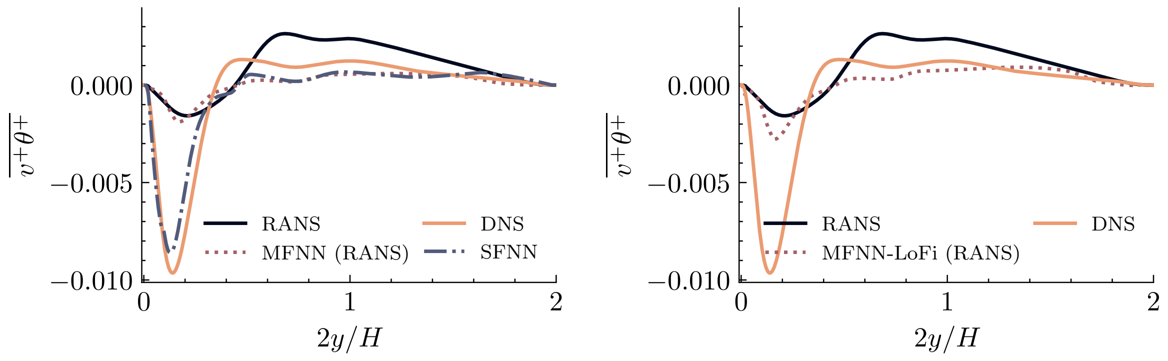


(d) SFNN, streamwise component.

Figure 5.27: Mean squared error comparison between the SFNN and the MFNN (S7). To compute the MSE, DNS data was used as ground truth where available, RANS elsewhere. These results were obtained without adding the mixed terms.

5.6 Impinging jet results

Up to this point, the data-driven models were only tested on channel flows. This was done for sake of simplicity but, at the same time, this is obviously a very restricted flow case, not representative of all the real use cases for a similar model.
The natural step forward is to test and train the model on more complex flows, as it was done in (Fiore, Koloszar, Fare, et al., 2022) for the single fidelity training. To this end the impinging jet flow (described in subsection 4.4.2) was considered, adding the RANS simulation to the low fidelity database and training an S7 structure.



(a) High fidelity prediction.

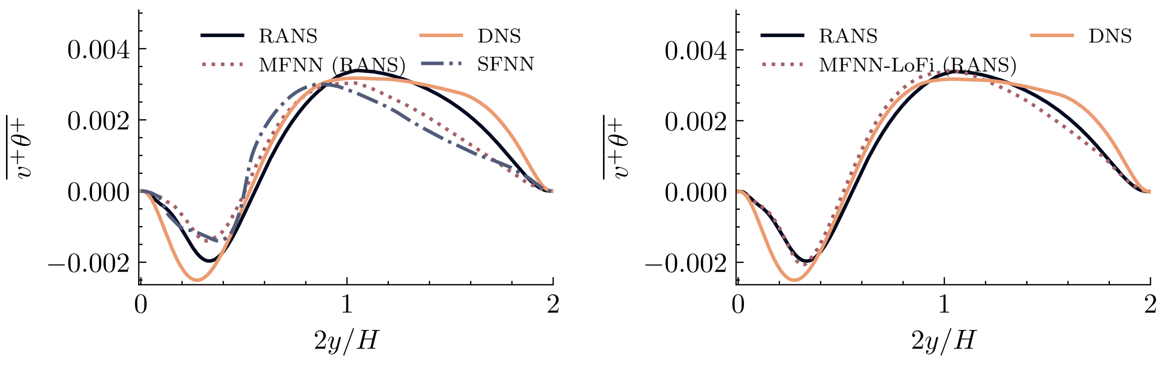
(b) Low fidelity prediction.

Figure 5.28: Multi-fidelity model prediction of $\overline{u\theta}$ for $x/B = 1$. Both the final (high fidelity) and the low fidelity prediction are shown.

In the impinging jet case, the key factor determining the accuracy of the multi-fidelity structure is the distance from the jet. Both the low and high fidelity predictions for $x/B = 1$ are shown in Figure 5.28. In this case the single fidelity model is actually outperforming the multi-fidelity, accurately capturing the minimum of the $\overline{v\theta}$ term. For $x/B > 0.2$, the two data-driven models are almost identical, both under-predicting the heat flux. Figure 5.28b shows the relative low fidelity prediction: the MFNN over-predicts the minimum, and then gets wrongly corrected by the high fidelity network as to match the RANS data.

