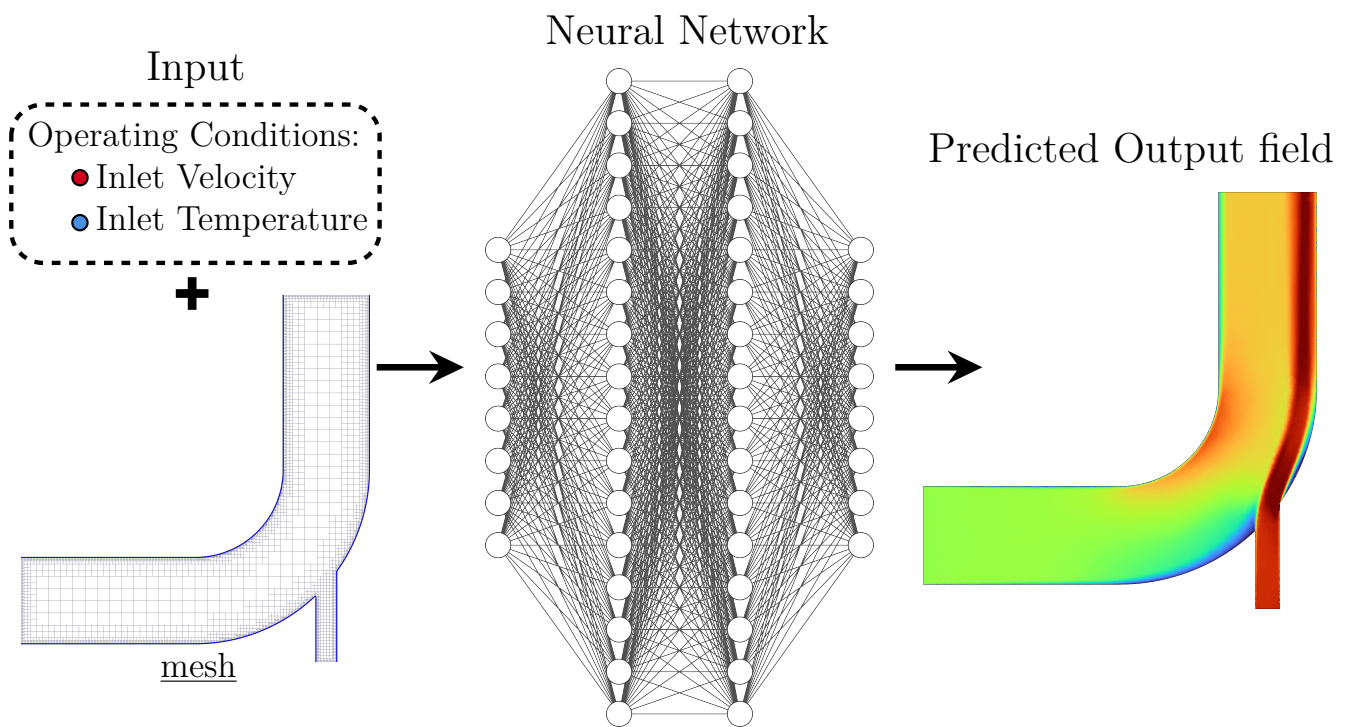




# Assessment of AI Tools for CFD Acceleration and Generalization

Master's thesis in Applied Mechanics

KARTHIK TAPASI HIMANTH

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2026  
[www.chalmers.se](http://www.chalmers.se)





MASTER'S THESIS IN APPLIED MECHANICS

# Assessment of AI Tools for CFD Acceleration and Generalization

KARTHIK TAPASI HIMANTH



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences  
Division of Fluid Dynamics  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2026

Assessment of AI Tools for CFD Acceleration and Generalization  
KARTHIK TAPASI HIMANTH

© KARTHIK TAPASI HIMANTH 2026.

Supervisor: Patrik Sondell, FS Dynamics AB  
Examiner: Håkan Nilsson, Mechanics and Maritime Sciences

Master's Thesis 2026  
Department of Mechanics and Maritime Sciences  
Chalmers University of Technology  
SE-412 96 Gothenburg  
Sweden  
Telephone +46 31 772 1000

Cover: Schematic of the surrogate modelling framework, showing mesh and operating conditions as inputs to a neural network for prediction of flow field quantities.

Typeset in L<sup>A</sup>T<sub>E</sub>X  
Gothenburg, Sweden 2026

Assessment of AI Tools for CFD Acceleration and Generalization  
KARTHIK TAPASI HIMANTH  
Department of Mechanics and Maritime Sciences  
Division of Fluid Dynamics  
Chalmers University of Technology

## Abstract

Computational Fluid Dynamics (CFD) simulations are widely used in engineering applications to study fluid flow and heat transfer problems, but high fidelity simulations usually require large computational time and resources. In recent years, artificial intelligence (AI) and machine learning (ML) based surrogate models have shown strong potential for predicting CFD results using data driven approaches. These methods are becoming increasingly important in industrial applications where faster design evaluation and reduced simulation cost are desired. This thesis investigates different AI based tools and frameworks for CFD applications, including the commercial framework Altair PhysicsAI and the open source framework NVIDIA PhysicsNeMo. Different AI architectures such as Graph Neural Networks (GNNs), MeshGraphNet, Physics Informed Neural Networks (PINNs), and Fourier Neural Operators (FNOs) are studied using CFD datasets generated from ANSYS Fluent and STAR-CCM+. Two study cases are considered in the present work, namely a mixing elbow flow problem and a CPU cooling problem involving conjugate heat transfer. The main objective of the work is to evaluate the prediction accuracy, generalization capability, computational efficiency, and practical applicability of AI based surrogate models for engineering CFD problems.

Keywords: Machine Learning, Surrogate Modelling, PhysicsAI, PhysicsNeMo, Graph Neural Networks, MeshGraphNet, Fourier Neural Operator, CFD Modelling



# Acknowledgements

I would like to express my sincere gratitude to FS Dynamics AB for providing me with the opportunity to carry out this master's thesis project. I would especially like to thank my manager, Stewart Whitworth, for giving me the opportunity to work on this project. I would also like to express my sincere appreciation to my supervisor Patrik Sondell at FS Dynamics AB for his continuous guidance, support, encouragement, and valuable feedback throughout the project. His support and technical insights have been extremely valuable during the course of this work. I would also like to thank my examiner Håkan Nilsson at Chalmers University of Technology for his guidance and feedback throughout the project. In addition, I would like to thank Elias Chalhoub, CFD Engineer at FS Dynamics AB, for his support during the initial stages of the project, especially regarding the cluster and HPC related workflows.

Karthik Tapasi Himanth, Gothenburg, June 2026



# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

|      |                                 |
|------|---------------------------------|
| AI   | Artificial Intelligence         |
| ANN  | Artificial Neural Network       |
| CFD  | Computational Fluid Dynamics    |
| CPU  | Central Processing Unit         |
| DOE  | Design of Experiments           |
| FFT  | Fast Fourier Transform          |
| FNO  | Fourier Neural Operator         |
| FVM  | Finite Volume Method            |
| GNN  | Graph Neural Network            |
| GPU  | Graphics Processing Unit        |
| HPC  | High Performance Computing      |
| LHS  | Latin Hypercube Sampling        |
| ML   | Machine Learning                |
| MLP  | Multi Layer Perceptron          |
| PDE  | Partial Differential Equation   |
| PINN | Physics Informed Neural Network |
| RANS | Reynolds Averaged Navier Stokes |
| SST  | Shear Stress Transport          |
| VRM  | Voltage Regulator Module        |
| KPI  | Key Performance Indicator       |





# Nomenclature

Below is the nomenclature of indices, parameters, and variables that have been used throughout this thesis.

## Indices

|        |                            |
|--------|----------------------------|
| $i, j$ | Spatial coordinate indices |
| $k$    | Turbulence related index   |
| $t$    | Time                       |

## Parameters

|               |                                   |
|---------------|-----------------------------------|
| $\rho$        | Fluid density                     |
| $\mu$         | Dynamic viscosity                 |
| $\mu_t$       | Turbulent viscosity               |
| $p$           | Pressure                          |
| $u_i$         | Velocity component in i-direction |
| $x_i$         | Spatial coordinate in i-direction |
| $\tau_{ij}$   | Viscous stress tensor             |
| $\sigma_{ij}$ | Total stress tensor               |
| $S_{ij}$      | Strain rate tensor                |
| $k$           | Turbulent kinetic energy          |
| $\omega$      | Specific dissipation rate         |
| $E$           | Total energy                      |
| $H$           | Total enthalpy                    |
| $q_j$         | Heat flux                         |

|               |                                        |
|---------------|----------------------------------------|
| $\delta_{ij}$ | Kronecker delta                        |
| $f_i$         | Body force component                   |
| $P_k$         | Production of turbulent kinetic energy |
| $D_\omega$    | Cross diffusion term                   |
| $\phi$        | General flow variable                  |
| $\phi'$       | Fluctuating component of flow variable |
| $\hat{y}$     | Predicted neural network output        |
| $w_{ij}$      | Neural network weight                  |
| $b_j$         | Neural network bias                    |

## Boundary Condition Parameters

|             |                            |
|-------------|----------------------------|
| $U_{large}$ | Velocity at large inlet    |
| $U_{small}$ | Velocity at small inlet    |
| $T_{large}$ | Temperature at large inlet |
| $T_{small}$ | Temperature at small inlet |
| CFM         | Cubic feet per minute      |

## Operators

|                |                             |
|----------------|-----------------------------|
| $\partial$     | Partial derivative operator |
| $\nabla$       | Gradient operator           |
| $\nabla \cdot$ | Divergence operator         |

# Contents

|                                                                |           |
|----------------------------------------------------------------|-----------|
| <b>List of Acronyms</b>                                        | <b>ix</b> |
| <b>Nomenclature</b>                                            | <b>xi</b> |
| <b>1 Introduction</b>                                          | <b>1</b>  |
| 1.1 Background . . . . .                                       | 1         |
| 1.2 Aim . . . . .                                              | 2         |
| 1.3 Limitations . . . . .                                      | 3         |
| <b>2 Theory</b>                                                | <b>5</b>  |
| 2.1 Governing Equations of Fluid Flow . . . . .                | 5         |
| 2.1.1 Continuity Equation . . . . .                            | 5         |
| 2.1.2 Momentum Equation . . . . .                              | 6         |
| 2.1.3 Energy Equation . . . . .                                | 7         |
| 2.1.4 Turbulence Modelling: $k - \omega$ SST Model . . . . .   | 7         |
| 2.2 Finite Volume Method . . . . .                             | 8         |
| 2.3 Machine Learning . . . . .                                 | 9         |
| 2.3.1 Neural Networks . . . . .                                | 9         |
| 2.3.2 Surrogate Modelling . . . . .                            | 10        |
| 2.3.3 Surrogate Modelling using PhysicsAI . . . . .            | 10        |
| 2.3.3.1 Model Architectures . . . . .                          | 11        |
| 2.3.3.2 Hyperparameter Configuration . . . . .                 | 11        |
| 2.3.4 NVIDIA PhysicsNeMo Framework . . . . .                   | 12        |
| 2.3.4.1 Basic Surrogate Models using Multi Layer Perceptrons   | 12        |
| 2.3.4.2 Graph Neural Networks and MeshGraphNet . . . . .       | 13        |
| 2.3.4.3 Physics Informed Neural Networks and Physics In-       |           |
| formed GNNs . . . . .                                          | 14        |
| 2.3.5 Fourier Neural Operators . . . . .                       | 14        |
| <b>3 Methods</b>                                               | <b>17</b> |
| 3.1 Overall Methodology . . . . .                              | 17        |
| 3.2 Study 1: Mixing Elbow Case . . . . .                       | 18        |
| 3.2.1 Geometry . . . . .                                       | 19        |
| 3.2.2 Mesh Generation . . . . .                                | 19        |
| 3.2.3 Solver Settings . . . . .                                | 21        |
| 3.2.4 Boundary Conditions . . . . .                            | 22        |
| 3.2.4.1 Design of Experiments and Dataset Generation . . . . . | 22        |

|           |                                                               |           |
|-----------|---------------------------------------------------------------|-----------|
| 3.3       | Study 2: CPU Cooling Case . . . . .                           | 22        |
| 3.3.1     | Geometry . . . . .                                            | 23        |
| 3.3.2     | Mesh Generation . . . . .                                     | 24        |
| 3.3.3     | Solver Settings . . . . .                                     | 25        |
| 3.3.4     | Boundary Conditions and Dataset Generation . . . . .          | 26        |
| 3.4       | Altair PhysicsAI Methodology . . . . .                        | 26        |
| 3.5       | NVIDIA PhysicsNeMo Methodology . . . . .                      | 27        |
| 3.6       | AI Architectures Investigated . . . . .                       | 28        |
| 3.6.1     | Basic Surrogate Model (MLP) . . . . .                         | 28        |
| 3.6.2     | MeshGraphNet . . . . .                                        | 29        |
| 3.6.3     | Physics Informed GNN . . . . .                                | 30        |
| 3.6.4     | Fourier Neural Operator . . . . .                             | 31        |
| 3.7       | Evaluation Metrics . . . . .                                  | 32        |
| <b>4</b>  | <b>Results and Discussion</b>                                 | <b>35</b> |
| 4.1       | Mixing Elbow Case . . . . .                                   | 35        |
| 4.1.1     | Overview of the Mixing Elbow Study . . . . .                  | 35        |
| 4.1.2     | Two Dimensional Mixing Elbow Results . . . . .                | 37        |
| 4.1.2.1   | Altair PhysicsAI Results . . . . .                            | 37        |
| 4.1.2.1.1 | Within Range Unseen Case: . . . . .                           | 38        |
| 4.1.2.1.2 | Extrapolated Cases: . . . . .                                 | 40        |
| 4.1.2.2   | NVIDIA PhysicsNeMo Results . . . . .                          | 43        |
| 4.1.2.2.1 | Within Range Unseen Case: . . . . .                           | 43        |
| 4.1.2.2.2 | Extrapolation Cases: . . . . .                                | 48        |
| 4.1.3     | Three Dimensional Mixing Elbow Results . . . . .              | 53        |
| 4.1.3.1   | Altair PhysicsAI Results . . . . .                            | 53        |
| 4.1.3.2   | NVIDIA PhysicsNeMo Results . . . . .                          | 57        |
| 4.1.3.2.1 | Within range unseen case: . . . . .                           | 57        |
| 4.1.3.2.2 | Extrapolated Cases: . . . . .                                 | 60        |
| 4.2       | CPU Cooling Case . . . . .                                    | 65        |
| 4.2.1     | Overview of the CPU Cooling Study . . . . .                   | 65        |
| 4.2.2     | MeshGraphNet Results . . . . .                                | 66        |
| <b>5</b>  | <b>Conclusion</b>                                             | <b>71</b> |
| 5.1       | Altair PhysicsAI . . . . .                                    | 71        |
| 5.2       | NVIDIA PhysicsNeMo . . . . .                                  | 72        |
|           | <b>Bibliography</b>                                           | <b>75</b> |
| <b>A</b>  | <b>Appendix</b>                                               | <b>I</b>  |
| A.1       | Two Dimensional Mixing Elbow: Extrapolation Cases . . . . .   | I         |
| A.2       | Three Dimensional Mixing Elbow: Extrapolation Cases . . . . . | III       |
| A.3       | Structured Grid Conversion for FNO Workflow . . . . .         | IV        |
| A.4       | CPU Cooling Case . . . . .                                    | VI        |
| A.5       | Python Scripts Used in NVIDIA PhysicsNeMo Workflow . . . . .  | VII       |
| A.5.1     | MLP Based Surrogate Model Workflow . . . . .                  | VIII      |
| A.5.2     | GNN Model Workflow . . . . .                                  | XI        |

|       |                                   |    |
|-------|-----------------------------------|----|
| A.5.3 | GNN PINN Model Workflow . . . . . | XV |
| A.5.4 | FNO Model Workflow . . . . .      | XX |



# 1

## Introduction

### 1.1 Background

Computational Fluid Dynamics (CFD) is an important tool in modern industrial product development, enabling detailed analysis of complex flow and multiphysics phenomena while reducing reliance on expensive physical prototyping. It is widely applied across automotive, aerospace, energy, and process industries, where accurate prediction of flow behavior directly influences performance, safety, and design efficiency. Despite its maturity, high-fidelity CFD remains computationally expensive, particularly for complex geometries, turbulent flows, and coupled physics problems, resulting in long turnaround times that limit the number of feasible design iterations [1, 2]. These limitations have motivated sustained research efforts toward reduced order modeling and surrogate based acceleration techniques.

Recent advances in artificial intelligence (AI) and machine learning (ML) have significantly reshaped this landscape by enabling data-driven surrogate models that approximate CFD solutions at a fraction of the computational cost. Physics informed learning methods further enhance these approaches by embedding physical constraints into the training process, improving robustness and interpretability [3]. In parallel with academic developments, several commercial platforms have emerged that integrate AI directly into established CAE workflows. Notably, **ANSYS SimAI** and **Altair PhysicsAI** employ deep learning and geometric learning techniques to learn from high fidelity simulation data and deliver rapid predictions for new designs [4, 5]. Their growing adoption reflects a broader industrial shift toward AI-assisted simulation and design exploration.

The effectiveness of such tools has been demonstrated across a range of industrial and engineering applications. **SimAI** has been successfully applied to accelerate CFD based mixing simulations [6], automotive crash and pedestrian protection analyses [7, 8], and benchmarking efforts in airfoil aerodynamics through large scale ML competitions [9]. Similarly, **Altair PhysicsAI** has been used to accelerate durability and structural analyses in automotive components [10], as well as multiphysics optimization of UAV wings and winglets [11, 12]. While these studies demonstrate substantial speed ups and promising accuracy, they also indicate that model performance strongly depends on training data quality, problem complexity, and generalization beyond the training domain. As a result, a systematic and comparative

evaluation of AI based CFD acceleration tools under controlled and industrially relevant conditions remains an open and important challenge.

In parallel with academic developments, several open source and industrial frameworks have emerged for AI assisted engineering simulations. NVIDIA PhysicsNeMo has recently gained attention as an open source framework for developing physics informed and data driven AI models for computational engineering applications. The framework provides scalable workflows for surrogate modelling, operator learning, transformer based architectures, and benchmarking of AI models for CFD applications [13, 14]. Recent studies using PhysicsNeMo have demonstrated promising results in automotive aerodynamics, including high fidelity surface pressure prediction, aerodynamic benchmarking, and generalization across complex vehicle geometries [13, 15]. These developments show that AI based surrogate models can significantly reduce computational cost while maintaining reasonable predictive accuracy. However, previous studies also indicate that model performance strongly depends on factors such as training data quality, geometry complexity, model architecture, and generalization capability beyond the training domain [13, 15]. As a result, systematic evaluation of AI based CFD acceleration methods under controlled and industrially relevant conditions remains an important research challenge.

## 1.2 Aim

The aim of this master’s thesis is to systematically evaluate AI based surrogate modeling tools for accelerating Computational Fluid Dynamics (CFD) simulations in industrially relevant applications. The project focuses on assessing commercial AI platforms, specifically Altair PhysicsAI, with respect to prediction accuracy, generalization capability, training efficiency, and integration into industrial simulation workflows.

A central objective is to investigate how reliably these AI based tools can reproduce high-fidelity CFD and multiphysics simulation results across different application domains and levels of physical complexity. Building on prior application studies in CFD acceleration, automotive crash and durability analysis, and aerospace design optimization [6–8, 10, 11], the study aims to identify which classes of simulation problems are most suitable for AI-assisted modeling. Particular emphasis is placed on quantifying the trade offs between computational speed up and predictive accuracy, as highlighted in recent literature on machine learning for fluid mechanics and data-driven modeling [1, 2].

In addition to commercial solutions, the project aims to extend the evaluation to open-source physics informed learning frameworks, such as NVIDIA Physics NeMo, in order to assess the potential benefits and limitations of physics informed and open-source approaches relative to proprietary tools. This comparison is intended to provide insight into issues of transparency, flexibility, and generalization, which are critical for industrial adoption of AI based simulation technologies [3, 9]. The CFD datasets used for training and validation of the AI models will be generated using



established industrial solvers, namely **ANSYS Fluent** and **STAR CCM+**. By starting from relatively simple flow configurations and progressively increasing the complexity toward more industrially representative cases, the project aims to deliver practical, evidence based guidance for the responsible and effective use of AI based CFD acceleration tools in industrial engineering practice.

### 1.3 Limitations

This project focuses on the evaluation of AI based surrogate models for accelerating CFD simulations and is therefore subject to several limitations. First, the scope of the study is restricted to a limited number of flow and multiphysics configurations that are representative of industrial applications but do not span the full spectrum of possible CFD problems. In addition, the CFD simulations are limited to specific numerical settings and modelling approaches, including the use of the SST  $k-\omega$  turbulence model, and therefore do not include a comparative assessment of different turbulence models. Highly transient flows, multiphase phenomena, and chemically reacting flows are also excluded, as these cases typically require substantially larger datasets and more specialized modeling strategies, as highlighted in the machine learning for fluid mechanics literature [1, 2].

The evaluation of commercial AI tools is limited to the workflows and capabilities provided by **Altair PhysicsAI** at the time of the study. As these platforms rely on proprietary architectures and training pipelines, internal model structures and optimization strategies are not accessible to the user [5]. Consequently, the assessment is based on observed predictive performance and workflow integration rather than a detailed inspection of the underlying learning algorithms. This limitation is consistent with previously published application studies of **PhysicsAI**, which primarily report performance outcomes rather than internal model interpretability [11].

Furthermore, the performance of AI based surrogate models is inherently dependent on the quality and diversity of the training data. While established industrial solvers such as **ANSYS Fluent** and **STAR CCM+** are used to generate consistent and reliable datasets, the trained models are expected to perform best within the bounds of the defined parameter space. Extrapolation beyond the training domain may lead to reduced accuracy, a limitation that has been reported in both academic benchmarking studies and industrial case studies [3, 9]. Finally, the comparison with open-source physics informed learning frameworks is limited to a proof of concept level. Constraints related to computational resources and project duration restrict the extent of large scale hyperparameter optimization and extensive architecture tuning. In particular, limited GPU memory availability during the project restricted the use of larger model architectures and high resolution three dimensional training cases. As a result, findings related to open-source approaches should be interpreted as indicative rather than exhaustive and are intended primarily to provide contextual comparison with commercial AI based simulation tools.



# 2

## Theory

This chapter presents the theoretical foundation for the present work, which combines Computational Fluid Dynamics (CFD) and data-driven modelling techniques. The governing equations describing fluid flow and heat transfer are first introduced, followed by turbulence modelling using the  $k - \omega$  Shear Stress Transport (SST) model. The numerical framework used to solve these equations, namely the Finite Volume Method (FVM), is then described.

In addition, fundamental concepts of machine learning are presented, with a focus on neural networks and their application as surrogate models. These models are trained on CFD generated data to approximate complex flow behaviour, enabling faster predictions compared to conventional simulations. Together, these theoretical components form the basis for the coupled CFD and machine learning approach adopted in this study.

### 2.1 Governing Equations of Fluid Flow

The governing equations describing fluid flow and heat transfer are derived from the fundamental conservation laws of mass, momentum and energy. These equations are expressed in the form of partial differential equations and are commonly referred to as the Navier Stokes equations. They describe the motion of viscous fluids and form the foundation for fluid flow analysis [16]. Due to their complexity, analytical solutions are generally not possible for practical engineering problems, and numerical methods such as Computational Fluid Dynamics (CFD) are used to obtain approximate solutions.

#### 2.1.1 Continuity Equation

The continuity equation expresses the conservation of mass for a fluid element. It states that the rate of change of mass within a control volume is balanced by the net mass flux across its boundaries [16]. The continuity equation is written as

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0, \quad (2.1)$$

where  $\rho$  is the fluid density,  $u_j$  is the velocity component in the  $j$ -direction, and  $x_j$  represents the spatial coordinates. The first term represents the temporal rate of change of density, while the second term represents the convective transport of mass. This equation ensures that mass is conserved locally at every point in the flow field. For incompressible flows, where the density is constant, the continuity equation reduces to

$$\frac{\partial u_j}{\partial x_j} = 0. \quad (2.2)$$

This condition implies that the velocity field is divergence-free. Physically, this means that there is no volumetric expansion or compression of the fluid, and the inflow and outflow of any control volume must be equal. The continuity equation is solved together with the momentum and energy equations to fully describe the fluid flow and thermal behaviour in CFD simulations.

### 2.1.2 Momentum Equation

The momentum equation describes the motion of a fluid under the influence of various forces and is derived from Newton's second law applied to a fluid element [16]. It states that the rate of change of momentum is equal to the sum of pressure, viscous and body forces acting on the fluid. In tensor notation, the momentum equation is given by

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho f_i, \quad (2.3)$$

where  $u_i$  is the velocity component,  $p$  is the pressure,  $\tau_{ij}$  is the viscous stress tensor, and  $f_i$  represents body forces. The first term represents the temporal change of momentum, while the remaining terms account for convective transport, pressure forces, viscous forces, and external body forces.

The total stress tensor  $\sigma_{ij}$  can be decomposed into pressure and viscous contributions as

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij}. \quad (2.4)$$

For a Newtonian fluid, the viscous stress tensor is expressed as

$$\tau_{ij} = 2\mu S_{ij} - \frac{2}{3}\mu \frac{\partial u_k}{\partial x_k} \delta_{ij}, \quad (2.5)$$

where  $\mu$  is the dynamic viscosity and the strain rate tensor is defined as

$$S_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right). \quad (2.6)$$

Substituting the above relations gives the complete stress tensor. For incompressible flows, where  $\frac{\partial u_j}{\partial x_j} = 0$ , the equations reduce to the classical Navier Stokes form, which governs the balance of inertial, pressure and viscous forces in fluid motion.

### 2.1.3 Energy Equation

The energy equation is derived from the first law of thermodynamics, which states that the rate of change of energy within a system is equal to the net rate of heat addition and the work done on the system [16]. In fluid flow, heat transfer is typically represented through conductive heat flux, while work is associated with forces acting on the fluid element.

In tensor notation, the energy equation can be expressed as

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial(\rho E u_j)}{\partial x_j} + \frac{\partial p u_j}{\partial x_j} - \frac{\partial(u_i \tau_{ij})}{\partial x_j} + \frac{\partial q_j}{\partial x_j} = \rho \epsilon + S_u, \quad (2.7)$$

where  $E$  is the total energy,  $q_j$  is the heat flux,  $\epsilon$  represents internal heat sources,  $\tau_{ij}$  is the stress tensor, and  $S_u$  is an additional energy source term. The different terms in the equation represent, respectively, the temporal change of energy, convective transport, pressure work, viscous work and heat conduction. Together, they describe the evolution of thermal energy within the flow field. For practical applications, it is often convenient to express energy in terms of enthalpy. The total energy can then be written as

$$E = H - \frac{p}{\rho}, \quad (2.8)$$

where  $H$  is the total enthalpy. This formulation is commonly used in CFD simulations involving heat transfer, as it simplifies the coupling between thermal and flow variables.

### 2.1.4 Turbulence Modelling: $k - \omega$ SST Model

Most engineering flows are turbulent in nature, characterized by chaotic fluctuations in velocity and pressure over a wide range of spatial and temporal scales. Directly resolving all turbulent scales using Direct Numerical Simulation (DNS) is computationally expensive and not feasible for practical applications. Therefore, turbulence modelling approaches such as Reynolds-Averaged Navier–Stokes (RANS) are commonly used, where the instantaneous flow variables are decomposed into mean and fluctuating components. This decomposition is expressed as

$$\phi = \bar{\phi} + \phi', \quad (2.9)$$

where  $\bar{\phi}$  represents the time-averaged component and  $\phi'$  represents the fluctuating

component. Substituting this decomposition into the Navier Stokes equations and applying time-averaging results in additional terms known as Reynolds stresses, which require modelling to close the system of equations [16].

The Reynolds stresses are commonly modelled using the Boussinesq hypothesis, which relates them to the mean velocity gradients through an eddy viscosity concept

$$-\overline{\rho u'_i u'_j} = \mu_t \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \rho k \delta_{ij}, \quad (2.10)$$

where  $\mu_t$  is the turbulent viscosity and  $k$  is the turbulent kinetic energy, defined as

$$k = \frac{1}{2} \overline{u'_i u'_i}, \quad (2.11)$$

This approximation enables closure of the RANS equations by introducing turbulence models that provide transport equations for additional variables.

Among the commonly used two-equation models, the  $k - \omega$  Shear Stress Transport (SST) model combines the advantages of the  $k - \omega$  model near walls and the  $k - \epsilon$  model in the free stream. This hybrid formulation improves the prediction of adverse pressure gradients and flow separation, making it suitable for complex internal flows such as those encountered in the present study [16].

The transport equation for turbulent kinetic energy  $k$  is given by

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho u_j k)}{\partial x_j} = P_k - \beta^* \rho k \omega + \frac{\partial}{\partial x_j} \left[ (\mu + \sigma_k \mu_t) \frac{\partial k}{\partial x_j} \right], \quad (2.12)$$

and the transport equation for the specific dissipation rate  $\omega$  is

$$\frac{\partial(\rho \omega)}{\partial t} + \frac{\partial(\rho u_j \omega)}{\partial x_j} = \alpha \frac{\omega}{k} P_k - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[ (\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + D_\omega, \quad (2.13)$$

where  $P_k$  represents the production of turbulent kinetic energy and  $D_\omega$  accounts for cross-diffusion effects. The SST model employs blending functions to smoothly transition between the near-wall and free-stream regions, ensuring improved accuracy in predicting wall-bounded turbulent flows.

## 2.2 Finite Volume Method

The governing equations described in the previous section consist of a set of coupled, nonlinear partial differential equations. Due to their complexity, analytical solutions

are generally not feasible for practical engineering problems. Therefore, numerical methods are employed to obtain approximate solutions.

In this work, the Finite Volume Method (FVM) is used to discretize and solve the governing equations. In FVM, the computational domain is divided into a finite number of control volumes, and the integral form of the conservation equations is applied to each control volume. This ensures strict conservation of mass, momentum and energy over the domain. The discretization process converts the governing equations into a system of algebraic equations, which are solved iteratively to obtain the flow field. The FVM approach is widely used in CFD due to its robustness and suitability for complex geometries [16].

## 2.3 Machine Learning

Machine learning (ML) is a data-driven approach used to model complex relationships between input and output variables. In contrast to physics-based models, where governing equations explicitly describe the system, ML models learn these relationships directly from data. This makes them particularly useful for approximating nonlinear systems where conventional numerical simulations are computationally expensive.

In supervised learning, the model is trained using a dataset consisting of input-output pairs. The objective is to learn a mapping between the inputs and outputs such that the model can generalize to unseen data. This is achieved by minimizing a loss function through iterative optimization, allowing the model to improve its predictive capability.

### 2.3.1 Neural Networks

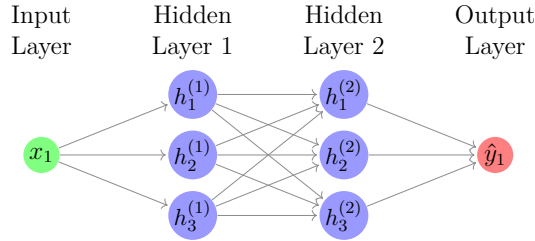
Artificial neural networks (ANNs) consist of interconnected neurons arranged in layers, typically including an input layer, one or more hidden layers, and an output layer, as illustrated in Figure 2.1. Each neuron receives inputs from the previous layer, applies a weighted sum, and passes the result through a nonlinear activation function. This layered structure enables the network to approximate complex nonlinear relationships between inputs and outputs [17].

The output of a neuron  $j$  is given by

$$y_j = f \left( \sum_{i=1}^n w_{ij} x_i + b_j \right), \quad (2.14)$$

where  $x_i$  are the inputs,  $w_{ij}$  are the weights,  $b_j$  is the bias term, and  $f(\cdot)$  is the activation function. The inclusion of nonlinear activation functions allows neural networks to capture complex patterns that cannot be represented using linear models alone [17]. During training, the network parameters are adjusted using optimization algorithms such as gradient descent, where the error between predicted and actual

outputs is minimized. The gradients required for this optimization are computed using the backpropagation algorithm, enabling efficient training of deep networks [17]. Once trained, the neural network can be used as a surrogate model to rapidly predict outputs for new input conditions.



**Figure 2.1:** Structure of a feedforward neural network.

### 2.3.2 Surrogate Modelling

Surrogate modelling is a data-driven approach used to approximate complex physical systems using computationally efficient models. In the context of Computational Fluid Dynamics (CFD), high-fidelity simulations are often computationally expensive, especially when multiple design evaluations are required. Surrogate models provide an alternative by learning the relationship between input parameters and output responses from precomputed simulation data [17].

