A Priori Uncertainty Quantification In Reynolds-averaged Turbulence Models With Bayesian Deep Learning

Master thesis

Letian Jiang

Delft University of Technology

TU Delft

A Priori Uncertainty Quantification In Reynolds-averaged Turbulence Models With Bayesian Deep Learning

Master thesis

by

Letian Jiang

Supervisors:	Dr. R.P.Dwight
Project Duration:	Jan., 2024 - Sep., 2024
Faculty:	Faculty of Aerospace Engineering, Delft

Cover: Photograph of a airfoil with separated flow over the top surface.(DLR, CC-BY 3.0)

TUDelft

Acknowledgment

Studying abroad has been a challenging yet enriching and rewarding journey, and I am grateful to be on the verge of graduation after two intense and fulfilling years of hard work.

Firstly, I would like to express my deepest gratitude to my parents for their support in allowing me to study in the Netherlands. I also owe a great deal of thanks to several individuals who significantly contributed to the completion of this thesis. I am especially grateful to Renzhi Tian and Tyler Buchanan, whose insightful discussions played a crucial role in shaping the structure of the BNNs and presenting the final results. Additionally, I want to thank Graham Pash for generously sharing his code, which helped me finalize the method selection. I wish them all the best in their future doctoral endeavors.

I would especially like to thank my supervisor, Dr. Richard Dwight. It was under his meticulous guidance and through our discussions that I was able to complete all the work for this thesis.

Letian Jiang
Delft, September 2024

Summary

This thesis explores the use of Bayesian Deep Learning to improve uncertainty quantification in Reynolds-Averaged Navier-Stokes (RANS) turbulence models. While RANS models are commonly used in computational fluid dynamics due to their efficiency, they are often criticized for inaccuracies in certain flow conditions, primarily due to the challenges in modeling the Reynolds stress term. The thesis acknowledges the limitations of traditional turbulence models, which rely heavily on empirical parameters and often fail to generalize across different flow scenarios, leading to significant uncertainties.

To address these issues, the research introduces a data-driven approach, leveraging Bayesian Neural Networks (BNNs). BNNs are particularly suitable for this task because they not only improve prediction accuracy but also provide a mechanism to quantify uncertainties arising from both the model and the data. This dual uncertainty quantification is critical, as it helps to address the inherent ”black box” nature of machine learning models, which can introduce additional uncertainties into the predictions.

The methodology involves correcting traditional turbulence models and integrating them with BNNs to capture both aleatoric (data-driven) and epistemic (model-driven) uncertainties. The thesis demonstrates the effectiveness of this approach through various flow case studies, comparing the results against more accurate but computationally expensive methods like Large Eddy Simulations (LES) and Direct Numerical Simulations (DNS).

The research concludes that the integration of Bayesian Neural Networks into RANS turbulence models not only enhances predictive accuracy but also provides a more comprehensive uncertainty quantification, making it a promising direction for future work in turbulence modeling.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AD	Automatic differential
AI	Artificial intelligence
BNNs	Bayesian neural networks
CBFS	Curved backward-facing step
CFD	Computational fluid dynamics
DNS	Direct numerical simulation
ELBO	Evidence lower bound
FNN	Feedforward neural network
KL	Kullback-Leibler divergence
LES	Large-eddy simulation
LEVM	Linear eddy viscosity model
ML	Machine Learning
NASA	National Aeronautics and Space Administration
NLL	Negative log likelihood
NLP	Natural language processing
NN	Neural network
PH	Periodic hills
RANS	Reynolds-averaged Navier–Stokes
RBF	Radial basis function
RST	Reynolds stress transport equation
ReLU	Rectified linear unit
RMS	Root mean square
SB	Separation bubble
SGD	Stochastic gradient descent
SVGD	Stein variational gradient descent
SGS	Subgrid-scale
SST	Shear stress transport
URANS	Unsteady Reynolds-averaged Navier–Stokes
TBNN	Tensor basis neural network
TBRF	Tensor basis random forest
TI	Turbulence intensity
TKE	Turbulent kinetic energy
UQ	Uncertainty quantificiation
	...

Symbols

Symbol	Definition	Unit
a_{ij}	Reynolds stress anisotropy	$[m^2/s^2]$
b_{ij}	Normalised reynolds stress anisotropy	$[m^2/s^2]$
b_{ij}^{Δ}	Reynolds stress anisotropy correction	$[m^2/s^2]$
$\mathcal{D}$	Dataset for the training data	[-]
$\mathbb{E}$	Expectation for the distribution	[-]
ℓ	Numbers of iteration	[-]
F_1	Blending function for $k\omega$ SST model	[-]
F_2	Blending function for $k\omega$ SST model	[-]
H	Matrix in extreme learning machine	[-]
$\mathcal{H}$	Hilbert space	[-]
J	Cost function	[-]
k	Turbulent kinetic energy	$[m^2/s^2]$
$\mathcal{N}$	Normal distribution	[-]
p	Pressure	$[m/s^4]$
$\mathcal{P}$	Turbulent kinetic energy production rate	$[m^2/s^2]$
$\mathcal{P}_k^{\Delta}$	Correction for turbulent kinetic energy production rate	$[m^2/s^3]$
Pr	Prandtl number	[-]
R	Correction for turbulent kinetic energy production rate	$[m^2/s^3]$
Re	Reynolds number	[-]
S_{ij}	Strain rate	$[1/s]$
T	Base tensors	[-]
t	Time	$[s]$
u_i	Velocity vector	$[m/s]$
u_i'	Instantaneous/Fluctuating velocity	$[m/s]$
$\bar{u}$	Mean velocity	$[m/s]$
w	Weights for neural networks	[-]
X	Input data for neural network	[-]
Y	Regression target for neural network	[-]
...		
α	Function coefficient in extreme learning machine	[-]
α_n	Coefficient for the invariant	[-]
Δ	Grid size	$[m]$
ϵ	Dissipation rate	$[m^2/s^3]$
η	Step size for the SVGD method	[-]
Ω_{ij}	Rotational tensor	$[1/s]$
ω	Specific rate of dissipation	$[1/s]$
ρ	Density	$[kg/m^3]$
μ	Dynamic viscosity	$[kg/ms]$
μ_t	Dynamic eddy viscosity	$[kg/ms]$

Symbol	Definition	Unit
ν	Kinematic viscosity	$[m^2/s]$
ν_t	Kinematic eddy viscosity	$[m^2/s]$
σ	Standard deviation	[-]
θ	Neural networks parameters	[-]
τ_{ij}	Viscous stress tensor	$[N/m^2]$
ε	Noise added to the training data	[-]
...		

1 Introduction

Computational fluid dynamics (CFD) has become a more popular method for fluid flow analysis in recent years because of the quick progress achieved in computer technology, particularly in the industrial area. The Reynolds-Averaged Navier-Stokes (RANS) equation is still a popular CFD method because it is effective in estimating time-averaged turbulent flow quantities. However, because of the Reynolds stress term modeling, RANS simulations are frequently criticized for being inaccurate in specific flow conditions [1]. Even while more precise techniques like Direct Numerical Simulations (DNS) and Large Eddy Simulations (LES) are now more widely available, their high processing costs make them impractical for use in engineering applications, particularly in design and optimization jobs that call for many iterations. Thus, it is imperative to increase the predictive power and accuracy of RANS simulations.

Despite significant advancements in the field of RANS methods, the core issue of turbulence modeling theory has seen minimal theoretical progress over the past two decades. Traditional turbulence models largely rely on the Boussinesq assumption and a set of empirically-based parameters, encapsulated within either one-equation or two-equation frameworks, such as the k-epsilon and k-omega models. These models often perform well for specific flow scenarios but fail to generalize across diverse conditions, highlighting a major source of model form uncertainty in RANS simulations.

The rise of machine learning has led to new research methods for improving RANS turbulence modeling. Broadly, these approaches can be classified into three categories: direct modeling of the anisotropic terms of the Reynolds stress, adjustment of the coefficients in existing turbulence models, and the introduction of new terms to the turbulence equations. However, the inherent "black box" nature of machine learning models introduces another layer of uncertainty, which remains a critical challenge for RANS models. Furthermore, the dependency of machine learning outcomes on the quality of training data adds an additional dimension of uncertainty that needs to be carefully managed.

A promising direction that seeks to enhance prediction quality while also quantifying uncertainties is the application of Bayesian Neural Networks (BNNs). Pioneering studies by Geneva et al. [2] and Tang et al[3] have utilized BNNs to develop new turbulence models that incorporate model-driven uncertainty. Nevertheless, these approaches have not fully considered the uncertainties arising from the data itself. Building on their groundwork, this thesis proposes a novel framework employing BNNs that aims to quantify uncertainties originating both from the model and the data.

This research seeks to investigate the following question:

Can the integration of the Bayesian Neural Network framework in data-driven turbulence modeling not only improve the accuracy of RANS results but also capture the dual uncertainties inherent in both the model and the data, surpassing current methods?

1.1. Outline of the thesis

This thesis has mainly six chapters. The text provides an initial introduction to the fundamentals of CFD, encompassing the mathematical formulation and the pros and cons of several RANS models. Additionally, a brief overview of both DNS and LES methods is included. Chapter 2 provides a thorough explanation of the necessity of turbulence models in Reynolds-averaged Navier-Stokes (RANS) simulations. This chapter also examines the current body of work on data-driven turbulence modeling and the measurement of uncertainty in turbulence modeling. The concluding section of Chapter 2 centers on an exhaustive examination of the thesis's aim.

The methodology is covered in Chapter 3, where the applied turbulence model modifications are described. The BNN framework's operation is also described, along with the steps needed to train it. Additionally, the discussion includes the new appropriate neural network topology for the uncertainty quantified turbulence modeling problem.

Chapter 4 addresses the experimental setup and the hyperparameter optimization process essential for enhancing the neural network's performance. This chapter also identifies the most effective training methods selected based on their performance during testing phases.

In Chapter 5, the application of the BNN to derive improved turbulence models and their uncertainty ranges is presented. The results include a priori tests and comparisons with LES/DNS results. Moreover, to evaluate the neural network's generalization capabilities, various flow cases are tested, and the outcomes are thoroughly discussed.

Finally, Chapter 6 offers suggestions for further investigation as well as a summary of the results.

2
Background and literature review

2.1. Introduction to computational fluid dynamics

CFD analyzes fluid flows using numerical methods to examine interactions between fluids, solids, and gases. It is widely used in fields like aerodynamics and hydrodynamics to study lift, drag, pressure, and velocity distributions. Governed by physical laws as partial differential equations, CFD solvers convert these into algebraic equations for efficient numerical solutions.

CFD encompasses various methodologies, including DNS, les, and RANS. DNS resolves all scales of turbulent motion, providing highly accurate results but at a significant computational cost, making it suitable for fundamental studies rather than practical engineering applications. LES offers a balance by resolving large-scale turbulent structures while modeling smaller scales, reducing computational demands compared to DNS while still capturing essential flow features. RANS, on the other hand, averages all effects of turbulence, leading to substantial simplifications and lower computational costs, making it ideal for industrial applications despite its reduced accuracy in capturing detailed turbulence structures[4]. This section will briefly discussed about the application and limitation of this method which is mainly come from Pope[1].

2.1.1. Laminar flow and turbulence flow

In laminar flow, fluid particles traverse in orderly, parallel layers with negligible interlayer mixing. The flow exhibits a smooth and predictable behavior, with each layer gliding past its neighboring layers with minimal disturbance. This flow regime is typically associated with low fluid velocities and high viscosities. The transition from laminar to turbulent flow is governed by the dimensionless Reynolds number, which encapsulates the ratio of inertial forces to viscous forces within the fluid system. When the Reynolds number exceeds a critical threshold, the flow becomes unstable and transitions to turbulence

$$Re = \frac{\rho L U}{\mu}. \tag{2.1}$$

Upon reaching a critical Reynolds number, fluid flows undergo a transition from laminar to turbulent behavior. Turbulent flows are characterized by their unsteady, irregular, and seemingly random and chaotic nature.

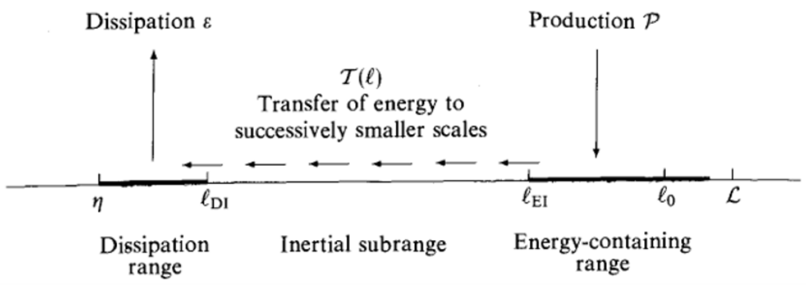


Figure 2.1: An energy cascade schematic diagram at for the CFD method at high Reynolds numbers[1].

Due to the intricate nature of turbulence, scientists have found that describing it in terms of energy is a particularly effective approach. Turbulent flows are frequently characterized by the concept of the energy cascade, initially introduced by Richardson [5] and later mathematically formalized by Kolmogorov [6]. The energy cascade describes the hierarchical process in turbulent flows wherein large vortices, referred to as eddies, break-down into progressively smaller eddies. This phenomenon facilitates the transfer of kinetic energy from macroscopic scales to microscopic scales. As eddies fragment into increasingly smaller structures, the kinetic energy they harbor is transferred down the cascade until the eddies reach a scale where viscous forces dominate, leading to the dissipation of kinetic energy as heat. This entire process can be visualized in Figure 2.1, which illustrates the multi-scale energy transfer in turbulent flows.

Additionally, the Kolmogorov hypothesis postulates that at sufficiently high Reynolds numbers, the small-scale statistics of turbulence become universal and are predominantly determined by the rate of energy dissipation and the viscosity of the fluid. This universality suggests that, despite the chaotic appearance of turbulence, underlying statistical properties remain consistent across different turbulent systems.

Given the inherent complexity of turbulence, addressing turbulence problems becomes critically important in the field of CFD. This importance is particularly evident in the context of RANS turbulence modeling. The following sections will explore the details of turbulence modeling and its significance in capturing the intricate nature of turbulent flows.

2.1.2. DNS and LES

In the process of averaging the Navier-Stokes equations, turbulent fluctuations are eliminated, with all turbulence effects on the flow being represented through turbulence models. Alternatively, turbulence can be directly modeled, though this approach incurs a high computational cost.

The Navier-Stokes equations may be numerically solved without the need for turbulence models is called the DNS method[7]. This method requires the resolution of the whole range of temporal and spatial turbulence scales. The computational mesh must capture all spatial scales, ranging from the lowest dissipative scales (Kolmogorov microscales, η) to the integral scale L, which includes the majority of the kinetic energy. DNS thus has very high computing requirements. The number of mesh points and time steps determines how many floating-point operations are needed.

Thus, even at low Reynolds values, the computing cost of DNS remains unaffordable. DNS would require computing resources greater than even the most sophisticated supercomputers, especially for Reynolds numbers common of most high applications.

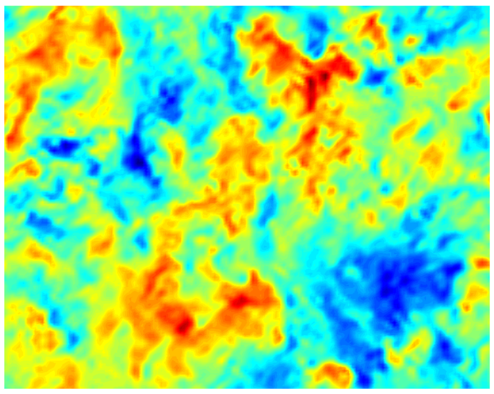


Figure 2.2: Velocity field from a DNS simulation of homogeneous decaying turbulence in a periodic box. DNS data is from Lu and Rutland.[8]

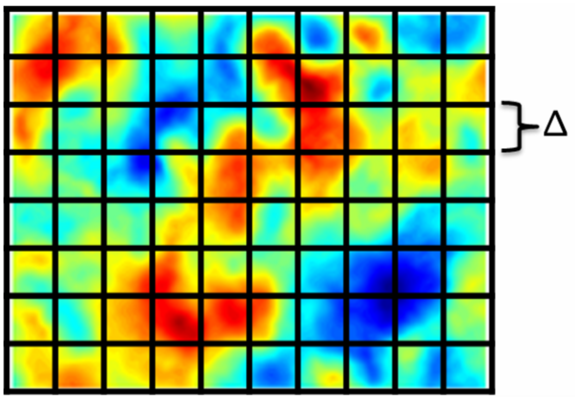


Figure 2.3: Velocity field from a LES simulation of homogeneous decaying turbulence in a periodic box with a LES filter [9]

Large-Eddy Simulation (LES), first introduced by Joseph Smagorinsky in 1963 [10], is a more realistic simulation technique. Like DNS, the biggest turbulent scales in LES are resolved immediately. Nevertheless, the impacts of the lowest turbulent scales are represented by simplified models rather than being calculated explicitly. As opposed to DNS, this method drastically lowers computing requirements by avoiding the direct resolution of the tiniest turbulence scales. Low-pass filtering the Navier-Stokes equations effectively eliminates small-scale turbulence information from the numerical solution, resulting in this decrease. Figures 2.2 and 2.3 show the differences between the DNS and LES results. It is obviously that results from LES show lower resolution status compared to DNS.

Although this small-scale information is not negligible, its impact on the flow field must be modeled, which remains an active research area, particularly for cases where small scales are significant. Despite this reduction in computational cost compared to DNS, LES still entails considerable computational expense relative to RANS methods. Consequently, RANS continues to be the predominant approach for flow analysis, and enhancing the accuracy of RANS models remains an ongoing area of research.

2.1.3. RANS Governing Equations

Since this work is mainly discussed about the RANS turbulence modeling, a short introduction of this method will be first discussed here. The Navier-Stokes equations for incompressible Newtonian viscous flows are

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.2}$$

$$\rho\frac{\partial u_i}{\partial t} + \rho\frac{\partial}{\partial x_j}\left(u_i u_j\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}. \tag{2.3}$$

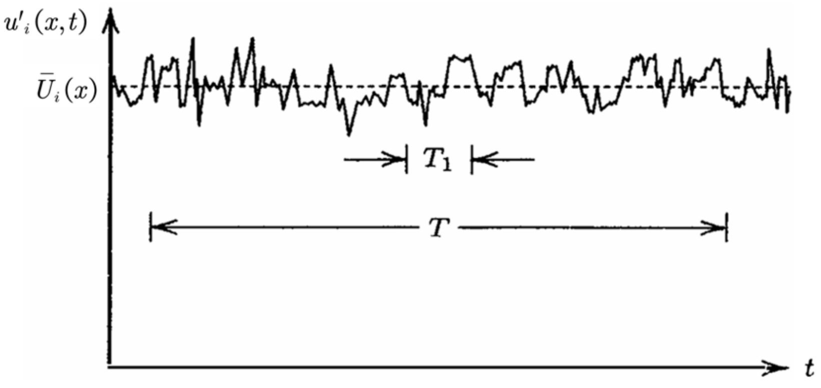


Figure 2.4: Time averaging for stationary turbulence. Although not evident due to the scale of the graph, the instantaneous velocity $u\prime_i\,(x,t)$ possesses continuous derivatives of all orders.[11].

The conservation of mass equation is found in equation 2.2, whereas the conservation of momentum equation is found in equation 2.3. The velocity vector in this case is denoted by u_i, the density by ρ, the pressure by p, and the viscous stress tensor by τ_{ij}. Given that τ_{ij} is proportional to both the dynamic viscosity μ and the strain rate S_{ij}, the strain rate tensor may be computed using velocity gradients as follow

$$\tau_{ij} = 2\mu S_{ij}, \tag{2.4}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{2.5}$$

Derivation of the Reynolds-Averaged Navier-Stokes (RANS) equations presupposes that the fluid flow can be represented as the sum of a mean component and a fluctuating component, a concept known as Reynolds decomposition. And the Reynolds decomposition of velocity is

$$u = \overline{u} + u'. \tag{2.6}$$

The fluctuation term is u', while the mean term is $\overline{u}$. Additionally, it is presumable that u' has an average of zero and that $\overline{u}$ has an average of itself

$$\overline{\overline{u}} = \overline{u}, \overline{u'} = 0. \tag{2.7}$$

By substituting the decomposition of the velocity field into its mean and fluctuating components, as described by equation 2.6, into the Navier-Stokes equations, and subsequently applying the operator rules outlined in Appendix A, the Reynolds-Averaged

Navier-Stokes (RANS) equations can be systematically derived as

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0, \tag{2.8}$$

$$\rho \frac{\partial \overline{u}_i}{\partial t} + \rho \frac{\partial}{\partial x_j}\left(\overline{u}_i \overline{u}_j\right) = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\overline{\tau}_{ij} - \rho \overline{u_i' u_j'}\right). \tag{2.9}$$

The resulting equations are nearly identical to the original momentum equations for the velocity component u, with the exception of an additional term $\overline{u_i' u_j'}$. This term is referred to as the Reynolds stress tensor, as it functions similarly to an apparent stress within the fluid. The value of the Reynolds stress tensor is not inherently known and must be modeled or calculated, presenting what is known as the closure problem in the context of RANS equations. Specifically, the introduction of the Reynolds stress term introduces additional unknowns, leading to a situation where there are more unknowns than equations—five unknowns in four equations. To resolve this discrepancy, turbulence models are employed. These models approximate the Reynolds stress tensor in terms of known parameters, thereby closing the system and allowing for a solvable set of equations.

2.1.4. RANS Turbulence Modeling

Fortunately a conservation law for the Reynolds stress tensor can be easily derived which is called Reynolds Stress Transport equation (RST). The general form for this equation is

$$\frac{\overline{u_i' u_j'}}{\partial t} + \underbrace{K_{ij}}_{\text{Advection}} = \underbrace{P_{ij}}_{\text{Production}} + \underbrace{T_{ij} + D_{ij}^{v} + D_{ij}^{p}}_{\text{Diffusion}} + \underbrace{\Phi_{ij}}_{\text{Pressure strain correlation}} - \underbrace{\varepsilon_{ij}}_{\text{Dissipation}}. \tag{2.10}$$

The Reynolds stress transport (RST) equation is fundamental in turbulence modeling but inherently complex due to the inclusion of unknown correlations like $\overline{u_i' p'}$ and $\overline{u_i' u_j' u_k'}$. For each unknown correlation, one can derive a corresponding conservation law. However, this process introduces additional unknown quantities, creating a cascade of higher-order terms. For instance, the transport equation for the triple correlation $\overline{u_i' u_j' u_k'}$ involves a fourth-order tensor, complicating the equation further. This phenomenon, known as the closure problem, implies that there will always be more unknowns than equations, making direct solutions mathematically infeasible.

Empirical approximations, namely turbulence models, have been created to overcome this difficulty. These models use assumptions taken from theory or observation to finish the system of equations, simplifying the complicated reality of turbulence. A broad variety of turbulence models are available, the most popular of which are one- or two-equation linear eddy viscosity models (LEVM). Because of their ease of use and computational efficiency, these models are expected to be popular for some time to come [12]. Ludwig Prandtl established the idea of the mixing length in 1925 [13], which laid the groundwork for contemporary turbulence theories and gave rise to the field of turbulence modeling. Since two-equation turbulence models are the most common in data-driven turbulence research, they will be the main topic of discussion in this section.

The turbulent viscosity theory, initially formulated by Boussinesq [14], posits a representation of the Reynolds stress that is mathematically analogous to the stress-strain relationship in a Newtonian fluid

$$\overline{u_i' u_j'} = \frac{2}{3}\delta_{ij}k + a_{ij}, \tag{2.11}$$

$$k = \frac{1}{2}\overline{u_k' u_k'}, \tag{2.12}$$

$$a_{ij} \approx -v_t \left(\frac{\overline{u_i}}{\partial x_j} + \frac{\overline{u_j}}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\overline{u_k}}{\partial x_k} \right). \tag{2.13}$$

The variables k and ν_t represent the turbulent kinetic energy (TKE) and turbulent viscosity, respectively. By assuming that the flow is incompressible, the following outcomes for the Reynolds stress tensor are obtained

$$\overline{u_i' u_j'} \approx \frac{2}{3}\delta_{ij}k - v_t \left(\frac{\overline{u_i}}{\partial x_j} + \frac{\overline{u_j}}{\partial x_i} \right). \tag{2.14}$$

The method known as Boussinesq's eddy-viscosity model is used to express the Reynolds stress tensor as the result of multiplying the mean strain-rate tensor by the eddy viscosity. Nevertheless, the model presents two unknowns: the eddy viscosity (ν_t) and the turbulent kinetic energy (k). The two-equation turbulence models are meant to help identify these unknowns so that the system of equations may be solved. Two turbulence-related parameters in these models are obtained by solving two transport equations. One equation is usually solved for k, and the other one is usually solved for some other variable.Once t are determined, the system of RANS equations is closed and can be solved. Different models use various second variables, with some of the most well-known models listed below in chronological order.