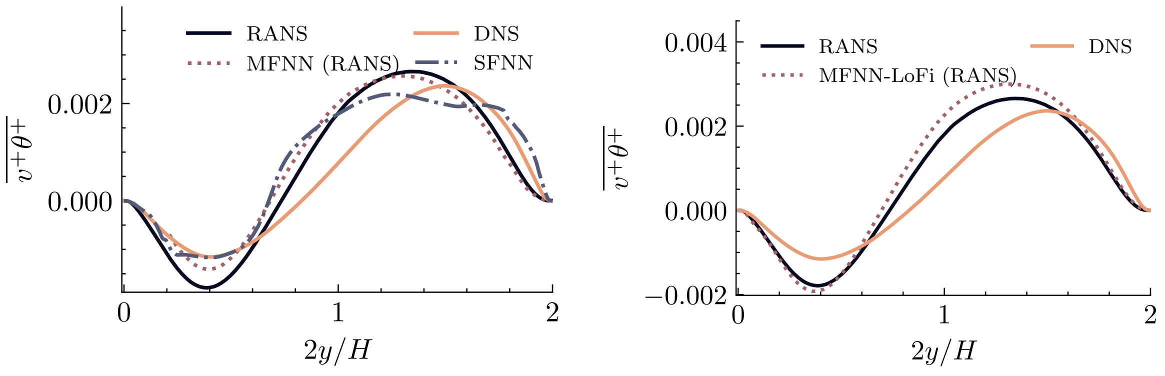
At $x/B = 5$ the situation improves, on one hand because the MFNN model is more accurate with respect to the RANS data, and on the other hand because the RANS data itself is closer to the DNS data. Comparing the two data-driven models, the difference is still minimal, even though the MFNN has a more similar shape to those of RANS and DNS.
The low fidelity prediction (Figure 5.29b) is quite accurate, to the point that it is more accurate than the high fidelity prediction. Once again, like for $x/B = 1$, the correction applied by the high fidelity part of the network is worsening the prediction made by the model.

For the furthest point from the jet available $x/B = 9$ very similar observations can be made. In this case the MFNN is superior to the single fidelity model in shape, but



(a) High fidelity prediction. (b) Low fidelity prediction.

Figure 5.29: Multi-fidelity model prediction of $\overline{u\theta}$ for $x/B = 5$. Both the final (high fidelity) and the low fidelity prediction are shown.



(a) High fidelity prediction. (b) Low fidelity prediction.

Figure 5.30: Multi-fidelity model prediction of $\overline{u\theta}$ for $x/B = 9$. Both the final (high fidelity) and the low fidelity prediction are shown.

once again the low fidelity prediction gets wrongly corrected, ending up further from the high fidelity data.

It must be noted that in the channel flow the RANS always overestimates the magnitude of the $\overline{v\theta}$ component. What happens is that the low fidelity correctly predicts the RANS value (as shown in Figure 5.23) and the high fidelity then correctly reduces its magnitude. One hypothesis is that, having no high fidelity jet data, the model still applies the same learned correction, that is clearly not valid for this use case.

As a final remark it is worth noting that adding the impinging jet RANS data to the low fidelity training pool did not alter the model's performance on the channel flow. For this reason, those results are not shown again in this section as they are practically identical to those in subsection 5.5.1.

Chapter 6

Conclusions and future works

Over the course of this project, several multi-fidelity data-driven models for the the turbulent heat flux have been developed with the intent of finding a solution to the lack of high fidelity training data.

The main objective was to prove that such models could effectively learn patterns given by the low-fidelity data and apply them to give physical predictions outside of the range of availability of high fidelity data.

This target was achieved with extensive tests for channel flows. These tests showed how adding the low fidelity data allowed the model to learn the characteristic behavior for high Reynolds numbers. Of all the structures tested, the S7 is the most promising, since it shows the ability to match the single fidelity predictions in regions covered by high fidelity training data, while resorting to apply the lower fidelity trends elsewhere.