In this work, neural networks are employed as surrogate models to approximate the mapping between input variables and flow characteristics obtained from CFD simulations. The training dataset consists of input-output pairs, where the inputs represent design or operating parameters and the outputs correspond to quantities of interest such as velocity, pressure or temperature fields. Once trained, the surrogate model can predict these outputs for new input conditions with significantly reduced computational cost compared to full CFD simulations [17].

The accuracy of a surrogate model depends on the quality and diversity of the training data, as well as the architecture and training procedure of the neural network. A well trained surrogate model is capable of capturing the underlying trends in the data while generalizing to unseen cases. This makes surrogate modelling particularly useful for applications such as design exploration, optimization and real-time prediction [17]. By combining the physical accuracy of CFD with the computational efficiency of machine learning, surrogate models enable efficient analysis of complex fluid flow problems and form a key component of the methodology adopted in this study.

### 2.3.3 Surrogate Modelling using PhysicsAI

Physics-based machine learning models, such as Altair PhysicsAI, extend traditional surrogate modelling by learning directly from high-fidelity simulation data. These models establish a mapping between input parameters, including geometry and boundary conditions, and the resulting physical response of the system [18].



Unlike conventional surrogate models that typically predict scalar outputs, PhysicsAI models can generate full-field predictions, such as velocity, pressure and temperature distributions, while handling complex geometries. This enables rapid prediction of simulation results with significantly reduced computational cost compared to conventional CFD simulations [18]. In addition to the general surrogate modelling concept, PhysicsAI introduces specific neural network architectures and training configurations tailored for engineering simulation data. These aspects influence the accuracy, computational efficiency and generalization capability of the model. The key components related to architecture selection and hyperparameter tuning are briefly described below based on the guidelines provided in Altair PhysicsAI documentation [18].

### **2.3.3.1 Model Architectures**

PhysicsAI provides multiple neural network architectures designed to handle simulation data with varying levels of complexity. One of the primary approaches is the graph-based architecture, which operates directly on mesh data and captures spatial relationships between nodes. This makes it suitable for CFD applications where the solution depends strongly on geometric and topological features.

More recent developments include transformer-based architectures, which improve prediction capability by capturing long-range dependencies in the data. These models are particularly effective for complex geometries and large-scale problems, although they typically require higher computational resources. In addition, simplified regression-based models are available for cases where only scalar or reduced outputs are required, offering significantly faster prediction times [18].

### **2.3.3.2 Hyperparameter Configuration**

The performance of PhysicsAI models is strongly influenced by the choice of hyperparameters, which control the structure and training behavior of the neural network. Key parameters include the network width, which determines the number of neurons in each layer, and the depth, which defines the number of layers in the network. Increasing these parameters generally improves the model's ability to capture complex patterns, but also increases computational cost and memory requirements.

Training-related parameters such as the number of epochs, batch size and learning rate also play a critical role. The number of epochs determines how many times the model is exposed to the training data, while the batch size controls how many samples are processed simultaneously during training. The learning rate governs the step size during optimization and must be chosen carefully to ensure stable and efficient convergence.

Proper tuning of these hyperparameters is essential for achieving a balance between accuracy and computational efficiency. However, as highlighted in PhysicsAI guidelines, hyperparameter tuning cannot compensate for poor quality training data, making dataset preparation a critical aspect of model development [18].

### 2.3.4 NVIDIA PhysicsNeMo Framework

NVIDIA PhysicsNeMo is an open source deep learning framework developed for scientific computing and physics based machine learning applications. The framework supports the development of surrogate models, neural operators, graph neural networks, and physics informed learning approaches for engineering simulations. PhysicsNeMo provides GPU accelerated workflows for training and inference, making it suitable for large scale Computational Fluid Dynamics (CFD) and multi-physics applications [13, 14]. The framework has recently gained attention in AI assisted engineering because of its flexibility, scalability, and support for modern machine learning architectures for scientific computing problems.

One of the major advantages of PhysicsNeMo is its ability to handle complex engineering datasets generated from high fidelity numerical simulations. The framework supports multiple architectures such as Multi Layer Perceptrons (MLPs), Graph Neural Networks (GNNs), MeshGraphNet, Fourier Neural Operators (FNOs), and physics informed learning approaches. These architectures are particularly relevant for CFD applications because they can learn relationships between flow variables, geometry, and boundary conditions directly from simulation data. In addition, PhysicsNeMo provides tools for handling structured and unstructured mesh based data, which is important for realistic industrial CFD problems involving complex geometries [14].

Another important feature of PhysicsNeMo is its focus on physics informed and operator learning approaches. Traditional surrogate models often rely only on data driven training, whereas physics informed methods incorporate governing equations and physical constraints directly into the learning process. This improves model robustness, physical consistency, and generalization capability for unseen flow conditions. Similarly, neural operator based methods such as Fourier Neural Operators aim to learn mappings between entire function spaces rather than predicting isolated outputs, making them highly suitable for fast approximation of CFD solutions [19].

In the present work, PhysicsNeMo is used as the primary framework for investigating AI based surrogate modelling approaches for CFD applications. Different architectures within the framework are evaluated and compared for predicting fluid flow and thermal fields. The objective is to study their prediction capability, computational efficiency, and applicability to industrially relevant CFD problems while also assessing the practical challenges associated with training and deployment of large scale AI models for engineering simulations.

#### 2.3.4.1 Basic Surrogate Models using Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are among the simplest and most widely used neural network architectures for supervised learning problems. An MLP consists of multiple fully connected layers, where each neuron in a layer is connected to all neurons in the following layer. Information is passed through the network using weighted connections and nonlinear activation functions, allowing the model to learn complex relationships between input and output variables. In PhysicsNeMo,

the fully connected network architecture is implemented as a densely connected neural network with configurable hidden layers and activation functions [20]. The framework also supports features such as adaptive activation functions and weight normalization for improved training stability and performance.

In the context of CFD surrogate modelling, MLPs can be used to learn mappings between simulation input parameters and flow quantities such as pressure, velocity, and temperature. Compared to more advanced architectures, MLP based models are relatively simple to implement and computationally efficient for low dimensional problems. However, fully connected networks do not explicitly account for spatial relationships or mesh connectivity within the computational domain. As a result, their performance may become limited for high dimensional CFD problems involving complex geometries and large flow fields. Nevertheless, MLPs are commonly used as baseline models for evaluating the performance of more advanced architectures such as Graph Neural Networks and Fourier Neural Operators [13].

#### 2.3.4.2 Graph Neural Networks and MeshGraphNet

Graph Neural Networks (GNNs) are deep learning architectures designed to operate on unstructured data, where information is represented using nodes and edges. In CFD applications, the computational mesh can naturally be represented as a graph, where mesh cells or nodes form the graph nodes and their connectivity forms the edges. This allows GNNs to directly learn spatial relationships and local interactions within irregular computational domains. Unlike traditional convolution based neural networks, GNNs can efficiently handle unstructured meshes and complex geometries commonly encountered in engineering simulations [21].

MeshGraphNet is a graph based neural network architecture developed for learning mesh based physical simulations using message transfer operations on computational meshes [21]. The model follows an encoder, processor and decoder structure, where the encoder converts the simulation mesh into graph embeddings, the processor performs multiple rounds of message passing between neighbouring nodes, and the decoder predicts the target physical quantities. During message passing, information is exchanged between connected mesh nodes, allowing the network to capture local flow physics and spatial dependencies. The architecture is capable of learning complex physical systems including fluid dynamics, structural mechanics, and coupled multiphysics problems directly from simulation data. The original MeshGraphNet framework demonstrated strong performance for aerodynamic and fluid flow predictions while achieving significantly faster inference compared to traditional numerical solvers [21].

One of the major advantages of MeshGraphNet is its ability to work directly on unstructured meshes without requiring interpolation onto structured grids. This makes the architecture highly suitable for industrial CFD applications involving complex geometries and varying mesh resolutions. In the present work, MeshGraphNet is investigated as one of the primary surrogate modelling architectures for predicting fluid flow and thermal fields generated from CFD simulations. The model is

implemented using the NVIDIA PhysicsNeMo framework, which provides scalable GPU accelerated workflows for training graph based surrogate models for scientific computing applications [13].

### 2.3.4.3 Physics Informed Neural Networks and Physics Informed GNNs

Physics Informed Neural Networks (PINNs) are deep learning models that incorporate physical laws directly into the training process by embedding governing equations into the loss function of the neural network [22]. Instead of relying only on data driven learning, PINNs use the residuals of partial differential equations to guide the optimization process, allowing the network to learn solutions that satisfy known physical constraints. In fluid dynamics applications, this typically involves enforcing conservation laws such as the continuity and momentum equations during training. By including physical information within the optimization process, PINNs can improve generalization capability, physical consistency, and prediction robustness, especially in cases with limited training data [22,23]. PhysicsNeMo Sym provides built in support for physics informed learning workflows for CFD related problems, including implementations for incompressible Navier Stokes equations using continuity and momentum conservation equations .

Physics informed approaches can also be combined with Graph Neural Networks to form Physics Informed GNNs, where graph based spatial learning is integrated with physics based constraints. In such approaches, the GNN learns spatial relationships and mesh connectivity while the governing equations act as additional constraints during optimization. This combination is particularly useful for CFD applications involving unstructured meshes and complex geometries, since the graph representation preserves local mesh connectivity while the physics informed loss improves solution consistency. In the present work, the physics informed implementation is limited to the continuity and momentum equations for incompressible flow problems. Energy equation based constraints are not included within the current implementation. The objective of this approach is to investigate whether incorporating physical conservation laws into the training process improves the predictive capability and robustness of graph based surrogate models for CFD applications.

### 2.3.5 Fourier Neural Operators

Fourier Neural Operators (FNOs) are neural network architectures developed for learning mappings between input and output fields in partial differential equation (PDE) problems [19]. FNOs learn relationships between entire fields, making them suitable for surrogate modelling of CFD simulations where quantities such as velocity, pressure, and temperature fields must be predicted. FNOs have gained significant attention in scientific machine learning because they can provide fast predictions while maintaining good accuracy for complex flow problems. The main idea behind the Fourier Neural Operator is to perform the learning process in Fourier space instead of directly in the physical domain. The input data is transformed into Fourier space using the Fast Fourier Transform (FFT), where the model learns relationships between important frequency modes. The transformed data is then

converted back to the physical domain using the inverse Fourier transform [19]. By operating in Fourier space, FNOs can efficiently capture large scale spatial relationships and flow structures within the computational domain.

One important requirement of FNO based models is the need for structured grid based data. In many industrial CFD applications, simulations are performed on unstructured meshes because of complex geometries. As a result, the CFD data must first be interpolated onto a structured grid before it can be used for FNO training. This interpolation process may introduce information loss and additional preprocessing requirements, which can become a limitation for complex unstructured CFD cases. Despite this drawback, FNOs remain highly attractive for CFD surrogate modelling because of their computational efficiency and ability to handle large scale field predictions.

In the present work, the Fourier Neural Operator architecture available within NVIDIA PhysicsNeMo is investigated for surrogate modelling of flow and thermal fields generated from CFD simulations. The performance of the FNO model is evaluated and compared with other architectures implemented within the framework.



# 3

## Methods

This chapter presents the methodology adopted in the present work for the development and evaluation of AI based surrogate modelling approaches for CFD applications. The overall workflow consists of CFD data generation, dataset preparation, AI model development, model training, and performance evaluation. Two study cases are considered in the present work, namely a mixing elbow case and a CPU cooling case. CFD simulations are performed using industrial solvers **Ansys Fluent** and **Star-CCM+** to generate datasets for training and validation of the AI models. For each case, the complete CFD setup is described, including geometry preparation, mesh generation, solver settings, boundary conditions, and the design of experiments used for systematic dataset generation. In addition, procedures related to data export, preprocessing, and preparation of datasets for machine learning workflows are presented.

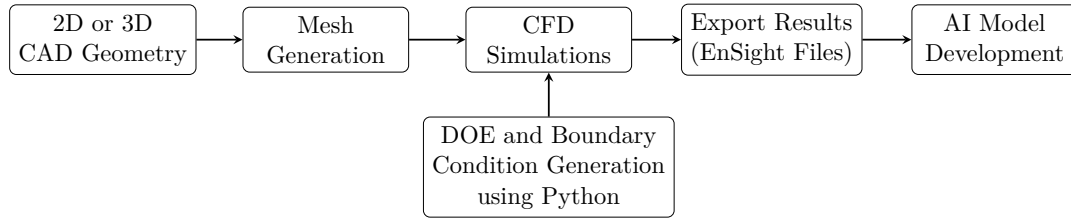
The chapter further describes the AI based surrogate modelling approaches investigated in the present study using both commercial and open source frameworks. The commercial workflow is based on Altair PhysicsAI, where the dataset preparation process, model training workflow, hyperparameter configuration, and prediction methodology are discussed. In parallel, open source AI workflows are implemented using the NVIDIA PhysicsNeMo framework, including graph based and operator learning approaches for CFD surrogate modelling. The different neural network architectures investigated in the present work include Multi Layer Perceptrons, Mesh-GraphNet based Graph Neural Networks, Physics Informed GNNs, and Fourier Neural Operators. Finally, the chapter presents the procedures used for model training, validation, and performance evaluation, together with the computational resources and hardware configuration employed during the study.

### 3.1 Overall Methodology

The overall workflow adopted in the present work for CFD data generation and AI based surrogate model development is illustrated in Figure 3.1. The study begins with the preparation of the CFD geometries corresponding to the selected study cases. CFD simulations are then performed using ANSYS Fluent and STAR-CCM+ in order to generate the datasets required for training the AI models. To generate multiple flow conditions and improve the diversity of the dataset, a Design of Ex-

periments (DOE) approach based on Latin Hypercube Sampling (LHS) is employed using Python. Different inlet velocities, temperatures, and boundary conditions are systematically varied to create multiple CFD simulation cases within the defined parameter ranges.

After completion of the CFD simulations, the generated results are exported in EnSight format for further processing and AI based model development. The exported datasets are then used for surrogate modelling workflows using both commercial and open source AI frameworks. In the commercial workflow, Altair PhysicsAI is used for training and prediction of CFD fields directly from the generated datasets. In parallel, open source workflows are developed using NVIDIA PhysicsNeMo for investigating different neural network architectures, including graph based and operator learning approaches. The final trained models are then used to predict flow and thermal fields for unseen cases, and their performance is evaluated through comparison with the original CFD simulation results.



**Figure 3.1:** Overall workflow adopted in the present study for CFD data generation, dataset preparation, and AI based surrogate model development using commercial and open source frameworks.

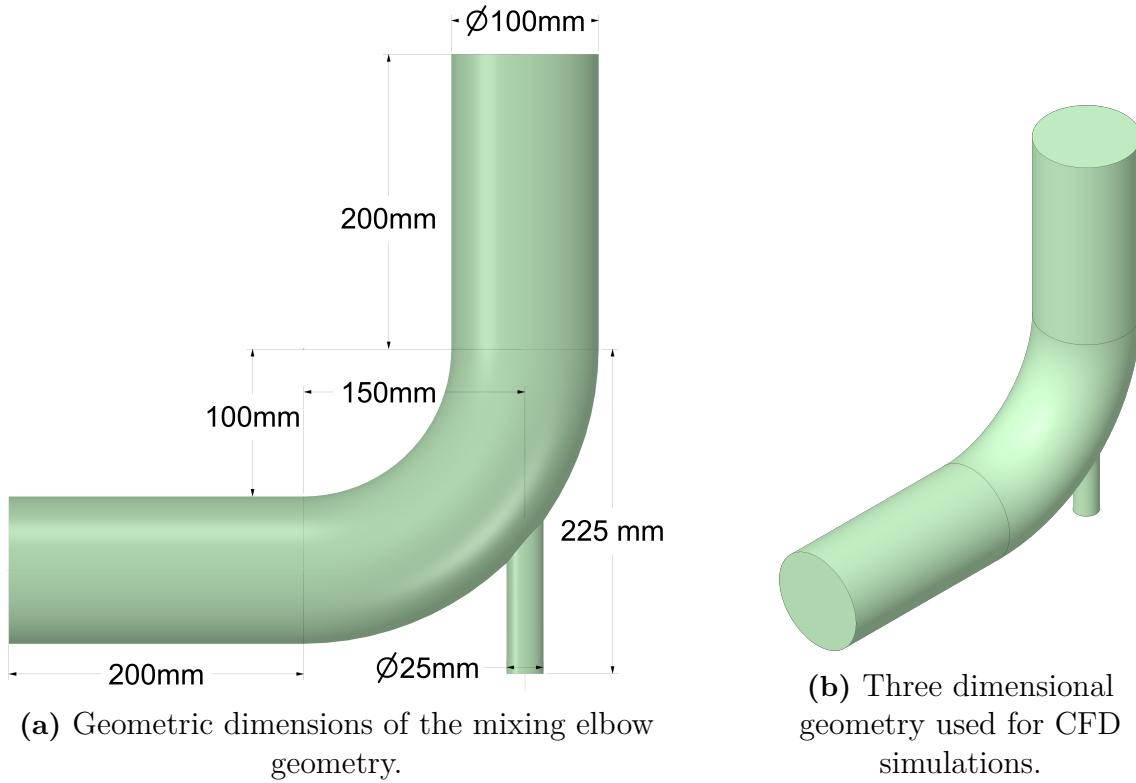
## 3.2 Study 1: Mixing Elbow Case

The first study case considered in the present work focuses on a mixing flow within a 90° elbow geometry, which serves as a simplified benchmark problem for investigating coupled velocity and thermal transport. The case is selected as an initial validation problem because of its relatively low geometric complexity while still exhibiting important flow features such as flow interaction, recirculation, and thermal mixing. This makes the case suitable for evaluating the capability of AI based surrogate models in predicting both flow and thermal fields generated from high fidelity CFD simulations. In the present study, both two dimensional (2D) and three dimensional (3D) versions of the mixing elbow geometry are investigated. The 2D configuration is primarily used for initial workflow development, testing, and validation of the AI models, while the 3D configuration is used to investigate more realistic flow behaviour and increased geometric complexity. By varying the inlet boundary conditions, multiple flow scenarios are generated to create datasets for training and evaluation of the AI based surrogate models.



### 3.2.1 Geometry

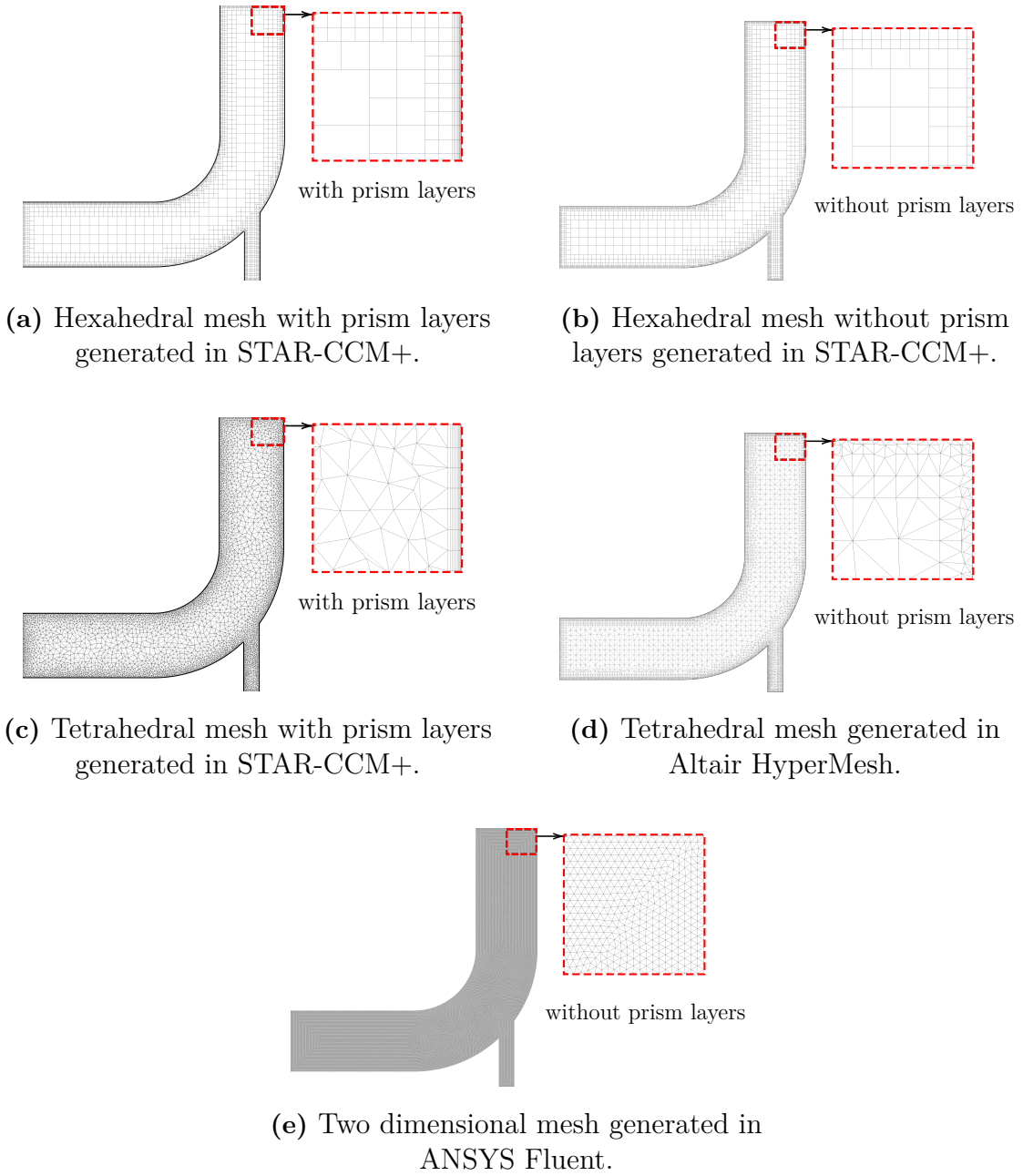
The mixing elbow geometry consists of two inlets and one outlet connected through a curved elbow section, as illustrated in Figure 3.2. The domain includes a main horizontal inlet with a diameter of 100 mm, a secondary vertical inlet with a diameter of 25 mm, and a vertical outlet section. The two inlet streams merge within the elbow region, where the resulting flow behaviour is governed by convective transport and turbulent mixing. Such geometries are commonly used as benchmark configurations for investigating mixing and heat transfer problems in Computational Fluid Dynamics (CFD). In the present study, the geometry remains fixed while the inlet velocities and temperatures are varied systematically in order to generate different flow conditions for dataset creation and surrogate model training.



**Figure 3.2:** Mixing elbow geometry considered in the present study.

### 3.2.2 Mesh Generation

Different mesh configurations were investigated in the present study in order to evaluate the influence of mesh topology, near wall refinement, and software compatibility on both CFD simulations and AI based surrogate modelling workflows. For the three dimensional (3D) mixing elbow case, four different meshes were generated and investigated, while a separate two dimensional (2D) mesh was generated for the 2D study case. The same mesh configurations were used in both ANSYS Fluent and STAR-CCM+ simulations in order to maintain consistency during comparison of the CFD results and subsequent AI model training.



**Figure 3.3:** Different mesh configurations investigated for the mixing elbow case in the present study.

The first mesh configuration shown in Figure 3.3a was generated in STAR-CCM+ using predominantly hexahedral elements together with 10 prism layers near the wall regions for improved boundary layer resolution. The second mesh configuration shown in Figure 3.3b was also generated in STAR-CCM+ using hexahedral elements, but without prism layers, in order to investigate the influence of near wall refinement on the CFD results and AI prediction quality. The third mesh configuration shown in Figure 3.3c was generated using tetrahedral elements together with 10 prism layers in STAR-CCM+, allowing comparison between hexahedral and tetrahedral

dominant meshes for the same geometry and flow conditions.

The fourth mesh configuration shown in Figure 3.3d was generated using Altair HyperMesh with tetrahedral elements. The primary motivation for generating this mesh in HyperMesh was related to compatibility requirements associated with Altair PhysicsAI. Although PhysicsAI is capable of reading hexahedral meshes during the data curation stage, hexahedral elements are currently not supported during the prediction stage. In addition, PhysicsAI does not directly support the native mesh formats generated by ANSYS Fluent or STAR-CCM+. Due to these limitations, a tetrahedral mesh generated using the Altair ecosystem was adopted in order to ensure compatibility with the complete PhysicsAI workflow. The HyperMesh generated mesh was also used for CFD simulations in both STAR-CCM+ and ANSYS Fluent in order to evaluate its performance relative to meshes generated directly within the respective CFD platforms.

For the two dimensional (2D) study case, a structured quadrilateral dominant mesh shown in Figure 3.3e was generated in ANSYS Fluent. The 2D case was primarily used for initial testing, workflow validation, and development of the AI based surrogate modelling approaches before extending the investigation to the more computationally intensive 3D configurations.

### 3.2.3 Solver Settings

**Table 3.1:** Solver settings used for the mixing elbow case.

| Setting          | Configuration          |
|------------------|------------------------|
| Flow Type        | Steady State           |
| Solver Approach  | Segregated Flow Solver |
| Turbulence Model | SST $k-\omega$         |
| Density Model    | Constant Density       |
| Energy Equation  | Enabled                |
| Working Fluid    | Water                  |

The CFD simulations for the mixing elbow case were performed using a steady state formulation in both ANSYS Fluent and STAR-CCM+. The simulations were carried out using a segregated flow solver together with the SST  $k-\omega$  turbulence model in order to accurately capture the near wall flow behaviour and mixing characteristics within the elbow region. In addition, the energy equation was enabled to account for thermal transport and temperature mixing between the two inlet streams. A constant density assumption was employed throughout the simulations, which is consistent with the incompressible flow assumptions discussed previously. The same solver configuration was maintained in both CFD platforms to ensure consistency

in the generated datasets and allow direct comparison of the simulation results obtained from the different software environments.

### 3.2.4 Boundary Conditions

The mixing elbow domain consists of two velocity inlets and one pressure outlet. The large inlet and small inlet were both defined as velocity inlet boundaries where the inlet velocity and temperature were prescribed. The outlet boundary was defined as a pressure outlet with atmospheric pressure conditions. The inlet velocities and temperatures were treated as variable input parameters for the surrogate modelling workflow, since variations in these quantities directly influence the resulting velocity and thermal fields within the mixing region.

**Table 3.2:** Boundary condition ranges used for the mixing elbow case.

| Parameter          | Minimum Value | Maximum Value |
|--------------------|---------------|---------------|
| $U_{\text{large}}$ | 0.2 m/s       | 0.6 m/s       |
| $U_{\text{small}}$ | 0.6 m/s       | 2.0 m/s       |
| $T_{\text{large}}$ | 293.15 K      | 333.15 K      |
| $T_{\text{small}}$ | 303.15 K      | 353.15 K      |

#### 3.2.4.1 Design of Experiments and Dataset Generation

In order to systematically generate multiple flow conditions for surrogate model training, a Design of Experiments (DOE) approach based on Latin Hypercube Sampling (LHS) was employed. LHS was selected because it allows efficient sampling of the input parameter space while ensuring good coverage of the prescribed parameter ranges. Using this approach, different combinations of inlet velocities and inlet temperatures were generated automatically using Python scripts. A total of 50 design points were generated within the specified parameter bounds, where each design point corresponds to an individual CFD simulation case. This structured sampling strategy ensured that the generated dataset captured a sufficiently diverse range of flow and thermal conditions for training and evaluating the AI based surrogate models.

The generated DOE samples were exported as input files and subsequently used for automated CFD simulations in both ANSYS Fluent and STAR-CCM+. The Python script used for generating the DOE samples using Latin Hypercube Sampling is provided in Appendix: DOE Code.

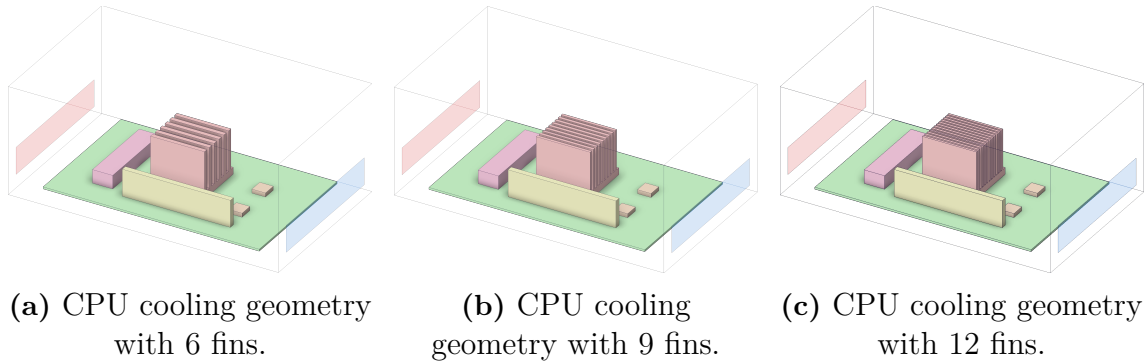
## 3.3 Study 2: CPU Cooling Case

The second study considers a CPU cooling problem involving forced convection cooling inside an electronic enclosure. Compared to the mixing elbow case discussed

previously, this case represents a more complex conjugate heat transfer problem with multiple solid components and more complicated flow interactions. The setup consists of a CPU, RAM modules, VRM section, miscellaneous electronic components, and a heat sink mounted on top of the CPU. Air enters the computational domain through a velocity inlet and exits through a pressure outlet, thereby creating a forced convection cooling environment inside the enclosure.

Unlike Study 1, which involved multiple surrogate modelling frameworks and software platforms, this study is performed specifically using ANSYS Fluent for CFD simulations and NVIDIA PhysicsNeMo for surrogate model development. The objective of this case is to investigate the capability of PhysicsNeMo for predicting the coupled flow and thermal behaviour inside electronic cooling systems using CFD generated datasets. In addition, the influence of heat sink geometry on the cooling performance is also analysed by varying the number of fins used in the heat sink configuration.

### 3.3.1 Geometry

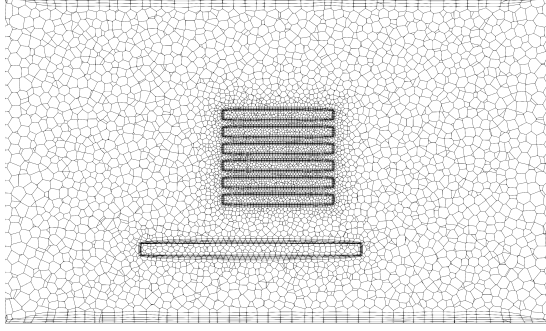


**Figure 3.4:** CPU cooling configurations considered in the present study with varying heat sink fin counts. The red surface indicates the inlet boundary and the blue surface indicates the outlet boundary.

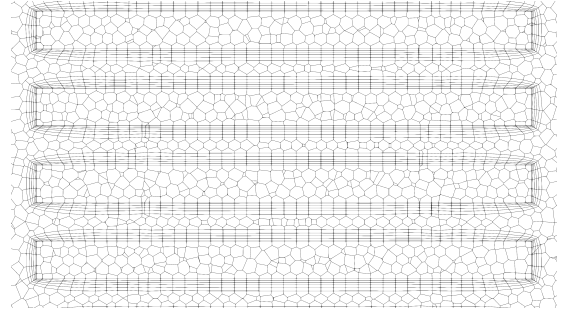
The computational setup used for the CPU cooling case is shown in Figure 3.4. The domain consists of simplified representations of the major electronic components typically present on a pcb, including the CPU, RAM modules, VRM section, and additional small electronic components. A heat sink is mounted on top of the CPU in order to enhance heat dissipation through forced convection cooling. In all configurations, the red shaded surface represents the inlet boundary, while the blue shaded surface represents the outlet boundary.

In this study, three different heat sink geometries are investigated by varying the number of cooling fins. The first geometry consists of 6 fins, the second geometry consists of 9 fins, and the third geometry consists of 12 fins. This allows investigation of how changes in fin density influence the resulting flow distribution and thermal cooling performance within the computational domain.

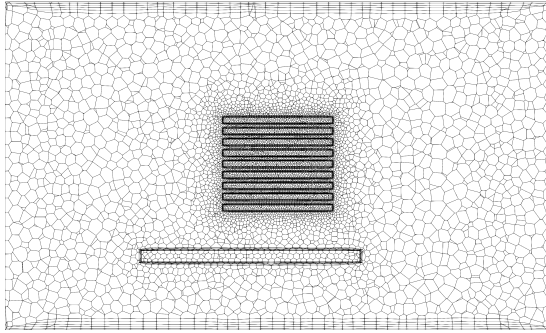
### 3.3.2 Mesh Generation



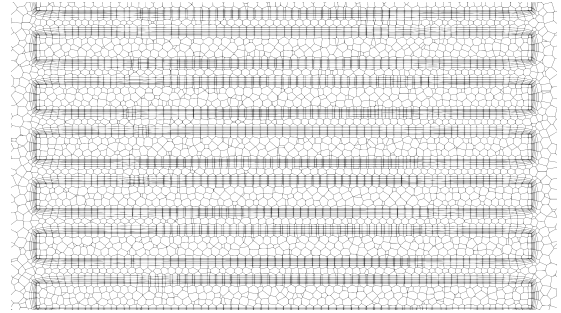
(a) Mesh for 6 fin configuration.