The $k - \epsilon$ Model

The $k - \epsilon$ model is a widely utilized RANS turbulence model, originally developed by Jones and Launder [15] in their seminal work on turbulence prediction. This model comprises two transport equations: one governing the turbulent kinetic energy, k, and another describing the rate of dissipation, ϵ. These governing equations are represented in Equation 2.15 and Equation 2.16 respectively. The $k - \epsilon$ model has been instrumental in advancing CFD by providing a robust framework for simulating turbulence in a wide range of engineering applications, from aerodynamic flows to process engineering.

$$\frac{\partial k}{\partial t} + \langle u_j \rangle \frac{\partial k}{\partial x_j} = \tau_{ij}\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\left[\frac{1}{\mathrm{Re}} + \frac{v_t}{\mathrm{Pr}_k} \right] \frac{\partial k}{\partial x_j} \right) - \varepsilon, \tag{2.15}$$

$$\frac{\partial \varepsilon}{\partial t} + \langle u_j \rangle \frac{\partial \varepsilon}{\partial x_j} = C_{\varepsilon 1}\frac{\varepsilon}{k}\tau_{ij}\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\left[\frac{1}{\mathrm{Re}} + \frac{v_t}{\mathrm{Pr}_\varepsilon} \right] \frac{\partial \varepsilon}{\partial x_j} \right) - C_{\varepsilon 2}\frac{\varepsilon}{k}\varepsilon. \tag{2.16}$$

The eddy viscosity ν_t is then calculated as follows

$$\nu_t = C_D \frac{k^2}{\varepsilon}. \tag{2.17}$$

The widely used constant of this model transport equation are

$$C_D = 0.09, \quad \mathrm{Pr}_k = 1, \quad \mathrm{Pr}_\varepsilon = 1.3, \quad C_{\varepsilon 1} = 1,44, \quad ,C_{\varepsilon 2} = 1,92. \tag{2.18}$$

These parameter values have been determined by analyzing simple reference flows, where many terms in the transport equations for k and ϵ cancel out, allowing for the isolated calibration of individual parameters.

The $k - \epsilon$ model is widely used due to its relatively low computational cost, requiring only two additional transport equations. This model is particularly suitable for external aerodynamic flows. However, it is best applied to scenarios without strong pressure gradients, significant streamline curvature, or flow separation. One of the challenges with this model is the complex formulation of numerical boundary conditions for ϵ. Eddy viscosity models, such as the $k - \epsilon$ model, are based on the assumption that the Reynolds stress is proportional to the mean strain rate, which restricts their ability to capture the distinct components of the Reynolds stress tensor. As a result, the $k - \epsilon$ model struggles to accurately account for anisotropic effects, including streamline curvature and directional volume forces like gravity[16][17].

The $k - \omega$ Model

Another popular Reynolds-RANS turbulence model is the $k - \omega$ model, which was first put forth by Wilcox [18]. It is predicated on the solution of a transport equation for the specific turbulence dissipation rate ω and the transport equation for the turbulent kinetic energy k. The following are the transport equations for the specific dissipation rate and kinetic energy of turbulence

$$\frac{\partial k}{\partial t} + \langle u_j \rangle \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial}{\partial x_j} \left(\left[\frac{1}{\mathrm{Re}} + \frac{v_t}{\mathrm{Pr}_k} \right] \frac{\partial k}{\partial x_j} \right) - C_D k \omega, \tag{2.19}$$

$$\frac{\partial \omega}{\partial t} + \langle u_j \rangle \frac{\partial \omega}{\partial x_j} = \alpha \frac{\omega}{k} \tau_{ij} \frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial}{\partial x_j} \left(\left[\frac{1}{\mathrm{Re}} + \frac{v_t}{\mathrm{Pr}_\omega} \right] \frac{\partial \omega}{\partial x_j} \right) - \beta \omega^2. \tag{2.20}$$

And the turbulent viscosity can be written as

$$\nu_T = \frac{k}{\omega} \quad , \quad \omega = \frac{1}{C_D} \frac{\varepsilon}{k}. \tag{2.21}$$

The model constants in the $k - \omega$ model are

$$C_D = 0.09, \quad \mathrm{Pr}_k = 2, \quad \mathrm{Pr}_\omega = 2, \quad \alpha = \frac{5}{9} \quad , \quad \beta = \frac{3}{40}. \tag{2.22}$$

Because it requires only two new transport equations and has a low computing overhead, the $k - \omega$ model is preferred. It performs exceptionally well in flows with pressure gradients and separation as well as boundary layer flows. The model also includes basic wall boundary requirements for ω. Nevertheless, the $k - \epsilon$ model is better suited for external aerodynamics because of its extreme sensitivity to inflow and freestream boundary conditions. The $k - \omega$ model has been found to have a propensity to overestimate the creation of turbulence at stagnation points[19].

The $k-\omega$ SST Model

Menter [20] created the SST $k-\omega$ turbulence model, a popular two-equation eddy-viscosity model. The $k-\omega$ and $k-\epsilon$ models' beneficial features are combined in the shear stress transfer (SST) formulation. The model may be applied right down to the wall via the viscous sub-layer by using a $k-\omega$ formulation inside the inner regions of the boundary layer. As a result, extra damping functions are not required for the SST $k-\omega$ model to perform as an efficient low-Reynolds number turbulence model. Moreover, the SST formulation shifts to a $k-\epsilon$ behavior in the free-stream, avoiding the problem of high sensitivity to the inlet free-stream turbulence conditions that is typical of the $k-\omega$ model[21]. Moreover, the following two transport equations relate to the specific dissipation rate and turbulent kinetic energy

$$\frac{\partial k}{\partial t}+U_j\frac{\partial k}{\partial x_j}=P_k-\beta^*k\omega+\frac{\partial}{\partial x_j}\left[\left(\nu+\sigma_k\nu_T\right)\frac{\partial k}{\partial x_j}\right], \tag{2.23}$$

$$\frac{\partial \omega}{\partial t}+U_j\frac{\partial \omega}{\partial x_j}=\alpha S^2-\beta\omega^2+\frac{\partial}{\partial x_j}\left[\left(\nu+\sigma_\omega\nu_T\right)\frac{\partial \omega}{\partial x_j}\right]+2\left(1-F_1\right)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}. \tag{2.24}$$

And the model constants in the model are

$$\begin{aligned}&\alpha_1=\frac{5}{9},\quad \alpha_2=0.44,\quad \beta_1=\frac{3}{40},\quad \beta_2=0.0828,\quad \beta^*=\frac{9}{100}\\&\sigma_{k1}=0.85,\quad \sigma_{k2}=1,\quad \sigma_{\omega 1}=0.5,\quad \sigma_{\omega 2}=0.856.\end{aligned} \tag{2.25}$$

The $k-\omega$ and $k-\epsilon$ models can be adjusted between each other by the F_1 which is the blending function. In the viscous sublayer and the boundary layer's logarithmic region, the blending function F_1 equals one; in the other regions of the domain, it gradually decreases to zero. This suggests that the last part in equation 2.24 is removed in the inner region of the boundary layer, leading to the use of the $k-\omega$ model. In contrast, the $k-\epsilon$ model is used when F_1 equals zero in the domain's outer regions. The blending function F_1 depends on how far away the closest wall is.

The eddy viscosity can be written as follow

$$\nu_t=\frac{a_1k}{\max\left(a_1\omega,SF_2\right)}. \tag{2.26}$$

When $S=\sqrt{2\overline{S}_{ij}\overline{S}_{ij}}$ and $a_1=0.31$ are found. It is observed by Mentor that the excessively high values of ν_t in the inner boundary layer caused by the above model still led to overestimations of wall shear stress. As a result, he limited the eddy viscosity ν_t using a different blending function F_2. Additionally, the following equation defines F_2:

$$F_2=\tanh\left[\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega y},\frac{500\nu}{y^2\omega}\right)\right]^2\right]. \tag{2.27}$$

2.2. Discuss for machine learning

Computers can learn and make judgments based on data according to a subset of artificial intelligence(AI) called machine learning (ML), which eliminates the need for explicit task programming. Large datasets are used to train algorithms so they can recognize patterns, draw conclusions, and forecast results on newly collected data. The fundamental function of machine learning is its capacity to extrapolate from training data and apply previously acquired knowledge to new contexts.

Three primary categories of machine learning approaches exist: reinforcement learning, unsupervised learning, and supervised learning. In supervised learning, algorithms are trained to predict results for fresh inputs using labeled data, where the desired output is known. Neural networks, decision trees, and linear regression are examples of common algorithms.

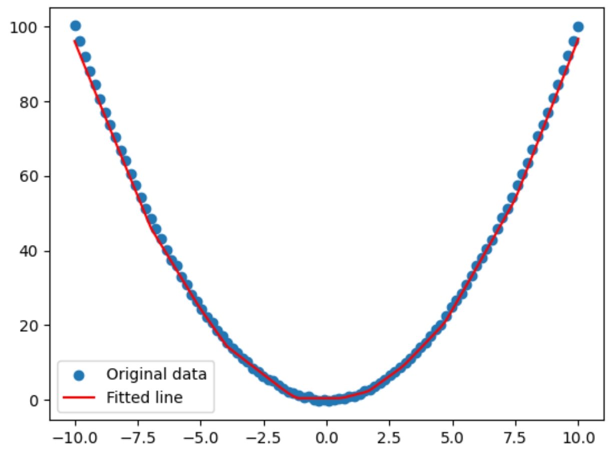


Figure 2.5: Toy example showing how the neural network regress the simple 1D data $y = x^2$

Figure 2.5 shows a simple results for supervised learning by neural network. The regress target is a very simple case :$y = x^2$, and the performance of the model is evaluated by the root mean square (RMS) function

$$RMS = \arg\min_{\theta} \sqrt{\frac{1}{n} \sum_{m=1}^{n} (f(x|\theta) - y)^2}. \tag{2.28}$$

where θ is the parameter of the neural networks, n is the number of the data , $f(x|\theta)$ is the prediction by the networks.

Fitting a polynomial is a widely recognized statistical technique, and this example is simple. But there is much more to the process of training a model and applying it to forecasts than this straightforward scenario. Learning very complicated representations is made possible by its ability to handle high-dimensional input and wide range of model applicability. This adaptability makes it possible to apply machine learning techniques to a wide range of complex datasets, making it easier to extract insightful patterns and information.

Unsupervised learning, on the other hand, deals with unlabeled data and focuses on uncovering hidden patterns or intrinsic structures within the data. Techniques such as clustering and dimensionality reduction fall into this category. Reinforcement learning involves training algorithms to make sequences of decisions by rewarding them for desirable actions, commonly used in robotics and game playing.

2.2.1. Neural networks

Neural networks have emerged as a powerful and versatile tool for approximating a wide range of target functions, whether they are real-valued, discrete-valued, or vector-valued. Their strength lies particularly in their ability to handle complex and high-dimensional input data, which can be challenging to interpret and process using traditional methods. This makes them particularly well-suited for applications where the underlying patterns in the data are intricate and not easily captured by simpler models. In recent years, neural networks have achieved significant breakthroughs across various domains, particularly in natural language processing (NLP), where they have revolutionized tasks such as machine translation, sentiment analysis, and text generation. Similarly, they have shown remarkable success in fields such as handwritten character recognition, where they can accurately identify and classify digits and letters, and in audio recognition, where they have enhanced the ability to recognize and interpret spoken language. Furthermore, neural networks have made substantial contributions to object identification in computer vision, enabling systems to detect and classify objects within images and videos with unprecedented accuracy.

Among the many types of neural networks that have been developed, the feedforward neural network (FNN), also known as a multilayer perceptron (MLP), stands out as one of the most fundamental architectures. An FNN is a type of artificial neural network where the connections between the nodes do not form a cycle. This network structure is specifically designed to approximate a function f that maps input data X to corresponding output data Y:

$$NN : X \to Y, \tag{2.29}$$

where X can be a continuous space of a discrete space. Neural networks were named after and inspired by biological systems, specifically the neural structures found in the human brain. Despite this inspiration, there is actually very little resemblance between the architecture of artificial neural networks and the complex workings of biological neural systems, which we still do not fully understand. In essence, a neural network is a machine learning algorithm characterized by its unique architecture. This architecture consists of layers of interconnected nodes, or neurons, which process input data and learn to make predictions or decisions based on that data.

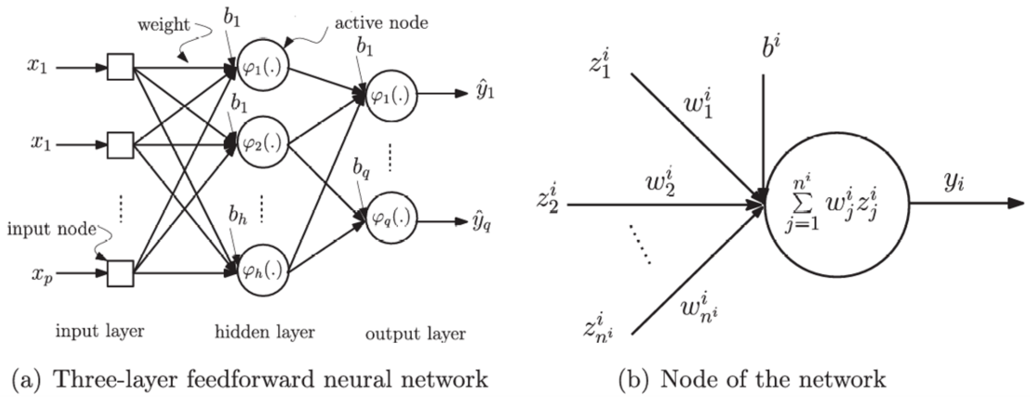


Figure 2.6: A diagram for a three-layer feedforward neural network (a), the hidden layer has h activation functions, the output layer has q nodes, and the input layer has p input nodes (b) Node unit diagram of neural network[22].

Multiple layers make up a neural network, as Figure 2.6 illustrates. The network is fed input data x into the input layer, which is followed by one or more hidden layers and an output layer that produces the final output y. The number of hidden layers within a network is referred to as its ”depth”. Nodes are the basic computational units of a neural network, and they make up each layer. A weight (w) is assigned to each input to a node, indicating its importance in relation to the other inputs. A node's output, y_{output}, is computed by adding the weighted sum of its inputs from the nodes in the layer before it. Each node's weighted sum is subjected to a non-linear activation function, enabling the network to simulate non-linear processes. Reversed linear unit (ReLU), sigmoid, and hyperbolic tangent functions are examples of common activation functions [23].

The primary objective of a neural network's training process is to decrease prediction error by adjusting the weights linking the neurons. These weights are often initialized with tiny, arbitrary values. During training, the dataset is iterated over repeatedly, with each iteration consisting of a forward pass and a backward pass. The input data is sent through the network during the forward pass, producing an output that is compared to the real target values using a loss function that calculates the prediction error. Using the chain rule, the backward pass, sometimes referred to as backpropagation [24], determines the gradient of the loss function with regard to each weight. These gradients lead the update of weights in an error-reducing direction, usually via the use of optimization methods like Adam's adaptive moment estimation and stochastic gradient descent (SGD). The weights eventually converge to values that minimize the loss function over repeated iterations, improving the accuracy of the network on fresh, unknown input.

The functions of neural networks can be broadly categorized into classification and regression tasks. However, for classification tasks, these models may encounter limitations in their generalization ability. For instance, a neural network trained to distinguish between apples and oranges can achieve high performance on this specific task. However, when presented with an image of a banana, the network will still provide probabilities for it being an apple or an orange, reflecting its training scope. This highlights that, despite their impressive classification capabilities, neural networks often fall short of a human's ability to understand and interpret diverse and unfamiliar inputs. On the other hand, for regression tasks, neural networks excel at processing large datasets to uncover underlying patterns. It has been mathematically proven that neural networks can approximate any continuous function, given that the activation function is bounded, continuous, and non-constant [25]. This universal approximation capability allows neural networks to model complex relationships within data effectively, making them powerful tools for regression analysis.

Beyond the standard feedforward neural networks, various specialized architectures have been developed to address specific types of data and tasks. One notable example is the Convolutional Neural Network (CNN), which is particularly effective for image and spatial data. CNNs utilize convolutional layers that apply filters to the input data, enabling the network to detect local patterns such as edges, textures, and shapes. This hierarchical feature extraction makes CNNs highly proficient in tasks like image classification, object detection, and segmentation. Other advanced structures include Recurrent Neural Networks (RNNs), designed to handle sequential data by maintaining hidden states that capture temporal dependencies, making them suitable for time series analysis and natural language processing. Additionally, Transformer networks, which is the most popular research area, with their self-attention mechanisms, have revolutionized the field of NLP by allowing for parallel processing and capturing long-range dependencies within the data, leading to significant advancements in language modeling and translation tasks [26]. Furthermore, for uncertainty quantification, Bayesian neural networks (BNNs) are another important tool. BNNs extend traditional neural networks by incorporating Bayesian inference principles, allowing them to provide probabilistic interpretations of model parameters and predictions. This approach enables the estimation of uncertainty in both the model weights and the outputs, which is particularly valuable in applications where understanding the confidence of predictions is crucial.

2.3. Data Driven Turbulence Modelling

In recent years, the field of CFD has witnessed significant advancements with the integration of machine learning techniques to enhance the precision and predictive reliability of RANS simulations. Initially, approaches in this domain concentrated on the calibration of turbulence model parameters by treating them as random variables, employing Monte Carlo sampling methods to obtain a predictive distribution of outcomes[27], which facilitated a more nuanced understanding of underlying uncertainties. This foundational work set the stage for the direct manipulation of anisotropy components within the Reynolds stress, thereby refining turbulence modeling and enhancing simulation accuracy.

As the application of machine learning in CFD has evolved, several innovative methodologies have emerged. For instance, Edeling et al. [28] utilized Bayesian calibration for multiple turbulence models across diverse flow scenarios, illustrating that model coefficients exhibited considerable variability, underscoring the importance of adaptive, condition-specific calibration strategies. This adaptation ensures that models remain robust across varying operational contexts, thereby enhancing their applicative value.

Simultaneously, Ling et al. [29] developed tensor basis neural networks (TBNNs), which integrate invariance principles to accurately predict the anisotropic tensors of the Reynolds stress. This method represents a significant shift towards embedding machine learning deeply within the fabric of turbulence modeling, leading to more accurate and dynamically responsive CFD simulations.

Further extending the capabilities of machine learning in this area, Kaandorp and Dwight [30] introduced a Tensor Basis Random Forest (TBRF) approach. This methodology, inspired by TBNNs, utilizes random forests—a simpler and potentially more robust machine learning technique compared to deep neural networks—for the prediction of anisotropy tensors. The use of random forests facilitates easier training processes and enhances stability when integrated into CFD solvers. This approach not only streamlines the computational process but also mitigates some of the challenges associated with the high computational costs of deep learning models.

2.3.1. Uncertainty quantification analysis of RANS simulations

There are two primary types of uncertainty that can be modeled in the uncertainty quantification problem: epistemic uncertainty and aleatoric uncertainty [31] [32].

The aleatoric uncertainty refers to the intrinsic noise present in the observations. This includes, for example, sensor noise or motion noise, which engenders uncertainty that cannot be reduced even by gathering supplementary data. In contrast, epistemic uncertainty refers to the uncertainty in the parameters of a model, which represents our limited understanding of the model responsible for producing the observed data. This particular form of uncertainty, sometimes referred to as model uncertainty, can be diminished with adequate data. Aleatoric uncertainty may be also categorised as homoscedastic uncertainty, which remains consistent across various inputs, and heteroscedastic uncertainty, which fluctuates based on the inputs to the model, potentially leading to outputs that are noisier than others.

These types of uncertainties play a crucial role in various modeling and simulation disciplines, including computational fluid dynamics (CFD). In recent years, there has been a growing interest in leveraging machine learning techniques to better quantify and manage these uncertainties, particularly in Reynolds-Averaged Navier-Stokes (RANS) simulations.

In the field of computational fluid dynamics, the integration of machine learning techniques for uncertainty quantification in RANS simulations has seen substantial advancements. Notably, Xiao et al. [33] developed a Bayesian data-driven methodology that leverages high-fidelity observations to iteratively refine Reynolds-stress fields, demonstrating effectiveness even with limited data. However, this approach is constrained to specific flow configurations for which the model was explicitly trained.

Further extending the capability of machine learning in this domain, Wu et al. [34] implemented a method using Mahalanobis distance and kernel density estimation to estimate model confidence. While this technique enables the identification of regions with lower confidence post-training, it is limited to predicting the anisotropic stress and does not provide true probabilistic bounds.

Building on these methods, Geneva et al. [2] employed an invariant Bayesian deep neural network, trained via Stein variational gradient descent, to predict the anisotropic tensor components of Reynolds stress. This model not only enhances the accuracy of RANS predictions but also quantifies uncertainties systematically, capturing both aleatoric and epistemic sources. Despite its advancements, the framework's reliance on comprehensive training data and the computational demands of deep neural networks highlight persistent limitations.

Recent contributions by Tang et al. [3] and Cherroud et al. [35] have further refined this approach, addressing challenges related to complex flow scenarios and sparse data environments, respectively. However, issues with model generalization remain, particularly in cases involving high Reynolds numbers or significant flow separations.

The latest study by Graham et al. [36] introduces BNNs to model both data and uncertainty in predicting closure models for reacting turbulence, showcasing improved generalization capabilities. This work will be used as a reference point for developing our study, aiming to enhance model reliability across diverse flow conditions.

2.4. Research Objective

The linear eddy viscosity theory is the main source of errors in standard RANS turbulence models. The mean strain rate and the Reynolds stress anisotropy tensor are said to be directly correlated under this idea; however, experimental data has shown that this link is not always true. As such, a large amount of data-driven turbulence modeling research is devoted to improving and resolving this assumption.

Nowadays in turbulence modeling problem combine with neural network become more and more popular. However, uncertainty quantification is a larger problem especially for the uncertainty from both the data and model.

Bayesian Neural Networks (BNNs) present a promising approach to estimate and predict modeling uncertainties. They leverage large data volumes, offer fast inference times (relative to Gaussian processes), provide rigorous uncertainty assessments, and demonstrate high expressivity.

This study will build upon Geneva's framework to extend BNNs' capacity to encompass both aleatoric and epistemic uncertainties. The central research question is:

Can the integration of the Bayesian Neural Network framework in data-driven turbulence modeling not only improve the accuracy of RANS results but also capture the dual uncertainties inherent in both the model and the data, surpassing current methods?

The two primary study areas related to this challenge are data-driven turbulence modeling and the Bayiesan neural network architecture. The performance of uncertainty quantification for BNNs will be the primary emphasis, and the outcomes will be evaluated in a data-driven turbulence scenario. In order to address the research issue, several related questions come up:

- Can BNNs both capture the uncertainty from the model and the uncertainty from the data?
- Does predicting uncertainty reduce the prediction accuracy of the mean?
- Does the BNN model have a good results apply to different flow case?

3 Methodology

3.1. Bayesian neural network

In contrast to simple neural networks, Bayesian neural networks provide a probability distribution to its weights rather than a single value or point estimate. These probability distributions can be used to evaluate prediction uncertainty and represent the uncertainty in weights. This is accomplished by using the Bayes theorem to create a posterior distribution by updating the prior distribution of the weights with the data's information. In order to approximate the intractable posterior distribution, training BNNs often includes techniques like variational inference or Markov Chain Monte Carlo (MCMC)[37].

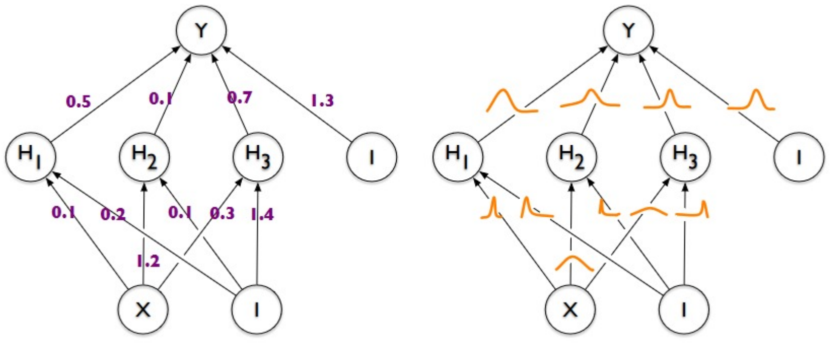


Figure 3.1: Left: each weight has a fixed value, as provided by classical neural network. Right: each weight is assigned a distribution, as provided by Basyian neural network[38]

Consider a deterministic neural net $y = f(x, w)$ with input x, output y, and all parameters w including the weights and biases. The final output for a Bayesian can be viewed as probabilistic model $p(y|\mathbf{x}, \mathbf{w})$. For classification, y is a set of classes and $p(y|\mathbf{x}, \mathbf{w})$ is a categorical distribution. For regression, y is a continuous variable and $p(y|\mathbf{x}, \mathbf{w})$ is a Gaussian distribution. Considering the prior on the weight as a distribution:$p(\mathbf{w})$.Given a training dataset sampled distribution:$\mathcal{D} = \{x^{(i)}, y^{(i)}\}$,The likelihood function can be written as

$$p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{N} p(y^{(i)}|x^{(i)}, w). \tag{3.1}$$

Multiplying the likelihood with a prior distribution $p(\mathbf{w})$ is, by Bayes theorem, proportional to the posterior distribution

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w}). \tag{3.2}$$

Maximizing $p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$ gives the maximum a posteriori (MAP) estimate of $\mathbf{w}$. The usual optimization objective during training is the negative log likelihood. The posterior predictive distribution can be calculated by

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w}. \tag{3.3}$$

3.1.1. Kullback-Leibler divergence

Unfortunately, it is impossible to solve the posterior $p(\mathbf{w}|\mathcal{D})$ analytically in neural networks. Consequently, it has to use a variational distribution $q(\mathbf{w}|\theta)$ to approximate the actual posterior, where θ denotes the BNN's parameters, which have a known functional form and whose parameters need to be estimated. Minimizing the Kullback-Leibler divergence is one way to do this [39].