At the same time, the model was also tested on the more complex flow represented by the impinging jet. This showed only slight, if any, improvement over the already existing single fidelity model. The reasons attributed to the poor performance in the impinging jet case are believed to be *a*) that is a more complex flow to begin with and RANS itself proved to be less accurate than in the channel case and *b*) only low fidelity data was fed to the model in the training process, therefore severely hindering its possibility to learn the relationship between the two levels. In turns, this signals the absolute necessity to extend the training database, in particular the high fidelity one, to 2D and 3D flows.

Moving forward the immediate next step is to augment the training database, as the data to do so is already available and more will come in the future. Extending the training database will hopefully improve the prediction for complex flows and perhaps more importantly give the opportunity to fairly assess the model's performance. Another path to explore is making the model capable of training on more than 2 levels of fidelity: this goes hand in hand with the previous point, as it would widen the choice of CFD models to draw data from. Examples of readily available data to include in the training dataset are the DNS of a turbulent boundary layer (Li et al., 2009) and the DNS of a backward facing step (Oder et al., 2019); ad-hoc LES simulations could also be run in the near future.

These two improvements could lead to the creation of a multi-fidelity input/multi-fidelity output model, that would leverage the findings of (Fiore et al., 2024; Saccaggi, 2023) to cure the modeling-related inconsistencies via multi-fidelity inputs. Such a model would then apply the techniques explored in the current project to output a multi-fidelity prediction, making sure that the model learns phenomenological trends, even in regions where no high fidelity data is available, as was done for high Re_τ in the case of channel flows.

Lastly, the multi-fidelity model must be implemented in a CFD suite so that an *a posteriori* validation can be carried out. The groundwork to do so already exists, as the single fidelity network was implemented in OpenFOAM in (Fiore, Koloszar, Fare, et al., 2022).

Bibliography

Abe, K., & Suga, K. (2001). Towards the development of a reynolds-averaged algebraic turbulent scalar-flux model. *International Journal of Heat and Fluid Flow*, *22*(1), 19–29. https://doi.org/10.1016/S0142-727X(00)00062-X

Alcántara-Ávila, F., Hoyas, S., & Pérez-Quiles, M. J. (2018). DNS of thermal channel flow up to reτ=2000 for medium to low prandtl numbers. *International Journal of Heat and Mass Transfer*, *127*, 349–361. https://doi.org/10.1016/j.ijheatmasstransfer.2018.06.149

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bricteux, L., Duponcheel, M., Winckelmans, G., Tiselj, I., & Bartosiewicz, Y. (2012). Direct and large eddy simulation of turbulent heat transfer at very low prandtl number: Application to lead–bismuth flows. *Nuclear Engineering and Design*, *246*, 91–97. https://doi.org/10.1016/j.nucengdes.2011.07.010

Brunton, S. L., & Kutz, J. N. (2019). *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press. https://doi.org/10.1017/9781108380690

Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine learning for fluid mechanics [Publisher: Annual Reviews]. *Annual Review of Fluid Mechanics*, *52*, 477–508. https://doi.org/10.1146/annurev-fluid-010719-060214

Cherkassky, V. S., & Mulier, F. (2007). *Learning from data: Concepts, theory, and methods* (2nd ed) [OCLC: ocm76481553]. IEEE Press : Wiley-Interscience.

Da Vià, R., Cerroni, D., Menghini, F., & Manservisi, S. (2016). Numerical validation of a four parameter logarithmic turbulence model. *1*. https://doi.org/10.7712/100016.1857.6906

Daly, B. J., & Harlow, F. H. (1970). Transport equations in turbulence. *The Physics of Fluids*, *13*(11), 2634–2649. https://doi.org/10.1063/1.1692845

De Santis, A., Ortiz, A. V., Shams, A., & Koloszar, L. (2019). Modelling of a planar impinging jet at unity, moderate and low prandtl number: Assessment of advanced RANS closures. *Annals of Nuclear Energy*, *129*, 125–145. https://doi.org/10.1016/j.anucene.2019.01.039

Duponcheel, M., & Bartosiewicz, Y. (2021). Direct numerical simulation of turbulent heat transfer at low prandtl numbers in planar impinging jets. *International Journal of Heat and Mass Transfer*, *173*, 121179. https://doi.org/10.1016/j.ijheatmasstransfer.2021.121179