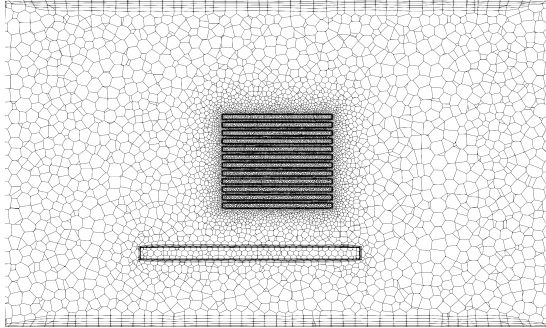
(b) Zoomed view of 6 fin mesh.



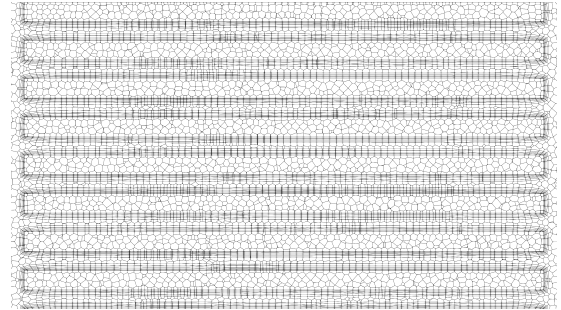
(c) Mesh for 9 fin configuration.



(d) Zoomed view of 9 fin mesh.



(e) Mesh for 12 fin configuration.



(f) Zoomed view of 12 fin mesh.

**Figure 3.5:** Polyhedral meshes used for the CPU cooling case along with zoomed views showing the prism layers and local mesh refinement near the heat sink fins.

For the CPU cooling case, polyhedral meshes were generated for all three heat sink configurations using Ansys Fluent. In contrast to the previous study where multiple mesh types were investigated, the present case uses only polyhedral elements since NVIDIA PhysicsNeMo supports polyhedral mesh elements directly for training and prediction purposes. The generated meshes also include 5 prism layers near the wall regions in order to better capture the near wall velocity and thermal gradients. Figure 3.5 shows the generated meshes for all three configurations along with the corresponding zoomed views near the fin regions, where the prism layers and local mesh refinement around the solid surfaces can be clearly observed. Similar mesh

generation strategy was applied for all three geometries containing 6, 9, and 12 heat sink fins, respectively. A minimum mesh size of 0.25 mm and a maximum mesh size of 5 mm were used throughout the computational domain. Local mesh refinement was introduced around the heat sink fins and electronic components using the proximity based refinement option in Fluent, where the Cells Per Gap parameter was set to 2 and the proximity scope was defined as faces-and-edges. This helped in achieving good mesh resolution in for the solid components.

### 3.3.3 Solver Settings

The CFD simulations for the CPU cooling case were performed in ANSYS Fluent using a steady state pressure based coupled solver. Since the present study involves forced convection heat transfer inside an electronic enclosure, the energy equation was enabled to capture the coupled fluid flow and thermal transport behaviour. The SST  $k-\omega$  turbulence model was used for turbulence modelling due to its good near wall prediction capability and robustness for internal flow applications involving separation and recirculation regions.

Air was used as the working fluid inside the computational domain. Different solid materials were assigned to the electronic components and enclosure parts in order to represent realistic thermal conduction behaviour. Aluminium was used for the heat sink, PCB material was assigned to the motherboard, and the remaining electronic components were modelled using steel material properties.

**Table 3.3:** Solver settings used for the CPU cooling case.

| Setting          | Configuration       |
|------------------|---------------------|
| Flow Type        | Steady State        |
| Solver Approach  | Coupled Flow Solver |
| Turbulence Model | SST $k-\omega$      |
| Density Model    | Constant Density    |
| Energy Equation  | Enabled             |
| Working Fluid    | Air                 |

**Table 3.4:** Material assignment used in the CPU cooling simulations.

| Component             | Material     |
|-----------------------|--------------|
| Heat Sink             | Aluminium    |
| PCB                   | PCB Material |
| Electronic Components | Steel        |
| Fluid Domain          | Air          |

### 3.3.4 Boundary Conditions and Dataset Generation

The computational domain consists of a velocity inlet and a pressure outlet representing a forced convection cooling configuration. Heat sources were applied to the CPU, RAM modules, and VRM section in order to model the thermal loading inside the electronic enclosure. The inlet airflow was specified in terms of volumetric flow rate (CFM), while the outlet was defined as a pressure outlet condition.

The input parameters varied in this study include the CPU heat source, RAM heat source, VRM heat source, and inlet airflow rate. Different combinations of these parameters were used to generate multiple operating conditions for dataset creation and surrogate model training. The dataset was generated by systematically varying the boundary conditions listed in Table 3.5. In total, 48 simulation cases were generated and solved for each heat sink configuration. Since three different heat sink geometries were considered in the present study, the final dataset also captures the influence of geometric variation on the thermal and flow behaviour inside the enclosure. The generated CFD datasets were later used for data curation, preprocessing, and surrogate model training within the NVIDIA PhysicsNeMo framework.

**Table 3.5:** Boundary condition ranges used for the CPU cooling case.

| Parameter       | Values           |
|-----------------|------------------|
| CPU Heat Source | 15, 20, 22, 25 W |
| RAM Heat Source | 1.5, 2 W         |
| VRM Heat Source | 5, 8 W           |
| Inlet Flow Rate | 5, 10, 15 CFM    |

## 3.4 Altair PhysicsAI Methodology

The CFD simulation results generated from ANSYS Fluent and Star-CCM+ were exported as EnSight case files and imported into Altair PhysicsAI for dataset creation and surrogate model training. During the dataset preparation stage, PhysicsAI converts the imported simulation results into its own internal data format suitable for geometric deep learning based training. The imported datasets included both the mesh information and the corresponding flow and thermal field results obtained from the CFD simulations.

The input parameters corresponding to the operating conditions were provided using a global JSON hook file which acts as a global KPI parameter. These global inputs included parameters such as inlet flow rate and temperature, which remain constant throughout a given simulation case but vary between different cases. The use of global hooks allows PhysicsAI to associate the varying boundary conditions with the corresponding CFD results during model training. In the present workflow, the inlet boundary conditions such as inlet velocity and temperature were provided to



the surrogate models as global input parameters through the hook file rather than being directly prescribed at the corresponding inlet nodal locations. As a result, the AI models were required to simultaneously learn both the boundary condition influence and the resulting flow field behaviour from the training data. While this approach was sufficient for interpolation cases within the training distribution, it can introduce larger deviations during extrapolation cases where the inlet conditions differ significantly from the training dataset.

The present study uses the Graph Context Neural Simulator (GCNS) architecture available within PhysicsAI for training the surrogate models. During training, different combinations of network width, depth, and number of epochs were investigated in order to study their influence on the prediction accuracy and training stability. The width and depth hyperparameters control the network size and represent the number of neurons and hidden layers used in the model respectively, while the number of epochs controls the total training iterations performed during optimization.

PhysicsAI automatically partitions the curated dataset into training and validation datasets. By default, approximately 85% of the available data is used for model training and the remaining 15% is used for validation. The validation dataset is used to monitor the model performance during training and helps in identifying overfitting or underfitting behaviour through the training and validation loss curves.

The trained PhysicsAI models were later used for prediction and validation on unseen cases in order to evaluate the capability of geometric deep learning for fast CFD field prediction in electronic cooling applications.

### 3.5 NVIDIA PhysicsNeMo Methodology

The workflow adopted in the present work using NVIDIA PhysicsNeMo mainly consists of three stages, namely data curation, model training, and prediction. This workflow allows the CFD simulation data generated from ANSYS Fluent and STAR-CCM+ to be converted into machine learning compatible datasets, followed by neural network training and prediction of CFD fields for unseen operating conditions. An example implementation of the corresponding workflow scripts is provided in the Appendix using `main_data.py`, `main_train.py`, and `main_pred.py`.

The first stage involves dataset preparation using the `main_data.py` script. During this stage, the CFD simulation results exported in EnSight format are imported into PhysicsNeMo and converted into `.npz` files suitable for neural network training. The data curation stage also allows the user to select the required CFD variables, such as pressure, velocity, and temperature fields, which are later used as output targets during training. In addition, normalization of the input and output variables is performed during this stage through the creation of `x_scaler.pkl` and `y_scaler.pkl` files. These scaler files are later reused during both training and prediction in order to maintain consistency in the variable scaling process.

The second stage involves neural network training using the `main_train.py` script.

During training, the curated datasets are used for supervised learning of the selected neural network architecture. The complete dataset is divided into 80% training data and 20% validation data in order to evaluate the generalization capability of the model during training. The framework computes both training and validation losses throughout the optimization process to monitor convergence behaviour and model performance. Different architectures investigated in the present work, including MLP based models, MeshGraphNet, Physics Informed GNNs, and FNO were trained using this workflow. After completion of training, the best performing model based on the minimum validation loss is automatically saved in `.pth` format for later prediction and evaluation.

The final stage involves prediction using the `main_pred.py` script. In this stage, the trained surrogate model takes new user defined operating conditions and boundary condition values as input parameters and predicts the corresponding CFD solution fields. The predicted quantities include variables such as pressure, velocity, and temperature fields depending on the study case. Finally, the predicted results are exported and saved in `.vtk` format, which can later be used for CFD post processing and visualization.

## 3.6 AI Architectures Investigated

This section discusses the different AI architectures investigated in the present work for surrogate modelling of CFD simulations. The primary objective of these models is to learn the relationship between the input parameters and the resulting flow and thermal fields generated from CFD simulations. Similar to the approach used in Altair PhysicsAI, the operating conditions and boundary conditions are provided to the neural networks as global input parameters along with the spatial coordinates of the computational domain. In the present work, the inlet velocity and temperature conditions were incorporated as global input parameters rather than being directly prescribed at the corresponding inlet nodal locations. Consequently, the surrogate models were required to simultaneously learn both the influence of the inlet conditions and the resulting flow field behaviour from the training dataset. While this approach performed reasonably well for interpolation cases, it can introduce larger localized deviations during extrapolation conditions outside the trained parameter distribution. The trained models are then used to predict quantities such as pressure, velocity, and temperature fields for unseen operating conditions.

### 3.6.1 Basic Surrogate Model (MLP)

The first surrogate modelling approach investigated in the present study is a Multi Layer Perceptron (MLP) based neural network. The model was implemented using the FullyConnected architecture available within the NVIDIA PhysicsNeMo framework. The MLP model takes the spatial coordinates and boundary condition parameters as input features and predicts the corresponding CFD solution variables at each spatial location. In the present work, the MLP based surrogate modelling approach was investigated for both the two dimensional (2D) and three dimensional

(3D) mixing elbow cases.

For the mixing elbow case, the input features consist of the spatial coordinates together with the inlet boundary conditions provided as global input parameters. For example, in the 2D mixing elbow case, the node input vector consists of  $(x, y, U_{\text{large}}, U_{\text{small}}, T_{\text{large}}, T_{\text{small}})$ , while the output variables correspond to pressure, velocity components, and temperature fields. The corresponding input and output feature dimensions were modified accordingly for the 3D mixing elbow case based on the additional spatial coordinate and flow variables. Before training, the input and output datasets were normalised using the StandardScaler method in order to improve training stability and convergence behaviour. The processed datasets from all simulation cases were combined and used for supervised learning.

The MLP architecture used in the present work consists of multiple fully connected hidden layers with tanh activation functions. For the 2D mixing elbow case, the network configuration used during training includes 6 input features, 4 output variables, a hidden layer width of 512 neurons, and 6 hidden layers. The corresponding architecture dimensions were modified accordingly for the 3D mixing elbow case depending on the number of spatial coordinates and output solution variables. The model was trained using the Adam optimizer together with Mean Squared Error (MSE) loss. After training, the surrogate model was used for prediction on unseen operating conditions by providing new inlet boundary conditions together with the spatial coordinates of the domain. The predicted results were then converted back into CFD compatible formats for post processing and visualization.

### 3.6.2 MeshGraphNet

The second surrogate modelling approach investigated in the present study is the MeshGraphNet architecture available within the NVIDIA PhysicsNeMo framework. Unlike the fully connected MLP model discussed previously, MeshGraphNet directly operates on graph based mesh representations and therefore preserves the spatial connectivity information of the computational mesh. This makes the architecture particularly suitable for CFD applications involving unstructured meshes and complex geometries. In the present work, the CFD mesh is converted into a graph representation where the mesh nodes act as graph nodes and the mesh connectivity information forms the graph edges. During preprocessing, the EnSight case files exported from the CFD simulations are read using PyVista, and the mesh connectivity is extracted automatically from the computational mesh. Edge connections are generated between neighbouring mesh points, and edge features are computed using the Euclidean distance between connected nodes.

Similar to the MLP implementation, the node input features consist of the spatial coordinates together with the inlet boundary conditions provided as global input parameters. For example, in the 2D mixing elbow case, the node input vector includes  $(x, y, U_{\text{large}}, U_{\text{small}}, T_{\text{large}}, T_{\text{small}})$ , while the output variables correspond to pressure, velocity components, and temperature fields. The corresponding input and output feature dimensions were modified accordingly for the 3D mixing elbow case and the

CPU cooling case based on the geometry, operating conditions, and thermal loading configuration. The datasets were normalised using the StandardScaler method before training in order to improve convergence behaviour and training stability.

The MeshGraphNet model implemented in the present work consists of graph message passing layers together with edge feature processing for learning the local flow interactions within the computational domain. For the 2D mixing elbow case, the implemented architecture uses 6 node input features, 1 edge feature representing edge distance, and 4 output variables corresponding to the CFD solution fields. A processor size of 5 was used for the graph processor network. The corresponding architecture dimensions were modified accordingly for the 3D mixing elbow case and the CPU cooling case depending on the number of spatial coordinates, boundary condition parameters, and output solution variables. During training, the processed CFD cases were divided into training and validation datasets using an 80% - 20% split. The model was trained using the Adam optimizer together with Mean Squared Error (MSE) loss. The training and validation losses were monitored throughout the training process, and the best performing model based on the validation loss was saved for later prediction and evaluation. After training, the MeshGraphNet model was used to predict CFD fields for unseen inlet conditions by reconstructing the graph representation of the computational mesh and providing the new boundary condition inputs together with the node coordinates. The predicted flow and thermal fields were subsequently exported into CFD compatible visualization formats for post processing and comparison with the original CFD simulation results.

#### 3.6.3 Physics Informed GNN

In addition to the standard MeshGraphNet implementation, a Physics Informed Graph Neural Network (Physics Informed GNN) approach was also investigated in the present study. The primary objective of this approach is to improve the physical consistency of the predicted CFD fields by incorporating governing equation residuals directly into the training process. Similar to the standard MeshGraphNet workflow, the CFD meshes were converted into graph representations where mesh nodes form the graph nodes and mesh connectivity information forms the graph edges. The same preprocessing workflow, graph generation strategy, normalization procedure, and input feature construction used for the standard MeshGraphNet model were also adopted for the Physics Informed GNN implementation.

The implemented Physics Informed GNN architecture is based on the MeshGraphNet model available within the NVIDIA PhysicsNeMo framework. For example, in the 2D mixing elbow case, the node input features consist of the spatial coordinates together with the inlet boundary conditions, while the output variables correspond to pressure, velocity components, and temperature fields. The model architecture for this case uses 6 node input features, 1 edge feature representing edge distance, and 4 output variables corresponding to the CFD solution fields. A processor size of 4 was used for the graph processor network. Similar workflows were also adopted for the 3D mixing elbow case and the CPU cooling case, where the input parameters, output variables, and network dimensions were modified according to the

corresponding geometry, operating conditions, and thermal loading configuration.

Unlike the standard data driven MeshGraphNet approach, the Physics Informed GNN additionally incorporates the governing incompressible Navier–Stokes equations during training using the PhysicsInformer module available in NVIDIA PhysicsNeMo. In the present work, the continuity equation together with the momentum equations were incorporated into the loss formulation as physics based constraints. During each training iteration, the predicted flow fields are converted back into physical space and used to compute the residuals corresponding to the continuity and momentum conservation equations. These residuals are then included as additional physics based loss terms together with the standard data loss. The total loss function therefore consists of a combination of the supervised data loss and the governing equation residual loss, allowing the network to learn solutions that not only fit the CFD data but also satisfy the underlying physical conservation laws.

Due to GPU memory limitations encountered during the present study, the physics informed training for the 3D mixing elbow case was performed using a reduced number of nodal points sampled from the original CFD mesh instead of the complete computational domain for physics loss. This reduction was necessary to maintain feasible memory usage and training time during optimization of the Physics Informed GNN model. As a result, the implemented Physics Informed GNN framework for the 3D case in the present work should be interpreted as a proof of concept level implementation rather than a fully optimized large scale physics informed CFD surrogate model.

The model was trained using the Adam optimizer together with Mean Squared Error (MSE) loss for the data driven component of the training process. The training process was performed for 1000 epochs, and the best performing model based on the training loss was saved for later prediction and evaluation. After training, the Physics Informed GNN model was used for prediction on unseen operating conditions using the same graph reconstruction and preprocessing workflow adopted for the standard MeshGraphNet model.

### 3.6.4 Fourier Neural Operator

The Fourier Neural Operator (FNO) architecture available within the NVIDIA PhysicsNeMo framework was also investigated in the present study for the 2D mixing elbow case. Unlike the MLP and graph based approaches, the FNO architecture operates on structured grid data. Therefore, during the data preparation stage, the CFD results exported in EnSight format were interpolated from the original unstructured mesh onto a uniform structured grid before training. The input channels used for the FNO model include the spatial coordinates, geometry mask, and operating condition parameters, while the output channels correspond to the CFD field variables such as pressure, velocity, and temperature fields.

A binary mask field was used to distinguish the valid fluid region from the void or non physical regions within the structured grid. During interpolation of the CFD results

onto the structured grid, the `vtkValidPointMask` generated during the sampling process was used to identify whether a structured grid point lies inside the actual CFD domain or outside it. Grid locations with mask values greater than 0.5 were treated as valid fluid regions, while locations with values less than 0.5 were treated as void regions. This was particularly important near wall boundaries where parts of the structured grid extend outside the original CFD geometry. The same masking approach was also applied during both training and prediction in order to prevent the network from learning non physical values in regions outside the fluid domain.

The processed datasets were normalized using global normalization scalers and divided into 80% training data and 20% validation data. During training, a masked Mean Squared Error (MSE) loss function was used such that the loss was computed only within the valid fluid regions of the computational domain. The model was trained using the Adam optimizer, and the best performing model based on the minimum validation loss was automatically saved for later prediction and evaluation. The predicted fields were subsequently exported into CFD compatible visualization formats for post processing and visualization.

### 3.7 Evaluation Metrics

The performance of the AI based surrogate models for the mixing elbow case is evaluated using both interpolation and extrapolation test cases. First, the trained models are tested on unseen cases that lie within the parameter ranges defined previously in Table 3.2. These cases are used to evaluate the interpolation capability of the models, since the operating conditions remain inside the training domain but are not included in the training dataset itself. In addition, five extrapolation test cases are also considered in order to evaluate the generalization capability of the models outside the training domain. In these cases, one or more boundary condition parameters are intentionally selected outside the ranges used during dataset generation.

**Table 3.6:** Interpolation and extrapolation test cases used for evaluation of the surrogate models for the mixing elbow case.

| Case Type                               | $U_{\text{large}}$ [m/s] | $U_{\text{small}}$ [m/s] | $T_{\text{large}}$ [K] | $T_{\text{small}}$ [K] |
|-----------------------------------------|--------------------------|--------------------------|------------------------|------------------------|
| Case 1: Within range                    | 0.582                    | 1.027                    | 327.98                 | 339.01                 |
| Case 2: $U_{\text{large}}$ out of range | 0.1                      | 1.0                      | 310                    | 333                    |
| Case 3: $U_{\text{small}}$ out of range | 0.3                      | 2.5                      | 313                    | 333                    |
| Case 4: $T_{\text{large}}$ out of range | 0.3                      | 1.2                      | 283                    | 343                    |
| Case 5: $T_{\text{small}}$ out of range | 0.3                      | 1.2                      | 313                    | 363                    |
| Case 6: All out of range                | 0.1                      | 2.5                      | 283                    | 363                    |

Table 3.6 shows the different evaluation cases considered in the present study. The prediction performance of the surrogate models is evaluated by comparing the predicted velocity, pressure, and temperature fields against the corresponding CFD simulation results for these test cases. Case 2 corresponds to a lower value of  $U_{\text{large}}$  selected outside the boundary condition range listed in Table 3.2. Similarly, Case 3 corresponds to a higher value of  $U_{\text{small}}$ , Case 4 corresponds to a lower value of  $T_{\text{large}}$ , and Case 5 corresponds to a higher value of  $T_{\text{small}}$  outside their respective training ranges. Finally, Case 6 represents a fully extrapolated case where all four boundary condition parameters are selected outside the ranges used during dataset generation.

In the present work, the model predictions are mainly evaluated using difference contours instead of percentage error contours. This is because the pressure and velocity fields contain regions where the reference CFD solution becomes very small or close to zero. Under such conditions, percentage based error calculations can produce extremely large or misleading values due to the small denominator, even when the actual prediction difference remains physically small. Therefore, difference contours provide a more stable and physically meaningful comparison of the prediction accuracy across the computational domain.





# 4

## Results and Discussion

### 4.1 Mixing Elbow Case

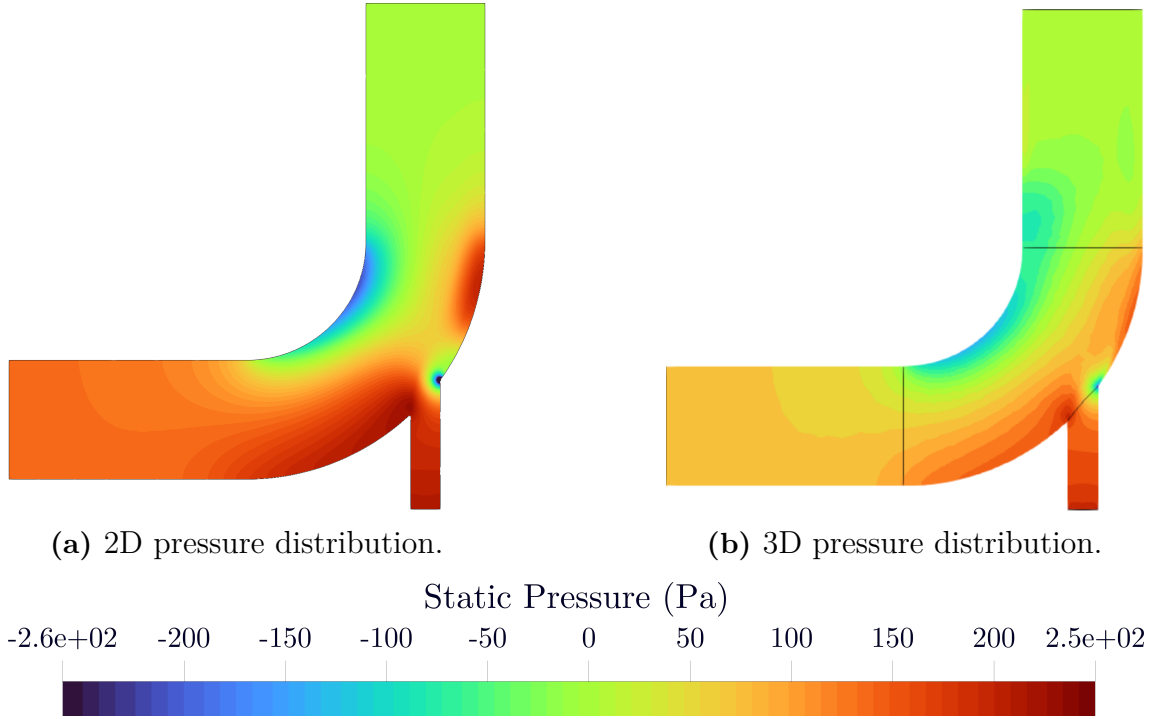
The mixing elbow case was selected as the initial benchmark problem for evaluating AI based surrogate models for coupled fluid flow and heat transfer predictions. Although geometrically simple, the mixing elbow contains important physical phenomena including pressure gradients, flow acceleration, thermal mixing, and wall induced velocity variations. These characteristics make the configuration suitable for validating both the CFD workflow and the capability of the investigated AI architectures before extending the study toward more computationally intensive industrial applications.

The study was divided into two configurations consisting of a two dimensional (2D) case and a three dimensional (3D) case. The 2D case was primarily used for rapid workflow development, dataset generation, and initial evaluation of the surrogate models, while the 3D case was later introduced to investigate the influence of additional geometric complexity and three dimensional flow structures on the prediction capability of the AI models.

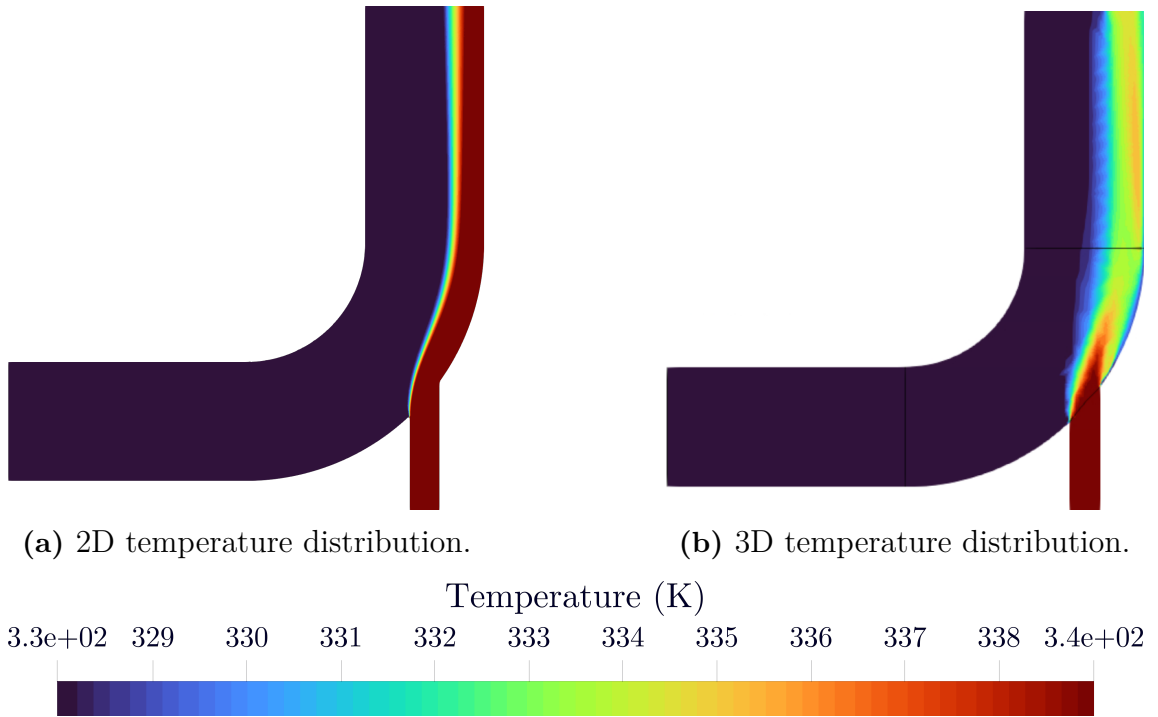
#### 4.1.1 Overview of the Mixing Elbow Study

Figures 4.1 - 4.3 present the CFD results obtained for both the 2D and 3D mixing elbow configurations. Figure 4.1 compares the static pressure distributions obtained from the 2D and 3D mixing elbow CFD simulations. In both cases, a strong pressure gradient is observed near the junction region where the secondary inlet interacts with the primary flow. The lowest pressure region appears near the smaller inlet due to local flow acceleration. While both configurations show similar overall pressure behaviour, the 3D case exhibits more spatial variation within the elbow region compared to the smoother pressure distribution observed in the 2D case due to the additional flow redistribution possible in the third dimension. Similarly, the temperature distributions shown in Figure 4.2 demonstrate the thermal mixing process between the colder primary stream and the hotter secondary inlet flow. In both configurations, a thermal mixing layer develops downstream of the inlet interaction region. Nevertheless, the 3D case shows a more spread out temperature distribution near the bend and outlet regions. The hot and cold fluids mix more gradually in the 3D pipe flow, resulting in a wider mixing region compared to the sharper

temperature boundary observed in the 2D simulation.

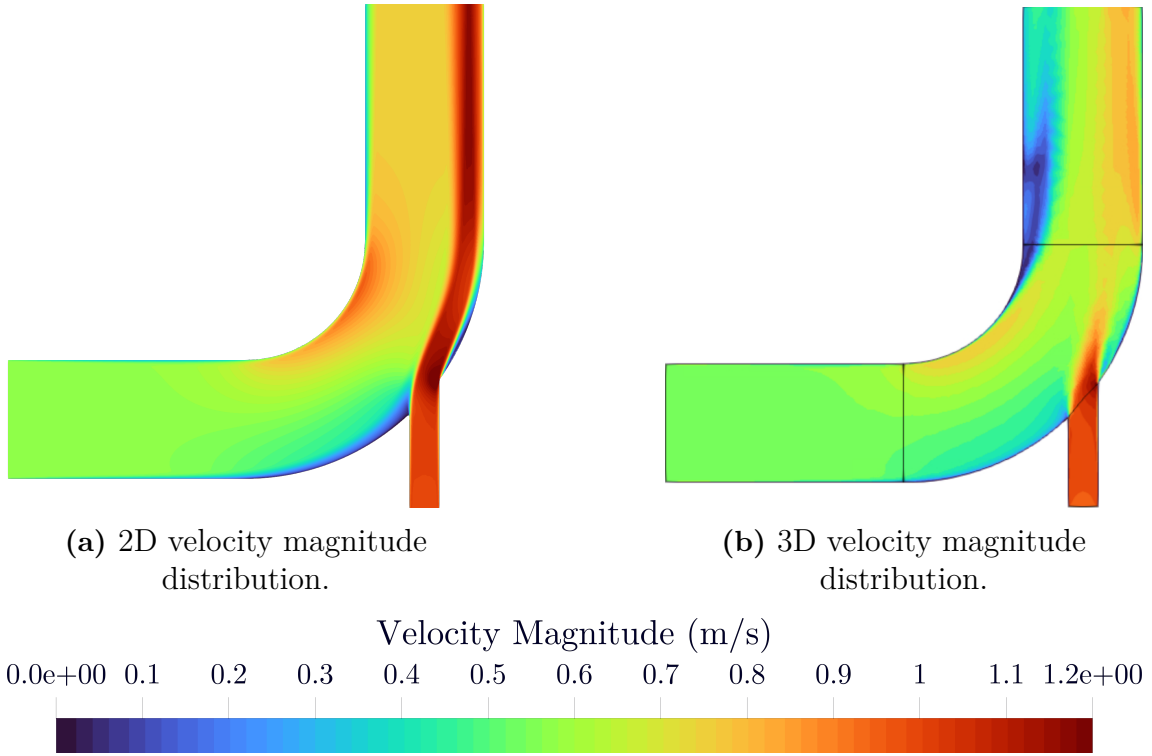


**Figure 4.1:** Comparison of static pressure distributions obtained from the 2D and 3D mixing elbow CFD simulations.



**Figure 4.2:** Comparison of temperature distributions obtained from the 2D and 3D mixing elbow CFD simulations.

The velocity magnitude contours presented in Figure 4.3 show local flow acceleration near the secondary inlet and along the curved elbow section. Higher velocity regions are observed near the inner wall due to the turning of the flow through the curved geometry, while lower velocity regions appear close to the walls because of viscous boundary layer effects. Compared to the 2D case, the 3D configuration shows more uneven velocity distribution, especially near the bend and outlet region. This is because the flow can also move in the depth direction in the 3D domain, leading to more complex flow patterns and localized velocity changes.



**Figure 4.3:** Comparison of velocity magnitude distributions obtained from the 2D and 3D mixing elbow CFD simulations.

#### 4.1.2 Two Dimensional Mixing Elbow Results

In this section, the results obtained for the two dimensional mixing elbow case using the AI based surrogate models are discussed. The behaviour and prediction capability of both Altair PhysicsAI and NVIDIA PhysicsNeMo are investigated for unseen test cases as well as extrapolation cases. Particular focus is given to how the AI models capture the velocity, pressure and temperature fields compared to the CFD reference solutions and how their prediction behaviour changes outside the training range.