The KL divergence between the variational distribution $q(\mathbf{w}|\theta)$ and the true posterior $p(\mathbf{w}|\mathcal{D})$ is defined as the following equation

$$\begin{aligned} \mathrm{KL}(q(\mathbf{w}|\theta) \mid\mid p(\mathbf{w}|\mathcal{D})) &= \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{p(\mathbf{w}|\mathcal{D})} d\mathbf{w}, \\ &= \mathbb{E}_{q(\mathbf{w}|\theta)} \log \frac{q(\mathbf{w}|\theta)}{p(\mathbf{w}|\mathcal{D})}. \end{aligned} \tag{3.4}$$

Since it is difficult to write the posterior equation explicitly, the final objective is to minimize the previously specified loss function 3.4. Two variational methods, the Evidence Lower Bound (ELBO) method and the Stein Variational Gradient Descent (SVGD) method separately, will be covered in the next section.

3.1.2. Stein Variational Gradient Descent

In contemporary research on variational inference, the common practice involves confining the approximate posterior within a designated parametric variational family. This approach, while standard, often encounters significant drawbacks, particularly in the context of sampling-based methods, which are notorious for their slow convergence rates and operational challenges. In light of these limitations, this study proposes the adoption of a cutting-edge, non-parametric approach to variational inference, termed Stochastic Variational Gradient Descent (SVGD)[40]. This innovative methodology merges the foundational principles of traditional gradient descent with the advanced efficiency of particle methods, presenting a novel pathway in the optimization of variational inference.

The foundational premise of SVGD is anchored in the utilization of an extensive ensemble of particles, each represented through neural network architectures, to effectively minimize the Kullback-Leibler (KL) divergence. This strategic approach facilitates a more accurate approximation of the target distribution. The ensuing sections of this document are dedicated to a comprehensive exposition of SVGD, delineating its theoretical underpinnings and operational mechanisms in detail, thereby demonstrating its potential as an effective tool in variational inference.

In this case, an initial feasible distribution (e.g., the prior) represented by samples is the source of the variational family, which are distributions produced by performing smooth transformations. Each particle's transformations are stated as follows

$$Trans(\boldsymbol{\theta}) = \boldsymbol{\theta} + \eta\psi(\boldsymbol{\theta}), \tag{3.5}$$

where $\psi(\boldsymbol{\theta}) \in \mathbf{F}$ is the perturbation direction within a function space $\mathbf{F}$, η is the step size, and $\boldsymbol{\theta}$ denotes the neural network's parameters. The transformation term $Trans$ changed the starting density $q(\boldsymbol{\theta})$ to a density near the ultimate goal density if ϵ is small. This can be expressed as follows

$$q_{[Trans]}(\boldsymbol{\theta}) = q\left(Trans^{-1}(\boldsymbol{\theta})\right)\left|\det\left(\nabla Trans^{-1}(\boldsymbol{\theta})\right)\right|. \tag{3.6}$$

The variational posterior is approximated using a particle approximation, which is given by a collection of samples $\{\theta^i\}_{i=1}^S$ with the empirical measure instead of a parametric form. $\mu_S(d\boldsymbol{\theta}) = \frac{1}{S}\sum_{i=1}^S \boldsymbol{\delta}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^i\right)d\boldsymbol{\theta}$. The goal is to guarantee that the measure of the genuine posterior $\nu_p(d\boldsymbol{\theta}) = p(\boldsymbol{\theta})d\boldsymbol{\theta}$ weakly converges to μ. These samples are transformed using $Trans$, and the push-forward measure of μ is indicated as $Trans\mu$.

The intension is to solve the functional optimization issue that follows by identifying the direction that minimizes the KL divergence between the variational approximation and the target distribution

$$\max_{\psi\in\mathbf{F}}\{-\frac{d}{d\epsilon}KL(Trans\mu||\nu_p)|_{\epsilon=0}\}. \tag{3.7}$$

And then it can be derived that [40]

$$-\frac{d}{d\epsilon}KL(Trans\mu||\nu_p)|_{\epsilon=0} = \mathbb{E}_\mu[\mathcal{A}_p\psi], \tag{3.8}$$

where $\mathcal{A}_p$ is called the Stein operator, and it is defined as the following equation

$$\mathcal{A}_p = \frac{\nabla\cdot(p\psi)}{p} = \frac{(\nabla p)\cdot\psi + p(\nabla\cdot\psi)}{p} = (\nabla\log p)\cdot\psi + \nabla\cdot\psi. \tag{3.9}$$

And the expectation $\mathbb{E}_\mu\left[\mathcal{A}_p\phi\right]$ evaluates the difference between p and μ, and its maximum is defined as the Stein discrepancy,

$$S(\mu,p) = \max_{\psi\in\mathcal{F}}\mathbb{E}_\mu\left[\mathcal{A}_p\boldsymbol{\psi}\right]. \tag{3.10}$$

According to Liu [41], the optimal perturbation direction (also known as the Stein discrepancy) has a closed-form solution that is the following equation when the functional space $\mathbf{F}$ is selected to be the unit ball in a product reproducing kernel Hilbert space $\mathcal{H}$ with the positive kernel $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$

$$\boldsymbol{\psi}^*(\boldsymbol{\theta}) \propto \mathbb{E}_{\boldsymbol{\theta}'\sim\mu}\left[\mathcal{A}_p^{\boldsymbol{\theta}'} k\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right)\right] = \mathbb{E}_{\boldsymbol{\theta}'\sim\mu}\left[\nabla_{\boldsymbol{\theta}'} \log p\left(\boldsymbol{\theta}'\right) k\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right) + \nabla_{\boldsymbol{\theta}'} k\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right)\right]. \qquad (3.11)$$

Here $\boldsymbol{\psi}^*(\boldsymbol{\theta})$ means the optimized direction. And finally the algorithm to transform an initial distribution q to the target posterior distribution is written as following algorithm:

Algorithm 1 Bayesian Inference via Stein Variational Gradient Descent

Require: $\left\{\boldsymbol{\theta}_i^0\right\}_{i=1}^n$, a target distribution with density function $p(\boldsymbol{\theta})$, and an initial particle set $\{\boldsymbol{\theta}_i\}_{i=1}^n$, a collection of particles that approximates the target distribution

```
1: for iteration $l$ do
2:     $\boldsymbol{\theta}_i^{\ell+1} \leftarrow \boldsymbol{\theta}_i^{\ell} + \eta_\ell \psi^*\left(\boldsymbol{\theta}_i^{\ell}\right)$
3:     where $\psi^*(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^n \left[k\left(\boldsymbol{\theta}_j^\ell, \boldsymbol{\theta}\right)\nabla_{\boldsymbol{\theta}_j^\ell} \log p\left(\boldsymbol{\theta}_j^\ell\right) + \nabla_{\boldsymbol{\theta}_j^\ell} k\left(\boldsymbol{\theta}_j^\ell, \boldsymbol{\theta}\right)\right]$ and $\eta_\ell$ is the step
       size at the $\ell$-th iteration.
4: end for
```

In this online algorithm, the repulsive force term $\nabla k(\boldsymbol{\theta}, \boldsymbol{\theta}^{'})$ maintains a degree of diversity while the gradient $\psi^*(\boldsymbol{\theta})$ pushes the samples towards the high posterior region by kernel smoothed gradient term $k(\cdot,\cdot)\nabla \log p$. The procedure reduces to the Maximums posterior estimation (MAP) estimate of the posterior when the number of samples reaches 1. Appendix B contains a thorough proof and derivation for the SVGD approach.

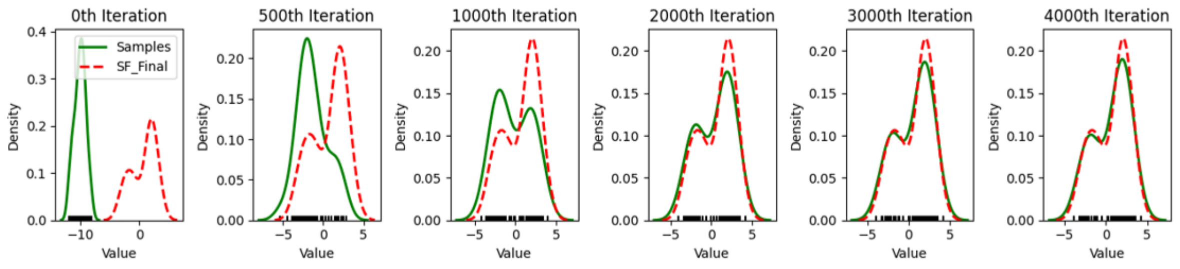


Figure 3.2: Consider a toy example involving a 1D Gaussian mixture. The red dashed lines represent the target density function, while the solid green lines depict the particle densities at various iterations of our algorithm. It is important to note that the initial distribution is intentionally set up with minimal overlap with the target distribution. This example demonstrates how SVGD effectively escapes from the local mode nearby and successfully identifies the more distant mode on the left.

The demonstration illustrated in Figure 3.2 begins with 50 samples drawn from a Normal distribution, specifically $\mathcal{N}(-10, 1)$. The objective is to iteratively transform these samples to conform to a specified target distribution, which in this case is a Gaussian mixture model defined as $\frac{1}{3}\mathcal{N}(-2, 1) + \frac{2}{3}\mathcal{N}(2, 1)$. This transformation process follows the steps outlined in Algorithm 1.

SVGD method used in turbulence modeling

In the section 3.1.2, the mathematical foundations of the SVGD methond have been discussed. However, to use this method in the turbulence modeling, the prior and likelihood distribution forms need to be discussed in this section. The choice ofthe distribution is inspired by the work by [2] and [42].

To set up the network, it is necessary to initially define a prior for the weights. Given the extensive quantity of weights in a fully-connected neural network, we postulate that these weights follow a probability density function characterized by a fully-factorizable zero mean Gaussian, along with a precision scalar α that adheres to a Gamma distribution

$$p(\mathbf{w} \mid \alpha)=\mathcal{N}\left(\mathbf{w} \mid 0, \alpha^{-1} \mathbf{I}_K\right), \quad p(\alpha)=\Gamma\left(\alpha \mid m_0, n_0\right), \tag{3.12}$$

where the form parameters are $n_0 = 0.03$ and the rate is $m_0 = 1$. Moreover, the identity matrix in $R^{n \cdot n}$ is indicated by I_n. The density of a small Student's T distribution centered at zero may be found in the resultant prior. This mitigates the risk of over-fitting and encourages sparsity [43]. Such a prior places minimal limitation on the network's final functional shape because of the very non-linear nature of the neural network and the application of a significant number of weights [44].

To get the final distribution, the output of the neural network should include both the mean and the variance. And for the final output structure, it include both of them. The details of such neural network output can be found in the following section in section 3.4.1 and the section 3.1.4. Therefore, the output of the network can be written as

$$output = [f(x, w), \sigma^2(x, w)], \tag{3.13}$$

where $y = f(x, w) + \epsilon$ is the system's output, with $\epsilon \in \mathcal{N}(0, \sigma^2(x_i, w))$. To capture the aleatoric uncertainty for each input and coefficient, consider the situation $\sigma^2(x, w)$. In order to impose the positive variance requirement in practice, the second component of the neural net output needs be transformed using a softplus algorithm: $\sigma^2 = log(1 + exp(\cdot)) + eps$, with numerical stability provided by $eps = 10^4$. The likelihood equation can be expressed as follows if the likelihood has a normal distribution.

$$p(\mathcal{D}|\mathbf{w})=\prod_{i=1}^{M}\left[\mathcal{N}\left(\boldsymbol{y}_i \mid \boldsymbol{f}\left(\boldsymbol{x}_i, \mathbf{w}\right), \sigma^2(x_i, w)\right)\right]. \tag{3.14}$$

Multiply equation 3.14 with the prior equation 3.12 (using the Bayes theorem), the posterior equation can be written as

$$p(\mathbf{w}, \mathcal{D})=\prod_{i=1}^{M}\left[\mathcal{N}\left(\boldsymbol{y}_i \mid \boldsymbol{f}\left(\boldsymbol{x}_i, \mathbf{w}\right), \sigma^2(x_i, w)\right)\right] \mathcal{N}\left(\mathbf{w} \mid 0, \alpha^{-1} \mathbf{I}_K\right) \Gamma\left(\alpha \mid m_0, n_0\right). \tag{3.15}$$

In practice, it is often more convenient to use the negative log-likelihood (NLL) as the final optimization objective. By taking the logarithm of both sides of equation 3.15, the multiplication terms are converted into addition, simplifying the expression. This transformation can significantly accelerate the training process of neural networks. The negative log-likelihood is then expressed as

$$\log p(\mathcal{D}|\mathbf{w})=\sum_{i=1}^{M}(\log \frac{1}{\sigma(x_i, w)\cdot\sqrt{2\pi}}-\frac{1}{2}(\frac{f(x_i, w)-y_i}{\sigma(x_i, w)})^2). \tag{3.16}$$

The typical radial basis function kernels are selected for $k(\theta, \theta')$ in this work. This formulation yields a straightforward updating process where algorithm 1 computes and updates the optimal decent direction for all particles. Next, the predicted mean can be estimated using Monte Carlo approximations

$$\begin{aligned}\mathbb{E}\left(y \mid x, \mathcal{D}\right) &= \mathbb{E}_{p(\mathbf{w},\beta|\mathcal{D})}\left(\mathbb{E}\left(y \mid x, \mathbf{w}, \beta\right)\right) \\ &= \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}\left(\boldsymbol{f}\left(x, \mathbf{w}\right)\right) \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{f}\left(x, \mathbf{w}_i\right).\end{aligned} \tag{3.17}$$

Furthermore, the variance is calculated similarly as the equation 3.22:

$$\mathrm{Var}(\mathbf{y}) = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\sigma(x_i, w)] + \mathrm{Var}\left(\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\mathbf{y} \mid \mathbf{x}, \mathcal{D}]\right). \tag{3.18}$$

3.1.3. Evidence lower bound (ELBO) objective function

As stated in the preceding section 3.1.1, the objective of training BNNs is to reduce the KL divergence between the weight distribution and the actual Bayesian posterior, contingent upon the dataset $\mathcal{D}$. In addition to the SVGD approach, the expansion and rearrangement equation 3.4 provides the evidence lower bound (ELBO) objective function[45]. This objective function can serve as an alternative cost function to minimise the KL divergence. The ELBO function can be written as

$$\begin{aligned}\mathrm{KL}(q(\mathbf{w}|\boldsymbol{\theta}) \mid\mid p(\mathbf{w}|\mathcal{D})) &= \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\theta})} \log \frac{q(\mathbf{w}|\boldsymbol{\theta})}{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}, \\ &= \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\theta})}\left[\log q(\mathbf{w}|\boldsymbol{\theta}) - \log p(\mathcal{D}|\mathbf{w}) - \log p(\mathbf{w})\right].\end{aligned} \tag{3.19}$$

And function 3.19 can be formally rewritten as

$$\theta^* = \arg\min_{\theta} \mathrm{KL}[q(\mathbf{w}|\theta) \| p(\mathbf{w}|\mathcal{D})] = \arg\min_{\theta} \underbrace{\mathrm{KL}[q(\mathbf{w}|\theta) \| p(\mathbf{w})]}_{\text{prior-informed}} - \underbrace{\mathbb{E}_{q(\mathbf{w}\theta)}[\log p(\mathcal{D}|\mathbf{w})]}_{\text{data-informed}}. \tag{3.20}$$

The initial component of equation 3.20 represents the KL divergence between the prior distribution and the learnt distribution of the BNN parameters. The second part in the equation denotes the data misfit, which is measured by the anticipated negative log-likelihood of the data across the distribution of credible models, as determined by the weight assignment. Thus, the objective function may be estimated by selecting samples $\mathbf{w}^{(i)}$ from the set $q(\mathbf{w}|\boldsymbol{\theta})$

$$ELBO \approx \frac{1}{N}\sum_{i=1}^{N}\left[\log q(\mathbf{w}^{(i)}|\boldsymbol{\theta}) - \log p(\mathbf{w}^{(i)}) - \log p(\mathcal{D}|\mathbf{w}^{(i)})\right]. \tag{3.21}$$

3.1.4. Uncertainty quantification

As mentioned in section 2.3.1, the uncertainty needed to be calculated is aleatoric and epistemic uncertainty. While the main emphasis of authors using closure models in CFD solvers is on epistemic uncertainty, it is equally important to assess aleatoric uncertainty. Since BNNs ultimately train a collection of neural networks, the epistemic uncertainty can be easily calculated from variance of the expected outputs of the networks. To calculate the aleatoric uncertainty, an effective strategy to do this is by adjusting the parameters of each output dimension to account for heteroscedastic uncertainty present in the data [46] [47]. In this work, the output random variable is modelled using the Gaussian form derived from [48].

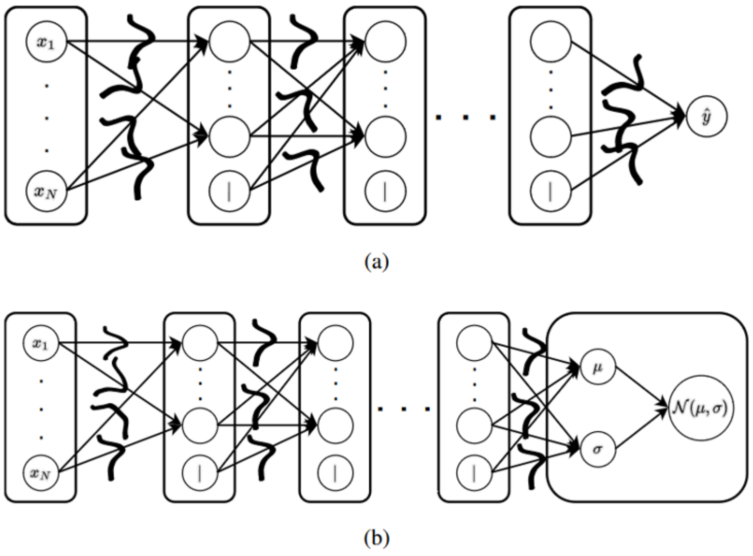


Figure 3.3: (a) A BNN that only accounts for epistemic uncertainty, and (b) a BNN that accounts for both epistemic and aleatoric uncertainty[36].

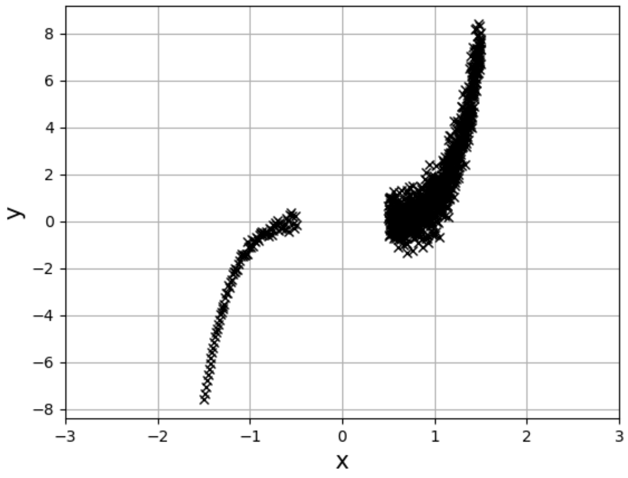


Figure 3.4: The left part of this dataset is characterised by the epistemic uncertainty with few data. And the right part of the dataset is characterised by the aleatoric uncertainty with large amount data. This kind of dataset can show the ability of the method to capture such two uncertainty.

The structural difference between a neural network that can describe epistemic uncertainty and one that can model both epistemic and aleatoric uncertainty is demonstrated in Figure 3.3. In the case of [48] type architecture, the last layer of the output is mapped to $(\mathbf{y}_{\boldsymbol{\mu}}, \mathbf{y}_{\boldsymbol{\sigma}})^T \in \mathbb{R}^{2d_o}$ in order to capture the aleatoric uncertainty. The final output variable is then parameterised as $\mathbf{y} \in \mathbb{R}^{d_o} \sim \mathcal{N}(\mathbf{y}_{\boldsymbol{\mu}}, \operatorname{diag}(\mathbf{y}_{\sigma}))$ by $\mathbf{y}_{\boldsymbol{\mu}}, \mathbf{y}_{\boldsymbol{\sigma}}$. The law of total variance [49] may be used to break down the overall variance (uncertainty) in the predictions into its epistemic and aleatoric components within the context of such a neural network. The following equation [50] represents this division

$$\operatorname{Var}(\mathbf{y}) = \mathbb{E}_{q(\mathbf{w}|\theta)}[\operatorname{Var}(\mathbf{y} \mid \mathbf{x}, \mathcal{D})] + \operatorname{Var}\left(\mathbb{E}_{q(\mathbf{w}|\theta)}[\mathbf{y} \mid \mathbf{x}, \mathcal{D}]\right). \tag{3.22}$$

The predictive variance $\operatorname{Var}(\mathbf{y})$ is divided into an aleatoric component $\mathbb{E}_{q(\boldsymbol{w}|\theta)}[\operatorname{Var}(\mathbf{y} \mid \mathbf{x}, \mathcal{D})]$ and an epistemic component $\operatorname{Var}\left(\mathbb{E}_{q(\boldsymbol{w}|\theta)}[\mathbf{y} \mid \mathbf{x}, \mathcal{D}]\right)$. The aleatoric uncertainty is calculated by the mean of all the standard deviation outputed by each neural netowr. And the epistemic uncertainty is easily equals to the variability of the model's mean predictions. For a given $\mathbf{w} \sim q(\mathbf{w} \mid \theta)$, the predictive mean is given by the first network output, $\mathbf{y}_{\mu}$. The methods for calculating this two kind of uncertainties are detailed and discussed in the Appendix C.

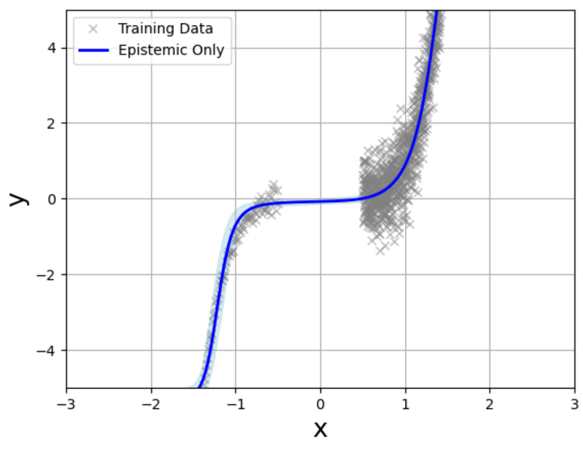


Figure 3.5: Model with only epistemic uncertainty

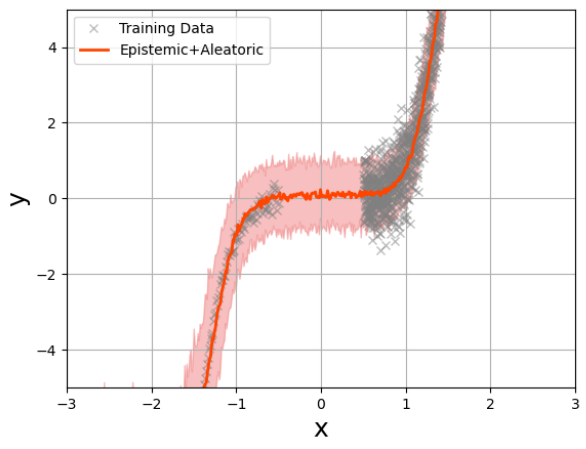


Figure 3.6: Models with both aleatoric and epistemic uncertainty

As an illustration, consider Figures 3.5 and 3.6, which display the magnitudes of the uncertainties for the one-dimensional example (Figure 3.4). The following equations are used to create the training data

$$y = x^5 + 0.1(1.6 + x)\varepsilon, \tag{3.23}$$

where ε is a function defined as normal distribution $\mathcal{N}$ with mean equals to 0 and standard deviation equals to σ^2. Within this particular framework, the level of noise in the data escalates in correlation with the variable x. In contrast to the 600 data points in the region on the right, only 60 data points are retained in the left zone. The blue line in figure 3.5illustrates the model's predictions only from an epistemic uncertainty perspective, which specifically accounts for the uncertainty arising from the model's parameters.

Greyscale representations of the training data points are displayed. The proposed model well represents the fundamental function, but its level of uncertainty is negligible in areas with ample data and rises in places with little data, such as those located to the left of the origin. Furthermore, picture 3.6 introduces a model that incorporates both epistemic and aleatoric uncertainties. The red line denotes the forecasts made by the model, while the shaded region represents the collective uncertainty. The aleatoric uncertainty, which considers the intrinsic noise in the data, broadens as the variable x rises, as shown by the expansion of the shaded region. The data points used for training are once again displayed in grey. This design incorporates both the uncertainty caused by model parameters and the intrinsic variability in the data, leading to a wider range of uncertainty, particularly in areas with significant data noise.

3.1.5. Discussion on the selection of prior distribution for weights

Since the first term of the ELBO 3.20 represents the prior distribution over the weights, which significantly influences the loss function, it is crucial to select an appropriate prior distribution. An improper prior can severely impact model performance, particularly when the data is limited. In extreme cases, prior misspecification can only be mitigated in the asymptotic limit of an infinitely large dataset, where the model's posterior distribution will eventually converge to the true distribution. However, in practical scenarios with finite data, the choice of prior remains a critical factor in ensuring robust and accurate model inference [51].

3.2. Introduction to Extreme learning machine

Extreme learning machines offer a potentially useful approach to mitigate the uncertainty of results that is exacerbated by an excessive number of neural network parameters. In the ELM algorithm, the hidden nodes are initially defined randomly and then fixed without undergoing iterative tweaking. It is sufficient to learn only the weights between the hidden layer and the output layer, resulting in a significant reduction in the amount of variable parameters that impact the final outcome[52]. Figure 3.7 illustrated structure of ELM.