Errico, O., & Stalio, E. (2014). Direct numerical simulation of turbulent forced convection in a wavy channel at low and order one prandtl number. *International Journal of Thermal Sciences*, *86*, 374–386. https://doi.org/10.1016/j.ijthermalsci.2014.07.021

Errico, O., & Stalio, E. (2015). Direct numerical simulation of low-prandtl number turbulent convection above a wavy wall. *Nuclear Engineering and Design*, *290*, 87–98. https://doi.org/10.1016/j.nucengdes.2014.12.005

Fernández-Godino, M. G. (2023). Review of multi-fidelity models. *Advances in Computational Science and Engineering*, *1*(4), 351–400. https://doi.org/10.3934/acse.2023015

Fiore, M. (n.d.). *Matilde fiore's PhD thesis* [Doctoral dissertation].

Fiore, M., Koloszar, L., Fare, C., Mendez, M. A., Duponcheel, M., & Bartosiewicz, Y. (2022). Physics-constrained machine learning for thermal turbulence modelling at low prandtl numbers. *International Journal of Heat and Mass Transfer*, *194*, 122998. https://doi.org/10.1016/j.ijheatmasstransfer.2022.122998

Fiore, M., Koloszar, L., Mendez, M. A., Duponcheel, M., & Bartosiewicz, Y. (2022). Turbulent heat flux modelling in forced convection flows using artificial neural networks. *Nuclear Engineering and Design*, *399*, 112005. https://doi.org/10.1016/j.nucengdes.2022.112005

Fiore, M., Saccaggi, E., Koloszar, L., Bartosiewicz, Y., & Mendez, M. A. (2024, September 5). Data-driven turbulent heat flux modeling with inputs of multiple fidelity. https://doi.org/10.48550/arXiv.2409.03395

Forrester, A., Sóbester, A., & Keane, A. (2008). *Engineering design via surrogate modelling: A practical guide* [OCLC: 264714649]. Wiley ; John Wiley [distributor].

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Grötzbach, G. (2013). Challenges in low-prandtl number heat transfer simulation and modelling. *Nuclear Engineering and Design*, *264*, 41–55. https://doi.org/10.1016/j.nucengdes.2012.09.039

Horizon 2020 - european commission. (n.d.). Retrieved August 18, 2024, from https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-2020_en

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2019, February 25). Averaging weights leads to wider optima and better generalization. https://doi.org/10.48550/arXiv.1803.05407

Jolliffe, I. T. (2010). *Principal component analysis* (2. ed). Springer.

Kawamura, H., Abe, H., & Shingai, K. (2000). DNS of turbulence and heat transport in a channel flo w with different reynolds and prandtl numbers and boundary conditions. *Proceedings of the 3rd International Symposium on Turbulence, Heat and Mass Transfer*.

Kays, W. M. (1994). Turbulent prandtl number—where are we? *Journal of Heat Transfer*, *116*(2), 284–295. https://doi.org/10.1115/1.2911398

Li, Q., Schlatter, P., Brandt, L., & Henningson, D. S. (2009). DNS of a spatially developing turbulent boundary layer with passive scalar transport. *International Journal of Heat and Fluid Flow*, *30*(5), 916–929. https://doi.org/10.1016/j.ijheatfluidflow.2009.06.007

Liao, P. (, Song, W. (, Du, P. (, & Zhao, H. ((2021). Multi-fidelity convolutional neural network surrogate model for aerodynamic optimization based on transfer learning. *Physics of Fluids*, *33*(12), 127121. https://doi.org/10.1063/5.0076538

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, *807*, 155–166. https://doi.org/10.1017/jfm.2016.615

Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., & Wilson, A. G. (2019, December 31). A simple baseline for bayesian uncertainty in deep learning. https://doi.org/10.48550/arXiv.1902.02476

Manceau, R., & Hanjalić, K. (2002). Elliptic blending model: A new near-wall reynolds-stress turbulence closure. *Physics of Fluids*, *14*(2), 744–754. https://doi.org/10.1063/1.1432693

Manceau, R. (2015). Recent progress in the development of the elliptic blending reynolds-stress model. *International Journal of Heat and Fluid Flow*, *51*, 195–220. https://doi.org/10.1016/j.ijheatfluidflow.2014.09.002

Manservisi, S., & Menghini, F. (2014). A CFD four parameter heat transfer turbulence model for engineering applications in heavy liquid metals. *International Journal of Heat and Mass Transfer*, *69*, 312–326. https://doi.org/10.1016/j.ijheatmasstransfer.2013.10.017

Mendez, M. A., Ianiro, A., Noack, B. R., & Brunton, S. L. (Eds.). (2023). *Data-driven fluid mechanics: Combining first principles and machine learning*. Cambridge University Press.