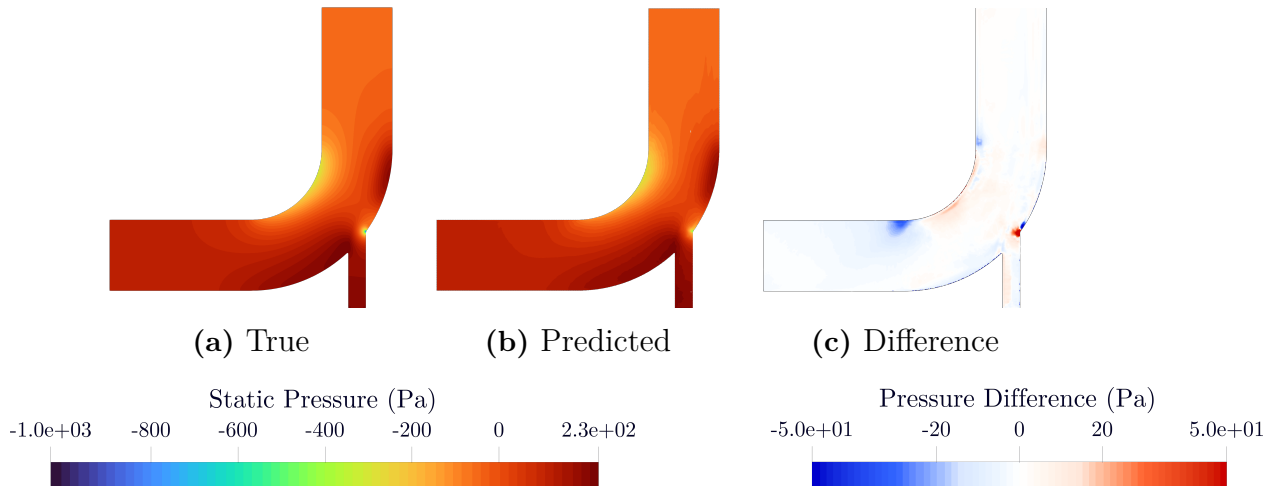
##### 4.1.2.1 Altair PhysicsAI Results

In this section, the results obtained using Altair PhysicsAI for the two dimensional mixing elbow case are discussed. The prediction capability of the model is first

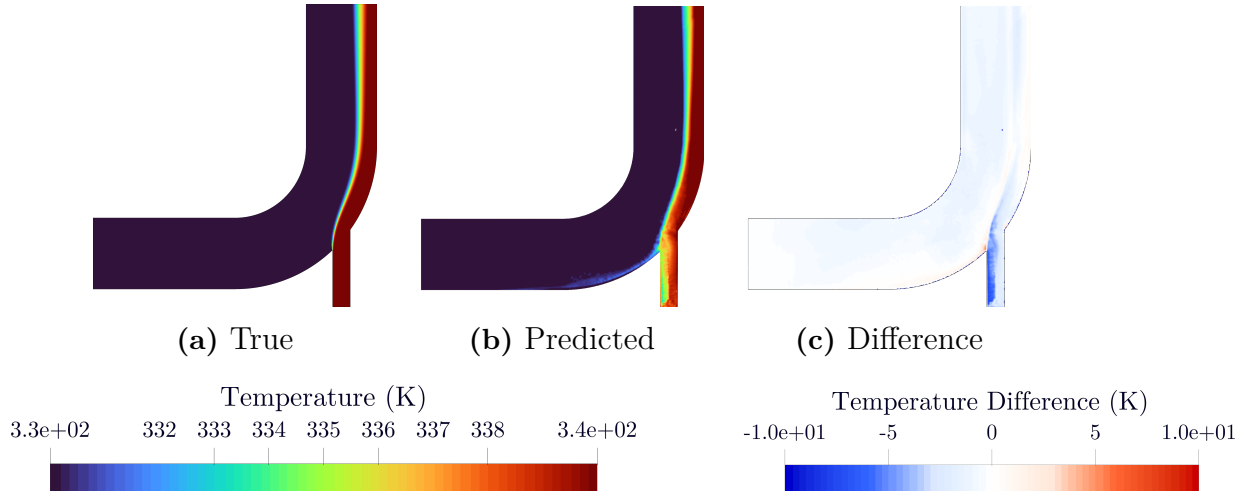
evaluated for an unseen within range test case, followed by its behaviour for extrapolated cases outside the training range. For the extrapolated cases, only the difference contours are presented and discussed, while the corresponding true CFD reference contours are provided in Appendix A.1.

**4.1.2.1.1 Within Range Unseen Case:** The first evaluation case corresponds to the within range unseen condition listed in Table 3.6, where the input parameters remain inside the training range but the exact combination was not seen during training. This case is important since it evaluates the interpolation capability of the Altair PhysicsAI surrogate model under realistic operating conditions. Figures 4.4, 4.5, and 4.6 compare the CFD reference solution with the corresponding Altair PhysicsAI predictions for the static pressure, temperature, and velocity magnitude fields, respectively. In all three cases, the difference plots represent the deviation between the CFD reference and the surrogate prediction.

From Figure 4.4, it can be observed that the predicted pressure field follows the overall spatial distribution of the CFD solution with good agreement across most regions of the mixing elbow. The high pressure regions near the inlet sections and the gradual pressure variation through the curved section are captured reasonably well by the surrogate model. The difference plot shows that the majority of the domain exhibits relatively small deviations, indicating that the bulk pressure behavior is predicted accurately. However, localized deviations can be observed near the inlet branch and around the curved mixing region where strong local pressure gradients are present. These regions are generally more difficult for surrogate models to predict accurately because even small spatial shifts can lead to larger local pressure differences. Nevertheless, despite these localized errors, the overall pressure prediction remains within acceptable limits and demonstrates that the surrogate model is capable of predicting flow physics.



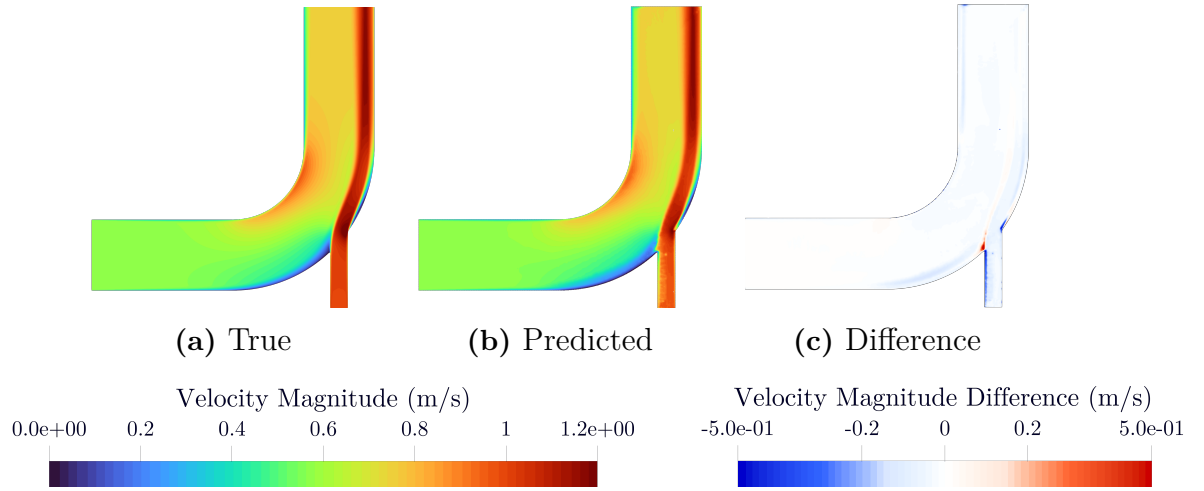
**Figure 4.4:** Comparison between the CFD reference solution and the Altair PhysicsAI prediction for the static pressure field.



**Figure 4.5:** Comparison between the CFD reference solution and the Altair PhysicsAI prediction for the static temperature field.

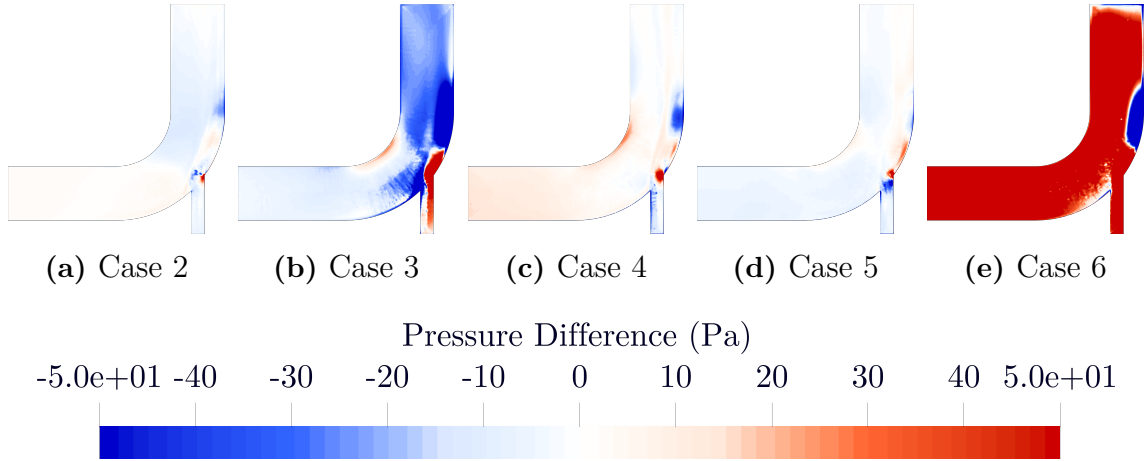
The temperature prediction shown in Figure 4.5 also demonstrates good overall agreement with the CFD reference solution. The surrogate model is able to reproduce the thermal mixing behavior inside the elbow and correctly predicts the high temperature gradients formed between the two inlet streams. Larger deviations are mainly observed near the inlet regions, particularly close to the smaller inlet branch. This behavior is expected because, as discussed previously in the methodology section, the inlet quantities were treated as global KPI parameters instead of directly imposing boundary conditions at specific coordinates. As a result, the model must simultaneously determine the inlet thermal conditions and the resulting internal temperature distribution. Even with this additional complexity, the predicted field remains physically consistent and closely follows the CFD solution over most of the computational domain. The difference plot further confirms that the major deviations are concentrated near the inlet boundaries, while the bulk flow region and the mixing zone show comparatively smaller errors.

A similar trend can be observed for the velocity magnitude prediction shown in Figure 4.6. The surrogate model captures the overall acceleration and flow development through the mixing elbow reasonably well, including the high velocity region near the smaller inlet branch. Similar to the temperature field, the velocity inputs were also provided as KPI based global parameters rather than explicitly prescribed boundary conditions. Because of this, the model is required to learn both the inlet velocity conditions and the resulting flow field simultaneously, which increases the prediction difficulty near the inlet regions. Consequently, localized deviations are visible near the smaller inlet where steep velocity gradients are present. However, outside these localized regions, the predicted velocity field shows good agreement with the CFD reference solution, and the difference plot indicates relatively small deviations over most of the domain. Overall, the Altair PhysicsAI model demonstrates stable interpolation performance for the unseen within range case and is able to reproduce the primary flow and thermal features of the mixing elbow with good accuracy.



**Figure 4.6:** Comparison between the CFD reference solution and the Altair PhysicsAI prediction for the velocity magnitude field.

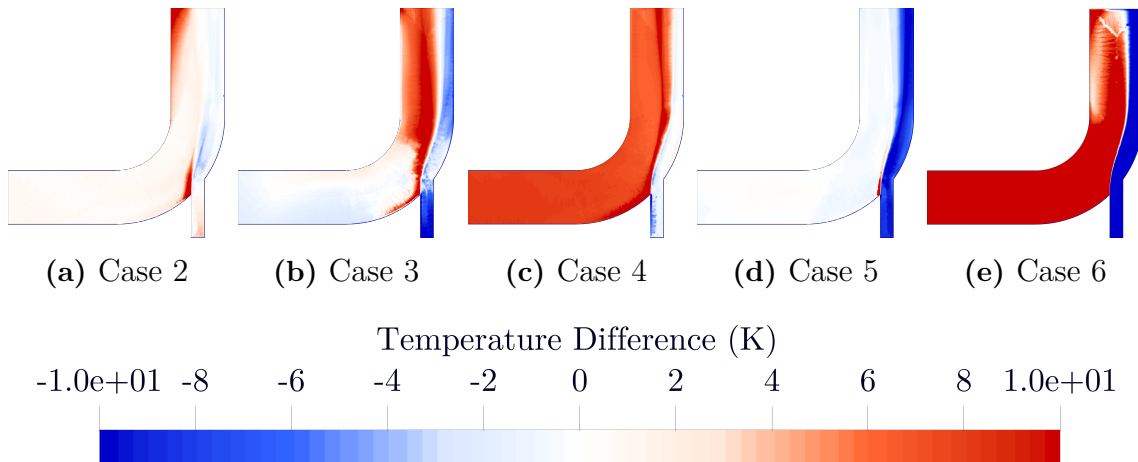
**4.1.2.1.2 Extrapolated Cases:** Figures 4.7, 4.8, and 4.9 present the extrapolation test cases listed in Table 3.6. These cases are important since they evaluate the capability of the Altair PhysicsAI surrogate model to predict flow behavior outside the parameter combinations seen during training. Unlike the within crange interpolation case discussed previously, these extrapolation cases introduce boundary conditions beyond the training distribution, making the prediction task significantly more challenging. The difference contours shown in these figures represent the deviation between the CFD reference solution and the surrogate prediction for each extrapolation case.



**Figure 4.7:** Pressure difference contours for the extrapolation test cases listed in Table 3.6.

From the pressure difference contours shown in Figure 4.7, it can be observed that Cases 2, 4, and 5 have relatively stable bulk flow predictions with few deviations near the inlet branch and curved mixing region. In Case 2, the large inlet velocity is extrapolated to a lower value than the training range. Because of this,

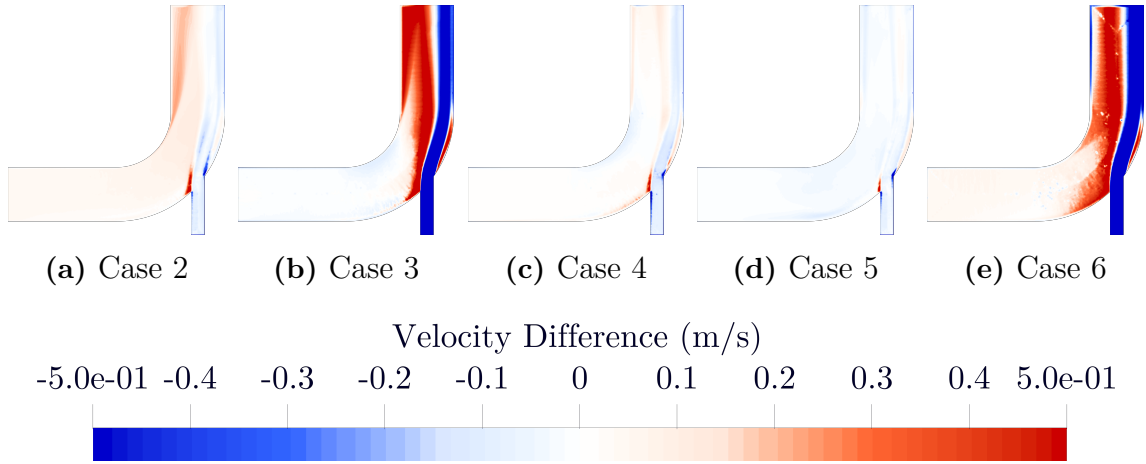
the momentum contribution from the large inlet becomes weaker. This produces localized pressure deviations mainly around the mixing region and bend section. However, since the remaining inputs are still within the training distribution, the overall pressure field remains reasonably stable. In Cases 4 and 5, the extrapolation is performed only on the inlet temperatures while the velocity conditions remain within range. Since the pressure field in this incompressible mixing elbow is mainly governed by momentum transport rather than thermal effects, the surrogate predictions remain comparatively close to the CFD solution with only localized deviations. Among all the single variable extrapolation cases, Case 3 produces the largest pressure deviation. In this case, the small inlet velocity is extrapolated to a much higher value than the training range, creating a stronger jet entering through the side branch. This results in stronger momentum interaction and sharper pressure gradients near the mixing region, which becomes difficult for the surrogate model to predict accurately. Finally, Case 6 shows the highest overall pressure deviation because all the inlet velocities and temperatures are simultaneously extrapolated outside the training distribution. The combined effect of extreme momentum and thermal conditions creates a flow behavior significantly different from the learned physics manifold, causing the surrogate model to lose prediction accuracy over a larger portion of the elbow geometry.



**Figure 4.8:** Temperature difference contours for the extrapolation test cases listed in Table 3.6.

The temperature extrapolation results shown in Figure 4.8 indicate that the surrogate model struggles more for temperature prediction compared to pressure prediction. Among all the extrapolation cases, Case 2 shows relatively lower deviation, although the errors are still present near the outlet region. Since the large inlet velocity is reduced in this case, the penetration of the hot stream becomes weaker, reducing thermal mixing inside the vertical branch. This creates localized temperature deviations near the upper outlet region due to backflow structures observed in the CFD solution. However, such backflow behavior was not present in the training dataset, making it difficult for the surrogate model to accurately reproduce these thermal patterns. A similar trend is observed for Case 3, where the small inlet veloc-

ity is extrapolated to a much higher value. The stronger side jet creates recirculation near the outlet branch and significantly changes the thermal mixing behavior inside the elbow. Since this type of recirculation dominated flow structure was also not part of the training distribution, the temperature prediction error becomes much larger near the outlet region. In Cases 4 and 5, where the large and small inlet temperatures are extrapolated individually, the model struggles to accurately predict the temperature field outside the trained thermal range. This indicates that the thermal field learned by the surrogate model is more sensitive to extrapolation compared to the pressure field. Finally, in Case 6, where all inlet conditions are extrapolated simultaneously, the temperature prediction shows very large deviations throughout the domain, indicating that the surrogate model is unable to generalize accurately under strongly extrapolated thermal and flow conditions.



**Figure 4.9:** Velocity difference contours for the extrapolation test cases listed in Table 3.6.

The velocity deviation contours shown in Figure 4.9 shows that the extrapolation behavior is primarily influenced by the inlet velocity conditions rather than the inlet temperatures. Case 2 exhibits relatively smaller velocity deviations because only the large inlet velocity is reduced, leading to weaker momentum transport but limited changes to the overall flow structure. Most of the bulk flow region still shows comparatively lower deviation. Case 3 produces the largest velocity deviation among the single variable extrapolation cases since the small inlet velocity is increased significantly beyond the training range. The stronger jet interaction generates sharper velocity gradients and stronger mixing near the junction region, making the prediction more difficult for the surrogate model. In contrast, Cases 4 and 5 show relatively smaller velocity deviations because only the temperature boundary conditions are extrapolated while the momentum related inlet conditions remain within the training range. As a result, the global velocity structure remains closer to the CFD solution. Similar to the pressure and temperature fields, Case 6 produces the highest overall velocity deviation due to the simultaneous extrapolation of all inlet velocities and temperatures, causing the predicted flow structure and mixing behavior to deviate significantly from the learned training distribution.

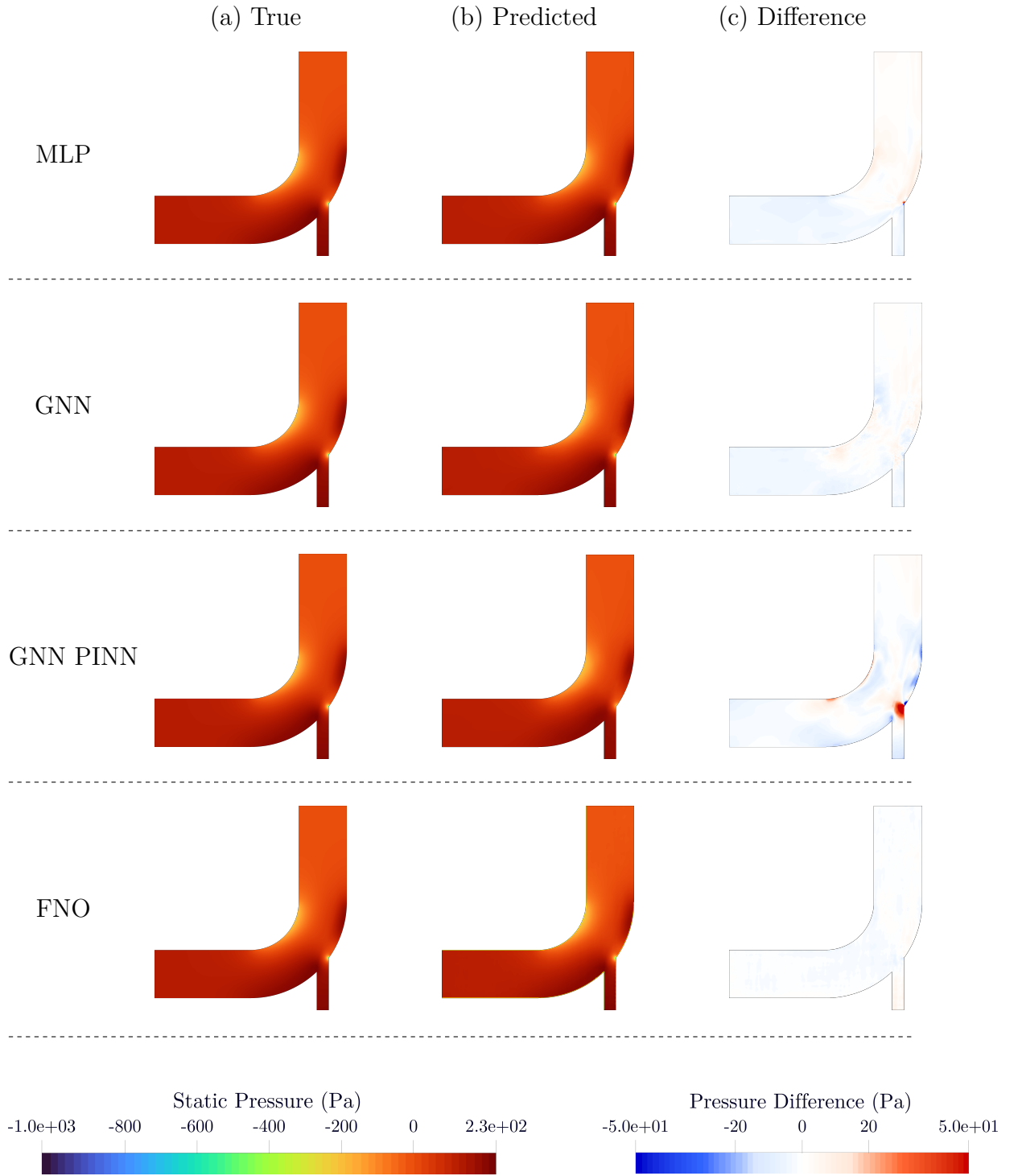


Another important aspect is that the surrogate model uses the inlet conditions as KPI based global parameters rather than explicitly prescribing the boundary conditions at fixed spatial coordinates. Because of this, the model must simultaneously determine both the inlet boundary behavior and the resulting internal flow field from the global parameter representation. This effect becomes more visible in the extrapolation cases whenever the inlet conditions move outside the training distribution. For example, in Cases 4 and 5, where the inlet temperatures are extrapolated beyond the trained range, we observe that there is moderate to high temperature deviations near the inlet regions in Figure 4.8. Similarly, in the velocity difference contours shown in Figure 4.9, Cases 2 and 3 show larger deviations near the inlet branches where the inlet velocities are extrapolated. This behavior is expected because the surrogate model is required to reconstruct unseen inlet boundary conditions from the KPI representation while also predicting the internal flow evolution. Nevertheless, despite these localized inlet deviations, some parts of the bulk flow regions still maintain comparatively stable predictions for most of the extrapolation cases, indicating that the model is still able to capture the global flow behavior reasonably well.

#### 4.1.2.2 NVIDIA PhysicsNeMo Results

This section presents the results obtained using the different NVIDIA PhysicsNeMo based surrogate models investigated in the present study. The performance of the MLP, GNN, GNN-PINN, and FNO architectures is evaluated by comparing their predicted flow and thermal fields against the CFD reference solutions. The analysis includes both the within-range unseen case and the extrapolation test cases listed in Table 3.6. The within-range case is used to evaluate the interpolation capability of the models for unseen operating conditions that still remain inside the training distribution, while the extrapolation cases are used to assess the generalization capability of the different architectures outside the trained parameter space. The objective of this comparison is to understand how different AI architectures perform for CFD surrogate modelling in terms of prediction accuracy, physical consistency, and robustness for both interpolation and extrapolation conditions.

**4.1.2.2.1 Within Range Unseen Case:** The predictions obtained using the different PhysicsNeMo based surrogate models are presented in Figures 4.10, 4.11, and 4.12. The figures compare the CFD reference solution with the predictions obtained using the MLP, GNN, GNN-PINN, and FNO models for the static pressure, temperature, and velocity magnitude fields, respectively. Similar to the previous Altair PhysicsAI analysis, this case corresponds to an unseen interpolation condition where the boundary conditions remain within the training distribution. The objective of this comparison is to evaluate how different AI architectures perform for CFD field prediction under the same flow condition while also identifying the strengths and limitations associated with each approach.

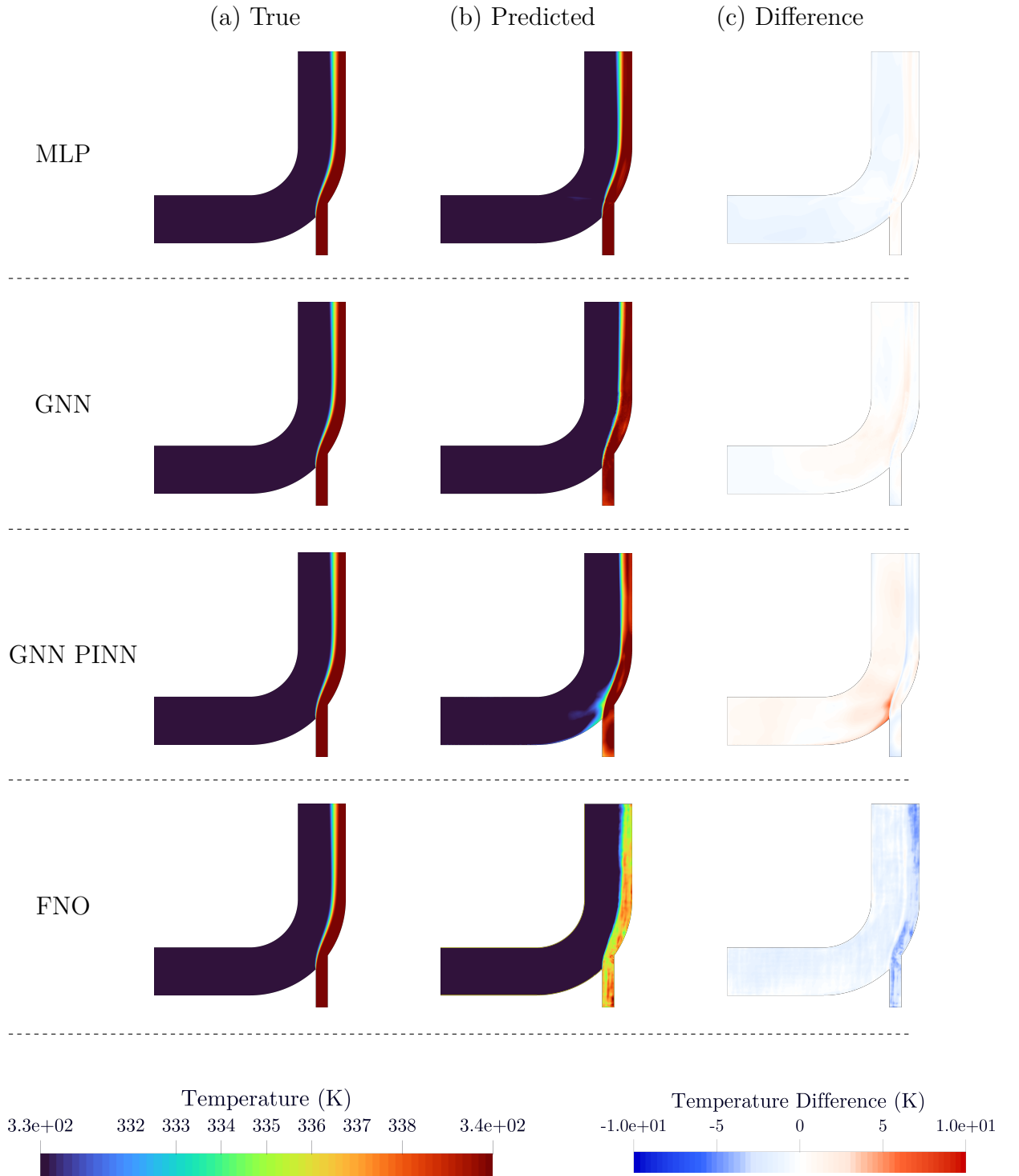


**Figure 4.10:** Comparison between the CFD reference solution and the predictions of MLP, GNN, GNN-PINN, and FNO models for the static pressure field.

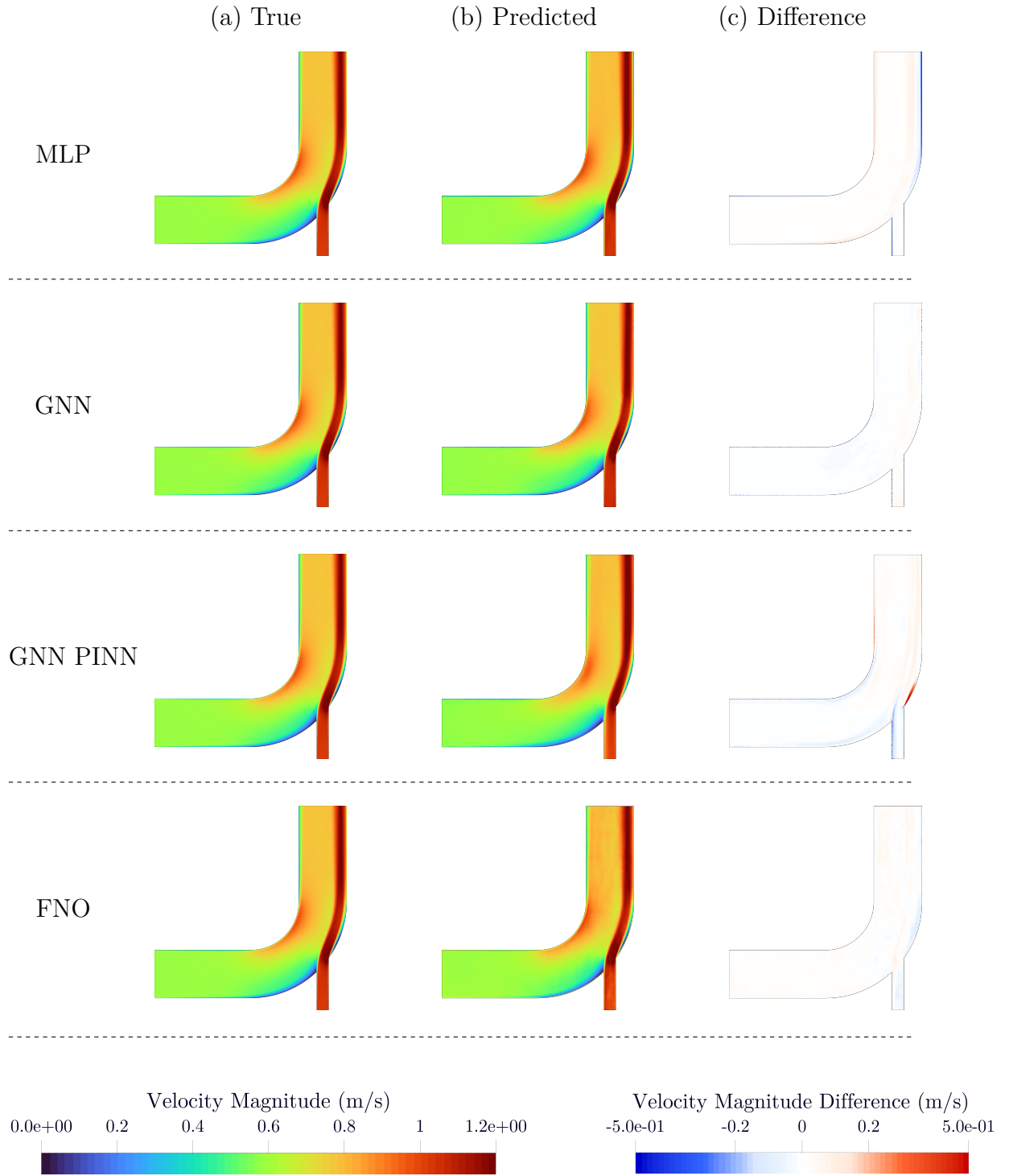
From Figure 4.10, it can be observed that all the investigated models are able to predict the pressure field reasonably well for the within-range unseen case, which is expected since the operating condition lies inside the trained parameter distribution. The MLP and GNN models show very small deviations over most of the

computational domain and both models are able to reproduce the overall pressure distribution accurately. However, in the MLP prediction, a small localized deviation can be observed near the inlet branch junction where strong local pressure gradients exist. This localized error is comparatively smaller in the GNN model because the graph based architecture utilizes mesh connectivity and local node interactions through message passing, allowing improved representation of local flow physics near complex regions. The GNN-PINN model shows comparatively larger deviations across the pressure field. This behaviour is likely related to the limitations associated with the present physics informed implementation. In the present work, the implemented PINN formulation is physics informed rather than fully physics driven, since only continuity and momentum residuals were incorporated into the loss function together with the supervised data loss. Although this improves physical consistency, the imposed physics constraints may still be insufficient to fully capture all localized flow behaviour, especially near regions with strong pressure gradients and mixing interactions. Among all the investigated architectures, the FNO model provides the best overall pressure prediction with the smallest difference contours. This is likely because the Fourier based operator learning approach efficiently captures smooth global spatial structures and large scale pressure variations across the structured computational domain.