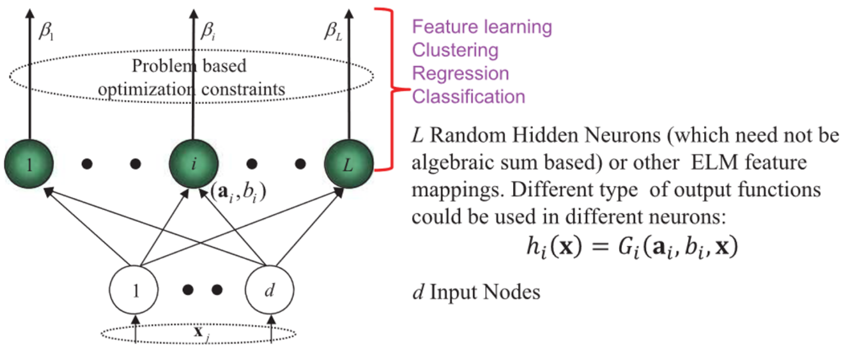

β_1
β_i
β_L
Problem based optimization constraints
Feature learning
Clustering
Regression
Classification
1
i
L
$(\mathbf{a}_i, b_i)$
L Random Hidden Neurons (which need not be algebraic sum based) or other ELM feature mappings. Different type of output functions could be used in different neurons:
$h_i(\mathbf{x}) = G_i(\mathbf{a}_i, b_i, \mathbf{x})$
1
d
d Input Nodes
$\mathbf{x}_j$

Figure 3.7: Diagram of the basic ELM architecture. A variety of computational node types may be combined to form the hidden nodes in ELM [52].

Based on the diagram shows in figure 3.7 and the mathematical formulation for neural networks, the output function of ELM can be written as the following equation

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\alpha}_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\alpha}. \tag{3.24}$$

Consider the set $\alpha = [\alpha_1, ..., \alpha_L]^T$. The production weight vector between the hidden layer of L nodes and the $m \geq 1$ output nodes is denoted as $h(x) = [h_1(x), ..., h_L(x)]$. represents the row-wise output vector of the hidden layer in relation to the input x. An output of the i_{th} hidden node is denoted as $h_i(x)$. Specifically, in practical scenarios, $h_i(x)$ can take form

$$h_i(\mathbf{x}) = G\left(\mathbf{a}_i, b_i, \mathbf{x}\right), \quad \mathbf{a}_i \in \mathbf{R}^d, b_i \in R, \tag{3.25}$$

The activation function for each hidden layer, such as the sigmoid function, is denoted as $G(a, b, x$. Moreover, a, b represents the weights and bias parameter associated with each layer. Inside the ELM framework, the hidden node parameters (a, b) are produced randomly, regardless of the training data, based on a continuous probability distribution rather than being explicitly taught. Initially setting all the hidden layer settings is the first step of the ELM process. Given that, obtaining the learnable parameter α becomes a straightforward linear inverse task. Using the linear algebra knowledge, Problem can be set as follows

$$\mathbf{H}\alpha = Y, \tag{3.26}$$

where $\mathbf{H}$ is the hidden layer output matrix

$$\mathbf{H} = \begin{bmatrix} h_1\left(x_1\right) & \cdots & h_L\left(x_1\right) \\ \vdots & \vdots & \vdots \\ h_1\left(x_N\right) & \cdots & h_L\left(x_N\right) \end{bmatrix}, \tag{3.27}$$

and Y is the training data target matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix} = \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \vdots & \vdots \\ y_{N1} & \cdots & y_{Nm} \end{bmatrix}. \tag{3.28}$$

And the optimal solution to α is simplified given by using the Moore–Penrose generalized inverse of matrix method

$$\alpha = \mathbf{H}^{\dagger}\mathbf{T}, \quad \mathbf{H}^{\dagger} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T. \tag{3.29}$$

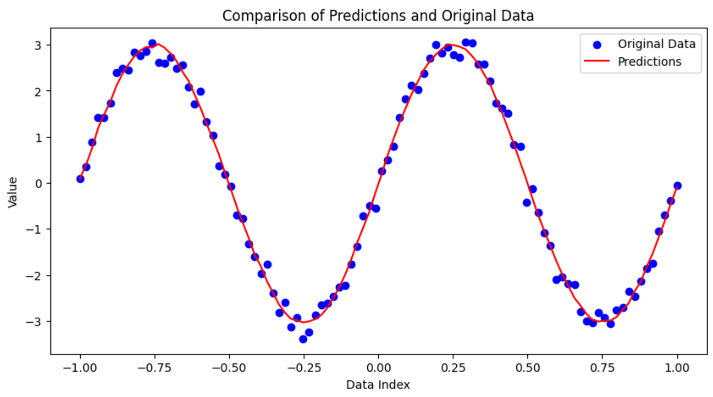


Figure 3.8: Regression from the sinusoidal data with added noise . And the neural network only have one hidden layer and using the noraml distribution $\mathcal{N}(0, 1)$ to initially get the weights for hidden layer.

Employing the previously outlined approach, a simplified example was calculated. This neural network design consists of a single input layer, one hidden layer, and one output layer. Initialisation of all weights and biases was performed using a normal distribution $\mathcal{N}(0,1)$. Finally, the hidden layer parameters were set to a constant value, and the ReLU function was chosen as the activation function. An equation denoted as 3.29 was used to establish the parameter connecting the hidden layer to the output layer. Finally, the training data is computed using the following equation

$$y = 3sin(x) + \varepsilon, \tag{3.30}$$

where $x \in (-1,1)$ and $\varepsilon \in \mathcal{N}(0,0.5)$. The outcomes are presented in Figure 3.8. The figure visually confirms that the ELM model effectively captures the underlying pattern of the noisy sinusoidal data, as evidenced by the close alignment between the predicted values and the original data points.

The next phase involves adapting the ELM concept to SVGD-BNN. A challenge is encountered because equation 3.29 is unsuitable for obtaining the parameter α, as it does not provide information regarding uncertainty. The proposed approach is to fix all parameters in the neural network's hidden layer during the training phase, while employing the SVGD method exclusively to update the parameters between the hidden layer and the output layers. This implies that θ in algorithm 1 includes only the parameters for the final layer, which means

$$\{\boldsymbol{\theta}_i\}_{i=1}^N, \boldsymbol{\theta}_i = \{\alpha_i\}, \tag{3.31}$$

where N is the number of the neural networks.

3.2.1. Extreme learning machine combine with Bayesian neural network

The detailed mathematical formulation to integrate the ELM with the SVGD method will be elaborated in the subsequent section, aligning the formulation with the discussions in section 3.1.2. Given that this scenario involves quantifying uncertainty, the output function, as defined in equation 3.24, will incorporate data noise as follows

$$T = \mathbf{h}(x)\alpha + \varepsilon. \tag{3.32}$$

Here ε is from the following distribution

$$\begin{aligned} p(\boldsymbol{\varepsilon}) &= \mathcal{N}\left(\boldsymbol{\varepsilon} \mid 0, \beta^{-1}\mathbf{I}_9\right), \\ p(\beta) &= \operatorname{Gamma}\left(\beta \mid a_1, b_1\right). \end{aligned} \tag{3.33}$$

And then the likelihood can be written as

$$p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{M}\left[\mathcal{N}\left(t_i \mid h\left(x\right)\alpha, \beta^{-1}\right)\right]\Gamma\left(\beta \mid a_1, b_1\right). \tag{3.34}$$

With the same prior written in equation 3.12, the posterior can be written as

$$p(\alpha \mid \mathcal{D}) = \prod_{i=1}^{M}\left[\mathcal{N}\left(t_i \mid h\left(x\right)\alpha, \beta^{-1}\right)\right]\mathcal{N}\left(\alpha \mid 0, \eta^{-1}\mathbf{I}_K\right)\Gamma\left(\eta \mid a_0, b_0\right)\Gamma\left(\beta \mid a_1, b_1\right). \tag{3.35}$$

The reason for using such prior is for comparison with the results from BNN svgd method. Futhermore, the inital weight for the hidden layer's parameters is come from the normal distribution $\mathcal{N}(0,1)$. After getting the above posterior formulation, SVGD can be used to update the α. And the results of this intergated method will discuss in section 4.1.1.

Training from the same data as figure 3.7, the results are given by figure 4.3. It can be concluded that the the new combine method demonstrates commendable efficiency in data regression. Furthermore, the analysis reveals a notable expansion in the uncertainty region in the absence of data, corroborating the anticipated theoretical predictions.

3.3. Corrections for turbulence model

In this study, the k$-\omega$ SST model, as outlined in Section 2.1.4, is selected for applications. Nevertheless, in order to address the constraints of this model, two correction terms will be included, as outlined in SpaRTA by Schmeltzer[53]. The approach implemented an additional correction term to the conventional anisotropy tensor. An enhanced constitutive relation is formed by equating the residual for the constitutive relation to an additive term b_{ij}^{Δ}

$$b_{ij} = -\frac{\nu_t}{k} S_{ij} + b_{ij}^{\Delta}. \tag{3.36}$$

Editing the anisotropy tensor will impact the generation of turbulent kinetic energy. Therefore, a second correction term, denoted as R, is included to account for this modification. The equations that regulate the variables k and ω in this improved k$-\omega$ SST model are as specified below

$$\partial_t k + U_j \partial_j k = \partial_j \left[\left(\nu + \sigma_k v_t \right) \partial_j k \right] + P_k + R - \beta^* \omega k, \tag{3.37}$$

$$\partial_t \omega + U_j \partial_j \omega = \partial_j \left[\left(\nu + \sigma_\omega \nu_t \right) \partial_j \omega \right] + CD_{k\omega} + \frac{\gamma}{\nu_t} \left(P_k + R \right) - \beta \omega^2. \tag{3.38}$$

Equations 3.37 and 3.38 utilize b_{ij}, k, and u from LES or DNS to acquire values of ω and R. The k-corrective frozen RANS technique is used to calculate the magnitude of b_{ij}^{Δ} and R. With the use of the LES or DNS data and the value of ν_t obtained via this freezing method, the required adjustment b_{ij}^{Δ} is found.

3.3.1. Non-linear eddy viscosity model

A modeling strategy needs to be chosen in order to find adjustments for the model-form errors b_{ij}^{Δ} and R. Additionally, the Bayesian neural network will simulate both the term and the uncertainty. Numerous prior works [29] [33] have employed a similar methodology. The underlying premise is that the anisotropy of the Reynolds stress b_{ij} relies on both the rotation rate tensor $\Omega_{ij} = \frac{1}{2\omega}(\partial_j U_i - \partial_i U_j)$ and the strain rate tensor $S_{ij} = \frac{1}{2\omega}(\partial_j U_i + \partial_i U_j)$. According to the Cayley-Hamilton theorem, the anisotropic component of the Reynolds stress may be written in the following general form

$$b_{ij} \left(S_{ij}, \Omega_{ij} \right) = \sum_{n=1}^{N} T_{ij}^{(n)} \alpha_n \left(I_1, \ldots, I_5 \right). \tag{3.39}$$

Composed of five interconnected invariant I_m and ten nonlinear base tensors $T_{ij}^{(n)}$. Additionally, this method's specifics are covered in [54]. Furthermore, this research exclusively considers two-dimensional flow scenarios, where the first three base tensors constitute a linearly independent basis and only the first two invariants are non-zero.Duraisamy (2019) Turbulence. Thus, the collection of invariants and base tensors may be formulated as

$$\begin{aligned} T_{ij}^{(1)} &= S_{ij}, T_{ij}^{(2)} = S_{ik}\Omega_{kj} - \Omega_{ik}S_{kj}, \\ T_{ij}^{(3)} &= S_{ik}S_{kj} - \frac{1}{3}\delta_{ij}S_{mn}S_{nm}, \\ I_1 &= S_{mn}S_{nm}, I_2 = \Omega_{mn}\Omega_{nm}. \end{aligned} \tag{3.40}$$

The method may be operationalized by substituting equations 3.40 into equation 3.39. Only the coefficient functions α_n requires investigation at this juncture. According to Leschziner [55], the transport equations for turbulent quantities incorporating convection and diffusion components can only consider nonlocal effects when considering the normal stresses $\frac{2}{3}k\delta_{ij}$. This limitation arises from the restrictions imposed by the local closure hypothesis. The term R contributes local information to rectify the transport equations. Dependent on the local sign of R, it either increases or decreases the net production P_k locally. Consequently, it functions as an additional term for production or dissipation that aids in minimizing the error in k. Moreover, it may be modeled in a related manner to the development of turbulence

$$R = 2kb_{ij}^{\Delta}\partial_j U_i + \alpha_4\epsilon, \tag{3.41}$$

where $\epsilon = \omega k$, $\alpha_4 = f(I_1, I_2)$. Adding this term to correct R is justified by equation 3.37, as ωk significantly influences R. With equation 3.41, the nonlinear eddy viscosity framework can also model R. Furthermore, by applying the base tensor and invariants as described in equation 3.40, the remaining task involves using a neural network to determine the appropriate functions $\alpha_n(I_1, I_2)$ for $n = 1, \ldots, 4$ to correct the model-form error. This approach leverages the neural network's capability to learn complex mappings, thereby improving the accuracy of the turbulence model by addressing the discrepancies in the predicted turbulent stresses.

3.4. Tensor basis neural network

Tensor-based neural networks may be created based on the theoretical component on non-linear eddy viscosity found in section 3.3.1. Furthermore, [29] is the first to propose this concept. Using the symmetric and antisymmetric tensor components of the velocity gradient tensor, this neural network predicts the anisotropic tensor of the Reynolds stress. The neural network accomplishes both Galilean invariance and transformation coordination invariance by using tensor invariants. As a result, this model works quite well for forecasting flows whose geometries are different from those in the training dataset.

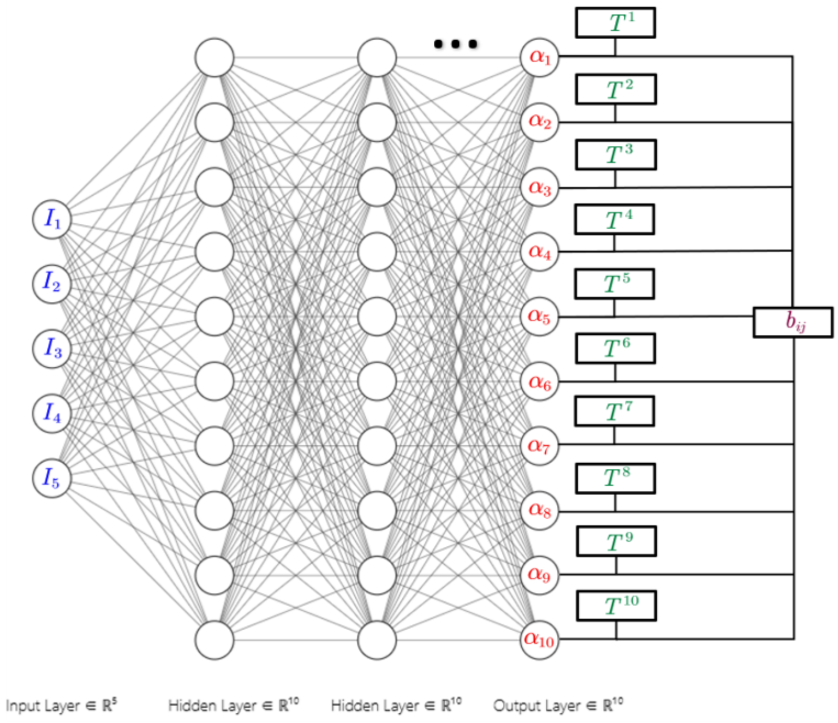


Figure 3.9: Invariant, fully-connected neural network architecture proposed by [29], with some connections omitted for clarity. Scalar values are represented by circles, while 3×3 second-order tensors are represented by rectangles.

The anisotropic term in the neural network model is modeled by using the linear combination found in Equation 3.39. The tensor basis functions and invariants in Equations 3.40 are employed in place of the components of the symmetric and antisymmetric tensors (s and ω), as shown in Figure 3.9 (all ten tensors and invariants are depicted). Because of the neural network's structure, the model may be (a) Galilean invariant, which is accomplished by using the rotation tensor and rate-of-strain functions of the velocity gradient. (b) By using the invariant inputs I_i, invariance to coordinate transformations is preserved (c) Since every variable is nondimensionlized, any activation function and scaling technique may be applied. In addition, this model of eddy viscosity is the most generic formulation; that is, it is not bound by the restrictions of simpler models that dictate the anisotropic term's shape. However, a fundamental premise of this model is that the invariants I_i can provide a thorough representation of the mapping between the RANS and LES physical domains. More input characteristics may jeopardize coordinate system invariance, which would limit the model's capacity to generalize, however this is not guaranteed.

3.4.1. Tensor basis Bayesian neural network

In this work, the final prediction target is R as described in Equation 3.41. Integrating the discussion from Section 3.4 on the tensor basis neural network and Section 3.1.4 on capturing both epistemic and aleatoric uncertainty through the neural network, the structure for the tensor basis Bayesian neural network is developed to obtain the final

R. For the 2D flow case, Equation 3.41 can be reformulated as

$$R = \frac{Dk}{Dt} = 2kb_{ij}\partial_i U_j + \epsilon = 2k \sum_{n=1}^{3} T_{ij}^{(n)} \alpha_n \left(I_1, I_2\right) \partial_i U_j + \alpha_4(I_1, I_2)\epsilon. \tag{3.42}$$

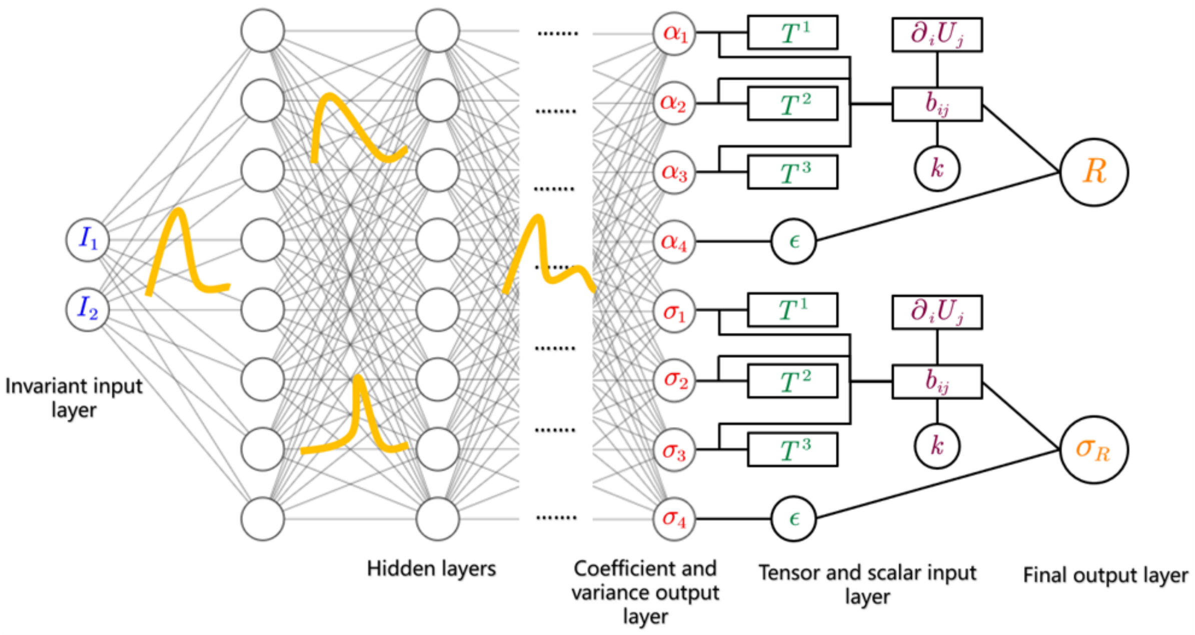


Figure 3.10: Invariant, fully-connected Bayesian neural network architecture with output aleatoric uncertainty for 2D flow case. Circles denote scalar values, while rectangles represent 3×3 second-order tensors.

To model the aleatoric uncertainty, the neural network architecture is extended by adding an additional unit for each coefficient α_n in the final layer, which corresponds to the standard deviation σ_{α_n} of each coefficient. Figure 3.10 shows the fully neural network structure in 2D flow case. The structure can be divided into two parts. The first part utilizes the invariants as the input layer to obtain both the tensor coefficients and their respective variances. The second part involves a tensor and scalar input layer, which multiplies the coefficients to derive the final b_{ij}. Finally, b_{ij} is multipled by the the velocity gradient and added to the turbulence kinetic energy to yield the final R.

Assuming that each α_n $(n = 1, ..., 4)$ is independent, the principle of uncertainty propagation can be applied to compute the final aleatoric uncertainty for R. This calculation is based on the propagation of variances through the model, as detailed in [56]. The resulting equation for the final aleatoric uncertainty is presented as follows

$$\sigma_R = \sqrt{4k^2 \sum_{n=1}^{3} (T_{ij}^{(n)})^2 \sigma_{\alpha_n} (\partial_i U_j)^2 + \sigma_{\alpha_4} \epsilon^2}. \tag{3.43}$$

Utilizing the neural network structure described in this section, it is possible to effectively retain the benefits of the tensor basis neural network presented in Section 3.4, while also accurately calculating the epistemic and aleatoric uncertainty of R. Such a neural network structure is also one of the innovations of this thesis.

4 Experimental Setup

In this chapter, the selection of the optimal method for enhancing the performance of the Bayesian neural network will be examined. This evaluation will consider both one-dimensional (1D) data and turbulence datasets to ascertain which method performs more effectively under varying conditions. Following the determination of the most suitable optimization method, an extensive hyperparameter search will be conducted. This process will involve a comprehensive exploration of the hyperparameter space to identify the optimal configuration, ultimately defining the final neural network architecture. The objective is to achieve an optimal balance between model complexity and generalization capability, thereby ensuring robust and precise predictive performance across different data types.

4.1. Method selection

The optimization objective for Bayesian neural networks is to minimize the KL divergence, as demonstrated in Equation 3.4. Two methods for achieving this have been introduced in Section 3.1.2 and Section 3.1.3. This section focuses on determining which method is most effective for minimizing KL divergence. Additionally, one-dimensional data and turbulence data are utilized to validate the accuracy of the final predictions and the precision of the uncertainty estimations.

4.1.1. Test on one-dimensional data

The simple one-dimensional data is created by the following equation

$$y = x^3 + 0.1(1.5 + x)\varepsilon, \tag{4.1}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $\sigma = 0.25$. This equation closely resembles the data example in Section 3.1.4, with the substitution of x^5 by x^3. This modification is motivated by the need for reduced computational time and enhanced contrast effects. Similarly, only 60 data points are retained in the left region, in contrast to 600 data points in the right region. This configuration ensures that both epistemic and aleatoric uncertainties can be effectively evaluated. Additionally, it facilitates the assessment of model performance in the transition area, thereby providing a comprehensive evaluation of the neural network's predictive capabilities under varying data densities. This setup is designed to challenge the model's ability to generalize and accurately capture uncertainty across different regions, ensuring robust performance in practical applications.

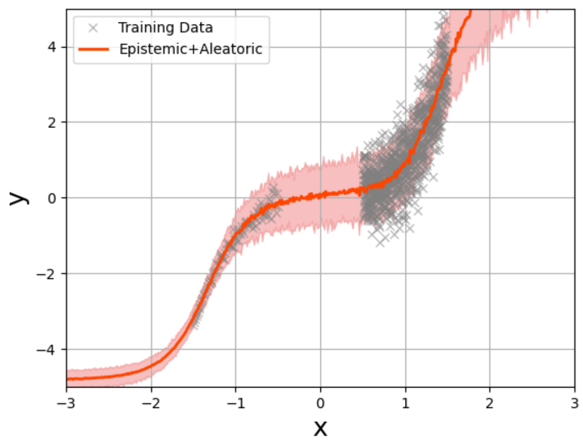


Figure 4.1: Models with both aleatoric and epistemic uncertainty using ELBO method

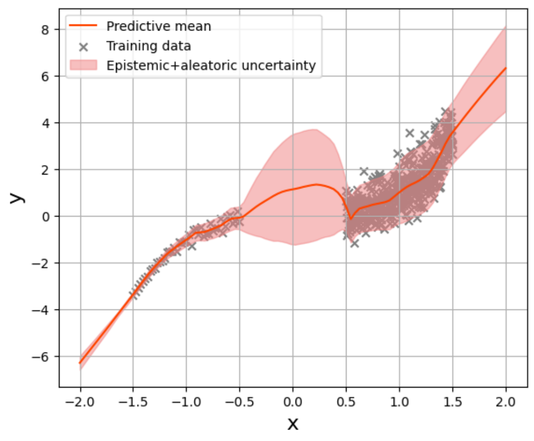


Figure 4.2: Models with both aleatoric and epistemic uncertainty using SVGD method

The figures illustrate the modeling of both aleatoric and epistemic uncertainty using two distinct methods: ELBO (Figure 4.1) and SVGD (Figure 4.2). Both methods were applied to the same training and prediction datasets, utilizing an identical network structure. The two approaches effectively capture data uncertainty, as indicated by the shaded regions surrounding the predictive means. The ELBO method, however, presents a more narrowly bounded uncertainty region compared to the SVGD method, suggesting potentially higher confidence in its predictions. Conversely, the SVGD method exhibits a broader uncertainty range, particularly noticeable in the interval between -0.5 and 0.5, reflecting a more conservative stance in accounting for model uncertainty. This is especially evident in the region devoid of training data, where large epistemic uncertainty is present. The SVGD method's larger uncertainty region than the ELBO method indicates a tendency to assign greater uncertainty to areas lacking data. Both methods demonstrate predictive means that closely align with the training data trend, with slight variations in uncertainty expression. Overall, while both techniques are effective in uncertainty modeling, the ELBO method appears to provide more precise estimates, whereas the SVGD method offers a more cautious representation.