Meng, X., & Karniadakis, G. E. (2020). A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *Journal of Computational Physics*, *401*, 109020. https://doi.org/10.1016/j.jcp.2019.109020

MYRRHA research and transmutation endeavour | MYRTE project | fact sheet | h2020 [CORDIS | european commission]. (n.d.). Retrieved August 18, 2024, from https://cordis.europa.eu/project/id/662186

Nuttall, W. J. (2022, June 16). *Nuclear renaissance: Technologies and policies for the future of nuclear power* (2nd ed.). CRC Press. https://doi.org/10.1201/9781003038733

Oder, J., Shams, A., Cizelj, L., & Tiselj, I. (2019). Direct numerical simulation of low-prandtl fluid flow over a confined backward facing step. *International Journal of Heat and Mass Transfer*, *142*, 118436. https://doi.org/10.1016/j.ijheatmasstransfer.2019.118436

Pearson, K. (1901). LIII. *On lines and planes of closest fit to systems of points in space*. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

Saccaggi, E. (2023, April 13). *Robustness-based training and explainability of a data-driven model to cure the inconsistency between RANS and DNS datasets* [laurea]. Politecnico di Torino.

Shams, A., Roelofs, F., Niceno, B., Guo, W., Angeli, D., Stalio, E., Fregni, A., Duponcheel, M., Bartosiewicz, Y., Tiselj, I., & Oder, J. (2019). Reference numerical database for turbulent flow and heat transfer in liquid metals. *Nuclear Engineering and Design*, *353*, 110274. https://doi.org/10.1016/j.nucengdes.2019.110274

Shams, A., Roelofs, F., Tiselj, I., Oder, J., Bartosiewicz, Y., Duponcheel, M., Niceno, B., Guo, W., Stalio, E., Angeli, D., Fregni, A., Buckingham, S., Koloszar, L. K., Villa

Ortiz, A., Planquart, P., Narayanan, C., Lakehal, D., van Tichelen, K., Jäger, W., & Schaub, T. (2020). A collaborative effort towards the accurate prediction of turbulent flow and heat transfer in low-prandtl number fluids. *Nuclear Engineering and Design*, *366*, 110750. https://doi.org/10.1016/j.nucengdes.2020.110750

Smith, L. N. (2017, April 4). Cyclical learning rates for training neural networks. https://doi.org/10.48550/arXiv.1506.01186

Tiselj, I., Bergant, R., Mavko, B., Bajsic´, I., & Hetsroni, G. (2001). DNS of turbulent heat transfer in channel flow with heat conduction in the solid wall. *Journal of Heat Transfer*, *123*(5), 849–857. https://doi.org/10.1115/1.1389060

Wu, J., Feng, X., Cai, X., Huang, X., & Zhou, Q. (2023). A deep learning-based multi-fidelity optimization method for the design of acoustic metasurface. *Engineering with Computers*, *39*(5), 3421–3439. https://doi.org/10.1007/s00366-022-01765-9

Wydler, P. (2005). Liquid metal cooled reactors [Number: 12]. *CHIMIA*, *59*(12), 970–970. https://doi.org/10.2533/000942905777675381

Yarlanki, S., Rajendran, B., & Hamann, H. (2012). Estimation of turbulence closure coefficients for data centers using machine learning algorithms [ISSN: 1087-9870]. *13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 38–42. https://doi.org/10.1109/ITHERM.2012.6231411

Zhang, X., Xie, F., Ji, T., Zhu, Z., & Zheng, Y. (2021). Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. *Computer Methods in Applied Mechanics and Engineering*, *373*, 113485. https://doi.org/10.1016/j.cma.2020.113485

Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability [Conference Name: IEEE Transactions on Emerging Topics in Computational Intelligence]. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(5), 726–742. https://doi.org/10.1109/TETCI.2021.3100641