For the temperature field shown in Figure 4.11, the MLP and GNN models again predict the overall thermal distribution reasonably well, although localized deviations can still be observed near the thermal mixing region where sharp temperature gradients are present. Temperature transport depends strongly on coupled convection and diffusion behaviour, making the thermal field comparatively more difficult to predict than pressure. The GNN-PINN model shows larger localized temperature errors despite the physics informed formulation. One important reason for this behaviour is that the implemented PINN constraints only included continuity and momentum equations, while the energy equation was not incorporated into the physics informed loss formulation. As a result, the model improves flow consistency but does not directly enforce thermal conservation within the learning process, leading to inaccuracies near strong thermal gradients and mixing regions. The FNO model shows comparatively weaker temperature predictions than the MLP and GNN models, particularly near the thermal mixing region where sharper temperature gradients are present. Although the overall thermal distribution is still captured reasonably well, larger localized deviations can be observed throughout the computational domain. One possible reason for this behaviour is related to the preprocessing requirement associated with FNO training. Since the original CFD data was generated on an unstructured mesh, the solution fields had to be interpolated onto a structured grid before training the FNO model. This structured conversion process may smooth or slightly distort the sharp thermal gradients present in the original CFD solution, which can reduce the prediction accuracy for the temperature field.



**Figure 4.11:** Comparison between the CFD reference solution and the predictions of MLP, GNN, GNN-PINN, and FNO models for the temperature field.



**Figure 4.12:** Comparison between the CFD reference solution and the predictions of MLP, GNN, GNN-PINN, and FNO models for the velocity magnitude field.

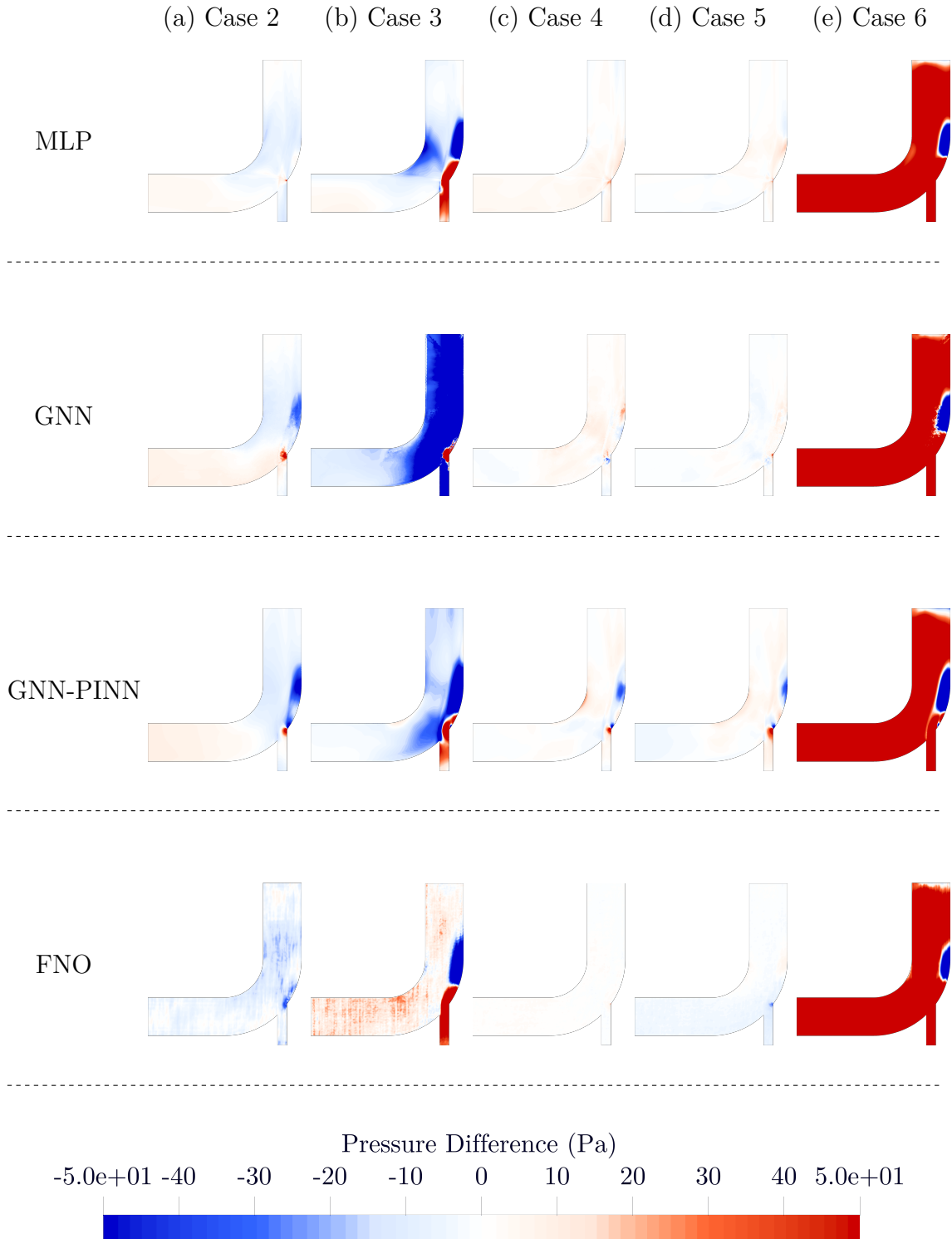
The velocity magnitude predictions shown in Figure 4.12 indicate that all the investigated models are able to predict the global velocity field with relatively small deviations. The MLP and GNN models again show similar overall behaviour, although the MLP exhibits small localized deviations near the outlet branch wall re-

gion which are comparatively smaller in the GNN prediction. This improvement can again be attributed to the graph based learning approach, where the GNN utilizes mesh connectivity and local node interactions to better capture spatial flow features and local velocity gradients. The GNN-PINN model performs comparatively better for velocity prediction than for pressure and temperature because the implemented physics informed constraints are directly based on continuity and momentum conservation equations, which are strongly related to the velocity field. Nevertheless, some localized deviations are still present near regions with stronger flow interaction and velocity gradients, indicating that the present physics informed formulation still has limitations in fully capturing all local flow structures. The FNO model also performs very well for velocity magnitude prediction with very small deviation contours over most of the computational domain. Compared to the temperature field, the velocity structures are comparatively smoother and easier to represent in Fourier space, allowing the FNO model to efficiently capture the dominant large scale flow behaviour with good overall prediction accuracy.

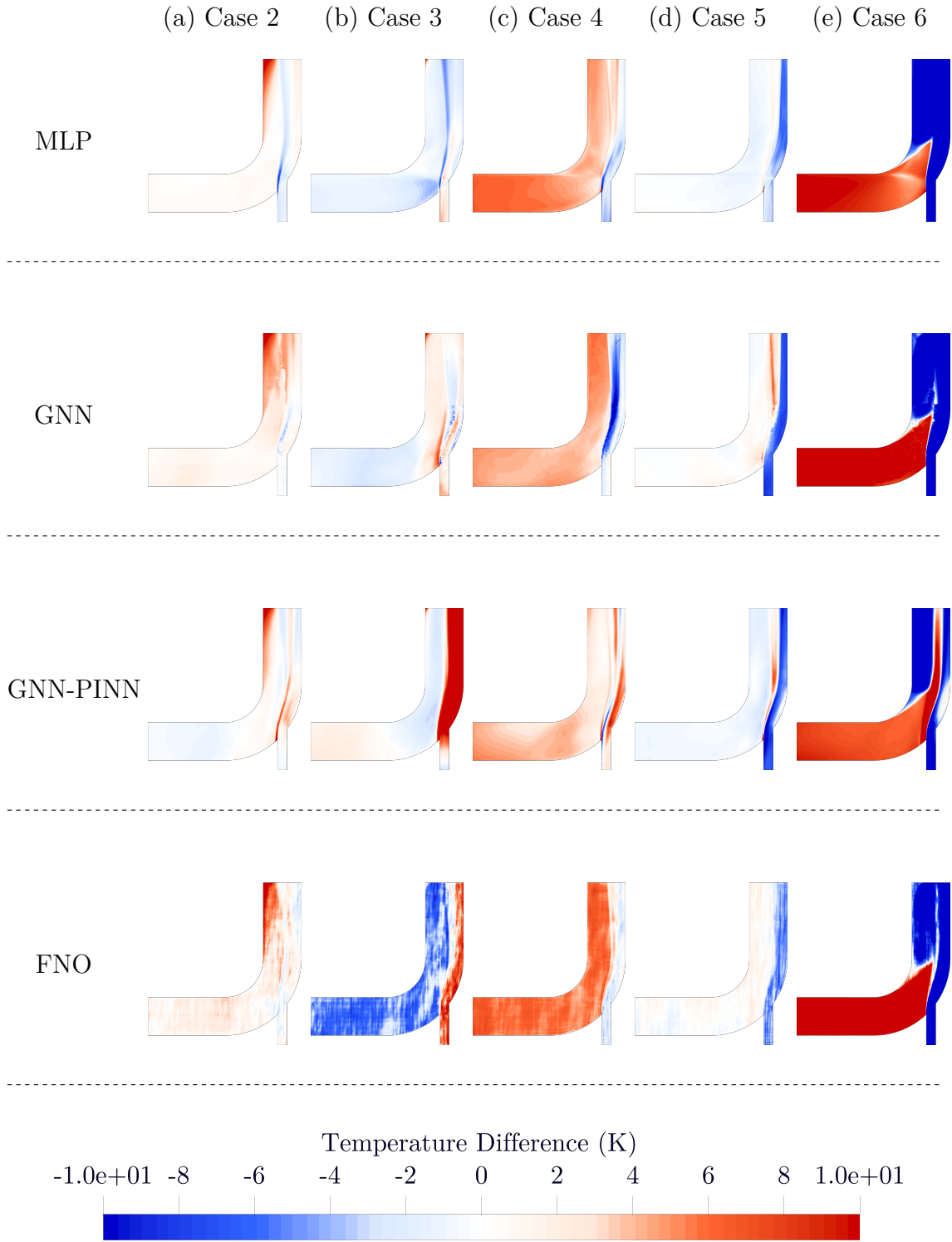
**4.1.2.2.2 Extrapolation Cases:** The extrapolation performance of the different PhysicsNeMo models is evaluated using the pressure, temperature, and velocity difference contours shown in Figures 4.13, 4.14, and 4.15. Similar to the Altair PhysicsAI results discussed previously, the extrapolation cases provide important insight into how well the surrogate models are able to predict flow behaviour outside the training distribution defined in Table 3.6.

For the pressure difference contours shown in Figure 4.13, all the models show relatively similar prediction behaviour for Cases 2, 4, and 5. In Case 2, the large inlet velocity is extrapolated to a lower value than the training range, and therefore the momentum contribution from the large inlet becomes weaker. Since the remaining input parameters still remain within the training range, the overall pressure field does not change drastically, and most models are still able to predict the bulk pressure distribution reasonably well. Here, the MLP and FNO models show slightly lower deviations when compared to the GNN and GNN-PINN models. For Cases 4 and 5, where only the inlet temperatures are extrapolated, all the models show relatively small pressure deviations because the pressure field in the incompressible mixing elbow case is primarily governed by momentum transport rather than thermal effects. Similar to the observations from the Altair PhysicsAI model, the largest single variable extrapolation error occurs in Case 3, where the small inlet velocity is increased beyond the training range. The stronger jet entering from the side branch produces higher momentum interaction and sharper pressure gradients near the mixing region, resulting in larger localized deviations. However, an interesting observation is that the GNN-PINN model shows slightly reduced deviations compared to the standard GNN model in this case, indicating that the inclusion of continuity and momentum based physics losses helps improve the prediction capability to some extent under stronger extrapolation conditions. Finally, Case 6 shows the largest overall deviation for all models since all inlet velocities and temperatures are simultaneously extrapolated outside the training distribution. Under these conditions, the resulting flow field differs significantly from the learned solution

manifold, and therefore all models struggle to maintain accurate prediction across the entire elbow region.



**Figure 4.13:** Comparison of pressure difference contours for the extrapolation test cases listed in Table 3.6 using MLP, GNN, GNN-PINN, and FNO models.



**Figure 4.14:** Comparison of temperature difference contours for the extrapolation test cases listed in Table 3.6 using MLP, GNN, GNN-PINN, and FNO models.

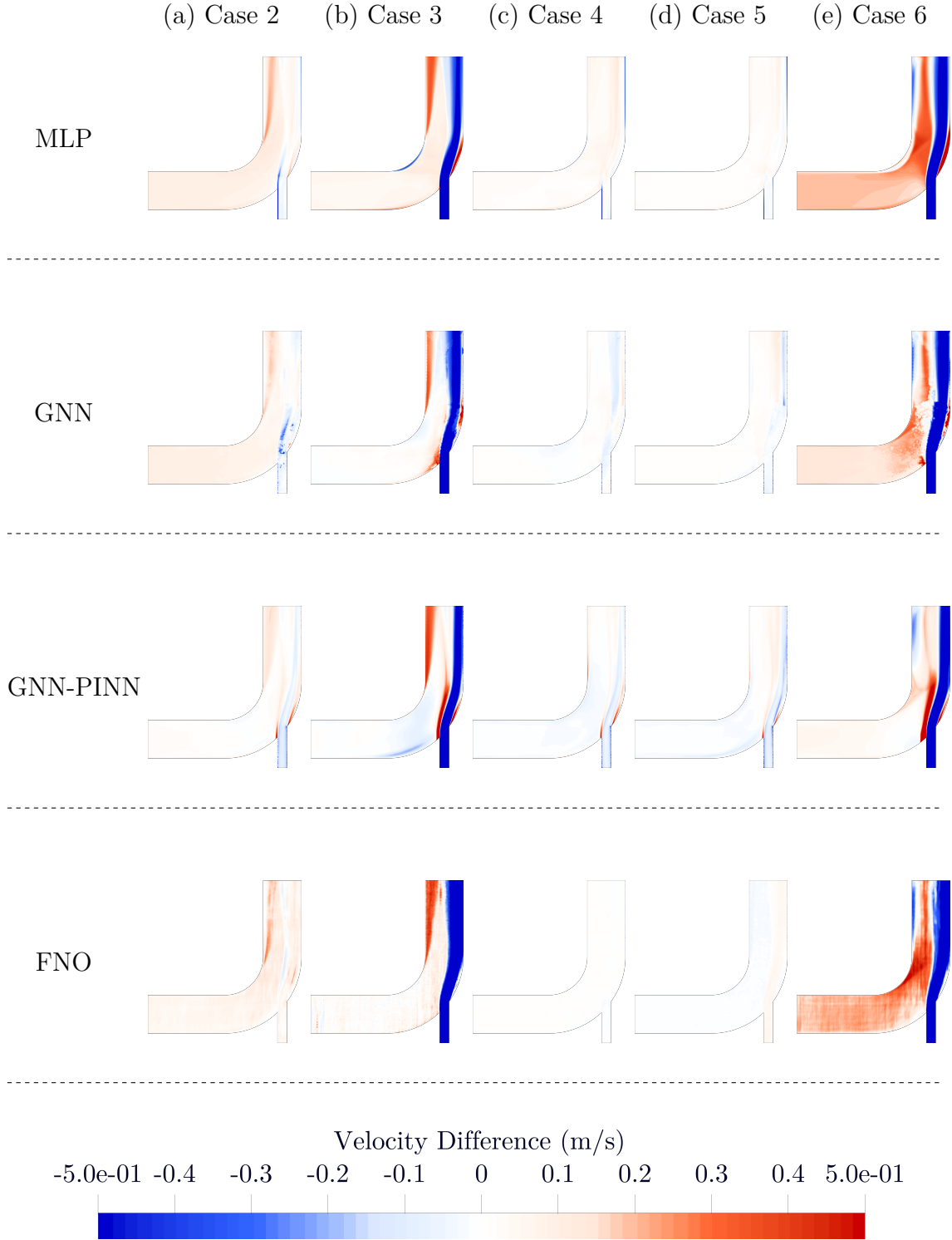
For the temperature difference contours shown in Figure 4.14, most of the models struggle to accurately predict the thermal field under extrapolation conditions. Among all the architectures, the FNO model shows comparatively larger tempera-



ture deviations than the other models. Although the overall thermal field behaviour is still captured reasonably well, noticeable distributed deviations can be observed throughout the computational domain. As discussed previously, one possible reason for this behaviour is related to the preprocessing requirement associated with FNO training, where the original unstructured CFD data must first be interpolated onto structured grids. This structured conversion process can introduce some losses especially near the thermal mixing regions, which affects the temperature prediction accuracy during extrapolation cases. One important reason for this behaviour is that the inlet conditions were provided to the models as global KPI parameters rather than being directly enforced at the corresponding inlet nodal locations. Because of this, the models must simultaneously determine both the thermal boundary condition and the resulting field behaviour, which becomes more difficult during extrapolation. Similar behaviour is also observed in Cases 2 and 3, where only the inlet velocities are varied. In these cases, the modified momentum interaction changes the thermal mixing behaviour significantly. For Case 2, the reduced large inlet velocity weakens the hot stream penetration and creates backflow behaviour near the outlet region, while in Case 3 the higher small inlet velocity generates stronger recirculation and reverse flow structures near the outlet branch. Since such flow structures were not sufficiently represented within the training dataset, the networks struggle to accurately reproduce the thermal field under these conditions. Nevertheless, despite these localized deviations, the bulk thermal mixing behaviour is still captured reasonably well by the MLP, GNN, and GNN-PINN models. As expected, Case 6 shows the largest overall temperature deviation since all boundary conditions are simultaneously extrapolated beyond the training distribution.

For the velocity difference contours shown in Figure 4.15, the extrapolation behaviour is mainly governed by the inlet velocity conditions rather than the temperature variations. Case 2 shows relatively smaller deviations because only the large inlet velocity is reduced, leading to weaker momentum transport but limited change in the bulk flow. In contrast, Case 3 produces the largest velocity deviations among the single variable extrapolation cases because the small inlet velocity is increased significantly beyond the training range, generating stronger jet interaction and sharper velocity gradients near the mixing region. Similar to the pressure field observations, the GNN-PINN model shows slightly improved prediction behaviour compared to the standard GNN model under this stronger extrapolation condition, indicating some benefit from the additional physics based constraints. Cases 4 and 5 show comparatively small velocity deviations since only the inlet temperatures are extrapolated while the velocity boundary conditions remain within the training range. The FNO model shows comparatively larger deviations than the graph based models mainly for Cases 2 and 3, where the inlet velocity conditions are extrapolated outside the training range. Since these cases introduce stronger momentum interaction and sharper velocity gradients near the mixing region, the Fourier based representation struggles comparatively more in capturing these localized changes. In contrast, Cases 4 and 5 show relatively smaller deviations for the FNO model because only the inlet temperatures are varied while the global velocity field structure remains comparatively similar to the training cases. Finally, Case 6 produces

the largest overall velocity deviation because all inlet conditions are extrapolated simultaneously, leading to flow behaviour that differs substantially from the cases observed during training.



**Figure 4.15:** Comparison of Velocity difference contours for the extrapolation test cases listed in Table 3.6 using MLP, GNN, GNN-PINN, and FNO models.

### 4.1.3 Three Dimensional Mixing Elbow Results

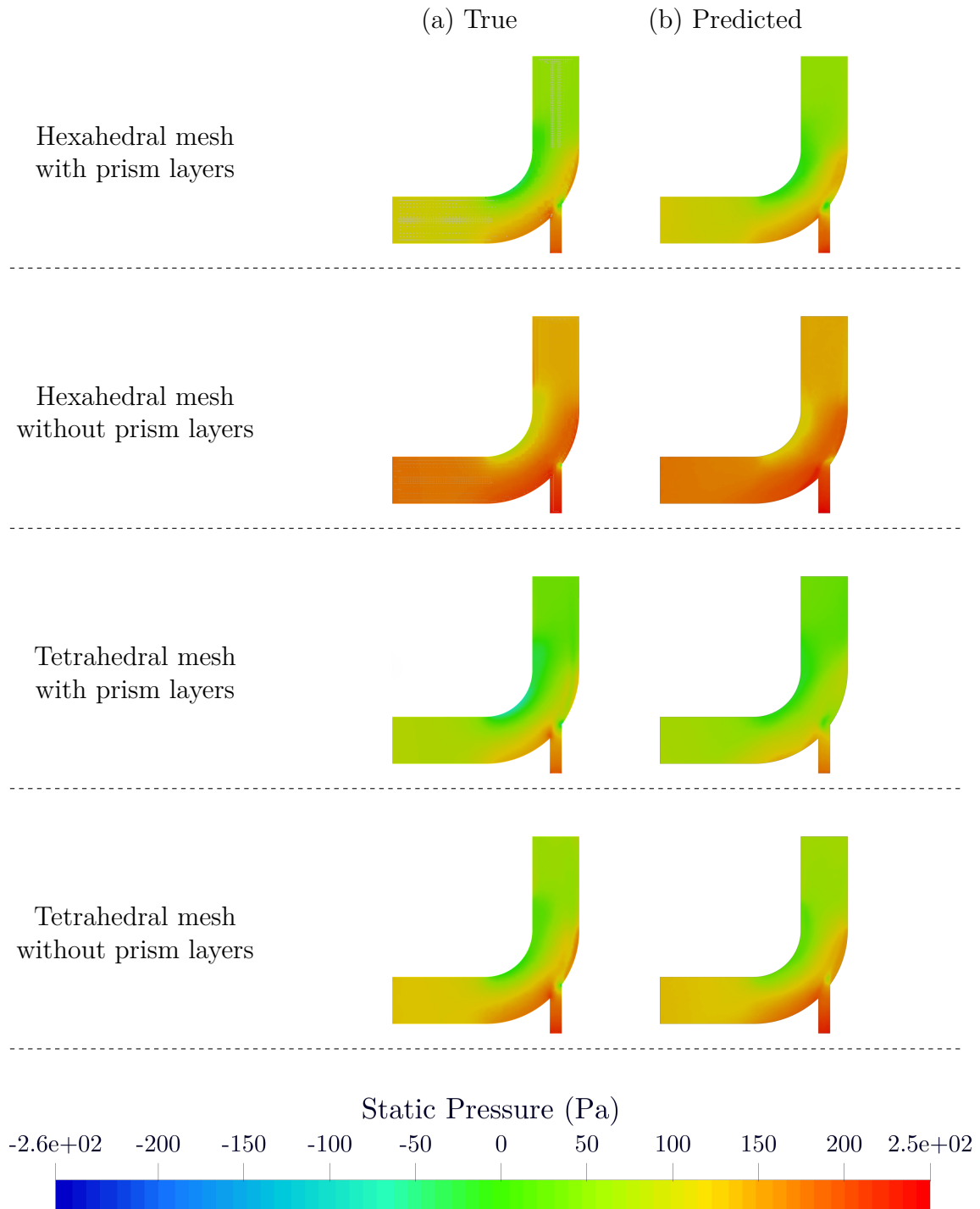
In this section, the results obtained for the three dimensional mixing elbow case are discussed. The prediction capability of both Altair PhysicsAI and NVIDIA PhysicsNeMo based models, including MLP, GNN, and Physics Informed GNN architectures, is evaluated through comparison with the CFD reference solutions. The discussion focuses on the prediction of pressure, temperature, and velocity fields for the three dimensional configuration.

#### 4.1.3.1 Altair PhysicsAI Results

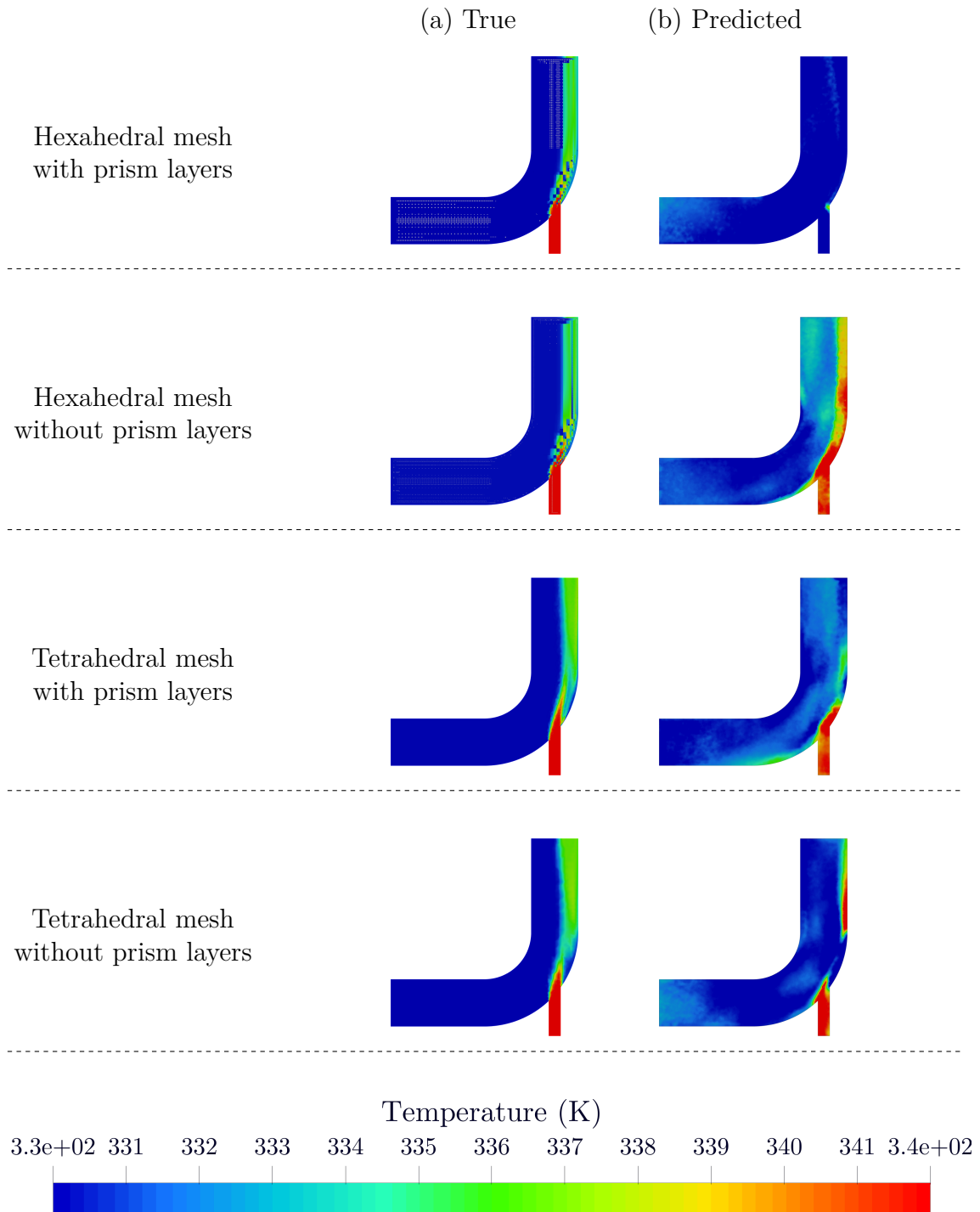
For the Altair PhysicsAI workflow, four different mesh configurations discussed previously in Section 3.2.1.1 were investigated, namely hexahedral meshes with and without prism layers, and tetrahedral meshes with and without prism layers. The prediction performance of the trained AI model for pressure, temperature, and velocity fields is compared against the corresponding CFD reference solutions for the unseen validation case listed in Table 3.6. Unlike the two dimensional study, further extrapolation investigations were not performed for the three dimensional case since the initial unseen case predictions themselves did not show sufficiently satisfactory performance for extended extrapolation analysis.

The pressure contour predictions shown in Figure 4.16 indicate that Altair PhysicsAI provides fairly good agreement with the CFD reference solutions for all investigated mesh configurations. The overall pressure distribution within the elbow region and outlet branch is reasonably captured for both hexahedral and tetrahedral meshes. However, for the hexahedral mesh cases with and without prism layers, visible artifacts can be observed in the reference contour plots. As discussed previously in the methodology section, these artifacts are mainly associated with the internal mesh decimation and mesh conversion procedures performed within the PhysicsAI workflow together with the limited support for hexahedral dominant meshes during prediction. Nevertheless, despite these limitations, the AI model is still able to reproduce the overall pressure field reasonably well across the computational domain.

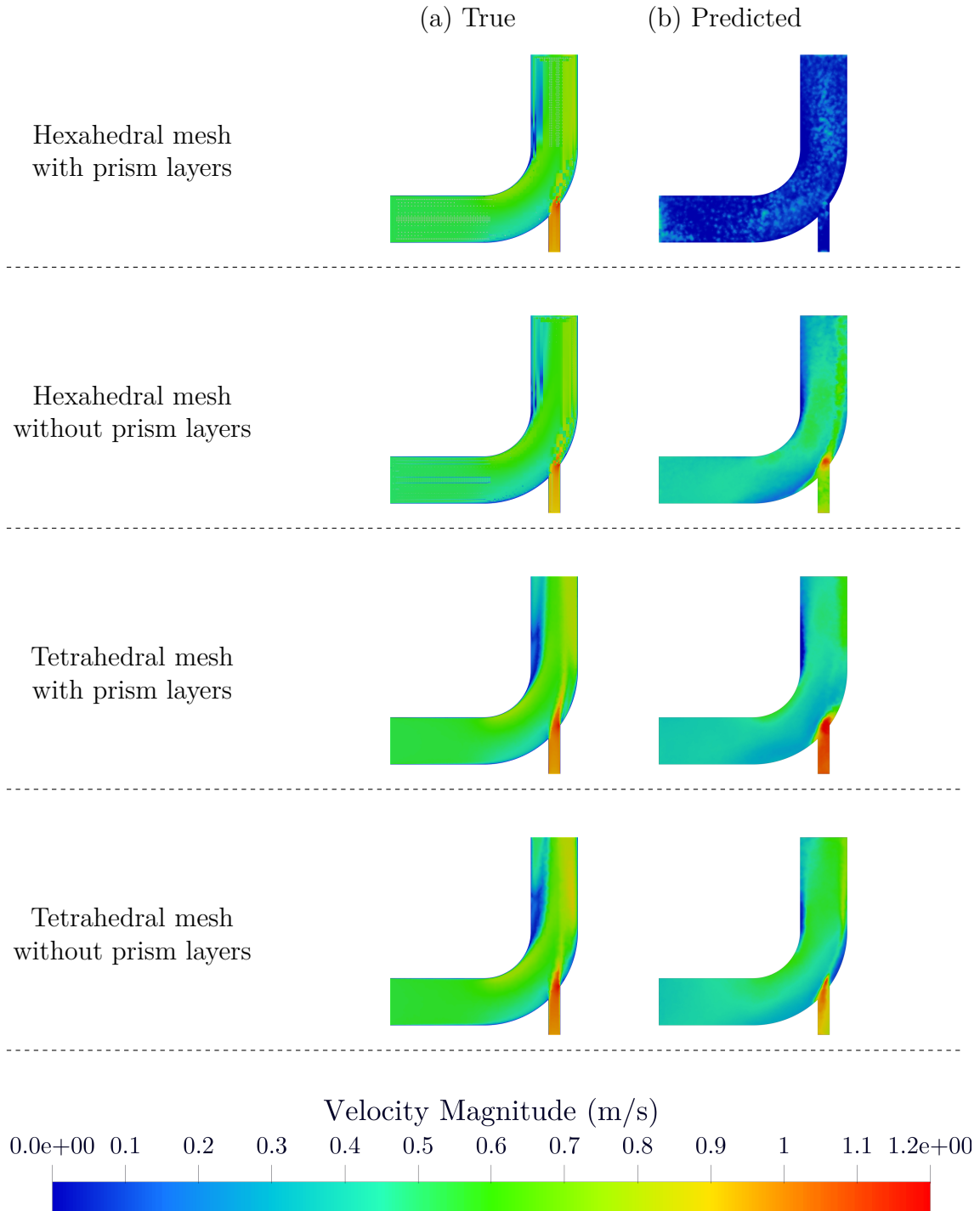
For the temperature field predictions shown in Figure 4.17, the model fails to accurately predict the thermal distribution for most mesh configurations. Significant deviations from the CFD reference solution can be observed throughout the domain, particularly near the mixing region and outlet branch. Similar to the pressure contours, noticeable artifacts are again present for the hexahedral mesh cases due to the mesh decimation and conversion procedures within the PhysicsAI workflow. The tetrahedral mesh cases show relatively better behaviour compared to the hexahedral meshes, where the model is at least able to partially recognize the inlet thermal boundary conditions and the general thermal gradient distribution. However, the overall temperature prediction quality still remains unsatisfactory for all investigated mesh configurations.