1-D Results for extreme learning machine

Training from the same data as Figure 3.8, the first simple results are given by Figure 4.3. It can be concluded that the the new combine method demonstrates commendable efficiency in data regression. Furthermore, the analysis reveals a notable expansion in the uncertainty region in the absence of data, corroborating the anticipated theoretical predictions.

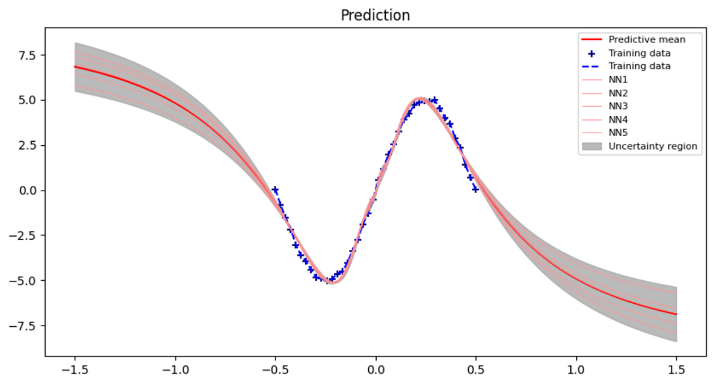


Figure 4.3: Outcomes from Regression of Simplified Training Data Utilizing ELM Combined with SVGD Method.

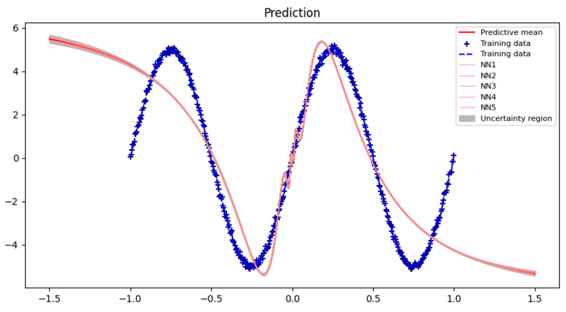


Figure 4.4: Regression by ELM-SVGD with more training data(a more complex data conidition and 5000 nodes for each layer)

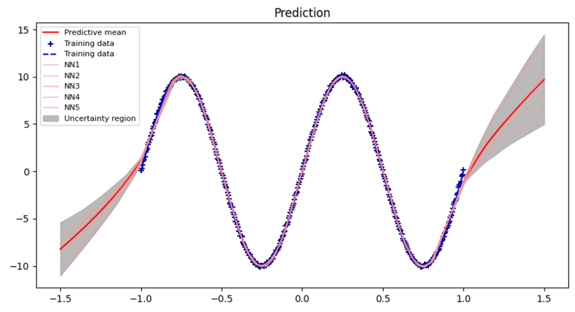


Figure 4.5: Regression by BNN-SVGD with more training data(a more complex data conidition)

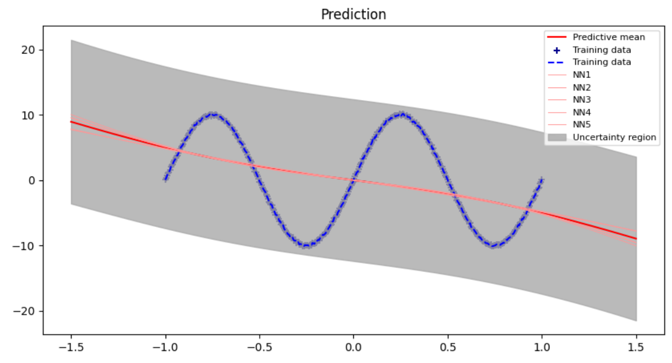


Figure 4.6: Regression from the same training data as figure 4.4 with one hidden layer and using the uniform distribution $\mathcal{U}(-1.5, 1.5)$ to initially get the weights for hidden layer.

To further assess the efficacy of the ELM-SVGD method, the training data was made more complex by introducing an additional sin cycle, which displayed higher volatility, as depicted in Figure 4.4. The outcomes did not show improvement in regression, even after increasing the number of nodes in both hidden and output layers, compared to those shown in Figure 4.3. Additionally, when compared to Figure 4.5, which represents results computed using the original BNN-SVGD method, the regression capability of

ELM-SVGD was found to be inferior. This can be attributed to the inherent simplicity of ELM, which becomes a limitation when processing complex, high-dimensional data patterns or intricate non-linear relationships that demand a more sophisticated analysis. An experiment with a single hidden layer, where weights for the hidden layer were drawn from a uniform distribution, is shown in Figure 4.6. Regrettably, this approach yielded even poorer results. Given the increased complexity inherent in turbulence data, the ELM-SVGD method combined with BNN is no longer considered a viable option.

4.1.2. Test on turbulence data

The turbulence scenario used to assess the approach is the Periodic Hills (PH) flow, characterized by the movement of fluid through a channel with uniformly distributed hills on the lower surface, as seen in Figure 4.7. This instance is widely used to evaluate the effectiveness of RANS turbulence modeling. The high-fidelity data for this scenario is obtained from a Large Eddy Simulation (LES) carried out by Breuer et al.[57] at a Reynolds number of 10,595. A revised mesh consisting of 120 x 130 cells is used to conduct the related RANS simulations. Cyclic boundary conditions are implemented at both the entrance and exit to replicate a channel of indefinite length, therefore guaranteeing the presence of periodicity in the turbulent flow. An extensive comparison between LES and RANS findings is facilitated by this configuration, therefore confirming the accuracy and resilience of the approach in forecasting turbulence properties.

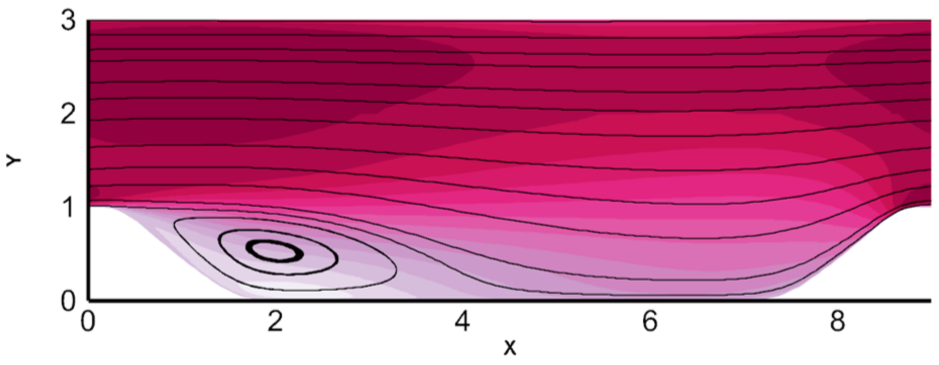


Figure 4.7: The flow domain of the PH_{10595} case is illustrated with streamlines, colored according to the magnitude of velocity in the x-direction and y-direction, based on LES data.

SVGD set up

In section 3.1.2, the mathematical foundation for turbulence modeling question have been discussed. Here, y is the TKE correction term R and the x are different feature to input. The invariant inputs to the neural network tended to vary strongly in magnitude including very large values near fixed boundaries. To increase training performance and efficiency, there two techniques are included:

- At the beginning of each epoch, training points are randomly permuted and then partitioned into mini-batches. This allows for the data from many flows to be integrated into a single mini-batch. The use of this method reduces the likelihood of the model being overly tailored to a certain flow, hence improving the precision of the forecasts.

- The inputs to the neural network had a tendency to fluctuate greatly in amplitude, specifically, they included values that were very large and near to set boundaries. Therefore, this approach normalizes the strain and rotation tensor by the mean flow time scale.

Given the preceding detailed mathematical basis, the architecture of the neural network employed in the study for the PH scenario is outlined in the following table.

Name	Value
Number of units in hidden layer	20
Number of hidden layers	5
Batch size	64
Weight decay	0.01
Learning rate	5e-4 with learning rate decay on plateau
Nonlinear activation function	Leaky ReLu
SVGD Particles	15
Optimizer	ADAM [58]

Table 4.1: Neural network architecture for SVGD BNN

ELBO set up

Following the preceding detailed mathematical basis in section 3.1.4, and the algorithm is Appendix C the architecture of the neural network employed in the study for the PH scenario is outlined in the following table. After several experiments, it was determined that the Sigmoid function as the activation function yields more stable outcomes. Hence, the Sigmoid function was adopted in the ELBO method. Furthermore, the two techniques that can improved performance also applied in the ELBO method.

Name	Value
Number of units in hidden layer	20
Number of hidden layers	4
Batch size	64
Learning rate	5e-4 with learning rate decay on plateau
Nonlinear activation function	Sigmoid
Optimizer	ADAM

Table 4.2: Neural network architecture for ELBO BNN

The above table shows the neural network structure for the ELBO method. And the results of first attempt for this two method will be discussed in the following section.

Results comparison

The R results shaded with the total uncertainty(Aleatoric + Epistemic) with ELBO and SVGD method are shown in Figure 4.8 and 4.9. The green line is the $y = x$ line which shows the difference between predicted and actual values

In this evaluation, both techniques yield predictions that align well with the selected training data at the low value part. With the increase of the r, both of the prediction will for away from the straight line but the results calculated by the SVGD further away. It is obviously that the prediction by the SVGD method is less accurate than the results calculated by ELBO. The rrms for the evidence lower bound (ELBO) approach is 0.020, whereas for the Stein variational gradient descent (SVGD) method it is 0.044. This indicates that the ELBO method achieves a 54% improvement in accuracy compared to the SVGD method.

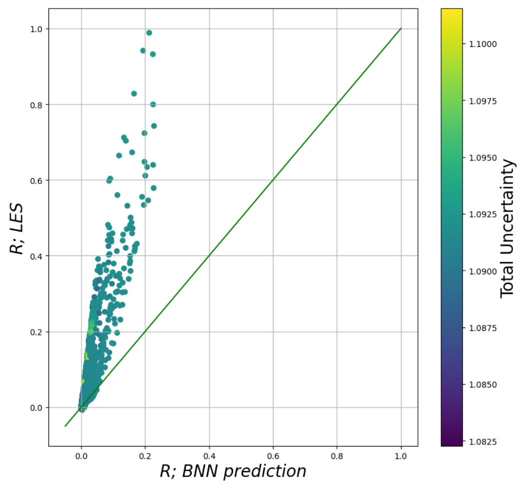


Figure 4.8: Models Incorporating Total uncertainty(epistemic + aleatoric) Using SVGD Methods in Turbulence Data Analysis

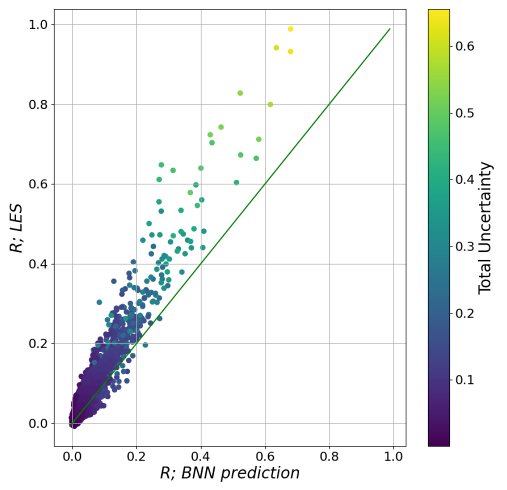


Figure 4.9: Models Incorporating Total uncertainty(epistemic + aleatoric) Using ELBO Methods in Turbulence Data Analysis

The difference in uncertainty scales highlights the contrasting behavior of the two methods. The ELBO methods shows a suitable uncertainty change that uncertainty increases as error increases. This means that the uncertainty obtained by this method is as expected. In contrast, the SVGD method, show almost a constant value in the uncertainty prediction with a little difference on finite point. The method fail to fully capture the variability in the data which the same uncertainty for every data point. Given these findings, the ELBO method is ultimately chosen as the preferred approach for turbulence data analysis, offering a more robust and reliable framework for uncertainty quantification in complex data.

Following the selection of the method, the subsequent step involves hyperparameter optimization. This process is a critical and empirically driven aspect of neural network development. Given the multitude of influencing parameters, it is essential to systematically explore various combinations to identify the optimal configuration that minimizes loss and enhances generalization performance. The primary objective is to determine the set of hyperparameters that achieves the lowest possible loss function value while simultaneously ensuring robust generalization to unseen data and preventing overfitting.

4.2. Hyperparameter search

Name	Value
Number of units in hidden layer	25
Number of hidden layers	4
Batch size	32
Learning rate	5e-4, with learning rate decay on plateau
Nonlinear activation function	Sigmoid
Epochs	400
Optimizer	ADAM

Table 4.3: Neural network architecture and training details

The architecture of the model is detailed in Table 4.3. This configuration was established through a grid search process for hyper-parameter optimization, considering hidden dimensions $N_h \in \{5, 10, 15, 20\}$, the number of hidden layers $N_l \in \{2, 3, 4\}$, batch sizes $M \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$, and learning rates $\eta \in \{5 \times 10^{-3}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-5}, 5 \times 10^{-5}, 5 \times 10^{-6}\}$. Each candidate model was trained over a span of 400 epochs. The selected model was implemented using TensorFlow Probability in Python [59] [60]. Training was conducted on an NVIDIA GeForce RTX 3060 graphics card, requiring approximately one hour per training session of 400 epochs.

5 Results and discussion

This chapter presents the results of applying Bayesian neural networks to the turbulence modeling problem and the performance of the BNN model is demonstrated through a priori testing. The training and evaluation scenarios involve basic two-dimensional geometries that demonstrate flow separation across a curved boundary. This is one of the flow conditions where traditional RANS turbulence models make the largest errors. Initially, the outcomes based on the training data are discussed, highlighting the network's performance on the established dataset. Subsequently, the trained Bayesian neural network is applied to different flow cases to assess the model's generalizability. In all scenarios, both the predicted values and the associated uncertainties are analyzed, providing insights into the model's reliability and robustness. Furthermore, the implications of uncertainty quantification for practical turbulence modeling applications are explored, emphasizing the advantages of Bayesian approaches in capturing the inherent variability and uncertainty in complex flow phenomena. All results are derived using regression with the correction for the turbulence kinetic energy (R) as the prediction target.

5.1. Training results

As discussed in Section 4.1.2, the training dataset is derived from periodic hill data without the incorporation of additional synthetic data. The outcomes are illustrated in Figures 5.1 and 5.2. Notably, there is a remarkable concordance between the model predictions and the LES data, particularly in regions densely populated with data points. However, the model's performance deteriorates in regimes characterized by high turbulence correction terms. This finding is supported by Figure 5.1, which shows that as the R value increases, the predicted values deviate farther from the $y = x$ line. Based on the same figure, it can be inferred that the aleatoric uncertainty rises proportionally with the increase in inaccuracy in predicted values. Furthermore, this aligns with what was anticipated in theory. Aleatoric uncertainty quantifies the degree of random variation present in the data. The presence of significant data noise can readily result in substantial inaccuracies in data prediction.

Moreover, the regions with sparse data are correlated with elevated epistemic uncertainties, especially in areas that deviate significantly from the parity line between the model predictions and the LES data, most notably at R values greater than 0.2. This discrepancy aligns with theoretical expectations, as epistemic uncertainty will increase with the lack of data. For the periodic hill case, the flow is predominantly gentle, with only free shear layer regions capable of generating substantial turbulent energy, as indicated by the streamline patterns in Figure 4.7. And this character explain the relative small amount of number value in large kinetic region. Under conditions of abundant training data, the epistemic error is minimized, underscoring the need for comprehensive data coverage to mitigate uncertainty and improve model fidelity.

Compared with Figures 5.1 and 5.2, it is evident that these two uncertainties exhibit a close similarity in distribution and trends, differing numerically by at least an order of magnitude. This numerical difference arises because epistemic uncertainty, which reflects the model's lack of knowledge, decreases as the model converges. Naturally, this leads to a reduction in uncertainty over time. Consequently, aleatoric uncertainty emerges as the dominant source of uncertainty. The reason for this similarity will be explained in the next section.

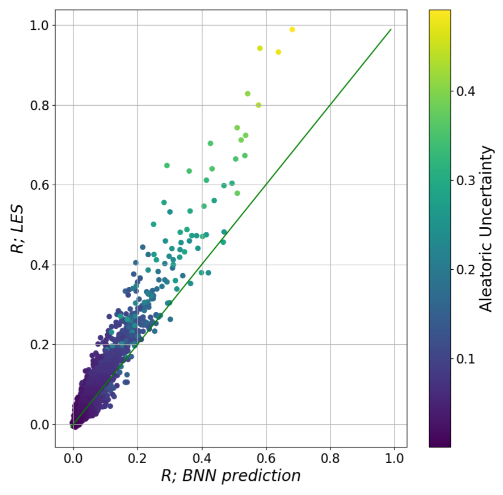


Figure 5.1: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of aleatoric uncertainty by the BNN model.

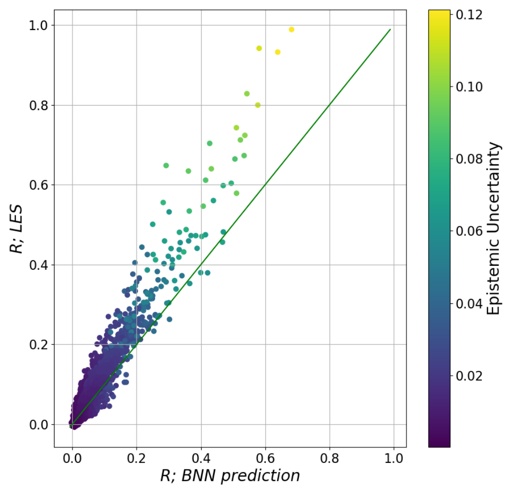


Figure 5.2: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of epistemic uncertainty by the BNN model.

5.1.1. Explaination for Aleatoric and Epistemic Uncertainties

In contrast to aleatoric uncertainty, the epistemic uncertainty quantified by BNN can be strategically utilized to refine datasets, thereby enhancing the TKE correction terms and improving the modeling of turbulence in anisotropic closure parts. This methodological enhancement is rooted in the principles of Optimal Experimental Design [61], a strategy that aims to optimize data selection processes to maximize model performance and minimize uncertainty. This approach is closely associated with Bayesian optimization [62], which serves to systematically refine predictive models by targeting data that most effectively constrains model uncertainties.

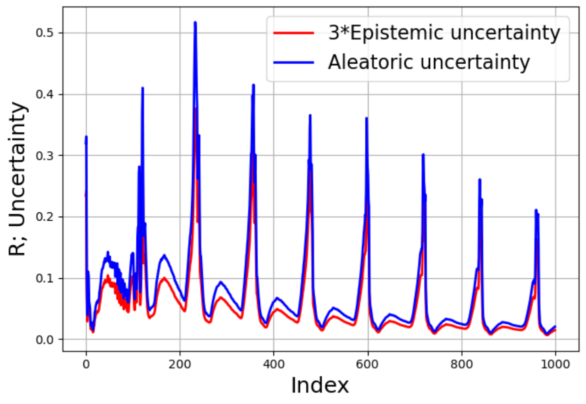


Figure 5.3: Aleatoric and epistemic uncertainty ($\times$ 4) with respect to the R region at the upper boundary

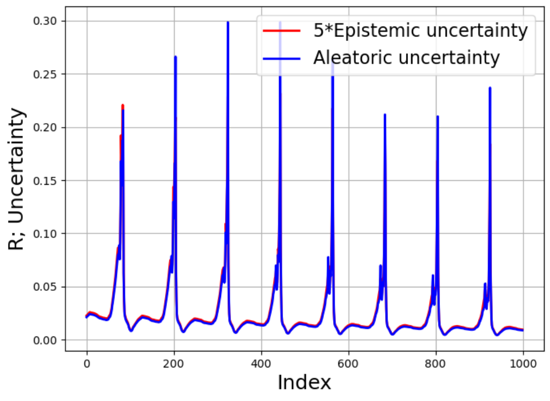


Figure 5.4: Aleatoric and epistemic uncertainty ($\times$ 5) with respect to the R region at the flow separation region

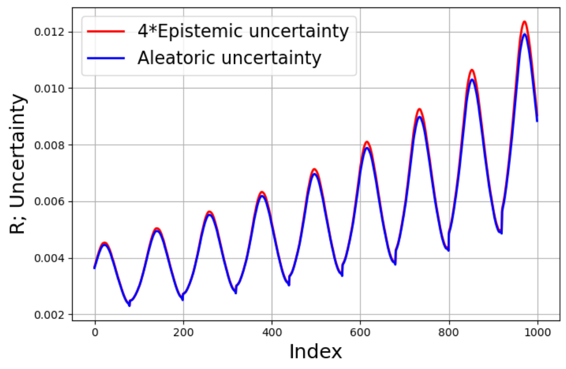


Figure 5.5: Aleatoric and epistemic uncertainty ($\times$ 4) with respect to the R region at the lower boundary

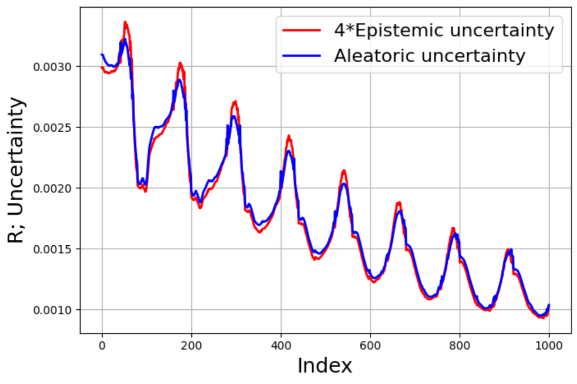


Figure 5.6: Aleatoric and epistemic uncertainty ($\times$ 4) with respect to the R region at the normal flow region

To quantify and visually represent epistemic uncertainty, 100 realizations of the model weights are generated from $\mathbf{w} \sim q(\mathbf{w} \mid \theta)$ the sample variance estimate of the epistemic uncertainty as $\mathrm{Var}\left(\mathbb{E}_{q(\mathbf{w}|\theta)}[\mathbf{y} \mid \mathbf{x}, \boldsymbol{D}]\right)$ is computed. This uncertainty is contrasted with the aleatoric uncertainty illustrated in Figures 5.4 and 5.3, considering various scaling dimensions. From the above figures, the structure of the uncertainties show a lot of spikes. This is due to the ordering of one-dimensional data. The data of the entire 2D flow field is extracted and spliced layer by layer to obtain one-dimensional data. Then each layer will have some large values at fixed positions, resulting in the appearance of periodic spikes in the final figures. Notably, the trends in epistemic uncertainty echo those observed in aleatoric uncertainty, particularly concerning the conditioning relative to different regions of the periodic hill. One explanation is because the structure of the target R in the flow zone is very simplistic, with high values only observed at the separation point and along the wall. As a result, there is minimal distinction between the aleatoric and epistemic uncertainty for the model to discern. The similar conclusion is also reached in the study of Kendall et al. [48]. If the image to be identified consists of a straightforward arrangement of basic shapes. It may be inferred that both aleatoric uncertainty and epistemic uncertainty have an equivalent impact on the R. This observation suggests that to effectively mitigate epistemic uncertainty, data collection strategies should focus on regions characterized by high aleatoric uncertainty, thereby enabling a more comprehensive understanding and modeling of the underlying physical processes.

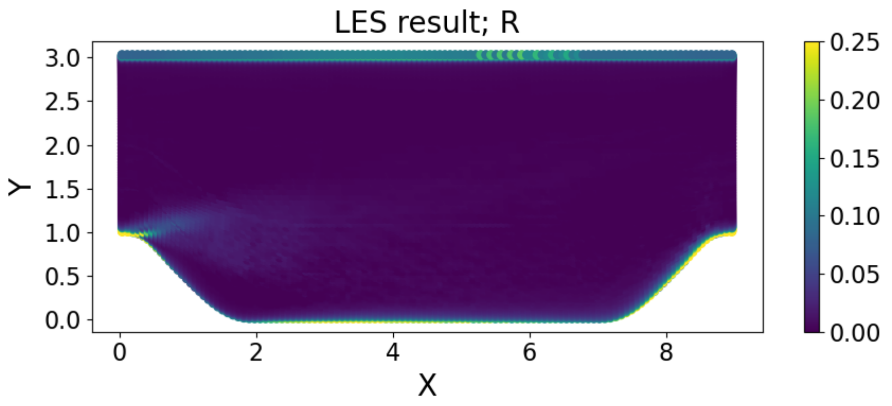
5.1.2. Interpretation in Physical Space

In order to better display the results and have a more in-depth discussion, the results of BNN will be discussed in physical space in this section. Firstly, the predicition of the BNN and the LES data in the periodic hill flow are shown in Figure 5.7a and Figure 5.7b . It reveals a high degree of similarity in their overall spatial distributions. Both the BNN prediction and the LES data effectively capture key flow features, including regions of high TKE in the recirculation zones downstream of the hill and the large value region near the boundary.

