Alexis Balayre, Deng Chuan Chang, Chung-Yueh Cheng, Julien Godfroy, Lyderic Faure, Majuran Chandrakumar, Marcell Gyorei

Applications in Practical High-End Computing

School of Aerospace, Transport and Manufacturing
Computational Software of Techniques Engineering

MSc
Academic Year: 2023 - 2024

Supervisors: Dr Irene Moulitsas, Dr Jun Li, and Dr Tamas Jozsa
3rd May 2024

School of Aerospace, Transport and Manufacturing
Computational Software of Techniques Engineering

MSc

Academic Year: 2023 - 2024

Alexis Balayre, Deng Chuan Chang, Chung-Yueh Cheng, Julien Godfroy, Lyderic Faure, Majuran Chandrakumar, Marcell Gyorei

Applications in Practical High-End Computing

Supervisors: Dr Irene Moulitsas, Dr Jun Li, and Dr Tamas Jozsa
3rd May 2024

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

Academic Integrity Declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Abstract

This project explores the integration of machine learning techniques to turbulence modeling in the realm of computational fluid dynamics (CFD). The study focus on overcoming the limitations of traditional Reynolds-Averaged Navier-Stokes (RANS) models by leveraging on machine learning techniques, Physics-Informed Neural Networks (PINNs) and Sparse Identification of Nonlinear Dynamical Systems (SINDy) approach. The aims are to develop predictive models that can simulate turbulent flows accurately under specific conditions and improve the efficiency of CFD simulation through the use of high-performance computers.The methodology involves the building of machine learning models and simulations environment, using Direct Numerical Simulation (DNS) data, emphasising on improving simulation capabilities across specific flow regimes. This process entails rigorous testing and validation to optimise the parameters before integrating into custom software. Additionally, a Streamlit application has been developed to facilitate user interaction and visualisation of turbulence modeling predictions. Results from the study indicates that the PINNs model significantly outperform conventional models in capturing the dynamics of turbulent flows in complex boundary layer contexts. The integration of these models into standard CFD workflows has the potential to drastically enhance the prediction accuracy of turbulent behaviors, offering substantial reductions in computational costs.

Keywords:
Physics-informed Neural Networks (PINNs); Computational Fluid Dynamics (CFD); Reynolds-averaged Navier-stokes (RANS); Sparse Identification of Nonlinear Dynamical Systems (SINDy); Turbulent Channel Flows

Acknowledgements

This project would have not been possible without the generous support of many people during the two months of the project work.

We would like to express our gratitude to our project supervisors Dr Irene Moulitsas, Dr Jun Li, and Dr Tamas Jozsa who have assisted the team through for solving numerous challenges encountered during this project.

We are grateful to the School of Aerospace, Transport and Manufacturing, Cranfield University, for providing essential resources that facilitated our progress in this project.

Lastly, we would like to express deep gratitude to our families members for their encouragement and motivation throughout the duration of this project.

Table of Contents

Academic Integrity Declaration i

Abstract ii

Acknowledgements iii

Table of Contents iv

List of Figures ix

List of Tables xi

List of Abbreviations xii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Motivation . . . 1
- 1.3 Aim . . . 1
- 1.4 Objectives . . . 2

2 Literature Review 3
- 2.1 Computational Fluid Dynamics . . . 3
  - 2.1.1 Navier-Stokes Equations . . . 3
  - 2.1.2 RANS Closure Problem . . . 3
  - 2.1.3 Models . . . 3
- 2.2 ANSYS Fluent Package . . . 4
- 2.3 Physics-informed Neural Networks (PINNs) . . . 4
  - 2.3.1 Fundamental Concepts . . . 4
  - 2.3.2 Turbulence Modeling for PINNs . . . 4
  - 2.3.3 Methodological Advances . . . 5
  - 2.3.4 Challenges and Future Directions . . . 5
- 2.4 PySINDy . . . 5
  - 2.4.1 Fundamental Concepts . . . 5
- 2.5 Transformer . . . 6
  - 2.5.1 Fundamental Concepts . . . 6
  - 2.5.2 Applications of Transformer . . . 7
  - 2.5.3 Future Directions . . . 7
- 2.6 Optimisation – Genetic Algorithm (GA) . . . 7
  - 2.6.1 PyGAD and DEAP . . . 7
  - 2.6.2 Applications and Challenges of GA . . . 7
  - 2.6.3 Future Directions . . . 8
- 2.7 Feature Extractions . . . 8
  - 2.7.1 PCA . . . 8
    - 2.7.1.1 Fundamental Concepts . . . 8
    - 2.7.1.2 Applications and Limitations . . . 9

2.7.2 t-SNE . . . 9
2.7.2.1 Fundamental Concepts . . . 9
2.7.2.2 Applications and Limitations . . . 9
2.7.3 CCA . . . 10
2.7.3.1 Fundamental Concepts . . . 10
2.7.3.2 Applications and Limitations . . . 10
2.7.4 PAF . . . 10
2.7.4.1 Fundamental Concepts . . . 10
2.7.4.2 Applications and Limitations . . . 11
2.7.5 ULS . . . 11
2.7.5.1 Fundamental Concepts . . . 11
2.7.5.2 Applications and Limitations . . . 11
2.7.6 GLS . . . 12
2.7.6.1 Fundamental Concept . . . 12
2.7.6.2 Applications and Limitations . . . 12

3 Methodology **13**
3.1 ANSYS (RANS) approach . . . 13
3.1.1 Methodology . . . 13
3.1.2 DNS Database . . . 13
3.1.3 Mesh Generation . . . 14
3.1.4 Simulation Set-up . . . 16
3.1.4.1 GUI . . . 16
3.1.4.2 TUI . . . 17
3.1.5 Outputs . . . 19
3.2 PINNs . . . 20
3.2.1 Data Exploration . . . 20
3.2.2 Targets & Features Selection . . . 21
3.2.2.1 Targets Selection . . . 21
3.2.2.2 Features Selection . . . 21
3.2.2.3 Justification for Feature Redundancy . . . 22
3.2.3 PINNs Model Workflow . . . 22
3.2.4 Loss Function Design . . . 24
3.2.4.1 Physic Loss Function . . . 24
3.2.5 Model Architecture . . . 25
3.2.5.1 Layers & Nodes Configuration . . . 26
3.2.5.2 Activation Functions . . . 26
3.2.5.3 Optimizer and Learning Rate Scheduler Configuration . . . 26
3.2.5.4 Balancing Loss Terms . . . 27
3.2.5.5 Hyperparameter Tuning . . . 27
3.2.6 Model Implementation . . . 27
3.2.6.1 Framework (PyTorch Lightning) . . . 27
3.2.6.2 Core Components . . . 27
3.2.7 Training Pipeline . . . 28
3.2.7.1 Data Sets Preparation . . . 28
3.2.7.2 Training Cycles . . . 29
3.2.7.3 Reproducibility of the Training . . . 30
3.3 PySINDy Models . . . 30

3.3.1 PySINDy . . . 30
3.3.1.1 Selection of the DNS variables . . . 31
3.3.1.2 Data processing . . . 31
3.3.1.3 Data partitioning . . . 31
3.3.1.4 Selection of hyperparameters (Feature_Library and parameters in optimizer) . . . 32
3.3.1.5 Training, prediction and evaluation of the PySINDy model . . . 32
3.3.2 PySINDy combined with PCA . . . 33
3.3.3 PySINDy optimised by GA . . . 33
3.4 Transformer . . . 34
3.4.1 Data processing . . . 34
3.4.2 Data partitioning . . . 34
3.4.3 Model Architecture . . . 35
3.4.4 Evaluation . . . 35
3.4.5 Visualisation . . . 35

4 Results & Discussions **36**
4.1 RANS models results and discussion . . . 36
4.1.1 Streamwise velocity . . . 36
4.1.2 Covariances . . . 37
4.2 PINNs Model . . . 37
4.2.1 Hyper-parameters Tuning . . . 38
4.2.2 Training Step . . . 39
4.2.3 Test Step . . . 41
4.2.3.1 Best Trained Model Selection . . . 41
4.2.3.2 Best Trained Model Inference Plots . . . 42
4.2.3.3 Best Trained Model Relative ℓ_2-norm of Errors Analysis . . . 43
4.2.3.4 Best Trained Model Mean Square Error Analysis . . . 44
4.2.4 Overall Conclusion . . . 44
4.3 Different PySINDy models Results Analysis . . . 45
4.3.1 PySINDy . . . 45
4.3.1.1 Interpolation Method . . . 45
4.3.1.2 Output Equations . . . 46
4.3.1.3 Comparison with DNS Data . . . 47
4.3.1.4 Improving PySINDy Model . . . 47
4.3.2 PySINDy combined with PCA . . . 48
4.3.3 PySINDy optimised by GA . . . 50
4.3.4 Conclusion on PySINDy Models . . . 51
4.4 Transformer . . . 52
4.4.1 Model Performance Evaluation . . . 52
4.4.2 Comparative Analysis with PINNs . . . 53
4.4.3 Conclusion on Transformer . . . 53
4.5 Comparisons of ML and RANS Models with DNS . . . 53

5 Software Development **56**
5.1 User Requirements . . . 56
5.1.1 Cardinal Requirements (Must-have) . . . 56
5.1.2 Non-cardinal Requirements (Good-to-have) . . . 56

5.2 Basic Software Architecture . . . 57
5.3 User Interface (UI) Implementation . . . 57
5.3.1 app.py . . . 58
5.3.2 pysindy_streamlit.py . . . 58
5.3.3 pinns_streamlit.py . . . 58
5.3.4 comparisons_streamlit.py . . . 58
5.4 Cloud-based Software Architecture . . . 59
5.5 Testing Case . . . 60
5.5.1 Accuracy . . . 60
5.5.2 Speed . . . 61
5.5.3 Scalability . . . 62
5.5.4 Reliability . . . 63

6 Marketing Strategy Overview **65**
6.1 Marketing Background . . . 65
6.2 Competitors Analysis . . . 66
6.3 Unique Selling Proposition (USP) . . . 66
6.4 Marketing Mix . . . 66
6.5 Strategic Planning . . . 67

7 Project Plan **68**
7.1 Group Dynamics . . . 68
7.2 Roles & Responsibilities . . . 68
7.3 Challenges . . . 69
7.4 Risk Register . . . 69
7.5 Branching Models . . . 69
7.6 Quality Plan . . . 69
7.6.1 Introduction . . . 69
7.6.2 Testing Scope . . . 70
7.6.2.1 Platform, Applications and Services . . . 70
7.6.2.2 Operational Policies & Procedures . . . 70
7.6.2.3 Test Deliverables . . . 71
7.6.2.4 Exclusions . . . 71
7.6.2.5 Quality Risks . . . 71
7.6.3 Test Organisation . . . 71
7.6.4 Test Approach . . . 71
7.6.4.1 Test Lifecycle . . . 71
7.6.5 Test Planning . . . 72
7.6.6 Test Management Tooling . . . 73
7.6.6.1 Test Environment . . . 74
7.6.6.2 The Generic Test Process . . . 74
7.7 Delivery Plan . . . 74
7.8 Task Management . . . 75

8 Conclusions **76**
8.1 Overall . . . 76
8.2 Future Works . . . 76

References **77**

1 User Requirements **80**
1.1 Understand the Problem . . . 80
1.2 Identify User Needs . . . 80
1.3 Define Specific Requirements . . . 81
1.3.1 Data Requirements . . . 81
1.3.2 Functional Requirements . . . 81
1.3.3 Performance Requirements . . . 82
1.3.4 System Requirements . . . 82
1.3.5 Constraints and Limitations . . . 83

2 Software Streamlit **84**

3 Project Plan **89**
3.1 Risk Register . . . 89
3.2 Branching Models . . . 90
3.2.1 Github-flow branching model . . . 90
3.2.2 Git-flow branching model . . . 90
3.3 Reproducibility - list of tools . . . 90
3.4 ML Checklists . . . 90
3.5 Test lifecycle . . . 90
3.6 Dual Track Agile with CRISP-ML . . . 92
3.7 RCTM . . . 92
3.8 Generic Test Process . . . 94
3.9 Gantt chart . . . 95

4 Group Meeting Minutes **98**

List of Figures

3.1 ANSYS Methodology Pipeline . . . 13
3.2 Example of Simulation from Published Paper . . . 14
3.3 Mesh Geometry Generated . . . 15
3.4 ICEM space set up . . . 16
3.5 Line set-up . . . 17
3.6 Turbulent Channel Flow . . . 20
3.7 PINNs Model Workflow Diagram . . . 24

4.1 Stream wise velocity plots for RANS models and DNS . . . 36
4.2 Covariances of uu, vv and uv (top, mid, bottom) for $Re_\tau \approx 2000$ . . . 37
4.3 Training Losses Over Epochs for Four Cycles . . . 40
4.4 Training & Validation Loss Over Epochs for Four Cycles . . . 41
4.5 MSE Over Training Cycles With Test Dataset . . . 42
4.7 Comparison of PINNs predictions and DNS Data . . . 43
4.8 Representation of velocity, covariances and variances of velocity fluctuations as a function of y^+ for different Reynolds numbers for DNS data (before interpolation) . . . 45
4.9 Representation of velocity, covariances and variances of velocity fluctuations as a function of y^+ for different Reynolds numbers for DNS data (after interpolation) 46
4.10 Optimal hyperparameter obtained through the grid search algorithm . . . 46
4.11 Output equations generated by PySINDy . . . 47
4.12 Comparison of PySINDy predictions and DNS Data (Re = 2000) . . . 47
4.13 Representation of the inertia (explained variance) of the axes generated by PCA 48
4.14 PySINDy equations generated after using PCA . . . 49
4.15 Comparison of PySINDy predictions combining PCA and DNS Data (Re = 2000) 49
4.16 PySINDy equations with the optimal coefficients obtained by GA . . . 50
4.17 Comparison of PySINDy predictions optimised by GA and DNS Data (Re = 2000) . . . 51
4.18 Training and Validation Loss curves over epochs. . . . 52
4.19 Comparison between PINNs, PySINDy, k-ω, and RSM Model Simulations against DNS Data (Re = 2000) . . . 54

5.1 General software architecture design . . . 57
5.2 Screenshot of Streamlit UI . . . 58
5.3 AWS-based software architecture . . . 59
5.4 Streamlit App Scalability Test . . . 63

6.1 The trend of CFD market . . . 65

7.1 Technical Skill Results . . . 68

1.1 Illustration of Boeing Organisational Chart . . . 81

2.1 PySINDy User Inferface . . . 84
2.2 Results of PySINDy . . . 85

2.3 PINNs User Interface 86
2.4 Results of PINNs 87
2.5 Comparison Results 87
2.6 Comparison Results 88

3.1 Risk Register 89
3.2 Test Lifecycle 91
3.3 CRIPS-ML 92
3.4 Gantt chart, page 1 96
3.5 Gantt chart, page 2 97

List of Tables

3.1 Surface Name Attribution 15
3.2 First Element Space Table 16

4.1 Selection of the best initial learning rate on a 4-layer model with 128 nodes/layer using the total MSE on the test dataset 38
4.2 Selection of the best model size with the final activation function (ELU) using the total MSE on the test dataset 38
4.3 Selection of the best activation function on a 4-layer model with 128 nodes/layer using the total MSE on the test dataset 39
4.4 Initial weights selection for physic and data losses in a 4-layer, 128 nodes/layer model based on total MSE on the test dataset 39
4.5 Relative ℓ_2-norm of errors, in the predictions using PINNs over the Test Dataset 44
4.6 MSE over the Test Dataset 44

5.1 Results of the accuracy test for the PySINDy model of the application 60
5.2 Results of the accuracy test for the PINNs model of the application 61
5.3 Results of the speed test for the PySINDy model of the application 61
5.4 Results of the speed test for the PINNs model of the application 62
5.5 Results of the scalability test for the PySINDy model of the application 62
5.6 Results of the scalability test for the PINNs model of the application 63
5.7 Results of the reliability test for the PySINDy model of the application 64
5.8 Results of the reliability test for the PINNs model of the application 64

6.1 Marketing Mix 66

3.1 RCTM 1 92
3.2 RCTM 2 93
3.3 RCTM 3 94

List of Abbreviations

AI	Artificial Intelligence
AdaGrad	Adaptive Gradient Algorithm
CAD	Computer-Aided Design
CCA	Canonical Correlation Analysis
CFD	Computational Fluid Dynamics
DEAP	Distributed Evolutionary Algorithms in Python
DNS	Direct Numerical Simulation
DL	Deep Learning
ELU	Exponential Linear Unit
GA	Genetic Algorithm
GLS	Generalised Least Squares
GUI	Graphical User Interface
IDE	Implicit Differential Equation
LR	Learning Rate
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean-Square Error
NLP	Natural Language Process
NN	Neural Network
OLS	Ordinary Least Squares
PAF	Principal Axis Factor
PCA	Principal Component Analysis
PINN	Physics-informed Neural Network
PINNs	Physics-informed Neural Networks
PDE	Partial Differential Equation
RAG	Red-Amber-Green
RANS	Reynolds-Averaged Navier–Stokes
ReLu	Rectified Linear Unit
RNN	Recurrent Neural Network
RMSProp	Root Mean Square Propagation
RSM	Reynolds Stress Model
RMSE	Root Mean Square Error
RST	Reynold Stress Tensor
SELU	Scaled Exponential Linear Unit
SINN	Shallow Inverse Neural Network
SINDy	Sparse Identification of Non-linear Dynamical System
SSL	Secure Sockets Layer
STLSQ	Sequential Threshold Least Squares
TANH	Hyperbolic Tangent
TKE	Turbulence Kinetic Energy
TLS	Transport Layer Security
t-SNE	t-distributed Stochastic Neighbor Embedding
TUI	Text-based User Interface
ULS	Unweighted Least Squares

Chapter 1 – Introduction

1.1 Background

The Reynolds-averaged Navier-Stokes (RANS) equations are the cornerstone for using computational fluid dynamics (CFD) for estimating fluid flow during turbulent conditions. The RANS model effectively decomposes fluid flow into mean and fluctuating components, ranging from simpler zero-equation to a more complex two-equation types. This decomposition facilitates accurate fluid flow simulations across a variety of operating conditions. This include high-speed and turbulent boundary layers. Such capabilities are essential for reliably predicting fluid behaviour under diverse environmental and operational scenarios [5, 23].

The applications of RANS models are extensively uterlised beyond aerospace and automotive sectors, playing a pivotal role in marine engineering, wind engineering, and industrial processes. These models incorporate key factors such as mean velocity (U), pressure (P), time variable (t), Reynolds Stress Tensor (τ) which are essential for the design and optimisation of machines and vehicles. Despite the wide use of RANS model, simulating the unpredictable behavior of turbulence remains a challenge. To address this, methods such as Boussinesq hypothesis, k-ε, and k-ω models are employed, enhancing the practically of fluid simulations for daily applications in research and engineering [38, 26].

1.2 Motivation

Computational fluid dynamics (CFD) plays a vital role in the aerospace and automotive sectors, which are used to optimise the aerodynamic design of aircraft and vehicles. This reduces drag and fuel consumption [24]. For instance, Boeing company uses CFD to redesign the wings of the 747 passenger aircraft, reducing the total drag by 5% [19]. To calculate the cost savings of a long-distance passenger aircraft with a 25-year service life and an annual fuel cost of around US$10 million, a 5% increase in lift-to-drag ratio can save approximately US$1 million in fuel costs per aircraft per annum [19]. In addition, one research shows that CFD can reduce drag and environmental pollution by up to 20%, improving fuel efficiency and reducing operating costs [10].

Inspired by the significant impact of CFD in multiple fields, this study aims to employ machine learning to improve the accuracy of predictions to resolve important issues such as engineering design and operational safety. The optimised design is expected to reduce fuel consumption by about 20%, greatly reducing the carbon emissions to the environment while promoting friendly and efficient designs.

1.3 Aim

The aim of this study is to perform an in-depth analysis of Direct Numerical Simulations (DNS) datasets and implement potential Machine Learning (ML) or Deep Learning (DL) approaches to develop robust turbulence models that can predict turbulent flows. A basic software will be

developed to facilitate the use of various predictive models and allow comparative analysis of their results against DNS data for visualisation. Finally, the research is crucial to identify the advantages and potential limitations of the different approaches to the prediction of turbulent flows to guide future research efforts, ensuring that further works are directed towards the most effective methodologies.

1.4 Objectives

Objectives of the project are as follows:

- Simulation of channel flows using CFD software
- Implementation of different machine learning techniques
- Assessment of the different machine learning techniques
- Developing a basic software for predicting turbulent flow

This project involves using CFD software to compute channel flow simulations. These simulations will serve to identify possible leads of improvements for machine learning, as well as comparing performances on the case between classic CFD package and machine learning. The data computed by these simulation will also be used as training data for ML models.

The project involves the design and training of ML models based on DNS data. An emphasis will be put on the encapsulation of the physical phenomena characteristic of turbulence in the development of the models. An iterative process of optimisation in the selection of model hyperparameters will be performed to ensure that the models not only conform to the data but also adhere to the physics of turbulent flows.

This research also includes the evaluation of the prediction accuracy and the computational efficiency of the ML models. A set of metrics such as the coefficient of determination (R^2), the Mean Square Error (MSE), the Root Mean Square Error (RMSE) and others are being used to assess the performance of the different models, including a direct comparison between their predictions and the DNS data. This assessment is needed in order to determine the suitability of the different models for turbulent channel flow.

Finally, this project will require the development of user-friendly software to exploit ML models to predict turbulent flows. This software will facilitate the computation of predictions, the visualisation of model results and the comparison of these results with DNS data. In addition, the design of this software should be evolutive to allow integration of new ML models for future works.

Chapter 2 – Literature Review

2.1 Computational Fluid Dynamics

CFD is a branch of fluid mechanics that employs numerical methods and algorithms to analyse and solve problems involving fluid flows. Rooted in the 1960s, it evolved rapidly with advances in computing technology. Initially developed for aerospace applications, it expanded to diverse fields due to its versatility, such as automotive engineering, environmental science and biomedical research. CFD simulations enable engineers and scientists to predict fluid behavior, such as velocity, pressure, and temperature distributions, in complex geometries and under various operating conditions. By providing insights into fluid flow phenomena, CFD plays a crucial role in the design optimisation, performance evaluation, and troubleshooting of engineering systems. This section compiles the key concepts to understand the basic theory behind CFD and further research in this report.

2.1.1 Navier-Stokes Equations

Navier-Stokes equations serve as the fundamental equations governing fluid motion in CFD. They mathematically describe the conservation of mass, momentum and energy, providing a framework to simulate and predict fluid behaviour in complex systems. By solving these partial differential equations numerically, CFD models can predict flow characteristics such as velocity, pressure, and temperature distributions.

2.1.2 RANS Closure Problem

The RANS equations, while effective for simulating mean flow behaviour in turbulent flows, faced challenges due to the RANS closure problem. This arises from the need to model the effects of unresolved turbulent fluctuations. Turbulence models, such as k-ε or k-ω, aim to complete the system by expressing turbulent quantities in terms of mean flow variables. However, these models rely on assumptions and simplifications, leading to inaccuracies in certain flow conditions. The sensitivity to flow conditions and the lack of universal models contribute to ongoing research efforts. Improving closure strategies remains crucial for enhancing the accuracy and reliability of RANS simulations.

2.1.3 Models

The k-ω and Reynolds Stress Model (RSM) are two popular turbulence models used in RANS simulations to predict turbulent flow behaviour.

The k-ω model is a two-equations turbulence model that solves transport equations for turbulent kinetic energy (k) and specific dissipation rate (ω). It excels in accurately predicting near-wall flows due to its enhanced treatment of near-wall turbulence. The k-ω model offers robustness and efficiency, making it suitable for a wide range of engineering applications, especially those involving complex geometries and boundary layers. However, its performance can be limited in highly anisotropic flows or those with strong rotational effects.

In contrast, the Reynolds Stress Model (RSM) directly solves for the RST components, offering a more comprehensive representation of turbulence. It captures anisotropic turbulence effects more accurately and is suitable for flows with significant rotational or strain-rate effects. The RSM provides detailed insights into turbulence behaviour, making it valuable for research and engineering applications requiring high-fidelity simulations. However, it is computationally expensive and sensitive to grid resolution and modeling assumptions, limiting its applicability in some cases. Choosing between the k-ω and RSM depends on the flow characteristics, computational resources, and accuracy requirements of the simulation.

2.2 ANSYS Fluent Package

Among numerous CFD packages available, Fluent stands out as one of the most widely used and versatile tools. It is robust, accurate, and user-friendly interface have made it indispensable in various sectors such as aerospace, automotive, energy, and biomedical engineering. The Fluent package is made of software that allows the team to work on the pre-processing and post processing aspects of CFD.

ICEM CFD is one of the software included in the ANSYS suite, it is a pre-processing software specifically designed for generating high-quality meshes for CFD. It is an integral component of the Fluent package, renowned for its robust meshing capabilities and versatility. It offers advanced tools for importing computer-aided design (CAD) geometries, meshing with structured, unstructured, or hybrid elements, and creating boundary layers with specified growth rates. Mesh quality control and adaptation techniques ensure accuracy and stability, while parallel meshing expedites large-scale simulations. Seamless integration with Fluent allows for efficient data transfer, making ICEM CFD a preferred choice for engineers seeking reliable and versatile mesh generation capabilities.

2.3 Physics-informed Neural Networks (PINNs)

Physics-Informed Neural Networks (PINNs) represent a novel approach that integrates differential equation solvers with the flexibility and power of deep learning. By embedding physical laws as part of the training process, PINNs offer the potential to solve complex Partial Differential Equations (PDEs) across various domains, including fluid dynamics, structural engineering, and beyond.

2.3.1 Fundamental Concepts

PINNs leverage the structure of neural networks to approximate the solutions of PDEs. These models are informed by physical laws, which are encoded directly into the loss function as regularisation terms. This approach ensures that the solutions not only fit the available data but also comply with the governing physical principles.

2.3.2 Turbulence Modeling for PINNs

Since the introduction of the PINN model design, many researchers have deployed it to solve the RANS equations. These include the following papers, which have made real advances in turbulence prediction models: Sandberg, Hanrahan, Kozul, Jagode, Anzt, Juckeland, and Ltaief

[31], Luo, Vellakal, Koric, Kindratenko, and Cui [22], Eivazi, Tahani, Schlatter, and Vinuesa [14], Pioch, Harmening, Müller, Peitzmann, Schramm, and el Moctar [28].

2.3.3 Methodological Advances

Several methodological advances have been instrumental in enhancing the performance and applicability of PINNs. For instance, Bischof and Kraus [4] introduced a Mixture-of-Experts-Ensemble Meta-Learning approach, which significantly improves PINN's ability to solve complex PDEs by utilising an ensemble of specialised neural networks. Additionally, Lagaris, Likas, and Fotiadis [21] pioneered the use of neural networks for solving differential equations, laying the groundwork for the development of PINNs.

Choosing the right activation function is critical for the success of PINNs, as it affects the model's ability to capture complex behaviours governed by the PDEs. Traditional activation functions like Rectified Linear Unit (ReLU) may not be suitable due to their limited differentiability. Instead, functions with higher order differentiability, such as the sine function, have been explored by Raissi, Perdikaris, and Karniadakis [30], Bischof and Kraus [4].

2.3.4 Challenges and Future Directions

Despite their promising capabilities, PINNs faced several challenges, including the need for large computational resources, difficulty in modeling highly non-linear systems, and the selection of appropriate hyperparameters. Future research directions include the development of more efficient training algorithms, exploration of new architectures, and the extension of PINNs to a broader range of applications.

2.4 PySINDy

PySINDy is recognised as a type of machine learning and has gained attention for its ability to identify actuated systems, PDEs and implicit differential equations (IDEs). It applies the concept of sparse identification of non-linear dynamics, using sparse matrices to provide a new approach in computational modeling [8].

2.4.1 Fundamental Concepts

The concept of PySINDy is to generate dynamic equations for each variable used. When multiple variables are used, PySINDy will not integrate these equations to form a single equation for the whole system. This is advantageous as it allows a better understanding of the behaviour of individual components within the whole dynamic system. The implementation of PySINDy has been developed based on the Sparse Identification of Non-linear Dynamical Systems (SINDy) method which is used to discover the connections between each variable that follows the time series in the dynamical system models. This method is instrumental in simplifying the process of modeling complex systems, enabling researchers to remain focused on the important areas of system dynamics [6].

The use of the SINDy method is to solve the modeling of dynamical systems as a sparse regression problem. Using the time series data matrix X of state variables and library functions $\theta(X)$,

SINDy seeks to find a sparse coefficient matrix Ξ to satisfy the following conditions [12]:

$$\dot{X} \approx \Theta(X)\Xi$$

Each row corresponds to a partial equation for a state variable. Subsequently, SINDy will solve the matrix approximation problem to reveal the interrelationships between each variable in dynamic systems. It is well-known for PySINDy to use at analysing the Lorentz system, the classic chaotic system [12]. It is a simple nonlinear differential equation system which composed of three variables that can produce extremely complex chaotic behavior, and PySINDy can accurately identify the governing equations of it from time series data, demonstrating its powerful ability to capture the essential dynamics of chaotic systems. Moreover, because PySINDy's input is suitable for time series format; this requirement aligns well to the method's focus on time series data processing, making it suitable for a wide range of applications from fluid dynamics to biological systems.

2.5 Transformer

Before the advent of Transformers, Recurrent Neural Networks (RNNs) were the predominant architecture for handling sequence modeling tasks and regression problems [33]. RNNs process sequential data by iterating through elements of the sequence and maintaining a hidden state vector that summarises all the previous inputs. Consequently, RNNs can capture patterns in sequences as well as establishing dependencies, making them appropriate for problems such as time series, language data, and translation. However, RNNs suffer from several limitations. One of the prominent drawback is the problem of exploding gradients, which hampers the model's ability to effectively learn dependencies over longer sequences. Additionally, RNNs face challenges with parallelisation during training due to their sequential data processing nature, which inherently limits the speed of computation compared to architectures that allow simultaneous processing of inputs.

2.5.1 Fundamental Concepts

The Transformer is a new neural network architecture introduced in the seminal 2017 paper "Attention is all you need" [35]. It has outperformed existing models in natural language processing (NLP) across diverse downstream tasks and applications. Transformer stands out from other neural networks due to its ability of working with long-range dependencies and parallelisable calculations. Unlike the RNNs, which process input sequentially due to their recurrent nature, Transformers utilises attention mechanism that allow them to perceive the entire input sequence simultaneously when generating outputs. The architecture is compromised of two main parts: the encoder, which maps the input sequence to a higher dimension, and the decoder, which generates the output based on the encoded representation.

The main innovation in Transformers is the self-attention mechanism, meaning that the model can give different parts of the input sequence a different "weight" when computing a representation of a particular element. Importantly, the attention is computed in parallel across the whole sequence, allowing the model to efficiently capture long-range dependencies. Additionally, the Transformer uses positional encoding, as the self-attention mechanism is not aware of the input elements order.

2.5.2 Applications of Transformer

Transformers have proven effective in tackling regression problems, especially in time series forecasting, although it is originally designed for sequence-to-sequence tasks. The input sequence typically consists of past observations or features, and the output is a continuous value or a predicted sequence of future values. The model's ability to process inputs in parallel and capture complex dependencies makes it suitable for financial time series forecasting, weather predictions, and sales forecasting. These tasks benefit from the Transformer's superior handling of long sequences and intricate patterns compared to traditional methods like ARIMA and even some RNN-based models.

2.5.3 Future Directions

Further innovations have led to the development of modified versions of the Transformer designed for specific tasks. The Temporal Fusion Transformer (TFT) and the Informer are examples which are designed specifically for time series and regression tasks. These variants include additional components to improve performance on tasks that involve complex temporal dynamics and require efficient handling of huge sequences.

2.6 Optimisation – Genetic Algorithm (GA)

Genetic algorithms are a class of optimisation and search algorithms inspired by Charles Darwin's process of natural selection. GAs belong to the family of evolutionary algorithms and are used for solving complex search and optimisation problems by simulating the process of biological evolution.

2.6.1 PyGAD and DEAP

PyGAD is a user-friendly Python GA module that offers a simple implementation for users. Distributed Evolutionary Algorithms in Python (DEAP) is a more comprehensive algorithm framework and more adaptable and adjustable. It supports a variety of algorithms [16]. The concept of DEAP is to use distinct data types, including populations, individuals, and fitness to represent the problem which perform operator operations through the toolbox module to efficiently execute evolutionary algorithms. Both PyGAD and DEAP reduce the threshold for using GA with their simple design and user-friendly interface, allowing more researchers to easily apply GA in their respective optimisation tasks.

2.6.2 Applications and Challenges of GA

Recently, the application of GAs has expanded into multiple domains, including machine learning, combinatorial optimisation, and multi-objective optimisation. Users deploy them to fine-tune hyperparameters in machine learning models, optimise neural network architectures, select optimal feature subsets, and often achieving remarkable outcomes [15].

GAs excel in scenarios where traditional optimisation methods falter due to complexity of the problem, a high number of variables, or non-linear constraints. The robustness and versatility of GAs allow them to deliver reliable solutions across a diverse range of problems and applications. Moreover, the integration of GAs with other optimisation algorithms to create hybrid

methods often leads to superior results. One significant limitation of GAs is premature convergence, which may lead to local rather than enabling them to discover global solutions. Also, GAs can be resource-intensive, requiring substantial computational time and power.

2.6.3 Future Directions

The advent of user-friendly libraries like PyGAD and DEAP, mark a new era in the development of genetic algorithms. These tools facilitate the practical application of GAs across a variety of fields and significantly contribute to the advancement of algorithmic research and theory. As GAs continue to evolve, they promise even greater efficiencies and breakthroughs in complex optimisation tasks.

2.7 Feature Extractions

Feature extraction plays a vital role in machine learning (ML) and statistics. It reduces the number of resources required for processing without losing important or relevant information. This is achieved by transforming the data into a lower-dimensional space, enhancing the efficiency and performance of data processing algorithms. Various feature extraction techniques including Principal Component Analysis (PCA), T-distributed stochastic neighbor embedding (t-SNE), Canonical Correlation Analysis (CCA), Principal Axis Factor (PAF), Unweighted Least Squares (ULS), Generalised Least Squares (GLS) are covered below.

2.7.1 PCA

2.7.1.1 Fundamental Concepts

PCA is a data analysis method that enables for dimension reduction for the statistical exploration of complex quantitative data. The aim of PCA is to transform highly correlated quantitative variables into new uncorrelated variables, known as principal components, while retaining as much information as possible.

PCA simplifies the complexities in high-dimensional data while preserving trends and patterns. This is achieved by by identifying a new coordinate system where the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. The axes in this new coordinate system are orthogonal, generated through linear combinations of the original variables, and ordered by the amount of variance explained by each new axis. An axis with a high explained variance signifies that it captures a large proportion of the total variance of the data and can explain more about the initial data structure [18, 29].

To determine these axes and principal components, PCA relies on mathematical principles involving the calculation of the covariance matrix to understand the relationships between the data variables and deriving the eigenvectors and eigenvalues of the matrix. The eigenvectors define the directions of the new axis, while the eigenvalues determine their magnitude. The eigenvectors represent the principal components, and the eigenvalues define their explanatory power regarding the variance in the data. It is important to standardise the data prior to applying PCA to prevent any single variable from dominating the principal component analysis due to differences in scale.

2.7.1.2 Applications and Limitations

PCA is particularly useful for reducing the dimensionality of large data sets as it facilitates easier visualisation and analysis. Reducing the number of variables, PCA can reveal the underlying structure of the datasets, enhancing the interpretation of the used variables. This process often improve the performance of machine learning algorithms. The visualisation of the data structure in two or three dimensions can be crucial for detecting patterns and making strategic decisions. Moreover, PCA helps in filtering 'noisy' data by reconstructing the original variables from the principal components [29, 20, 9].

However, PCA has its own challenges. The new variables or principal components, may be difficult to interpret at times, especially in terms of their physical meaning. This is because they are linear combinations of the original variables, which may not have a direct interpretation. Additionally, while focusing on the components with the highest variance explained can significantly clarify patterns in the data, it may also lead to the loss of information that resides in the less explanatory components, potentially overlooking subtle but important variations [29, 20, 9].

2.7.2 t-SNE

2.7.2.1 Fundamental Concepts

t-SNE is an approach for visualising and analysing complex datasets [17]. This ML algorithm is mainly used to handle non-linear and high-dimensional data. It is particularly suitable for tasks which require detailed interpretation such as image processing and language analysis. The fundamental concept of t-SNE involves using probability distribution to represent high-dimensional and low-dimensional data, and corresponding low-dimensional counterparts. This approach ensure that the proximity of similar data points is maintained during the dimensionality reduction process. t-SNE effectively preserves the local structure of the data, facilitating the identification and clustering of patterns and similarities within the dataset.

2.7.2.2 Applications and Limitations

t-SNE is widely used across different disciplines for its robust visualisation capabilities. In genomics, it helps researchers visualise complex genetic patterns and interactions. Network security professionals use t-SNE to detect unusual patterns that may signify security breaches. In cancer research, the algorithm assists in identifying patterns of gene expression that are characteristic of different cancer types. Additionally, t-SNE is employed in geological field interpretation to categorise different rock types based on multivariate data, and in biomedical signal processing, it enhances the detection of anomalies in physiological signals.

Despite being used extensively, t-SNE has several limitations. The algorithm's computational complexity can lead to long processing times when very large datasets are used. t-SNE typically reduces data to 2 or 3 dimensions only, which is useful for visualisation. It may not be sufficient for all analytical tasks. Moreover, t-SNE excels at preserving the local structure or similarity of data, it does not maintain the true global distances between points. Sometimes, this characteristic may result in misleading interpretations, particularly when precise distance metrics are crucial for analysis.

2.7.3 CCA

2.7.3.1 Fundamental Concepts

CCA is a method used to discover and interpret the relationships between two sets of multidimensional variables. The aim is to identify the maximal correlation between linear combinations of variables in two datasets, allowing a deeper understanding of how these variable sets are related to each other [36]. By calculating the canonical coefficients, CCA finds pairs of canonical variables whose linear combinations in each dataset that show the strongest interrelation. This process allows the extraction of pairs of canonical variables that highlight the most significant cross-dataset correlations.

The process of CCA involves a few critical steps. Firstly, calculation of the covariance matrices that represents the relationships within and between the two sets of variables and then, identify the eigenvectors and eigenvalues that correspond to the maximum cross-covariance which lead to the determination of canonical variables. These variables are then used to transform the data into a new representation that highlights the most significant correlations between the datasets.

2.7.3.2 Applications and Limitations

CCA synthesises information across datasets into a series of canonical variables, which are ranked according to the strength of their correlation. This provides a new perspective on the relationship between the two data sets, revealing insights that may not be apparent when analysing the datasets independently. Additionally, CCA facilitates the representation of these relationships in a lower-dimensional space, simplifying the visualisation and interpretation of complex inter-variable dynamics [11]. This makes CCA valuable in diverse fields such as genomics, where understanding relationships between different types of biological data is crucial, and psychology, for linking behavioral data with neuroimaging findings.

CCA has several limitations. The effectiveness of this method relies heavily on the size of the dataset. Using a small datasets may lead to overfitting and yield misleading correlations. Furthermore, CCA assumes linear relationships between the variables of both sets, which may not be suitable for capturing more complex, non-linear interactions. The complexity of interpreting the results, with high-dimensional data, poses challenges as canonical variables are not always intuitive and may require additional analysis to truly understand their implications.

2.7.4 PAF

2.7.4.1 Fundamental Concepts

PAF is a method to uncover hidden factors or variables that explain the observed variances among measured variables within a dataset. Different from other factor analysis techniques which assume that underlying common factors are the cause of all observed variations, PAF adopts a more nuanced approach by first estimating initial communalities. These initial communalities measure the extent of each variable's variance that can be attributed to common factors, recognising the existence of both common and unique variances [27].

This initial estimation of communalities is critical for a more precise analysis of the data structure in scenarios where the observed variables are influenced by shared underlying factors and their individual specific variances. PAF uses these first estimations to refine the exploration of

the dataset's underlying structure, extracting components that account for the shared variance after these initial communalities have been established.

2.7.4.2 Applications and Limitations

PAF is useful in scenarios where variables are influenced by shared underlying factors and their specific variances. This approach is used extensively in psychology to understand personality traits, finance to assess risk factors, and marketing research to identify consumer behavior patterns. Focusing on the shared variance across variables, PAF can help researchers and analysts to identify latent factors that play a crucial role in understanding the behaviour and characteristics of the dataset.

However, PAF has several challenges. One of the significant limitations of this approach lies in its reliance on the accurate estimation of initial communalities. Inaccurate estimations can lead to incorrect factor extraction and misinterpretation of the data structure. Additionally, PAF assumes linearity among variables, which may not be suitable to use in complex datasets where non-linear relationships exist. This method also requires a large sample size to achieve stable and reliable factor solutions.

2.7.5 ULS

2.7.5.1 Fundamental Concepts

ULS is an approach in statistical modeling that aims to optimise the fidelity of a model by minimising the sum of squared discrepancies between observed and predicted correlation matrices [7]. This is characterised by its "unweighted" strategy, ULS treats all differences uniformly across the correlation matrix, without assigning varying weights to discrepancies. This approach is implemented through an iterative computational process that incrementally refines the model parameters to enhance the accuracy of correlation predictions.

Through progressively aligning the predicted correlation matrix closer to the observed matrix, ULS helps reveal the latent structure underlying a dataset and facilitates a deeper understanding of the complex interrelations among variables.

2.7.5.2 Applications and Limitations

ULS excels in scenarios where data distribution assumptions are not strictly adhered to. The ability to provide reliable estimates without weight corrections or strict assumptions makes it a robust choice for exploratory data analysis and structural equation modeling. Its ability to handle discrepancies uniformly across the correlation matrix without the need for weight adjustments allows it to provide reliable estimates even when traditional assumptions are violated. ULS stands as a valuable tool in fields such as psychology, sociology, and market research, where understanding the underlying structures of complex datasets is crucial.

However, the simplicity of ULS can also be a limitation. The method does not account for the potential heteroscedasticity or the differing variances across the range of data can lead to inefficiencies and biased estimates in some cases. Additionally, since it treats all discrepancies equally, it may not be as effective when different parts of the data inherently carry different levels of importance or reliability. While ULS is advantageous for its direct application and fewer

assumptions, it may not always provide the most accurate or appropriate model fit compared to other methods that can weight discrepancies differently.

2.7.6 GLS

2.7.6.1 Fundamental Concept

GLS is a method that extends the capabilities of Ordinary Least Squares (OLS) to handle more complex models. It is particularly effective when dealing with heterogeneity of variance (heteroscedasticity) and correlations among error terms in regression models situations where the OLS assumption of constant variance of error terms (homoscedasticity) fails. GLS modifies the OLS approach by incorporating a specific transformation that weights observations differently, depending on their variability and interrelationships. This transformation involves using a covariance matrix that captures the variances and covariance of the error terms, allowing GLS to adjust for unequal reliability and complex dependencies between observations [25].

2.7.6.2 Applications and Limitations

GLS is useful in fields such as econometrics, finance, and environmental modeling, where the data often exhibit complex error structures that standard OLS cannot adequately handle. Given the limitations associated with unequal variances and correlation errors, GLS provides a robust framework for performing regression analysis and ensures that the resulting models are both statistically robust and highly representative of the underlying data structure. In econometrics, for example, GLS is used to estimate models where there might be a natural clustering of variance (e.g., over time in panel data). In finance, GLS helps in analysing time series data where the volatility of financial instruments can change over time, influencing the model's accuracy.

Despite its strengths, GLS has several limitations. The effectiveness of GLS heavily dependent on the correct specification of the covariance matrix. Inaccurately estimating this matrix can lead to inefficiencies and even biases in the model estimates, compromising the validity of the regression analysis. Additionally, GLS can be computationally intensive, especially as the size and complexity of the data set increase, which might limit its applicability in large-scale studies or real-time analysis scenarios.

Chapter 3 – Methodology

3.1 ANSYS (RANS) approach

To propose new improvements on the RANS closure problem, we need to identify where the actual models lack precision and reliability. This section compiles all the work led to achieve results, conclusion on the actual performances of models in CFD and provide guidelines for the machine learning aspects.

3.1.1 Methodology

The ANSYS approach will be decomposed in 5 main steps to obtain actual comparable and exploitable results. Each of these steps will be explained and detailed in this report.

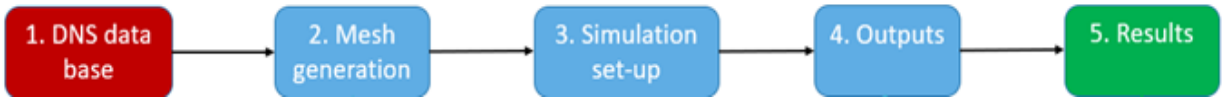


Figure 3.1: ANSYS Methodology Pipeline

- **DNS data base** – Understand the previous studies related to the RANS closure problem topic.
- **Mesh generation** - Preparation of the meshes for the simulations.
- **Simulation set-up** - Preparing simulations for the selected cases.
- **Outputs** - Discussion and election of the outputs for model comparison.
- **Results** - Results discussion.

This section aims to recreate cases from previous studies and set them up using the desired turbulence models. By replicating these cases, we seek to identify and highlight potential weaknesses or limitations in the computed results. This process allows us to assess the performance and accuracy of the chosen turbulence models under specific flow conditions and geometries. Through careful comparison with experimental data or benchmark solutions, we can evaluate the predictive capabilities of the CFD simulations and identify areas where improvements may be necessary. By systematically analysing the discrepancies between computed and observed results, we gain insights into the strengths and weaknesses of the selected turbulence models. This comparative analysis contributes to help the ML aspect of the project in their objective to propose a new model that will improve those computed on the same cases.

3.1.2 DNS Database

In 2015, the Cambridge university press published an article in their "fluid mechanics" journals named: "Direct Numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. This journal aims to analyse the performance of diverse models on a large-scale simulation. From these

simulations were published the results and data from their experiments. To lead our analysis, we will take their geometry and results and reproduce them for a lower scale scenario.

For the simulations cases, we will be following the model channel flow case with five different $Re\tau$ number being: 180, 550, 1000, 2000 and 5200. For computational cost resources reasons, we will reproduce the paper's geometry in a lower-scale and extract the same amount of data points as their data base. As for the model, we will run two models: $k-\omega$ and RSM. The results from these simulations will also be used by the different machine learning models to train them.

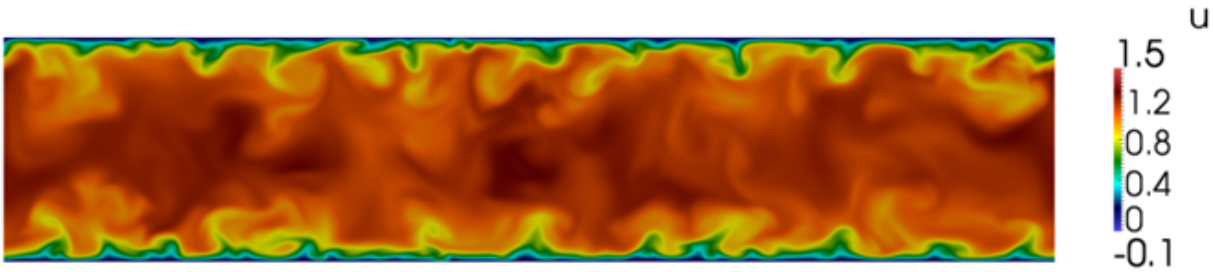


Figure 3.2: Example of Simulation from Published Paper

3.1.3 Mesh Generation

The first step of the ANSYS process is the mesh generation. Meshing is a critical step in the CFD simulation process, involving the division of the computational domain into smaller elements or cells. The purpose of meshing is to discretise the physical space, allowing for the application of numerical methods to solve the governing equations. Throughout this step, we can define two types of meshes being the coarse and the refine mesh.

Coarse meshes, composed of larger elements, offer computational efficiency and quick assessments of overall flow behaviour, making them suitable for preliminary studies and large-scale simulations. However, they sacrifice detail and accuracy, limiting their application in scenarios requiring precise representation of small-scale flow phenomena or complex geometries. Fine meshes, with their high density of smaller elements, excel in providing highly accurate simulations. They capture intricate flow details, including boundary layers and vortices. On the other hand, the computational cost is a drawback, demanding substantial processing power and potentially longer simulation times. Careful consideration of computational resources and validation is crucial when opting for a fine mesh.

For our cases, we will use a combination of both types, the idea is to refine the mesh on the near-wall zones to capture more detailed flow component and turbulences. As the mesh progresses in the geometry, it will become coarser to reduce computational cost. To create our geometry and meshes, we proceed as following.

First step is to create the geometry. The goal here is to recreate geometry from the DNS data base in a lower scale to reduce computational cost. To do so, we will only generate a rectangular geometry with a small volume. Using ICEM from the ANSYS fluent package and the following coordinates, we end up with this geometry:

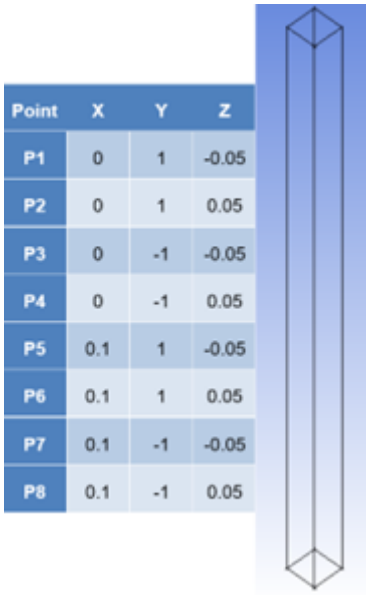


Figure 3.3: Mesh Geometry Generated

Now that we obtained the points, we can create lines from which we create surfaces. We will name the surfaces as following as for the future steps in Fluent:

Surface Name	Position
INLET	$x = 0$
OUTLET	$x = 0.1$
BOTTOM WALL	$y = -1$
TOP WALL	$y = 1$
SIDE Z0	$z = -0.05$
SIDE Z1	$z = 0.05$

Table 3.1: Surface Name Attribution

When generating meshes for fluid flow simulations, we need to pay closer look at the near wall conditions. When meshing the geometry, we need to calculate the space between where the mesh starts and where the physical boundary is located. To do so, we calculate the first mesh-geometry space (Δy_1) with the following formula:

$$Y^+ = \frac{u_\tau . \Delta y_1 . \rho}{\nu} \longleftrightarrow \Delta y_1 = \frac{\nu . \rho}{u_\tau . Y^+} = \frac{\nu}{u_\tau} \tag{3.1}$$

Since, the density of the computed fluid (ρ) is 1 and we assume Y^+ to be one, the formula becomes the kinematic viscosity (ν) over the friction velocity (u_τ). Since both kinematic viscosity and friction velocity are changing for each Re_τ we calculate Δy_1 for each one of the cases.

Re_τ	U_τ	ν	ρ	Y^+	Δy_1
180	0.06373090	0.00035	1	1	0.005491841477211210
550	0.05434960	0.00010	1	1	0.001839939944360220
1000	0.05002560	0.00005	1	1	0.000999488262009851
2000	0.04587940	0.000023	1	1	0.000501314315357219
5200	0.04148720	0.000008	1	1	0.000192830559787115

Table 3.2: First Element Space Table

After doing so we need to enter the number of required elements along the Y axis. To determine this number, we will simply take the number amount of element along the same axis in the DNS data base. Inside ICEM we set the parameters as following:

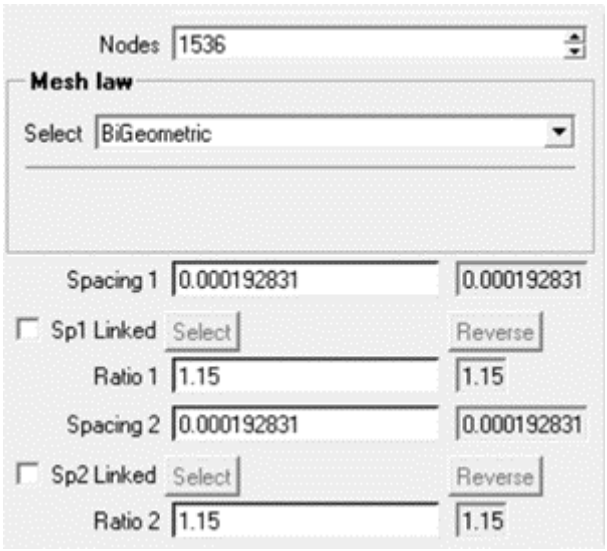


Figure 3.4: ICEM space set up

3.1.4 Simulation Set-up

The third step of our methodology consists of setting up our simulation. This process needs to automatised since we will need to run multiple simulations. To do so, we will use both the graphical user interface (GUI) of fluent along with the text user interface (TUI). Both of these UI will be set differently so we will view the process on both.

3.1.4.1 GUI

The GUI part is fairly simple. After creating folders and case file for both models on each Re_τ number. We need to include the corresponding mesh within the *.cas.h*5 file. After doing so, we need to create a line on the mesh to extract data for each simulation. To do so, we click on $results \longrightarrow create \longrightarrow line$. Then we enter these values (see figure 3.5) and call the line "line":

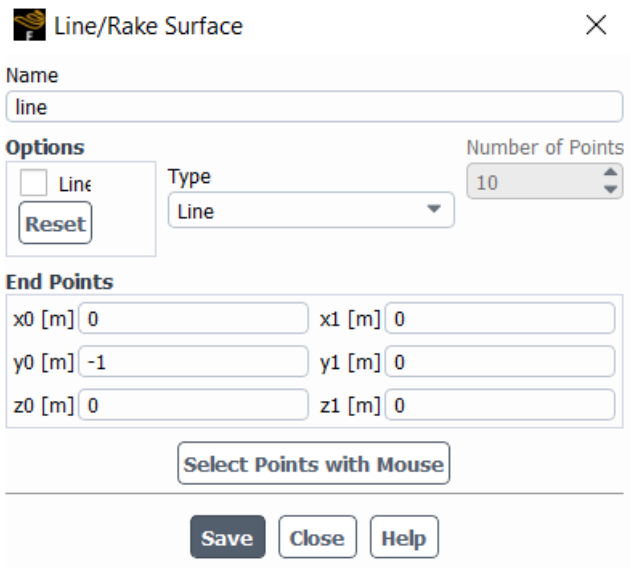


Figure 3.5: Line set-up

Since the mesh an the boundary conditions are symmetric, we only create a line that goes halfway across the mesh as we can mirror the data to obtain the other half. The rest of the GUI will be left as default. After this step we save the case file and by adding the ”to read” mention inside its name. After its done we can move on to the TUI.

3.1.4.2 TUI

The TUI is the most important part of the set-up. To set-up our models and simulations we will use the same TUI model with little differences for each model. To explain this part, we will detail the TUI file on both models.

```
file/set-tui-version "23.2"

/file/read-case Re_5200(to_read).cas.h5

/define/materials/change-create air air yes constant 1 no no yes
    constant 8.00000e-06 no no no

/mesh/modify-zones/make-periodic 14 15 no yes yes
/mesh/modify-zones/make-periodic 16 17 no yes yes

/define/boundary-conditions/fluid 10 no yes 0 1 yes 0.00172 0 0 0 0 no
    no no 0 no 0 no 0 no 0 no 0 no 1 no no no no no

/solve/monitors/residual/convergence-criteria 1e-6 1e-6 1e-6 1e-6 1e-6
    1e-6

/solve/set/discretization-scheme mom 0
/solve/set/discretization-scheme k 0
/solve/set/discretization-scheme omega 0

/solve/initialize/set-defaults x-velocity 1
/solve/initialize/initialize-flow yes
/solve/iterate 100
```

```
21
22 /solve/set/discretization-scheme mom 1
23 /solve/set/discretization-scheme k 1
24 /solve/set/discretization-scheme omega 1
25
26 /solve/initialize/set-defaults x-velocity 1
27 /solve/iterate 10000
28
29 /plot/plot yes U no no no x-velocity no no y-coordinate line ()
30 /plot/plot yes Dudy no no no dx-velocity-dy no no y-coordinate line ()
31 /plot/plot yes P no no no pressure no no y-coordinate line ()
32 /plot/plot yes y+ no no no y-plus no no y-coordinate line ()
33 /plot/plot yes k no no no turb-kinetic-energy no no y-coordinate line()
```

We will now detail and explain each of the commands used for this script. Please note that the script is the journal file for $Re_\tau \approx 5200$ with the k-ω model.

- **line 1 -** Sets the used TUI version.
- **line 3 -** Reads the case file within the folder.
- **line 5 -** Create and the materials with following properties: Density = $1kg/m^3$, Viscosity = $8e-06kg/ms$.
- **line 7 and 8 -** Transforms the zones with ID 14, 15 and 16, 17 (Top and bottom wall) into periodic boundary conditions.
- **line 10 -** Define the inner boundary condition called ”fluid”. Sets the material as air, defines 1 sources term as x-momentum sources with 0.00172 value. Finally defines the Rotary Z axis direction as 1.
- **line 12 -** Sets the convergence criteria to $1e-6$.
- **line 14, 15, 16 -** Sets the discretisation scheme of momentum, turbulence and dissipation rate as first order upwind.
- **line 18 -** Sets the default value of x-velocity as $1m/s$.
- **line 19 -** Initialises the simulation.
- **line 20 -** Solves the simulation for 100 iterations.
- **line 22,23,24 -** Sets the discretisation scheme of momentum, turbulence and dissipation rate as second order upwind.
- **line 26 -** Resets the default value of x-velocity as $1m/s$.
- **line 27 -** Solves the simulation until 10000 iterations.
- **line 29,30,31,32,33 -** Creates data files for x-velocity, $\frac{DU}{Dy}$ Velocity, pressure, y^+ and turbulence intensity k. Each of these files are created with the data along the line created in the GUI (see Figure 3.5). We will now review the differences, between this script and the RSM model script.

```
1  /define/models/viscous reynolds-stress-model yes
2
3  /define/boundary-conditions/fluid fluid no yes 0 1 yes 0.00172 0 0 0 0
       0 0 0 0 0 0 no no no 0 no 0 no 0 no 0 no 0 no 1 no yes no no no
4
5  /solve/monitors/residual/convergence-criteria 1e-6 1e-6 1e-6 1e-6 1e-6
       1e-6 0.001 0.001 0.001 0.001 0.001 0.001
6
7  /plot/plot yes uu no no no uu-reynolds-stress no no y-coordinate line()
8  /plot/plot yes vv no no no vv-reynolds-stress no no y-coordinate line()
9  /plot/plot yes ww no no no ww-reynolds-stress no no y-coordinate line()
10 /plot/plot yes uv no no no uv-reynolds-stress no no y-coordinate line()
11 /plot/plot yes uw no no no uw-reynolds-stress no no y-coordinate line()
12 /plot/plot yes vw no no no vw-reynolds-stress no no y-coordinate line()
```

- **line 1 -** Define the model as RSM.
- **line 3 -** Define the inner boundary condition fluid.
- **line 5 -** Set the Convergence criteria and the residuals criteria as 0.001.
- **line 7,8,9,10,10,11,12 -** Create data files for all RST components along the line.

The rest of the file is the same for both models. After completing the scripts, we need to read the TUI. From here the TUI will read itself and compute the simulation through fluent automatically. Afer doing so we can look at the outputs.

3.1.5 Outputs

After computing simulations on both models, we need choose which outputs we will compare between the DNS and RANS models.

Comparison of k-ω and RSM includes examination of flow rates and covariances. Flow velocity is critical to understanding fluid behavior, as it reflects how a fluid moves in the direction of flow. Covariance represents turbulent momentum exchange and can provide insight into the structure of turbulence. By analysing these parameters, the accuracy of the model can be evaluated. The k-ω model focuses on turbulent quantities such as kinetic energy and dissipation rate, predicting covariances indirectly through approximations such as the Boussinesq assumption.

In contrast, the RSM directly computes the RST components, providing a more detailed representation of turbulence. Comparing both models with experimental or high-precision simulation data illustrates their strengths and limitations.

3.2 PINNs

A PINNs model was designed and trained to predict the RST τ of a plate in a turbulent channel flow.

3.2.1 Data Exploration

The training dataset was generated using the DNS simulations of Oden database. These simulations are DNS of incompressible turbulent flow between two parallel planes. Periodic boundary conditions are applied in the streamwise (x) and spanwise (z) directions, and no-slip/no-penetration boundary conditions are applied at the wall. The computational domain sizes are $L_x = 8\pi\delta$ and $L_z = 3\pi\delta$, where δ is the channel half-width, so the domain size in the wall-normal (y) direction is 2δ. The flow is driven by a uniform pressure gradient, which varies in time to ensure that the mass flux through the channel remains constant.

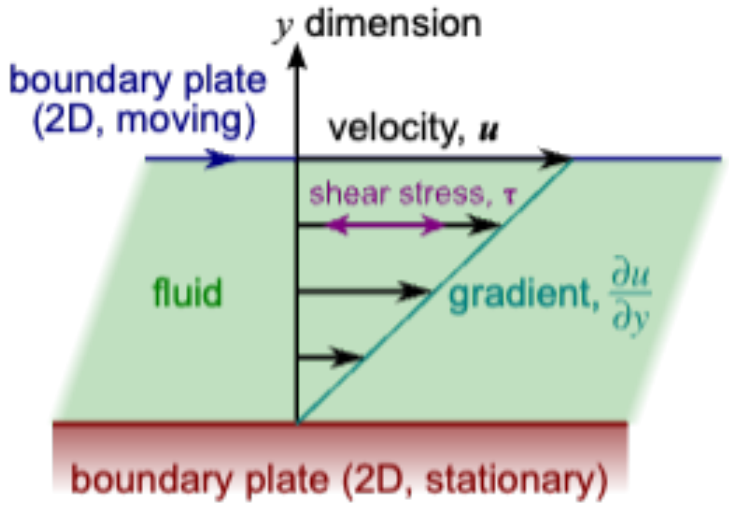


Figure 3.6: Turbulent Channel Flow

In channel flow, the primary motion of the fluid is in the streamwise (x) direction due to the pressure gradient applied in this direction. This makes $\overline{U}$, the mean streamwise velocity, the most significant component for characterising the flow. $\overline{V}$ (wall-normal velocity) and $\overline{W}$ (spanwise velocity) are usually much smaller by comparison and often average out to zero when considering fully developed flow, especially in simulations where the flow is assumed to be statistically steady and homogeneous in the streamwise and spanwise directions.

Turbulent channel flows are generally statistically homogeneous in the streamwise (x) and spanwise (z) directions, meaning that any statistical property of the flow (when averaged over time) does not vary along these directions. This homogeneity implies that the derivatives $\frac{\mathrm{d}\overline{U}}{\mathrm{d}x}$, $\frac{\mathrm{d}\overline{V}}{\mathrm{d}x}$, $\frac{\mathrm{d}\overline{W}}{\mathrm{d}x}$, and $\frac{\mathrm{d}\overline{P}}{\mathrm{d}x}$ tend to zero.

3.2.2 Targets & Features Selection

3.2.2.1 Targets Selection

The targets or outputs of the PINN model are designed to capture key characteristics of the turbulent flow, including:

- **Streamwise velocity variance**, $u'u'$, indicating the intensity of velocity fluctuations in the direction of the flow.
- **Wall-normal velocity variance**, $v'v'$, highlighting the fluctuations perpendicular to the wall.
- **Spanwise velocity variance**, $w'w'$, reflecting the fluctuations across the width of the channel or pipe.
- **Streamwise-wall-normal velocity covariance**, $u'v'$, shedding light on the momentum transfer in the flow.
- **Mean streamwise velocity**, U, providing the average velocity profile across the flow.
- **Mean streamwise velocity gradient**, $\partial U/\partial y$, which is essential for understanding shear rates within the flow.
- **Mean pressure**, P, offering insights into the force distribution across the flow field.
- **Turbulent kinetic energy**, k, which is a measure of the energy contained within the turbulent eddies.

3.2.2.2 Features Selection

The input features for turbulence modeling through PINNs are chosen to provide comprehensive insight into the flow's properties and behavior. These features include:

- **Kinematic viscosity**, ν, which is a fundamental property of the fluid that affects its flow characteristics.
- **Friction velocity**, u_τ, that scales the velocity near the wall, indicative of the wall shear stress.
- **Reynolds number based on friction velocity**, Re_τ, which provides a non-dimensional parameter describing the flow's regime, highlighting whether the flow is laminar or turbulent.
- **Wall-normal grid point in wall units**, y^+, a dimensionless number that scales the distance from the wall, crucial for analysing boundary layer flows.
- **Wall-normal grid point normalised by channel half-width**, y/δ, indicating the relative position within the flow domain.

3.2.2.3 Justification for Feature Redundancy

A cursory examination of the features selected for turbulence modeling using PINNs might suggest a redundancy in the flow characteristics represented. However, this apparent redundancy is by design and plays an essential role in representing the complex and multifaceted nature of turbulent flows. Here is a justification for these redundancies:

1. **Comprehensive Flow Characterisation:** The features encompass different aspects of the turbulent flow, offering a comprehensive characterisation. For instance, both the friction velocity, u_τ, and the Reynolds number based on friction velocity, Re_τ, may seem to provide overlapping information regarding the flow regime. Yet, u_τ offers a direct scaling for near-wall velocity measurements, while Re_τ provides a dimensionless parameter that universally characterises the flow's regime—capturing both the specific and general properties of turbulence.

2. **Capturing Flow Scales:** Turbulence is inherently multiscale, with important dynamics occurring at both small and large scales within the flow. The inclusion of features like wall-normal grid points in different units (y^+ and y/δ) ensures that the model can accurately account for phenomena at these varying scales. Specifically, y^+ is crucial for modeling near-wall behaviors—where the law of the wall prevails—whereas y/δ provides insight into the flow's behavior in the core region, away from the wall's influence.

3. **Enhancing Model Robustness and Accuracy:** Feature redundancy also enhances the robustness and accuracy of PINNs in modeling turbulent flows. By incorporating features that might seem redundant, the model can leverage different but related perspectives of the flow, which aids in training and reduces the likelihood of overfitting to specific flow configurations. Moreover, it allows for better generalisation across different flow conditions and configurations, ensuring the model's applicability to a wide range of problems in fluid dynamics.

These selected features and targets encompass both the inputs required by the PINN to understand the fluid flow's physics and the outputs necessary for a comprehensive turbulence model. Their selection is rooted in the understanding that an effective turbulence model must not only predict the flow patterns accurately but also adhere to the underlying physical principles that govern fluid dynamics.

3.2.3 PINNs Model Workflow

The PINNs model trained to model turbulence using the RANS equations runs as follows:

1. **Inputs (Features):** Encapsulate the physical parameters and non-dimensional numbers that characterise the turbulent flow regime, boundary layer properties and the influence of viscosity. Introducing these data into the model enables PINN to take into account the fundamental aspects of fluid dynamics that govern turbulence phenomena in the computational domain.

2. **Neural Network:** To model turbulence which is inherently a complex and highly nonlinear phenomenon, the neural network architecture within the PINN needs to be sufficiently expressive to capture the intricate relationships between the input features and the targets. This necessitates an architecture that can handle the variability and non-linearity

of turbulence, likely requiring multiple hidden layers (deep learning architecture) and advanced activation functions to facilitate the learning of subtle patterns in the flow characteristics from the provided features.

3. **Outputs (Targets):** The PINN targets for turbulence modeling are designed to represent a comprehensive set of variables that describe the state and energy characteristics of turbulent flow, providing insight into the flow dynamics, energy distribution and shear forces involved. The neural network therefore aims to predict these quantities in the flow domain, based on the physical context provided by the input characteristics.

4. **Automatic Differentiation:** Automatic differentiation plays a crucial role in enabling PINN to incorporate the fluid dynamics equations directly into the learning process. Thanks to automatic differentiation, the derivatives of the predicted targets with respect to the inputs can be calculated efficiently, which is essential for applying the Navier-Stokes equations that govern fluid motion. This capability allows the model to ensure that its predictions not only fit the available data (if any), but are also consistent with the fundamental laws of physics that dictate the behaviour of turbulence.

5. **Governing Equations:** For turbulence modeling, the governing equations include the RANS equations, which are integral to describing motion in fluid dynamics. These equations, when incorporated into the PINN through the loss function, require that predictions for mean velocities, pressure distributions, and turbulence kinetic energy (among others) adhere to the conservation of momentum and mass principles. This integration grounds the model's learning process in the physics of turbulence, ensuring that the PINN's predictions are physically plausible across various flow conditions and geometries.

6. **Loss Function:** The loss function in the context of turbulence modeling with PINNs is a composite that includes terms for data fidelity (minimising the difference between PINN predictions and all available experimental or simulation data for the targets) and physics fidelity (ensuring compliance with the governing RANS equations). The balance between these terms is essential, as it allows the model to exploit the available data as far as possible while respecting the constraints imposed by the laws of physics. This dual objective ensures that the model learns to generalise well, making physically consistent predictions even in regions of the flow where direct data may be sparse or unavailable.

7. **Optimizer:** The role of the optimizer is to adjust the neural network parameters to minimise the composite loss, by iteratively improving the model predictions regarding both the available data and the physical constraints. Given the complexity of turbulence phenomena and the high dimensionality of the problem space, advanced optimisation algorithms that efficiently navigate the loss landscape (such as Adam or SGD) are crucial to achieving convergence towards a physically meaningful solution.

By combining advanced neural network architectures with automatic differentiation and deep integration of physical laws, PINNs offer a powerful approach to turbulence modeling. This workflow not only accurately predicts the main characteristics of the flow, but also ensures that these predictions are consistent with the fundamental principles of fluid dynamics, even in the face of the complex and chaotic behaviour of turbulent flows. This workflow is illustrated by the Figure 3.7.

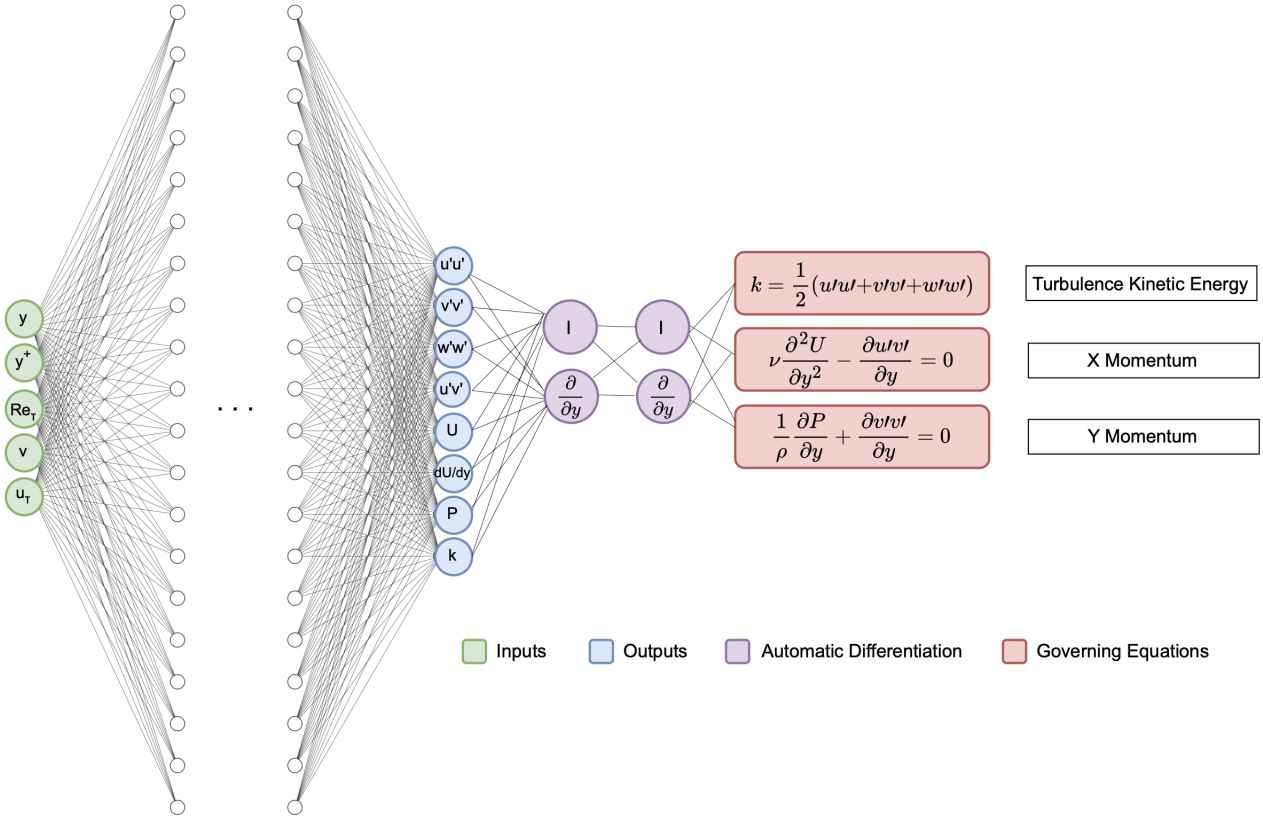


Figure 3.7: PINNs Model Workflow Diagram

3.2.4 Loss Function Design

The loss function for the PINNs model consists of two main components: the physics-informed loss and the data loss.

3.2.4.1 Physic Loss Function

The physics-informed loss is computed based on the deviation from the Navier-Stokes equations. It includes two terms:

1. *Momentum loss*:

$$\mathscr{L}_{\text{momentum}} = \mathscr{L}_{\text{x-momentum}} + \mathscr{L}_{\text{y-momentum}}$$
$$\mathscr{L}_{\text{x-momentum}} = \left\| \nu \frac{\partial^2 U}{\partial y^2} - \frac{\partial u'v'}{\partial y} \right\|_2$$
$$\mathscr{L}_{\text{y-momentum}} = \left\| \frac{1}{\rho}\frac{\partial P}{\partial y} + \frac{\partial v'v'}{\partial y} \right\|_2$$

2. *Turbulence kinetic energy (TKE) consistency loss*:

$$\mathscr{L}_{\text{TKE}} = \left\| k_{\text{pred}} - \frac{1}{2}(u'u'_{\text{pred}} + v'v'_{\text{pred}} + w'w'_{\text{pred}}) \right\|_2$$

The overall physics-informed loss is a weighted sum of these two terms:

$$\mathscr{L}_{\text{physics}} = w_{\text{momentum}}\mathscr{L}_{\text{momentum}} + w_{\text{TKE}}\mathscr{L}_{\text{TKE}}$$

Data Loss

The data loss is computed as the MSE between the model's predictions and the target data. It includes the following terms:

1. *Velocity magnitude loss*:

$$\mathscr{L}_{\text{U}} = \|U_{\text{pred}} - U_{\text{data}}\|_2$$

2. *Velocity gradient loss*:

$$\mathscr{L}_{\partial U/\partial y} = \left\| \frac{\partial U}{\partial y}_{\text{pred}} - \frac{\partial U}{\partial y}_{\text{data}} \right\|_2$$

3. *Pressure loss*:

$$\mathscr{L}_{P} = \|P_{\text{pred}} - P_{\text{data}}\|_2$$

4. *Turbulence kinetic energy loss*:

$$\mathscr{L}_{k} = \|k_{\text{pred}} - k_{\text{data}}\|_2$$

5. *Reynolds stress loss*:

$$\mathscr{L}_{\text{stress}} = \left\| \begin{bmatrix} uu_{\text{pred}} \\ v'v'_{\text{pred}} \\ w'w'_{\text{pred}} \\ u'v'_{\text{pred}} \end{bmatrix} - \begin{bmatrix} u'u'_{\text{data}} \\ v'v'_{\text{data}} \\ w'w'_{\text{data}} \\ u'v'_{\text{data}} \end{bmatrix} \right\|_2$$

The overall data loss is a weighted sum of these terms:

$$\mathscr{L}_{\text{data}} = w_{\text{U}}\mathscr{L}_{\text{U}} + w_{\partial U/\partial y}\mathscr{L}_{\partial U/\partial y} + w_{P}\mathscr{L}_{P} + w_{k}\mathscr{L}_{k} + w_{\text{stress}}\mathscr{L}_{\text{stress}}$$

The total loss function for the PINNs model is the sum of the physics-informed loss and the data loss:

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{physics}} + \mathscr{L}_{\text{data}}$$

3.2.5 Model Architecture

The architecture of a PINN is shaped by the need to balance several competing considerations, such as computational resources, model expressiveness and the specific characteristics of the physical problem in question. To address these considerations, a methodical approach to the design of the model architecture is adopted, underpinned by both theoretical insights and empirical results from extensive experiments.

3.2.5.1 Layers & Nodes Configuration

The choice of the optimal depth (number of layers) and width (number of neurons per layer) of the network is fundamental. The architecture must be complex enough to model the complex patterns and relationships intrinsic to the physical laws and data sets involved, but not so complex that it becomes intractable or subject to overfitting. A shallow but extensive network architecture is often preferred for PINNs for several reasons. Firstly, it mitigates the risk of gradients disappearing or exploding, a common challenge in deep neural network learning, particularly critical when differentiating across the network to apply the laws of physics. Secondly, wide networks are able to capture a broad spectrum of features or interactions in the input space with fewer parametric layers, which matches the continuous nature of many physical problems [4].

3.2.5.2 Activation Functions

The choice of activation function influences the network's ability to model complex, nonlinear phenomena and its differentiability—a key requirement in PINNs, where the model's output and its derivatives must adhere to specified differential equations. Activation functions with smooth, non-saturating gradients are favored to facilitate efficient learning and accurate physical representation.

3.2.5.3 Optimizer and Learning Rate Scheduler Configuration

The learning rate and optimisation strategy must be carefully calibrated to balance rapid convergence and stability of the learning process. For this model, a two-tier strategy incorporating a learning rate optimiser and a learning rate scheduler is used to improve the efficiency of the training process. This approach allows adaptive adjustments of the learning rate as the training progresses, providing optimised convergence properties.

- **Optimizer Selection**: The Adam optimizer is selected for its adaptive learning rate capabilities, which fine-tune the learning process by adjusting the learning rate for each parameter. Adam combines the advantages of two other extensions of stochastic gradient descent: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp).
- **Learning Rate Scheduler**: To further refine the training dynamics, we integrate a learning rate scheduler that adjusts the learning rate based on the model's performance on the validation set. The *ReduceLROnPlateau* scheduler is employed, decreasing the learning rate when a metric has stopped improving, aiming to find fine-grained optima by slow convergence when plateauing. This scheduler reduces the learning rate by a factor of 0.9 after a *patience* of 100 epochs without improvement in the monitored metric, specifically the validation Mean Squared Error (`val_mse`), enhancing the ability to converge to the optimal solution with refined learning rate adjustments.

The configuration ensures that the learning rate is automatically adjusted at an epoch-level frequency based on the validation set performance, specifically monitoring the MSE. This adaptive approach to learning rate management is designed to foster a more efficient and effective training process, particularly in the complex and nuanced task of physics-informed neural network optimisation.

3.2.5.4 Balancing Loss Terms

PINNs typically comprise multiple loss components, including terms that enforce adherence to the governing physical laws (physical loss) and, when applicable, terms that measure the discrepancy between the network's predictions and observed data (data loss). Properly balancing these components is essential for ensuring that the network does not disproportionately prioritise one type of loss over the other, potentially at the expense of the model's overall accuracy and generalisability.

3.2.5.5 Hyperparameter Tuning

The selection of hyperparameters, including activation functions, learning rate, and the balance between physical loss and data loss weights, is a delicate process that significantly impacts the performance of PINNs. We employed a grid search strategy to fine-tune the following hyperparameters:

- **Activation Functions:** Exponential Linear Unit (ELU), Scaled Exponential Linear Unit (SELU), and Hyperbolic Tangent (TANH) were considered, given their unique properties beneficial for maintaining smooth gradients essential for PINNs.
- **Learning Rates (LR):** A range of learning rates, 1×10^{-3}, 1×10^{-4}, 5×10^{-4}, were tested to find the optimal speed at which the model converges without overlooking significant features of the solution space.
- **Neural Network Size:** Both the number of layers (2, 3, 4, 8) and the number of nodes per layer (32, 64, 128, 256) were varied to ascertain the architecture's impact on the model's performance.
- **Weight of the Physical Loss and Data Loss:** The trade-off between physical law compliance (with weights 1, 10, 100) and fitting to the data (weights 1, 0.1, 0.01) was calibrated to ensure the model not only learns the underlying physics but also adheres closely to the empirical data.