**Figure 4.16:** Comparison between the CFD reference solution and the predicted static pressure contours for different mesh types.



**Figure 4.17:** Comparison between the CFD reference solution and the predicted temperature contours for different mesh types.



**Figure 4.18:** Comparison between the CFD reference solution and the predicted velocity magnitude contours for different mesh types.

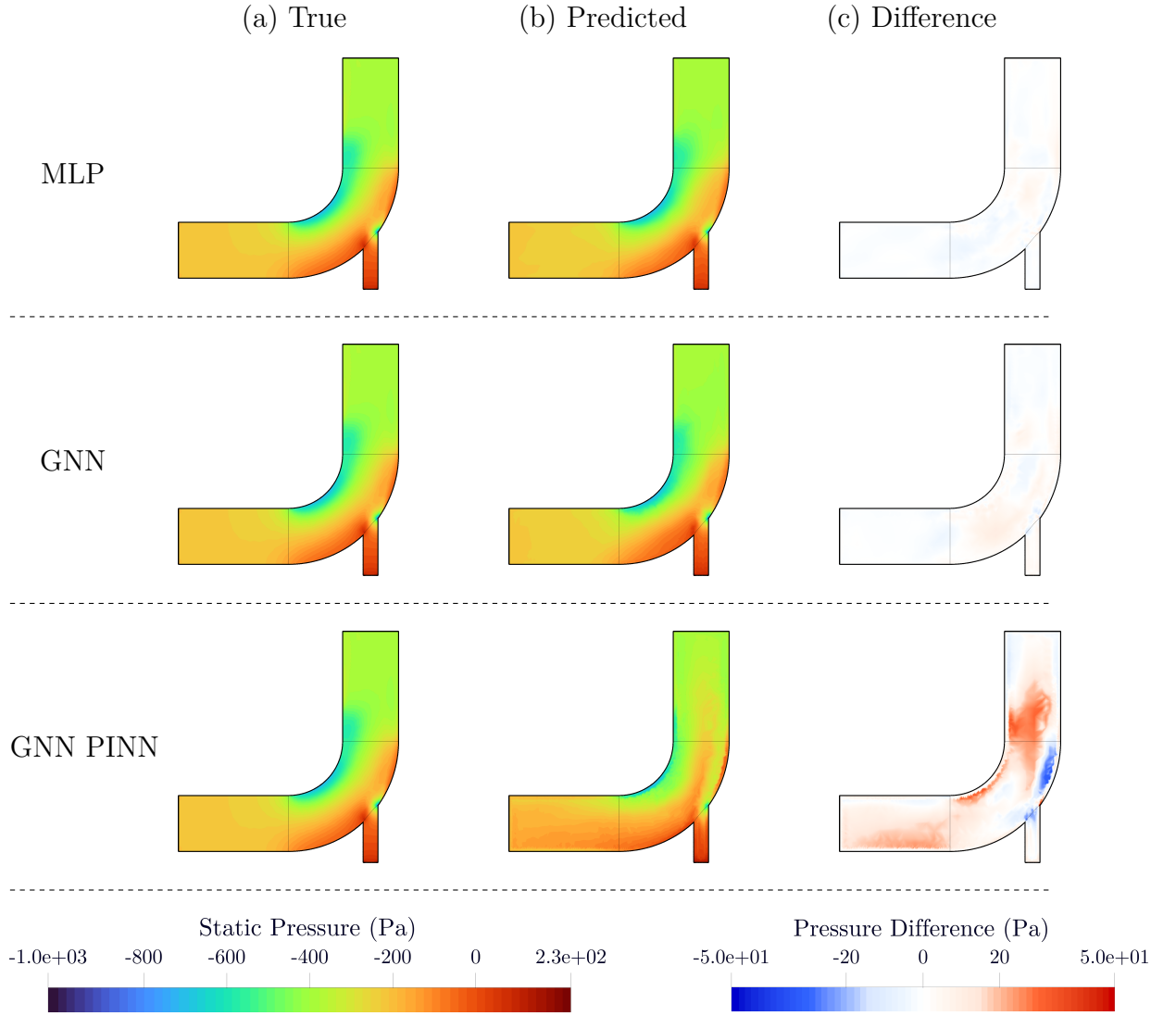
The velocity magnitude contours shown in Figure 4.18 indicate similar behaviour to the temperature predictions. The model is unable to accurately reproduce the complete velocity field across the domain, and visible artifacts are again observed particularly for the hexahedral mesh configurations. The tetrahedral mesh cases provide comparatively better predictions, where the general bulk flow behaviour

and high velocity regions near the outlet branch are captured more reasonably. This improved behaviour is likely related to the better compatibility of tetrahedral meshes within the PhysicsAI workflow. However, despite the improved qualitative behaviour for tetrahedral meshes, the overall prediction accuracy for the velocity field is still limited, especially near wall regions and within the mixing zone where strong flow interactions and recirculation effects are present.

#### 4.1.3.2 NVIDIA PhysicsNeMo Results

In this section, the prediction capability of the MLP, GNN, and Physics Informed GNN models developed using the NVIDIA PhysicsNeMo framework is discussed. The performance of the models is evaluated by comparing the predicted pressure, temperature, and velocity fields against the CFD reference solutions for the three dimensional mixing elbow case. Similar to the two dimensional study, both within range unseen cases and extrapolation cases listed in Table 3.6 were investigated. For the extrapolated cases, only the difference contours are presented and discussed, while the corresponding true CFD reference contours are provided in Appendix A.2.

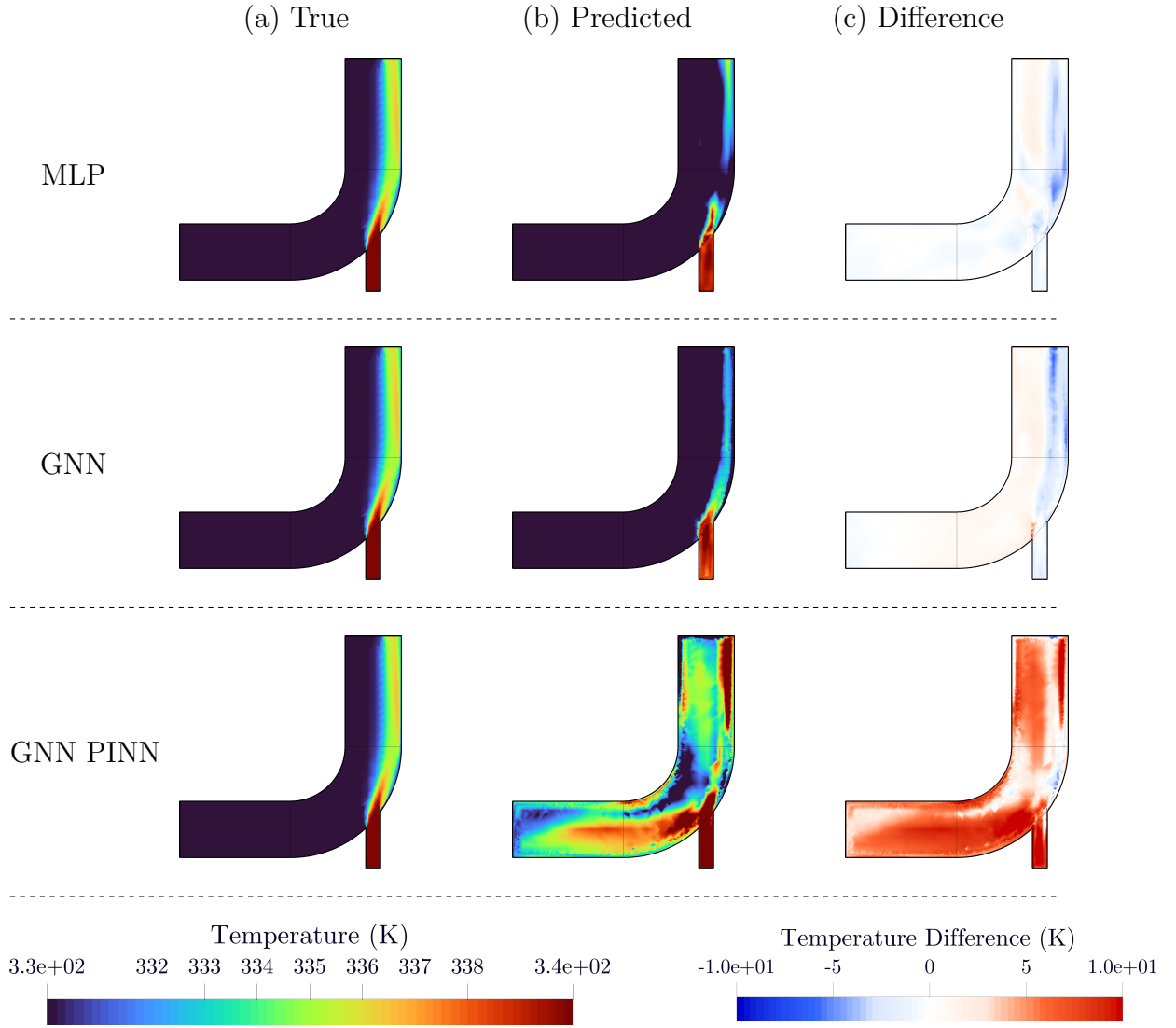
**4.1.3.2.1 Within range unseen case:** The pressure contour predictions shown in Figure 4.19 indicate that both the MLP and GNN models have performed very well with very less deviation between the true and predicted fields. The overall pressure distribution within the elbow region and outlet branch is captured accurately, and no major difference can be observed between the MLP and GNN predictions. However, when we move towards the GNN-PINN model, larger deviations can be observed across the computational domain. This is mainly due to the way the Physics Informed GNN model was trained. As discussed previously in the methodology section, for the three dimensional case the number of nodal points used during physics informed training was reduced because of GPU memory limitations, since the framework had to simultaneously store both the neural network computations and the PDE residual calculations. Due to these limitations, the current approach of reducing the number of nodal points did not provide satisfactory performance for the present three dimensional implementation. Nevertheless, the MLP and GNN results still demonstrate promising prediction capability of the PhysicsNeMo framework for pressure field prediction.



**Figure 4.19:** Comparison between the CFD reference solution and the predictions of MLP, GNN and GNN-PINN models for the static pressure field.

For the temperature contours shown in Figure 4.20, all three models show larger deviations when compared to the pressure field predictions. Again, the GNN-PINN model shows very high deviations throughout the domain, which is strongly influenced by the reduced nodal point strategy adopted during physics informed training. The MLP model is able to reasonably capture the bulk thermal field and the inlet thermal boundary conditions, but struggles to accurately predict the temperature gradients within the mixing region and outlet branch. Among the three models, the GNN performs comparatively better, where the thermal gradients are at least partially captured together with the bulk flow thermal behaviour. Overall, the PhysicsNeMo models do not perform as well for the temperature field compared to the pressure predictions, mainly because the thermal field contains sharper localized gradients and stronger coupling between convection and thermal mixing behaviour.

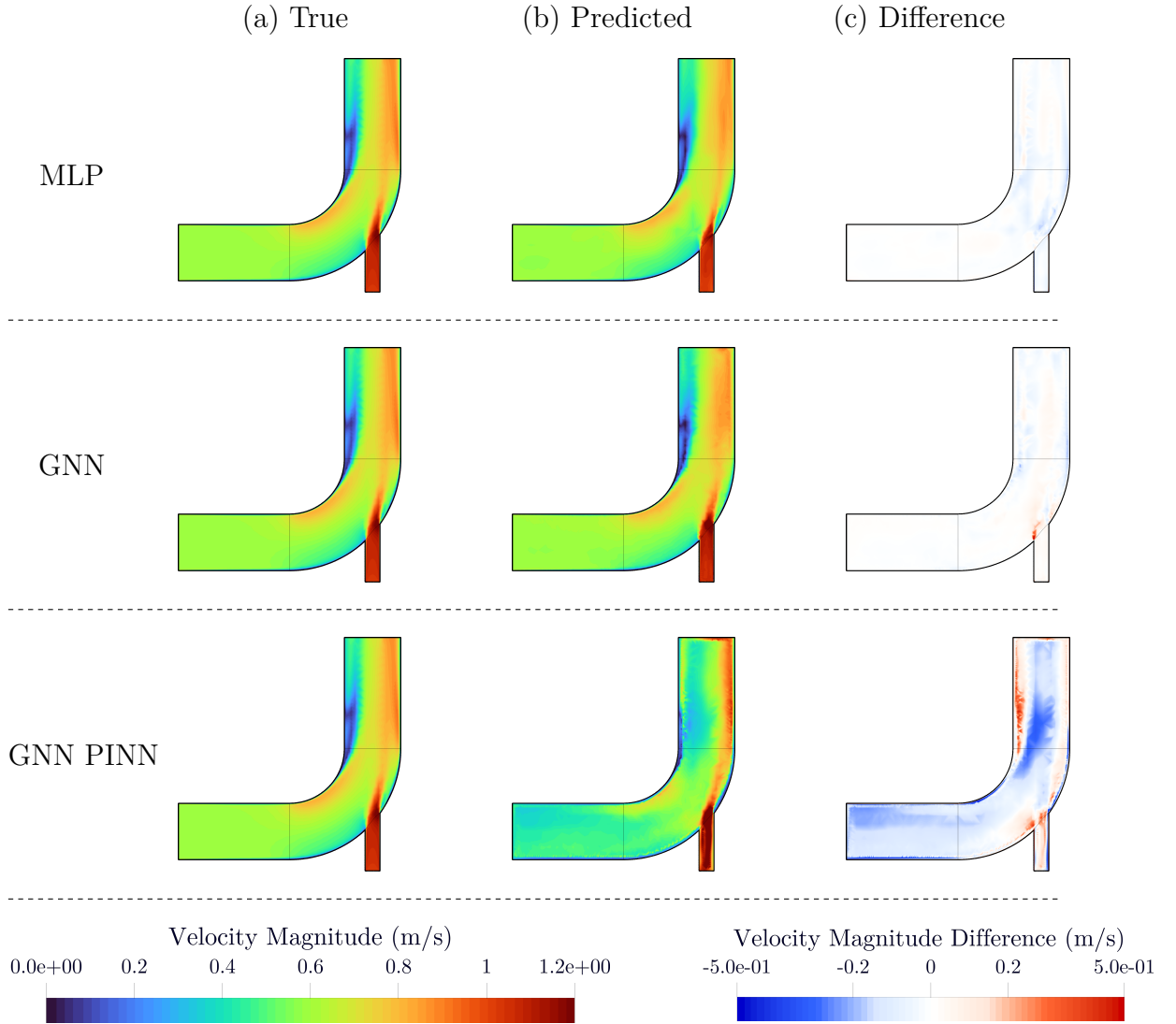




**Figure 4.20:** Temperature Difference between the CFD reference solution and the predictions of MLP, GNN and GNN-PINN models.

The velocity magnitude contours shown in Figure 4.21 indicate behaviour similar to the pressure field predictions. Both the MLP and GNN models have performed very well with very less deviation between the CFD reference solution and the predicted velocity field. The overall bulk flow structure, high velocity regions near the outlet branch, and the flow development through the curved elbow section are captured reasonably well by both models. Among the two, the GNN shows slightly better local prediction behaviour near the wall and outlet regions due to its ability to utilize mesh connectivity and local nodal interactions through the graph based message passing framework. Similar to the pressure and temperature fields, the GNN-PINN model again shows comparatively large deviations because of the reduced nodal point approach adopted for the three dimensional physics informed training process. Nevertheless, the MLP and GNN models still demonstrate that PhysicsNeMo based surrogate modelling has promising capability for prediction of complex three

dimensional velocity fields.

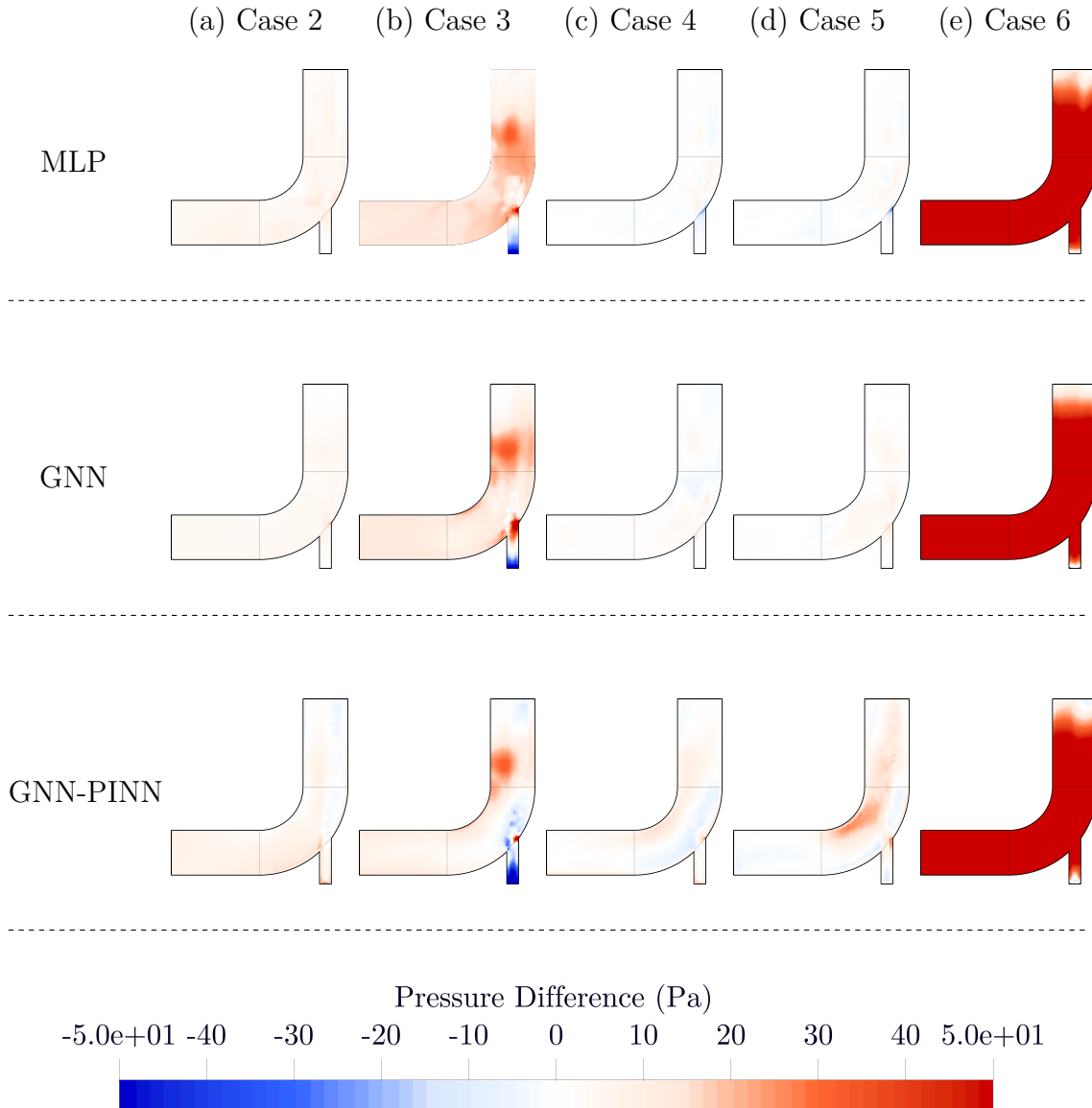


**Figure 4.21:** Velocity Difference between the CFD reference solution and the predictions of MLP, GNN and GNN-PINN models.

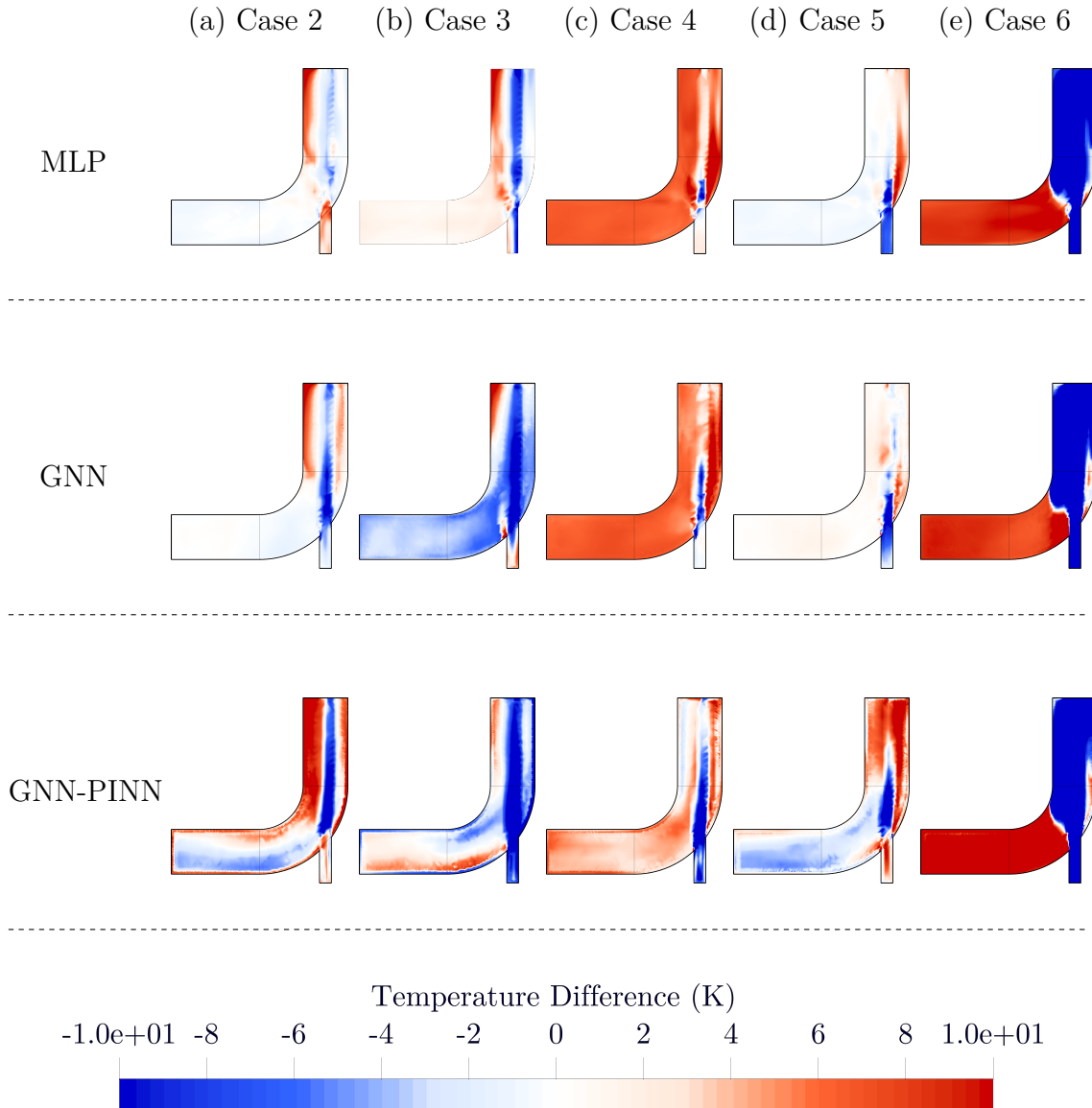
**4.1.3.2.2 Extrapolated Cases:** In this section, the extrapolated test cases listed in Table 3.6 are discussed for the three models, namely MLP, GNN, and GNN-PINN. The corresponding pressure, temperature, and velocity difference contours are shown in Figures 4.22, 4.23, and 4.24, respectively.

For the pressure field shown in Figure 4.22, it can be observed that the models have performed reasonably well for Cases 2, 4, and 5. Among the three models, the GNN gives the best overall predictions, while even the GNN-PINN performs fairly well despite the limitations associated with the reduced nodal point strategy used during training. For Case 2, only the large inlet velocity is extrapolated towards a smaller value, and therefore the momentum contribution from this inlet becomes comparatively weaker. Since the remaining parameters still remain within

the training range, the models are able to predict the pressure field reasonably well. Similarly, for Cases 4 and 5 only the inlet temperatures are extrapolated, which does not significantly influence the pressure field, and hence only very small deviations are observed. However, when moving towards Case 3, where the small inlet velocity is extrapolated to a much larger value, significant momentum changes occur within the mixing region, making the prediction considerably more difficult. Similar behaviour was also observed previously in the two dimensional study. Finally, for Case 6, where all input variables are extrapolated simultaneously, all three models fail to accurately predict the pressure field, resulting in large deviations throughout the domain.



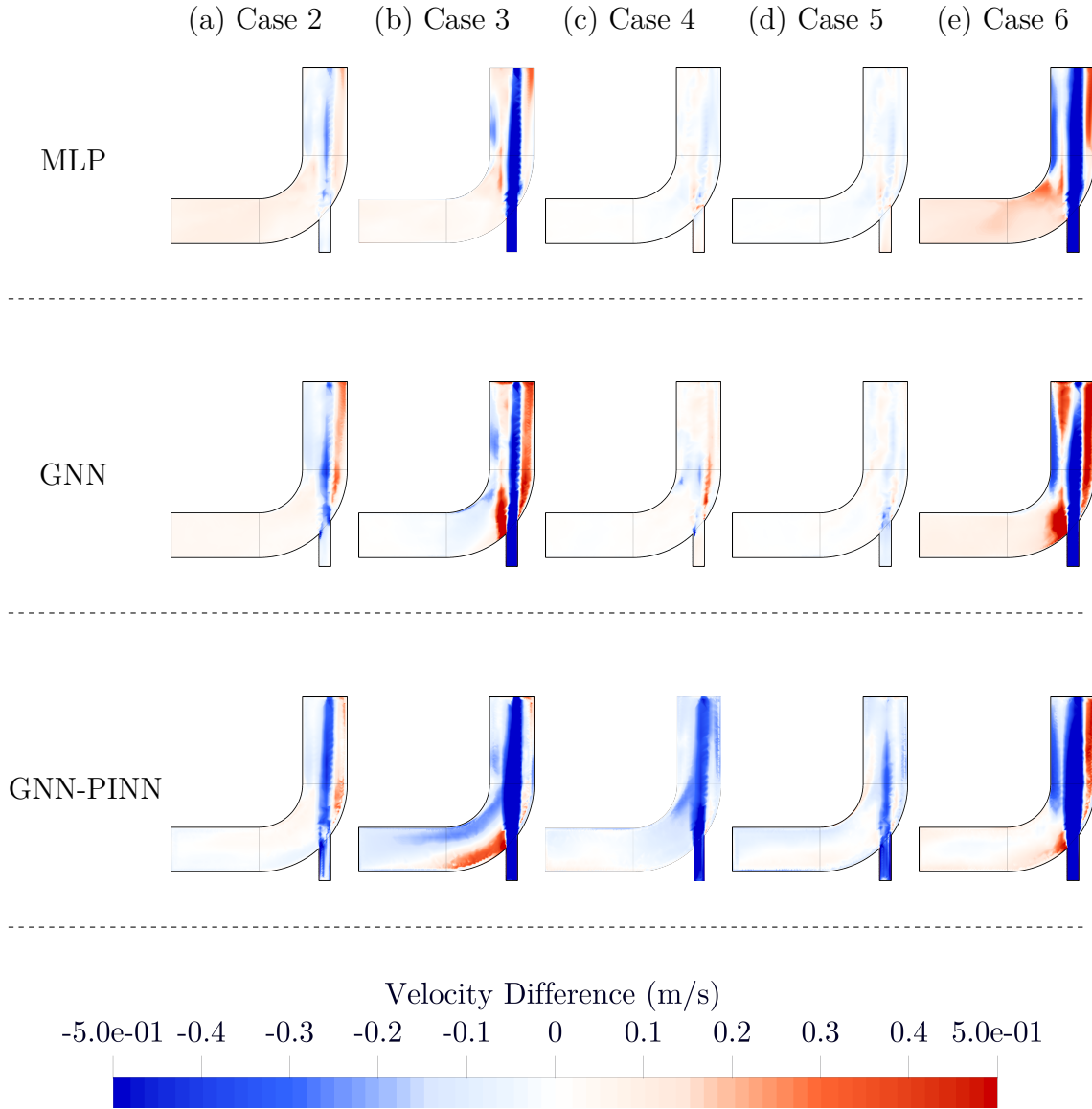
**Figure 4.22:** Comparison of pressure difference contours for the extrapolation test cases listed in Table 3.6 using MLP, GNN and GNN-PINN.



**Figure 4.23:** Comparison of Temperature difference contours for the extrapolation test cases listed in Table 3.6 using MLP, GNN and GNN-PINN.

For the temperature field shown in Figure 4.23, all three models show comparatively poor prediction capability across most extrapolation cases. Even for Cases 2 and 3, where only the inlet velocities are varied while the inlet temperatures remain close to the training range, noticeable deviations can still be observed. Similar to the observations from the two dimensional results, the presence of reverse flow and local recirculation regions strongly influences the thermal transport and mixing behaviour inside the elbow region, thereby making the temperature prediction significantly more challenging. For Cases 4 and 5, where the inlet temperatures themselves are extrapolated beyond the training range, large deviations are observed throughout the domain. This behaviour is mainly related to the way the models were trained. As discussed previously in the methodology section, the inlet boundary conditions were provided as global KPI parameters instead of being directly imposed at the

corresponding inlet nodal locations. Due to this approach, the models were required to simultaneously learn both the inlet thermal conditions and the resulting bulk thermal mixing behaviour, which made the prediction task considerably more difficult during extrapolation. Similar trends were also observed in the two dimensional case. Finally, for Case 6, where all variables are extrapolated simultaneously, the models completely fail to capture the temperature field accurately, resulting in very large deviations across the entire domain.



**Figure 4.24:** Comparison of Velocity magnitude difference contours for the extrapolation test cases listed in Table 3.6 using MLP, GNN and GNN-PINN.

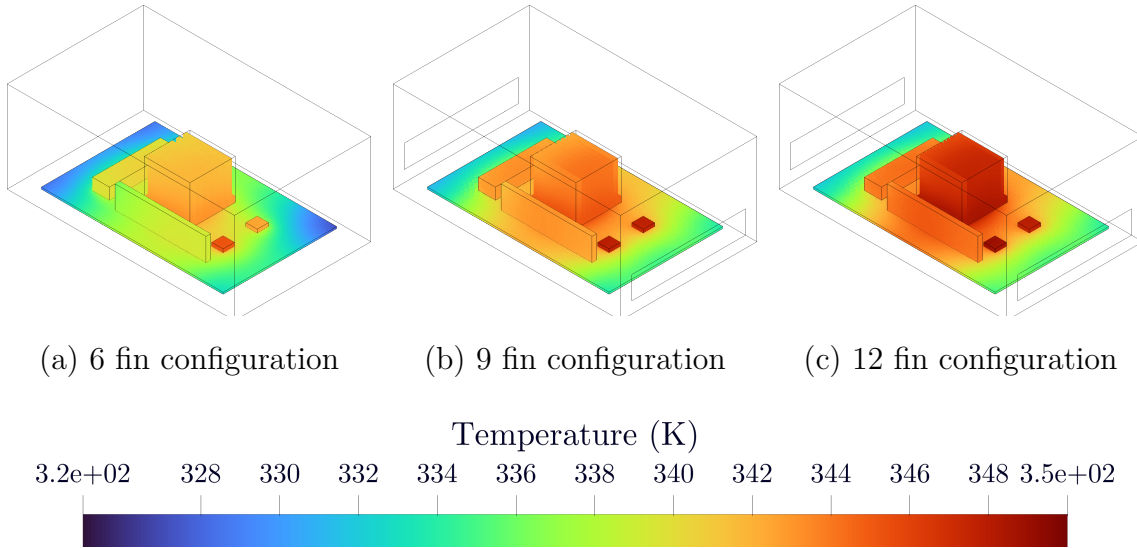
For the velocity field shown in Figure 4.24, the predictions can be considered reasonably acceptable, although not as accurate as the within range unseen cases. For Cases 2 and 3, the MLP and GNN models are able to capture the overall bulk flow reasonably well, but significant deviations can still be observed near the outlet

and mixing regions. Similar to the observations from the two dimensional study, the extrapolated inlet velocity conditions lead to reverse flow and local recirculation behaviour near the outlet region, which was not sufficiently represented within the training dataset. Due to this reason, the models struggle to accurately predict the local flow behaviour in these regions, resulting in larger errors near the outlet branch. For Cases 4 and 5, comparatively smaller deviations are observed since only the inlet temperatures are extrapolated, which has limited influence on the velocity field under the incompressible flow assumption adopted in the present work. However, for the GNN-PINN model, comparatively larger deviations are again observed due to the reduced nodal point strategy adopted during physics informed training for the three dimensional case. Finally, for Case 6, where all variables are extrapolated simultaneously, all three models show large deviations and fail to accurately capture the complex flow behaviour inside the elbow region. Nevertheless, among the three models, the GNN still provides the most stable and comparatively better extrapolation performance for the velocity field predictions.