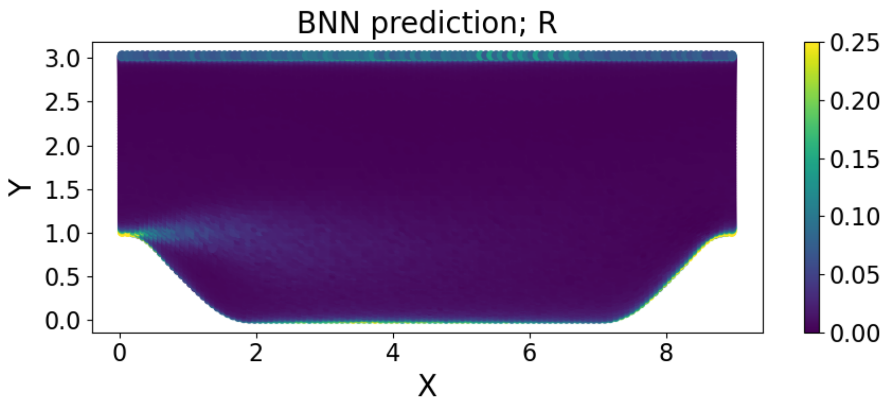
However, there are notable differences, particularly in the region immediately downstream of the hill, where the LES data exhibits slightly higher TKE values. Additionally, the BNN predictions show discrepancies near the boundaries, where the predicted TKE values are generally lower compared to the LES data. These differences suggest that while the BNN is effective in approximating the overall flow characteristics, it tends to underpredict TKE in regions of complex turbulence and near the boundaries. This highlights areas for potential refinement in the model to improve its accuracy, especially in capturing the intensity of turbulent structures and boundary effects. n the studies conducted by [29], [2], and [3], it is noted that the tensor basis neural networks exhibit limitations in accurately predicting values near boundaries. To address these discrepancies, hybrid LES/RANS methodologies[63] have been proposed as potential enhancements for improving boundary value predictions. Nonetheless, due to temporal constraints, such corrective measures could not be implemented within the scope of this current work.

Furthermore, the BNN uncertainty predictions for R in the periodic hill flow case was shown in Figure 5.7d and Figure 5.7c. These two picture reveal distinct patterns in aleatoric and epistemic uncertainties. Aleatoric uncertainty, which reflects inherent variability in the data, is higher near boundary layers and turbulent regions, particularly in the recirculation zones downstream of the hill, where the complex flow interactions amplify variability. In contrast, epistemic uncertainty, representing the model's lack of knowledge, is more localized and prominent around the hill crest and wake regions, indicating areas where the model is less confident due to limited data or incomplete understanding. These distinctions highlight that while aleatoric uncertainty is more uniformly distributed and irreducible, epistemic uncertainty is concentrated in specific regions, suggesting potential for reduction through additional data or improved modeling.

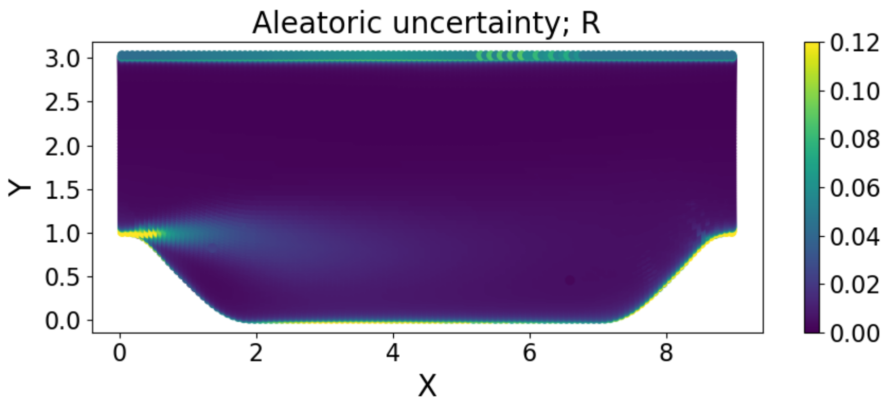
The results delineated in section 5.1.1 illustrate that the magnitude of epistemic uncertainty is consistently lower than that of aleatoric uncertainty, typically by one to two orders of magnitude. Additionally, the spatial distribution patterns of both types of uncertainties display pronounced similarities, as previously elucidated in section 5.1.1.



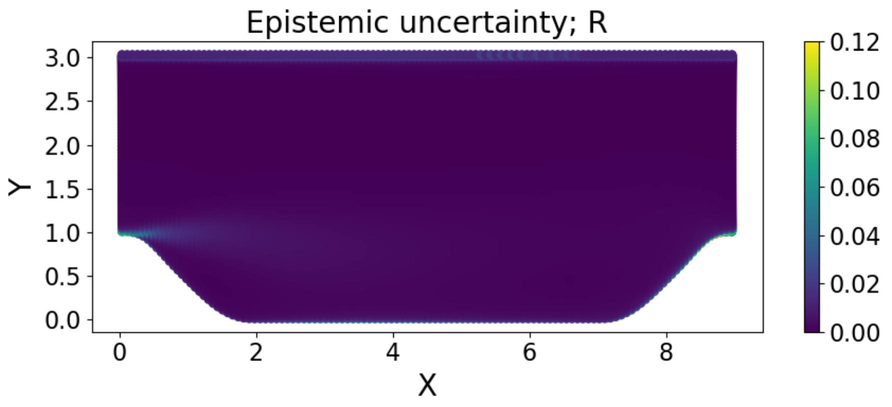
(a)



(b)

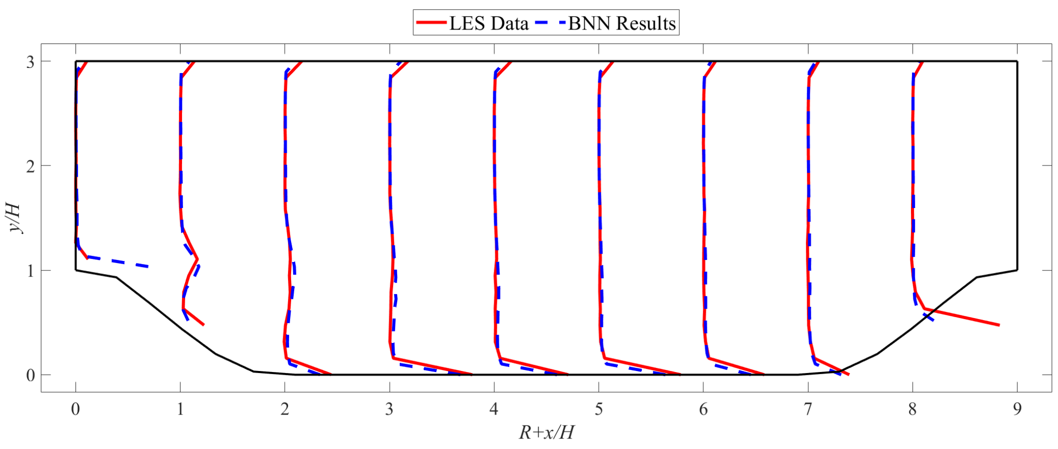


(c)

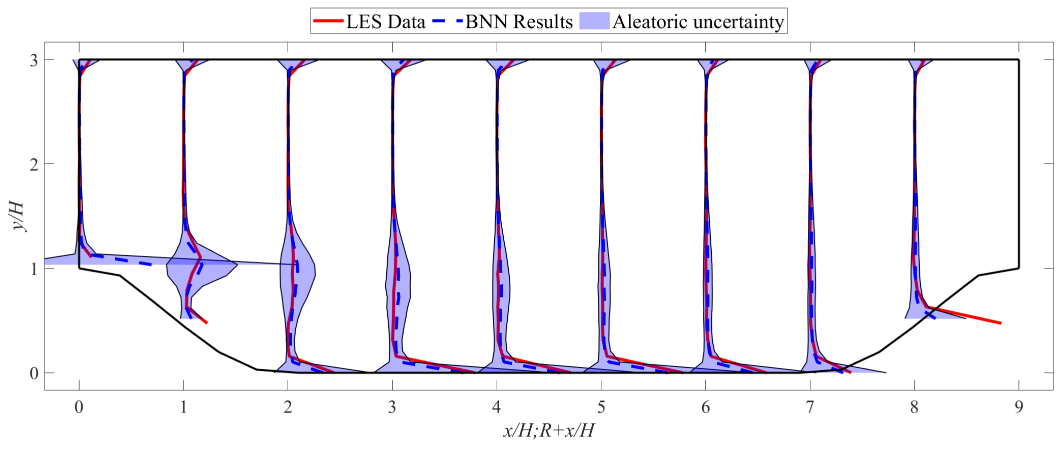


(d)

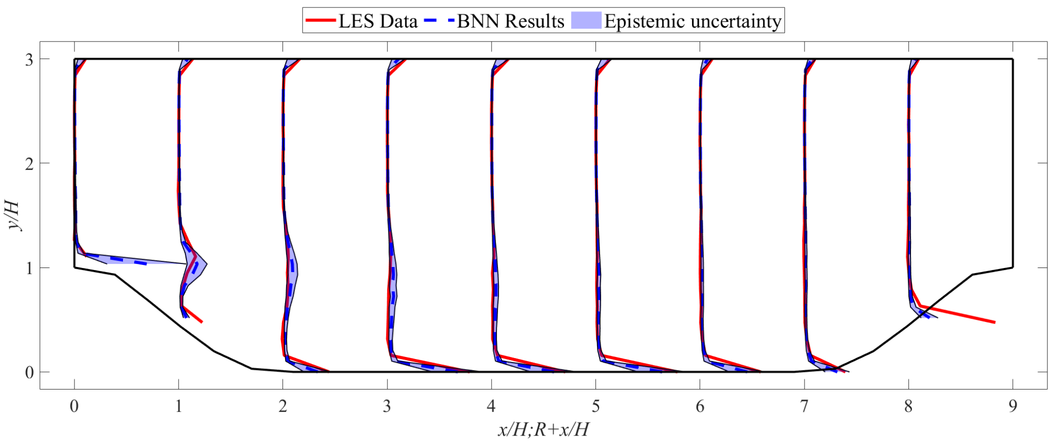
Figure 5.7: Comparison of LES data and BNN prediction for R in physical space with uncertainty region.



(a)



(b)



(c)

Figure 5.8: *R* profiles for high-fidelity LES and Bayesian neural network predictions. (a) Comparison of LES data and neural network predictions along the plane of symmetry at 9 distinct streamwise locations. (b) Comparison of LES data and neural network mean predictions with predicted aleatoric uncertainty. (c) Comparison of LES data and neural network mean predictions with predicted epistemic uncertainty. The red line represents the LES data, while the blue dotted line denotes the neural network predictions. In (b) and (c), the blue region represents the uncertainty region.

Comparative analysis of Figures 5.8a and 5.8b further corroborates that higher values correspond to elevated levels of uncertainty. The comparison of results along the plane of symmetry at nine distinct streamwise locations is depicted in Figure 5.8. The comparison demonstrates generally good concordance, especially within the central regions of the flow, aligning with observations reported in Figure 5.7b. Notably, deviations become apparent near the hill region and at the boundaries of the domain, where the BNN consistently underestimates the R relative to the LES data. The integration of aleatoric uncertainty within the BNN predictions accentuates these areas of disparity, particularly proximate to the hill and within the recirculation zones, where the flow's complexity and turbulence intensify, as visibly illustrated in Figure 5.8b.

Furthermore, the aleatoric uncertainty envelope encompasses the LES data effectively, reflecting consistency with one-dimensional data in capturing the inherent uncertainties. Conversely, Figure 5.8c also delineates a larger region affected by such uncertainties, albeit to a lesser extent than the aleatoric uncertainty. These observations indicate that although the BNN proficiently delineates general trends in TKE production corrections, it manifests heightened uncertainty and diminished accuracy within areas characterized by complex flow dynamics. This suggests that further model enhancements or additional data acquisition might be imperative for augmented predictive accuracy.

5.2. Validation analysis

In the subsequent sections, the neural network described in Section 5.1 is applied across three distinct test cases. The outcomes derived from the model discovery process are thoroughly analyzed and presented. However, given that the periodic hill represents a separation flow, the new flow cases, while varying in geometry, embody the same category of flow dynamics. This setup facilitates a type of cross-validation that evaluates the efficacy of models trained on specific datasets when applied to novel, yet analogous, test scenarios [64].

5.2.1. Curved backward-facing step (CBFS) flow case

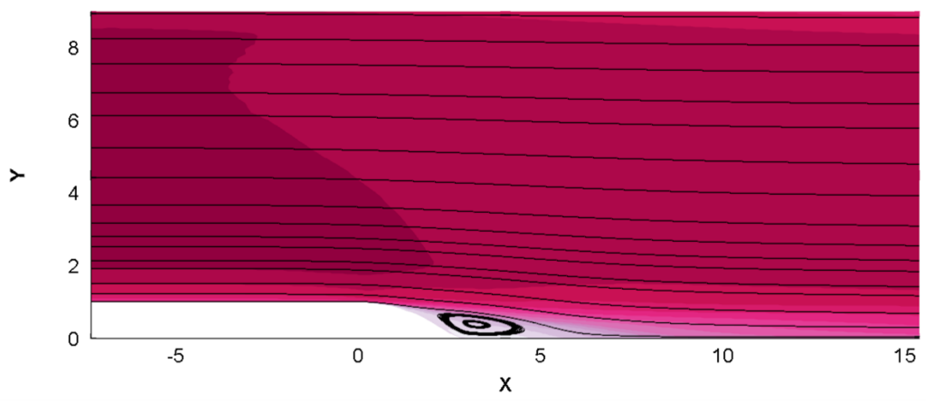
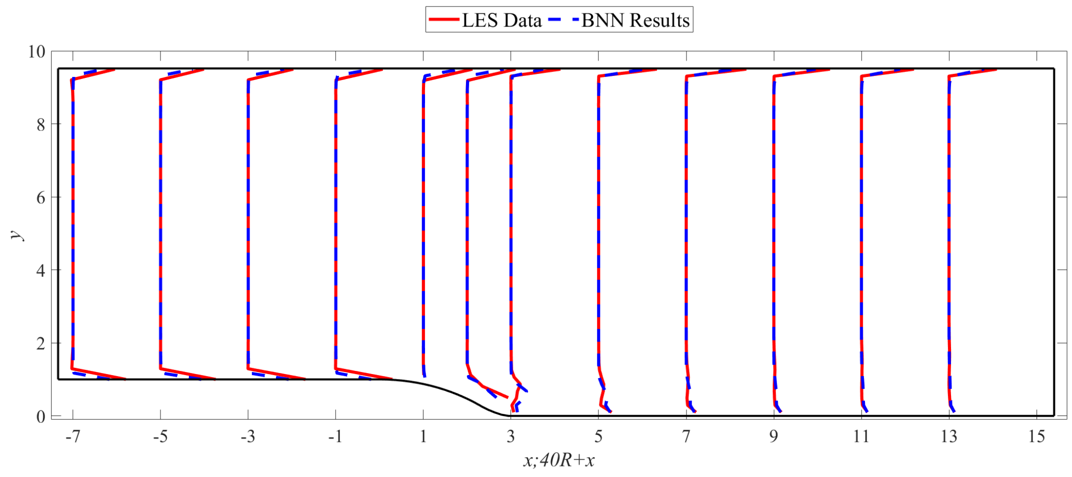
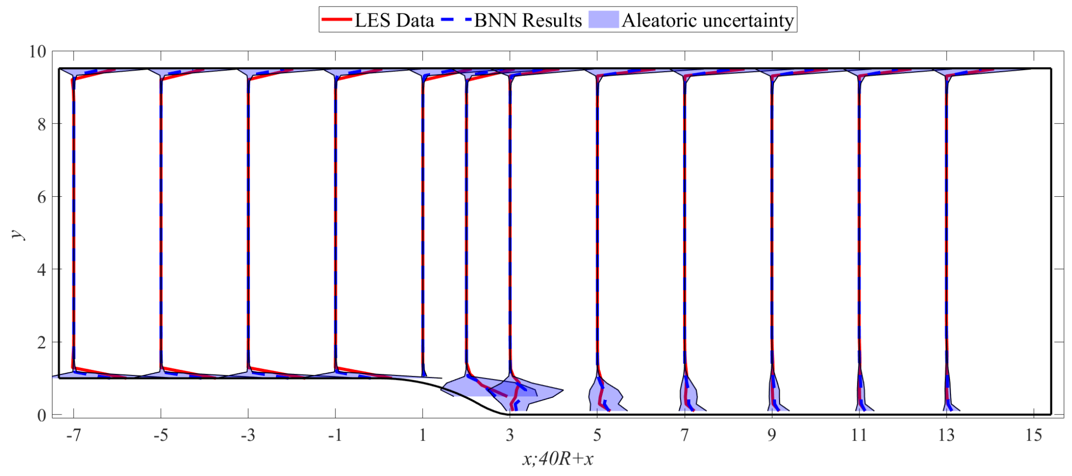


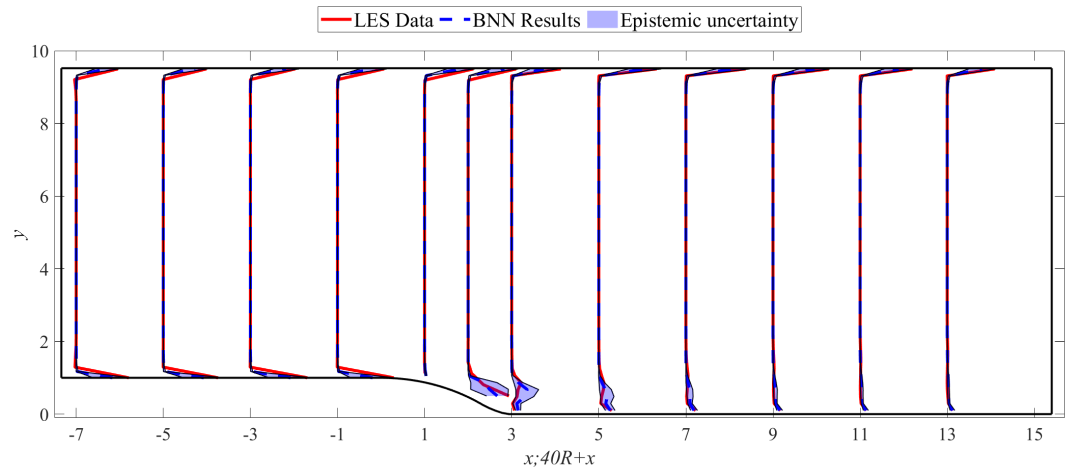
Figure 5.9: The flow domain of the $CBFS_{13700}$ case is illustrated with streamlines, colored according to the magnitude of velocity in the x-direction and y-direction, based on LES data.



(a)



(b)



(c)

Figure 5.10: R profiles for high-fidelity LES and Bayesian neural network predictions. (a) Comparison of LES data and neural network predictions along the plane of symmetry at 12 distinct streamwise locations. (b) Comparison of LES data and neural network mean predictions with predicted aleatoric uncertainty. (c) Comparison of LES data and neural network mean predictions with predicted epistemic uncertainty. The red line represents the LES data, while the blue dotted line denotes the neural network predictions. In (b) and (c), the blue region represents the uncertainty region.

A Large Eddy Simulation was used in [65] to examine the flow over a backward-facing step with a gentle curvature at a Reynolds number of 13700 ($CBFS_{13700}$).This study was primarily focused on examining the mean impacts of detachment and reattachment dynamics, much like the periodic hill (PH) instance. For this simulation, a computational grid of 140×150 cells is used. A completely developed boundary layer simulation was used to determine the inflow boundary condition. Shown in Figure 5.9 is the flow domain. Additionally, the entire domain spans between $0 < y < 9.5$ and $7.3 < x < 15.3$.

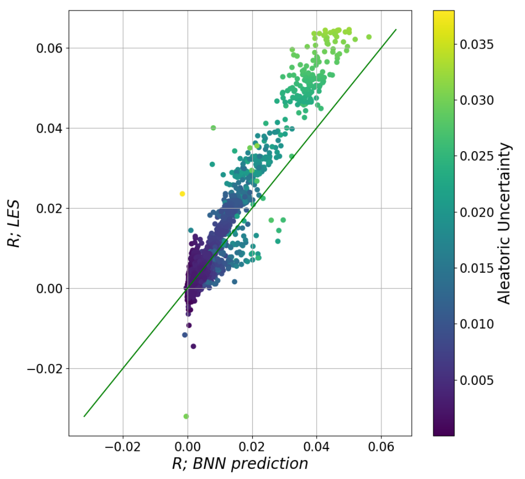


Figure 5.11: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of aleatoric uncertainty by the BNN model.

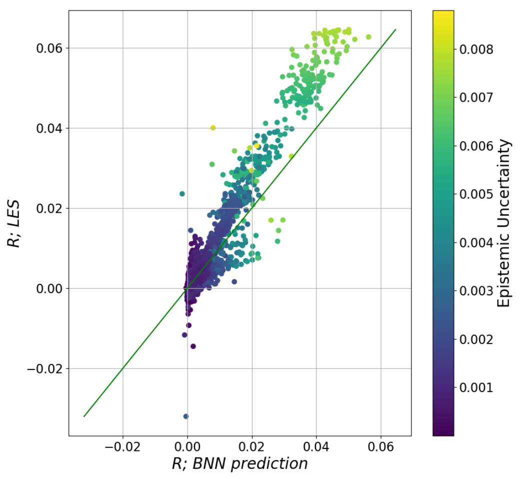


Figure 5.12: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of epistemic uncertainty by the BNN model.

The comparative analysis in Figures 5.11 and 5.12 demonstrates a strong congruence between the Bayesian Neural Network (BNN) outputs and Large Eddy Simulation (LES) data, with only a few outliers deviating from this trend. Uncertainty levels increase with value magnitudes and data sparsity, consistent with the model's training expectations. Figure 5.10 presents the streamline results within the systemic region. The BNN's mean predictions exhibit close alignment with LES data, underscoring the model's robust predictive accuracy. Additionally, the modeled uncertainties—both aleatoric and epistemic—are in line with anticipated behaviors. Notably, the uncertainty markedly escalates in the flow separation region (specifically in the region at x between 1 and 5), a zone notoriously difficult to model accurately. This heightened uncertainty within the separation region accurately reflects the model's capability to identify areas of increased predictive challenge, consistent with established physical insights into flow dynamics.

5.2.2. Turbulent separation bubbles flow case

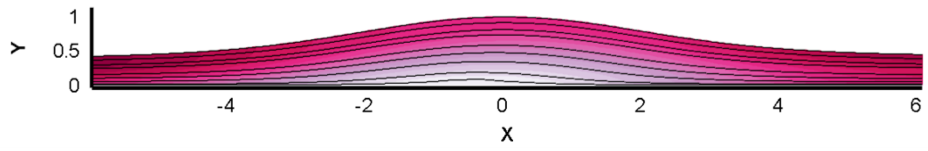


Figure 5.13: The flow domain of the *APG* case is illustrated with streamlines, colored according to the magnitude of velocity in the x-direction and y-direction, based on DNS data.

The DNS dataset for this investigation originates from Coleman of NASA [66]. The flow configuration under examination is a two-dimensional separation bubble, induced through transpiration velocity modifications—specifically suction and blowing—along the superior boundary. The approaching boundary layer in the zero-pressure-gradient sector is characterized by a Reynolds number approximately 2000. Computational analysis was executed utilizing the incompressible Navier-Stokes equations within a pseudo-spectral computational framework. Due to the extensive spatial extent of the flow field ($9.5 < x < 12.5$), only the pivotal segment of the separation bubble is depicted in the figure 5.13.

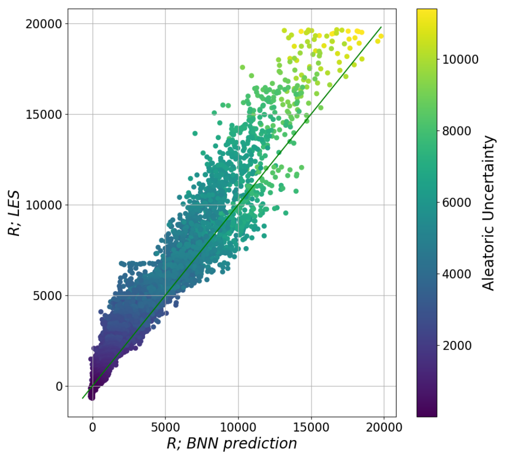


Figure 5.14: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of aletoric uncertainty by the BNN model.

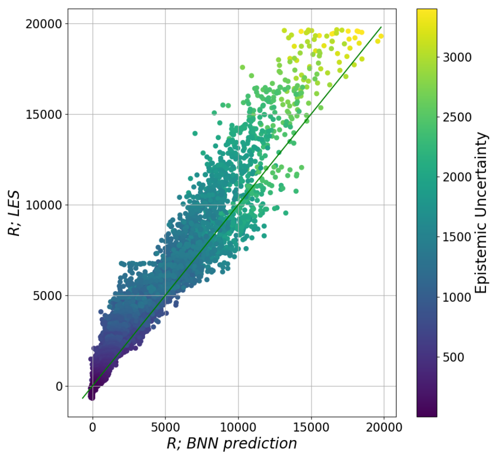
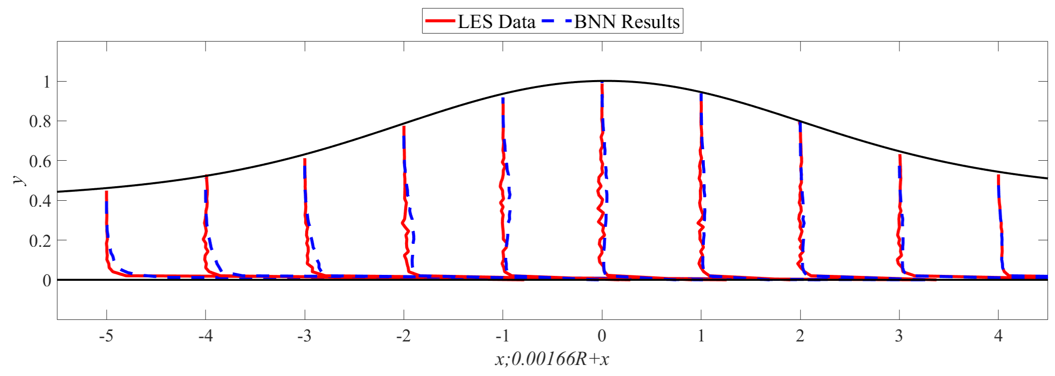


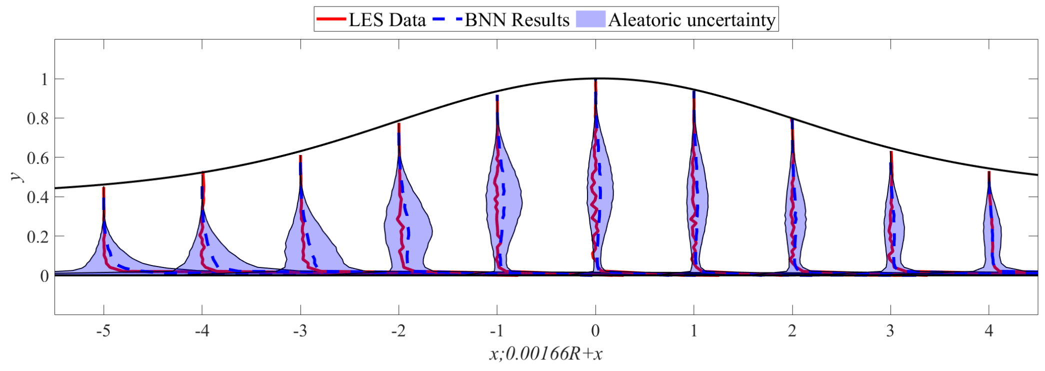
Figure 5.15: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of epistemic uncertainty by the BNN model.