3.2.6 Model Implementation

The source code for the PINNs model is available in the following GitHub repository: PINNs Repository.

3.2.6.1 Framework (PyTorch Lightning)

The PINNs model was developed using the PyTorch Lightning Framework. This is an AI framework that builds on PyTorch to provide a more organised code structure and reduce the amount of standard code required. It clearly separates model learning logic and automation code, making code easier to develop, debug and reuse. PyTorch Lightning supports advanced features such as multi-GPU, TPU, distributed training and mixed precision, making the training process more efficient and scalable.

3.2.6.2 Core Components

Here are the different elements that make up the PINNs model:

- **NNModel:** This component represents the backbone of the PINN. It is a customisable neural network architecture that can be tailored according to the specific requirements of the turbulence modeling task at hand. Parameters such as the dimensions of the input and output layers, the size and number of hidden layers, and the types of activation functions can be adjusted. This flexibility ensures that the network is capable of capturing the complex dynamics of turbulent flows, facilitating accurate predictions of the various turbulence characteristics.

- **TurbulenceDataset:** Specialised class that handles the loading, pre-processing and formatting of turbulence data. It structures the data in a way that is compatible with machine learning models, encompassing features such as spatial coordinates, Reynolds number and other relevant inputs, as well as target outputs such as velocity components and kinetic energy. This dataset provides the basic data needed to train and evaluate the PINN model, ensuring that the network has access to high-quality information that reflects the physical system it aims to model.

- **TurbulenceDataModule:** This component acts as a data management hub within the PyTorch Lightning framework, orchestrating the flow of data through different phases of the model lifecycle, including training, validation, and testing. The TurbulenceDataModule leverages the TurbulenceDataset to efficiently batch and shuffle data. It ensures that data is fed into the model in a structured and consistent manner, optimising the learning process and aiding in the model's convergence towards accurate predictions.

- **TurbulenceModelPINN:** Core component that integrates the neural network model (NNModel) with physics-informed learning. It goes beyond conventional neural network applications by incorporating the physics of turbulence directly into the learning process, using a physics-informed loss function. This function penalises the model not only for deviations from the training data, but also for violations of the physical laws governing it, such as the Navier-Stokes equations. It thus guides the PINN model to develop solutions that are both accurate and physically plausible. The TurbulencePINN model therefore represents the culmination of the physics-informed neural network approach, integrating both data-based knowledge and the rigour of physical laws into the model's predictions.

3.2.7 Training Pipeline

3.2.7.1 Data Sets Preparation

To effectively train, validate and test model performance, the dataset is carefully segmented as follows:

- **Training Set: 68%** - This majority portion is used for the basic training process, allowing the model to learn and adapt to the patterns, relationships and dynamics present in the turbulence data.

- **Validation Set: 12%** - This segment serves as a checkpoint for model performance on novel data, guiding hyperparameter tuning and model adjustments without compromising the integrity of the test set.

- **Test Set: 20%** - Reserved as a final evaluation reference, this set provides an unbiased assessment of the model's predictive capabilities and its generalisation to new, unseen data.

3.2.7.2 Training Cycles

The model undergoes several consecutive training cycles, each designed to progressively refine its predictive accuracy and adherence to physical principles. Importantly, each cycle following the first begins with the state of the model at the optimal snapshot obtained in the previous cycle, facilitating cumulative learning and refinement. Indeed, a callback is set to monitor the validation MSE and save the best model accordingly. This ensures that the model state corresponding to the best validation performance is always saved throughout the training process.

- **Training Cycle 1:**
  - **Learning Rate (LR):** Initialised at $1e^{-3}$, to encourage rapid convergence towards a promising region in the solution space.
  - **Physics/Train Loss Balance:** Set at a balanced ratio of 1/1 to equally prioritise fitting to the training data while adhering to the physical laws.
  - This cycle aims to establish a solid foundational model that demonstrates both high fidelity to the training data and respect for the incorporated physical constraints.
- **Training Cycle 2:**
  - **Learning Rate (LR):** Reduced to $1e^{-4}$, allowing for more precise adjustments and facilitating convergence to more accurate solutions.
  - **Physics/Train Loss Balance:** Maintained at 1/1, continuing to stress the equal importance of learning from empirical data and respecting physical constraints.
  - The model is initialised from the best snapshot of the preceding cycle, ensuring refinement upon a robustly established base, enhancing both accuracy and physical realism.
- **Training Cycle 3:**
  - **Learning Rate (LR):** Continues at $1e^{-4}$, supporting meticulous model tuning.
  - **Physics/Train Loss Balance:** Adjusted to 10/1, significantly increasing emphasis on adhering to physical laws, crucial for accurate turbulence modeling.
  - Starting from the optimum state of the previous cycle enables focused improvement towards physically plausible solutions that remain faithful to the training data.
- **Training Cycle 4:**
  - **Learning Rate (LR):** Remains at $1e^{-4}$, consistent with the strategy for precision optimisation.
  - **Physics/Train Loss Balance:** Significantly increased to 100/1, to firmly anchor physical realism, giving priority to physical accuracy.
  - The cycle commences from the concluding optimal state of the last cycle, allowing the model to fine-tune further, ensuring maximal conformance to physical laws alongside learned empirical patterns.

This structured, progressive learning approach ensures continuous evolution and refinement of the model, meticulously balancing empirical accuracy with physical fidelity, achieving a robust and finely-tuned model excellently suited for complex turbulence prediction tasks.

3.2.7.3 Reproducibility of the Training

Ensuring the reproducibility of the PINN training is paramount to asserting the validity of the results and facilitating independent verification. To achieve a high degree of reproducibility, several key practices are followed throughout the training process:

- **Fixed Random Seeds:** All random number generators used in the experiment (e.g., data shuffling, parameter initialisation) are seeded with a fixed value. This practice ensures that the stochastic elements of the training process are consistent across different runs.
- **Version Control for Code and Data:** The complete source code, including the scripts for data preprocessing, model definition, training, and evaluation, is maintained under version control. Similarly, the dataset versions are meticulously tracked to ensure that the exact data splits and preprocessing steps can be replicated.
- **Hyperparameters and Training Configuration:** All hyperparameters, including learning rates, loss weights, batch sizes, and architecture details, along with the training configuration (number of epochs, optimizer settings, etc.), are explicitly documented. These are either hardcoded in the scripts or specified in configuration files that are part of the version-controlled codebase.
- **Environment and Dependency Management:** The training environment, including the operating system, programming language version, and all library dependencies, is precisely specified. Virtual environment with a 'requirements.txt' file is used to recreate the training environment.
- **Logging and Monitoring:** Comprehensive logs of the training and validation metrics, model checkpoints at regular intervals, and the final model state are preserved. This includes detailed logging of the random seed values and any stochastic events, further supporting reproducibility.
- **Use of Deterministic Algorithms:** Where available and applicable, deterministic versions of algorithms and operations (e.g., convolution operations in deep learning libraries) are employed to minimise sources of variability.
- **Detailed Reporting:** The methodology section of any publications or reports resulting from this research includes detailed descriptions of the data, model, training pipeline, and analyses performed, alongside the rationale for key decisions.

3.3 PySINDy Models

The source code for the PySINDy models are available in the following GitHub repository: PySINDy_Models Repository.

3.3.1 PySINDy

The Boussinesq hypothesis simplifies the analysis of turbulence in fluid mechanics. However, this approximation leads to equations that include terms that are hard to measure directly, such as the RST τ.

As we know, PySINDy can discover underlying differential equations from complex data. By applying it to DNS data, the objective would be to derive a new system which models the relationship between the velocity U and the Reynolds stress tensor τ without relying on the Boussinesq hypothesis. Besides, PySINDy could identify dynamic relationships implicit in the DNS data that are not obvious at first glance. Once we have the PySINDy's equations, we will transform them in order to run simulations based on the new turbulence model. Therefore, we will compare this new model with existing models to validate whether or not the results obtained are similar.

3.3.1.1 Selection of the DNS variables

To implement a turbulence model with PySINDy that is able to be as accurate as traditional models such as DNS, it is necessary to select the variables that are important in the turbulent flow. From the DNS data, it was decided to select the following variables:

- y^+, the grid point in wall-normal direction;
- Re_τ, the Reynolds number;
- U, the mean profile of streamwise velocity;
- u'u', v'v', w'w' and u'v', the variances and covariances of velocity fluctuations;
- P, the mean profile of pressure;
- $\frac{dU}{dy}$, the derivative of U in wall-normal direction;

3.3.1.2 Data processing

To obtain a robust PySINDy model, some processing of the DNS data associated with the selected variables may be necessary. Firstly, it is important to use the linear interpolation method for our data. After all, looking at the dataset, the data are not measured at regular intervals with respect to y^+ and the variations between the data points appear to be effectively linear. Consequently, linear interpolation will allow us to fill in the missing observations at certain intervals and may also allow us to increase the number of data points that can be utilised for training and testing the model. From a general point of view, interpolation will help to increase the reliability of our data and will allow us to capture the behaviour of turbulence more accurately. Moreover, in order to enrich the PySINDy model and to improve its accuracy, it was decided to use numerical differentiation using the gradient method in order to compute the first and the second derivatives of the selected variables. To compute the derivatives, we used the gradient method from the Python library Numpy.

3.3.1.3 Data partitioning

We investigated two different approaches to splitting our DNS data into training and test datasets: For the first approach, we used the train_test_split method available in the Scikit-Learn Python library to split our data. However, we found out that the splitting performed by this method is random, which may alter the sequential structure inherent in the time series. For the second approach, we used a division based on the Reynolds number, allocating data for a specific Reynolds number to the training dataset and data corresponding to any other Reynolds number to the test dataset. This method keeps the temporal integrity of the time series data, an

essential condition for the coherent application of PySINDy and for capturing physical dynamics. It was finally decided to apply the second approach for training and testing the PySINDy model as it looked the most appropriate to be used in the development of a PySINDy model.

3.3.1.4 Selection of hyperparameters (Feature_Library and parameters in optimizer)

In order to optimise the PySINDy model and make it more robust, we implemented our own version of the GridSearch algorithm to select the most effective hyperparameters in order to obtain the most accurate model possible. Therefore, the application of this method will let us test a predefined set of values for a range of hyperparameters. In PySINDy, the most relevant hyperparameters are the Feature_Library and the parameters of the optimizer. With our PySINDy model, we will just use the GridSearch algorithm in order to select the optimal Feature_Library from the following set:

- PolynomialLibrary of degree 1;
- PolynomialLibrary of degree 2;
- FourierLibrary;

Due to an iterative process, this approach guarantees both the robustness and the fine-tuning of the method to our DNS data. For the optimizer, the default one will be employed: Sequential Threshold Least Squares (STLSQ). This optimizer has two hyperparameters: the threshold and the alpha coefficient. The GridSearch algorithm will not be used to select the value of these two hyperparameters because they are both extremely sensitive to the data used and it is therefore better at this stage to adjust them manually in order to obtain the best combination of parameters based on the training data provided to the model.

3.3.1.5 Training, prediction and evaluation of the PySINDy model

To train the PySINDy model, we used the training dataset obtained after data splitting and the hyperparameters obtained with the GridSearch algorithm. In the following, you will find the set of features variables used for the PySINDy model:

- U, u'u', v'v', w'w', u'v', P, $\frac{dU}{dy}$, $\frac{du'u'}{dy}$, $\frac{dv'v'}{dy}$, $\frac{dw'w'}{dy}$, $\frac{du'v'}{dy}$ and $\frac{dP}{dy}$

Here's the set of target variables used for the PySINDy model:

- $\frac{dU}{dy}$, $\frac{du'u'}{dy}$, $\frac{dv'v'}{dy}$, $\frac{dw'w'}{dy}$, $\frac{du'v'}{dy}$, $\frac{dP}{dy}$, $\frac{d^2U}{dy^2}$, $\frac{d^2u'u'}{dy^2}$, $\frac{d^2v'v'}{dy^2}$, $\frac{d^2w'w'}{dy^2}$, $\frac{d^2u'v'}{dy^2}$ and $\frac{d^2P}{dy^2}$

Training the PySINDy model will generate a set of equations. These PySINDy equations will be used to simulate the PySINDy model by using the integration method (solve_ivp) available in the scipy.integrate package on the PySINDy equations. With this simulation, we are able to predict the values for the velocity U and the variances and covariances of the velocity fluctuations (u'u', v'v', w'w', u'v'). For the evaluation of the model's performance and robustness, the focus remains on metrics such as the coefficient of determination (R^2) and MSE. Besides, in order to improve our analysis of the simulation's validity, it was decided to also use the Mean Absolute Error (MAE) metric. Our decision is therefore based on MAE's capability to deliver a clear measure of the average absolute error between our predictions and the actual values

3.3.2 PySINDy combined with PCA

After developing the PySINDy model, we tried to optimise it in order to have a model that predicts turbulent flow as accurately as possible. We thought about the possible use of feature extraction techniques and tried to integrate them into the PySINDy modeling process. First of all, it was necessary to decide which feature extraction method would be most appropriate to use in the development of the PySINDy model. Based on our literature review, we assumed that PCA would be the most suitable method for our case study.

Regarding the integration of PCA into the PySINDy modeling process, we noted that the use of PCA would take place just after the application of the interpolation method. For this, we have the Scikit-Learn library, which provides a complete set of methods for performing PCA. First, we created an instance of the PCA class by setting the n_components parameter in order to specify the number of principal components to be retained. For our case study, it was decided to compute the first three principal components, specifying n_components=3. For the PCA input variables, we have selected the covariances and variances of the velocity fluctuations (u'u', v'v', w'w' and u'v'). In order to determine the new representation and extract the principal components, we used the fit_transform method on a standardised dataset.

With this method, the first three principal axes will be calculated, and the dataset will be transformed accordingly. The result of this transformation will generate the three principal components derived from the dataset. In order determine whether all the principal components generated are necessary, we will check the value of the variance explained for each axis. The objective is to keep the number of axes to a strict minimum, in order to have a cumulative sum of explained variance close to 1. In addition, after identifying the axes to be kept, in order to have a better comprehension, we have drawn the correlation circle of the original variables in relation to the new axes. With this correlation circle, it will enable us to understand how the original variables are represented in relation to the new axes obtained through PCA.

3.3.3 PySINDy optimised by GA

In addition to using the feature extraction method to optimise the PySINDy model, we have also tried to implement the GA to optimise the PySINDy model in order to have the model that can most accurately capture the behaviours in the turbulent flow. Concerning the integration of the GA in the PySINDy process, it will be used after the model training and the generation of the set of equations. The objective will be to optimise all the coefficients present in the PySINDy equations. To achieve this, it was decided to use the methods and tools available in the Python DEAP package to help us develop the GA. Here are the steps below:

1. **Defining the cost function:**
   For the development of GAs, a key step is to define the cost function. In our case study of modeling turbulence with PySINDy model, the cost function is used in order to update the model coefficients. At this stage, the main metric that we are seeking to minimise is the MSE.

2. **Set up the genetic algorithm:**
   In order to construct the individual categories (Individual) and the fitness functions (FitnessMin) adapted to the optimisation, which are important concepts in GAs, it was decided to use the DEAP library. In our situation, as the objective is to minimise the overall

cost (i.e. MSE), we therefore set the fitness function weight to -1.0, indicating that lower fitness values represent more optimal solutions.

3. **Initialise the toolbox:** For GAs, the toolbox is an essential component that delimits the construction of individuals, essentially our candidate solutions. In order to randomly generate attributes, which in our case correspond to the coefficients of the PySINDy equations, we need to use the np.random.uniform function. This configuration also includes initialising the population and defining the basic genetic operations: evaluation, crossover (mating), mutation and selection process.

4. **Run the GA:** In order to generate a range of potential solutions, we initialise a population using the predefined individual generation procedure. Subsequently, it was decided to use the algorithms.eaSimple function, which manages the correct execution of the GA, including mating, mutation and selection operations. Using an iterative process where the algorithm will iterate over a specified number of generations (ngen), we will identify the fittest individuals and relevant statistics throughout the evolutionary process.

5. **Output:** Following the execution of the GA, the results are displayed in the form of optimised coefficient values for the top performing individuals.

3.4 Transformer

3.4.1 Data processing

In this project, the Transformer implementation has been studied in alongside the implementation of the PINNs model. To be able to compare the two models precisely, the same input and output are used. The dataset features were extended to include normalised wall-normal distance y/δ, non-dimensional wall distance y^+, friction velocity u_τ, kinematic viscosity ν, and Reynolds shear stress Re_τ. These features are crucial for capturing the physical properties and flow characteristics in fluid dynamics studies, particularly in turbulence modelling. The targets selected for the regression model encompass turbulent flow statistics and mean flow properties:

- Turbulent normal and shear stresses: $u'u'$, $v'v'$, $w'w'$, $u'v'$
- Mean velocity (U)
- Velocity gradient ($\frac{dU}{dy}$)
- Pressure (P)
- Turbulent kinetic energy (k)

Both features and targets were standardised using `StandardScaler` from `sklearn.preprocessing` to ensure they have zero mean and unit variance, which aids in the effective training of neural networks by providing numerical stability and improving convergence properties.

3.4.2 Data partitioning

Splitting Dataset: This normalisation is divided into training (80%), validation (10%), and test (10%) sets in order to provide a seamless ecosystem for the models to be trained and evaluated. Undoubtedly this split is needed because it tells whether the model works well or not on the

unseen records.

Conversion to PyTorch Tensors: The data is turned to tensors that are supported by PyTorch framework which in turn permits faster computation and training of models.

3.4.3 Model Architecture

The core of the model is a Transformer architecture designed specifically for regression tasks. The Transformer model comprises the following components:

- **Embedding Layer:** Converts input features of dimension `input_features` to a higher dimensional space (`model_dim`). This layer is implemented using the `nn.Linear` module.
- **Transformer Encoder:** Consists of multiple layers of `nn.TransformerEncoderLayer`, configured with a specified number of heads (`transformer_heads`) and an expansion factor in the feedforward network (`forward_expansion`). The encoder transforms the embedded input through self-attention mechanisms and feedforward networks.
- **Output Layer:** A linear layer (`nn.Linear`) that maps the output of the Transformer encoder to the desired output size (`output_features`).

The model was instantiated with specific hyperparameters such as the number of input features, model dimension, number of Transformer layers, number of heads, forward expansion, and the number of output features.

3.4.4 Evaluation

The model's performance was evaluated using the test set after training. The test loss was computed as the MSE between the model's predictions and the actual values, providing a quantitative measure of the model's generalisation ability.

3.4.5 Visualisation

Training and validation losses were plotted against the number of epochs to visualise the learning process and the convergence behavior of the model. This plot is crucial for diagnosing issues in training, such as overfitting or underfitting. Figure 4.18 shows the training and validation loss curves.

Chapter 4 – Results & Discussions

4.1 RANS models results and discussion

After running all simulations, we can compare results on both models. To run this comparison we will be looking at the stream wise velocity and and the co variances. The main aim of this section is to identify possible limitations within the models and see later how machine learning performs next to RANS models.

4.1.1 Streamwise velocity

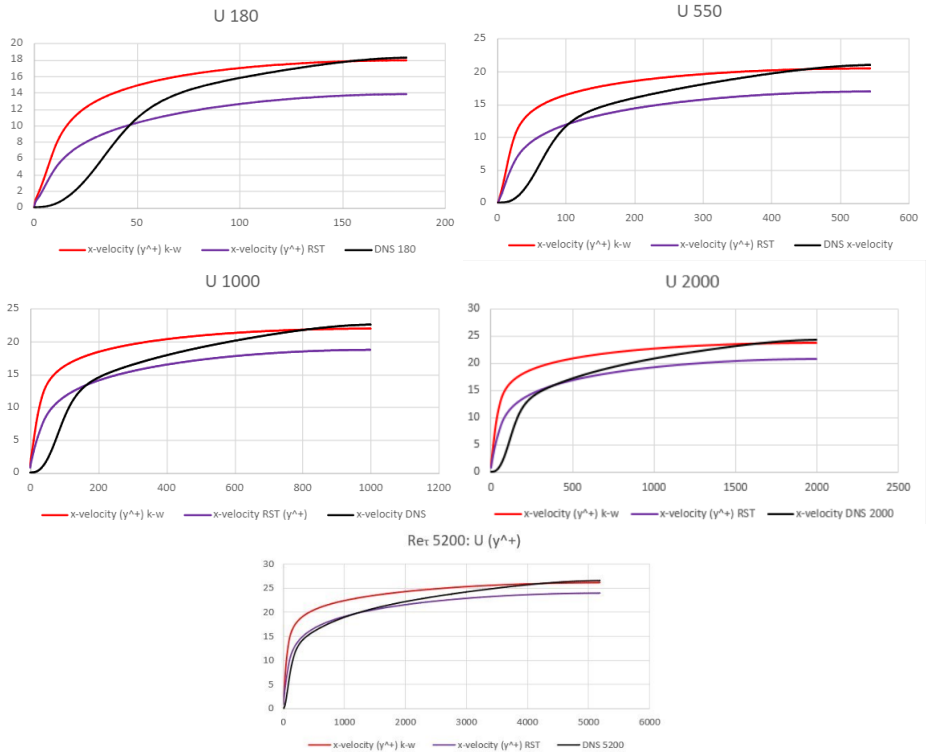


Figure 4.1: Stream wise velocity plots for RANS models and DNS

After comparing the results obtained from the k-omega and Reynolds Stress Model (RSM) simulations, we discern notable differences in the predictions, particularly when examining streamwise velocity. While both models aim to capture turbulent flow behavior, our analysis reveals nuanced disparities.

In our investigation, the k-omega model demonstrates a tendency to provide reasonably accurate predictions of streamwise velocity in certain flow regimes, especially in regions where turbulence levels are moderate. However, discrepancies emerge when applied to highly turbulent flows or near complex geometries, where the model's simplified assumptions may lead to inaccuracies.

Conversely, the RSM model exhibits a more comprehensive representation of turbulence by directly computing Reynolds stress tensor components. Consequently, it offers improved accuracy in predicting streamwise velocity, especially in regions characterized by complex turbulent structures and flow interactions. However it is behind in terms of precision and achieving results like the DNS data base.

The differences in predictions between the two models underscore the importance of selecting an appropriate turbulence model based on the specific characteristics of the flow under consideration. While the k-omega model provides a computationally efficient approach suitable for many engineering applications, the RSM model offers a more detailed and accurate depiction of turbulence physics, albeit at a higher computational cost .

4.1.2 Covariances

Figure 4.2: Covariances of uu, vv and uv (top, mid, bottom) for $Re_\tau \approx 2000$

Upon examining the covariances between the k-omega and Reynolds Stress Model (RSM) simulations, we encountered significant disparities, warranting a closer inspection. While the RSM model yielded coherent and physically plausible results for the covariances, the k-omega model presented challenges. Regrettably, we were unable to plot the k-omega covariances as they fell outside the acceptable range, suggesting potential limitations in the model's ability to accurately capture turbulent momentum exchanges.

Our observations underscore the importance of selecting turbulence models judiciously based on the flow characteristics and intended application. While the k-omega model remains valuable for its computational efficiency in certain scenarios, our findings emphasize the necessity of considering the limitations and accuracy of covariance predictions.

4.2 PINNs Model

It is important to note that the following results can be reproduced using seed '123' and the batch size of 32 training samples per step.

4.2.1 Hyper-parameters Tuning

During the hyper-parameter tuning process, we explored the influence of model size, learning rate, and activation functions on the performance of our PINN. Table 4.2 showcases the meticulous selection process for the ideal model size alongside the final choice of the ELU as the activation function, revealing a minimum total MSE on the test dataset when employing a neural network configuration of 4×128. Similarly, in our examination of the optimal learning rate, detailed in Table 4.1, the learning rate of 5×10^{-4} emerged as the most effective, further substantiating the findings from our model size exploration. The impact of the activation function on model performance is demonstrated in Table 4.3, where ELU outperformed other tested activation functions in terms of total MSE on the test dataset. Furthermore, the balancing of physical and data loss weights was critically evaluated, as summarised in Table 4.4, pinpointing an equal weighting strategy as most conducive to minimising the total MSE.

Batch Size	NN Size	Learning Rate	Epoch	Activation Func.	Total MSE
32	4×128	1×10^{-3}	1000	ELU	0.001421
32	$\mathbf{4 \times 128}$	$\mathbf{5 \times 10^{-4}}$	**1000**	**ELU**	**0.000554**
32	4×128	1×10^{-4}	1000	ELU	0.000719

Table 4.1: Selection of the best initial learning rate on a 4-layer model with 128 nodes/layer using the total MSE on the test dataset

Batch Size	NN Size	Learning Rate	Epoch	Activation Func.	Total MSE
32	2×32	5×10^{-4}	1000	ELU	0.173699
32	3×32	5×10^{-4}	1000	ELU	0.039388
32	4×32	5×10^{-4}	1000	ELU	0.010825
32	8×32	5×10^{-4}	1000	ELU	0.002820
32	2×64	5×10^{-4}	1000	ELU	0.107276
32	3×64	5×10^{-4}	1000	ELU	0.016589
32	4×64	5×10^{-4}	1000	ELU	0.004392
32	8×64	5×10^{-4}	1000	ELU	0.001045
32	2×128	5×10^{-4}	1000	ELU	0.484129
32	3×128	5×10^{-4}	1000	ELU	0.110587
32	$\mathbf{4 \times 128}$	$\mathbf{5 \times 10^{-4}}$	**1000**	**ELU**	**0.000554**
32	8×128	5×10^{-4}	1000	ELU	0.000778
32	2×256	5×10^{-4}	1000	ELU	0.053500
32	3×256	5×10^{-4}	1000	ELU	0.001884
32	4×256	5×10^{-4}	1000	ELU	0.000616
32	8×256	5×10^{-4}	1000	ELU	0.015669

Table 4.2: Selection of the best model size with the final activation function (ELU) using the total MSE on the test dataset

Batch Size	NN Size	Learning Rate	Epoch	Activation Func.	Total MSE
32	$\mathbf{4 \times 128}$	$\mathbf{5 \times 10^{-4}}$	**1000**	**ELU**	**0.000554**
32	4×128	5×10^{-4}	1000	SELU	0.001051
32	4×128	5×10^{-4}	1000	TANH	0.204692

Table 4.3: Selection of the best activation function on a 4-layer model with 128 nodes/layer using the total MSE on the test dataset

Batch Size	NN Size	Learning Rate	Epoch	Activation Func.	w_{phys}	w_{data}	Total MSE
32	$\mathbf{4 \times 128}$	$\mathbf{5 \times 10^{-4}}$	**1000**	**ELU**	**1**	**1**	**0.000551**
32	4×128	5×10^{-4}	1000	ELU	1	1×10^{-1}	0.009988
32	4×128	5×10^{-4}	1000	ELU	1	1×10^{-2}	0.002866
32	4×128	5×10^{-4}	1000	ELU	1×10^{1}	1	0.007187
32	4×128	5×10^{-4}	1000	ELU	1×10^{1}	1×10^{-1}	0.026153
32	4×128	5×10^{-4}	1000	ELU	1×10^{1}	1×10^{-2}	0.015185
32	4×128	5×10^{-4}	1000	ELU	1×10^{2}	1	0.002753
32	4×128	5×10^{-4}	1000	ELU	1×10^{2}	1×10^{-1}	0.044795
32	4×128	5×10^{-4}	1000	ELU	1×10^{2}	1×10^{-2}	0.602007

Table 4.4: Initial weights selection for physic and data losses in a 4-layer, 128 nodes/layer model based on total MSE on the test dataset

4.2.2 Training Step

Figure 4.3 illustrates the training losses, including the total training loss, the loss from the PDE, and the loss from the data over multiple epochs for four successive cycles of the PINN model.

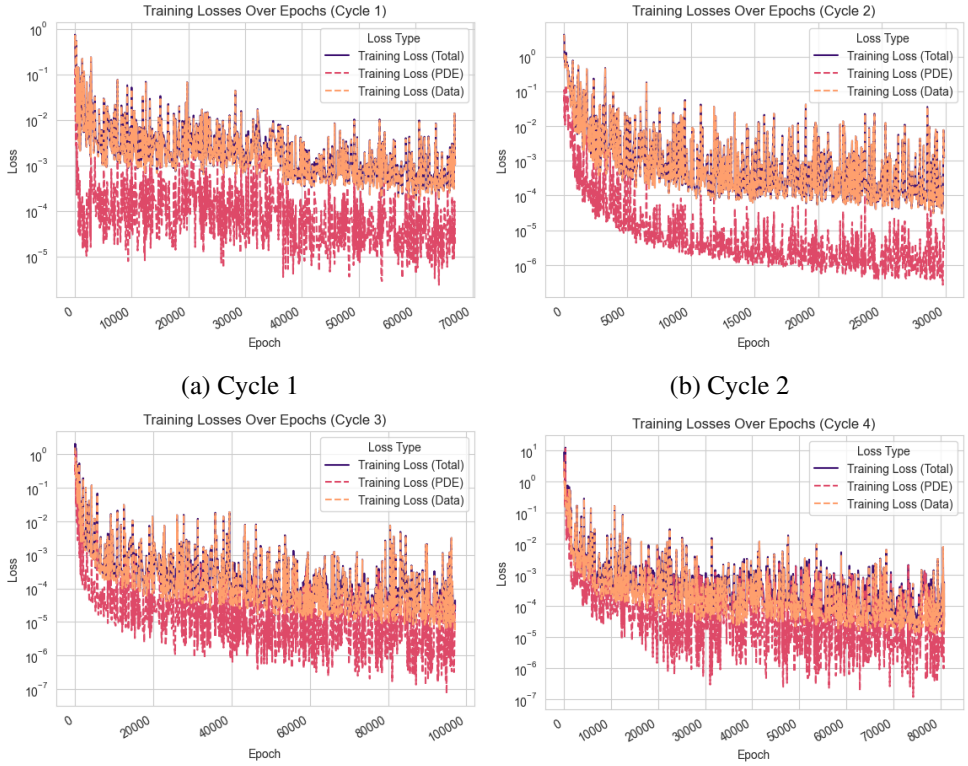
For Cycle 1 (Figure 4.3a), as the training progresses, the total loss and PDE loss decrease gradually, but the data loss remains higher, suggesting that the model is initially struggling to fit the data accurately. The PDE loss is lower than the data loss, implying that the model is prioritising the satisfaction of the governing equations over data fitting in this cycle.

In Cycle 2 (Figure 4.3b), the data loss, remains high initially but decreases as the training progresses. The model is focusing more on fitting the data while maintaining the ability to satisfy the governing equations learned in the previous cycle. The PDE loss and data loss converge to similar levels towards the end of the cycle, indicating a balance between the two objectives.

Cycle 3 (Figure 4.3c) shows a more consistent decrease in all losses throughout the training process. The total loss, PDE loss, and data loss all start at lower levels compared to the previous cycles, indicating that the model has improved its overall performance. The model is still prioritising the satisfaction of the governing equations over data fitting, but the gap between the two losses is smaller than in previous cycles.

In the final cycle (Figure 4.3d), all losses continue to decrease and converge to relatively low values. The total loss, PDE loss, and data loss are closely aligned, suggesting that the model has achieved a good balance between satisfying the governing equations and fitting the data.

The convergence of the losses indicates that the model has effectively learned the underlying physics and can accurately represent the data while adhering to the physics-based constraints.



(a) Cycle 1

(b) Cycle 2

(c) Cycle 3

(d) Cycle 4

Figure 4.3: Training Losses Over Epochs for Four Cycles

Figure 4.4 illustrates the training and validation losses over multiple epochs for four successive cycles of the PINN model.

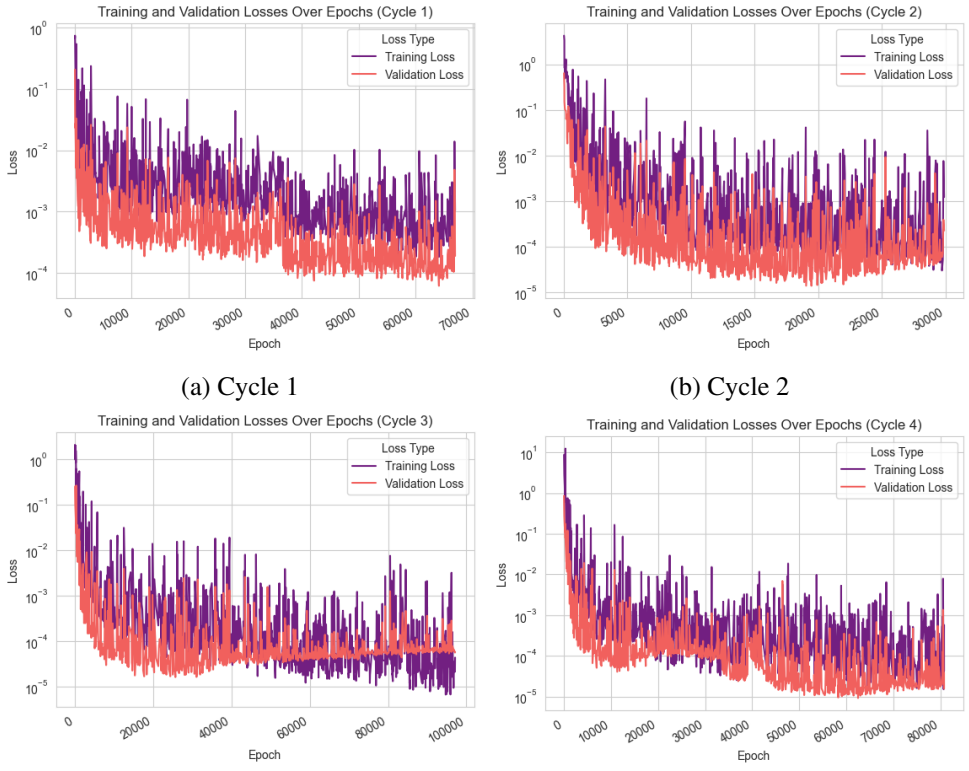
The first subfigure 4.4a shows the training and validation losses over epochs for cycle 1. Initially, both losses are high, indicating that the model has not learned the underlying patterns and relationships in the data. As learning progresses, the learning and validation losses decrease. The difference between the two losses indicates a certain level of over-fitting.

In Cycle 2 (Figure 4.4b), training and validation losses start at a lower level than in Cycle 1, implying that the model has retained some knowledge from the previous cycle. Both losses show a downward trend, but the validation loss remains higher than the learning loss throughout the cycle.

The Cycle 3 (Figure 4.4c) shows a more promising trend. While training and validation losses start at relatively high values, they decrease significantly as training progresses. Notably, the validation loss closely follows the training loss, with a smaller gap compared to previous cycles. This suggests that the model achieves better generalisation performance and is less prone to overfitting. This result is due to the increased weight of PDE loss, as shown in the figure 4.3c.

In the last cycle, cycle 4 (Figure 4.4d), the training and validation losses continue to decrease, reaching lower values than in the previous cycles. The gap between the two losses narrows

further, indicating that the model has improved its ability to generalise to the validation set. Indeed, the weight of the PDE loss was again increased. The validation loss closely follows the learning loss, suggesting that the model has achieved a good balance between adaptation to the learning data and generalisation to the unseen data.



(a) Cycle 1

(b) Cycle 2

(c) Cycle 3

(d) Cycle 4

Figure 4.4: Training & Validation Loss Over Epochs for Four Cycles

The total training time took 10.3 hours. Here is how long it took to obtain each of the best snapshots of each cycle:

- **Cycle 1:** 3.5 hours (58886 epoch)
- **Cycle 2:** 2.7 hours (16290 epoch)
- **Cycle 3:** 1.0 hour (20332 epoch)
- **Cycle 4:** 3.1 hours (54851 epoch)

4.2.3 Test Step

4.2.3.1 Best Trained Model Selection

After training, the best snapshot from each cycle was used to perform inference over the test dataset. As shown in Figure 4.5, the best model is the one from Cycle 3, which is why we decited to use it for the following analysis.

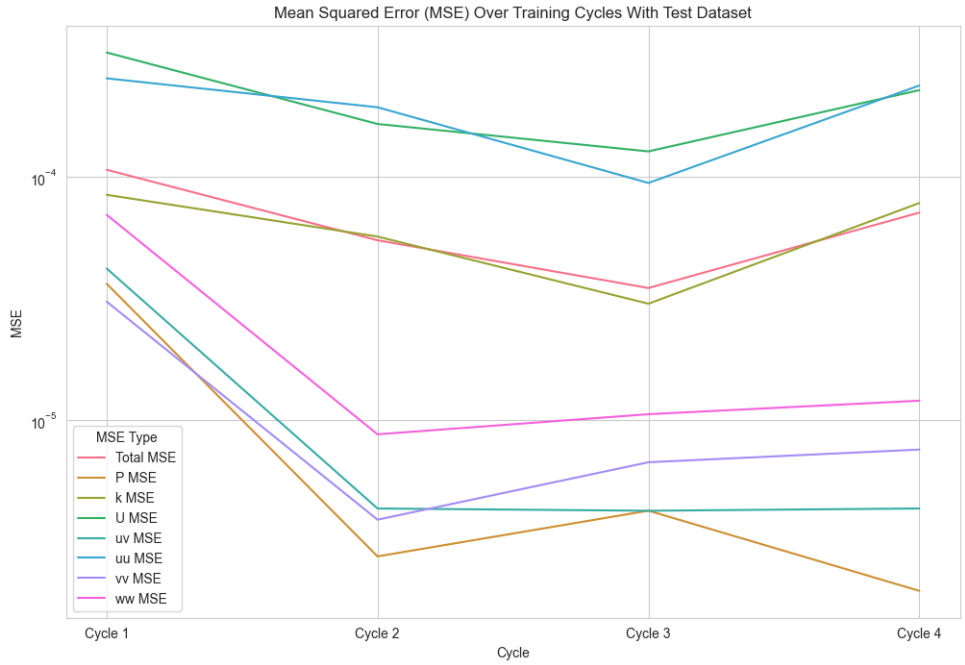
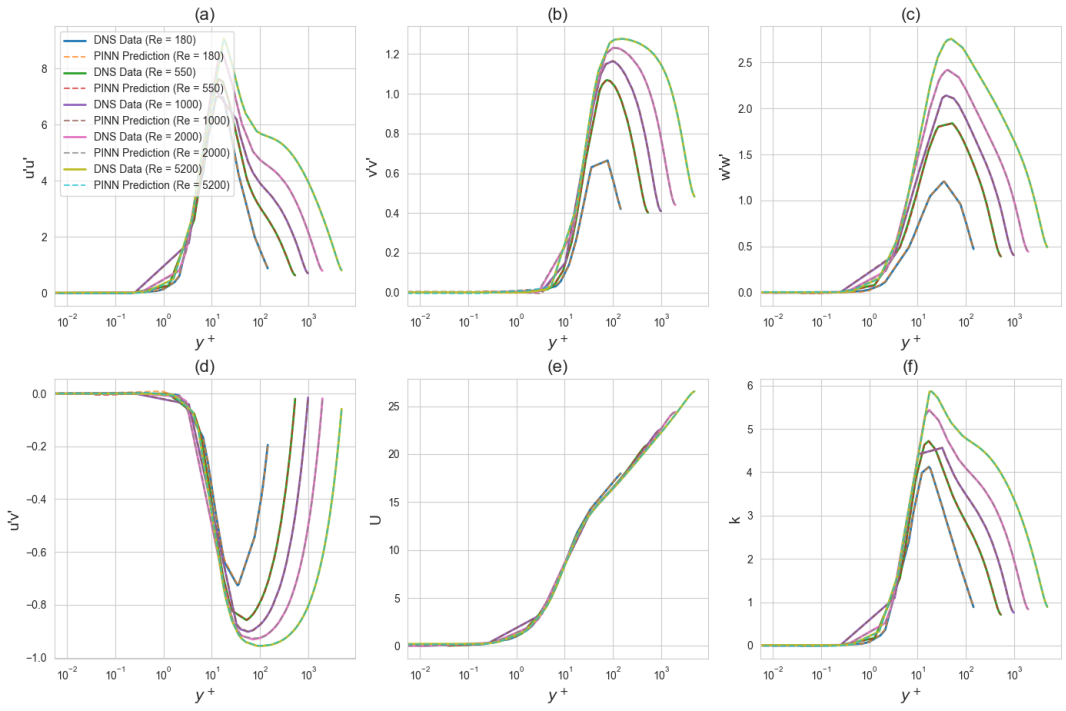


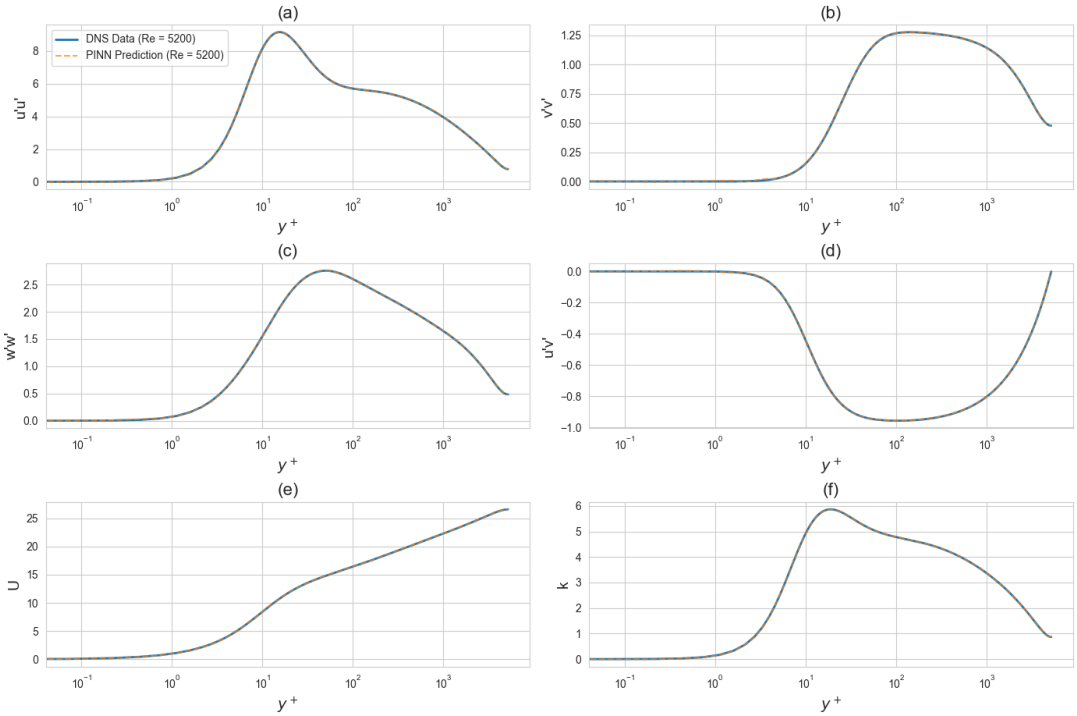
Figure 4.5: MSE Over Training Cycles With Test Dataset

4.2.3.2 Best Trained Model Inference Plots

As shown in the figures 4.7a and 4.6a, the trained PINN Model (Cycle 3) perfectly captures the dynamics of turbulent flow in the boundary layer.



(a) $Re \in (180, 550, 1000, 2000, 5200)$



(a) $Re = 5200$

Figure 4.7: Comparison of PINNs predictions and DNS Data

4.2.3.3 Best Trained Model Relative ℓ_2-norm of Errors Analysis

To evaluate the accuracy of the PINNs model predictions, we consider the relative ℓ_2-norm of errors E_i on all the computational points and for the ith variable as [14]:

$$E_i = \frac{\left\| U_i - \widetilde{U}_i \right\|_2}{\|U_i\|_2} \times 100 \tag{4.1}$$

where $\|.\|_2$ denotes ℓ_2-norm, U the vectors of reference data, and $\widetilde{U}$ indicate the vectors PINNs predictions. Table 4.5 details the relative ℓ_2-norm of errors for various quantities over different Reynolds numbers. Observations from the table indicate several key points:

- The relative errors generally decrease with the increase in Reynolds numbers, suggesting that PINNs potentially have improved prediction accuracy for higher flow rates. For example, the error in velocity prediction (E_u) noticeably decreases from 0.26% at Re=180 to 0.03% at both Re=2000 and Re=1000.
- Different quantities exhibit varying degrees of error, with turbulent properties (e.g., $E_{u'u'}$, $E_{v'v'}$) presenting higher errors compared to scalar quantities like velocity (E_u) and temperature (E_θ), highlighting the challenges in accurately predicting turbulent flows.
- A consistent trend in the reduction of error percentages across most quantities as Reynolds numbers increase, showcases the adaptability and reliability of the PINN approach in a range of fluid dynamic conditions.

Re	E_u	E_θ	E_k	$E_{u'u'}$	$E_{v'v'}$	$E_{w'w'}$	$E_{u'v'}$
180	0.26%	0.79%	0.57%	0.57%	1.19%	0.73%	1.12%
550	0.10%	0.34%	0.39%	0.52%	0.39%	0.51%	0.46%
1000	0.03%	0.33%	0.13%	0.18%	0.35%	0.20%	0.32%
2000	0.03%	0.21%	0.11%	0.18%	0.32%	0.17%	0.33%
5200	0.04%	0.17%	0.10%	0.16%	0.21%	0.15%	0.20%

Table 4.5: Relative ℓ_2-norm of errors, in the predictions using PINNs over the Test Dataset

4.2.3.4 Best Trained Model Mean Square Error Analysis

The MSE data provided in Table 4.6 underscore the model's performance in quantitative terms across different predictions:

- The Total MSE of 3.48×10^{-5} represents the overall prediction accuracy of the PINN model, indicating a generally strong performance across all tested metrics.
- Analysis of MSE for specific quantities reveals high accuracy in pressure prediction (MSE P = 4.23×10^{-6}) and challenges in accurately predicting velocity (MSE U = 1.27×10^{-4}), highlighting areas where PINNs perform well and where improvements might be beneficial.
- The low MSE values for velocity gradient (MSE dUdy = 1.81×10^{-6}) and turbulent kinetic energy (MSE k = 3.00×10^{-5}) further demonstrate the capability of PINNs in capturing crucial aspects of fluid motion and turbulence.
- Among the turbulent quantities, the higher MSE for uu (MSE uu = 9.42×10^{-5}) suggests predicting turbulent stresses, particularly in the streamwise direction, poses a significant challenge, indicating a potential area for focused improvement in modeling turbulent flows.

Test Metric	**Value**	
Total MSE	3.48	$\times 10^{-5}$
MSE P	4.23	$\times 10^{-6}$
MSE U	1.27	$\times 10^{-4}$
MSE dUdy	1.81	$\times 10^{-6}$
MSE k	3.00	$\times 10^{-5}$
MSE uu	9.42	$\times 10^{-5}$
MSE vv	6.69	$\times 10^{-6}$
MSE ww	1.05	$\times 10^{-5}$
MSE uv	4.23	$\times 10^{-6}$

Table 4.6: MSE over the Test Dataset

4.2.4 Overall Conclusion

The collective analysis of the relative ℓ_2-norm of errors and MSE values from Tables 4.5 and 4.6 reflects the efficacy of PINNs in accurately modeling fluid dynamics. The results showcase the promise of PINNs, especially at higher flow rates, while also highlighting the need

for enhanced modeling strategies to tackle the intricacies of turbulent flow predictions more effectively. Through targeted enhancements, there is potential for PINNs to achieve even greater accuracy, making them a powerful tool for simulating and understanding complex fluid dynamics systems.

4.3 Different PySINDy models Results Analysis

4.3.1 PySINDy

4.3.1.1 Interpolation Method

The results presented below from our PySINDy model were generated by analysing DNS data obtained from Oden, using the variables described in the methodology section for a set of Reynolds numbers (5200, 2000, 1000, 550, 180). Figure 4.8 shows several graphs describing velocity profiles, as well as the variances or covariances of velocity fluctuations, as a function of y^+.

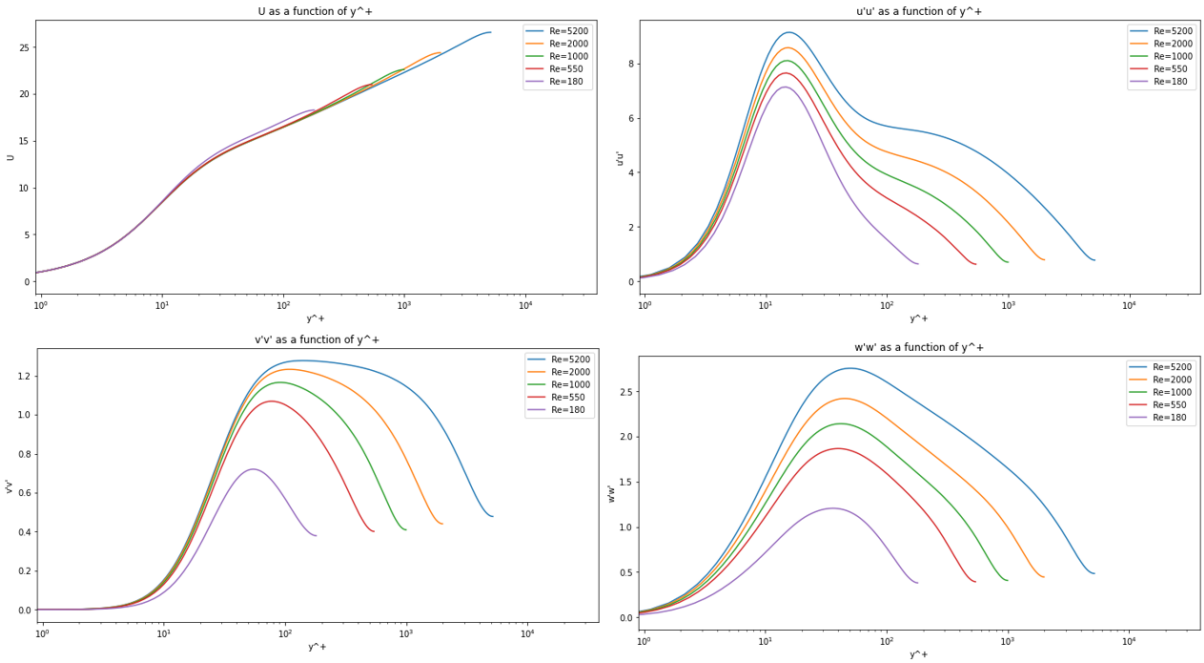


Figure 4.8: Representation of velocity, covariances and variances of velocity fluctuations as a function of y^+ for different Reynolds numbers for DNS data (before interpolation)

Building on the methods described in the methodology section 3.3.1.2, we used the interpolation method to enrich our data set, generating a complete set of 10,000 data points. This enriched data set enabled a more detailed and continuous representation of turbulent flow characteristics. The following graphs, which we present here, allow a visual comparison of velocity and velocity fluctuations as a function of y^+ before and after the interpolation process.

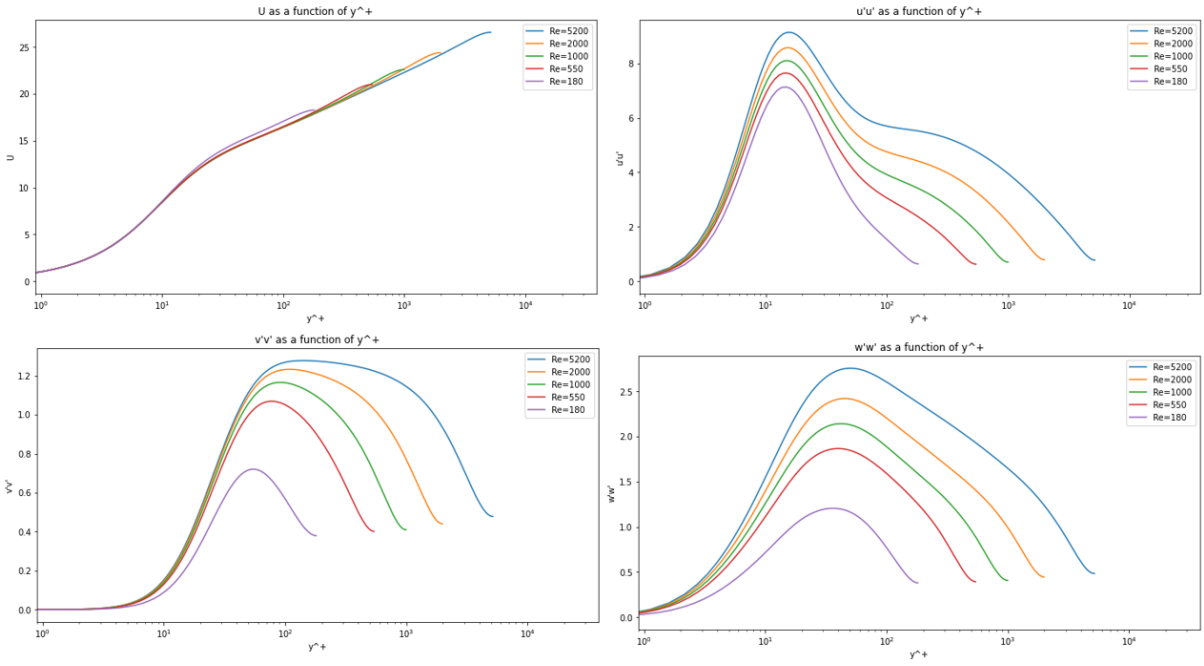


Figure 4.9: Representation of velocity, covariances and variances of velocity fluctuations as a function of y^+ for different Reynolds numbers for DNS data (after interpolation)

Figures 4.8 and 4.9 show that the representation of velocity and velocity fluctuations as a function of y^+ is similar before and after interpolation. Consequently, these graphs not only validate the interpolation method, but also provide a better understanding of the fluid dynamics involved at different turbulence scales, as indicated by the range of Reynolds numbers.

4.3.1.2 Output Equations

In this results section, we adopted the strategy of using the data corresponding to the highest Reynolds number available, 5200, for training, while utilising the data associated with a Reynolds number of 2000 for testing. The decision is based on the fact that higher Reynolds numbers can capture more complex turbulence behaviour.

The application of the GridSearch algorithm as mentioned in the methodology (section 3.3.1.4) enabled us to identify the optimal feature_library as the PolynomialLibrary of degree 1, as illutrated in figure 4.10.

```
Optimal hyperparameters: {'FeatureLibrary': 'PolynomialLibrary1'}
```

Figure 4.10: Optimal hyperparameter obtained through the grid search algorithm

After an extensive manual testing phase for a range of threshold and alpha coefficient values, we have converged on the parameters alpha = 0.001 and threshold = 0.0001.

This study has implemented the PySINDy Model to develop an advanced turbulence model of turbulence model. PySINDy Model generated dynamic equations for each variable to represent the turbulent flow. These equations are fundamental as they encapsulate the interactions between the velocity components and the pressure gradients throughout the dynamic flow field. The equations are shown below.

```
PysINDy Model:
(U)' = 1.000 dU/dy
(u'u')' = 1.000 du'u'/dy
(v'v')' = 1.000 dv'v'/dy
(w'w')' = 1.000 dw'w'/dy
(u'v')' = 1.000 du'v'/dy
(P)' = 1.000 dP/dy
(dU/dy)' = 0.010 v'v' + 0.011 P + -10.800 dv'v'/dy + 0.999 du'v'/dy + -10.787 dP/dy
(du'u'/dy)' = -0.096 1 + 0.003 U + -0.008 u'u' + 0.753 v'v' + 0.039 w'w' + -0.002 u'v' + 0.765 P + 0.230 dU/dy + -0.217 du'u'/d
y + -0.242 dv'v'/dy + 2.519 dw'w'/dy + 6.150 du'v'/dy + 1.270 dP/dy
(dv'v'/dy)' = 0.004 v'v' + 0.004 P + 1.023 dv'v'/dy + 0.015 dw'w'/dy + -0.046 du'v'/dy + 1.090 dP/dy
(dw'w'/dy)' = -0.061 v'v' + -0.061 P + 0.079 dU/dy + -0.056 du'u'/dy + -7.455 dv'v'/dy + 0.024 dw'w'/dy + 0.191 du'v'/dy + -6.7
80 dP/dy
(du'v'/dy)' = -0.008 v'v' + -0.008 P + 0.002 dU/dy + -0.021 du'u'/dy + 3.477 dv'v'/dy + -0.030 dw'w'/dy + -0.379 du'v'/dy + 3.6
24 dP/dy
(dP/dy)' = -0.005 v'v' + -0.004 P + -1.026 dv'v'/dy + -0.015 dw'w'/dy + 0.046 du'v'/dy + -1.092 dP/dy
```

Figure 4.11: Output equations generated by PySINDy

Notable terms in the equations, such as $\frac{dU}{dy}$ and $\frac{dP}{dy}$ represent the changes in velocity and pressure along the direction of fluid flow, which follow the physical principles of fluid dynamics. Such terms enrich our understanding of fluid behavior under different conditions.

4.3.1.3 Comparison with DNS Data

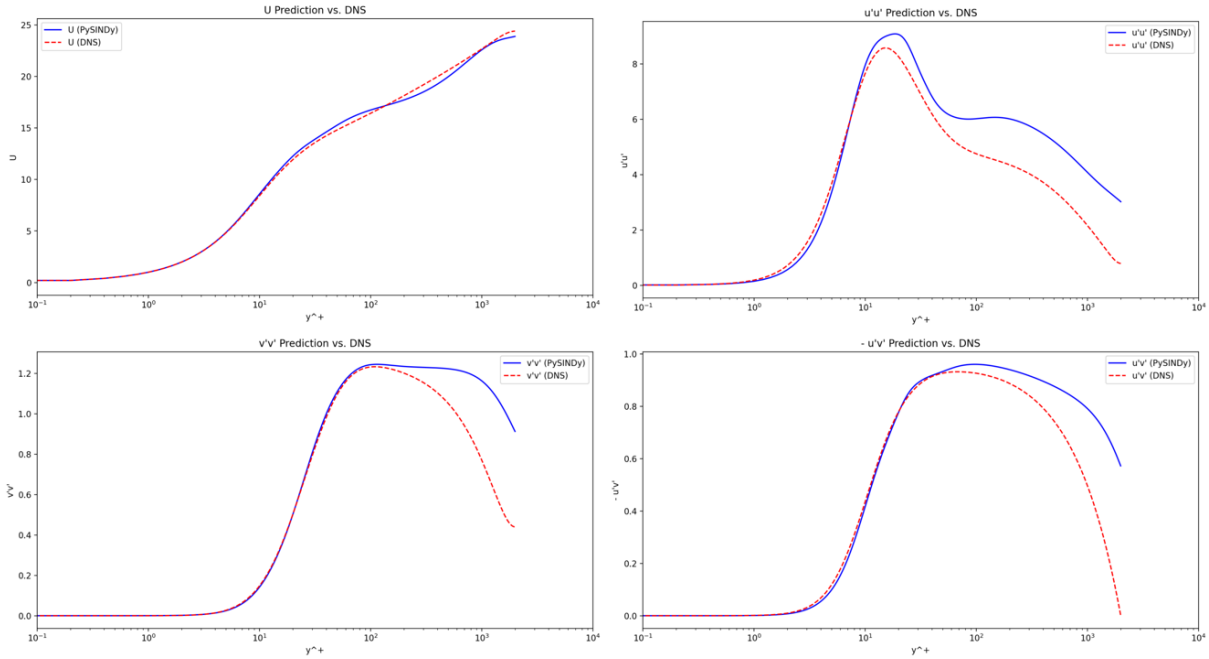


Figure 4.12: Comparison of PySINDy predictions and DNS Data (Re = 2000)

Upon comparison with DNS data (Figure 4.12), our PySINDy model demonstrates a close correspondence, particularly for terms like U, u'u', v'v', w'w' and u'v' as functions of y^+, the normalised distance from the wall. This strong alignment highlights PySINDy's capability to capture essential characteristics of fluid behavior, especially in the critical areas near boundary layers where turbulence is most affected by the presence of the wall.

4.3.1.4 Improving PySINDy Model

After training our PySINDy model and generating this set of equations, we evaluated model performance and obtained an R^2 of 0.9068654365 and an MSE of 0.0000045599. At first

glance, the scores obtained for the coefficient of determination and the MSE are excellent, indicating that the model is perfectly suited to the dataset used. However, in the case of PySINDy, although the R^2 value is already very high and the MSE is also very low, neither of these two values, on its own, allows us to conclude with certainty that the PySINDy model is effective in the specific field of fluid mechanics. From the figure above, we can clearly see that in the near-wall region (i.e. low y^+), PySINDy can accurately model the physics prevailing in this zone, but in the far-wall region (i.e. high y^+), the PySINDy model has some limitations with a clear deviation from the DNS data with regard to the variances and covariances of the velocity fluctuations, indicating that the model can still be improved.

To improve the PySINDy model's capability to predict turbulence, it will be necessary to refine the model in the future in order to take into account energy dynamics on larger scales. Indeed, as turbulence consists of a series of vortices and vortexes of varying sizes, we assume that it can predict these larger structures more accurately for realistic simulations by incorporating higher-order derivatives and nonlinear terms that reflect the energy cascades in turbulence.

Besides, we realise the necessity of a larger data set covering different flow conditions in order to enrich the model. By performing more high-fidelity experiments and direct numerical simulations, the PySINDy model will learn from a wider range of turbulence phenomena. At the end, it is important that the models offer better generalisation across a range of conditions, and deliver highly accurate predictions, in order to effectively meet the needs of industries that rely on accurate fluid dynamics modeling.

4.3.2 PySINDy combined with PCA

We have further developed the PySINDy model by integrating PCA as explained in the methodology section 3.3.2. For this purpose, we used the same DNS dataset as the one previously used in the original PySINDy model. The application of PCA indicated that the first two principal components significantly captured the dynamics of the input variables. Indeed, the cumulative sum of variances explained by these components amounted to 0.99, indicating a strong correlation with the original data and confirming the relevance of our feature selection. This high proportion of explained variance is illustrated in Figure 4.13, which shows graphically the contribution of each principal component.

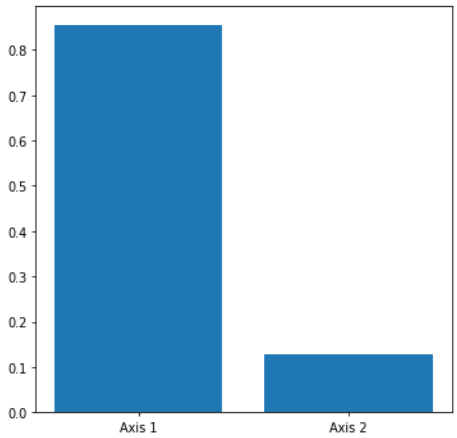


Figure 4.13: Representation of the inertia (explained variance) of the axes generated by PCA

To measure the effectiveness of integrating the PCA into the PySINDy model, we have kept

the PolynomialLibrary of degree 1 as the Feature_library for the hyperparameters of our model, and have also preserved the values of the previously defined hyperparameters, i.e a threshold of 0.001 and an alpha coefficient of 0.0001 for the STLSQ optimizer. These parameters were retained because of their performance during the initial development of the PySINDy model, therefore guaranteeing continuity in the quality of the results obtained. The resulting equations, presented below (Figure 4.14), give the modeling obtained by combining PySINDy and PCA. The objective of this hybrid method is to optimise the accuracy of the model by reducing noise and by focusing analysis on the most important variables.

```
PysINDy Model:
(U)' = 1.000 dU/dy
(PC1)' = 1.000 dPC1/dy
(PC2)' = 1.000 dPC2/dy
(P)' = 1.000 dP/dy
(dU/dy)' = 0.022 dU/dy + -0.233 dPC1/dy + 0.141 dPC2/dy + -0.611 dP/dy
(dPC1/dy)' = -0.012 1 + -0.002 PC1 + -0.003 PC2 + -0.014 P + 0.266 dU/dy + -0.622 dPC1/dy + 0.351 dPC2/dy + -1.933 dP/dy
(dPC2/dy)' = -0.010 1 + -0.002 PC1 + -0.004 PC2 + -0.011 P + 0.242 dU/dy + -0.647 dPC1/dy + 0.349 dPC2/dy + -2.433 dP/dy
(dP/dy)' = -0.012 dPC1/dy + 0.005 dPC2/dy + -0.084 dP/dy
```

Figure 4.14: PySINDy equations generated after using PCA

Figure 4.14 highlights the complexity of the physical interpretation of the variables projected on the first and second principal components (PC1 and PC2) from the PCA. Understanding their concrete meaning in the physical context of turbulent flows remains a challenge. To validate the equations produced by this hybrid model, we simulated them using the same integration method as the one applied to the initial PySINDy model. This approach enabled us to obtain velocity predictions as well as projections on the principal components of the PCA. The following graphs illustrate a direct comparison between the values predicted by our model and the actual data as a function of y^+.

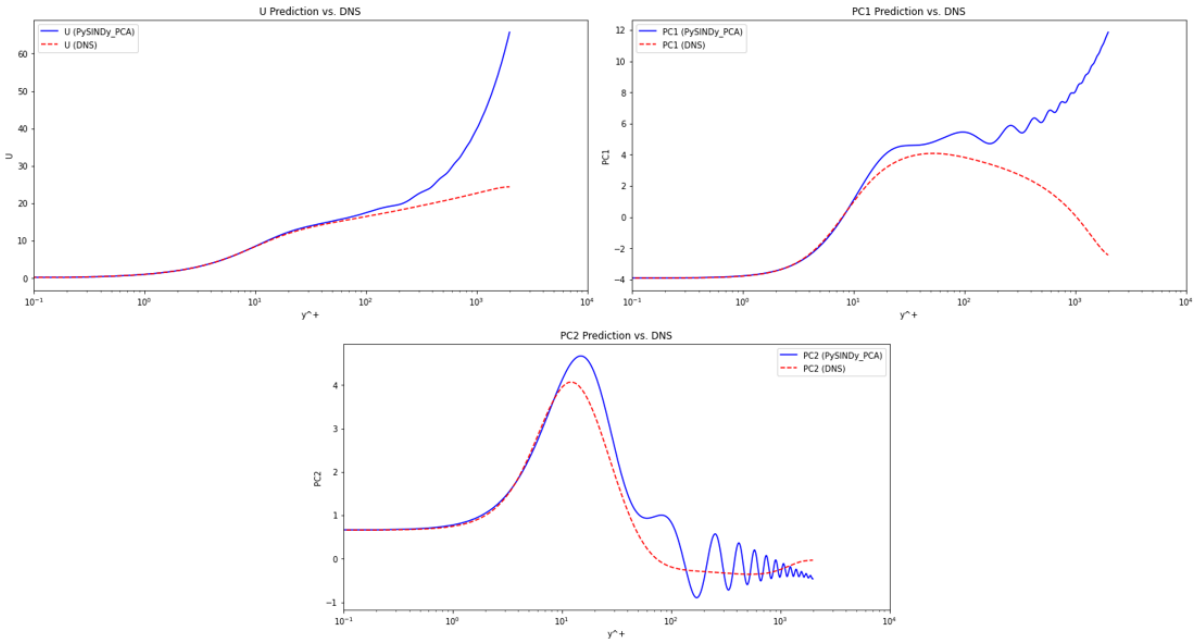


Figure 4.15: Comparison of PySINDy predictions combining PCA and DNS Data (Re = 2000)

From the graphs in figure 4.15, it can be seen that the hybrid PySINDy-PCA model is able to follow the general trend of the DNS data in the near-wall region, but has difficulty in capturing

all the physics and important dynamics of the turbulence as we get closer to the far-wall region, which is reflected in a significant divergence between the DNS data and the data predicted by the PySINDy model combined with PCA. Compared with the original PySINDy model, the observations are close, however the deviation is much more important for the hybrid model. Besides, unlike the original PySINDy model, the hybrid model is not capable to predict the covariances and variances of velocity fluctuations because of the use of PCA. Despite a satisfactory R^2 (0.89) and a competitive MSE (4.47e-6), the PySINDy model enriched with PCA brought no real improvement and failed to completely capture the complex dynamics of turbulent channel flow.

This suggests that the application of PCA in the context of turbulence has introduced a significant loss of physical information. Indeed, the reduction in dimensionality surely results in the omission of important and crucial dynamics captured by the DNS data. This simplification of the PySINDy model using PCA resulted in a less accurate prediction as the y^+ increased. As a result, the use of PySINDy with PCA in this context was excluded.

4.3.3 PySINDy optimised by GA

After generating a set of initial equations from the training of the original PySINDy model (Figure 4.11), we performed the GA as described in our methodology (section 3.3.3). The parameters used for the eaSImple GA function were a crossover probability (cxpb) of 0.5, a mutation probability (mutpb) of 0.2, and a number of generations (ngen) fixed at 50. The revised PySINDy equations, with optimised coefficients, are shown below.

```
PySINDy Model:
(U)' = -0.093 1 + -0.038 U + 0.286 u'u' + 0.022 v'v' + 0.124 w'w' + -1.810 u'v' + 1.636 P + -0.325 dU/dy + 5.268 du'u'/dy + -1.
815 dv'v'/dy + 0.049 dw'w'/dy + 0.326 du'v'/dy + 1.043 dP/dy
(u'u')' = 1.101 1 + 0.037 U + 0.269 u'u' + 0.403 v'v' + -1.875 w'w' + 0.291 u'v' + 0.933 P + 4.105 dU/dy + -0.104 du'u'/dy + -
0.322 dv'v'/dy + 0.073 dw'w'/dy + -0.338 du'v'/dy + 0.217 dP/dy
(v'v')' = -0.474 1 + 0.056 U + 0.144 u'u' + -0.398 v'v' + -0.878 w'w' + -0.926 u'v' + -0.433 P + -3.591 dU/dy + 0.144 du'u'/dy
+ 0.013 dv'v'/dy + -0.987 dw'w'/dy + 1.076 du'v'/dy + -0.373 dP/dy
(w'w')' = -0.288 1 + 0.079 U + -0.067 u'u' + -0.466 v'v' + -0.754 w'w' + 0.267 u'v' + -0.083 P + -1.679 dU/dy + -0.661 du'u'/dy
+ 0.744 dv'v'/dy + -2.097 dw'w'/dy + -0.053 du'v'/dy + -7.845 dP/dy
(u'v')' = -1.567 1 + 0.174 U + -0.144 u'u' + -0.275 v'v' + -0.592 w'w' + 1.275 u'v' + 0.252 P + 1.772 dU/dy + -0.190 du'u'/dy +
0.096 dv'v'/dy + -0.072 dw'w'/dy + 1.780 du'v'/dy + -0.484 dP/dy
(P)' = -0.083 1 + 0.052 U + 0.071 u'u' + 1.629 v'v' + -1.711 w'w' + -0.689 u'v' + -0.169 P + 0.376 dU/dy + 0.473 du'u'/dy + 1.1
59 dv'v'/dy + -1.239 dw'w'/dy + 0.975 du'v'/dy + -0.245 dP/dy
(dU/dy)' = 0.032 1 + -0.002 U + 0.339 u'u' + -0.953 v'v' + 1.555 w'w' + -0.249 u'v' + 3.407 P + 0.005 dU/dy + -1.879 du'u'/dy +
-1.807 dv'v'/dy + 0.420 dw'w'/dy + 0.075 du'v'/dy + 0.021 dP/dy
(du'u'/dy)' = -0.101 1 + -0.072 U + -0.724 u'u' + 1.018 v'v' + 2.366 w'w' + 0.144 u'v' + -0.060 P + -0.769 dU/dy + -1.144 d
u'u'/dy + 0.882 dv'v'/dy + 0.830 dw'w'/dy + 0.154 du'v'/dy + 0.127 dP/dy
(dv'v'/dy)' = 0.215 1 + -0.043 U + 0.004 u'u' + -0.328 v'v' + 0.568 w'w' + 0.618 u'v' + -1.860 P + 0.184 dU/dy + 2.925 du'u'/dy
+ 0.287 dv'v'/dy + 0.694 dw'w'/dy + -0.082 du'v'/dy + 0.279 dP/dy
(dw'w'/dy)' = 0.439 1 + 0.004 U + 0.494 u'u' + -0.737 v'v' + -1.756 w'w' + -1.425 u'v' + -0.273 P + -1.748 dU/dy + -1.130 d
u'u'/dy + 0.107 dv'v'/dy + 0.061 dw'w'/dy + 0.227 du'v'/dy + -0.666 dP/dy
(du'v'/dy)' = 0.263 1 + 0.018 U + -0.018 u'u' + 0.326 v'v' + -0.444 w'w' + 1.457 u'v' + -0.753 P + -0.840 dU/dy + -2.921 du'u'/
dy + 0.405 dv'v'/dy + -2.740 dw'w'/dy + 0.846 du'v'/dy + 0.525 dP/dy
(dP/dy)' = -0.100 1 + -0.051 U + 0.216 u'u' + 0.512 v'v' + 0.348 w'w' + 0.479 u'v' + -0.539 P + 0.476 dU/dy + -0.838 du'u'/dy +
0.090 dv'v'/dy + 0.724 dw'w'/dy + 1.111 du'v'/dy + 0.749 dP/dy
```

Figure 4.16: PySINDy equations with the optimal coefficients obtained by GA

However, as with the integration with PCA, the physical interpretation of the re-tuned equations remains complex, and it is difficult to determine whether they effectively capture the dynamics of turbulent flows. To evaluate these improved equations, we have, as with previous PySINDy models, performed numerical integration to predict velocity as well as the variances and covariances of velocity fluctuations. The graphs below show a comparison between the values predicted by our model combining PySINDy and GA and the actual DNS data, as a function of y^+.

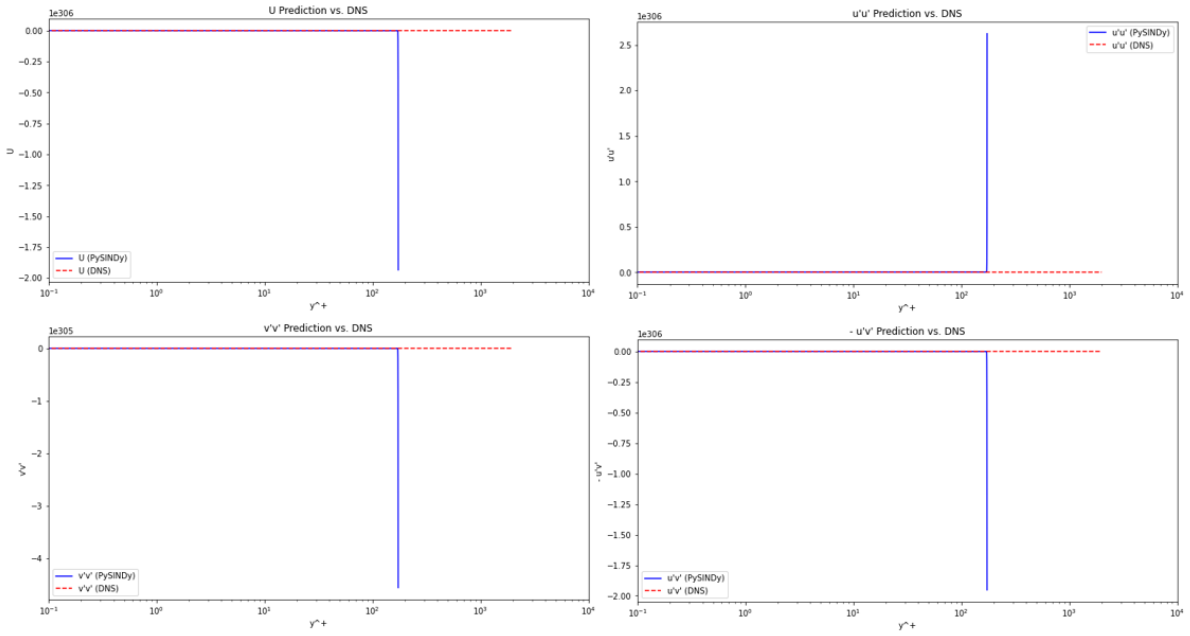


Figure 4.17: Comparison of PySINDy predictions optimised by GA and DNS Data (Re = 2000)

From the graphs in Figure 4.17, it can be seen that there is a considerable gap between the predictions obtained by the hybrid model combining PySINDy and GA and the DNS data. This suggests a complete mismatch between the simulation and the real data. Consequently, it is possible that the fact that the model has difficulty in following the trend observed by the DNS data is because the simulated physical dynamics are incorrect, or because the integration method has not converged. Besides, when focusing on performance metrics, the results are not excellent, if not disastrous, with an MSE value close to 0.83 and an aberrantly low R^2 (-1455131.6). This shows that the hybrid model has considerable limitations when it comes to modeling turbulence. Compared with the original PySINDy model and the hybrid PySINDy model enriched with PCA, this pySINDy approach combined with GA is by far the one that generates the worst model. Moreover, the execution of the GA requires resources and time (450.93 seconds), which can be problematic given that both of the above PySINDy models are particularly fast.

Although the combination of the PySINDy model with GA was able to generate optimised coefficients for the equations, we noted that this optimisation has increased the risk of failure to capture the essential turbulence dynamics. As a result, this optimisation with GA only leads to a loss of physics information, making it inappropriate for turbulence modeling projects. Finally, it was decided to drop this method, as it did not improve the accuracy of predictions and was quite time-consuming.

4.3.4 Conclusion on PySINDy Models

The evaluation of the different models implemented (PySINDy, PySINDy combined with PCA and PySINDy optimised by GA) has revealed the challenges associated with modeling turbulence data. The initial PySINDy model showed very high predictive accuracy compared with DNS data for low values of y^+ for velocity and the covariances and variances of velocity fluctuations. This reflects the model's capability to capture the complex dynamics of turbulent flows. However, when attempting to optimise the model and improve its accuracy, in particular

when the values of y^+ are relatively high, in order to have a reliable PySINDy model, we encountered several problems. Indeed, the integration of PCA, while simplifying the turbulence problem with a very interesting R^2 value and a very good MSE, resulted in a loss of crucial physical information concerning velocity fluctuations and complex turbulence behaviors, causing the model to become much less accurate and deviate much further from the DNS data. In addition, the approach of combining the original PySINDy model with a GA by optimising the coefficients of the set of equations in the model was not an effective method either. Indeed, this method showed a significant gap between the DNS data and the predicted values of the model, underlining its limits in capturing turbulent behavior.

The exploration of different methods has proven that the integration of additional methods such as PCA or GA in order to develop a hybrid model with PySINDy demands more attention in order to minimise the loss of physics information. At the end, the initial PySINDy model which achieves such accuracy in predictions compared with DNS data highly suggests that PySINDy can be a useful tool in fluid dynamics research, providing an effective alternative to more traditional methods, without compromising the accuracy needed in order to capture complex turbulent behavior.

4.4 Transformer

4.4.1 Model Performance Evaluation

Upon evaluating the Transformer model's performance for the regression task, we observed a test loss of 0.042. This result, while not optimal, indicates that the model has learned to some extent from the training data, as evidenced by a moderate level of predictive accuracy.

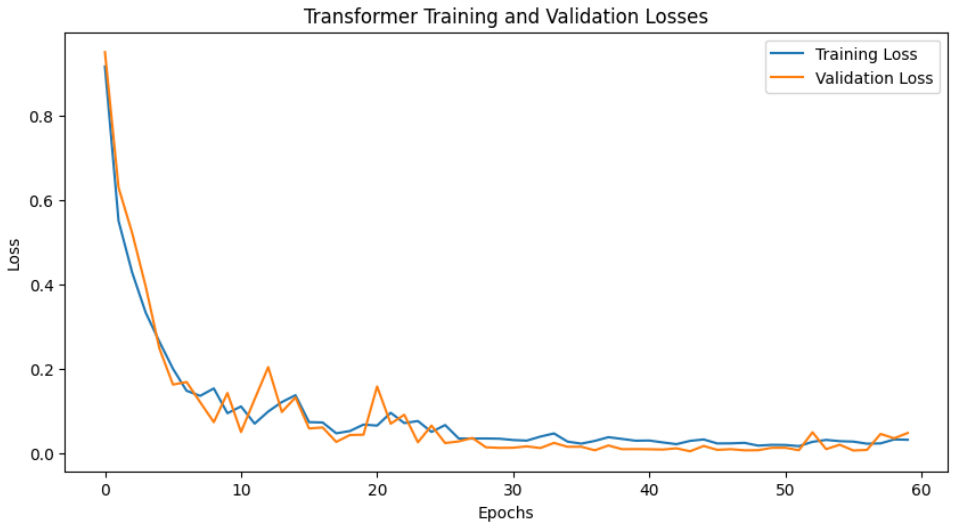


Figure 4.18: Training and Validation Loss curves over epochs.

Figure 4.18 demonstrates the training and validation loss trends. The training loss decreases over time, suggesting learning, whereas the validation loss exhibits volatility, which raises concerns about the model's consistency and generalisation. Nonetheless, the Transformer model achieved a test loss that deems it a moderately effective tool for the problem at hand.

4.4.2 Comparative Analysis with PINNs

However, in comparison, the PINNs model implemented in this study has shown superior performance for this regression task. PINNs integrate the governing physical laws into the learning process, which is particularly advantageous for the dataset under investigation, characterised by complex fluid dynamics phenomena.
The superiority of PINNs in this context can be attributed to the following:

1. **Incorporation of Physical Principles:** PINNs leverage known physical laws as regularisation during training, ensuring that the model predictions adhere to physical constraints, which is essential in fluid dynamics.

2. **Data Efficiency:** By utilising physical laws, PINNs can achieve higher accuracy with less data, an important feature given the dataset size.

3. **Generalisation:** The adherence to physical laws enables PINNs to generalise better to unseen scenarios, as they are not solely reliant on data-driven patterns that may not capture all underlying physics.

4. **Interpretability:** The structure of PINNs facilitates the interpretation of the model's behavior in terms of physical laws, which is beneficial for validation and trust in model predictions.

4.4.3 Conclusion on Transformer

While the Transformer model achieved a test loss of 0.042, indicating a moderate level of accuracy, the PINNs model outperformed it by effectively utilising domain knowledge of the underlying physics. The ability of PINNs to incorporate and enforce physical laws within the training process presents a significant advantage for regression problems in fluid dynamics and similar fields where physical understanding is paramount.

Consequently, despite the reasonable performance of the Transformer model, the results advocate for the use of PINNs in scenarios where the integration of physical laws can significantly enhance model performance and reliability.

4.5 Comparisons of ML and RANS Models with DNS

We performed an in-depth and detailed comparison between the various models implemented and simulated, and the DNS data that will serve as a reference. In the context of turbulence modeling, the accuracy of different models in predicting turbulent flows is critical. Our comparative analysis is focused on evaluating the performance and accuracy of four different approaches: the PINNs model, the PySINDy model, the k-ω turbulence model and the RSM.

Each model offers its unique perspectives and methodologies for capturing the complex turbulent flow patterns. By performing a comparison of these different models against DNS data, our objective is to evaluate their predictive accuracy and to identify their main strengths as well as their limitations. We have decided to compare them with DNS data, generally viewed as the standard for fluid simulations, as they offer an accurate reference for assessing the fidelity of

these models. The graphs below illustrate the results of the predictions of each model compared with the DNS data.

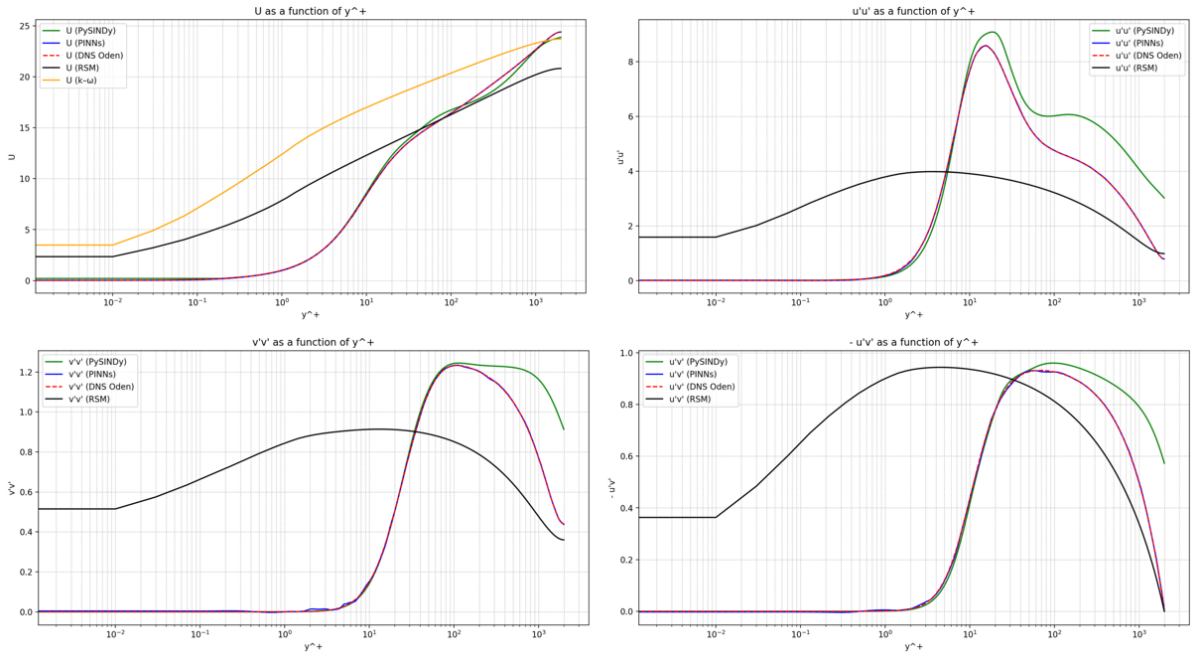


Figure 4.19: Comparison between PINNs, PySINDy, k-ω, and RSM Model Simulations against DNS Data (Re = 2000)

1. **k-ω Model:** From the literature review, it is known that the k-ω model is known for its high accuracy in velocity prediction. As shown in the figure 4.19, the simulation of the model confirms this observation, with a prediction fairly close to the DNS data near the walls, where turbulence dynamics are strongest. Moreover, the model converges rapidly towards the maximum velocity profile, which demonstrates its effectiveness. However, the model has significant limitations in terms of the accuracy of its predictions of the covariances and variances of velocity fluctuations.

2. **RSM:** From the literature review, it is known that RSM is able to directly compute the covariances and variances of velocity fluctuations. As we can see in figure 4.19, for the different velocity covariances, RSM succeeds in reproducing the trend in DNS data, despite some deviations. Besides, RSM is known to be accurate in describing transitions from turbulent to laminar flow, making it particularly suitable for high-fidelity simulations requiring in-depth analysis. However, the model has considerable limits in terms of computational intensity, requiring more computational resources, making it difficult to use in large-scale applications.

3. **PySINDy Model and PINNs Model:** As shown in figure 4.19 , these two approaches have delivered very promising results which can rival traditional models such as DNS. Indeed, these two models not only provide an accurate analysis of fluid dynamics, but also greatly reduce computational costs in comparison with other traditional methods such as RSM. These models can provide an effective alternative for a wide range of turbulence simulation applications.

In summary, although traditional models such as k-ω and RSM are still quite accurate in specific scenarios, with the emergence of machine learning methods such as PySINDy or PINNs

in turbulence modeling, we can expect to see a major evolution in this domain. Indeed, these innovative models are not only highly accurate, they also deliver other advantages in terms of relatively short execution times. As the CFD field is continuing evolving, these tools should play a key role in improving our understanding in the behavior of complex fluids.

Chapter 5 – Software Development

5.1 User Requirements

Firstly, the group conducted a market survey on the potential customers and their needs. The two major stakeholders are the aerospace and automotive industries. See Appendix 1 (User Requirements) for more details.

Secondly, the group discussed the data requirements, model functionality, performance, and system requirements.

The cardinal and non-cardinal requirements were then developed as elaborated below. The group decided to set a challenge of developing a simple cloud-based application after the development of desktop application within the two-months period.

5.1.1 Cardinal Requirements (Must-have)

- **Accuracy.** The software must achieve a minimum MSE of 0.01 for accuracy, as defined by domain-specific benchmarks.
- **Scalability.** The software must be able to process datasets of varying sizes (small scale and large scale $< 200MB$) and complexity without significant degradation in performance.
- **Compliance with Regulations.** The software must adhere to all relevant data privacy laws and industry regulations such as ITAR when handling sensitive flight data.
- **Automated Data Pre-Processing.** The software must have an automated pipeline for data cleaning and preparation. It must also be able to detect and handle outliers or missing values in the input data.
- **User Friendly Interface.** The software must be easy to use with a simple interface for interacting with the model, features are accessible to the users (engineers and analysts).
- **Computational Time.** The software must be able to generate the results not more than 10 minutes with $< 200MB$ of input data.
- **Alerts and Errors.** The software must prompt the user if incorrect files or parameters are input into the system.

5.1.2 Non-cardinal Requirements (Good-to-have)

- **Enhanced User Experience.** The software can be designed with interactive background and features for better user experience.
- **Enhanced Features.** The software can be designed to predict more than just the Reynolds stresses, such as heat transfer or acoustic noise generation, which may be useful in some applications.

- **Customisability.** The software can be customisable for the model, or its outputs based on user preferences or specific scenarios.
- **Integration Capability.** The software can be integrated with the existing software used by the customer for more holistic prediction of RANS in the simulations.

5.2 Basic Software Architecture

The group designed a simple achievable software architecture to meet the project timeline. It consists of two parts, the presentation layer and the application layer.

Presentation layer which is the user interface of the application where users interact with the app. It includes everything the user sees and interacts with. Potential technologies are HTML, CSS and JavaScript.

Application Layer (Server Side) handles the business logic, data validation, and session management of the application. Potential programming languages and framework are Node.js with Express, Python with Streamlit, Django, or Ruby. With the template codes available on GitHub, the group adopted Python with Streamlit.

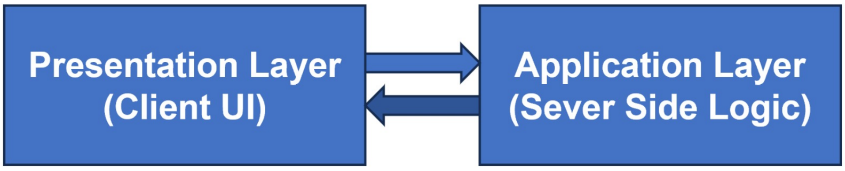


Figure 5.1: General software architecture design

5.3 User Interface (UI) Implementation

The prototype of this project takes the form of a Streamlit application in which all the models and methods involved are packed in. The Streamlit application utilises a structured and modular architecture to facilitate ease of development, maintenance, and user interaction. The `app.py` is the main entry point of our application. It utilises the sidebar menu of the Streamlit interface and manages the user journey to the different functional modules. These modules are organised under the `utils` directory, which further contains three subdirectories, one corresponding to each component of the application, i.e., `pysindy`, `pinns`, and `comparison`. For each subdirectory, we have one script in Streamlit that consumes the module of the same name in Python. This approach not only simplifies development but also caters for later scalability of the application.

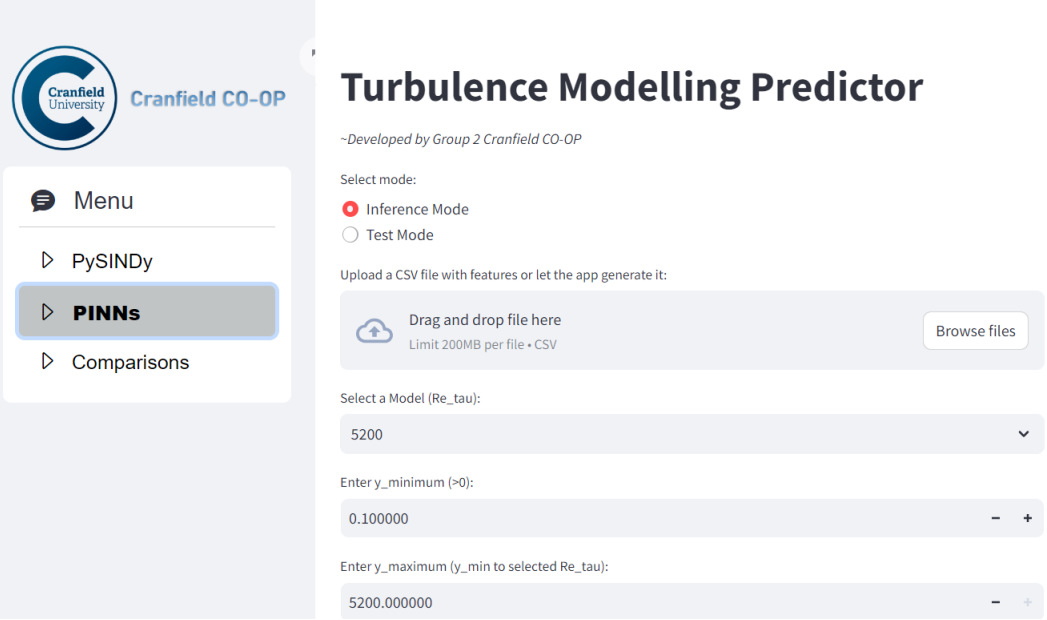


Figure 5.2: Screenshot of Streamlit UI

5.3.1 app.py

The file `app.py` is the entry point of our application that uses the library Streamlit to generate a web interface. The users can enjoy the different modules: PySINDy, PINNs, Comparisons in the navigation sidebar menu. In practice, each module is developed as a separate Streamlit app that is contained in the main app by a class called `MultiApp` which plays the role of handling the routing between the sub-apps. The sidebar allows navigating between the predictive models and comparisons, and it loads the correct content based on the user-selection. Besides, it improves the user interface by adding a logo in the sidebar.

5.3.2 pysindy_streamlit.py

This script is designed for the PySINDy model to predict fluid dynamics. It identifies governing differential equations from data. Users can upload a CSV file and input noise if required. Additionally, the user can train the PySINDy model and view the raw data or the model's output equations. The script is capable to depict performance metrics, including the execution time, the R^2, and the MSE. The user can also simulate the model's predictions and download the results.

5.3.3 pinns_streamlit.py

For the PINNs predictions, this script is for handling PINNs predictions. The user can choose either they want to predict the data Inference Mode or test the model with a known target Test Mode. They can upload a file or scrape the data.

5.3.4 comparisons_streamlit.py

This module enables users to compare results from different models and techniques, including PySINDy, PINNs, DNS, RSM, and others. Users can select which models' results to compare

and depending on the choice, upload or interact with different datasets. The module provides visual comparisons and statistical analysis of the selected datasets.

5.4 Cloud-based Software Architecture

The prototype is deployed in the cloud with several AWS services, proving that it can be used without huge and expensive machines. However, it comes with limitations.

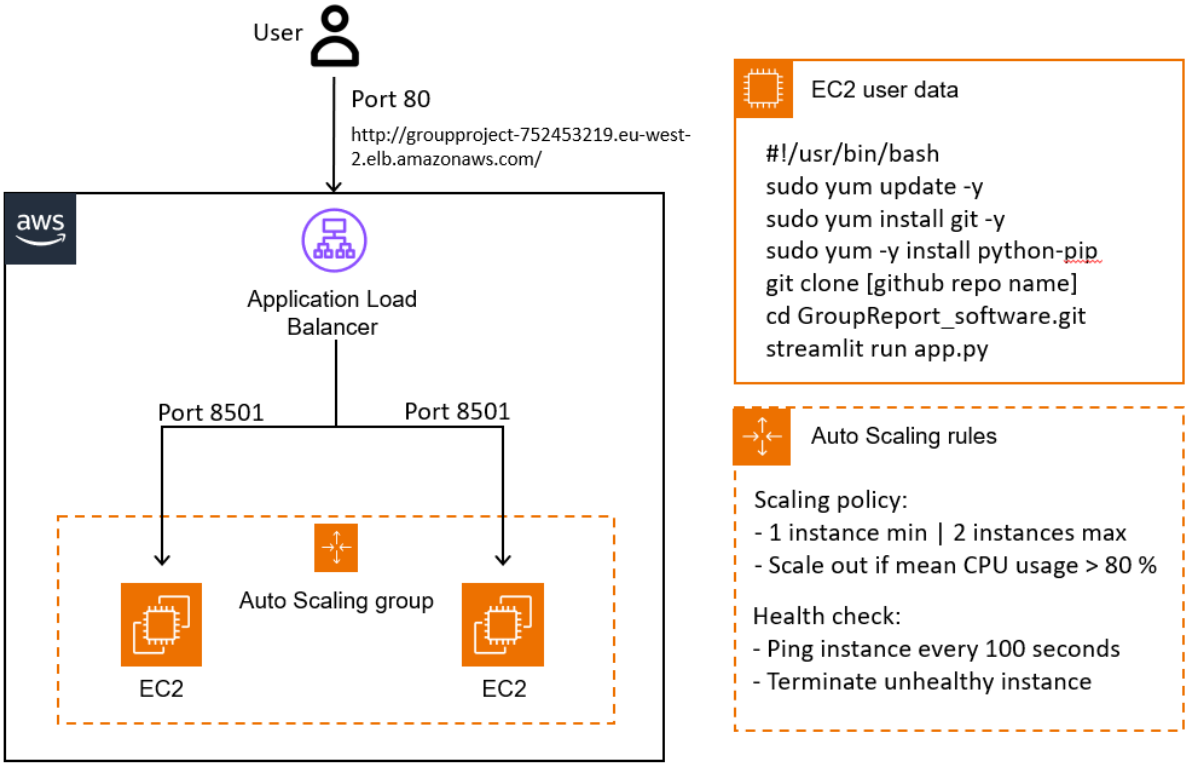


Figure 5.3: AWS-based software architecture

Users interact with the application through a browser, making requests to a specified URL served by the AWS Application Load Balancer (ALB). The ALB listens on port 8501 and efficiently distributes incoming traffic across multiple EC2 instances across different Availability Zones, enhancing the application's fault tolerance. These EC2 instances are members of an Auto Scaling group which is set to automatically increase the number of EC2 instances to respond to the load on the application.

The group automatically scales between a minimum of one instance and a maximum of two. Additional instances are created as soon as the CPU is used at more than 80% which disappears when the usage is less than indicated. The Auto Scaling group performs a regular 100-second health check on each instance, terminating it when it does not respond within the appropriate time to prevent the occurrence of stdout issues. User data scripts function as an entry point in this process and serve to conduct multiple initialisation tasks that are started at the execution time of an EC2 instance.

The installation process also involves some operations like performing system updates, installation of Git and Python pip (if needed), cloning the GitHub repository and finally calling the

Streamlit application by running `app.py`. This is fully automatic deployment that guarantees the desktop having every distant configured similarly and therefore it is ready to be served without manual input.

5.5 Testing Case

When designing an application, it is essential to integrate a number of tests to verify that the application; meets the different technical requirements and expectations of users. The testing phase is necessary to ensure the application can run correctly in any types of situation, before deployed in the market. This section of the report will focus on the evaluation of the application against four fundamental performance axes in turbulence modelling: accuracy, speed, scalability and reliability. These tests are designed to test the application's performance from different angles, according to the instructions in the RCTM plan (3.1,3.2,3.3).

It should be noted that the tests below will be carried out on the local application in order to closely monitor the performance and responsiveness of the model. Besides, the results of the PINNs model for the different tests will correspond to the results of the test mode of the PINNs module of the application.

5.5.1 Accuracy

As part of the development of our application, the accuracy is an essential criterion for the evaluation of any turbulence modeling software. The accuracy test is designed to guarantee that the models in the application can accurately predict the different components of the RST under specific channel flow conditions. The capacity to model turbulence correctly is fundamental for the reliability of the results of our application.

To show that the application can reach a high level of correlation with actual experimental results, PySINDy and PINNs models were tested with various datasets: the original DNS data as well as the DNS data augmented with the interpolation method for different quantities. For each of the tests performed, both models were required to obtain results with an MSE of less than 0.01, a threshold determined to indicate suitable accuracy within the scope of our application.

- **PySINDy:** Table 5.1 shows the results of accuracy tests for the PySINDy model with three different datasets, with and without interpolation.These results show that the PySINDy model offers exceptional accuracy with all MSEs well below the 0.01 threshold for each test, indicating an ability to model turbulence with high fidelity to DNS data.

Case Number	Interpolation	MSE	Success
1	No	0.0000689894	Yes
2	5000	0.0000049679	Yes
3	10000	0.0000047228	Yes

Table 5.1: Results of the accuracy test for the PySINDy model of the application

- **PINNs:** Table 5.2 shows the results of accuracy tests for the PINNs model with three different datasets, with and without interpolation. These results highlight that the PINNs model offers even more outstanding accuracy than the PySINDy model, with all the MSEs well below the 0.01 threshold for each test, also indicating an ability to easily capture turbulence dynamics.

Case Number	Interpolation	MSE	Success
1	No	0.0000119	Yes
2	5000	0,00000509	Yes
3	10000	0,00000509	Yes

Table 5.2: Results of the accuracy test for the PINNs model of the application

The accuracy tests have demonstrated that the application is adept at modelling turbulence accurately using both PySINDy and PINNs models. This meets the stringent criteria set for prediction accuracy. As a result, users can confidently rely on the results obtained by our applications for analysing and comparisons between different models.

5.5.2 Speed

Execution speed is another important component test of the efficiency of a software to compute the complex and intensive calculations. The speed test is designed to verify that the turbulence models (PySINDy and PINNs) are capable of providing accurate predictions of the RST within a reasonable time. Both PySINDy and PINNs models were tested with three different dataset sizes taken from DNS data. In each test, the models were required to produce accurate results with a delay of less than 10 minutes, a threshold that indicates that the application is sufficiently fast.

- **PySINDy:** Table 5.3 shows the results of speed tests for the PySINDy model with three different dataset sizes. These results showed that PySINDy model is capable of offering accurate results extremely quickly for each test, regardless of the input data, while fully meeting the performance requirements for execution time.

Case Number	Input Data Size	Execution Time	Success
1	10.5 MB	0.16 seconds	Yes
2	104.6 MB	1.50 seconds	Yes
3	198.8 MB	4.80 seconds	Yes

Table 5.3: Results of the speed test for the PySINDy model of the application

- **PINNs:** Table 5.4 shows the results of speed tests for the PINNS model with three different dataset sizes. These results proved that even though the PINNs model is slightly slower in producing accurate results compared to the PySINDy model, the execution time remains relatively quick for each test conducted, meeting the performance requirements for execution time.

Case Number	Input Data Size	Execution Time	Success
1	10.5 MB	34.28 seconds	Yes
2	104.6 MB	148.94 seconds	Yes
3	198.8 MB	240.60 seconds	Yes

Table 5.4: Results of the speed test for the PINNs model of the application

To conclude, the speed test proves that the application is capable of delivering accurate and fast predictions of the RST within the user requirements. The performance of both PINNs and PySINDy models shows that the application can effectively manage the prediction of turbulence dynamics independently of the input data. Moreover, this kind of performance is important for applications that require fast execution times for data analysis and decision-making processes, as well as for users who depend on just-in-time data processing for fluid dynamics research or industrial applications.

5.5.3 Scalability

The ability to efficiently manage different volumes of data is also an essential performance criterion. This scalability test is designed to evaluate the application when confronted with both small-scale and large-scale datasets (up to 200 MB), and to ensure that it always maintains an optimal performance without any significant degradation in prediction accuracy. To explore the ability of the application to efficiently handle CSV files to run turbulence models, the interpolation method was employed to generate CSV files of varying sizes.

For the results obtained for PySINDy, different hyperparameter values were used each time in order to have the best possible model associated with the input data.

- **PySINDy:** Table 5.5 shows the results of scalability tests for the PySINDy model with three different dataset sizes. The results indicate that the PySINDy model of the application manages to handle the increase in data size efficiently, by generating accurate predictions and by capturing the key turbulent flows, with an execution time well below the 10-minute limit and an MSE always below 0.01 for each case.

Case Number	Input Data Size	MSE	Execution Time	Success
1	10,5 MB	0.0000045599	0.16 seconds	Yes
2	104,6 MB	0.0001517711	1.50 seconds	Yes
3	198.8 MB	0.0002946088	4.80 seconds	Yes

Table 5.5: Results of the scalability test for the PySINDy model of the application

- **PINNs:** Table 5.6 shows the results of scalability tests for the PINNs model with three different dataset sizes. Even though the execution time of PINNs is significantly longer than the one of PySINDy in each case, it is still well below the acceptable limits (10 minutes). Moreover, the extremely low MSE suggests that the accuracy of the PINNs model is not affected by the scale of the data.

Case Number	Input Data Size	MSE	Execution Time	Success
1	10,5 MB	0.00000509	34.28 seconds	Yes
2	104,6 MB	0.00000509	148.94 seconds	Yes
3	198.8 MB	0.00000509	240.60 seconds	Yes

Table 5.6: Results of the scalability test for the PINNs model of the application

In addition, we have also tested the application when uploading a CSV file exceeding 200 MB for both the PySINDy and PINNs models, in order to check how the application handles this problematic situation. The results of this test for the PySINDy and PINNs models can be seen in Figure 5.4 .

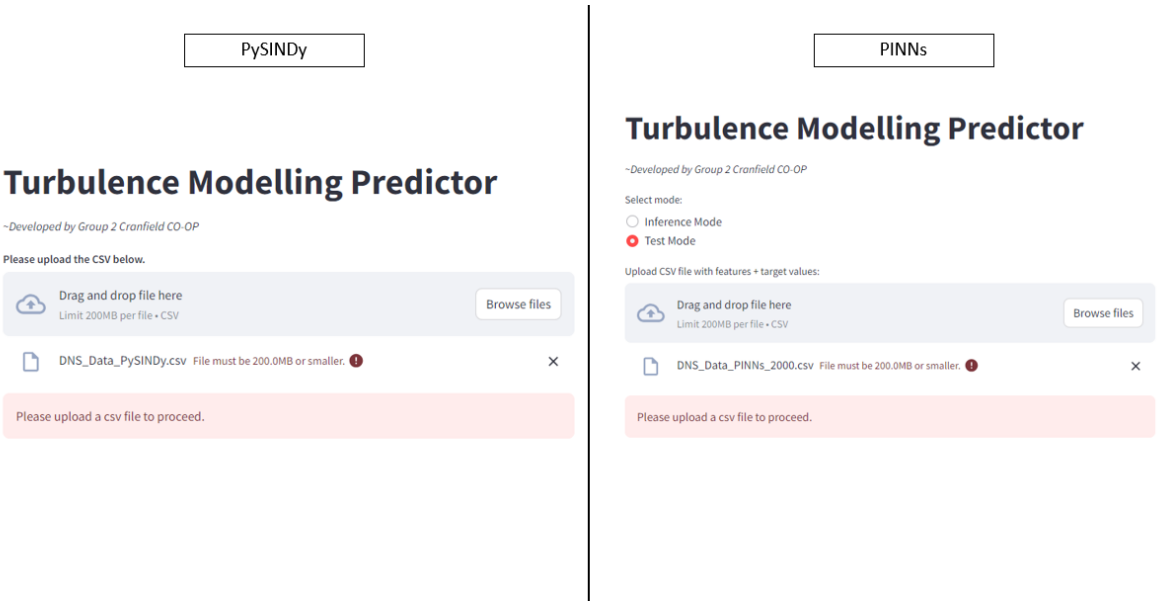


Figure 5.4: Streamlit App Scalability Test

Figure 5.4 demonstrates that the application returns a message, to inform the user that the file size is larger than the maximum capacity that can be processed by the software. Next, it will encourage the user to submit a CSV file of less than 200 MB. To sum up, the scalability test proves that this application is able to manage datasets of various sizes efficiently and in a reliable way, respecting the time and accuracy requirements for both models.

5.5.4 Reliability

Lastly, the software's reliability, particularly in the field of turbulence modeling, is fundamental to guaranteeing user confidence regarding the results obtained. This reliability test is designed to verify the ability of the application to provide consistent and reproducible results from the same input data. This criterion is essential and very useful for validating the correctness of the two models used (PySINDy and PINNs).

To conduct the reliability test, first, we have to run the PySINDy and PINNs models five times using the same DNS dataset. Next, we will evaluate the results of these runs by observing at the performance metrics (R^2 and MSE), to determine if both turbulence models produced similar results for each run.

- **PySINDy:** Table 5.7 shows the results of reliability tests for the PySINDy model. These results show that PySINDy offers a remarkable consistency in the results obtained across five successive tests, demonstrating the reliability of the model. Indeed, the R^2 remained consistent across all tests, as well as the MSE. Only the execution time of the model varied widely, yet it remained extremely fast for each test performed.

Case Number	R-Squared	MSE	Execution Time	Success
1	0.9068654467	0.0000045599	0.28 seconds	Yes
2	0.9068654467	0.0000045599	0.18 seconds	Yes
3	0.9068654467	0.0000045599	0.09 seconds	Yes
4	0.9068654467	0.0000045599	0.12 seconds	Yes
5	0.9068654467	0.0000045599	0.16 seconds	Yes

Table 5.7: Results of the reliability test for the PySINDy model of the application

- **PINNs:** Table 5.8 shows the results of reliability tests for the PINNs model. These results indicate that the PINNs model offers high reliability with consistent R^2 and MSE scores across all the executions. Similar to the PySINDy model, the execution time for the PINNs model varies for each run yet extremely fast.

Case Number	R-Squared	MSE	Execution Time	Success
1	1.00	0.0000119	3.14 seconds	Yes
2	1.00	0.0000119	0.41 seconds	Yes
3	1.00	0.0000119	0.62 seconds	Yes
4	1.00	0.0000119	0.23 seconds	Yes
5	1.00	0.0000119	0.24 seconds	Yes

Table 5.8: Results of the reliability test for the PINNs model of the application

In summary, the reliability test successfully demonstrated that the application is reliable for modeling turbulence using PySINDy and PINNs models. Indeed, both models of the application have met the performance requirements in terms of reliability, with consistent results across the different executions, which confirms the stability of the application.

Chapter 6 – Marketing Strategy Overview

6.1 Marketing Background

Embracing the rising wave of technological innovation, the global CFD market is undergoing a remarkable transformation. Recent data reports have shown significant growth and broadening applications across various industries. According to a latest report, the global CFD market is expected to reach USD 1,875 million in 2020 and is predicted to rise to USD 3,497.9 million by 2027, exhibiting a compound annual growth rate (CAGR) of 9.3% from 2020 to 2027 [1] as shown below.

Another report reveals that the CFD market reached a size of USD 2,520 million in 2023 and is expected to grow at a CAGR of 11.7%, predicted to reach a size of USD 6,830 million by 2032 [2]. The growth is affected by several factors, for instance, the rise of cloud computing, widen the availability and usage of CFD simulation. Increased demand for high-fidelity simulations in industries like automotive, aerospace, and national defence result in this market expansion [3].

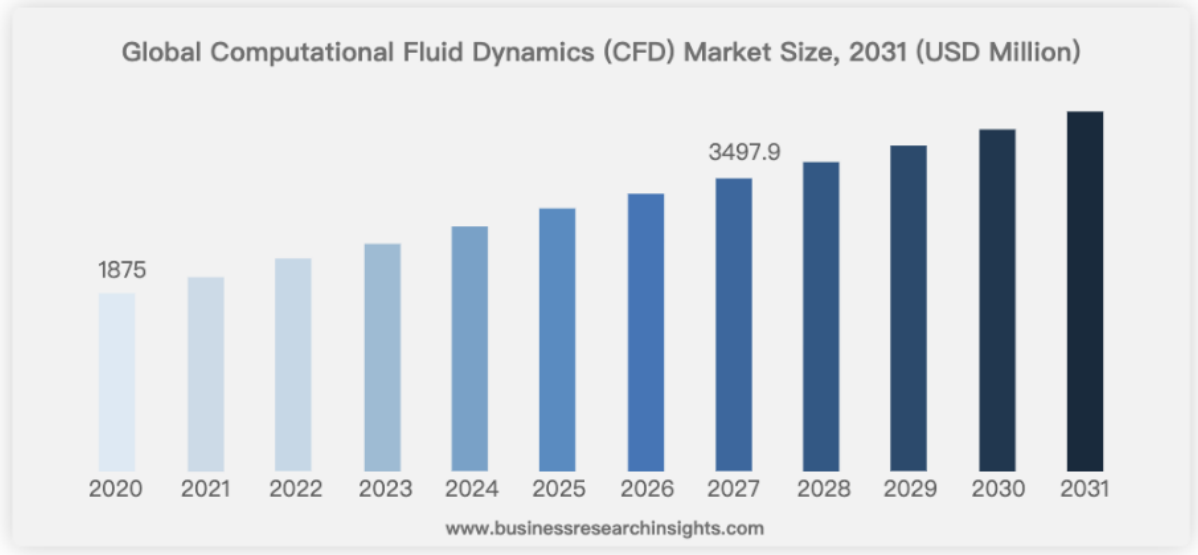


Figure 6.1: The trend of CFD market

This trend in the CFD [1] market demonstrates the wide application and significant growth potential. Based on the usage of CFD, the potential customer base includes researchers, engineers and academic institutions at the forefront of technology. The developed software provides a cost-effective solution that can effectively assist them in overcoming the challenges in their designs and analysis processes, thereby increasing the penetration of our brand in the market. In addition, our software is designed for both professionals and new users who do not possess deep understanding of physical principles. The user-friendly interface allows the analysis of turbulence models easily.

6.2 Competitors Analysis

1. **Wolf Dynamics:** This website is committed to use open-source applications, and their core business are consultation and Research-Innovation services in CFD. Besides, it offers training services and software support to assist users to overcome steep learning curve inherent to these tools.

2. **CFD-Online:** CFD Online is an online center for CFD and has been in service for 30 years. The services provided include a free CFD reference in the form of a CFD Wiki, several discussion forums, etc. They have linked with more than 3,400 websites around the world.

6.3 Unique Selling Proposition (USP)

Both Wolf Dynamics and CFD-Online are free to use websites, but only allow users to generate result for a single point. Comparing to them, our software provides similar functionality with option to select paid model. It comes with added features such as generating physical equations, simulating results, graphs, and the ability to compare results between models for cross checking and boosting confident in their results. Our UI is relatively user-friendly to improve the ease of learning for new users without in-depth background. Lastly, the current website application can be made into an offline application to allow users to use anywhere, even in cellular denied environment.

6.4 Marketing Mix

Marketing Mix	**Cranfield Coop**	**Wolf dynamics**	**CFD-Online**
Price	- Free (Predict one point value) - Paid (Physical equations, simulated results, graphs)	- Free	- Free
Product	- Website - App	- Website	- Website
Place	- Online or Offline mode	- Online	- Online
Additional Features	- CSV input file - Predict full flow - Provide dynamic equations - Multiple simulation model - Provide comparisons - Output CSV values	- Manual input - Predict specific flow value - Does not provide dynamic equations - Single simulation model - Does not provide comparisons - Only output value	

Table 6.1: Marketing Mix

6.5 Strategic Planning

In our strategic planning, we aim to capitalise on the growth potential of the CFD market by leveraging insights and market growth trends. The objective is to establish Cranfield CO-OP as the to-go-to tool among a diverse database consisting of researchers, engineers, data scientists and newcomers who require sophisticated simulation and analysis tools.

To meet the varied demands, Cranfield CO-OP offers two-tier pricing options and ensure seamless accessibility in online and offline modes.The software is deisgned to be intuitive, catering to both experienced and newcomers, facilitating a smoother usability.

We will employ targeted marketing campaigns. These will include case studies that demonstrate real-world applications and benefits of our software, user testimonials to build trust, credibility, and interactive demos that allow potential users to experience the functionality and user-friendliness of Cranfield CO-OP firsthand. Partnership with universities and companies can also further Widen the reach of our product into the market.

Through this strategic approaches, Cranfield CO-OP aims to participate and actively contribute to the growth of the CFD market. Providing users with powerful tools to advance their work, we support the broader evolution and application of CFD, driving innovation and efficiency in the field.

Chapter 7 – Project Plan

7.1 Group Dynamics

The team of 7 members is formed with a mixture of different expertise within CSTE (CIDA, CED and SETC). The team conducted two tests, Technical Skills and Myers-Briggs Type Indicator (MBTI) to identify the weaknesses and strengths of the group. The results are computed as shown below. It is observed that the group has a good mixture of skillsets, those that are rated 5 and above are the strength while those scored below 5 are area of improvement for the group. They are the GUI Design, Poster Design and Testing. Deliberate actions are taken such as distributing these tasks to the members equally, to seek more insights and feedback for each task. Overall, the group is assessed to meet the requirements to commence this project.

Name	MBTI Result	Responding to email	Suggesting Ideas	Completing Tasks	Helping Others	Planning & organisation	Design of system	Documentation	Organising meetings	Leadership	Presenting	Programming	GUI Design	Poster Design	Testing	Commenting Code	Conflict Resolution	Professional Behaviour
Alexis	ENFJ	1	2	2	1	1	0	2	2	2	1	2	1	0	-1	2	2	2
Deng Chuan	ENTJ / INTJ	1	1	1	2	2	1	1	2	1	1	1	0	2	1	1	1	1
Chung Yueh	ENTJ	1	1	1	1	1	0	0	1	1	1	1	0	-1	1	1	1	1
Julien	ESTJ	2	1	1	1	1	0	-1	2	1	1	2	1	-1	1	1	1	1
Lyderic	ENTJ	2	2	1	2	1	2	2	1	1	2	-1	0	1	-1	-1	2	1
Majuran	ESTJ	1	1	2	1	2	0	1	1	1	1	2	-1	0	1	1	1	1
Marcell	ENFJ	2	2	2	2	2	2	2	2	2	2	1	2	2	1	2	2	2
	Total	10.0	10.0	10.0	10.0	10.0	5.0	7.0	11.0	9.0	9.0	8.0	3.0	3.0	3.0	7.0	10.0	9.0

2 Primary
1 Secondary
-1 Weakness
0 I have not practice yet

Figure 7.1: Technical Skill Results

7.2 Roles & Responsibilities

After reviewing the compiled results, the group enumerated a list of tasking and discussed the roles and responsibilities of each member. It is agreed upon that anyone can assume any tasks to acquire new hard or soft skills during this project work. The CED student will take on CFD related tasking such as simulations using ANSYS/ ICEM software. The SETC student will take on software engineering related tasking such as quality plans. The remaining outstanding tasks are distributed to the CIDA students. These are the some of the identified roles and responsibilities.

- Secretary: Arrange for meetings, prepare agenda and minutes.
- Facilitator: Facilitate discussion and keeping track of time.
- Project Manager: Steer the direction of the group to achieve the goals. Generate weekly tasking and take on any adhoc administrative works.
- CED Experts: Generate simulations and results
- CIDA Experts: Develop ML algorithms (PySINDy, PINNS and Transformer)
- SETC Experts: Develop testing plans, documentation and processes.

7.3 Challenges

There were challenges during the period of the group project.

- **Communication Issues.** It is a common behaviour and can occur in new or seasoned groups due to lack of communications or miscommunications. It can lead to conflicts and dampened the morale of the team. This is the result of individual assumptions without clarifications with the others. Sometimes, members are uncontactable. The solution is to improve the team's communications by regular updates and seek clarifications when in doubts. More regular meetings and sub team discussions are organised.

- **Fail to meet task's deadline.** It is important to meet the deadlines set within the group as delaying the task completion may lead to delaying the progress of the project. Individual has to set realistic attainable goals. When faced with complex tasks, the team divided them into sub tasks and distributed out to improve the efficiency and hasten completion time.

- **Diverse Working Style.** Each member has a different background, experience and style of working. The team has to recognise them and adapt accordingly, otherwise, it may lead to friction and inefficiency. An example is that the bi weekly meetings are scheduled after working hours to allow part time student to join the meetings.

7.4 Risk Register

The project commences by asking direct plain language questions to identify and document delivery risks. Red-amber-green (RAG) were used to understand the probability, severity and the impact of those risks. Mitigation actions were identified shortly after that. Risks were monitored especially if those risks could not be mitigated. For the Risk Register please refer to Figure 3.1 in Appendix 3.1.

7.5 Branching Models

Git-flow branching model is applied throughout the software development. For more details please refer to Appendix 3.2.

7.6 Quality Plan

7.6.1 Introduction

Purpose of the Document & Audience

This quality plan is prepared to present how testing is organised, conducted and managed during the group project for the team. The quality plan with complementary documentation also provides enough information to enable external groups to repeat and reproduce the scientific experiment. This document is targeted at all parties involved in the requirement definition, development, testing and sign-off for production release within the group project.

Background & Objectives

The primary aim of the group project is deploy machine learning algorithm to predict the RST in a specific channel flow. The secondary aim of the group project is to develop a marketable software.

The quality plan is tailored to support both aims. The chapters of the quality plan specifies two different strategies of testing: the software engineering and machine learning. Machine learning development process and test strategy focus on the early research and discovery track of the project while the software engineering process and test strategy focus on the delivery track. This a dual-track agile methodology that is suitable and effective methodology for research and design focused projects.

7.6.2 Testing Scope

7.6.2.1 Platform, Applications and Services

Please refer to chapter 5.4.

7.6.2.2 Operational Policies & Procedures

Reproducibility

The group project targets "Results Reproduced" blue badge certification defined by Association for Computing Machinery. Please refer to section 3.2.7.3 and Appendix 3.3 for the approach and the list of tools used to ensure the research findings can be reproduced.

Development and Source Code Management

The application is developed using secure coding practices. The application developed and deployed according to a continuous delivery model.

Source code development is conducted against a distributed version control system with branch and merge management underpinned by manual code review processes. (Continuous testing and packaging pipeline ensures that builds are automatically documented, tested and checked for security, dependency implications before being integrated, packaged and deployed to an initial staging environment.)

Deployments to development (aka alpha), user acceptance test (beta) and production environments are managed by a gated review process including manual steps to approve release candidates for deployment to the next environment.

Machine Learning Development Guidelines

A robust end-to-end prescriptive machine learning development process is provided with industry best practices [34] to support achieving both aims of the group project. The guideline consist of a collection of high-level task items, checklist and consideration at the start and end of a sprint to define items in the development backlog. These tasks items and checklist are

allocated on a separate Kanban board. This can help to decompose high level tasks into sub-tasks and to prioritise and place items on the weekly backlog. (In agile terms these are potential spikes to solve a problem). At the end of each sprint, any sub-task item can be measured against the given checklists. For the checklist see Appendix 3.4.

User Manuals

Read me has been provided as part of the technical delivery.

Standards, Technical & Regulatory Requirements

UK-GDPR (Data Privacy) While the application does not hold any personal information about its users, GDPR requirements must remain at the forefront of testers minds to ensure that no changes have changed that position [37].

7.6.2.3 Test Deliverables

- Test Plan (7.6.5), Requirements Catalogue Traceability Matrix (3.7), Test Execution Log (section 5.5)

7.6.2.4 Exclusions

The testing will not include validation of third party systems and libraries.

7.6.2.5 Quality Risks

For the Risk Register refer to Figure 3.1, in the Appendix 3.1.

7.6.3 Test Organisation

Product ownership and project management roles are rotated in the weekly basis within the team. All other team members are responsible for contributing to business analysis processes to make user requirements and associated user acceptance criteria as complete as possible.

7.6.4 Test Approach

The group project follows Kanban development framework founded upon Agile principles and values. The developers work in 4 days sprints (Monday to Thursday), working on decomposed tasks, prioritised stories and bugs taken from a backlog of work (Trello). The content of each sprint can be adjusted dynamically to reflect the immediate priorities of the project.

7.6.4.1 Test Lifecycle

See Figure 3.2 in Appendix 3.5 for the Test Lifecycle.

Test Levels & Types

- **Unit level - unit testing**: Unit tests are automated tests, written and run by developers that ensure that part of an application (referred as ”unit”) behaves as expected. In object-oriented programming an unit is usually an interface or class. Possibly complex back-end resources, such as databases or web services, are usually stubbed or faked in a test, to ensure that tests do not fail when resource fails, and to ensure that tests run quickly.

- **Integration level - integration testing**: Integration tests are automated tests, written and run by developers, the test software modules that are combined into a group. They are used to ascertain if a software system or component meets functional requirements. They take unit tested modules as input, and generally group these into larger aggregates for testing. Possibly complex back-end resources may be mocked, as with unit test.

- **System, acceptance level - automated end-to-end testing**: end to end testing is a methodology that can be used to check if an entire software component, works as it should under production-like conditions with production-like data. To carry-out these tests, the component is initialised with real-world data, and then a special software automation tool is used to run the component, and check that it behaves as expected. Tests are often expressed using Gherkin syntax (feature, scenario, given, and, and, when, then, and) a simple plain language that allows test-cases to be expressed in a simple but structured way.

- **Nonfunctional testing of ML models**
  - Latency: test harnesses (stubs and drivers; data splitting) to be used to mirror production conditions
  - Throughput: test harnesses to be used to mirror production conditions
  - Memory footprint
  - Cost
  - Carbon footprint: https://codecarbon.io

Regression Test Approach

Unlike functional testing, regression testing does not test whether the software produces the correct results, but whether it behaves as it did before changes were introduced. Snapshot testing implements regression testing by recording the textual output of a test function and comparing this recorded output to a reference output [32].

7.6.5 Test Planning

Mapping User Requirements to Testing

Business analysis defines the user requirements for software components. User stories are initially drafted in a Requirements Catalgoue and Traceability Matrix (RCTM). This format is used to review and prioritise the requirements (functional and non-functional) and associated benefits. All requirements that are not descoped are to be added to the Trello’s weekly backlog. See appendix 3.7 for the Requirements Catalogue Traceability Matrix.

Definition of Ready (DOR)

The project's Definition of Ready (DOR) defines what is required for development of a story (in Trello) to start. Alongside a clear user requirement to be articulated, the DOR also requires that all stories have clear acceptance criteria, i.e. what is expected as an outcome and how it should be tested.

- Items have all requirements and acceptance criteria needed to work on the item
- As a ..., I need ..., So that I can ... (stories to be created in this format for breaking down)
- Acceptance criteria is clear for both development and test
- Aligned to agreed epics
- Broken into sub tasks by developers if needed
- Spikes must be timeboxed and have an outcome
- Align with standards and processes set out for user stories/bugs/tasks
- Item should be achievable within a single sprint where possible
- Design documentation

Definition of Done (DOD)

The DOR is backed up by the Definition of Done (DOD). This outlines the requirements to be able to move an item to 'Done' within the Trello Kanban board.

- Achieved all acceptance criteria
- Unit, integration, and end-to-end test have been written and are working
- Code has been reviewed
- Passed all testing
- Testing documented
- Documentation updated - Read me
- Ready to deploy to production

Traceability of test results to requirements

When an item has passed all testing, against all test scenarios identified, it is moved to "Done" on the Kanban board. The item is checked to ensure that the correct release is assigned.

RAID log or Risk Register

All risks, assumptions, issues and dependencies are managed at project level, including any aspects relating to testing. The project RAID log or Risk Register is a live document as is subject to regular review, it is stored within the project's Teams site.

7.6.6 Test Management Tooling

Item Types

Trello is the project management tool used on this project. It is used to hold the backlog, sprint prioritisation and Kanban board to facilitate iterative, incremental delivery. The following item types are used within Trello for the group project:

- **Epic:** A high level feature item, used as a parent to group user stories. Epics typically detail the high level user and business requirements from the feature.
- **Story:** User stories that outline the requirements of specific functionality within an Epic's feature. User stories are developed following a Behaviour Driven Development (BDD) process, using a standard "As a user, I want to feature functionality so that expected benefit" structure.
- **Sub-task**: sub-tasks represent discrete developer activities, logically broken down to help the development of feature/functionality.
- **Failure**: used to record system and acceptance testing failures against a user story, enabling clear visibility of outstanding issues and facilitating broader reporting on reasons for testing rejections. All failure items must capture steps to reproduce the issue and document any supporting evidence (screenshots, system exports etc.)
- **Spike**: time allocated to technical discovery and solution design ahead of development. Spikes are expected to create associated development tasks and update acceptance criteria within user stories as part of their outputs.
- **Wireframing**: visualise how the proposed features will look and function within the application.
- **Bug**: recorded issues identified in the live system or testing failures that have been agreed to be extracted from current development work.

Kanban board

The Kanban board use the following structure: To do, Doing, In review, Deployed, Sign off, Done

7.6.6.1 Test Environment

The project is developed and deployed according to a continuous delivery model using GitHub. The testing is undertaken within a Docker virtual environment.

- **Development Environment (Alpha)**: development environment used by developers to combine their local code development and for running automated unit, integration and end-to-end testing against consolidated work.
- **User Acceptance Testing Environment (Beta)**: this is the environment where all system and acceptance testing is undertaken.
- **Production Environment (Live)**: releases are only deployed to this environment following a sign-off of completed testing.

7.6.6.2 The Generic Test Process

For the Generic Test Process please refer to Appendix 3.8

7.7 Delivery Plan

As the project aims have been identified aim are decomposed into objectives. For each objectives a list of tasks assigned to. For the Gantt chart see Appendix 3.9.

7.8 Task Management

For the the Meeting Minutes please refer to Appendix 4.

Chapter 8 – Conclusions

8.1 Overall

This research highlights the transformative potential of machine learning for addressing the complex challenges in turbulence modelling, through the application of PINNs and PySINDy. These ML techniques have proven to be effective in resolving closure problems associated with the RANS equations, representing a significant advancement in the field.

Throughout the project, our team have successfully utilised ANSYS for simulations, developed and implemented various machine learning models using Python. These models were integrated into a user-friendly software tool, designed specifically for predicting turbulence. This integration improves the usability of advanced ML models in real-world applications and sets a new standard for software tools in this field. The efficiency and accuracy of PySINDy and PINNs models in handling complex simulations surpass traditional methods, potentially revolutionising the way researchers and engineers approach turbulence modelling.

8.2 Future Works

- **Enhancing the versatility of PySINDy:**
  To broaden the application of PySINDy across diverse flow conditions, advanced regularisation techniques can be explored and implemented. These techniques will improve the robustness of the model, ensuring optimal performance under a variety of challenging conditions. Regularisation will also prevent overfitting, thereby improving the predictive accuracy of the model across extensive datasets.

- **Expanding Development for PINNs:**
  The development checklist for PINNs will be expanded to include more complex turbulence scenarios. This expansion will involve refining the algorithms and integrating additional physics-informed constraints to accurately present the complex dynamics of flow patterns. The goal is to broaden the scopes of PINNs the practical utility for simulating complex turbulent flows.

- **Integration within the ANSYS Environment:**
  To assess the practical viability and effectiveness of PINNs and PySINDy, efforts to integrate the ML models into ANSYS software environment are required. This integration will allow comprehensive simulations and direct comparisons with traditional models. By conducting side-by-side evaluations on this industry-standard platform, will allow us to determine which models yield the most reliable and accurate results in real-world engineering scenarios.

References

1. Computational fluid dynamics (cfd) market size - forecast to 2031. *Business Research Insights*, Apr 2024. URL https://www.businessresearchinsights.com/market-reports/computational-fluid-dynamics-cfd-market-111787.

2. Global computational fluid dynamics market report and forecast 2024-2032. *Expert Market Research*, 2024. URL https://www.expertmarketresearch.com/reports/computational-fluid-dynamics-market.

3. Computational fluid dynamics (cfd) market — size, share, growth — 2024 - 2030. *Virtue Market Research*, 2024. URL https://virtuemarketresearch.com/report/computational-fluid-dynamics-market.

4. R. Bischof and M. A. Kraus. Mixture-of-experts-ensemble meta-learning for physics-informed neural networks. *Forum Bauinformatik*, 2022.

5. J. Blazek. *Computational fluid dynamics: Principles and applications*. Butterworth-Heinemann, 2015. URL https://books.google.com.tw/books?hl=zh-TW&lr=&id=r-ecBAAAQBAJ&oi=fnd&pg=PP1&dq=Bla%C5%BEek.

6. S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113. URL https://www.pnas.org/content/113/15/3932.

7. B. M. Byrne. *Structural equation modeling with Mplus : basic concepts, applications, and programming*. Routledge, 2012. ISBN 9781136663451.

8. K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116 (45):22445–22451, 2020. doi: 10.1073/pnas.1911366116. URL https://www.pnas.org/content/116/45/22445.

9. A. Chawla. The advantages and disadvantages of pca to consider before using it, May 2023. URL https://blog.dailydoseofds.com/p/the-advantages-and-disadvantages.

10. B. Cihan. Numerical drag reduction of a ground vehicle by naca2415 airfoil structured vortex generator and spoiler. *International Journal of Automotive*, 20:943–948, Aug 2019. doi: 10.1007/s12239-019-0088-6. URL https://link.springer.com/article/10.1007/s12239-019-0088-6.

11. J. Cohen, P. Cohen, S. G. West, and L. S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. LAWRENCE ERLBAUM ASSOCIATES, 2003. ISBN 9780805822236.

12. B. M. de Silva, K. Champion, M. Quade, J. Loiseau, J. N. Kutz, and S. L. Brunton. Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, 2020. doi: 10.21105/joss.02104. URL https://doi.org/10.21105/joss.02104.

13. V. Driessen. A successful git branching model. 2010. URL https://nvie.com/posts/a-successful-git-branching-model/.

14. H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7):075117, 2022. doi: 10.1063/5.0095270. URL https://doi.org/10.1063/5.0095270.

15. F.-A. Fortin, F.-M. De-Rainville, M.-A. Gardner, M. Parizeau, and C. Gagne. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012. URL https://www.jmlr.org/papers/volume13/fortin12a/fortin12a.pdf.

16. A. F. Gad. Pygad: An intuitive genetic algorithm python library. *Multimedia Tools and Applications*, 2023. doi: 10.1007/s11042-023-17167-y. URL https://doi.org/10.1007/s11042-023-17167-y.

17. G. Hinton and L. van der Maaten. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL https://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

18. Z. Jaadi. A step-by-step explanation of principal component analysis. *Built In*, Sep 2019. URL https://builtin.com/data-science/step-step-explanation-principal-component-analysis.

19. A. Jameson and M. Fatica. Using computational fluid dynamics for aerodynamics. Technical report, Stanford University. URL http://aero-comlab.stanford.edu/fatica/papers/jameson_fatica_hpc.pdf.

20. Keboola. A guide to principal component analysis (pca) for machine learning, Apr 2022. URL https://www.keboola.com/blog/pca-machine-learning.

21. I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

22. S. Luo, M. Vellakal, S. Koric, V. Kindratenko, and J. Cui. Parameter identification of rans turbulence model using physics-embedded neural network. *High Performance Computing*, 34:9, 2020. doi: 10.1007/978-3-030-59851-8_9. URL https://doi.org/10.1007/978-3-030-59851-8_9.

23. P. D. Morris, A. Narracott, H. von Tengg-Kobligk, D. A. Silva Soto, S. Hsiao, A. Lungu, P. Evans, N. W. Bressloff, P. V. Lawford, D. R. Hose, and J. P. Gunn. Computational fluid dynamics modelling in cardiovascular medicine. *Heart*, 102(1):18–28, 2015. doi: 10.1136/heartjnl-2015-308044. URL https://heart.bmj.com/content/102/1/18.

24. MR_CFD. Aerodynamic & aerospace engineering. URL https://www.mr-cfd.com/industries/aerodynamic-and-aerospace-engineering/.

25. S. G. Nicola Orsini, Rino Bellocco. A comparison of principal component analysis, maximum likelihood and the principal axis in factor analysis. *The Stata Journal*, pages 40–57, 2006. doi: 10.1177/1536867X0600600103. URL https://journals.sagepub.com/doi/epdf/10.1177/1536867X0600600103.

26. T. Norton and D. Sun. Computational fluid dynamics (cfd) – an effective and efficient design and analysis tool for the food industry: A review. *Trends in Food Science & Technology*, 17(11):600–620, Nov 2006. doi: 10.1016/j.tifs.2006.05.004. URL https://www.sciencedirect.com/science/article/abs/pii/S0924224406001981?casa_token=CTVLVlDTjQcAAAAA:5P7gBQs4yA3jhK7MzfnOTrHN4a-D1VkfAddWmpVB5ERKlyJwfQUAcGMKojJa-bmggvrl1CqUoU0.

27. O. S. O. Onyekachi Akuoma Mabel. A comparison of principal component analysis, maximum likelihood and the principal axis in factor analysis. *American Journal of Mathematics and Statistics*, pages 44–54, 2020. doi: 10.5923/j.ajms.20201002.03. URL https://www.researchgate.net/publication/348817620.

28. F. Pioch, J. H. Harmening, A. M. Müller, F.-J. Peitzmann, D. Schramm, and O. el Moctar. Turbulence modeling for physics-informed neural networks: Comparison of different rans models for the backward-facing step flow. *Fluids*, 8(2):43, 2023. doi: 10.3390/fluids8020043. URL https://doi.org/10.3390/fluids8020043.

29. Praxis. What is principal component analysis?, May 2022. URL https://praxis.ac.in/what-is-principal-component-analysis/.

30. M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

31. R. D. Sandberg, S. Hanrahan, M. Kozul, H. Jagode, H. Anzt, G. Juckeland, and H. Ltaief. Studying turbulent flows with physics-informed neural networks and sparse data. *International Journal of Heat and Fluid Flow*, 104:109232, 2023. doi: 10.1016/j.ijheatfluidflow.2023.109232. URL https://doi.org/10.1016/j.ijheatfluidflow.2023.109232.

32. U. Schmitt. pytest-regtest 2.1.1 documentation. 2024. URL https://pypi.org/project/pytest-regtest/.

33. A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

34. S. Thompson. *Managing Machine Learning Projects*. Manning, 2023. ISBN 9781633439023.

35. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

36. J. M. Wallace. Parameter space reduction using approximate canonical correlation analysis. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, pages 1–12, 2004. doi: 10.2514/6.2004-4513. URL https://arc.aiaa.org/doi/epdf/10.2514/6.2004-4513.

37. A. H. WSP. Test strategy - unpublished confidential document. 2023.

38. Z. Zhai. Application of computational fluid dynamics in building design: Aspects and trends. *Indoor and Built Environment*, 15(4):305–313, 2006. doi: 10.1177/1420326x06067336. URL https://heart.bmj.com/content/102/1/18.

Appendix 1 – User Requirements

1.1 Understand the Problem

Problem Definition: Solving the RST in the RANS equations (a set of partial differential equations) improve the predictive capabilities of CFD simulations. This task is crucial for accurate modelling of turbulent flows, which are prevalent in various engineering and scientific applications. In total, there are 10 different unknowns for a 3-dimensional flow, three mean velocities, one pressure, and six Reynolds stress components. Therefore, the aim is to use ML algorithm to predict the RST in specific channel flow (data given by customer), reducing the number of unknowns in the RANS equation for better simulation prediction.

RANS	Reynold Stress Tensor
$\frac{\partial \langle \boldsymbol{u} \rangle}{\partial t} + (\langle \boldsymbol{u} \rangle \cdot \nabla)\langle \boldsymbol{u} \rangle = -\frac{1}{\rho}\nabla\langle p \rangle + \nu\Delta\langle \boldsymbol{u} \rangle + \frac{1}{\rho}\nabla\bar{\tau}$	$\tau_{ij} = \mu_t\left(\frac{\partial\langle u_i \rangle}{\partial x_j} + \frac{\partial\langle u_i \rangle}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\rho k = \mu_t S_{ij} - \frac{2}{3}\delta_{ij}\rho k$

1.2 Identify User Needs

Potential Industry and customers as follow:

- Aerospace. For aircraft, the design and manufacture of aircraft can use the improve version of RANS to predict the aerodynamic performances on the various design more accurately. For space, it can be used on the spacecraft design and analysis to assess the effects of atmospheric re-entry where turbulent flows are significant. Potential customers are Boeing, Airbus, NASA, and SpaceX.
- Automotive. For vehicle aerodynamics, optimising the design improves the fuel efficiency and performance of vehicles, especially for fast moving vehicles like Formula One. Potential customers are Ferrari, Mecedes, McLaren, and Aston Martin

Figure 1.1 illustrates the Boeing organisational chart. The red boxes indicate the potential departments where our product will be delivering to:

- Commercial Airplanes (Main). The algorithm pertains to fluid dynamics and aerodynamics, this sector would be directly interested in the application of the algorithm to improve aircraft design and performance.
- Defense, Space & Security (Main). The algorithm can be used for designing military aircraft, spacecraft, and defense systems where fluid dynamics is a key consideration.
- Engineering (VP-Engineering). The central hub for Boeing's technical projects, which would be directly involved in the integration and implementation of the ML algorithm.
- Space and Missile Systems. For applications related to spaceflight dynamics and missile trajectory simulations where fluid dynamics models are essential.

- Phantom Works (VP-Phantom Works). As Boeing's advanced prototyping arm, they work on the development of new technologies for future aerospace systems and can be heavily involved in the practical application and testing of the algorithm.

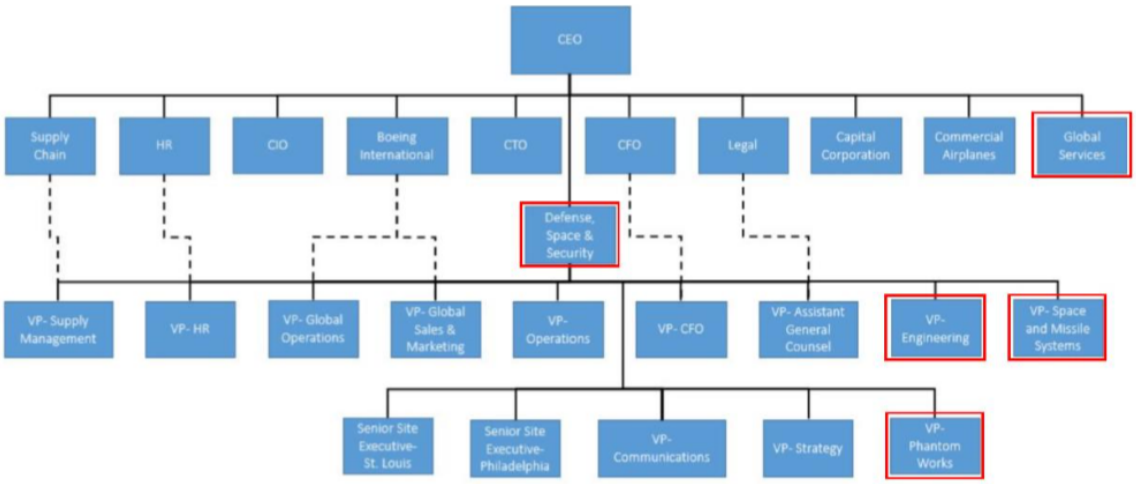


Figure 1.1: Illustration of Boeing Organisational Chart

1.3 Define Specific Requirements

1.3.1 Data Requirements

Type of Data Needed. To address the Reynolds stress tensor in RANS equations, we would require experimental or simulation data of fluid flow, which include velocity fields, pressure measurements, and turbulence characteristics. There should be sufficient dataset (different surfaces) for our ML algorithm training and testing. It should be compiled in csv or excel format.

Sources of Data. The datasets should be contributed by the clients such as historical flight tests or generated through simulations.

Data Collection Method. Clients should collect the live data using their own tools or through CFD simulations, handing over the datasets to our team. Our team will not be responsible for data collection.

Data Privacy Concerns: The given dataset by the clients is solely used within the project team and will be kept confidential. Encryption will be enforced during data handling, and it will comply with ITAR (International Traffic in Arms Regulations) and other relevant regulations.

1.3.2 Functional Requirements

Model Functionality. The ML model should accurately predict the RST at specific channel flow conditions.

Input Data. The inputs are shown below.

- Kinematic viscosity (ν)
- Friction velocity (u_τ)
- Reynolds number based on friction velocity (Re_τ)
- Wall-normal grid point in wall units (y^+)

Expected Output. The outputs are shown below. Graphical charts will be included for better illustrations.

- Streamwise velocity variance (u'u')
- Wall-normal velocity variance (v'v')
- Spanwise velocity variance (w'w')
- Streamwise-wall-normal velocity covariance (u'v')
- Mean streamwise velocity (U)
- Mean streamwise velocity gradient ($\frac{dU}{dy}$)
- Mean pressure (P)
- Turbulent kinetic energy (k)

System Integration The model can be integrated into Boeing's existing CFD software to enhance its predictive capabilities or used as a standalone tool within the design process for rapid prototyping.

1.3.3 Performance Requirements

Accuracy. The ML model should achieve a high degree of correlation with simulated/experimental results, MSE (¡0.01) and RMSE will be deployed to check.

Speed. The ML model should be able to provide predictions of the Reynolds stresses within 10 minutes with CSV file not exceeding 200MB. It is not used for live predictions with zero latency.

Robustness. The ML model should be able to handle different types of input data and maintain performance. It will be validated using sensitivity and stress testing.

Reliability. The model should be able to provide consistent result given the same inputs, repeated K-Fold cross-validation will be used.

1.3.4 System Requirements

Hardware.

- **Processing Power.** 3.8GHz processing power for training deep learning models and data preprocessing.
- **Memory.** 16GB RAM to handle large matrices in-memory during model training.
- **Storage.** SSDs (1TB) with high read/write speeds for faster data access during training and large HDDs for long-term data storage.

- **Network.** High-speed network interface cards for data-intensive workloads or cloud-based training sessions.

Software.

- **Operating Systems.** The OS needed to support the software stack, which could include specific versions of Linux, Windows, or macOS.
- **Programming Languages.** Python 3 with support for libraries such as TensorFlow, PyTorch, or Scikit-learn.
- **Storage.** SSDs (1TB) with high read/write speeds for faster data access during training and large HDDs for long-term data storage.
- **Network.** High-speed network interface cards for data-intensive workloads or cloud-based training sessions.

Infrastructure.

- **Cloud and On-premises Infrastructure.** Model Training will be conducted on Cloud while data storage will be on-premises.
- **Security.** Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) for secure data transmission and role-based access control for system access.

1.3.5 Constraints and Limitations

- **Technical Constraints.** Integrating of ML model into an established CFD software could pose compatibility challenge that requires extensive software development work.
- **Computational Constraints.** Training large models could require significant memory and storage when handling large datasets typical in CFD simulations. High performance computers will be employed, the data set file size should be kept within 200MB.

Appendix 2 – Software Streamlit

Figure 2.1 demonstrates PySINDy user interface (UI). The application provides a few options for the user. Firstly, interpolation allows the CSV data points to generate to the desired data points. For optimal result, it is recommended to have 10,000 data points. Secondly, user has the option to view the raw data visualisation. Thirdly, hyper parameters can be adjusted to find the optimal model when different dataset are uploaded.

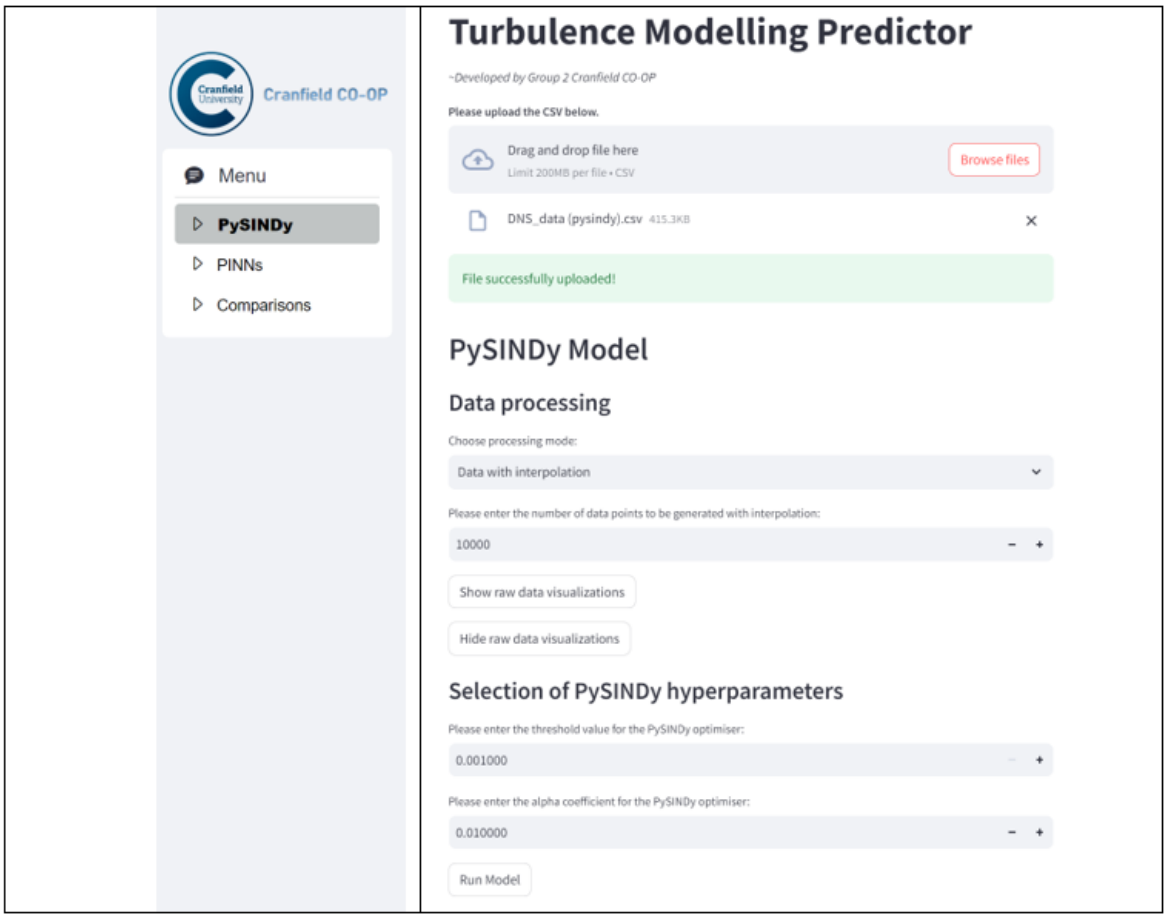


Figure 2.1: PySINDy User Inferface

Figure 2.2 shows the results generated from the inputs keyed in by the user. The application generates the PySINDy model equations, the execution time, the R^2, the MSE and remarks to notify the user if it is overfitting. Five different graphs are then generated (U, u'u'. v'v', w'w' u'v', against y^+) with predicted PySINDy model and DNS data. A download prediction results button is available for user to download the result.

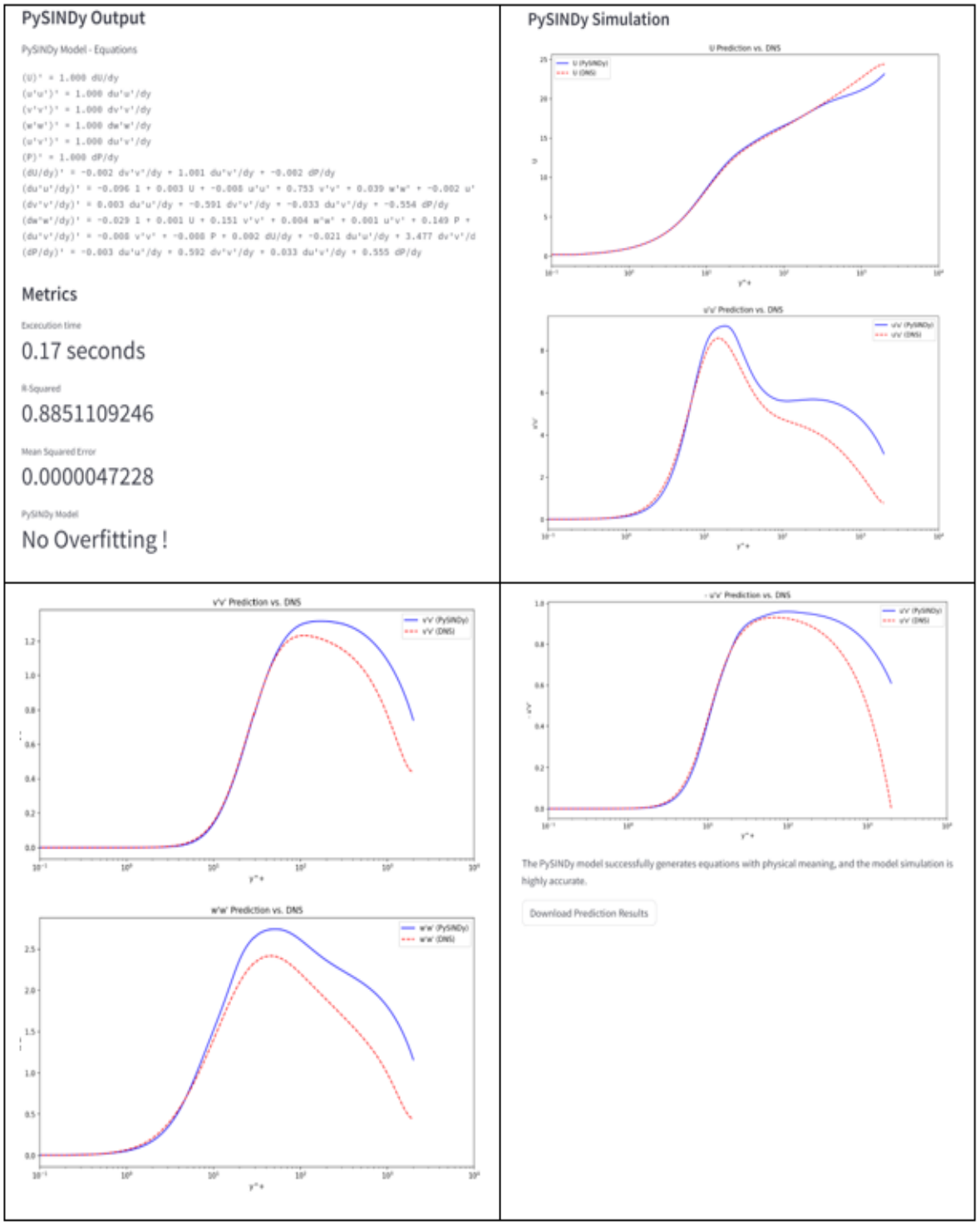


Figure 2.2: Results of PySINDy

Figure 2.3 shows two available modes for PINNs on the application, inference mode and test mode. Inference mode uses the existing database in the application, allows the user to select the desired Re_tau, y_minmum, y_maximum and y/δ. These inputs will filter away unrequired data from the database.

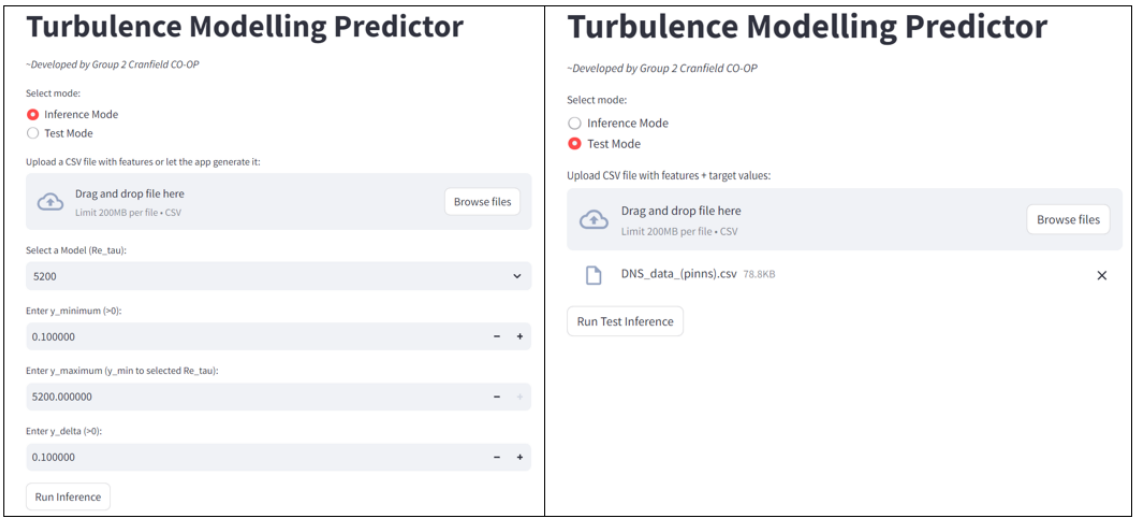
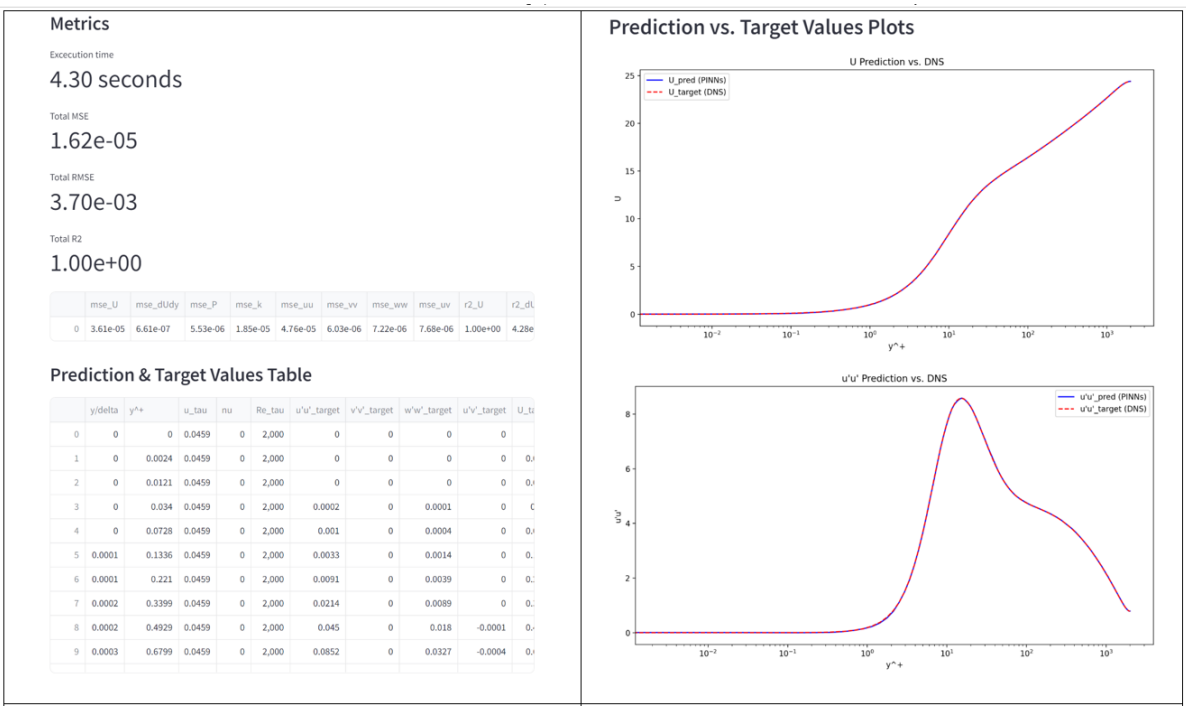


Figure 2.3: PINNs User Interface

Figure 2.4 shows the results generated from the inputs keyed in by the user. Similar to PySINDy results, the application generates the execution time, the R^2, the MSE and the RMSE. Five different graphs are then generated (U, u'u'. v'v', w'w' u'v', against y^+) with predicted PINNs model and DNS data. A download prediction results button is available for user to download the result.



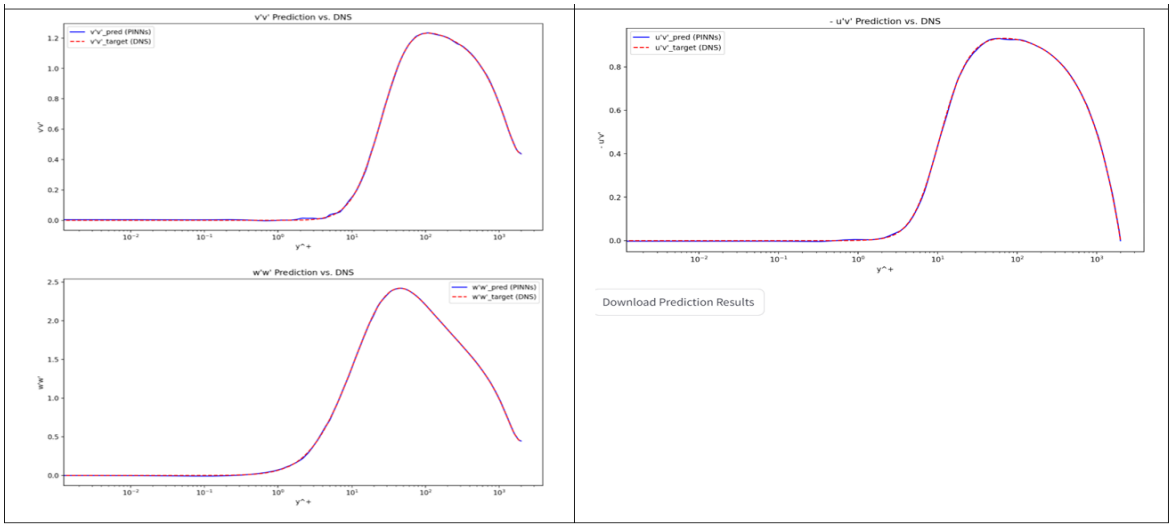


Figure 2.4: Results of PINNs

Figure 2.5 show the comparison user interface. The user has the options to select which model for comparison. DNS(Oden) refers to the online DNS database that was used. The application also allows new DNS CSV file to be uploaded for comparison. Next, user must upload the required files (downloaded from previous results). Lastly, press the comparison button to generate the results.

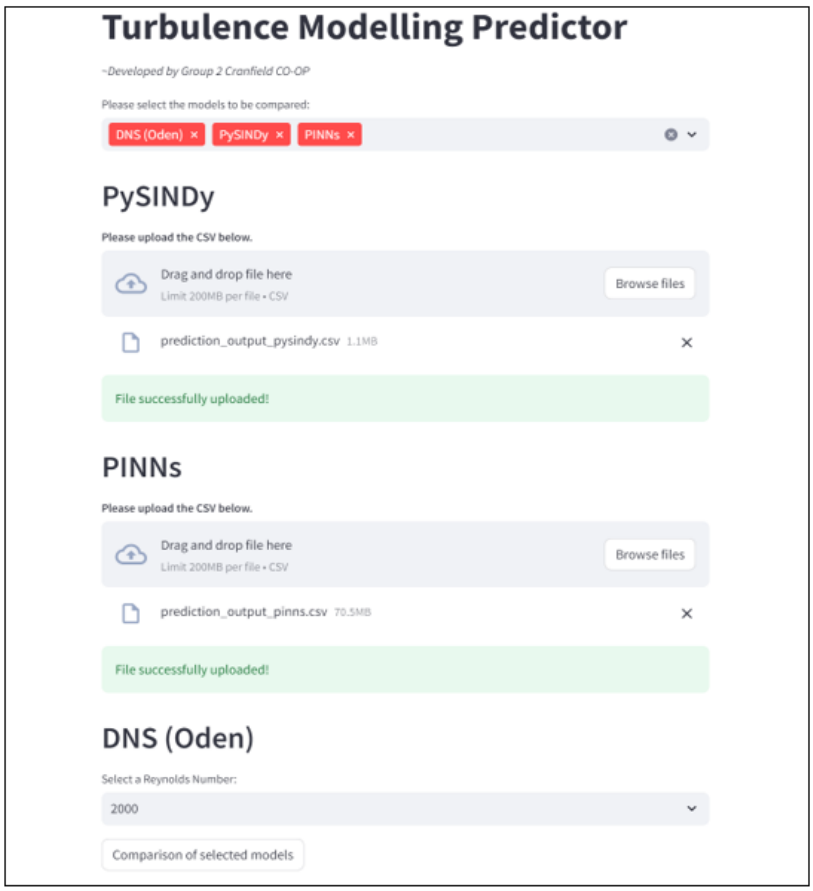


Figure 2.5: Comparison Results

Figure 2.6 shows the comparison results of DNS(Oden), PySINDy and PINNs. Five different graphs are then generated (U, u'u'. v'v', w'w' u'v', against y^+). Logarithmic scale and grids tools are available to improve the data visualisation for the users.

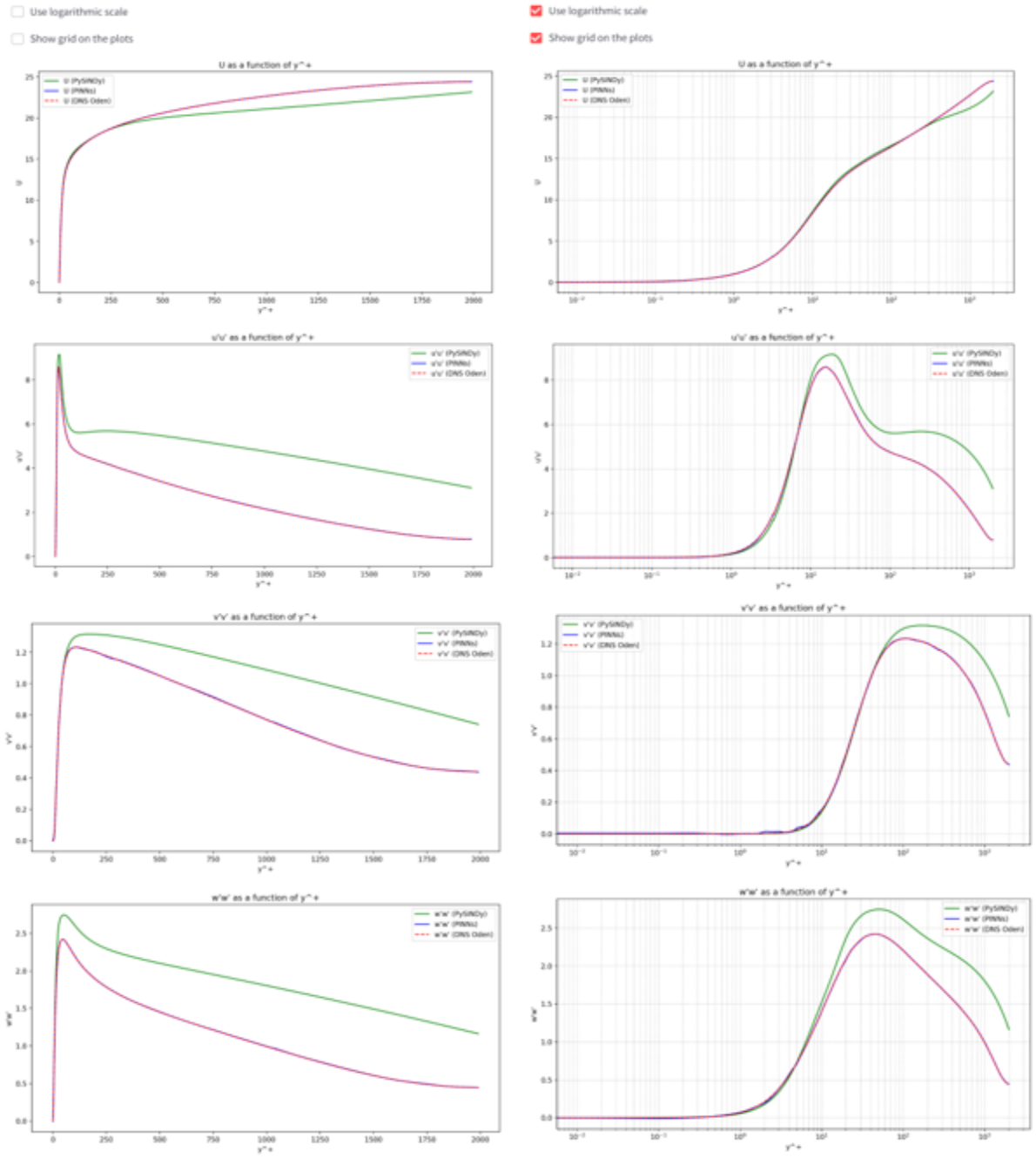


Figure 2.6: Comparison Results

Appendix 3 – Project Plan

3.1 Risk Register

Ref	Activity Element	Information Delivery Issue	Risk Rating			Measures Taken to Eliminate or Reduce the Hazard	Owner	Date Raised	Action by	Mitigation Date
			Likelihood	Severity	Risk					
0	PLQs	Do we know what do deliver?	2	2	4	-	-	-	-	-
		Do we know what requirements we are working to?	4	4	16	CIDA defined the user requirements	Marcell Gyorei	11.03.2024	-	04.04.2024
		Do we have agreed work methodologies?	2	2	4	-	-	-	-	-
		Do we have defined responisibilities and accountabilities?	4	4	16	No measures taken - agile methodology is used with task allocation on Monday and task review on Thursday	Marcell Gyorei	11.03.2024	-	-
		Do we have a delivery plan to work to?	2	2	4	-	-	-	-	-
		Do we have the capacities and capabilities to deliver?	2	2	4	-	-	-	-	-
1	Risk Management	No understanding of what ML specific risks to gather (too many unknowns) - potential impact: poor task allocation, poor model quality, poor integration	4	4	16	Risk Register to be filled continously	Marcell Gyorei	20.03.2024	Marcell Gyorei	20.03.2024
2	Project Specific Risk	No team-wide understanding of agile methodology on ML specific projects; too much push (work assignment) instead of pull - need balance	4	4	16	Kanban handover on Trello	Marcell Gyorei	20.03.2024	Marcell Gyorei	20.03.2024
	Project Specific Risk	SE don't understand the process on ML specific projects - potential impact: lack of focus on meetings and task allocation	4	4	16	Industry guidance - list of tasks, checklist	Marcell Gyorei	20.03.2024	Marcell Gyorei	20.03.2024

Figure 3.1: Risk Register

3.2 Branching Models

3.2.1 Github-flow branching model

Github-flow allows continuous delivery. It has only a ”main” branch that other developer branches-off whenever the developer works on a new feature. The branch name is descriptive to the feature the developer is working on. At integration the developer create a ”pull-request” that other collaborators review and approve before merging it back to the ”main”. Upon completion the feature branch will be deleted.

3.2.2 Git-flow branching model

Git-flow [13] allows explicit software versioning on the side of continuous delivery.

- ’origin/main’ branch: production ready state of source code. Integration branch. Every commit to main can hook up a script that automatically builds and rolls-out the software on the server. Every merge from the ”develop” branch tags the change with a release number.
- ’origin/develop’ branch: reflects to the latest delivered changes. Work-in-progress branch.
- ’feature’ branch: exists in developers repos only - not part of the origin. It is a temporary branch.
- ’release’ branch: allows the developer team to split temporary - some developers are fine-tuning the software for release on the ’release branch’ while others continue working on to integrate features to the ’develop’ branch. Upon completion it is merged to ’main’ and it must be tagged with a release number for future reference. Finally it also has to merged back to ’develop’ branch to update it with the latest bug-fixes. It is a temporary branch.
- ’hotfix’ branch: it is for unplanned release to fix a critical bug in production version (in the ’main’ branch). It is created from the ’main’ branch. After completion it needs to be merged back to ’main’ and then to ’develop’. If the ’release’ branch still exists it needs to be merged back to that first. It is a temporary branch.

3.3 Reproducibility - list of tools

- Pip: package management system. The team provided a requirements.txt file that contains the name and versions of all required packages.
- Jupyter: interactive computing service
- PyTorch: ml framework based on python
- Docker: platform for container application development and deployment

3.4 ML Checklists

3.5 Test lifecycle

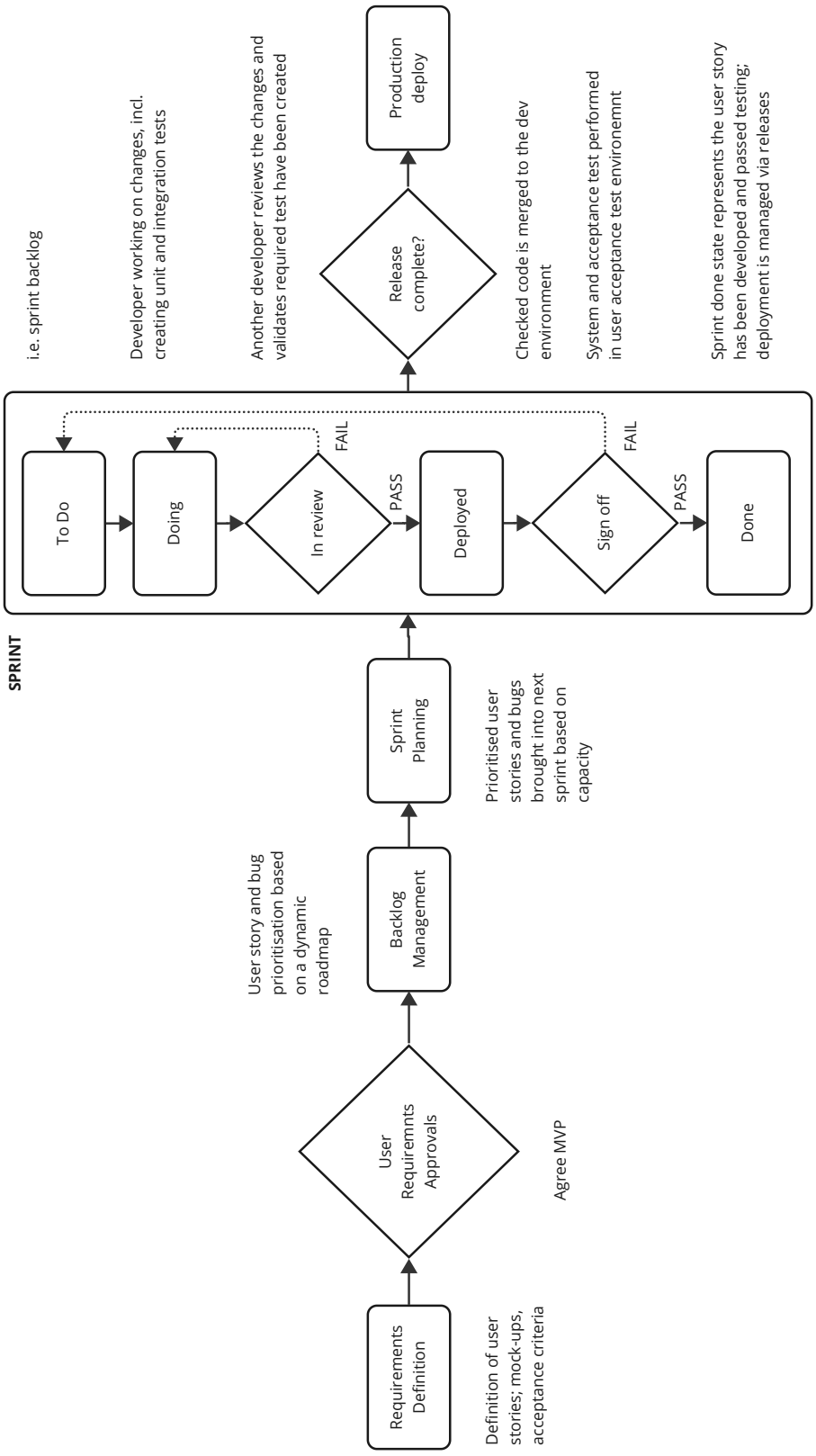


Figure 3.2: Test Lifecycle

3.6 Dual Track Agile with CRISP-ML

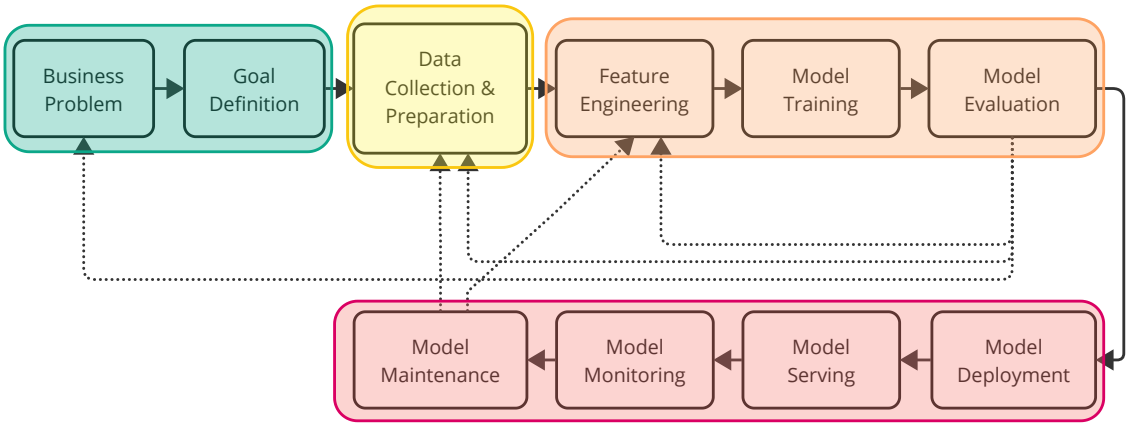


Figure 3.3: CRIPS-ML

3.7 RCTM

Rq. ID	Requirement Description - Functional	User Acceptance Criteria	Test ID	Status
R_2.1	PySINDy model should accurately predict the Reynolds stress tensor at specific channel flow conditions. Accuracy. The ML model should achieve a high degree of correlation with simulated/experimental results using MSE <0.01.	INPUT: y^+, Re_τ, U, $u'u'$, $v'v'$, $w'w'$, $u'v'$, P, $\frac{dU}{dy}$ OUTPUT: $u'u'$, $v'v'$, $w'w'$, $u'v'$, U MSE <0.01	T_210	Done
R_2.2	PINN model should accurately predict the Reynolds stress tensor at specific channel flow conditions; Accuracy. The ML model should achieve a high degree of correlation with simulated/experimental results using MSE <0.01.	INPUT: y/δ, y^+, Re_τ, u_τ, ν OUTPUT: $u'u'$, $v'v'$, $w'w'$, $u'v'$, k, U, $\partial U/\partial y$, P MSE <0.01	T_220	Done

Table 3.1: RCTM 1

Rq. ID	Requirement Description - Non-Functional	User Acceptance Criteria	Test ID	Status
R_3.1	Speed. The ML model should be able to provide predictions of the Reynolds stresses within 10 minutes with CSV file not exceeding 200Mb. It is not used for live predictions with zero latency.	t <10 minutes	T_310	Done
R_3.2	Scalability. The software must be able to process datasets of varying sizes (small scale and large scale <200MB) and complexity without significant degradation in performance.	File size <200MB	T_320	Done
R_3.3	Robustness. The ML model should be able to handle different types of input data and maintain performance. It will be validated using sensitivity & stress testing.	SCENARIO 1: user runs the simulation GIVEN a noisy dataset WHEN I want to run simulation THEN simulation should return a result within the required accuracy	T_330	Done
		SCENARIO 2: user runs the simulation GIVEN a limited dataset WHEN I want to run simulation THEN simulation should return a message that 'not enough data'	T_331	Done
R_3.4	Reliability. The model should be able to provide consistent result given the same inputs, repeated K-Fold cross-validation will be used.	Training set to be diveded into 5 to 10 subsets Train the model on four fold and evaluate it on the remaining fold	T_340	Done

Table 3.2: RCTM 2

Rq. ID	Requirement Description - Data	User Acceptance Criteria	Test ID	Status
R_1.1	Type of Data Needed. To address the Reynolds stress tensor in RANS equations we would require experimental or simulation data of fluid flow, which include velocity fields, pressure measurements,	SCENARIO 1: user updates the datasets GIVEN a non-csv file WHEN I want to update the file THEN it should return a ”Incorrect file format. Input data must be in csv format”	T_110	To Do
	and turbulence characteristics. There should be sufficient dataset (different surfaces) for our ML algorithm training and testing. It should be compiled in csv or excel format; Alerts and Errors. The software must prompt the user if incorrect files or parameters are input into	SCENARIO 2: user runs PySindy simulation GIVEN input paremeters, and one or more parameters are missing from the datasets WHEN I want to run simulation THEN it should return a ”Missing parameters”	T_111	To Do
		SCENARIO 3: user runs PINN simulation GIVEN input parameters, and one or more parameters are missing from the datasets WHEN I want to run simulation THEN it should return a ”Missing parameters”	T_112	To Do
R_1.2	Automated Data Pre-Processing. The software must have an automated pipeline for data cleaning and preparation. It must also be able to detect and handle outliers or missing values in the input data	The software must have an automated pipeline for data cleaning and preparation. It must also be able to detect and handle outliers or missing values in the input data. (IQR, upper fence, lower fence)	T_120	Done
R_1.3	Sources of Data. The datasets should be contributed by the clients such as historical flight tests or generated through simulations.	Part of the Exclusions in the Quality Plan (items that are not tested)	NA	NA
R_1.4	Data Collection Method. Clients should collect the live data using their own tools or through CFD simulations, handing over the datasets to our team. Our team will not be responsible for data collection.	Part of the Exclusions in the Quality Plan (items that are not tested)	NA	NA
R_1.5	Data Privacy Concerns: The given dataset by the clients is solely used within the project team and will be kept confidential Encryption will be enforced during data handling, and it will comply with ITAR (International Traffic Arms Regulations) and other relevant regulations.	Encryption is not a priority item on the roadmap	NA	NA

Table 3.3: RCTM 3

3.8 Generic Test Process

Offline test environment - actions

- Access to the test environment
- Deploy the testing infrastructure to the test environment
  - Data pipeline
  - Test harness / mock application
  - Selected models and associated artifacts for testing
  - Data gathering and feedback collection
- Smoke test
- Run the pipeline to hydrate the environments with data and other required components (initialisation weights/transfer models)
- Execute the test and gather the results

Online test environment - actions

- Shadow/stage testing using mock data and noisy data

Defect Categories

- Fault: the tester encountered a behaviour which does not comply with the specification or does not meet the intended requirement.
- Problem: tester encountered difficulty in proceeding with test execution, due to issues with the supporting test assets. For instance hallenges with the test environment build and configuration; installation of test items or any required tooling; missing or inaccurate test data; errors in a test case or test script
- Observation: the tester has noticed an unexpected feature of the item(s) under test which is neither considered a Fault or Problem, but should be brought to the attention of the others on the delivery.
- Query: the tester requires more information, to asses an aspect of the items(s) under test more fully or to better understand the context in which they are testing.
- Request for Change: the tester has encountered behaviour with the item(S) under test that, while correct with respect to the relevant specification, has highlighted a potential need to change this specification.

Defect Priorities

- Priority Level 1, 2, 3, 4, 5

Defect Severity (live business operation)

- Critical, High, Medium, Minor, Service Request, Query

Standard Exit Criteria as the Quality Bar - Machine Learning

- Test Coverage Criteria on ML: there is no established way to express testing coverage for ML models. Coverage does not refer to lines of code in machine learning as it does in software development.
- Defect Criteria
- Go-Live Contingency Criteria

3.9 Gantt chart

Objectives	Id	Task	26-1 Mar	4-8 Mar	11-15 Mar	18-22 Mar	25-29 Mar	1-5 Apr	8-12 Apr	15-19 Apr	22-26 Apr	29 Apr - 3 May
Objective 1 **Understand Background**	1.1	Understand the Equations & DNS data	Done									
	1.2	Conduct Literature Reviews of ML Techniques & ANSYS		Done								
	1.3	Develop Machine Learning Flowchart		Done								
	1.4	Mesh Generation		Done								
	1.5	Prepare materials for Supervisor Meeting 1		Done								
Objective 2 **Data Exploration & Analysis/ Mesh Automation**	2.1	Mesh Script			Extend	Done						
	2.2	Mesh Automation (MATLAB)			Extend	Done						
	2.3	Prepare the DNS data (Basic Cleaning)			Done							
	2.4	Visualisation of DNS Data using PCA & T-SNE			Done							
	2.4.1	Generate PCA diagrams			Done							
	2.4.2	Generate T-SNE diagrams			Done							
	2.5	DNS to Classic NN input			Done							
	2.6	PCA output to PySINDy to PINN			Done							
	2.6.1	Format PCA output			Done							
	2.6.2	Generate PySINDy output			Done							
	2.6.3	Format PySINDy output			Done							
	2.6.4	Generate PINN output			Done							
	2.7	Propose Alternate Approach of using PINN			Done							
	2.8	Prepare materials for Supervisor Meeting 2			Done							
Objective 3 **Develop Basic Algorithm**	3.1	Fluent automation (MATLAB)				Done						
	3.1.1	Simulation Set -up				Done						
	3.1.2	Simultation Validation				Done						
	3.1.3	Simulation Script				Done						
	3.2	Validate PySINDy Output				Done						
	3.2.1	Use DNS Data				Done						
	3.2.2	Use CFD				Done						
	3.3	Use DNS Data to generate results from PINN				Done						
	3.4	Use DNS Data to generate results from Transformer				Done						
	3.5	Use DNS Data to generate results from PySINDy				Done						
	3.5.1	Genetic Algo with PySINDY output				Extend	Done					
	3.5.2	PPO with PySINDY output				Suspend						
	3.5.3	PINN with PySINDY output				Suspend						
	3.6	Explore Factor Analysis				Done						
	3.6.1	Canonical Correlation Analysis				Done						
	3.6.2	Principal Axis Factor				Done						
	3.6.3	Factorial Analysis of Mixed Data				Done						
	3.6.4	Unweighted Least Squares				Done						
	3.6.5	Generalized Least Squares				Done						
	3.6.6	Spearman & Pearson				Done						
	3.7	Prepare materials for Supervisor Meeting 3				Done						
Objective 4 **Enhanced Algorithm**	4.1	Adjust parameters					Done					
	4.1.1	Genetic Algo					Done					
	4.1.2	PPO					Suspend					
	4.1.3	PINN					Done					
	4.2	Finalise the approach					Done					
	4.3	Evaluate the Performance using CFD					Done					
	4.4	Model Optimisation					Done					
	4.4.1	Adjust parameters					Done					
	4.4.2	Evaluate the Performance					Done					
	4.4.3	Attain the Best Performance					Done					
Objective 5 **Improve CFD**	5.1	Conduct PCA using CFD data						Suspend				
	5.2	Generate PySINDy output						Done				
	5.3	Improve PySINDy Output with GA (algorithms)						Done				
	5.3.1	Adjust parameters with PySINDy model						Done				

Figure 3.4: Gantt chart, page 1

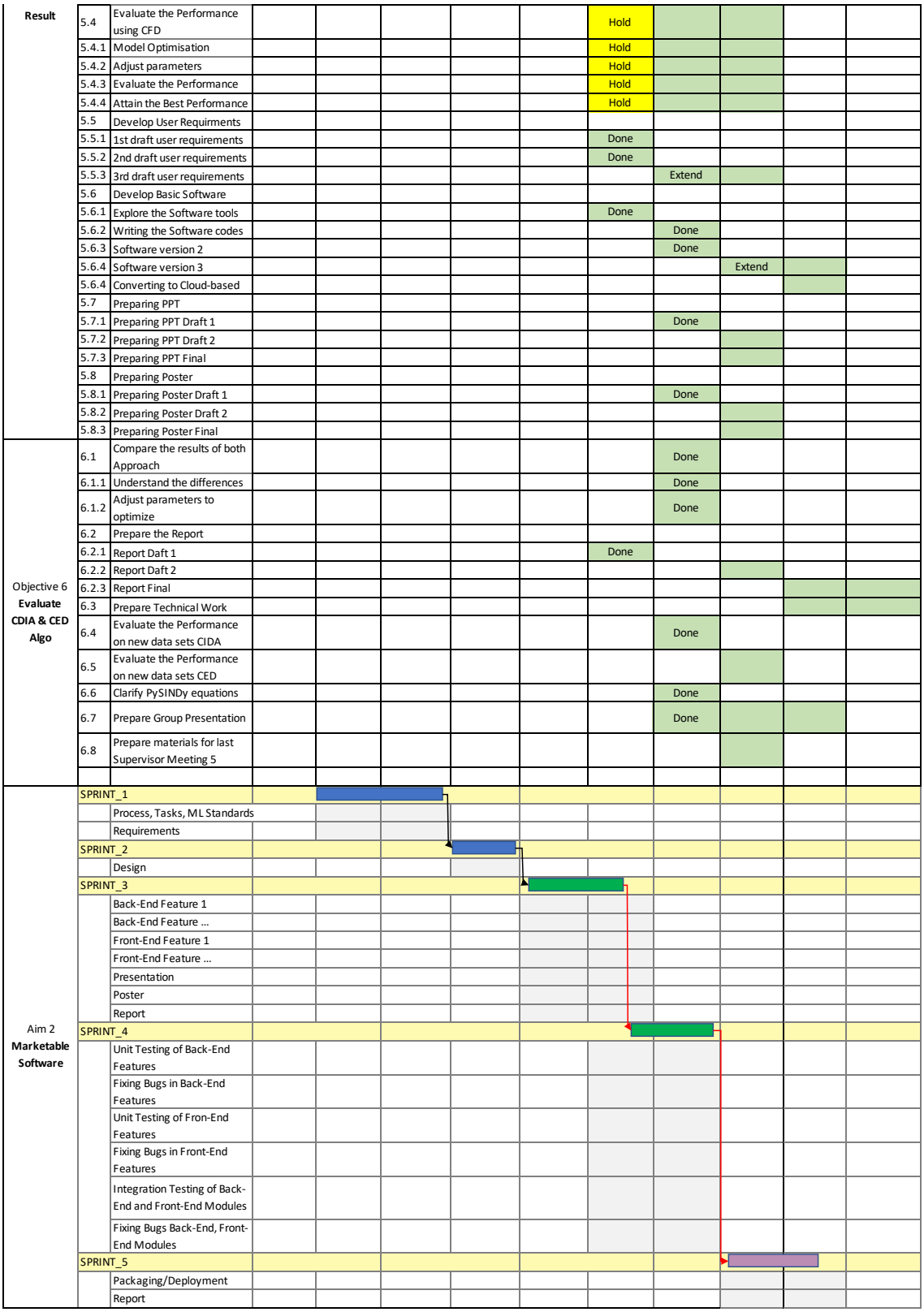


Figure 3.5: Gantt chart, page 2

Appendix 4 – Group Meeting Minutes

Meeting 1 on 1 March 2024 (Physical Meeting)

Agenda 1

S/N	Items
1.	Attendance Update
2.	Conducting Literature Reviews - Packages - Machine Learning Techniques - Data processing
3.	Project Management - Gantt Chart - Unified Modelling Language (UML) - Documentations
4.	Any other business

Secretary: Deng Chuan Chang

Minutes 1

S/N	Items	Action By
Item 1: Attendance Update		
1	The attendance for the group meeting are as follows: Present: - Alexis Balayre - Cheng Chung Yueh - Chang Deng Chuan (Secretary) - Julien Godfroy - Lyderic Faure - Majuran Chandrakumar - Marcell Gyorei Absent: Nil Apologies: Nil	-
Item 2: Conducting Literature Reviews		
2.	The group has discussed a list of literature reviews to be completed for the upcoming week to decide which approaches to follow. The ways to use it and its advantages and disadvantages will be presented and discussed in the meeting 2. a. PySINDy. To understand the inputs format and outputs. b. Genetic Algorithms. c. Machine Learning Packages. To understand Tensor Flow & PyTorch packages usage and best used for which type of machine learning technique(s). d. Proximal Policy Optimization (PPO).	 Chung Yueh Chung Yueh Alexis Majuran

	e. Convolutional neural network (CNN).	Majuran
	f. Principal Component Analysis (PCA).	Majuran
	g. Learning Processes. To understand the three types of biases observational, Inductive biases, learning biases.	Deng Chuan
	h. Physics-Informed Neural Network (PINN).	Julien
Item 3: Project Management		
3.	The various parts of project management are listed below. Marcell is the overall in-charge for project management. The different team members will assist him on some of the tasks.	
	a. Gantt Chart.	
	i. Computational Fluid Dynaimic	Lyderic
	ii. Machine Learning	Deng Chuan
	iii. Software Development	Marcell
	b. Unified Modelling Language (UML)	Marcell
	i. Understand and develop the deliverables for the project	
	ii. Create a basic flow charts	
	iii. Educate the group on UML	
	c. Documentations	
	i. Establishing the processes for different documentations	Marcell
	ii. Create a basic report for subsequent updates	
Item 4: Any Other Business		
4.	a. Data processing. Combined different DNS into a document. Data cleaning and automation will be done subsequently after the format for PySINDy is updated.	Julien
	b. Questions. Everyone is to compile a list of questions to be asked and clarified during the meeting 2 and presentation day.	All
	c. Agenda and Minutes documents. To be updated and keep tracked, including timesheet will be sent on Friday afternoon.	Deng Chuan
	d. Timesheet. Everyone is to update the timesheet for individual parts.	All
	e. Powerpoint Slides. The group aims to complete the presentation slide by 8 Mar 24.	All
	f. Next Meeting. Next meeting is scheduled on 7 Mar 24 from 6-8pm.	-

Secretary & Facilitator: Deng Chuan Chang

Meeting 2 on 7 March 2024 (Physical + Online Meeting)

Agenda 2

S/N	Items
1.	Attendance Update
2.	Confirmation of Meeting 1 Minutes
3.	Actions/ Updates from Previous Meetings
4.	Next Week Task & Allocation
5.	Any other business

Secretary: Deng Chuan Chang

Minutes 2

S/N	Items	Action By
Item 1: Attendance Update		
1.	The attendance for the group meeting are as follows: Present: - Alexis Balayre - Cheng Chung Yueh - Chang Deng Chuan (Secretary) - Julien Godfroy - Lyderic Faure - Majuran Chandrakumar - Marcell Gyorei Absent: Nil Apologies: Nil	-
Item 2: Confirmation of Meeting 1 Minutes		
2.	No changes to previous minutes.	
Item 3: Actions/ Updates from Previous Meetings		
3.	The outstanding items 2 and 3 are combined under this section. Each of the item is presented during the meeting. a. PySINDy. (Item closed) o Input format can be data frame or in matric.	Chung Yueh

	b. Genetic Algorithms. (Item closed) o To optimise the PySINDy equations and parameters.	Chung Yueh
	c. Machine Learning Packages. (Item closed) o PyTorch is more developed than Tensor Flow in research algorithms. The group can explore PyTorch Lightning for more features.	Alexis
	d. Proximal Policy Optimization (PPO). o Input format same as CNN, it can be numeric values or pixels. This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	Majuran
	e. Convolutional neural network (CNN). (Item closed) o Input format same as PPO, it can be numeric values or pixels	Majuran
	f. Principal Component Analysis (PCA). (Item closed) o After data processing (cleaning), PCA can be done prior to PySINDy or Machine Learning.	Majuran
	g. Learning Processes. (Item closed) o Observational bias can be controlled for CFD results input to machine learning. o Inductive bias is embedded into CNN technique.	Deng Chuan
	h. Physics-Informed Neural Network (PINN). (Item closed) o This approach may generate a better result as compared to CNN.	Julien
	i. Gantt Chart. (Item is not closed) - Computational Fluid Dynamics (to be updated) - Machine Learning (updated) - Software Development (to be updated)	Lyderic Deng Chuan Marcell
	j. Unified Modelling Language (UML). (Item is not closed) - Understand and develop the deliverables for the project - Create a basic flow charts - Educate the group on UML	Marcell
	k. Documentations. (Item closed) - Establishing the processes for different documentations - Create a basic report for subsequent updates	Marcell
Item 4: Next Week Task & Allocation		
4.	There is a list of papers to be read. a. Consistency and Correctness of Requirements for Artificial Intelligence Systems	Julien

	b. An MDE Method for Improving Deep Learning Dataset Requirements Engineering using Alloy and UML	Majuran
	c. An Analysis of Data Quality Requirements for Machine Learning Development Pipelines Frameworks	Deng Chuan
	d. Machine Learning Operations (MLOps) Overview, Definition, and Architecture	Chung Yueh
	e. Requirements Engineering in Machine Learning Projects	Julien
	f. A Core Conceptual Modeling construct for Capturing Complexity	Alexis
	g. Pairing Conceptual Modeling with Machine Learning	CIDA
	h. Requirements and software engineering for automotive perception systems_ an interview study	CIDA
	CIDA: a. Understand Sparse Matrix Algorithms b. Prepare of DNS data (cleaning & formatting) c. Conduct PCA Analysis (different parameters) d. Develop basic CNN & PINN algorithms e. Understand inputs from CNN & PINN to PySINDy CED: a. Generate meshes on CFD b. Understand the parameters of the meshes SETC: a. Unified Modelling Language (UML). - Understand and develop the deliverables for the project - Create a basic flow charts - Educate the group on UML	
Item 5: Any Other Business		
5.	a. Agenda and Minutes documents. To be updated and keep tracked, including timesheet will be sent on Friday afternoon.	Deng Chuan
	b. Timesheet. Everyone is to update the timesheet for individual parts.	-
	c. Next Meeting. Next meeting is scheduled on 11 Mar 24 from 2-4pm.	All

Secretary & Facilitator: Deng Chuan Chang

Project Meeting 3 on 11 March 2024 (Physical Meeting)

Agenda 3

S/N	Items
1.	Attendance Update
2.	Team Bonding Session
3.	Confirmation of Meeting 2 Minutes
4.	Actions/ Updates from Previous Meetings
5.	Task & Allocation
6.	Any other business

Secretary: Deng Chuan Chang

Minutes 3

S/N	Items	Action By
Item 1: Attendance Update		
1.	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Cheng Chung Yueh - Chang Deng Chuan (Secretary) - Julien Godfroy - Lyderic Faure - Majuran Chandrakumar - Marcell Gyorei Absent: Nil Apologies: Nil	-
Item 2: Team Bonding Session		
2.	The group has agreed that any issues surfaced within the group should be resolved internally with the team members. If the group fails to resolve the issues, it will then be escalated to the professors. a. Meeting Dates. (To implement)	All

	○ The group proposed to consolidate the weekly documents for submission earlier to avoid last minute changes. This is to ensure every member has sufficient time to review it. For this week, the documents will be updated by Thursday morning prior to the discussion. The group agreed. b. Timesheet. (Item is closed) ○ Deng Chuan had clarified that, he volunteered to compile the previous timesheets, and did not adjust to any member's timesheet. Marcell emphasised that timesheet is individual's responsibility, other member should not edit each other timesheet even though an error is spotted. The group agreed. c. Project Management. (Item is not closed) ○ Professors highlighted that everyone who participated in Management of Technology has the capability to assume the role as a Project Manager/ IC and can rotate within the team. Marcell highlighted that is a parttime student and should not take up this role as he has work commitment. The one who compiles the overall Gantt Chart, submission of materials have not been resolved and is to be discussed in Meeting 4. d. Risk Review Meeting. ○ The group has agreed to include the risk review meeting in the agenda when required. Marcell is assigned to lead and educate the group on risk review. e. Administrative Works. ○ The group has agreed to rotate the organiser and facilitator roles to provide learning opportunities for everyone.	All All Marcell -
Item 3: Confirmation of Meeting 2 Minutes		
3.	No changes to previous minutes.	
Item 4: Actions/ Updates from Previous Meetings		
4.	Below are the outstanding tasks to be followed up from previous meetings. a. Gantt Chart. (Item is not closed) - Professors requested to see the overall Gantt chart by next supervisor meeting 2 (18 Mar 24). Each individual working group to update their own specific portions by Thursday. b. Unified Modelling Language (UML). (Item is not closed) - Understand and develop the deliverables for the project - Create a basic flow charts - Educate the group on UML	Marcell Deng Chuan Lyderic Marcell

	- I have not committed myself into (Marcell) to stick to UML or any pre-defined methodology/design language for gathering requirements. The studied software engineering methodologies don't mention industry standards for ML (CRISP-DM) - so they don't directly apply into our group project. Will focus instead of following a SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. Their lesson learnt could give us directions and feedback loop when and where our effort should be placed in terms of software quality for a successful ML project. c. Proximal Policy Optimization (PPO). (Item is not closed) o This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li. d. List of Paper to be Read. (Item is closed) - The eight research papers that were consolidated were not useful in developing of machine learning and thus, will not be followed up by the CIDA students.	Majuran
Item 5: Task & Allocation		
5.	CED: (Tuesday Deliverables) a. Automate the mesh generation using the script by creating a MATLAB code. Make it so it can affect parameters inside the mesh. b. Set-up the simulation using the generated mesh. c. Compute the simulation through MATLAB and extract results. d. Gantt Chart to be updated. **CIDA: (Tuesday Deliverables)** a. Data cleaning for the DNS data. b. Updating of CIDA Gantt Chart from today's discussion. c. Updating Agenda & record of Minutes. d. How to use PCA output to interface with Classic Neural Networks using Pytorch. e. How to use PCA output to interface with PySINDy input. f. Prepare visualisation using PCA output from the DNS data. g. How to integrate PCA data with a Random Forest model. h. Propose alternate approach other than PINN for this project. i. Literature review on t-distributed stochastic neighbor embedding (T-SNE) j. Implementing T-SNE to visualize DNS data	Lyderic Majuran Deng Chuan Deng Chuan Alexis & Julien Chung Yueh Majuran & Julien Julien Deng Chuan & Julien Chung Yueh Majuran & Chung Yueh

	SETC: (Dates of completion is not specified) a. Extend ML development process diagram with inputs and outputs - interfaces of processing steps. b. Define high level of requirements, breaking down the requirements for aim1 and aim2 (options for user reqs., functional and non-functional reqs.) c. Read research paper to break down the data pipeline - N2 chart or domain model (Thursday) d. Quality plan - Traceability Matrix e. Risk Register - Extend it if needed f. Gantt chart to be updated	Marcell
Item 6: Any Other Business		
6.	a. Supervisors' Meeting 1. The following points are the feedback from the meeting. - For project status descriptions, ensure the collaboration/ connectivity between the working groups are included. It should not be a compilation of individual's group work. - Submission of Gantt Chart for next meeting to ensure the group is heading in the correct direction. - Objectives should be refined to be more specific for each week. - Expect to see diagrams or data to allow further comments by the professors. - Consider the output requirements as an additional task. - Finding alternate approach other than PINNS. - Ensure there are a few solutions to address a problem. b. Next Meeting. Next meeting is scheduled on 12 Mar 24 from 6-7.30pm.	All - All All All TBC All - All

Secretary & Facilitator: Deng Chuan Chang

Meeting 4 on 12 March 2024 (Physical Meeting)

Agenda 4

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 3 Minutes	5
3.	Actions/ Updates from Previous Meetings	45
4.	Task & Allocation	10
5.	Any other business	5
	Total:	62 mins

Secretary: Julien Godfroy

Minutes 4

S/N	Items	Action By
Item 1: Attendance Update		
1.	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Cheng Chung Yueh - Chang Deng Chuan - Julien Godfroy (Secretary) - Majuran Chandrakumar - Marcell Gyorei Absent: Lyderic Faure Apologies: Lyderic Faure	-
Item 2: Confirmation of Meeting 3 Minutes		
2.	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3.	The following are the outstanding tasks from different meetings. The tasks are as follows: <u>Meeting 2</u>: a. <u>GANNT Chart</u>. (Item is closed) Deng Chuan volunteered to compile the GANTT, provided the rest of the Gantt Chart is submitted on Wed night. Everyone can adjust accordingly. Marcell proposes to upgrade the chart with more advanced GANTT conventions. If there is no changes, it will be submitted as it is.	Deng Chuan Marcell

	b. Project Management. (Item is closed) Deng Chuan will take up the compiling and uploading of documents till anyone volunteers.	Deng Chuan
	c. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts.	Marcell
	d. Proximal Policy Optimization (PPO). (Item is not closed) This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	Majuran
	Meeting 3: **CED:** (Tuesday Deliverables) a. Generate mesh script file (item closed) b. Set-up the simulation using the generated mesh. (item is not closed) The environment setting for the simulation requires more time to understand and adjust. Lyderic will update the progress on Thursday meeting 5.	Lydéric
	CIDA: (Tuesday Deliverables) a. Data cleaning for the DNS data. (item closed) b. Updating of CIDA Gantt Chart. (item closed) c. Updating Agenda & record of Minutes. (item closed) d. How to use PCA output to interface with Classic Neural Networks using Pytorch (item closed) e. How to use PCA output to interface with PySINDy input (item closed) f. Prepare visualisation using PCA output from the DNS data (item closed) g. How to integrate PCA data with a Random Forest model. (item closed) h. Propose alternate approach other than PINN for this project -> Deep Learning or Transformer (to be confirmed again) i. Literature review on t-distributed stochastic neighbour embedding (T-SNE) (item closed) j. Implementing T-SNE to visualize DNS data (item closed) Overall, all the different machine learning methods have been well studied to understand method meets our requirements. Some proof-of-concepts (PoC) have been implemented as well. The global learning cycle is well understood.	Majuran Deng Chuan Deng Chuan Alexis Chung Yueh Majuran Julien Julien & Deng Chuan Chung Yueh Majuran & Chung Yueh
	SETC: a. Extend ML development process diagram with inputs and outputs - interfaces of processing steps. - next sprint (will be represented differently - ML standards to be followed)	Marcell

	b. Define high level of requirements, breaking down the requirements for aim1 and aim2 (options for user reqs., functional and non-functional reqs.) - next sprint (ML standards to be followed) c. Read research paper to break down the data pipeline - N2 chart or domain model - next sprint (ML standards to be followed) d. Quality plan - Traceability Matrix - next sprint or the sprint after (after requirements has been defined) e. Risk Register - extend it if needed next sprint (to be updated with ML specific items - such as assumptions, unknowns - those risks are already being mitigated by actions such as assignment of standard tasks into the CRISP-DM and ML specific agile process) f. Gantt chart to be updated - next sprint - that can be more detailed from SE side but currently it stays as it is g. Process is being chopped into standard task items: such as task backlog lists, checklist, consideration, design guidelines. h. Kanban Board - standard task items to be added i. Evaluation of tasks items, completion of the checklists, and the business-requirement analysis is ongoing There were no deliverables presented on Tue Meeting 4 as the tasks listed were huge. Deng Chuan suggested that these tasks should be split into smaller manageable tasks to keep the work going. Tasks that were listed should try or be completed prior to each meeting.	
Item 4: Task & Allocation		
4.	The following tasks were discussed and allocated to individual as follows. CED: a. Automate the mesh generation using the script by creating a MATLAB code. Make it so it can affect parameters inside the mesh. b. Set-up the simulation using the generated mesh. c. Compute the simulation through MATLAB and extract results. d. CED Gantt Chart to be updated. CIDA: a. Understanding more in depth how does PySindy work. Is it using DL method inside? b. Clarify with Dr Jun Li on PySindy to PINN implementation. c. Request more data from Lyderic or/and Tamas Jozsa.	Lydéric Alexis All All

	d. Update project description e. Compile the GANTT CHART f. Write the Minutes & next agenda SETC: No updates General Comment: All the CED, CIDA and SETC GANTT Charts will be based on current CIDA format. The format as be accessed through (General>Project Plan>programme.xlsx> CIDA (2) sheet). Marcell proposed to compile a list of backlogs using the best tool to track and allocate the different tasks (excel, Miro, Trello, etc.).	Deng Chuan Deng Chuan Julien Marcell Marcell & Julien
Item 5: Any other business		
5.	a. Risk review and user requirements were discussed previously to include into the meeting. However, the materials were not ready and will be delayed further. b. Next meeting is scheduled on Thursday: 7.30am - 8.30am (1hr) as proposed in meeting 3. c. The group has agreed that 8pm will be the cut off timing to edit the submission materials. Deng Chuan volunteered to compile and upload the material by 10pm. d. The group has agreed the sprint length (working time) as 1 week (from Thursday to Thursday).	

Secretary & Facilitator: Julien Godfroy

Meeting 5 on 14 March 2024 (Physical Meeting)

Agenda 5

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 4 Minutes	2
3.	Actions/ Updates from Previous Meetings	26
4.	Task & Allocation	15
5.	Any other business (Discuss the responses from the professors)	20
	Total:	65 mins

Secretary: Chung-Yueh Cheng

Minutes 5

S/N	Items	Action By
Item 1: Attendance Update		
	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng (Secretary) - Chang Deng Chuan - Julien Godfroy - Majuran Chandrakumar - Marcell Gyorei Absent: Lyderic Faure Apologies: Lyderic Faure	-
Item 2: Confirmation of Meeting 4 Minutes		
2.	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3.	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	
	Meeting 3: **CED:** (Tuesday Deliverables) a. Set-up the simulation using the generated mesh. (item is not closed)	Lyderic
	Meeting 4: **CED:** a. Set-up the simulation using the generated mesh. (to be updated) b. Automate the mesh generation using the script by creating a MATLAB code. Make it so it can affect parameters inside the mesh. (to be updated) c. Set-up the simulation using the generated mesh. (to be updated) d. Compute the simulation through MATLAB and extract results (to be updated) e. Gantt Chart to be updated. (Item is closed)	Lyderic
	CIDA: (Thursday Deliverables) a. Propose Alternate Approach to PINN. (to be confirmed again). b. Understanding more in depth how does PySINDy work. Is it using Deep Learning method? (Item is closed) c. Clarify with Dr Jun Li on PySINDy to PINN implementation. d. Request more data from Lyderic and Dr Tamas for machine learning. Alternatively, find online data to use. (to be updated) e. Update project description (Item is closed) f. Compile the GANTT CHART (Item is closed) g. Write the Minutes & next Agenda () **SETC:** Tasks are in progress.	Deng Chuan & Julien Chung Yueh Majuran Chung Yueh Deng Chuan Deng Chuan Chung Yueh Marcell
Item 4: Task & Allocation		
4.	**CED:** a. No updates, to be updated on Monday. **CIDA:** a. Reading the reference paper of turbulence modelling b. How to use PySINDy's output to GA, and need to know what's the output of GA. c. Understand how to compare output of PySINDy and PINN's output d. Discuss features library of PySINDy	Lydéric All All All All

	SETC: a. No updates, to be updated on Monday.	Marcell
Item 5: Any other business		
5.	a. Tracking of Tasks. Trello cards will be used to track the tasks, how many meeting per week, all the documents are needed for each meeting. b. Project Status Description. For the project status description, Marcell requested to attach his appendix for weekly submission. Deng Chuan noted and will submit accordingly. c. Next meeting is schedule on next Monday 18 Mar at 9:30 am. d. Deng Chuan emphasized that deliverables list for each meeting should be met, and if required, break the task into manageable steps to complete. Otherwise, the group will be delayed in progression. On Monday, the group will discuss if this week implementation of more meetings and timings are feasible and sustainable. e. Administrative. Deng Chuan commented that it is important to provide individual updates for each meeting prior to the next meeting as it will be hard to follow up on the actions or items that are not closed. Everyone to be more proactive.	Marcell Marcell All All

Secretary & Facilitator: Chung-Yueh Cheng

Meeting 6 on 18 March 2024 (Physical Meeting)

Agenda 6

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 4 Minutes	3
3.	Actions/ Updates from Previous Meetings	15
4.	Task & Allocation	10
5.	Any other business (Discuss the responses from the professors)	10
	Total:	40 mins

Secretary: Deng Chuan Chang

Minutes 6

S/N	Items	Action By
Item 1: Attendance Update		
1.	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng - Chang Deng Chuan (Secretary) - Lydéric Faure - Julien Godfroy - Majuran Chandrakumar - Marcell Gyorei Absent: Apologies:	-
Item 2: Confirmation of Meeting 5 Minutes		
2.	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3.	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li. Meeting 4: CIDA: a. Propose Alternate Approach to PINN. (to be confirmed again). a. Clarify with Dr Jun Li on PySINDy to PINN implementation. (Item closed) b. Request more data from Lyderic and Dr Tamas for machine learning. Alternatively, find online data to use. (to be updated) **SETC:** Tasks are in progress. Meeting 5. a. Reading the reference paper of turbulence modelling. (Item is closed) b. How to use PySINDy's output to GA, and need to know what's the output of GA. (Item is not closed) c. Understand how to compare output of PySINDy and PINN's output. (Item is not closed) d. Discuss features library of PySINDy. (Item is not closed) These tasks will be closed after results from PINNS and PySINDy are proved to be accurate.	Julien Chung Yueh & Majuran Majuran Chung Yueh Marcell
Item 4: Task & Allocation		
4.	**CED: (this week)** – tasks from meeting 4&5 a. Set-up the simulation using the generated mesh. b. Automate the mesh generation using the script by creating a MATLAB code. Make it so it can affect parameters inside the mesh. c. Set-up the simulation using the generated mesh. d. Compute the simulation through MATLAB and extract results **CIDA: (this week)** a. Generate results taking DNS to PINNS b. Generate results taking DNS to Transformer c. Generate results from PySINDy d. Archive all the work done (literature and methodology) e. Explore other feature extraction approaches	Lydéric Alexis Julien Majuran & Cheng Yueh All Deng Chuan

	SETC: (this week) – tasks from meeting 4&5 Tasks are in progress	Marcell
Item 5: Any other business		
5.	a. Supervisors' Meeting. It was mentioned that the group can commence into compiling the literature review and methodology. Discussions on adopting various approaches need to be documented down as the rationale behind the decision making may be forgotten.	All
	b. Group Meeting. Based on last week review of three meetings, the group decided to organise for two proper meetings (Monday & Wednesday), with discussions on other days. Next meeting is schedule on next Wednesday 20 Mar at 6pm.	All
	c. Administrative. Majuran will compile for this week submission materials.	All

Secretary & Facilitator: Deng chuan Chang

Meeting 7 on 18 March 2024 (Physical Meeting)

Agenda 6

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 6 Minutes	3
3.	Actions/ Updates from Previous Meetings	34
4.	Task & Allocation	13
5.	Any other business (Discuss the responses from the professors)	17
	Total:	69 mins

Secretary: Majuran Chandrakumar

Minutes 7

S/N	Items	Action By
Item 1: Attendance Update		
1.	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng - Chang Deng Chuan - Lydéric Faure - Julien Godfroy - Majuran Chandrakumar (Secretary) - Marcell Gyorei Absent: Apologies:	-
Item 2: Confirmation of Meeting 6 Minutes		
2.	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3.	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	
	Meeting 4:	
	CIDA:	
	a. Propose Alternate Approach to PINN – Transformer (item is closed)	Julien
	b. Request more data from Lyderic and Dr Tamas for machine learning. Alternatively, find online data to use. (item is closed)	Majuran & Chung Yueh
	SETC:	
	Tasks are in progress.	Marcell
	Meeting 5.	
	CIDA:	
	a. How to use PySINDy's output to GA, and need to know what's the output of GA. (Item is not closed) b. Understand how to compare output of PySINDy and PINN's output. (Item is not closed) c. Discuss features library of PySINDy. (Item is closed)	
	Meeting 6:	
	CED: – tasks from meeting 4&5 a. Set-up the simulation using the generated mesh. (Item is closed) b. Automate the mesh generation using the script by creating a MATLAB code. Make it so it can affect parameters inside the mesh. (Delay) c. Compute the simulation through MATLAB and extract results. (working)	Lydéric
	CIDA:	
	a. Generate results taking DNS to PINNS (Item is closed)	Alexis
	b. Generate results taking DNS to Transformer (Item is closed)	Julien
	c. Generate results from PySINDy (Item is closed)	Chung Yueh & Majuran
	d. Archive all the work done (literature and methodology) (Item is not closed)	All

	e. Explore other feature extraction approaches (Item is not closed)	Deng Chuan
	SETC: (this week) • Risk Register - extend it (done) • Process to be chopped into standard task items: such as task backlog lists, checklist, consideration, design guidelines (done - item closed) • Kanban Board - standard task items to be added (done - item closed) • Kanban board - evaluation of tasks items, completion of the checklists (done - item closed) • Kanban board - consulted, handover, guidance for the team (done - item closed)	Marcell
Item 4: Task & Allocation		
4.	**CED:** a. Compute the simulation through MATLAB and extract results.	Lydéric
	CIDA: a. Optimizing the PINN model b. Optimizing the Transformer model c. Simulate and evaluate the PySINDy model d. Optimizing the PySINDy model e. Archive all the work done (literature and methodology) f. Explore and implement other feature extraction approaches	Alexis Julien Chung Yueh & Majuran Chung Yueh & Majuran All Deng Chuan
	SETC: • Asked by lecturers to focus on testing; the following sub-tasks are identified: a. S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (to do) b. S2.3.2 - define test environment (backlog) c. S2.3.3 - setup the testing tools (backlog) d. PS4.1 - define business requirements phase 1 (backlog) e. PS10.1 - define functional, no-functional requirements phase 1 (backlog)	Marcell

	• Quality plan - traceability matrix (backlog) • Gantt chart to be updated (backlog) • Extend ML development process diagram with inputs and outputs - interfaces of processing steps (to be decomposed) • Read research paper to break down the data pipeline - N2 chart or domain model (to be decomposed)	
Item 5: Any other business		
5.	a. Questions: • Discuss the time steps used for each data point. • What is the format input to ANSYS for validating simulation data? Is it UDF? (needs to be discussed on next meeting) • Why do we need data such as $\frac{d^2v}{dy^2}$ to enrich our model? (needs to be discussed on next meeting)	All
	b. Group Meeting: Based on the review of this week's meetings, the group found it challenging to complete the assigned tasks as well as to edit the documents to be submitted, such as the project status description, by Wednesday evening. Therefore, the group decided to hold two formal meetings (Monday and Thursday evenings), with discussions on other days. The group agreed that all documents to be submitted must be completed by Thursday 12:00 PM so that there is time to review them during Thursday evening's meeting. The next meeting is scheduled on next Monday 25 Mar.	All
	c. Risk Register: The group agreed to allocate time to update the risk register each week.	All
	d. Trello: The group agreed to trial Trello next week to see if it could improve work efficiency.	All

Secretary & Facilitator: Majuran Chandrakumar

Meeting 8 on 25 March 2024 (Physical + Online Meeting)

Agenda 8

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 7 Minutes	3
3.	Actions/ Updates from Previous Meetings	15
4.	Task & Allocation	10
5.	Any other business	10
	Total:	40 mins

Secretary: Deng Chuan Chang

Minutes 8

S/N	Items	Action By
Item 1: Attendance Update		
1	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng - Chang Deng Chuan (Secretary) - Lydéric Faure - Julien Godfroy - Majuran Chandrakumar - Marcell Gyorei Absent: Apologies:	-
Item 2: Confirmation of Meeting 7 Minutes		
2	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	
	Meeting 5. **CIDA:** a. How to use PySINDy's output to GA, and need to know what's the output of GA. (Item is not closed) b. Understand how to compare output of PySINDy and PINN's output. (Item is not closed)	Chung Yueh & Majuran
	Meeting 6: **CED:** – tasks from meeting 4&5 a. Automate the mesh generation using the script by creating a MATLAB code. Make it so it can affect parameters inside the mesh. (Item is not closed)	Lyderic
	CIDA: a. Archive all the work done (literature and methodology) (Item is not closed) b. Explore other feature extraction approaches (Item is not closed)	CIDA Deng Chuan
	Meeting 7: **CED:** a. Compute the simulation through MATLAB and extract results. (Item is not closed)	Lyderic
	CIDA: a. Optimizing the PINN model (Item is not closed) b. Optimizing the Transformer model (item is not closed) c. Simulate and evaluate the PySINDy model (Item is closed) d. Optimizing the PySINDy model (Item is not closed) e. Archive all the work done (literature and methodology) f. Explore and implement other feature extraction approaches (Item is closed)	Alexis Julien Chung Yueh & Majuran Deng Chuan
	SETC: a. Asked by lecturers to focus on testing; the following sub-tasks are identified:	Marcell

	• S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (ongoing, item is not closed) • S2.3.2 - define test environment (to do this week) • S2.3.3 - setup, catch-up, experiment with the tools (for offline & online testing) (Item is not closed) • PS4.1 - define business requirements phase 1 (Deng Chuan will assist) • PS10.1 - define functional, no-functional requirements phase 1 (Item is not closed) b. Quality plan - traceability matrix (to do this week) c. Gantt chart to be updated (Item is not closed) d. Extend ML development process diagram with inputs and outputs - interfaces of processing steps (to be decomposed, item is not closed) e. Read research paper to break down the data pipeline - N2 chart or domain model (to be decomposed, item is not closed)	
Item 4: Task & Allocation		
4.	**CED:** a. Compute the simulation through MATLAB and extract results. b. Drafting of CED literature review **CIDA:** a. PySINDy resolve the time step issues b. Implement PySINDy with irregular time series c. Optimise PySINDy inputs with PINN & Transformer d. Drafting of PySINDy literature review e. Drafting of PCA, T-SNE literature review f. Optimise Transformer implementation with same inputs for PINN implementation g. Drafting of Transformer literature review h. Optimise PINN implementation to validate the result i. Generalization of PINN model to various channel flow j. Drafting of PINN literature review k. Define basic user requirements l. Develop basic software interfaces to run user' input and ML algo (complete by next week) m. Drafting of other feature extraction literature review	Lydéric Chung Yueh & Majuran Julien Julien Alexis Alexis Alexis Deng Chuan Deng Chuan Deng Chuan

	SETC: a. Asked by lecturers to focus on testing; the following sub-tasks are identified: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc • S2.3.2 - define test environment b. Quality plan - traceability matrix	Marcell
Item 5: Any other business		
5.	a. Group Meeting: The next meeting is scheduled on Friday 6pm on Team's meeting. b. Risk Register: The group agreed to allocate time on Friday's meeting to update the risk register. c. Trello: The group started the task allocations on Trello for tracking and improving work efficiency.	All All -

Secretary & Facilitator: Deng Chuan Chang

Meeting 9 on 29 March 2024 (Physical + Online Meeting)

Agenda 9

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 8 Minutes	3
3.	Actions/ Updates from Previous Meetings	15
4.	Task & Allocation	10
5.	Any other business	10
	Total:	40 mins

Secretary: Julien Godfroy

Minutes 9

S/N	Items	Action By
Item 1: Attendance Update		
1	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng - Chang Deng Chuan - Julien Godfroy (Secretary) - Majuran Chandrakumar - Marcell Gyorei Absent: Lydéric Faure Apologies: Lydéric Faure	-
Item 2: Confirmation of Meeting 8 Minutes		
2	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	
	Meeting 5.	
	CIDA:	
	a. How to use PySINDy's output to GA, and need to know what's the output of GA. (Item is not closed) b. Understand how to compare output of PySINDy and PINN's output. (Item is not closed)	Chung Yueh & Majuran
	Meeting 6:	
	CED: – tasks from meeting 4&5 a. Automate the mesh generation using the script by creating a MATLAB code. Make it so it can affect parameters inside the mesh. (Item is not closed)	Lyderic
	CIDA: a. Archive all the work done (literature and methodology) (Item is not closed) b. Explore other feature extraction approaches (Item is not closed)	CIDA Deng Chuan
	Meeting 7:	
	CED: a. Compute the simulation through MATLAB and extract results. (Item is not closed)	Lyderic
	CIDA: a. Optimizing the PINN model (Item is not closed) b. Optimizing the Transformer model (item is not closed) c. Simulate and evaluate the PySINDy model (Item is closed) d. Optimizing the PySINDy model (Item is not closed) e. Archive all the work done (literature and methodology) f. Explore and implement other feature extraction approaches (Item is closed)	Alexis Julien Chung Yueh & Majuran Deng Chuan
	SETC: a. Asked by lecturers to focus on testing; the following sub-tasks are identified: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (ongoing, item is not closed) • S2.3.2 - define test environment (to do this week)	Marcell

	• S2.3.3 - setup, catch-up, experiment with the tools (for offline & online testing) (Item is not closed) • PS4.1 - define business requirements phase 1 (Deng Chuan will assist) • PS10.1 - define functional, no-functional requirements phase 1 (Item is not closed) b. Quality plan - traceability matrix (to do this week) c. Gantt chart to be updated (Item is not closed) d. Extend ML development process diagram with inputs and outputs - interfaces of processing steps (to be decomposed, item is not closed) e. Read research paper to break down the data pipeline - N2 chart or domain model (to be decomposed, item is not closed)	
	Meeting 8: **CED:** a. Compute the simulation through MATLAB and extract results. b. Drafting of CED literature review	Lydéric
	CIDA: a. PySINDy resolve the time step issues (item not closed) b. Implement PySINDy with irregular time series (item closed) c. Optimise PySINDy inputs with PINN & Transformer (item closed) d. Drafting of PySINDy literature review (item not closed) e. Drafting of PCA, T-SNE literature review (item closed)	Chung Yueh & Majuran
	f. Optimise Transformer implementation with same inputs for PINN implementation (item closed) g. Drafting of Transformer literature review (item closed) h. Optimise PINN implementation to validate the result (item not closed) i. Generalization of PINN model to various channel flow (item closed) j. Drafting of PINN literature review (item not closed) k. Define basic user requirements (item not closed) l. Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed) m. Drafting of other feature extraction literature review (item not closed)	Julien Julien Alexis Alexis Deng Chuan Deng Chuan Deng Chuan
	SETC: a. Asked by lecturers to focus on testing; the following sub-tasks are identified:	Marcell

	• S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) • S2.3.2 - define test environment (Doing) • Latex report template - incorporate requested changes; add a guidance (Done) • Reproducibility (Doing) b. Quality plan - traceability matrix (To Do)	
Item 4: Task & Allocation		
4.	**CED:** a. Compute the simulation through MATLAB and extract results. b. Drafting of CED literature review **CIDA:** a. PySINDy resolve the time step issues (item not closed) b. Implement PySINDy with irregular time series (item not closed) c. Optimise PySINDy inputs with PINN & Transformer (item not closed) d. Drafting of PySINDy literature review (item not closed) **e.** Drafting of Transformer literature review (item not closed) **f.** Optimise PINN implementation to validate the result (item not closed) **g.** Generalization of PINN model to various channel flow (item not closed) **h.** Drafting of PINN literature review (item not closed) **i.** Define basic user requirements (item not closed) **j.** Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed) **k.** Drafting of other feature extraction literature review (item not closed) **SETC:** a. Asked by lecturers to focus on testing; the following sub-tasks are identified: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) • S2.3.2 - define test environment (Doing) • Reproducibility (Doing) b. Quality plan - traceability matrix (To Do)	Lydéric Chung Yueh & Majuran Julien Julien Alexis Deng Chuan Deng Chuan Deng Chuan Marcell

Item 5: Any other business		
5.	<u>What do you think about Trello?</u> It needs to be updated more often but overall, the tool is well designed for this project. We'll keep using it. <u>Next meetings</u> 2 meetings next week: • Tuesday • Thurday evening	All All

Secretary & Facilitator: Deng Chuan Chang

Meeting 10 on 2 April 2024 (Physical + Online Meeting)

Agenda 10

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 9 Minutes	10
3.	Actions/ Updates from Previous Meetings	20
4.	Task & Allocation	8
5.	Any other business	5
	Total:	45 mins

Secretary: Chung Yueh Cheng

Minutes 10

S/N	Items	Action By
Item 1: Attendance Update		
1	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng (Secretary) - Chang Deng Chuan - Lydéric Faure - Julien Godfroy - Majuran Chandrakumar Absent: Marcell Gyorei Apologies: Marcell Gyorei	-
Item 2: Confirmation of Meeting 9 Minutes		
2	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	
	Meeting 6: **CED:** – tasks from meeting 4&5	Lydéric
	CIDA: a. Archive all the work done (literature and methodology) (Item is not closed)	CIDA
	Meeting 7: **SETC:** a. Asked by lecturers to focus on testing; the following sub-tasks are identified: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (ongoing, item is not closed) • S2.3.2 - define test environment (to do this week) • S2.3.3 - setup, catch-up, experiment with the tools (for offline & online testing) (Item is not closed) • PS4.1 - define business requirements phase 1 (Deng Chuan will assist) • PS10.1 - define functional, no-functional requirements phase 1 (Item is not closed) b. Quality plan - traceability matrix (to do this week) c. Gantt chart to be updated (Item is not closed) d. Extend ML development process diagram with inputs and outputs - interfaces of processing steps (to be decomposed, item is not closed) e. Read research paper to break down the data pipeline - N2 chart or domain model (to be decomposed, item is not closed)	Marcell
	Meeting 8: **CED:** a. Compute the simulation through MATLAB and extract results. b. Drafting of CED literature review	Lydéric
	CIDA: a. PySINDy resolve the time step issues (item not closed) b. Drafting of PySINDy literature review (item not closed) c. Optimise PINN implementation to validate the result (item not closed)	Chung Yueh & Majuran Julien

	d. Drafting of PINN literature review (item not closed)	Julien Alexis
	e. Define basic user requirements (item not closed) f. Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed)	Deng Chuan
	g. Drafting of other feature extraction literature review (item not closed)	Deng Chuan
	SETC: a. Asked by lecturers to focus on testing; the following sub-tasks are identified: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) • S2.3.2 - define test environment (Doing) • Reproducibility (Doing) b. Quality plan - traceability matrix (To Do)	Marcell
	Meeting 9. **CED:** a. Extract data from new simulation model b. Run simulations through HPC **c.** Drafting of CED literature review	Lydéric
	CIDA: a. PySINDy resolve the time step issues (item not closed) b. Implement PySINDy with irregular time series (item not closed) c. Optimise PySINDy inputs with PINN & Transformer (item not closed) d. Drafting of PySINDy literature review (item not closed)	Chung Yueh & Majuran
	e. Drafting of Transformer literature review (item not closed)	Julien
	f. Optimise PINN implementation to validate the result (item not closed) g. Generalization of PINN model to various channel flow (item not closed) h. Drafting of PINN literature review (item not closed)	Alexis
	i. Define basic user requirements (item not closed) j. Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed) k. Drafting of other feature extraction literature review (item not closed)	Deng Chuan

	SETC: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) • S2.3.2 - define test environment (Doing) • Reproducibility (Doing) • Latex report template - incorporate requested changes; add a guidance (Done) • Quality plan - traceability matrix (To Do)	Marcell
Item 4: Task & Allocation		
4.	CED: a. Extract data from new simulation model b. Run simulations through HPC **c.** Drafting of CED literature review Next meeting, Lyderic will generate the correct simulations. CIDA: a. Optimise PINN implementation to validate the result (item not closed) b. Generalization of PINN model to various channel flow (item not closed) c. Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed) d. Drafting of other feature extraction literature review (item not closed) SETC: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) • S2.3.2 - define test environment (Doing) • Reproducibility (Doing) • Quality plan - traceability matrix (Doing)	Lydéric Alexis Deng Chuan Marcell
Item 5: Any other business		
5.	<u>Next meetings</u> meeting next week: • Thursday evening 6pm Do words documents for this week in apa 7 and use [ ] to label the numbers Finish all documents	All All All

Secretary & Facilitator: Chung Yueh Cheng

Meeting 11 on 4 April 2024 (Physical + Online Meeting)

Agenda 11

S/N	Items	Time Allocated
1.	Attendance Update	1
2.	Confirmation of Meeting 10 Minutes	25
3.	Actions/ Updates from Previous Meetings	18
4.	Task & Allocation	3
5.	Any other business	0
	Total:	47 mins

Secretary: Chung Yueh Cheng

Minutes 11

S/N	Items	Action By
Item 1: Attendance Update		
1	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng (Secretary) - Chang Deng Chuan - Lydéric Faure - Julien Godfroy - Majuran Chandrakumar - Marcell Gyorei Absent: Apologies:	-
Item 2: Confirmation of Meeting 9 Minutes		
2	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
3	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li.	
	Meeting 8: **CED:** a. Compute the simulation through MATLAB and extract results.	Lydéric
	CIDA: a. Optimise PINN implementation to validate the result (item not closed) b. Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed)	CIDA
	Meeting 9. **CED:** a. Run simulations through HPC	Lydéric
	CIDA: a. Optimise PINN implementation to validate the result (item not closed) b. Generalization of PINN model to various channel flow (item not closed) c. Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed)	Alexis Alexis Deng Chuan
	Meeting 10. CED: a. Run simulations through HPC	Lydéric
	CIDA: a. Optimise PINN implementation to validate the result (item not closed) b. Generalization of PINN model to various channel flow (item not closed) c. Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed)	Alexis Alexis Deng Chuan
	SETC: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) • S2.3.2 - define test environment (Doing) • Reproducibility (Doing) • Quality plan - traceability matrix (Doing)	Marcell

Item 4: Task & Allocation		
4.	CED: a. Run simulations through HPC CIDA: a. Implement the prototype of PySINDy model from the data through PCA **b.** Optimise PINN implementation to validate the result (item not closed) **c.** Generalization of PINN model to various channel flow (item not closed) **d.** Develop basic software interfaces to run user' input and ML algo (complete by next week) (item not closed) SETC: • S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) • S2.3.2 - define test environment (Doing) • Reproducibility (Doing) • Quality plan - traceability matrix (Doing)	Lydéric Majuran& Chung-Yueh Alexis Alexis Deng Chuan Marcell
Item 5: Any other business		
5.	Next meetings meeting next week: • Monday Morning 9:30am	All

Secretary & Facilitator: Chung Yueh Cheng

Meeting 12 on 8 April 2024 (Physical + Online Meeting)

Agenda 12

S/N	Items	Time Allocated
1.	Attendance Update	1
2.	Confirmation of Meeting 11 Minutes	25
3.	Actions/ Updates from Previous Meetings	18
4.	Task & Allocation	3
5.	Any other business	0
	Total:	47 mins

Secretary: Lyderic Faure

Minutes 12

S/N	Items	Action By
Item 1: Attendance Update		
	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng (Secretary) - Chang Deng Chuan - Lydéric Faure - Majuran Chandrakumar - Marcell Gyorei Absent: Julien Godfroy Apologies: Julien Godfroy	-
Item 2: Confirmation of Meeting 11 Minutes		
	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li. Meeting 8: **CED:** a. Compute the simulation through MATLAB and extract results (Item is closed, it will not be pursued) Meeting 11: **CED:** a. Run simulations through HPC (Item not closed) **CIDA:** a. Implement the prototype of PySINDy model from the data through PCA (Item closed) b. Optimise PINN implementation to validate the result (item not closed) c. Generalization of PINN model to various channel flow (item closed) d. Develop basic software interfaces to run user' input and ML algo (in progress) (item not closed) **SETC:** a. S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (To Do) b. S2.3.2 - define test environment (Doing) c. Reproducibility (Doing) d. Quality plan - traceability matrix (Doing)	Lydéric Lydéric Majuran& Chung-Yueh Alexis Alexis Deng Chuan Marcell
Item 4: Task & Allocation		
	CED: a. Data comparison k-w and RST model comparison with DNS and ML model first ouptuts **CIDA:** a. Develop basic software interfaces to run user' input and PySINDy **All:** a. Include Literature Review for current solutions to resolve RANS closure problem b. Poster Draft 1 c. Group Presentation Draft 1 d. Conclusion of Transformer in report e. Comparison of RANS, PINNs, & PySINDy	Lydéric Deng Chuan Alexis & Julien Lydéric & Deng Chuan Chung Yueh & Majuran Julien All
Item 5: Any other business		
	• Next meetings. It is scheduled on Thursday 11/04, 6:00 pm at B37. • Poster and presentation draft 1 to be done by 11/04	All

Secretary & Facilitator: Lyderic Faure & Majuran Chandrakumar

Meeting 13 on 11 April 2024 (Physical + Online Meeting)

Agenda 13

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 12 Minutes	3
3.	Actions/ Updates from Previous Meetings	20
4.	Task & Allocation	5
5.	Any other business	0
	Total:	30 mins

Secretary: Deng Chuan Chang

Minutes 13

S/N	Items	Action By
Item 1: Attendance Update		
	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng - Chang Deng Chuan (Secretary) - Lydéric Faure - Marcell Gyorei - Julien Godfroy Absent: Apologies: Majuran Chandrakumar	-
Item 2: Confirmation of Meeting 12 Minutes		
	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
	Meeting 2: a. Unified Modelling Language (UML). (Item is not closed) Propose to use SE methodology developed for on standard machine learning (CRISP-DM, design thinking, MLOps, etc). It includes ML specific tasks backlogs, checklist and guidance documents prepared by industry experts. b. Proximal Policy Optimization (PPO). (Item is not closed)	Marcell Majuran

	This optimization technique can be done after CNN, to clarify on Monday with Dr Jun Li. Meeting 11: **CIDA:** a. Implement the prototype of PySINDy model from the data through PCA (Item closed) b. Generalization of PINN model to various channel flow (item closed) c. Develop basic software interfaces to run user' input and ML algo (in progress) (item not closed) Meeting 12. CED: a. Run simulations through HPC (doing) (Item not closed) b. Data comparison, k-w and RST model comparison with DNS and ML model first ouptuts (working on it) (Item not closed) CIDA: a. Develop basic software interfaces to run user' input and PySINDy (Item closed) b. Optimise PINN implementation to validate the result (item not closed) **SETC: (To be updated)** a. Quality plan - Outline (Done) **All:** a. Include Literature Review for current solutions to resolve RANS closure problem (Item not closed) b. Conclusion of Transformer in report (Item not closed) c. Poster Draft 1 (It is closed) d. Group Presentation Draft 1 (Item is closed) e. Comparison of RANS, PINNs, & PySINDy (Item not closed)	Majuran & Chung-Yueh Alexis Deng Chuan Lydéric Majuran & Deng Chuan Alexis Marcell Julien Julien Deng Chuan & Lyderic Majuran & Chung-Yueh All
Item 4: Task & Allocation		
	CED: a. Run simulations through HPC (doing) (Item not closed) b. Data comparison, k-w and RST model comparison with DNS and ML model first ouptuts (working on it) (Item not closed) **CIDA:** a. Improve basic software interfaces to run user' input and PySINDy b. Optimise PINN implementation to validate the result (item not closed)	Lydéric Majuran & Deng Chuan Alexis

	SETC: (To be updated) a. S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (Doing) b. S2.3.2 - define test environment (Doing) c. Reproducibility (Doing) d. Traceability matrix (Doing) **All:** a. Include Literature Review for current solutions to resolve RANS closure problem (Item not closed) b. Conclusion of Transformer in report (Item not closed) **c.** Comparison of RANS, PINNs, & PySINDy (Item not closed)	Marcell Julien Julien All
Item 5: Any other business		
	• Next meetings. It is scheduled on Monday after supervisors' meeting 15/04. • Poster design and presentation draft 2 to be done by 16/04, draft 3 to be done on 17/04 • UML will be discussed on Monday if to close the item. • PPO will be discussed on Monday if to close the item.	All

Secretary & Facilitator: Deng Chuan Chang

Meeting 14 on 17 April 2024 (Physical + Online Meeting)

Agenda 14

S/N	Items	Time Allocated
1.	Attendance Update	2
2.	Confirmation of Meeting 13 Minutes	3
3.	Actions/ Updates from Previous Meetings	20
4.	Task & Allocation	5
5.	Any other business	0
	Total:	30 mins

Secretary: Deng Chuan Chang

Minutes 14

S/N	Items	Action By
Item 1: Attendance Update		
	The attendance for the group meeting is as follows: Present: - Alexis Balayre - Chung-Yueh Cheng - Chang Deng Chuan (Secretary) - Lydéric Faure - Majuran Chandrakumar - Marcell Gyorei - Julien Godfroy Absent: - Apologies: -	-
Item 2: Confirmation of Meeting 12 Minutes		
	No changes to previous minutes.	
Item 3: Actions/Updates from Previous Meetings		
	Meeting 2: a. Unified Modelling Language (UML). ML specific tasks backlogs, checklist and guidance documents prepared by industry experts has been done. (Item closed) b. Proximal Policy Optimisation (PPO).	Marcell Majuran

	PPO will not be pursued as the ML models include optimisation functions; thus, further optimisation is not required. (Item closed)	
	Meeting 12: **CIDA:** a. Optimise PINN implementation to validate the result (item closed) **All:** a. Include Literature Review for current solutions to resolve RANS closure problem (Item closed) b. Conclusion of Transformer in report (Item closed) c. Comparison of RANS, PINNs, & PySINDy (Item closed)	Alexis Julien Julien Majuran & Deng Chuan Alexis
	Meeting 13: **CED:** a. Run simulations through HPC (Item closed) b. Data comparison, k-w and RST model comparison with DNS and ML model first ouptuts (Item closed) **CIDA:** a. Improve basic software interfaces to run user' input with PySINDy & PINNs (item closed) b. Develop cloud-base app (Item closed) c. Optimise PINN implementation to validate the result (item closed) **SETC:** a. S2.3.1 - break down ml testing in general and specific to this project - offline, online etc (Item closed) b. S2.3.2 - define test environment (Item closed) c. Reproducibility (Item closed) d. Traceability matrix (Item closed)	Lydéric Majuran & Deng Chuan Julien Alexis Marcell
Item 4: Task & Allocation		
	• Poster & Presentations. Submitted on CANVAS. Preparation of slides for Q&A and rehearsals to be done next week. • StreamLit App. The interfaces will continue to be improved till submission. • Technical Work. Archiving of codes for submission. • Report. The group report is agreed to be in words document, and the scientific paper will be in LATEX format.	All Majuran/ Julien Alexis All
Item 5: Any other business		
	• This will be the last meeting update.	

Secretary & Facilitator: Deng Chuan Chang