## 4.2 CPU Cooling Case

This section presents the second study case considered in the present work, namely the CPU cooling problem involving conjugate heat transfer. Compared to the mixing elbow case discussed previously, the present case is considerably more complex since it involves both fluid and solid domains together with thermal interactions between multiple electronic components. The study includes the CPU, RAM modules, VRM section, and heat sink geometry inside the computational domain. For the present study, only the temperature field predictions are investigated using the NVIDIA PhysicsNeMo framework together with the MeshGraphNet based GNN architecture. Unlike the previous study, no Altair PhysicsAI workflow or physics informed training approach is considered for this case. The objective of this study is to evaluate the capability of graph based surrogate models for handling more complex industrial conjugate heat transfer problems involving geometric variations and multiple solid-fluid thermal interactions.

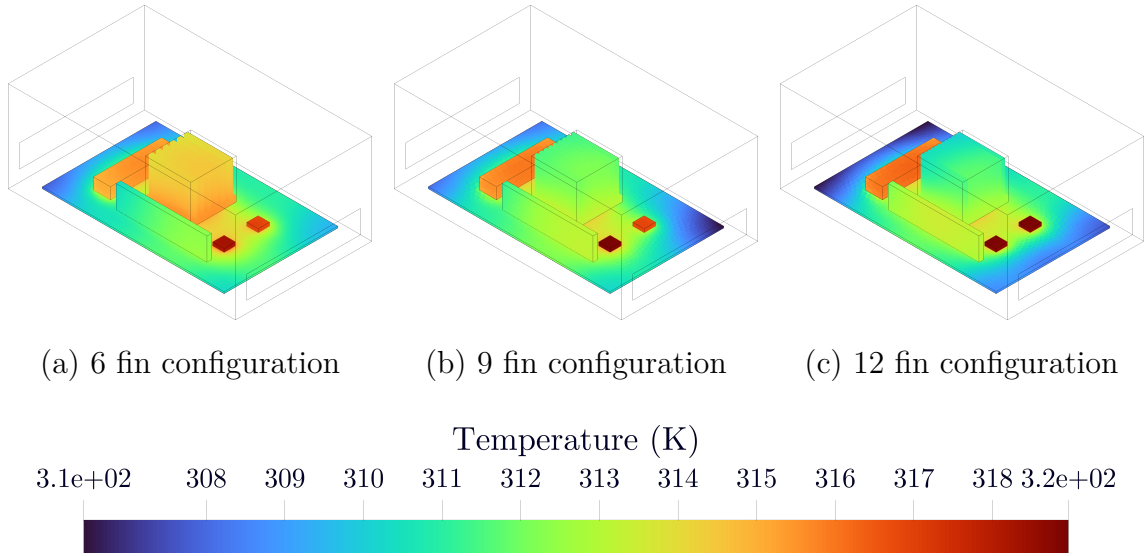
### 4.2.1 Overview of the CPU Cooling Study



**Figure 4.25:** CFD temperature contours for the CPU cooling case with different heat sink fin configurations at CPU, RAM, and VRM heat source values of 15 W, 1.5 W, and 5 W respectively with an inlet airflow rate of 5 CFM.

Figure 4.25 shows the CFD temperature contours obtained for the three different heat sink fin configurations considered in the present study, namely 6 fins, 9 fins, and 12 fins. All three cases correspond to the same operating condition with CPU, RAM, and VRM heat source values of 15 W, 1.5 W, and 5 W respectively, together with an inlet airflow rate of 5 CFM. From the contours, it can be observed that the 6 fin configuration results in comparatively lower temperatures around the CPU and heat sink region, while the 9 fin and especially the 12 fin configurations show higher temperatures. Although increasing the fin count increases the available heat transfer

surface area, the lower airflow rate of 5 CFM may not be sufficient to effectively pass air through the denser fin configurations. This can reduce the airflow between the fins and weaken the convective cooling performance, leading to higher temperatures for the 9 fin and 12 fin cases. Whereas, when observing Figure 4.26, it can be seen that increasing the inlet airflow rate to 15 CFM significantly improves the cooling performance of the 9 fin and 12 fin configurations. At the higher airflow rate, the incoming air is able to pass more effectively through the denser fin arrangements, improving convective heat transfer within the heat sink region. As a result, the 9 fin and 12 fin configurations show lower temperatures and a more uniform temperature distribution compared to the lower airflow case. This indicates that the cooling performance of higher fin count heat sinks strongly depends on having sufficient airflow to effectively utilize the increased heat transfer surface area.

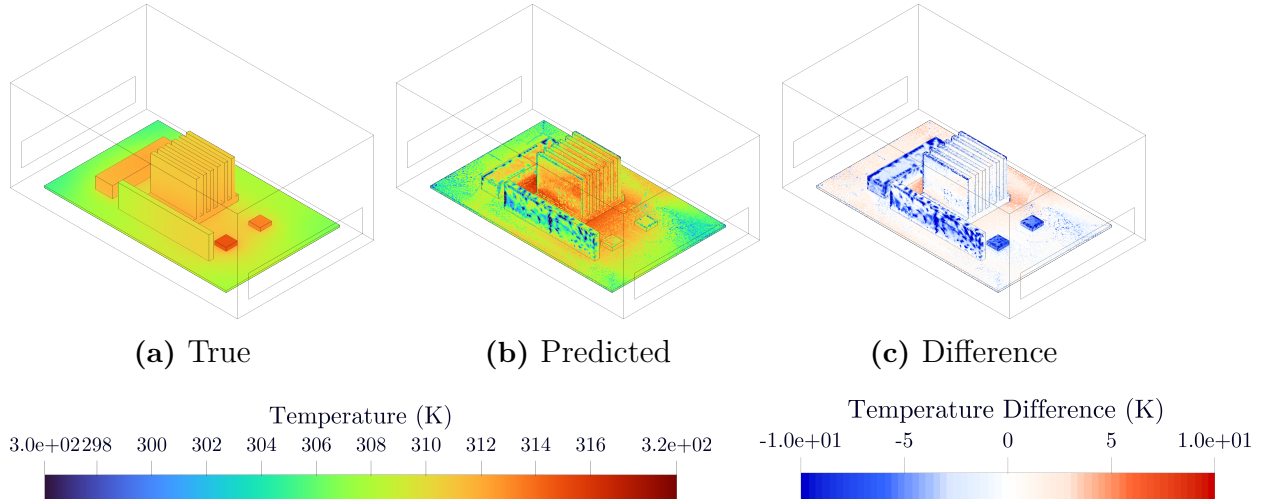


**Figure 4.26:** CFD temperature contours for the CPU cooling case with different heat sink fin configurations at CPU, RAM, and VRM heat source values of 15 W, 1.5 W, and 5 W respectively with an inlet airflow rate of 15 CFM.

### 4.2.2 MeshGraphNet Results

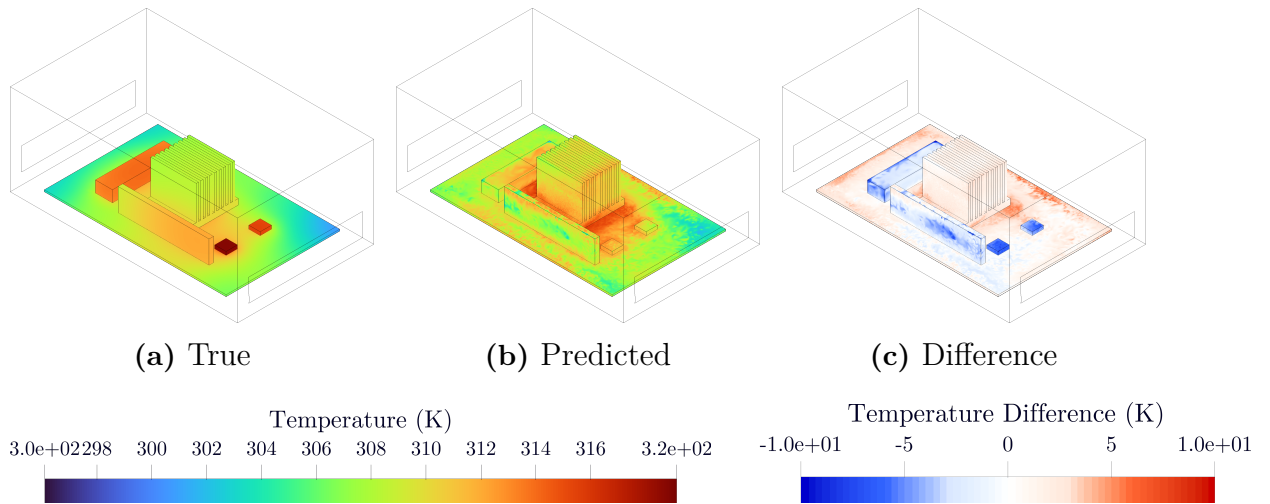
In this section, the prediction capability of the GNN based MeshGraphNet model for the CPU cooling case is discussed. Unlike the previous mixing elbow case, the present study contains both solid and fluid regions, making the learning process more difficult due to the coupled heat transfer at the solid fluid interfaces. For the current study, all the three fin configurations were used during training, and the model was evaluated on an unseen case with an inlet airflow rate of 15 CFM and CPU, RAM, and VRM heat source values of 15 W, 2 W, and 5 W respectively. The comparison between the CFD reference solution and the predicted temperature field is shown for the 6 fin, 9 fin, and 12 fin configurations.



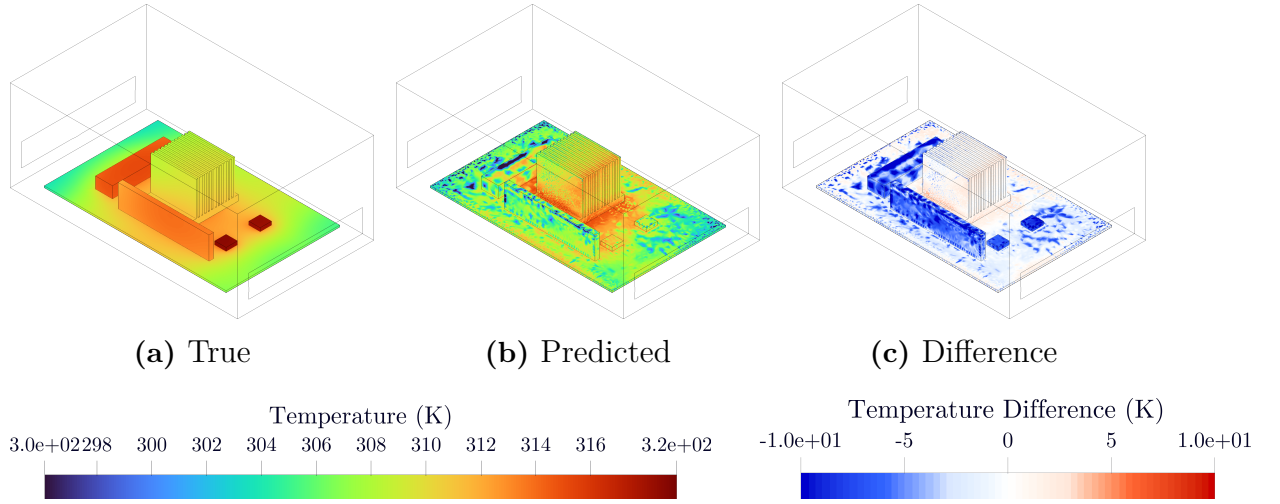


**Figure 4.27:** Comparison between the CFD reference solution and the MeshGraphNet prediction for the 6 fin CPU cooling configuration.

For the 6 fin, 9 fin, and 12 fin configurations shown in Figures 4.27, 4.28, and 4.29, the MeshGraphNet model is able to capture the overall bulk temperature distribution and the main heated regions reasonably well. The general thermal behaviour around the CPU heat sink and across the board is predicted fairly well for all three configurations. However, noticeable local deviations are still present near the walls, fins, and solid-fluid interface regions. The difference contours also show scattered errors across the domain, especially near the side walls and around the heat sink fins where the thermal interaction becomes more complex. Compared to the 6 fin and 9 fin cases, the 12 fin configuration shows comparatively larger deviations due to the tighter fin spacing and increased geometric complexity, which makes the learning process more difficult for the model.



**Figure 4.28:** Comparison between the CFD reference solution and the MeshGraphNet prediction for the 9 fin CPU cooling configuration.



**Figure 4.29:** Comparison between the CFD reference solution and the MeshGraphNet prediction for the 12 fin CPU cooling configuration.

The overall prediction quality for the CPU cooling case was lower compared to the previous mixing elbow cases mainly because of the increased complexity of the conjugate heat transfer problem. Unlike the earlier cases which involved only fluid regions, the present study contains both solid and fluid domains with thermal coupling occurring at the interfaces. In the current implementation, the model input mainly consisted of spatial coordinates, global KPI parameters, and a simple node type identifier distinguishing solid and fluid regions. As a result, the model had to learn the thermal interaction between solids and fluids mainly through local graph connectivity and neighbouring node information.

Another limitation comes from the graph construction itself, where the edge features only contained the distance between connected nodes. While this was sufficient for the simpler mixing elbow cases, it becomes less effective for the present CPU cooling problem where the temperature field depends on more complex thermal transport mechanisms through the fins, solid regions, and fluid passages. In addition, the local message passing approach used in the MeshGraphNet model limits the ability of the network to propagate thermal information across larger regions of the geometry, especially near the fin surfaces and solid-fluid interfaces where sharp temperature gradients are present.

Another challenge for the model was learning the different thermal behaviour associated with varying fin configurations under different airflow conditions. From the CFD results, it was observed that at lower airflow rates, the 6 fin configuration provided better cooling performance compared to the denser 9 fin and 12 fin configurations due to reduced flow resistance and improved airflow penetration between the fins. However, at higher airflow rates, the 9 fin and 12 fin configurations performed better because the increased airflow was sufficient to effectively utilize the larger heat transfer surface area. Capturing this changing thermal behaviour requires the model to learn complex nonlinear interactions between geometry, airflow rate, and conjugate heat transfer mechanisms, which further increases the difficulty of the

prediction task.

Furthermore, the present model architecture was kept relatively small with limited processor layers and hidden dimensions due to computational limitations. For such a geometrically complex three dimensional conjugate heat transfer problem, the network capacity was likely insufficient to fully capture the detailed thermal behaviour across the entire computational domain. Since the present work was mainly intended as an initial exploratory investigation using a limited number of training cases and fin configurations, the GNN model was able to reproduce the overall hotspot regions and global temperature trends reasonably well, but struggled to accurately capture the local thermal gradients and fine temperature variations near the fins, walls, and interface regions.



# 5

## Conclusion

### 5.1 Altair PhysicsAI

In the present work, Altair PhysicsAI was investigated as a commercial AI based surrogate modelling framework for accelerating CFD simulations involving fluid flow and thermal transport problems. The framework was evaluated using both two dimensional and three dimensional mixing elbow cases in order to assess its prediction capability, generalization behaviour, workflow compatibility, and practical usability for engineering CFD applications. For the two dimensional mixing elbow case, PhysicsAI was able to predict the interpolated unseen cases reasonably well. The overall velocity, pressure, and temperature field distributions were captured with acceptable accuracy for cases that remained within the training parameter range. However, for extrapolated cases outside the training distribution, the prediction quality reduced noticeably and the model performance became only moderate. Although the overall field trends were still captured to some extent, larger deviations and loss of accuracy were observed compared to the interpolation cases. This behaviour is consistent with the general limitations of data driven surrogate models, where prediction capability strongly depends on the similarity between training and testing conditions.

For the three dimensional mixing elbow case, the performance of PhysicsAI was considerably weaker and the framework was unable to accurately reproduce the CFD results. Significant deviations were observed in the predicted flow and thermal fields, and the model failed to capture the three dimensional flow structures and coupled thermal behaviour within the domain. During the investigation, several workflow related limitations were also identified. One major limitation was related to mesh compatibility and support for different cell topologies. Although PhysicsAI was capable of reading hexahedral meshes during the data curation stage, hexahedral cells were not properly supported during the prediction stage and often resulted in artifacts and corrupted predictions. Because of this limitation, the workflow became restricted mainly to tetrahedral meshes. Even with tetrahedral meshes, the prediction quality for the three dimensional cases was not satisfactory. In addition, it was also observed that native mesh formats generated directly from ANSYS Fluent and STAR-CCM+ were not fully supported for the complete workflow. As a result, tetrahedral meshes generated using Altair HyperMesh had to be used in order

to maintain compatibility with the PhysicsAI workflow. However, this introduced an additional meshing workflow separate from the original CFD workflow adopted in the study. In practical industrial applications, the mesh generated in HyperMesh may not necessarily match the mesh quality or meshing strategy preferred in STAR-CCM+ or ANSYS Fluent simulations. Therefore, at the current stage of development, the workflow compatibility of PhysicsAI remains somewhat limited for the type of industrial CFD workflow investigated in the present research.

Overall, the present investigation showed that Altair PhysicsAI still appears to be in a developing stage for complex CFD surrogate modelling applications involving three dimensional flow and thermal problems. While the framework demonstrated reasonable capability for simpler two dimensional interpolation cases, the performance reduced significantly for extrapolation and more complex three dimensional configurations. Nevertheless, the objective of the present work was not only to evaluate the final prediction accuracy, but also to investigate the practical applicability, workflow limitations, and current maturity of commercial AI based CFD acceleration tools for industrial engineering problems. In this regard, the study provides useful insight into the present capabilities and limitations of PhysicsAI for CFD surrogate modelling applications.

### 5.2 NVIDIA PhysicsNeMo

The NVIDIA PhysicsNeMo framework was investigated in the present work as an open source alternative for AI based CFD surrogate modelling using different neural network architectures including Multi Layer Perceptrons (MLPs), MeshGraphNet based Graph Neural Networks (GNNs), Physics Informed GNNs, and Fourier Neural Operators (FNOs). Compared to the commercial PhysicsAI workflow, PhysicsNeMo provided greater flexibility and control over the complete surrogate modelling pipeline, including dataset preparation, preprocessing, network architecture development, training configuration, and postprocessing. Since the framework is fully open source, the present work allowed direct control over how the AI models were structured and trained, which made it easier to investigate different modelling approaches and analyse their behaviour for CFD applications. However, this flexibility also introduces certain practical challenges. Unlike commercial workflows where much of the model configuration and optimization process is automated, PhysicsNeMo requires the user to manually select architectures, preprocessing strategies, training parameters, and workflow configurations depending on the specific CFD problem being investigated. As a result, the overall workflow can become more time consuming and requires a stronger understanding of both CFD and machine learning concepts. In addition, developing a generalized and robust workflow applicable across different industrial CFD problems can be considerably more challenging compared to commercial AI platforms that are designed to provide more streamlined and user friendly implementations. Overall, the investigated PhysicsNeMo workflows showed promising performance for the present CFD surrogate modelling cases considered in this study.

For the two dimensional mixing elbow case, the GNN and MLP based models performed particularly well and produced highly accurate predictions for pressure, velocity, and temperature fields for the interpolated unseen cases. The predicted field distributions closely matched the original CFD results and the models were able to capture both the global flow and local thermal behaviour reasonably well. The Fourier Neural Operator also showed very strong performance for pressure and velocity predictions. However, the temperature predictions using the FNO model were comparatively weaker. This behaviour is likely related to the structured grid conversion required for the FNO workflow, where interpolation from unstructured CFD meshes to structured grids introduced some loss of information. Since temperature gradients are generally more sensitive to local interpolation errors, the structured grid resolution becomes highly important for achieving accurate thermal predictions using FNO based approaches. Nevertheless, the FNO model still demonstrated very good prediction capability for pressure and velocity fields, especially for the unseen interpolation cases. The Physics Informed GNN model also performed reasonably well, although the accuracy was generally lower compared to the standard GNN and MLP architectures.

For the extrapolated two dimensional cases, the GNN and MLP models again showed comparatively better performance than the other investigated architectures. Although none of the models achieved the same level of accuracy observed for interpolation cases, the overall field distributions and major flow features were still captured reasonably well even outside the training parameter range. This indicates that the models retained some degree of generalization capability despite the increased difficulty associated with extrapolation problems.

For the three dimensional mixing elbow case, PhysicsNeMo again demonstrated good performance. Both the GNN and MLP based models produced good agreement with the CFD results for the interpolated unseen cases, and both architectures showed fairly similar prediction quality for the velocity, pressure, and temperature fields. The models were able to capture the major three dimensional flow structures and thermal mixing behaviour reasonably well within the computational domain. However, the Physics Informed GNN model did not perform well for the present three dimensional study. One of the primary reasons for this limitation was related to the reduced number of nodal points used during the physics informed training process due to computational and memory limitations. Because the PINN based workflow could not use the complete mesh resolution of the original CFD simulations, the resulting prediction quality was reduced compared to the standard graph based models.

The final part of the study investigated the CPU cooling case involving conjugate heat transfer using the MeshGraphNet architecture within PhysicsNeMo. Compared to the mixing elbow case, this problem was significantly more complex since it involved coupled solid and fluid domains together with thermal interaction across solid fluid interfaces. The results showed that the MeshGraphNet model was able to predict the overall temperature distribution and few hotspot regions reasonably well for the different heat sink configurations. The general thermal behaviour inside

the computational domain was captured fairly well, especially around the CPU and heat sink regions. However, noticeable local deviations were still present near the fins, walls, and solid-fluid interface region. The increasing geometric complexity associated with the higher fin count configurations also resulted in larger deviations. In addition, the model also had to learn the changing cooling behaviour associated with different airflow rates, where lower fin count configurations performed better at low airflow conditions while higher fin count configurations showed improved cooling performance at higher airflow rates. Nevertheless, the present CPU cooling investigation serves as a useful initial study for applying graph based AI surrogate modelling approaches to industrially relevant conjugate heat transfer problems involving coupled solid-fluid thermal behaviour.

Overall, the present study demonstrated that open source AI frameworks such as NVIDIA PhysicsNeMo currently provide greater flexibility, transparency, and better prediction capability for CFD surrogate modelling applications compared to the investigated commercial workflow. Among the investigated architectures, the Mesh-GraphNet based GNN and MLP models generally produced the most stable and accurate results for both interpolation and extrapolation cases. The study also highlighted that model performance strongly depends on the complexity of the physical problem, mesh representation, data preparation workflow, and the suitability of the selected neural network architecture for the underlying physics. Although several limitations still remain, especially for highly complex three dimensional conjugate heat transfer problems, the present work demonstrates that AI based surrogate modelling frameworks show strong potential for accelerating future CFD workflows and industrial engineering simulations.



# Bibliography

- [1] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, “Machine learning for fluid mechanics,” *Annual Review of Fluid Mechanics*, vol. 52, pp. 477–508, 2020.
- [2] K. Duraisamy, G. Iaccarino, and H. Xiao, “Turbulence modeling in the age of data,” *Annual Review of Fluid Mechanics*, vol. 51, pp. 357–377, 2019.
- [3] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.
- [4] ANSYS Inc. (2023) Simai: Ai-powered simulation. Accessed: 2026-01-29. [Online]. Available: <https://www.ansys.com/products/simulation/simai>
- [5] Altair Engineering Inc. (2023) Altair physicsai. Accessed: 2026-01-29. [Online]. Available: <https://altair.com/physicsai>
- [6] K. Rojek, R. Wyrzykowski, and P. Gepner, “Ai-accelerated cfd simulation based on openfoam and cpu/gpu computing,” in *International Conference on Computational Science*. Springer, 2021, pp. 373–385.
- [7] S. Adya, A. Jawale, and P. Yser, “Accelerate pedestrian protection assessments using simai,” in *International LS-DYNA Conference*. ANSYS Inc., 2024.
- [8] C. Karadogan, S. S. Vasu, S. Adya *et al.*, “Reimagining crash simulations with ai: Transforming legacy data into scalable surrogate models,” *Engineering with Computers*, 2024.
- [9] M. Yagoubi, D. Danan, M. Leyli-Abadi *et al.*, “Ml4cfd competition: Harnessing machine learning for computational fluid dynamics,” *arXiv preprint*, 2024.
- [10] A. Patil and P. Sonavane, “Ai-enhanced cae simulations: A revolutionary approach to automotive design,” *SAE Technical Paper*, no. 2025-01-8241, 2025.
- [11] R. R. Mahale and V. Kumar, “Ai-driven structural optimization of uav wings through a multi-physics approach,” in *ASME IMECE India*, no. IMECE-INDIA2025-160637, 2025.
- [12] H. Rewatkar and V. Kumar, “Fixed wing uav winglet optimization through

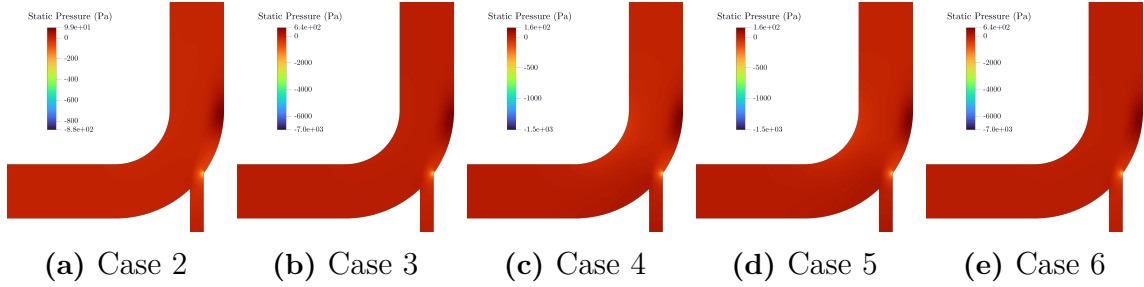
- ai-integrated aerodynamic simulations,” in *ASME IMECE India*, no. IMECE-INDIA2025-160832, 2025.
- [13] K. Tangsali, R. Ranade, M. A. Nabian, A. Kamenev, P. Sharpe, N. Ashton, R. Cherukuri, and S. Choudhry, “A benchmarking framework for ai models in automotive aerodynamics,” *arXiv preprint arXiv:2507.10747*, 2025.
  - [14] C. Adams, R. Ranade, R. Cherukuri, and S. Choudhry, “Geotransolver: Learning physics on irregular domains using multi-scale geometry aware physics attention transformer,” *arXiv preprint arXiv:2512.20399*, 2025.
  - [15] M. Elrefaie, D. Shu, M. Klenk, and F. Ahmed, “Carbench: A comprehensive benchmark for neural surrogates on high-fidelity 3d car aerodynamics,” *arXiv preprint arXiv:2512.07847*, 2025.
  - [16] L. Davidson, “Fluid mechanics, turbulent flow and turbulence modeling,” 2026, course material for TME226 and MTF271. [Online]. Available: <https://www.tfd.chalmers.se/~lada>
  - [17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
  - [18] Altair Engineering, “Altair physicsai documentation and best practices,” 2024. [Online]. Available: [https://help.altair.com/hwdesktop/hwx/topics/reference/extensions/physicsai\\_best\\_practices\\_r.htm](https://help.altair.com/hwdesktop/hwx/topics/reference/extensions/physicsai_best_practices_r.htm)
  - [19] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, “Fourier neural operator for parametric partial differential equations,” in *International Conference on Learning Representations (ICLR)*, 2021.
  - [20] NVIDIA, “Fully connected network - nvidia physicsnemo documentation,” [https://docs.nvidia.com/physicsnemo/latest/physicsnemo/api/models/fully\\_connected.html](https://docs.nvidia.com/physicsnemo/latest/physicsnemo/api/models/fully_connected.html), 2025.
  - [21] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, “Learning mesh-based simulation with graph networks,” in *International Conference on Learning Representations (ICLR)*, 2021.
  - [22] 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*, vol. 378, pp. 686–707, 2019.
  - [23] NVIDIA, “Physics informed neural networks in physicsnemo sym,” [https://docs.nvidia.com/physicsnemo/latest/physicsnemo-sym/user\\_guide/theory/phys\\_informed.html](https://docs.nvidia.com/physicsnemo/latest/physicsnemo-sym/user_guide/theory/phys_informed.html), 2025.

# A

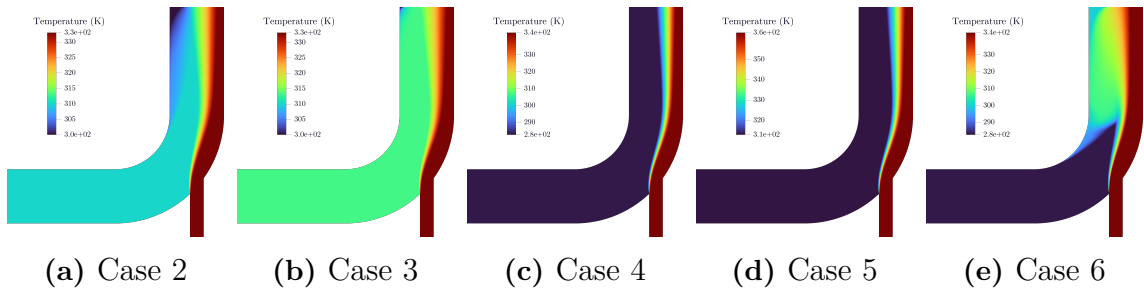
## Appendix

### A.1 Two Dimensional Mixing Elbow: Extrapolation Cases

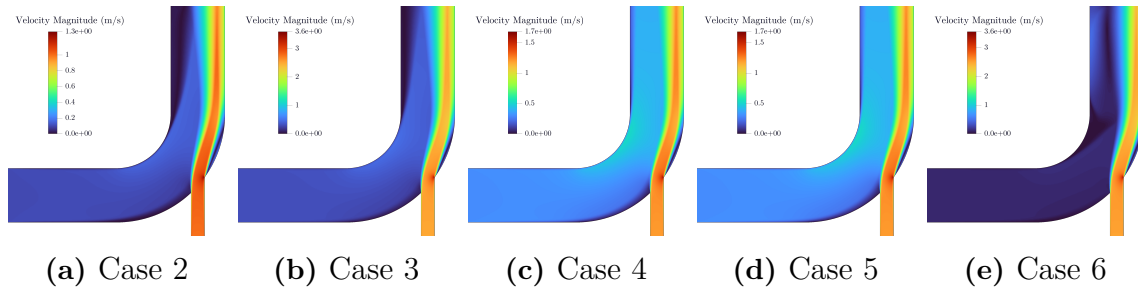
This section presents the CFD reference solutions for the two dimensional extrapolation test cases considered in the present study. The figures include the pressure, temperature, and velocity magnitude contours corresponding to the extrapolated operating conditions listed in Table 3.6. These cases were used to evaluate the generalization capability of the AI based surrogate models beyond the training parameter range.



**Figure A.1:** CFD pressure contours for the two dimensional extrapolation test cases listed in Table 3.6.



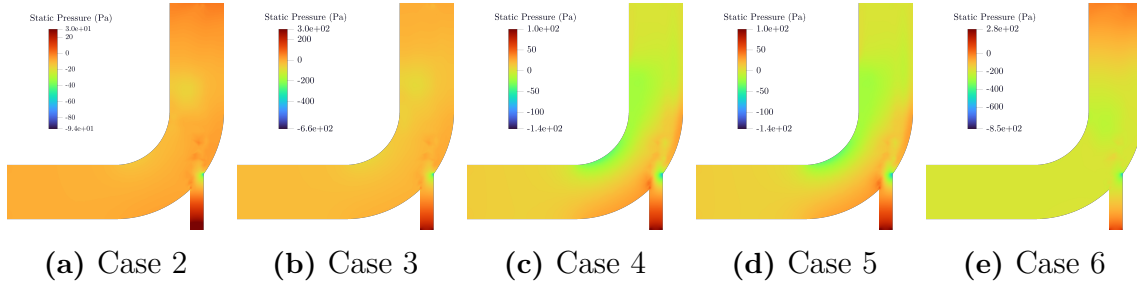
**Figure A.2:** CFD temperature contours for the two dimensional extrapolation test cases listed in Table 3.6.