In the comparison analysis of Figures 5.18 and 5.19, the BNN predictions demonstrate a high degree of congruence with LES data, particularly when the value of R is less than 12500. The uncertainty profiles resemble those observed in the periodic hill case, characterized by escalating uncertainty with increasing values and more pronounced uncertainty at sparser data points. From Figure 5.16, It is evident that significant uncertainty and data misalignment occur within the separation region. However, all the

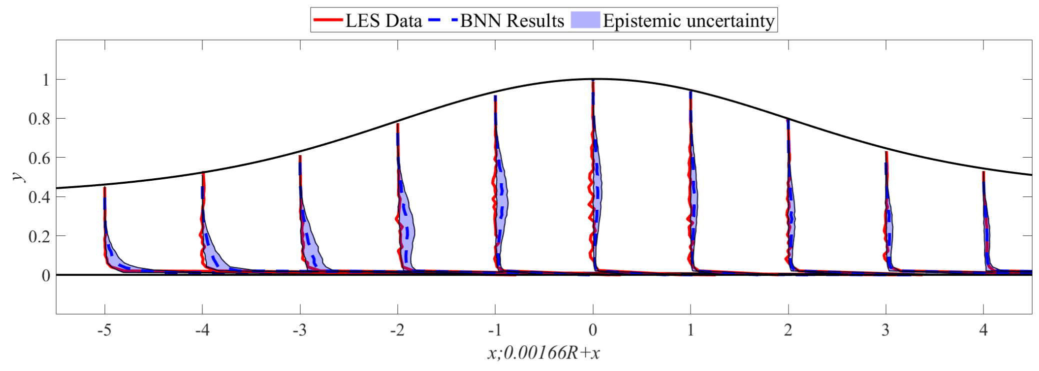
LES data falls within the aleatoric uncertainty bounds, indicating a impressive predictive performance. Given that the entire domain encompasses the separation flow region, all streetwise locations manifest considerable uncertainty compared to previous flow cases. Additionally, regions proximate to the boundary exhibit substantial uncertainty.



(a)



(b)



(c)

Figure 5.16: R profiles for high-fidelity LES and Bayesian neural network predictions. (a) Comparison of LES data and neural network predictions along the plane of symmetry at 10 distinct streamwise locations. (b) Comparison of LES data and neural network mean predictions with predicted aleatoric uncertainty. (c) Comparison of LES data and neural network mean predictions with predicted epistemic uncertainty. The red line represents the LES data, while the blue dotted line denotes the neural network predictions. In (b) and (c), the blue region represents the uncertainty region.

5.2.3. NASA-Hump flow case

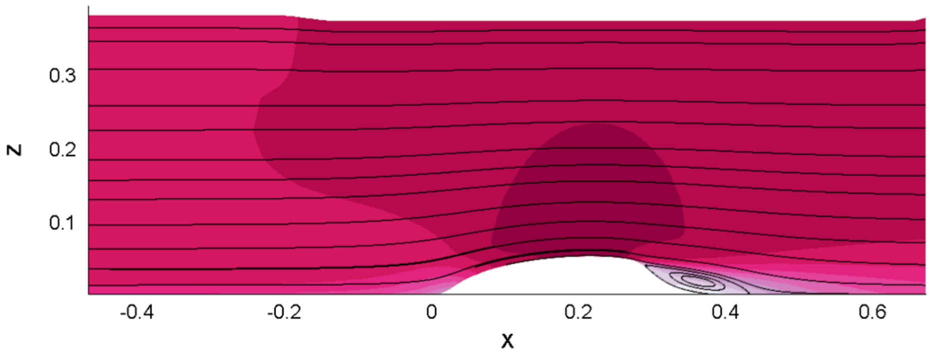


Figure 5.17: The flow domain of the NASA-Hump case is illustrated with streamlines, colored according to the magnitude of velocity in the x-direction and z-direction, based on LES data.

The NASA-Hump setup is used as a standard to assess how well turbulence models can forecast two-dimensional flow separation—caused by an unfavorable pressure gradient—from a smooth surface, as well as the subsequent reattachment and recovery of the boundary layer. The model is positioned between two glass endplates, and a wind tunnel splitter plate is completely incorporated with both the leading and trailing edges. For this flow instance, simulations were run at $Re = 93600$ Reynolds number. Detailed information on this setup can be found in [67], whereas [68] provides empirical data. In Figure 5.17, the real flow situation is depicted.

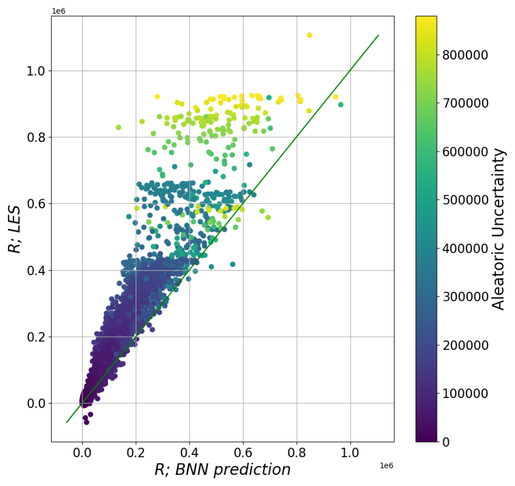


Figure 5.18: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of aletoric uncertainty by the BNN model.

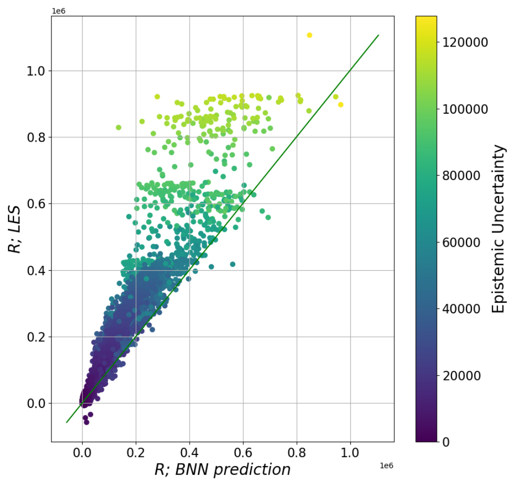
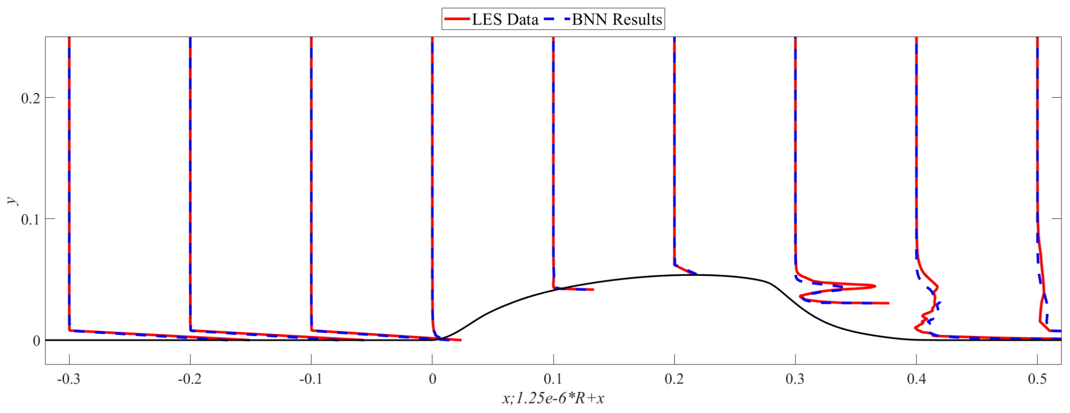


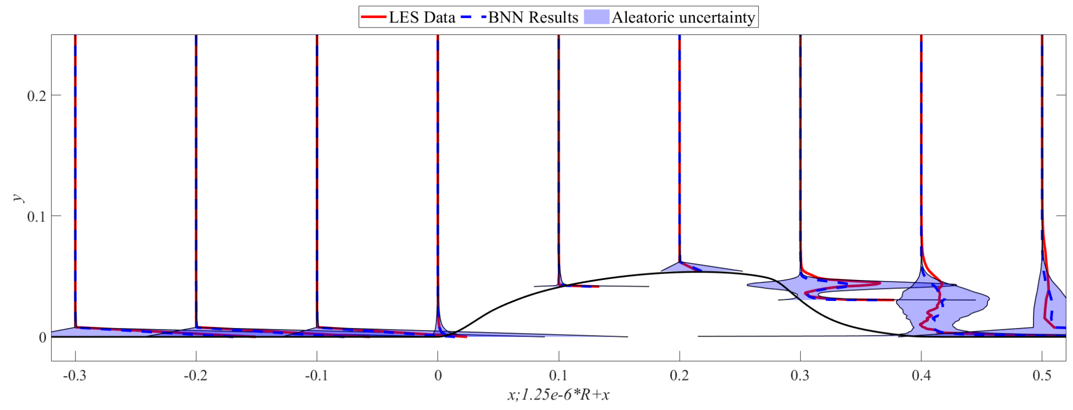
Figure 5.19: A scatter plot visually representing the average predictions for each data point, colored based on the calculation of epistemic uncertainty by the BNN model.

As in the preceding analysis, the results from the BNN are compared with the LES data, with uncertainties categorized as either aleatoric or epistemic, depicted in Figures 5.18

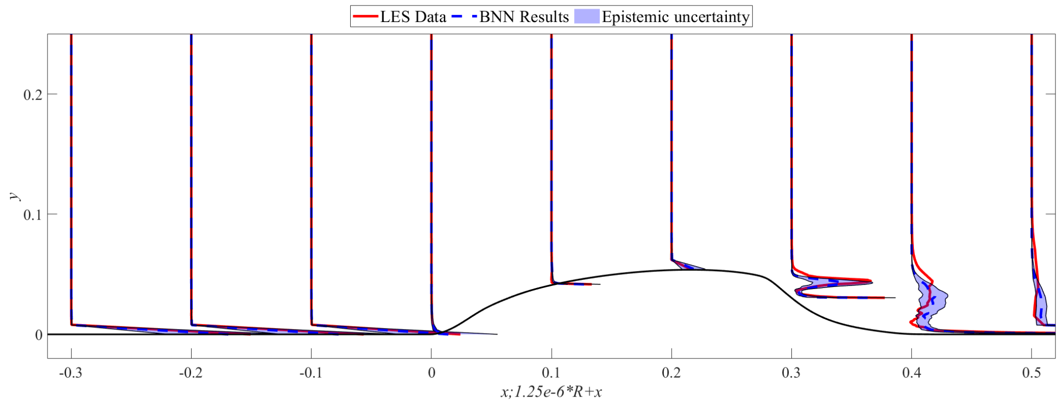
and 5.19. Relative to other flow cases examined, the BNN predictions for the NASA-Hump configuration exhibit substantial discrepancies, as evidenced by the preponderance of data points residing above the $y = x$ line.



(a)



(b)



(c)

Figure 5.20: R profiles for high-fidelity LES and Bayesian neural network predictions. (a) Comparison of LES data and neural network predictions along the plane of symmetry at 9 distinct streamwise locations. (b) Comparison of LES data and neural network mean predictions with predicted aleatoric uncertainty. (c) Comparison of LES data and neural network mean predictions with predicted epistemic uncertainty. The red line represents the LES data, while the blue dotted line denotes the neural network predictions. In (b) and (c), the blue region represents the uncertainty region.

This deviation is partly attributable to the inherent limitations of the RANS models in accurately predicting complex flows such as those encountered in the NASA-Hump scenario. Notably, the uncertainty distribution aligns with the expected pattern: points clustered closely within the dataset exhibit lower uncertainty, whereas more dispersed points display increased uncertainty, which escalates with the magnitude of the values, corroborating the findings from Section 5.1. A comparative analysis of Figure 5.20 reveals a marked congruence between the LES data and prediction in regions devoid of separation flow, whereas significant discrepancies in the separation flow regions are highlighted by a pronounced aleatoric uncertainty, affirming the observations from Figure 5.8. Despite these variations, the overall predictive performance remains robust, particularly in terms of uncertainty quantification.

6 Conclusions and Recommendations

Given the growing prevalence of machine learning technologies in the computational fluid dynamics (CFD) field, it is becoming increasingly crucial to extract uncertainty from both the data and the model. This study introduced the initial implementation of a Bayesian neural network method for data-driven closure modeling in RANS, which includes estimates of aleatoric and epistemic uncertainty. The primary objective was to simulate the adjustment of turbulence kinetic energy between the LES and RANS data. However, the techniques used in this study may be extended to closure modeling, particularly for anisotropic tensors, using the neural network structure specified in section 3.4.1. Derived upon Ling's influential study [29], this neural network design preserves essential invariant characteristics, therefore guaranteeing the resilience and dependability of the modeling procedure.

In the couse of study, a priori tests showed a good mean prediction of the R, indicating that the integration of uncertainty estimates during the training phase does not compromise model precision. This work also found that the SVGD method is not well adapted to identify the uncertainty problem of turbulence modeling. The SVGD method tends to give an inaccurate aleatoric uncertainty range even at a low training error level, which is reflected in previous work by Geneva [2] and section 4.1. Conversely, employing ELBO method to directly minimize the Kullback-Leibler divergence has proven more effective.

Further tests were conducted to assess the model's generalization capabilities across three additional flow separation scenarios not included in the initial training dataset. These tests revealed acceptable accuracy levels, although notable discrepancies were observed in regions of intense flow separation. Such discrepancies suggest that the aleatoric uncertainty is significantly pronounced in these regions. This study hypothesizes that the reliance on merely two invariant inputs and the ϵ parameter is insufficient for capturing the nuanced transitions from coarse to high-fidelity flow physics accurately. Despite the model's design to retain desired invariant properties, it is likely that incorporating additional flow variables could enhance the prediction quality.

A critical avenue for future research is the development of more precise Reynolds stress representations, which will involve identifying both local and non-local variables that critically influence these values. While several models in existing literature offer improved predictions within limited flow scenarios, their generalization across diverse flow conditions remains a significant challenge for the data-driven CFD community.

This research has established that aleatoric uncertainty generally exceeds epistemic uncertainty, with both types of uncertainty exhibiting similar variations across the phase space. However, disparities are more pronounced in regions with high progress variable values, likely due to the uniform data sampling method employed. These observations suggest the potential benefits of adopting a non-uniform data sampling strategy to address areas with elevated aleatoric uncertainty effectively.

The dual capability of this Bayesian framework to quantify both aleatoric and epistemic uncertainties provides invaluable insights for enhancing data-driven models. Not only does this framework facilitate a better understanding of a model's predictive confidence, but it also identifies potential zones of reduced accuracy, which could guide the deployment of finer mesh resolutions or more detailed simulations. Moreover, the application of a Bayesian neural network enables the calculation of predictive bounds for critical quantities, offering substantial advantages in scenarios characterized by limited training data.

Based on the current research, numerous enhancements can be implemented in future studies. Initially, the training dataset incorporates only a single flow case; introducing additional flow cases could enhance the model's predictive accuracy. Secondly, adjusting the prediction target to the b_{ij} tensor represents a comprehensive turbulence modeling challenge; integrating such models into OpenFOAM for posterior analysis could provide a more thorough evaluation of their performance. Furthermore, extending the analysis from 2-D to 3-D would allow the examination of flow scenarios with the complete set of five invariants, rather than just two. Currently, aleatoric uncertainty is estimated using basic uncertainty propagation methods; future work could include the development of a covariance matrix among the coefficients. Lastly, the existing approach utilizes a simple Gaussian prior; exploring alternative distributions or deriving the prior from empirical flow data could enhance model robustness and facilitate transfer learning strategies.

References

[1] Stephen B Pope. "Turbulent flows". In: *Measurement Science and Technology* 12.11 (2001), pp. 2020–2021.

[2] Nicholas Geneva and Nicholas Zabaras. "Quantifying model form uncertainty in Reynolds-averaged turbulence models with Bayesian deep neural networks". In: *Journal of Computational Physics* 383 (2019), pp. 125–147.

[3] Hongwei Tang et al. "Data-driven Reynolds-averaged turbulence modeling with generalizable non-linear correction and uncertainty quantification using Bayesian deep learning". In: *Physics of Fluids* 35.5 (2023).

[4] W Rodi. "Comparison of LES and RANS calculations of the flow around bluff bodies". In: *Journal of wind engineering and industrial aerodynamics* 69 (1997), pp. 55–75.

[5] Lewis F Richardson. *Weather prediction by numerical process*. University Press, 1922.

[6] Andrei Nikolaevich Kolmogorov. "The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 434.1890 (1991), pp. 9–13.

[7] Steven A Orszag. "Analytical theories of turbulence". In: *Journal of Fluid Mechanics* 41.2 (1970), pp. 363–386.

[8] Hao Lu, Christopher J Rutland, and Leslie M Smith. "A priori tests of one-equation LES modeling of rotating turbulence". In: *Journal of Turbulence* 8 (2007), N37.

[9] Stefan Hickel. *Large eddy Simulation:LES , CFD for aerospace engineer*. Tech. rep. Tu Delft, 2022.

[10] Joseph Smagorinsky. "General circulation experiments with the primitive equations: I. The basic experiment". In: *Monthly weather review* 91.3 (1963), pp. 99–164.

[11] David C Wilcox et al. *Turbulence modeling for CFD*. Vol. 2. DCW industries La Canada, CA, 1998.

[12] Jeffrey P Slotnick et al. *CFD vision 2030 study: a path to revolutionary computational aerosciences*. Tech. rep. 2014.

[13] Ludwig Prandtl. "7. Bericht über Untersuchungen zur ausgebildeten Turbulenz". In: *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* 5.2 (1925), pp. 136–139.

[14] Jean Boussinesq. *Essai sur la théorie des eaux courantes*. Imprimerie nationale, 1877.

[15] W Peter Jones and Brian Edward Launder. “The prediction of laminarization with a two-equation model of turbulence”. In: *International journal of heat and mass transfer* 15.2 (1972), pp. 301–314.

[16] Stefan Hickel. *Reynolds - Averaged Navier- Stokes : RANS , CFD for aerospace engineer.* Tech. rep. Tu Delft, 2022.

[17] W Rodi and G Scheuerer. “Scrutinizing the k-ε turbulence model under adverse pressure gradient conditions”. In: (1986).

[18] David C Wilcox. “Reassessment of the scale-determining equation for advanced turbulence models”. In: *AIAA journal* 26.11 (1988), pp. 1299–1310.

[19] Florian R Menter. “Influence of freestream values on k-omega turbulence model predictions”. In: *AIAA journal* 30.6 (1992), pp. 1657–1659.

[20] Florianr Menter. “Zonal two equation kw turbulence models for aerodynamic flows”. In: *23rd fluid dynamics, plasmadynamics, and lasers conference.* 1993, p. 2906.

[21] Florian R Menter. “Two-equation eddy-viscosity turbulence models for engineering applications”. In: *AIAA journal* 32.8 (1994), pp. 1598–1605.

[22] Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. “Metaheuristic design of feedforward neural networks: A review of two decades of research”. In: *Engineering Applications of Artificial Intelligence* 60 (2017), pp. 97–116.

[23] Bin Ding, Huimin Qian, and Jun Zhou. “Activation functions and their characteristics in deep neural networks”. In: *2018 Chinese control and decision conference (CCDC).* IEEE. 2018, pp. 1836–1841.

[24] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* SIAM, 2008.

[25] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257.

[26] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[27] Todd A Oliver and Robert D Moser. “Bayesian uncertainty quantification applied to RANS turbulence models”. In: *Journal of Physics: Conference Series.* Vol. 318. 4. IOP Publishing. 2011, p. 042032.

[28] WN Edeling, Pasquale Cinnella, and Richard P Dwight. “Predictive RANS simulations via Bayesian model-scenario averaging”. In: *Journal of Computational Physics* 275 (2014), pp. 65–91.

[29] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance”. In: *Journal of Fluid Mechanics* 807 (2016), pp. 155–166.

[30] Mikael LA Kaandorp and Richard P Dwight. “Data-driven modelling of the Reynolds stress tensor using random forests with invariance”. In: *Computers & Fluids* 202 (2020), p. 104497.

[31] Armen Der Kiureghian and Ove Ditlevsen. “Aleatory or epistemic? Does it matter?” In: *Structural safety* 31.2 (2009), pp. 105–112.

[32] Habib N Najm. “Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics”. In: *Annual review of fluid mechanics* 41.1 (2009), pp. 35–52.

[33] Heng Xiao et al. “Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach”. In: *Journal of Computational Physics* 324 (2016), pp. 115–136.

[34] Jin-Long Wu et al. “A priori assessment of prediction confidence for data-driven turbulence modeling”. In: *Flow, Turbulence and Combustion* 99 (2017), pp. 25–46.

[35] Soufiane Cherroud et al. “Sparse Bayesian learning of explicit algebraic Reynolds-stress models for turbulent separated flows”. In: *International Journal of Heat and Fluid Flow* 98 (2022), p. 109047.

[36] Graham Pash, Malik Hassanaly, and Shashank Yellapantula. “A Priori Uncertainty Quantification of Reacting Turbulence Closure Models using Bayesian Neural Networks”. In: *arXiv preprint arXiv:2402.18729* (2024).

[37] Ethan Goan and Clinton Fookes. “Bayesian neural networks: An introduction and survey”. In: *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018* (2020), pp. 45–87.

[38] José Miguel Hernández-Lobato and Ryan P. Adams. *Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks*. en. arXiv:1502.05336 [stat]. July 2015. URL: http://arxiv.org/abs/1502.05336 (visited on 12/19/2023).

[39] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[40] Qiang Liu and Dilin Wang. “Stein variational gradient descent: A general purpose bayesian inference algorithm”. In: *Advances in neural information processing systems* 29 (2016).

[41] Qiang Liu, Jason Lee, and Michael Jordan. “A kernelized Stein discrepancy for goodness-of-fit tests”. In: *International conference on machine learning*. PMLR. 2016, pp. 276–284.

[42] Yinhao Zhu and Nicholas Zabaras. “Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification”. In: *Journal of Computational Physics* 366 (2018), pp. 415–447.

[43] Michael E Tipping. “Sparse Bayesian learning and the relevance vector machine”. In: *Journal of machine learning research* 1.Jun (2001), pp. 211–244.

[44] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.

[45] Charles Blundell et al. “Weight uncertainty in neural network”. In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[46] Venkat Nemani et al. “Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial”. In: *Mechanical Systems and Signal Processing* 205 (2023), p. 110796.

[47] Lara Hoffmann and Clemens Elster. “Deep ensembles from a bayesian perspective”. In: *arXiv preprint arXiv:2105.13283* (2021).

[48] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems* 30 (2017).

[49] Andrea Saltelli et al. “Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index”. In: *Computer physics communications* 181.2 (2010), pp. 259–270.

[50] Lucy R Chai. “Uncertainty estimation in bayesian neural networks and links to interpretability”. In: *Master’s Thesis, Massachusetts Institute of Technology* (2018).

[51] Pavel Izmailov et al. “What are Bayesian neural network posteriors really like?” In: *International conference on machine learning*. PMLR. 2021, pp. 4629–4640.

[52] Gao Huang et al. “Trends in extreme learning machines: A review”. In: *Neural Networks* 61 (2015), pp. 32–48.

[53] Martin Schmelzer, Richard P Dwight, and Paola Cinnella. “Discovery of algebraic Reynolds-stress models using sparse symbolic regression”. In: *Flow, Turbulence and Combustion* 104 (2020), pp. 579–603.

[54] Stephen B Pope. “A more general effective-viscosity hypothesis”. In: *Journal of Fluid Mechanics* 72.2 (1975), pp. 331–340.

[55] Michael Leschziner. *Statistical turbulence modelling for fluid dynamics-demystified: an introductory text for graduate engineering students*. World Scientific, 2015.

[56] James Kirchner. “Data analysis toolkit# 5: uncertainty analysis and error propagation”. In: *University of California Berkeley Seismological Laboratory. Available online at: http://seismo. berkeley. edu/~ kirchner/eps_120/Toolkits/Toolkit_05. pdf* (2001).

[57] Michael Breuer et al. “Flow over periodic hills–numerical and experimental study in a wide range of Reynolds numbers”. In: *Computers & Fluids* 38.2 (2009), pp. 433–457.

[58] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[59] Martín Abadi et al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”. In: *CoRR* abs/1603.04467 (2016). arXiv: 1603.04467. URL: http://arxiv.org/abs/1603.04467.

[60] Joshua V Dillon et al. “Tensorflow distributions”. In: *arXiv preprint arXiv:1711.10604* (2017).

[61] Valerii Fedorov. “Optimal experimental design”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.5 (2010), pp. 581–589.

[62] Francesco Di Fiore, Michela Nardelli, and Laura Mainini. “Active learning and bayesian optimization: a unified perspective to learn with a goal”. In: *Archives of Computational Methods in Engineering* (2024), pp. 1–29.

[63] X Xiao, JR Edwards, and HA Hassan. “Blending functions in hybrid large-eddy/Reynolds-averaged Navier-Stokes simulations”. In: *AIAA journal* 42.12 (2004), pp. 2508–2515.

[64] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[65] Yacine Bentaleb, Sylvain Lardeau, and Michael A Leschziner. “Large-eddy simulation of turbulent boundary layer separation from a rounded step”. In: *Journal of Turbulence* 13 (2012), N4.

[66] GN Coleman, CL Rumsey, and PR Spalart. “Numerical study of turbulent separation bubbles with varying pressure gradient and Reynolds number”. In: *Journal of Fluid Mechanics* 847 (2018), pp. 28–70.

[67] David Greenblatt et al. “Experimental investigation of separation control part 1: baseline and steady suction”. In: *AIAA journal* 44.12 (2006), pp. 2820–2830.

[68] Ali Uzun and Mujeeb R Malik. “Wall-resolved large-eddy simulation of flow separation over NASA wall-mounted hump”. In: *55th AIAA Aerospace Sciences Meeting*. 2017, p. 0538.

A

Necessary rules for deriving RANS equations

The following equation shows the rules to derive the RANS control equatons :

$$
\begin{aligned}
&\langle u + v \rangle = \langle u \rangle + \langle v \rangle \\
&\langle au \rangle = a\langle u \rangle \quad , a = \text{const} \\
&\langle \langle u \rangle v \rangle = \langle u \rangle \langle v \rangle \\
&\left\langle \frac{\partial u}{\partial x} \right\rangle = \frac{\partial \langle u \rangle}{\partial x} \\
&\langle u' \rangle = 0 \\
&\langle \langle u \rangle v' \rangle = 0 \\
&\langle uv \rangle = \langle u \rangle \langle v \rangle + \langle u'v' \rangle
\end{aligned}
\tag{A.1}
$$

Where $\langle \cdot \rangle$ means average and u' means fluctuating term.

B

Derivation and proof for Stein Variational Gradient Descent

Stein's Identity

Assume p and q are two distributions, if

$$p = q \quad \Longleftrightarrow \quad \mathbb{E}_{x\sim q}\left[\mathcal{A}_p f(x)\right] = 0 \tag{B.1}$$

Here is called the Stein's Identity. In the equation of the Identity $\mathcal{A}_p$ is called the Stein's operator

$$\mathcal{A}_p f(x) = s_p(x) f(x) + \nabla_x f(x). \tag{B.2}$$

$s_p(x)$ is called the score function:$s_p = \nabla_x \log p(x) = \frac{\nabla_x p(x)}{p(x)}$ And define that a function $f : \mathcal{X} \to \mathbb{R}$ is in the Stein class of p if f is smooth and satisfies

$$\int_{x\in\mathcal{X}} \nabla_x (f(x)p(x))dx = 0 \tag{B.3}$$

Stein's discrepancy

If

$$p \neq q \quad \Longrightarrow \quad \exists f, \text{ such that } \mathbb{E}_{x\sim q}\left[\mathcal{A}_p f(x)\right] \neq 0 \tag{B.4}$$

Stein's discrepancy can be defined as :

$$\mathbb{S}(q,p) = \max_{f\in\mathcal{F}} \mathbb{E}_{x\sim q}\left[\text{trace}\left(\mathcal{A}_p f(x)\right)\right] \tag{B.5}$$

(Trace just to make it become a scalar).Here the choice of this function set F is critical, and decides the discriminative power and computational tractability of Stein discrepancy.

Kernelized Stein discrepancy (KSD)

A kernel $k\left(x, x'\right)$ is defined as to be integrally strictly positive definition, if for any function g that satisfies $0<\|g\|_2^2<\infty$,

$$\int_{\mathcal{X}} g(x) k\left(x, x'\right) g\left(x'\right) d x d x'>0 .$$

And if $k(x, x')$ be a positive definite kernel. The spectral decomposition of $k(x, x')$, as implied by Mercer's theorem, is defined as

$$k\left(x, x'\right)=\sum_j \lambda_j e_j(x) e_j\left(x'\right) \tag{B.6}$$

The kernelized Stein discrepancy (KSD) $\mathbb{S}(q, p)$ between distribution p and q is defined as

$$\mathbb{S}(q, p)=\mathbb{E}_{x, x' \sim p}\left[\boldsymbol{\delta}_{q, p}(x)^{\top} k\left(x, x'\right) \boldsymbol{\delta}_{q, p}\left(x'\right)\right],$$

where $\delta_{q,p}(x)=s_q(x)-s_p(x)$ is the score difference between p and q, and x, x' are i.i.d. draws from $p(x)$.

A kernel $k\left(x, x'\right)$ is defined as to be in the Stein class of p if $k\left(x, x'\right)$ has continuous second order partial derivatives, and both $k(x, \cdot)$ and $k(\cdot, x)$ are in the Stein class of p for any fixed x.

Theorem 1.1 Assume p and q are smooth densities and $k\left(x, x'\right)$ is in the Stein class of p. Define

$$\begin{aligned} u_q\left(x, x'\right)= & s_q(x)^{\top} k\left(x, x'\right) s_q\left(x'\right)+s_q(x)^{\top} \nabla_{x'} k\left(x, x'\right)+ \\ & +\nabla_x k\left(x, x'\right)^{\top} s_q\left(x'\right)+\operatorname{trace}\left(\nabla_{x, x'} k\left(x, x'\right)\right) \end{aligned} \tag{B.7}$$

then $\mathbb{S}(p, q)=\mathbb{E}_{x, x' \sim p}\left[u_q\left(x, x'\right)\right]$

Theorem 1.2 Assume $k(x, x')$ is a positive definite kernel in the Stein class of p, with positive eigenvalues λ_j and eigenfunctions $e_j(x)$, then $u_q(x, x')$ is also a positive definite kernel, and can be rewritten into

$$u_q\left(x, x'\right)=\sum_j \lambda_j\left[\mathcal{A}_q e_j(x)\right]^{\top}\left[\mathcal{A}_q e_j\left(x'\right)\right] \tag{B.8}$$

In addition

$$\mathbb{S}(p, q)=\sum_j \lambda_j\left\|\mathbb{E}_{x \sim p}\left[\mathcal{A}_q e_j(x)\right]\right\|_2^2 \tag{B.9}$$

Theorem 1.3 Let $\mathcal{H}$ be the RKHS related to a positive definite kernel $k\left(x, x'\right)$ in the Stein class of p. Denote by $\boldsymbol{\beta}\left(x'\right)=\mathbb{E}_{x \sim p}\left[\mathcal{A}_q k_{x'}(x)\right]$, then

$$\mathbb{S}(p, q)=\|\boldsymbol{\beta}\|_{\mathcal{H}^d}^2 \tag{B.10}$$

Further, we have $\langle\boldsymbol{f}, \boldsymbol{\beta}\rangle_{\mathcal{H}^d}=\mathbb{E}_x\left[\operatorname{trace}\left(\mathcal{A}_q \boldsymbol{f}\right)\right]$ for $\boldsymbol{f} \in \mathcal{H}^d$, and hence

$$\sqrt{\mathbb{S}(p, q)}=\max _{\boldsymbol{f} \in \mathcal{H}^d}\left\{\mathbb{E}_x\left[\operatorname{trace}\left(\mathcal{A}_q \boldsymbol{f}\right)\right] \quad \text{s.t.} \quad \|\boldsymbol{f}\|_{\mathcal{H}^d} \leq 1\right\} \tag{B.11}$$

where the maximum is achieved when $f^*=\beta /\|\beta\|_{\mathcal{H}^d}=\mathbb{E}_{x \sim q}\left[\mathcal{A}_p k(x, \cdot)\right]$.

Definition of KL divergence:

$$\mathrm{KL}(q\|p) \equiv \mathbb{E}_q[\log q(x)] - \mathbb{E}_q[\log p(x)] \tag{B.12}$$

Theorem 1.4 Let $\boldsymbol{T}(x) = x + \epsilon f(x)$ and $q_{[T]}(z)$ the density of $z = \boldsymbol{T}(x)$ when $x \sim q(x)$, we have

$$\nabla_\epsilon \mathrm{KL}\left(q_{[\boldsymbol{T}]}\|p\right)\Big|_{\epsilon=0} = -\mathbb{E}_{x\sim q}\left[\operatorname{trace}\left(\mathcal{A}_p f(x)\right)\right] \tag{B.13}$$

Lemma 1.1 From theorem 1.3, consider all the perturbation directions $f(x)$

in the ball $\mathcal{B} = \left\{\phi \in \mathcal{H}^d : \|f\|_{\mathcal{H}^d} \le \mathbb{D}(q,p)\right\}$ of vector-valued RKHS $\mathcal{H}^d$, the direction of steepest descent that maximizes the negative gradient is

$$f^*_{q,p}(\cdot) = \mathbb{E}_{x\sim q}\left[\nabla_x \log p(x) k(x,\cdot) + \nabla_x k(x,\cdot)\right] \tag{B.14}$$

And in this moment $\nabla_\epsilon \mathrm{KL}\left(q_{[\boldsymbol{T}]}\|p\right)\Big|_{\epsilon=0} = -\mathbb{S}(q,p)$

The outcome in Lemma (1.1) implies an iterative algorithm that converts an initial reference distribution q_0 into the target distribution p. The process starts by implementing the transform $T_0^*(x) = x + \epsilon_0 \cdot f^*_{q_0,p}(x)$ on q_0. This transform reduces the Kullback-Leibler (KL) divergence by $\epsilon_0 \cdot \mathbb{S}(q_0,p)$, where ϵ_0 is a small step size. The outcome of this is a fresh allocation $q_1(x) = q_{0[T_0]}(x)$. Afterwards, an additional transformation $T_1^*(x) = x + \epsilon_1 \cdot f^*_{q_1,p}(x)$ can be used to reduce the KL divergence by $\epsilon_1 \cdot \mathbb{S}(q_1,p)$. By iteratively doing this procedure, a sequence of distributions $\{q_\ell\}_{\ell=1}^n$ is generated, which starts from q_0 and ends at p.

$$q_{\ell+1} = q_{\ell[T_\ell^*]}, \qquad \text{where} \qquad T_\ell^*(x) = x + \epsilon_\ell \cdot f^*_{q_\ell,p}(x). \tag{B.15}$$

Proof Theorem 1.1

Lemma 2.1 Assume $p(x)$ and $q(x)$ are smooth densities supported on $\mathbb{X}$ and $f(x)$ is in the Stein class of p,it can be derived that

$$\mathbb{E}_p\left[\mathcal{A}_q \boldsymbol{f}(x)\right] = \mathbb{E}_p\left[\left(s_q(x) - s_p(x)\right)\boldsymbol{f}(x)^\top\right] \tag{B.16}$$

Proof Since $\mathbb{E}_p\left[\mathcal{A}_p f(x)\right] = 0$, we have $\mathbb{E}_p\left[\mathcal{A}_q f(x)\right] = \mathbb{E}_p\left[\mathcal{A}_q \boldsymbol{f}(x) - \mathcal{A}_p \boldsymbol{f}(x)\right] = \mathbb{E}_p[(s_q(x)$ $-s_p(x))\boldsymbol{f}(x)^\top]$

1) Denote by $v\left(x,x'\right) = k\left(x,x'\right) s_q\left(x'\right) + \nabla_{x'} k\left(x,x'\right) = \mathcal{A}_q k_x\left(x'\right)$; applying Lemma 2.1 on $k(x,\cdot)$ with fixed x,

$$\begin{aligned}\mathbb{S}(p,q) &= \mathbb{E}_{x,x'\sim p}\left[\left(s_q(x) - s_p(x)\right)^\top k\left(x,x'\right)\left(s_q\left(x'\right) - s_p\left(x'\right)\right)\right] \\ &= \mathbb{E}_{x,x'\sim p}\left[\left(s_q(x) - s_p(x)\right)^\top \boldsymbol{v}\left(x,x'\right)\right]\end{aligned} \tag{B.17}$$

Because $k\left(\cdot,x'\right)$ is in the Stein class of p for any x', It can be shown that $\nabla_{x'} k\left(\cdot,x'\right)$ is also in the Stein class, since

$$\int_x \nabla_x\left(p(x)\nabla_{x'} k\left(x,x'\right)\right) dx = \nabla_{x'}\int_x \nabla_x\left(p(x) k\left(x,x'\right)\right) dx = 0, \tag{B.18}$$

and hence $v(\cdot, x')$ is also in the Stein class; apply Lemma 2.3 on $v(\cdot, x')$ with fixed x' gives

$$\begin{aligned} \mathbb{S}(p, q) &= \mathbb{E}_{x,x'\sim p}\left[\left(\boldsymbol{s}_q(x) - s_p(x)\right)^\top \boldsymbol{v}(x, x')\right] \\ &= \mathbb{E}_{x,x'\sim p}\left[s_q(x)^\top \boldsymbol{v}(x, x') + \text{trace}\left(\nabla_x \boldsymbol{v}(x, x')\right)\right] \end{aligned} \tag{B.19}$$

The result then follows by noting that $\nabla_x v(x, x') = \nabla_x k(x, x')\, s_q(x')^\top + \nabla_{x'x'} k(x, x')$.

Proof Theorem 1.2 Note that

$$\nabla_x k(x, x') = \sum_j \lambda_j \nabla_x e_j(x) e_j(x'), \qquad \nabla_{x,x'} k(x, x') = \sum_j \lambda_j \nabla_x e_j(x) \nabla_{x'} e_j(x')^\top \tag{B.20}$$

and hence

$$\begin{aligned} u_q(x, x') &= s_q(x)^\top k(x, x') s_q(x') + s_q(x)^\top \nabla'_x k(x, x') \\ &\quad + s_q(x')^\top \nabla_x k(x, x') + \text{trace}(\nabla_{x,x'} k(x, x')) \\ &= \sum_j \lambda_j \Big[s_q(x)^\top e_j(x) e_j(x') s_q(x') + s_q(x)^\top e_j(x) \nabla_{x'} e_j(x') \\ &\quad + s_q(x')^\top \nabla_x e_j(x) e_j(x') + \nabla_x e_j(x)^\top \nabla_{x'} e_j(x') \Big] \\ &= \sum_j \lambda_j \Big[s_q(x) e_j(x) + \nabla_x e_j(x) \Big]^\top \Big[s_q(x') e_j(x') + \nabla_{x'} e_j(x') \Big] \\ &= \sum_j \lambda_j \Big[\mathcal{A}_q e_j(x) \Big]^\top \Big[\mathcal{A}_q e_j(x') \Big] \end{aligned} \tag{B.21}$$

In addition,

$$\begin{aligned} \mathbb{S}(p, q) &= \mathbb{E}_{x,x'}\left[u_q(x, x')\right] \\ &= \sum_j \lambda_j \mathbb{E}_x\left[\mathcal{A}_q e_j(x)\right]^\top \mathbb{E}_{x'}\left[\mathcal{A}_q e_j(x')\right] \\ &= \sum_j \lambda_j \left\|\mathbb{E}_x\left[\mathcal{A}_q e_j(x)\right]\right\|_2^2. \end{aligned} \tag{B.22}$$

Proof of Theorem 1.3 Equation B.10 is first be poved by applying the reproducing property $k(x, x') = \langle k(x, \cdot), k(x', \cdot)\rangle_{\mathcal{H}}$ on the defination of $\mathbb{S}(p, q)$:

$$\begin{aligned} \mathbb{S}(p, q) &= \mathbb{E}_{x,x'\sim p}\left[\left(s_q(x) - s_p(x)\right)^\top k(x, x')\left(s_q(x') - s_p(x')\right)\right] \\ &= \mathbb{E}_{x,x'\sim p}\left[\left(s_q(x) - s_p(x)\right)^\top \langle k(x, \cdot), k(x, \cdot)\rangle_{\mathcal{H}} \left(s_q(x') - s_p(x')\right)\right] \\ &= \sum_{\ell=1}^d \left\langle \mathbb{E}_x\left[\left(s_q^\ell(x) - s_p^\ell(x)\right) k(x, \cdot)\right], \mathbb{E}_{x'}\left[k(x, \cdot)\left(s_q^\ell(x) - s_p^\ell(x)\right)\right]\right\rangle_{\mathcal{H}} \\ &= \sum_{\ell=1}^d \langle \boldsymbol{\beta}_\ell, \boldsymbol{\beta}_\ell \rangle_{\mathcal{H}} \\ &= \|\boldsymbol{\beta}\|_{\mathcal{H}^d}^2 \end{aligned} \tag{B.23}$$

where the fact that $\boldsymbol{\beta}(x') = \mathbb{E}_{x\sim p}[\mathcal{A}_q k_{x'}(x)] = \mathbb{E}_{x\sim p}[(s_q(x)k(x,x') + \nabla_x k(x,x')] = \mathbb{E}_x[(s_q(x) - s_p(x))\,k(x,x')]$ was used. In addition,

$$\begin{aligned}\langle \boldsymbol{f}, \boldsymbol{\beta}\rangle_{\mathcal{H}^d} &= \sum_{\ell=1}^{d} \left\langle f_\ell, \mathbb{E}_{x\sim p}\left[\left(s_q^\ell(x)k(x,\cdot) + \nabla_{x_\ell}k(x,\cdot)\right]\right\rangle_{\mathcal{H}} \\ &= \sum_{\ell=1}^{d} \mathbb{E}_{x\sim p}\left[\left(s_q^\ell(x)\,\langle f_\ell, k(x,\cdot)\rangle_{\mathcal{H}} + \langle f_\ell, \nabla_{x_\ell}k(x,\cdot)\rangle_{\mathcal{H}}\right] \\ &= \sum_{\ell=1}^{d} \mathbb{E}_{x\sim p}\left[\left(s_q^\ell(x)f_\ell(x) + \nabla_{x_\ell}f_\ell(x)\right] \\ &= \mathbb{E}_{x\sim p}[\mathrm{trace}(\mathcal{A}_q f(x))]\end{aligned} \tag{B.24}$$

where the fact that $\nabla_x f(x) = \langle f(\cdot), \nabla_x k(x,\cdot)\rangle_{\mathcal{H}}$ was used. The variational form Equation B.11 then follows the fact that $\|\boldsymbol{\beta}\|_{\mathcal{H}^d} = \max_{\boldsymbol{f}\in\mathcal{H}^d}\{\langle \boldsymbol{f}, \boldsymbol{\beta}\rangle_{\mathcal{H}^d}, \text{.s.t. } \|\boldsymbol{f}\|_{\mathcal{H}^d} \le 1\}$.

Finally, the $\boldsymbol{\beta}(\cdot) = \mathbb{E}_{x\sim p}[(s_q(x)k(x,\cdot) + \nabla_x k(x,\cdot)]$ is in the Stein class of p because $k(x,\cdot)$ and $\nabla_x k(x,\cdot)$ are in the Stein class of p for any fixed x (see the proof of Theorem 1.2).

Proof of Theorem 1.4

Denote by $q_{[\boldsymbol{T}^{-1}]}(z)$ the density of $z = \boldsymbol{T}^{-1}(x)$ when $x \sim q(x)$, then

$$q_{[\boldsymbol{T}^{-1}]}(x) = q(\boldsymbol{T}(x)) \cdot |\det(\nabla_x \boldsymbol{T}(x))|\,. \tag{B.25}$$

By the change of variable, it can be derived that

$$\mathrm{KL}\left(q_{[\boldsymbol{T}]}\|p\right) = \mathrm{KL}\left(q\|p_{[\boldsymbol{T}^{-1}]}\right) \tag{B.26}$$

and hence

$$\mathrm{KL}\left(q_{[\boldsymbol{T}]}\|p\right) = \mathrm{KL}\left(q\|p_{[\boldsymbol{T}^{-1}]}\right) = -\mathbb{E}_{x\sim q}\left[\nabla_\epsilon \log p_{[\boldsymbol{T}^{-1}]}(x)\right] \tag{B.27}$$

With $p_{[T^{-1}]}(x) = p(T(x) \cdot |det(\nabla_x T(x)|$, it can be derived that:

$$\nabla_\epsilon \log p_{[\boldsymbol{T}^{-1}]}(x) = s_p(\boldsymbol{T}(x))^\top \nabla_\epsilon \boldsymbol{T}(x) + \mathrm{trace}\left((\nabla_x \boldsymbol{T}(x))^{-1} \cdot \nabla_\epsilon \nabla_x \boldsymbol{T}(x)\right) \tag{B.28}$$

Therefore,

$$\nabla_\epsilon \mathrm{KL}\left(q_{[\boldsymbol{T}]}\|p\right) = -\mathbb{E}_q\left[s_p(\boldsymbol{T}(x))^\top \nabla_\epsilon \boldsymbol{T}(x) + \mathrm{trace}\left((\nabla_x \boldsymbol{T}(x))^{-1} \cdot \nabla_\epsilon \nabla_x \boldsymbol{T}(x)\right)\right] \tag{B.29}$$

When $T(x) = x + \epsilon f(x)$ and $\epsilon = 0$, it can be easily got that:

$$\boldsymbol{T}(x) = x, \quad \nabla_\epsilon \boldsymbol{T}(x) = f(x), \quad \nabla_x \boldsymbol{T}(x) = I, \quad \nabla_\epsilon \nabla_x \boldsymbol{T}(x) = \nabla_x f(x), \tag{B.30}$$

where I is the identity mateix.Put them back to equation B.29, it gives that result that

$$\nabla_\epsilon \mathrm{KL}\left(q_{[\boldsymbol{T}]}\|p\right)\Big|_{\epsilon=0} = -\mathbb{E}_{x\sim q}[\mathrm{trace}(\mathcal{A}_p f(x))] \tag{B.31}$$

C

Computation of Uncertainties

In the appendix, the procedures for calculating a predictive distribution (Algorithm 2), epistemic uncertainty (Algorithm 3), and aleatoric uncertainty (Algorithm 4) are provided. The Bayesian Neural Network (BNN) facilitates quick assessment of weight samples, though precise averaging is necessary to distinguish between epistemic and aleatoric uncertainties. After obtaining the weight distribution after training, use monto carlo sampling and then average it to obtain the final result.

Algorithm 2 Calculate the predictive distribution for a dataset $\{x_i\}_{i=1}^{N}$ using the posterior distribution $p(\mathbf{w}|\theta)$ of a BNN, incorporating a specified number of epistemic samples N_{epi} and aleatoric samples N_{ale}.

```
Require: A posterior with density function $p(w|\theta)$ and number of samples
1: for $i = 1, \ldots, N$ do
2:     for $j = 1, \ldots, N_{epi}$ do
3:         Sample $w_j \sim p(\mathbf{w}|\theta)$
4:         Compute BNN prediction of $y_\mu$ and $y_\sigma$
5:         for $k = 1, \ldots, N_{ale}$ do
6:             Sample $\mathbf{y} \sim \mathcal{N}(y_\mu, \mathrm{diag}(y_\sigma))$ and append to collection of predictions
7:         end for
8:     end for
9: end for
```

Algorithm 3 Calculate an estimate of the epistemic uncertainty associated with an input datum x

```
Require: A posterior with density function $p(w|\theta)$ and number of epistemic samples $N_{epi}$
1: for $j = 1, \ldots, N_{epi}$ do
2:     Sample $w_j \sim p(\mathbf{w}|\theta)$
3:     Calculate BNN prediction of $\mathbf{y}_{\boldsymbol{\mu}}$ corresponding to $\mathbb{E}_{\mathbf{w_j} \sim q(\mathbf{w}|\theta)}[\mathbf{y} \mid \mathbf{x}, \boldsymbol{D}]$ and append to collection $\left\{\mathbf{y}_{\boldsymbol{\mu}_i}\right\}_i^{N_{epi}}$
4: end for
5: Compute the sample variance of collection $\{\mathbf{y}_{\mu_i}\}_i^{N_{epi}}$ corresponding to $\mathrm{Var}\left(\mathbb{E}_{q(\mathbf{w}|\theta)}[\mathbf{y} \mid \mathbf{x}, \boldsymbol{D}]\right)$
```

Algorithm 4 Calculate an estimate of the aleatoric uncertainty associated with an input datum x

```
Require: A posterior with density function $p(w|\theta)$ and number of epistemic samples $N_{ale}$
1: for $j = 1, \ldots, N_{ale}$ do
2:     Calculate BNN prediction of $\mathbf{y}_{\boldsymbol{\sigma}}$ corresponding to $\mathrm{Var}(\mathbf{y} \mid \mathbf{x}, \boldsymbol{D})$ and append to collection $\{\mathbf{y}_{\boldsymbol{\sigma}_i}\}_i^{N_{ale}}$
3: end for
4: Calculate the mean of collection $\{\mathbf{y}_{\sigma_i}\}_i^{N_{ale}}$ corresponding to $\mathbb{E}_{q(\mathbf{w}|\theta)}[\mathrm{Var}(\mathbf{y} \mid \mathbf{x}, \boldsymbol{D})]$
```