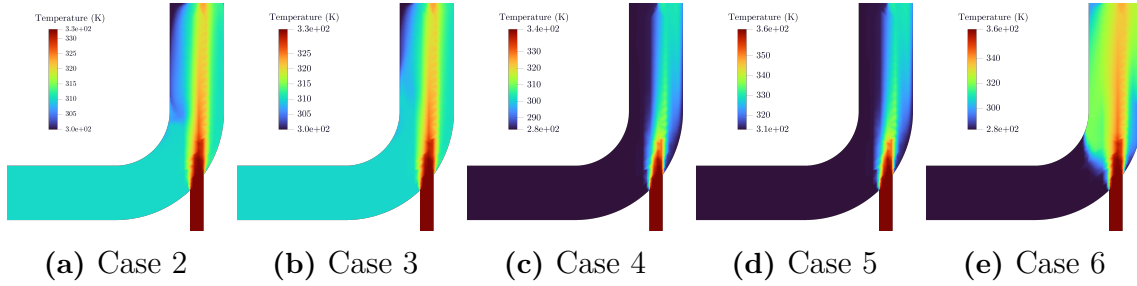
**Figure A.3:** CFD velocity magnitude contours for the two dimensional extrapolation test cases listed in Table 3.6.

## A.2 Three Dimensional Mixing Elbow: Extrapolation Cases

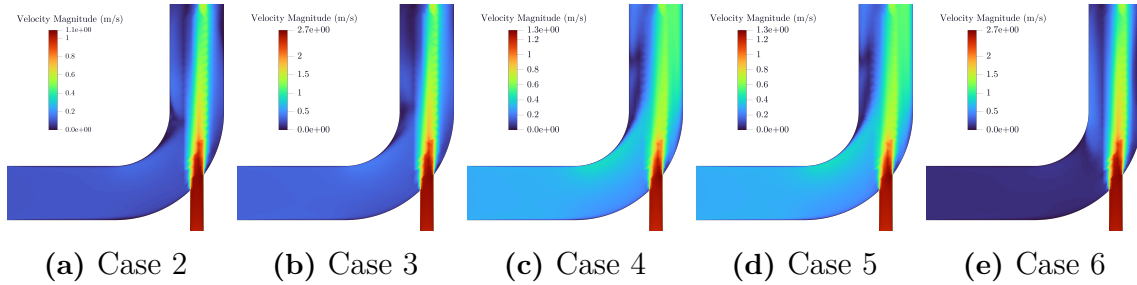
This section presents the CFD reference solutions for the three dimensional extrapolation test cases used in the present work. The pressure, temperature, and velocity magnitude contours shown below correspond to the extrapolated flow conditions used for evaluating the performance and generalization capability of the investigated AI based surrogate models for the three dimensional mixing elbow geometry.



**Figure A.4:** CFD pressure contours for the three dimensional extrapolation test cases listed in Table 3.6.



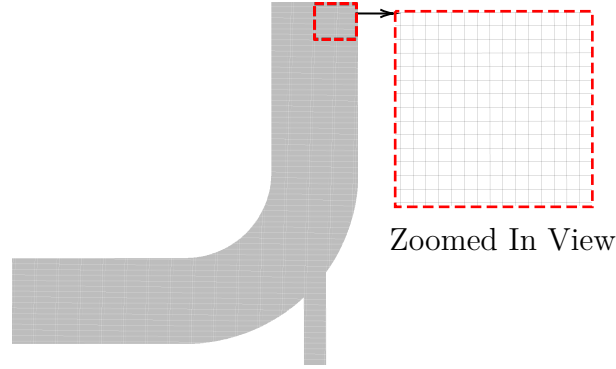
**Figure A.5:** CFD temperature contours for the three dimensional extrapolation test cases listed in Table 3.6.



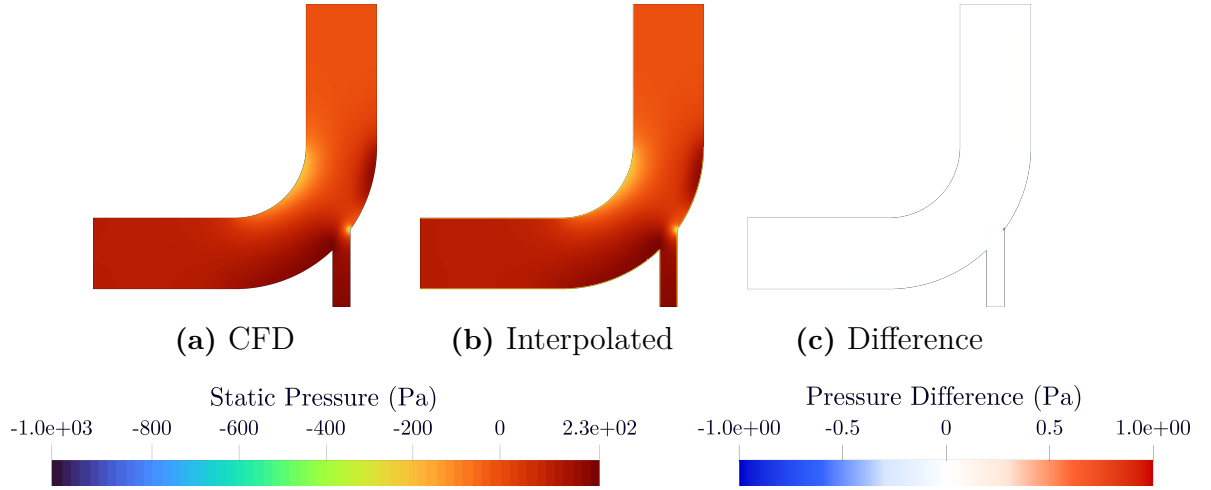
**Figure A.6:** CFD velocity magnitude contours for the three dimensional extrapolation test cases listed in Table 3.6.

### A.3 Structured Grid Conversion for FNO Workflow

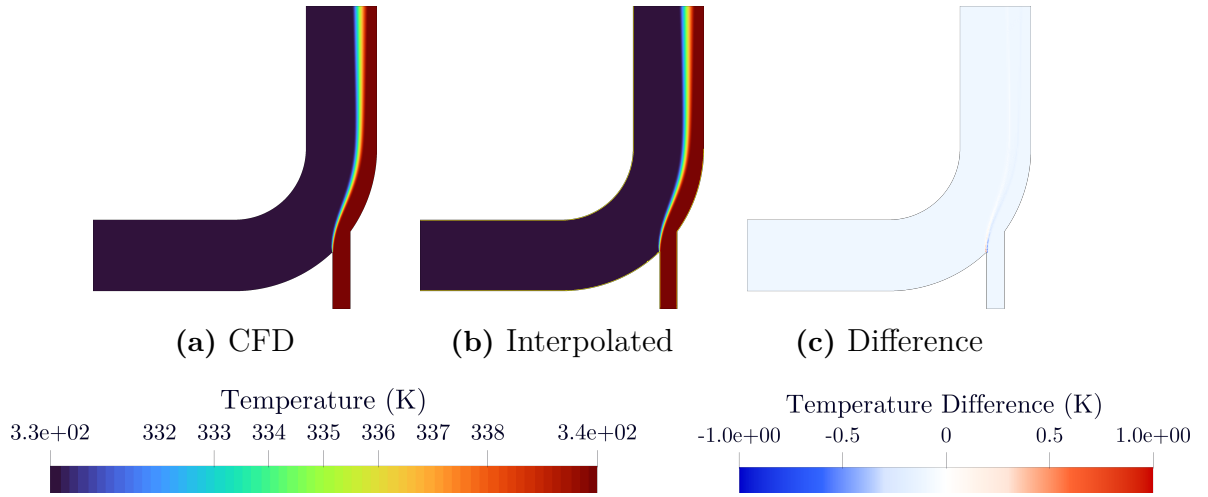
The Fourier Neural Operator (FNO) workflow investigated in the present study requires structured grid based input data. Since the original CFD simulations were performed on unstructured meshes, the solution fields were interpolated onto structured grids before training and prediction. Figure A.7 illustrates the structured grid representation obtained after the interpolation process used for the FNO workflow. Figure A.8, A.9 and A.10 represents the contour filed after interpolation.



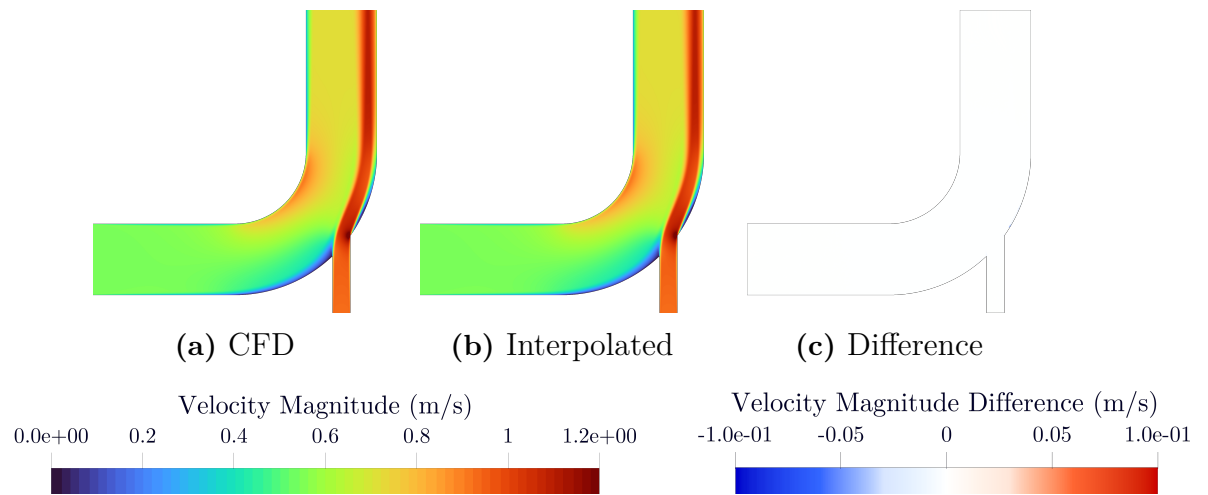
**Figure A.7:** Example of structured grid representation obtained from the original unstructured CFD data for the FNO based workflow.



**Figure A.8:** Comparison between the CFD reference solution and the FNO interpolated results on structured grid for the pressure field.



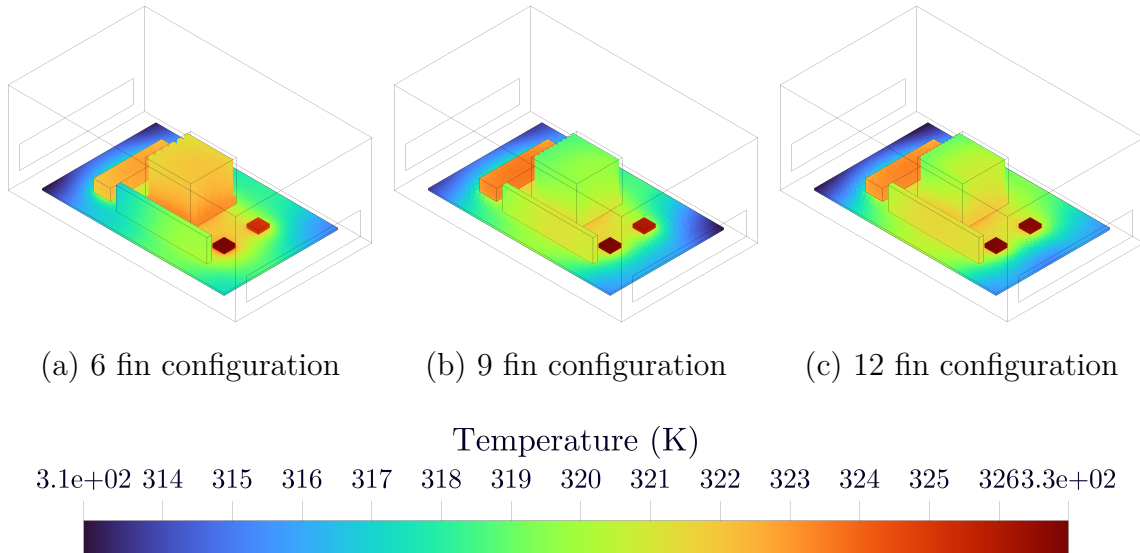
**Figure A.9:** Comparison between the CFD reference solution and the FNO interpolated results on structured grid for the temperature field.



**Figure A.10:** Comparison between the CFD reference solution and the FNO interpolated results on structured grid for the velocity magnitude field.

## A.4 CPU Cooling Case

Figure A.11 shows the CFD temperature contours obtained for the three different heat sink fin configurations considered in the present study, namely 6 fins, 9 fins, and 12 fins. All three cases correspond to the same operating condition with CPU, RAM, and VRM heat source values of 15 W, 1.5 W, and 5 W respectively, together with an inlet airflow rate of 10 CFM. From the contours, it can be observed that the 6 fin configuration results in comparatively higher temperatures around the CPU and heat sink region, while the 9 fin and 12 fin configurations show lower temperatures, with the 9 fin configuration providing the lowest temperatures among the three cases. Compared to the lower airflow case shown in Figure 4.25, the increased airflow rate of 10 CFM provides improved airflow penetration through the fin passages, allowing the 9 fin configuration to achieve more effective convective cooling. However, the airflow is still not fully sufficient to effectively utilize the denser 12 fin configuration, resulting in slightly higher temperatures compared to the 9 fin case. Nevertheless, the 12 fin configuration still performs better than the 6 fin configuration due to the increased available heat transfer surface area.



**Figure A.11:** CFD temperature contours for the CPU cooling case with different heat sink fin configurations at CPU, RAM, and VRM heat source values of 15 W, 1.5 W, and 5 W respectively with an inlet airflow rate of 10 CFM.



## A.5 Python Scripts Used in NVIDIA PhysicsNeMo Workflow

This section presents the primary Python scripts developed and used in the present work for dataset preparation, training, and prediction using the NVIDIA PhysicsNeMo framework.

**Listing A.1:** Generation of DOE samples using Latin Hypercube Sampling.

```

1  ###
2
3  import numpy as np
4  from pyDOE2 import lhs
5  import pandas as pd
6
7  # Number of DOE samples
8  n_samples = 50
9
10 # Variable ranges
11 ranges = {
12     "U_large": (0.2, 0.6),
13     "U_small": (0.6, 2.0),
14     "T_large": (293.15, 333.15),
15     "T_small": (303.15, 353.15)
16 }
17 # LHS sampling
18 lhs_unit = lhs(len(ranges), samples=n_samples)
19
20 # Scale samples
21 data = np.zeros_like(lhs_unit)
22 for i, (low, high) in enumerate(ranges.values()):
23     data[:, i] = low + lhs_unit[:, i] * (high - low)
24
25 df = pd.DataFrame(data, columns=ranges.keys())
26
27 df.to_csv("DOE_cases_2.csv", index=False)
28 print('done')

```

### A.5.1 MLP Based Surrogate Model Workflow

The MLP based surrogate modelling workflow developed using NVIDIA PhysicsNeMo consists of three main stages, namely dataset preparation, model training, and prediction with postprocessing. The primary Python scripts used for these stages are listed below.

**Listing A.2:** Example syntax of the MLP dataset preparation used in NVIDIA PhysicsNeMo..

```
1 label={lst:main_data_mlp}
2
3 import pyvista as pv
4 import numpy as np
5
6 # -----
7 # Read CFD case
8 # -----
9 mesh = pv.read(case_file)
10 coords = mesh.points[:, :2]
11
12 # -----
13 # Extract CFD fields
14 # -----
15
16 pressure = mesh.point_data["Pressure"]
17 temperature = mesh.point_data["Temperature"]
18 u = mesh.point_data["Velocityi"]
19 v = mesh.point_data["Velocityj"]
20
21 # -----
22 # Build input features
23 # -----
24 input_array = np.tile(
25     [U_large, U_small, T_large, T_small],
26     (coords.shape[0], 1)
27 )
28 X = np.hstack([coords, input_array])
29
30 # -----
31 # Build output targets
32 # -----
33 Y = np.column_stack([
34     pressure,
35     u,
36     v,
37     temperature
38 ])
39
40 # -----
41 # Save processed dataset
42 # -----
43 np.savez_compressed(
44     "case_data.npz",
45     X=X,
46     Y=Y
```

47 )

**Listing A.3:** Example syntax of the MLP training workflow used in NVIDIA PhysicsNeMo..

```

1 label={lst:main_train_mlp}
2
3 import torch
4 from torch.utils.data import DataLoader
5 from physicsnemo.models.mlp import FullyConnected
6
7 # -----
8 # Load processed CFD dataset
9 # -----
10 dataset = CFDDataset("data/processed")
11 loader = DataLoader(
12     dataset,
13     batch_size=100000,
14     shuffle=True
15 )
16
17 # -----
18 # Define MLP model
19 # -----
20 model = FullyConnected(
21     in_features=6,
22     out_features=4,
23     layer_size=512,
24     num_layers=6,
25     activation_fn="tanh"
26 ).to(device)
27
28 # -----
29 # Training setup
30 # -----
31 optimizer = torch.optim.Adam(
32     model.parameters(),
33     lr=1e-3
34 )
35
36 loss_fn = torch.nn.MSELoss()
37 # -----
38 # Training loop
39 # -----
40 for epoch in range(epochs):
41     for x, y in loader:
42         pred = model(x)
43         loss = loss_fn(pred, y)
44         optimizer.zero_grad()
45         loss.backward()
46         optimizer.step()
47 # -----
48 # Save trained model
49 # -----
50 torch.save(

```

```

51     model.state_dict(),
52     "surrogate_model_nemo.pth"
53 )

```

**Listing A.4:** Example syntax of the MLP Prediction workflow used in NVIDIA PhysicsNeMo.

```

1  label={lst:main_pred_mlp}
2
3  import torch
4  import pyvista as pv
5  import numpy as np
6
7  # -----
8  # Load trained model
9  # -----
10 model.load_state_dict(
11     torch.load("surrogate_model_nemo.pth")
12 )
13 model.eval()
14
15 # -----
16 # Build prediction input
17 # -----
18 X = np.hstack([
19     coords,
20     boundary_conditions
21 ])
22 X_tensor = torch.tensor(
23     X,
24     dtype=torch.float32
25 ).to(device)
26
27 # -----
28 # Predict CFD fields
29 # -----
30 with torch.no_grad():
31     pred = model(X_tensor)
32
33 # -----
34 # Extract predicted variables
35 # -----
36 pressure = pred[:, 0]
37 u = pred[:, 1]
38 v = pred[:, 2]
39 temperature = pred[:, 3]
40
41 # -----
42 # Save prediction as VTK
43 # -----
44 mesh = pv.PolyData(points)
45 mesh["Pressure"] = pressure
46 mesh["Temperature"] = temperature
47 mesh.save("prediction.vtk")

```

### A.5.2 GNN Model Workflow

The GNN workflow developed using NVIDIA PhysicsNeMo consists of three main stages, namely dataset preparation, model training, and prediction with postprocessing. The primary Python scripts used for these stages are listed below.

**Listing A.5:** Example implementation syntax of the GNN dataset preparation workflow used in NVIDIA PhysicsNeMo.

```

1  import pyvista as pv
2  import numpy as np
3
4
5  # -----
6  # Read CFD mesh
7  # -----
8  mesh = pv.read(case_file)
9  coords = mesh.points[:, :2]
10
11 # -----
12 # Build graph connectivity
13 # -----
14 edge_index = build_graph(mesh)
15 edge_attr = compute_edge_distance(
16     coords,
17     edge_index
18 )
19
20 # -----
21 # Extract CFD fields
22 # -----
23 pressure = mesh.point_data["Pressure"]
24 temperature = mesh.point_data["Temperature"]
25 u = mesh.point_data["Velocityi"]
26 v = mesh.point_data["Velocityj"]
27
28 # -----
29 # Build node features
30 # -----
31 input_array = np.tile(
32     [U_large, U_small, T_large, T_small],
33     (coords.shape[0], 1)
34 )
35 X = np.hstack([coords, input_array])
36
37 # -----
38 # Build output targets
39 # -----
40 Y = np.column_stack([
41     pressure,
42     u,
43     v,
44     temperature
45 ])
46
47 # -----

```

```

48 # Save graph dataset
49 # -----
50 np.savez_compressed(
51     "case_data.npz",
52     X=X,
53     Y=Y,
54     edge_index=edge_index,
55     edge_attr=edge_attr
56 )

```

**Listing A.6:** Example implementation syntax of the GNN training workflow used in NVIDIA PhysicsNeMo.

```

1
2 import torch
3 import dgl
4 from physicsnemo.models.meshgraphnet import MeshGraphNet
5
6 # -----
7 # Load graph dataset
8 # -----
9 dataset = CFDDataset("data_gnn/processed")
10 loader = DataLoader(
11     dataset,
12     batch_size=1,
13     shuffle=True
14 )
15
16 # -----
17 # Define MeshGraphNet model
18 # -----
19 model = MeshGraphNet(
20     input_dim_nodes=6,
21     input_dim_edges=1,
22     output_dim=4,
23     processor_size=5
24 ).to(device)
25
26 # -----
27 # Training setup
28 # -----
29 optimizer = torch.optim.Adam(
30     model.parameters(),
31     lr=1e-3
32 )
33
34 loss_fn = torch.nn.MSELoss()
35
36 # -----
37 # Training loop
38 # -----
39
40 for epoch in range(epochs):
41     for X, Y, edge_index, edge_attr in loader:
42         graph = build_dgl_graph(edge_index,

```

```

43         X.shape[0]
44     )
45     pred = model(X,
46                 edge_attr,
47                 graph
48     )
49     loss = loss_fn(pred, Y)
50     optimizer.zero_grad()
51     loss.backward()
52     optimizer.step()
53
54     # -----
55     # Save trained model
56     # -----
57     torch.save(
58         model.state_dict(),
59         "gnn_meshgraphnet.pth"
60     )

```

**Listing A.7:** Example implementation syntax of the GNN prediction workflow used in NVIDIA PhysicsNeMo.

```

1
2 import torch
3 import pyvista as pv
4 import dgl
5
6 from physicsnemo.models.meshgraphnet import MeshGraphNet
7
8 # -----
9 # Load trained model
10 # -----
11
12 model.load_state_dict(
13     torch.load("gnn_meshgraphnet.pth")
14 )
15
16 model.eval()
17
18 # -----
19 # Build graph input
20 # -----
21
22 X = np.hstack([
23     coords,
24     boundary_conditions
25 ])
26
27 graph = build_dgl_graph(
28     edge_index,
29     num_nodes
30 )
31
32 # -----
33 # Predict CFD fields

```

```
34 # -----
35
36 with torch.no_grad():
37
38     pred = model(
39         X_tensor,
40         edge_attr,
41         graph
42     )
43
44 # -----
45 # Extract predictions
46 # -----
47 pressure = pred[:, 0]
48 u = pred[:, 1]
49 v = pred[:, 2]
50 temperature = pred[:, 3]
51
52 # -----
53 # Save prediction as VTK
54 # -----
55 mesh = pv.UnstructuredGrid(
56     cells,
57     celltypes,
58     points
59 )
60
61 mesh["Pressure"] = pressure
62 mesh["Temperature"] = temperature
63 mesh.save("gnn_prediction.vtk")
```



### A.5.3 GNN PINN Model Workflow

The GNN-PINN workflow developed using NVIDIA PhysicsNeMo consists of three main stages, namely dataset preparation, model training, and prediction with post-processing. The primary Python scripts used for these stages are listed below.

**Listing A.8:** Example implementation syntax of the GNN-PINN dataset preparation workflow used in NVIDIA PhysicsNeMo.

```

1 import pyvista as pv
2 import numpy as np
3 # -----
4 # Read CFD mesh
5 # -----
6 mesh = pv.read(case_file)
7 coords = mesh.points[:, :2]
8
9 # -----
10 # Build graph connectivity
11 # -----
12 edge_index = build_graph(mesh)
13 edge_attr = compute_edge_distance(
14     coords,
15     edge_index
16 )
17
18 # -----
19 # Extract CFD fields
20 # -----
21 pressure = mesh.point_data["Pressure"]
22 temperature = mesh.point_data["Temperature"]
23 u = mesh.point_data["Velocityi"]
24 v = mesh.point_data["Velocityj"]
25
26 # -----
27 # Build node features
28 # -----
29 input_array = np.tile(
30     [U_large, U_small, T_large, T_small],
31     (coords.shape[0], 1)
32 )
33 X = np.hstack([coords, input_array])
34
35 # -----
36 # Build output targets
37 # -----
38 Y = np.column_stack([
39     pressure,
40     u,
41     v,
42     temperature
43 ])
44
45 # -----
46 # Save graph dataset
47 # -----

```

```

48 np.savez_compressed(
49     "case_data.npz",
50     X=X,
51     Y=Y,
52     edge_index=edge_index,
53     edge_attr=edge_attr
54 )

```

**Listing A.9:** Example implementation syntax of the GNN-PINN training workflow used in NVIDIA PhysicsNeMo.

```

1
2 import torch
3 import dgl
4 from physicsnemo.models.meshgraphnet import MeshGraphNet
5 from physicsnemo.sym.eq.phy_informer import PhysicsInformer
6 from physicsnemo.sym.eq.pdes.navier_stokes import NavierStokes
7
8 # -----
9 # Load graph dataset
10 # -----
11 dataset = CFDDataset("data_gnn/processed")
12 loader = DataLoader(
13     dataset,
14     batch_size=1,
15     shuffle=True
16 )
17
18 # -----
19 # Define GNN model
20 # -----
21 model = MeshGraphNet(
22     input_dim_nodes=6,
23     input_dim_edges=1,
24     output_dim=4,
25     processor_size=4
26 ).to(device)
27
28 # -----
29 # Define Navier-Stokes equations
30 # -----
31 ns = NavierStokes(
32     nu=0.89e-6,
33     rho=997.561,
34     dim=2,
35     time=False
36 )
37 phy_informer = PhysicsInformer(
38     required_outputs=[
39         "continuity",
40         "momentum_x",
41         "momentum_y"
42     ],
43     equations=ns
44 )

```

```

45
46 # -----
47 # Training setup
48 # -----
49 optimizer = torch.optim.Adam(
50     model.parameters(),
51     lr=1e-3
52 )
53 loss_fn = torch.nn.MSELoss()
54
55 # -----
56 # Training loop
57 # -----
58
59 for epoch in range(epochs):
60
61     for X, Y, edge_index, edge_attr in loader:
62
63         graph = build_dgl_graph(edge_index,
64                                 X.shape[0]
65         )
66         pred = model(X,
67                     edge_attr,
68                     graph
69         )
70
71         # -----
72         # Data loss
73         # -----
74         loss_data = loss_fn(pred, Y)
75
76         # -----
77         # Physics residuals
78         # -----
79         residuals = phy_informer.forward({
80             "coordinates": coords,
81             "u": u,
82             "v": v,
83             "p": p
84         })
85         loss_phy = (
86             residuals["continuity"]**2
87         ).mean()
88         # -----
89         # Total loss
90         # -----
91         loss = loss_data + 0.01 * loss_phy
92         optimizer.zero_grad()
93         loss.backward()
94         optimizer.step()
95
96     # -----
97     # Save trained model
98     # -----
99     torch.save(
100         model.state_dict(),
101         "best_gnn_pinn_model.pth"

```

101 )

**Listing A.10:** Example implementation syntax of the GNN-PINN prediction workflow used in NVIDIA PhysicsNeMo.

```
1
2 import torch
3 import pyvista as pv
4 import dgl
5 from physicsnemo.models.meshgraphnet import MeshGraphNet
6
7 # -----
8 # Load trained model
9 # -----
10 model.load_state_dict(
11     torch.load("best_gnn_pinn_model.pth")
12 )
13 model.eval()
14
15 # -----
16 # Build graph input
17 # -----
18 X = np.hstack([
19     coords,
20     boundary_conditions
21 ])
22 graph = build_dgl_graph(
23     edge_index,
24     num_nodes
25 )
26
27 # -----
28 # Predict CFD fields
29 # -----
30 with torch.no_grad():
31
32     pred = model(
33         X_tensor,
34         edge_attr,
35         graph
36     )
37
38 # -----
39 # Extract predictions
40 # -----
41 pressure = pred[:, 0]
42 u = pred[:, 1]
43 v = pred[:, 2]
44 temperature = pred[:, 3]
45
46 # -----
47 # Save prediction as VTK
48 # -----
49 mesh = pv.UnstructuredGrid(
50     cells,
```

```
51     celltypes,  
52     points  
53 )  
54 mesh["Pressure"] = pressure  
55 mesh["Temperature"] = temperature  
56 #....  
57 #....  
58 mesh.save("gnn_pinn_prediction.vtk")
```

### A.5.4 FNO Model Workflow

The FNO workflow developed using NVIDIA PhysicsNeMo consists of three main stages, namely dataset preparation, model training, and prediction with postprocessing. The primary Python scripts used for these stages are listed below.

**Listing A.11:** Example implementation syntax of the FNO dataset preparation workflow used in NVIDIA PhysicsNeMo.

```
1
2 import pyvista as pv
3 import numpy as np
4
5 # -----
6 # Read CFD mesh
7 # -----
8 mesh = pv.read(case_file)
9
10 # -----
11 # Create structured grid
12 # -----
13 Xi, Yi = np.meshgrid(xi, yi)
14 structured_grid = create_uniform_grid(
15     Xi,
16     Yi
17 )
18
19 # -----
20 # Interpolate CFD data
21 # -----
22 sampled = structured_grid.sample(mesh)
23 mask = sampled["vtkValidPointMask"]
24
25 # -----
26 # Build input channels
27 # -----
28 input_channels = []
29 input_channels.append(Xi)
30 input_channels.append(Yi)
31 input_channels.append(mask)
32 input_channels.append(U_large)
33 input_channels.append(U_small)
34 input_channels.append(T_large)
35 input_channels.append(T_small)
36 X = np.stack(input_channels, axis=0)
37
38 # -----
39 # Build output channels
40 # -----
41 pressure = sampled["Pressure"]
42 temperature = sampled["Temperature"]
43 velocity = sampled["Velocity"]
44 Y = np.stack([
45     pressure,
46     temperature,
47     velocity
```

```

48 ], axis=0)
49
50 # -----
51 # Save structured dataset
52 # -----
53 np.savez_compressed(
54     "case_data.npz",
55     X=X,
56     Y=Y,
57     mask=mask
58 )

```

**Listing A.12:** Example implementation syntax of the FNO training workflow used in NVIDIA PhysicsNeMo.

```

1
2 import torch
3 from physicsnemo.models.fno import FNO
4
5 # -----
6 # Load structured dataset
7 # -----
8 dataset = CFDDataset("data_fno/processed")
9 loader = DataLoader(
10     dataset,
11     batch_size=1,
12     shuffle=True
13 )
14
15 # -----
16 # Define FNO model
17 # -----
18 model = FNO(
19     in_channels=7,
20     out_channels=4,
21     dimension=2,
22     latent_channels=48,
23     num_fno_layers=8,
24     num_fno_modes=48
25 ).to(device)
26
27 # -----
28 # Define masked loss
29 # -----
30 loss_fn = MaskedMSELoss()
31 optimizer = torch.optim.Adam(
32     model.parameters(),
33     lr=1e-3
34 )
35
36 # -----
37 # Training loop
38 # -----
39 for epoch in range(epochs):
40     for X, Y, mask in loader:

```

```

41     pred = model(X)
42     loss = loss_fn(pred,
43                     Y,
44                     mask
45                 )
46     optimizer.zero_grad()
47     loss.backward()
48     optimizer.step()
49
50     # -----
51     # Save trained model
52     # -----
53     torch.save(
54         model.state_dict(),
55         "best_fno_model.pth"
56     )

```

**Listing A.13:** Example implementation syntax of the FNO prediction workflow used in NVIDIA PhysicsNeMo.

```

1
2 import torch
3 import pyvista as pv
4 from physicsnemo.models.fno import FNO
5
6 # -----
7 # Load trained model
8 # -----
9 model.load_state_dict(
10     torch.load("best_fno_model.pth")
11 )
12 model.eval()
13
14 # -----
15 # Build structured input
16 # -----
17 input_channels = []
18 input_channels.append(Xi)
19 input_channels.append(Yi)
20 input_channels.append(mask)
21 input_channels.append(U_large)
22 input_channels.append(U_small)
23 input_channels.append(T_large)
24 input_channels.append(T_small)
25 X = np.stack(input_channels, axis=0)
26
27 # -----
28 # Predict CFD fields
29 # -----
30 with torch.no_grad():
31     pred = model(X_tensor)
32
33 # -----
34 # Convert prediction to VTK
35 # -----

```



```
36 grid = pv.StructuredGrid()
37 grid.points = points
38 grid.dimensions = (Nx, Ny, 1)
39 grid["Pressure"] = pressure
40 grid["Temperature"] = temperature
41 #.....
42 #.....
43 grid.save("fno_prediction.vtk")
```



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY