ABSTRACT

CHANG, CHIH-WEI. Data-Driven Modeling of Nuclear System Thermal-Hydraulics. (Under the direction of Dr. Nam T. Dinh).

The goal of this work is to develop a methodology to enhance predictive power of data-driven nuclear system thermal-hydraulics (NSTH) simulation using machine learning. NSTH simulation is instrumental for reactor design, safety analysis, and operator training. Traditionally, it takes extensive research efforts to develop insights and mechanistic understanding of physical processes in reactor system through analysis of experimental data and capture the data in a compact model form. The long time and large resources required for model development constrain the simulation code applicability in dealing with newly designed systems involving new geometries and new coolants. As an alternative to mechanistic and semi-analytical models, some machine learning methodologies, especially deep learning, can effectively capture underlying correlations behind multi-scale data using nonparametric models, or so-called data-driven models. Such approach is referred to as data-driven modeling.

The technical approach of the dissertation consists of three components. First, the technical background overview navigates the essential knowledge from related disciplines, including thermal-hydraulics models, system simulation, and machine learning. Second, a methodology is developed to accomplish data-driven modeling of NSTH. The development includes a system that classifies machine learning frameworks for NSTH based on data and knowledge requirements. Finally, framework demonstration focuses on the use of deep learning, which has demonstrated the capability of a universal approximator. Synthetic examples are formulated to investigate technical challenges of using deep learning to achieve data-driven modeling of NSTH.

Five machine learning frameworks for NSTH have been introduced in the dissertation including physics-separated ML (PSML or Type I ML), physics-evaluated ML (PEML or Type II ML), physics-integrated ML (PIML or Type III ML), physics-recovered (PRML or Type IV ML), and physics-discovered ML (PDML or Type V ML). The framework classification is based on knowledge and data requirements. Type III ML framework is formulated for the first time in this study. The insights obtained from synthetic examples indicate that Type III ML has the highest potential in leveraging the value from “big data” in thermal fluid research while ensuring data-model consistency.

Various numerical experiments are formulated ranging from system-level simulation to computational fluid dynamics (CFD) to exhibit the advantage of deep learning (DL) for model development. The case studies of system-level simulation using Type I, Type II, and Type III ML frameworks ensure that simulation results satisfy conservation laws with a moderate amount of data. The results indicate that system-level two-phase mixture models can be solved with DL-based closure relations without interference of numerical instability.

The CFD case study exhibits that the DL-based Reynolds stress model can assimilate millions of data points to reduce forecast error. Performance of the DL-based stress can be quantified by flow features coverage mapping. The results show that Reynolds-averaged turbulence modeling with the DL-based Reynolds stress model can replicate the transient flow prediction by Reynolds-averaged Navier-Stokes simulation with the k-ε model.

Data-Driven Modeling of Nuclear System Thermal-Hydraulics

by
Chih-Wei Chang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Nuclear Engineering

Raleigh, North Carolina

2018

APPROVED BY:

_____	_____
Dr. Nam T. Dinh Committee Chair	Dr. Joseph Michael Doster
_____	_____
Dr. Paul Turinsky	Dr. Maria Avramova
_____	
Dr. Ralph Smith	

DEDICATION

The dissertation is dedicated to my parents.

BIOGRAPHY

Chih-Wei Chang was born in 1987, Taichung, Taiwan.

He attended National Tsing Hua University from 2005 to 2009, and graduated with a B.S. in Engineering and System Science. While at Tsing Hua, he led the BWR core loading pattern design project for 20% power uprate using CASMO-3/SIMULATE-3, funded by National Science Council. He was also elected as the president for Tsing Hua Equestrian Club.

He attended National Tsing Hua University from 2009 to 2011, and graduated with a M.S. in Nuclear Engineering and Science. During his master study, he did intern at Idaho National Laboratory and developed a 3D multigroup nodal diffusion code with the capability of random source treatment, supervised by Dr. Abderrafi M. Ougouag and Prof. Yen-Wan Hsueh.

In 2013, he entered the PhD program at North Carolina State University (NCSU). He has been working on methodology development of data-driven thermal fluid simulation using deep learning, supervised by Prof. Nam T. Dinh. Before Chang joined NCSU, he was a research assistant at Idaho State University and worked on the phylogenetic tree project, supervised by Prof. Shu-Chuan (Grace) Chen.

ACKNOWLEDGEMENTS

I would like to thank Prof. Nam T. Dinh for his advice and encouragement during my PhD study, and Prof. Dinh's pioneering perspective on thermal-hydraulics inspires me to accomplish the dissertation research. I would like to thank Dr. Abderrafi M. Ougouag for his encouragement and recommendation to allow me to pursue the degree at North Carolina State University. I would like to thank Prof. Paul Turinsky for helping me with framework development, dissertation structure, and English editing. I would like to thank Prof. Ralph Smith for helping me with structured thinking and statistical methodology. I would like to thank Prof. Joseph Michael Doster for helping me with two-phase modeling and system simulation. I would like to thank Prof. Maria Avramova for helping me with data sources and code development. I would like to thank my dissertation committee members for their extraordinary support in this dissertation that improves the quality of my PhD research.

I would like to thank Dr. Sacit M. Cetiner, Dr. Olumuyiwa Omotowa, Dr. Xianliang Lei, Dr. Jin-Seok Hwang, and Dr. Michael Scott Greenwood for their expert advice on system simulation. I would like to thank Prof. Shu-Chuan (Grace) Chen and Prof. Jay Kunze for their support during my study at Idaho State University.

I would like to thank my parents and grandparents for their support that allow me to concentrate on my study and pursue my dream. I would like to thank my colleagues, Dr. Jun Fang, Dr. Guojing Hou, Dr. Juntao Liu, Yangmo Zhu, Yang Liu, Dr. Hao-Ping Chang, Dr. Jinyong Feng, Dr. Ching-Yun Cheng, Benjamin Bond, Paridhi Athe, Yuwei Zhu, Yan Zhang, Linyu Lin, Han Bao, Botros Hanna, Dr. Michael Fusco, Abdullah Zafar, and Joomyung Lee for inspiring talks. It has been wonderful working and having fun with them during my PhD life.

Finally, I would like to acknowledge the support from the US Department of Energy via the Consortium for Advanced Simulation of Light Water Reactors (CASL), NEUP Integrated Research Project, Oak Ridge National Laboratory with the Nuclear-Renewable Hybrid Energy Systems Project, and NVIDIA Corporation for the Titan Xp GPU used for this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AI	Artificial intelligence
API	Application program interface
ARAED	Available, relevant, and adequately evaluated data
CFD	Computational fluid dynamics
CNN	Convolutional neural networks
CR	Closure relation
DaaS	Data as a service
DDM	Data-driven modeling
DL	Deep learning
DNN	Deep neural networks
DNS	Direct numerical simulation
FNN	Feedforward neural networks
GPU	Graphics processing unit
GSN	Goal structuring notation
GUI	Graphical user interface
HL	Hidden layer
HU	Hidden unit
IET	Integral effects tests
IR	Inferred
LES	Large eddy simulation
MaaS	Method as a service
MC	Model complexity
MIC	Model-insight consistency
ML	Machine learning
MMD	Method of manufactured data
NIST	National Institute of Standards and Technology
NN	Neural networks
NPP	nuclear power plant
NSTH	Nuclear system thermal-hydraulics
PaaS	Platform as a service
PCDL	Physics-constrained deep learning
PCML	Physics-constrained machine learning
PDE	Partial differential equaiton
PDE-DL	PDE-constrained deep learing
PDE-ML	PDE-constrained machine learning
PDML	Physics-discovered machine learning (Type V ML)
PEML	Physics-evaluated machine learning (Type II ML)
PIML	Physics-integrated machine learning (Type III ML)
PIV	Particle image velocimetry

PRML	Physics-recovered physics-recovered (Type IV ML)
PSML	Physics-separated machine learning (Type I ML)
RANS	Reynolds-averaged Navier–Stokes
RANS-DL	RANS simulation using DL-based closures
RELAP	Reactor Excursion and Leak Analysis Program
RF	Recovery factor
ROM	Reduced-order model
RPV	Reactor pressure vessel
SET	Separate effects tests
SGS	Sub-grid-scale
TDMI	Total data-model integration
TFM	Two-fluid model
TFS	Thermal Fluid Simulation
TMM	Two-phase mixture model
TMM-DL	TMM using DL-based closures
TMM-ZF	Two-phase mixture model with Zuber-Findlay correlation
TRAC	Transient Reactor Analysis Code
TRACE	TRAC/RELAP Advanced Computational Engine
t-SNE	t-distributed stochastic neighbor embedding

NOMENCLATURE

A	Area
E	Two-phase correction term in the internal energy balance equation
G	Mass flux
M	Two-phase correction term in the momentum balance equation
P	Pressure
q	Heat flux
u	Internal energy
v	Velocity
x	Steam quality

Greek

α	Void fraction
ρ	Density
τ	Shear stress
υ	Specific volume

Subscripts

2Φ	Two-phase mixture
l	liquid
i	Interfacial
g	gas

CHAPTER 1. INTRODUCTION

1.1. Motivation

Nuclear System Thermal-Hydraulics (NSTH) features multi-scale and multi-physics dynamics, involving coupled mass-momentum-energy transport phenomena over multiple scales. Conducting simulation for such complex systems requires knowledge from related disciplines, including thermal-hydraulics, neutronics, and material science. NSTH simulation is based on solving mass-momentum-energy conservation equations (partial differential equations, PDEs) with embedded sub-grid-scale physics (SGS) models [1, 2]. SGS models are often referred to as "closure relations" (CRs) or constitutive models, as they serve to close PDE-based models ranging from large-eddy simulation to system-level simulation.

Traditionally, it takes extensive efforts to gain insights and develop mechanistic understanding of physical processes in reactor systems through analysis of experimental data to represent said data in a compact model form. The long time required for model development constrains the application of simulation when dealing with newly designed systems including new coolants and new geometries. As an alternative to mechanistic and semi-analytical models, some machine learning (ML) methodologies can effectively capture underlying correlations behind multi-scale data using nonparametric models, or so-called data-driven models. Such approach is referred to as data-driven modeling (DDM).

The goal of this dissertation is to establish a technical basis for novel data-driven model development and employ data-driven modeling to maximize the predictive capability of NSTH simulation with machine learning. The dissertation research is motivated by the growing interest [3] and development of machine learning models in thermal-hydraulics. The trend is powered by the advent of data-intensive research methods such as high-fidelity simulations, large-scale GPU

(graphic processing unit) computing, and advanced machine learning algorithms, particularly deep learning (DL) [4].

1.2. Applications of nuclear system thermal-hydraulics simulation

Before establishing data-driven methodology for Nuclear System Thermal-Hydraulics (NSTH), it is essential to understand applications and requirements of NSTH simulation. Then we can use data-driven modeling to extend the applicability of next generation NSTH codes. NSTH simulation [5] involves three applications as follows.

a) **System design.** System design requires NSTH simulation platform to include extensible models such that researchers can customize simulation layouts to explore various reactor designs.

b) **Safety analysis.** Safety analysis requires NSTH models to be adaptive for distinct scenarios based on two strategic approaches. First, risk assessment is a conservative analysis that aims at understanding how severe an accident can be. The priority is to bound accidents such that regulators can make laws for nuclear power plant licensing. Second, risk management focuses on how to mitigate an accident scenario. The priority is to control an accident progression. These two strategies require different model assumptions for system analysis.

c) **Operator training.** Operator training requires interactive GUI (graphical user interface) systems with reconfigurable system modules and growable data libraries. Therefore, plant operators can be trained by various accident scenarios.

Based on lessons learned [5-8] from present state-of-the-art NSTH codes such as RELAP [9], TRACE [10], and MELCOR [11], we classify three functions includes methods from related disciplines to develop next generation NSTH code packages as follows.

a) **Model development**. Model development requires models to be adaptive and extensive. Thus, those models can discover underlying physics behind data and build closure relations to close conservation equations.

b) **Hierarchical model repository.** Hierarchical model repository allows model selection based on flow patterns. For instance, when two phases (liquid and vapor) are tightly coupled [8], two-phase mixture models [12] performs better than the two-fluid model [1]. When two phases are loosely coupled [8], the two-fluid model is more applicable than two-phase mixture models.

c) **Simulation platform.** Simulation platform can benefit the development and maintenance of NSTH codes. Such platform allows mathematicians to focus on verification of numerical solvers. Then physics models can be quickly deployed on a platform to achieve cost-effective and reliable simulation.

1.3. Data convergence

Data-driven modeling requires the use of a substantial amount of data. A fundamental assumption for data-driven modeling is that model accuracy can be improved when training models with large data. However, the increase of data does not guarantee that models can fully capture trends of data. Figure 1.1 [13] qualitatively depicts performance of data-driven models by different learning algorithms. Performance refers to the capability of models to capture hidden correlations behind data. Traditional learning algorithms are limited by model forms, and their

performance is saturated after a threshold. However, traditional ML models can achieve a specified fidelity with limited data. Those models are usually developed based on knowledge and they could be derived from physics dimensional analysis. When data are limited, traditional machine learning models may have more predictive capabilities than deep learning (DL) models, which require a substantial amount of data for training. On the contrary, deep learning includes adaptive model forms, which can change degrees of freedom based on different amount of data. Therefore, data-driven models using deep learning can benefit from "big data" in thermal-hydraulics. An experiment is formulated in CHAPTER 8 to demonstrate that DL-based closures can assimilate millions of data points.

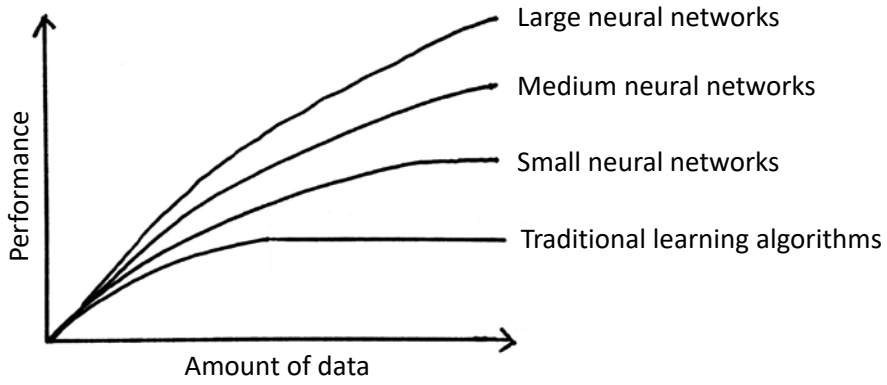


Figure 1.1. Comparison of DL to traditional machine learning (adopted after Ng [13]).

1.4. Total data-model integration (TDMI)

The concept of total data-model integration (TDMI) [14] refers to the integrated use of datasets, models, and simulations to support decision making. Research has been pursued in formulating and applying the TDMI framework to accomplish data-driven modeling of NSTH, particularly leveraging on advances in machine learning techniques. Nowadays, thermal-hydraulics data accumulate rapidly in a significant amount from high-resolution simulations and

experiments, primarily due to the affordability of high-performance computing and advances in flow diagnostics, thermal imaging, and other measurement equipment such as high-speed, high-resolution optical and infrared cameras.

The value of those high-fidelity data lies with their use (usability) to reduce uncertainty in simulation. The "high fidelity" refers to the data which have been adequately evaluated, and hence trustworthy. Lewis *et al.* [15] investigated a strategy of using high-fidelity data from computational fluid dynamics (CFD) to inform low-fidelity models in system-level thermal hydraulics simulation. They also demonstrated this high-to-low (Hi2Lo) strategy by utilizing a neutron transport equation to inform a neutron diffusion equation. Methodologically, TDMI belongs to the Hi2Lo strategy. Its distinctive features relax closure relations from their traditional "mechanistic" models to ML-based models, which have the potential to extract values of a substantial amount of data.

For the "Big Data" to become useful in TDMI, it has to undergo several processing steps [16]. First, results of high-fidelity simulations and experiments need to be collected, categorized, and archived in an easily accessible storage format. Second, the value of data as information needs to be assessed, to establish their relevance to the conditions and models under consideration, so that these data become useful information. Third, data are processed by various methods (including ML) to recognize underlying correlations behind the information. The so-developed intelligence (e.g., in the form of closure relations) is used to enable thermal fluid simulation in applications.

Stemming from the preceding discussion, Figure 1.2 depicts the TDMI framework that includes the concept of (Data/Method/Platform) "as a Service (aaS)" [17]. The "aaS" notations are employed to denote components (modules) of a workflow in a "divide-and-conquer" strategy that allows us to decompose the framework by different disciplines. We can define requirements,

evaluate methods, and review the essential knowledge in each discipline such as machine learning, thermal fluid experiment and simulation, and numerical methods. This concept allows each module to be reused, extended, and improved based on newly observed data as well as newly developed state-of-the-art methods.

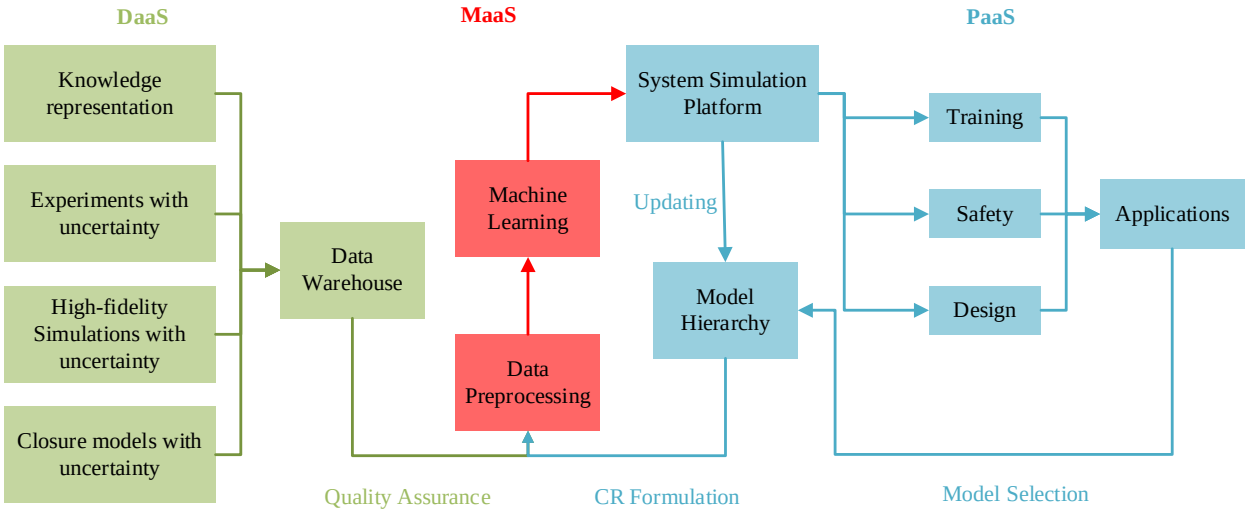


Figure 1.2. Overview of the total data-model integration (TDMI) framework.

The DaaS module integrates data from various sources to support closure developments. The MaaS module includes different ML algorithms that can be deployed to infer models from data. The PaaS module contains thermal fluid models that are adaptive to various applications. The detail functions of each module are described:

Data as a Service (DaaS). Four types of data need to be stored in the data warehouse including the knowledge representation, experiments, high-fidelity simulations, and existing closure models. Most importantly, uncertainty information needs to be stored as well. Experts' knowledge needs to be formalized and quantified so that the information can be used to improve modeling and simulation. The experiment provides evidence to support the development of closure relations. The simulation includes direct numerical simulation (DNS) or validated CFD results to

support the model development when the budget or time frame limits full-scale experiments. The existing models are compact forms of data from past researches and experiments. They are used under appropriate conditions, and often serve as first estimates when new observations and directly relevant data are not available.

Method as a Service (MaaS). The MaaS module is emphasized by red color to indicate that ML methods can fill the gap between data and thermal fluid models. Machine learning methods are essential for data-driven modeling due to its capability to capture trends of data by nonparametric models. However, if the source of data is uncertain, the data-driven model is also uncertain. Data preprocessing is required to check the consistency[1] between the model and data before training a data-driven model. For instance, when we use a 3D (three-dimensional) simulation to inform a 1D (one-dimensional) model, we should confirm that the spatiotemporal averaging methods for high-resolution data are consistent with the 1D model. After data reprocessing, machine learning techniques are applied to accomplish data-driven modeling. Eventually, ML-based closure relations are incorporated into system simulation platforms to enhance the predictability of simulation for a newly designed system and system with different coolants or geometries.

Platform as a Service (PaaS). Thermal fluid models with distinct hypotheses can be adapted based on each particular condition, and hence minimize the uncertainty for simulation. Each thermal fluid model requires distinct ML-based closures and may need specific numerical schemes for solutions. Therefore, simulation platforms store the thermal-fluid-model hierarchy based on different degrees of averaging, and provide numerical solvers that are verified by

[1] NSTH simulation involves conservation equations with various degrees of averaging from the first principle based on distinct hypotheses. The underlying physics of the conservation equations should be consistent with the experiment or simulation where the available, relevant, and adequately evaluated data (ARAED) are obtained.

mathematicians. The model selection will become application-oriented, and users can deploy a customized system for dynamics analysis by assembling pre-existing components in the model repository. For instance, when liquid and vapor phases are tightly coupled, it is hard to distinguish interfacial details, and a drift-flux model should be used for this condition [8].

1.5. Dissertation overview

1.5.1. Significance and Objectives

The significance of using data-driven modeling to develop nuclear system thermal-hydraulics (NSTH) models stems from three beneficial impacts of such use. These, henceforth identified with objectives, are:

a) Shorten the model development phase;

b) Leverage values of data from advanced validation experiments and high-fidelity numerical simulations;

c) Maximize the predictive capability of NSTH models.

1.5.2. Technical approach

The technical approach to achieve data-driven NSTH simulation includes:

a) Review the essential knowledge from multidisciplinary fields including the thermal-hydraulics, system simulation, and machine learning;

b) Developing a methodology to achieve data-driven modeling of nuclear system thermal-hydraulics. The development includes a system to characterize different approaches to use machine learning in NSTH simulation;

c) Demonstrate through synthetic examples how to employ the developed methodology to accomplish data-driven modeling of NSTH.

1.5.3. Dissertation structure

The dissertation is structured into following chapters.

CHAPTER 2 includes technical background overview. Data-driven modeling of NSTH involves methods from multiple disciplines including thermal-hydraulics, system simulation, and machine learning.

CHAPTER 3 focuses on the formulation of machine learning frameworks to accomplish data-driven modeling of NSTH. Based on distinct strategies of incorporating machine learning (ML) into NSTH, we propose a classification into five frameworks including physics-separated ML (PSML or Type I ML), physics-evaluated ML (PEML or Type II ML), physics-integrated ML (PIML or Type III ML), physics-recovered (PRML or Type IV ML), and physics-discovered ML (PDML or Type V ML).

CHAPTER 4 demonstrates how to employ Type I ML for system-level single-phase flow simulation. The goal is to find the conditions by which DL-based closure relations work compatibly, stably, and effectively with PDE-constrained forward prediction problems. The case study shows how to employ the physics-constrained deep learning strategy to build DL-based closure relations, which are well-posed.

CHAPTER 5 includes the case study that uses Type I ML to close a two-phase mixture model (TMM) [12]. The machine learning strategy developed in Section 3.4.4 is employed to find a reliable and robust DL-based slip closure for mixture models. The result indicates that the two-phase mixture model with the DL-based slip model has predictive capability over a range of flow regimes.

CHAPTER 6 compares performance of Type I, Type II, Type III, and Type V ML frameworks using a case study with nonlinear heat conduction. Thermal conductivity models are

formulated by convolutional neural networks (CNNs) and feedforward neural networks (FNNs). The result indicates a preference for Type II ML under deficient data support. Type III ML can effectively utilize field data, generating more robust predictions than Type I, Type II and Type V ML. CNN-based models exhibit more predictive capabilities than FNN-based models, but CNN-based models require more training data to achieve prediction.

CHAPTER 7 exhibits Type I and Type II ML frameworks applied to Reynolds-averaged turbulence modeling using reference works [18, 19].

CHAPTER 8 demonstrates how to employ Type I and Type II ML frameworks to achieve Reynolds-averaged turbulent flow modeling for unsteady flow. The goal is to construct DL-based Reynolds stress to close Reynolds-averaged Navier-Stokes (RANS) equations. The case study also shows how to use flow features coverage mapping (FFCM) to quantify the coverage of physics. FFCM also has the potential to quantify the predictive capability of DL-based RANS simulation. The result shows that the DL-based Reynolds stress model that assimilated millions of data points can replicate the transient flow prediction by the RANS(k-ε) model.

CHAPTER 9 provides the conclusions and recommendations for future work about applying ML frameworks to NSTH simulation.

1.6. Glossary

This section provides interpretation of several key terminology used in the dissertation.

Big data

Big data refers to a substantial amount of data that cannot readily be interpreted by human beings. Big data hierarchy includes four parts. First, results of high-fidelity simulations and experiments need to be collected, categorized, and archived in an easily accessible storage format. Second, the value of data as information needs to be assessed, to establish their relevance to the

conditions and models under consideration, so that these data become useful information. Third, data are processed by various methods (including ML) to recognize underlying correlations behind the information. The so-developed intelligence (e.g., in the form of closure relations) is used to enable thermal fluid simulation in applications.

Hi2Lo

Hi2Lo refers to the use of high-fidelity models to inform low-fidelity models. The method is proposed by Lewis, Smith, Williams & Figueroa [15]. High-fidelity models can tune model parameters that make low-fidelity models capture expected system characteristics.

High-fidelity model

High-fidelity models refer to models that have been calibrated by data from various sources, and they are valid in a range of flow regimes. Therefore, high-fidelity models are more trustworthy, in the sense of having lower uncertainty, than low-fidelity models.

Methods of manufactured data (MMD)

The method of manufactured data (MMD) refers to generate data for training and testing by high-fidelity models. We can manipulate data quantity and uncertainty to evaluate the performance of data-driven models.

Model calibration

Model calibration refers to the use of a learning algorithm or human efforts to infer model parameters based on data.

Engineering surrogate construction

Engineering surrogate construction uses statistical methods to build models for analysis of system dynamics that is highly nonlinear. Such a system involves multi-scale and multi-physics models, requiring substantial computing power that makes uncertainty quantification unachievable

given a limited time frame. Instead of solving complete models, engineering surrogate construction preserves the features between inputs and outputs. Therefore, it is possible to run surrogates thousands of times that make uncertainty quantification possible.

Total data-model integration (TDMI)

Total data-model integration (TDMI) refers to the integrated use of data, models, and simulations, including integral effects tests, separate effects tests, multi-scale and multi-physics models, and high-fidelity numerical simulations (i.e., DNS, LES, and CFD). TDMI can be used to support decision making. The concept is proposed by Dinh, Nourgaliev, Bui & Lee [14].

Scale separation

System dynamics usually involves multi-scale and multi-physics models. For instance, viscosity is a microscopic property, and it does not depend on flow velocity for a Newtonian fluid. We can separately measure this material property, and then use it in conservation equations. However, viscosity depends on flow velocity for a non-Newtonian fluid. To obtain the property, we need to account for the entire system dynamics.

CHAPTER 2. TECHNICAL BACKGROUND OVERVIEW

2.1. Introduction

This chapter provides technical background about data-driven modeling (DDM) of nuclear system thermal-hydraulics (NSTH) including thermal-hydraulics models, system simulation as well as machine learning (ML).

2.2. Thermal-hydraulics models

2.2.1. Reynolds-averaged Navier-Stokes equations

Reynolds-averaged Navier-Stokes (RANS) equations are widely used in fluid engineering simulation and analysis due to its computational efficiency. The next generation system code is expected to be multi-dimensional. Therefore, the dissertation uses RANS models to demonstrate that DDM with DL can leverage values from a substantial amount of data. Eq. (2.1) and Eq. (2.2) show the Reynolds-averaged continuity and momentum equations [20] without the body force for an incompressible Newtonian fluid where $\bar{u}$ is the time averaged velocity. In Eq. (2.2), *D/Dt*, ρ, $\bar{p}$, $(\bar{\tau}_{ij})_{lam}$, $(\bar{\tau}_{ij})_{turb}$ are the material derivative, fluid density, mean pressure, laminar shear stress, and Reynolds stress tensor.

$$\frac{\partial \bar{u}_j}{\partial x_j} = 0 \tag{2.1}$$

$$\rho \frac{D\bar{u}_i}{Dt} = -\frac{\partial \bar{p}}{\partial x_j} + \frac{\partial \left(\bar{\tau}_{ij}\right)_{lam}}{\partial x_j} + \frac{\partial \left(\bar{\tau}_{ij}\right)_{turb}}{\partial x_j} \tag{2.2}$$

Eq. (2.3) gives the laminar shear stress with Stokes' hypothesis where μ, δ_{ij}, and *k* are the molecular viscosity, Kronecker delta, and direction. Eq. (2.4) shows the Reynolds stress where u' is the fluctuation velocity.

$$\left(\bar{\tau}_{ij}\right)_{lam} = \mu\left[\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \bar{u}_k}{\partial x_k}\right] \tag{2.3}$$

$$\left(\bar{\tau}_{ij}\right)_{lam} = -\rho\overline{u_i' u_j'} \tag{2.4}$$

The linear eddy viscosity model (LEVM) has been widely used to represent Reynolds stress that leads to various mechanistic turbulence models [21] such as Spalart-Allmaras [22], *k*-ε [23], and *k*-ω [24] models.

2.2.2. Two-phase flow modeling

2.2.2.1. 1D area-averaged two-fluid model

The two-fluid model (TFM) [1] includes the mass-momentum-energy conservation equation for two phases with two fields. Eq. (2.5) and Eq. (2.6) show mass balance equations for liquid (*l*) and vapor (*g*) where $\dot{\delta}_i'$ is the interfacial mass transfer rate.

$$A_x \frac{\partial \alpha_l \rho_l}{\partial t} + \frac{\partial \alpha_l \rho_l v_l A_x}{\partial z} = -\dot{\delta}_i' \tag{2.5}$$

$$A_x \frac{\partial \alpha_g \rho_g}{\partial t} + \frac{\partial \alpha_g \rho_g v_g A_x}{\partial z} = \dot{\delta}_i' \tag{2.6}$$

Eq. (2.7) and Eq. (2.8) give momentum balance equations for each phase where $\hat{v}_i$ and P_i are the interfacial velocity and perimeter.

$$A_x \frac{\partial \alpha_l \rho_l v_l}{\partial t} + \frac{\partial \alpha_l \rho_l v_l v_l A_x}{\partial z} = -\dot{\delta}_i' \hat{v}_i - \alpha_l A_x \frac{\partial P}{\partial z} - \tau_{wl} P_{wl} + \tau_i P_i + \alpha_l \rho_l g_z A_x \tag{2.7}$$

$$A_x \frac{\partial \alpha_g \rho_g v_g}{\partial t} + \frac{\partial \alpha_g \rho_g v_g v_g A_x}{\partial z} = \dot{\delta}_i' \hat{v}_i - \alpha_g A_x \frac{\partial P}{\partial z} - \tau_{wg} P_{wg} - \tau_i P_i + \alpha_g \rho_g g_z A_x \tag{2.8}$$

Eq. (2.9) and Eq. (2.10) show internal energy balance equations for each phase. When the liquid is vaporizing, $\hat{h}_l$ and $\hat{h}_g$ are the liquid enthalpy and saturated vapor enthalpy. If the vapor is condensing, $\hat{h}_l$ and $\hat{h}_g$ are the saturated liquid enthalpy and vapor enthalpy.

$$A_x \frac{\partial \alpha_l \rho_l u_l}{\partial t} + \frac{\partial \alpha_l \rho_l u_l v_l A_x}{\partial z} + \dot{\delta}_i' \hat{h}_l = -P \frac{\partial \alpha_l v_l A_x}{\partial z} - P \frac{\partial \alpha_l A_x}{\partial t} + q_{wl}'' P_{wl} + q_{il}'' P_i \quad (2.9)$$

$$A_x \frac{\partial \alpha_g \rho_g u_g}{\partial t} + \frac{\partial \alpha_g \rho_g u_g v_g A_x}{\partial z} - \dot{\delta}_l' \hat{h}_g = -P \frac{\partial \alpha_g v_g A_x}{\partial z} - P \frac{\partial \alpha_g A_x}{\partial t} + q_{wg}'' P_{wg} + q_{ig}'' P_i \quad (2.10)$$

2.2.2.2. 1D area-averaged two-phase mixture model

The two fluid model resolves detail information for each phase, but it may suffer from significant uncertainty due to model form uncertainty of interfacial transfer correlations. As an alternative, a two-phase mixture model (TMM) [12] reduces the number of closure relations and TMM is capable to handle phase appearance and disappearance without any singular point. Eq.(2.11)-(2.15) give the mass-momentum-energy conservation equation for the three-equation TMM where $M_{2\Phi}$ and $E_{2\Phi}$ are two-phase correction terms for momentum and internal energy balance equations. The three-equation TMM is the fundamental mixture model, and Eq. (2.16) defines the mixture property where ϕ can be 1, v, u, or υ. When single-phase flow is present, $M_{2\Phi}$ and $E_{2\Phi}$ are equal to zero and the three-equation TMM becomes the single-phase flow model.

$$A_x \frac{\partial \rho}{\partial t} + \frac{\partial \rho v A_x}{\partial z} = 0 \quad (2.11)$$

$$A_x \frac{\partial \rho v}{\partial t} + \frac{\partial \rho v v A_x}{\partial z} = -A_x \frac{\partial P}{\partial z} - \tau_w P_w + \rho g_z A_x - M_{2\Phi} \quad (2.12)$$

$$A_x \frac{\partial \rho u}{\partial t} + \frac{\partial \rho u v A_x}{\partial z} = -P \frac{dv A_x}{dz} + q_w'' P_w - E_{2\Phi} \quad (2.13)$$

$$M_{2\Phi} = \frac{\partial}{\partial z}\left[\frac{\alpha_g \rho_g \alpha_l \rho_l}{\rho} (v_g - v_l)^2 \right] A_x \quad (2.14)$$

$$E_{2\Phi} = \frac{\partial}{\partial z}\left[\frac{\alpha_g \rho_g \alpha_l \rho_l}{\rho}(u_g - u_l)(v_g - v_l)\right]A_x + P\frac{\partial}{\partial z}\left[\frac{\alpha_g \rho_g \alpha_l \rho_l}{\rho}(\upsilon_g - \upsilon_l)(v_g - v_l)\right]A_x \quad (2.15)$$

$$\rho\phi = \alpha_l \rho_l \phi_l + \alpha_g \rho_g \phi_g \quad (2.16)$$

The simplest three-equation TMM is the homogeneous equilibrium model that requires assumptions of the equal velocity, temperature, and pressure for each phase. This model also assumes the liquid and vapor are at saturation. However, the homogeneous (equal phasic velocities) assumption is not valid for a vertical pipe problem because of buoyancy. We can add a slip (v_g/v_l) closure to capture the effect of buoyancy, and there are more discussions on how to select a slip closure for two-phase mixture models in Section 2.2.2.3. In the meanwhile, we need drift-flux-based void fraction models to close the three-equation TMM such as the Zuber-Findlay correlation [25].

TMMs can consistently increase the fidelity of prediction for different flow patterns by increasing phasic equations. Figure 2.1 [5] summarizes family of TMMs based on various assumptions. For example, the three-equation TMM is not valid when system includes a subcooled liquid. We can add Eq. (2.5) (phasic mass equation) to the three-equation TMM to extend its applicability. The new four-equation TMM requires the closure of interfacial mass transfer details.

The five-equation TMM can be formulated in two diverse ways by either the phasic mass-momentum or mass-energy equation. The five-equation TMM with the phasic mass-momentum equation assumes that the vapor is at saturation. Therefore, we cannot model the system with both subcooled liquid and superheated vapor. The error made by this assumption is not significant because the superheated vapor is quickly cooled by subcooled liquid. On the contrary, the five-equation TMM with the phasic mass-energy equation does not require the assumption of saturation; instead, it requires closures to resolve the relative phasic velocity.

The required closure relations are inceased as we adds euqations in TMMs. The five-equation TMM can model non-homogeneous non-equilibrium flow. However, if uncertainty is induced by closure relations, the three-equation model may perform well since it only requires limited closure models.

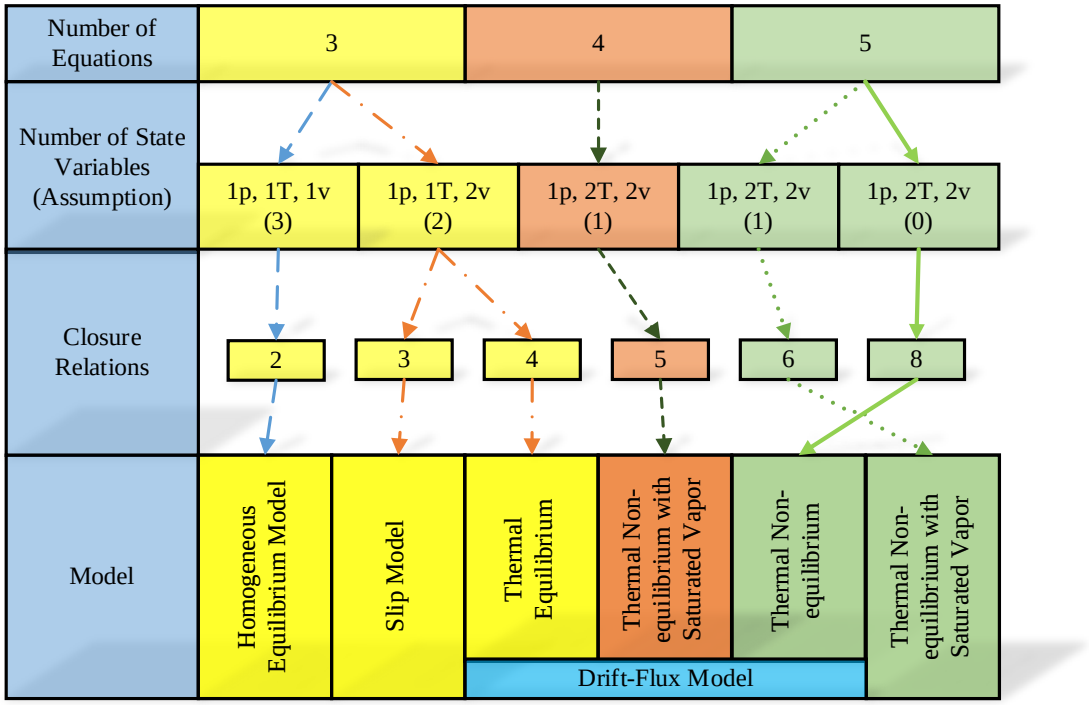


Figure 2.1. Family of two-phase mixture models (adopted after Wulff [5]).

2.2.2.3. Void fraction closures for two-phase mixture models

A void fraction (α) closure model is essential for two-phase mixture models. Void fraction can determine two-phase characteristics such as the mixture density and slip velocity. Traditionally, there are two approaches to obtain void fraction models: analytical and empirical derivations. Both methods result in numerous models for each flow regime, and the use of multiple models results in several issues such as discontinuities for flow regime transitions and model form uncertainty. In the meanwhile, it usually requires extensive efforts to gain the insights and

mechanistic understanding to develop a void fraction closure. The long time required for new model development limits the applicability of two-phase mixture models while dealing with newly designed systems, including new geometries and new coolants. We review the traditional approach for the development of void fraction models in this section.

The cross-section void fraction is widely used in two-phase mixture models. It can be obtained by mean liquid (l) and vapor (g) velocities as given in Eq. (2.17) and Eq. (2.18) where G, ρ, and x are the total mass flux, density, and steam quality.

$$v_l = \frac{(1-x)G}{\alpha_l \rho_l} \tag{2.17}$$

$$v_g = \frac{xG}{\alpha_g \rho_g} \tag{2.18}$$

We can divide Eq. (2.18) by Eq. (2.17), and define the slip factor ($S = v_g/v_l$) to obtain the void-quality-slip model:

$$\alpha = \left[1 + \frac{1-x}{x}\frac{\rho_g}{\rho_l} S\right]^{-1} \tag{2.19}$$

For a homogeneous flow, the slip is equal to one since the liquid and vapor have the same velocity. Most of TMMs require closure relations to resolve the slip factor, for example, three-equation and four-equation two-phase mixture models. Traditionally, there are two approaches to obtain the slip.

The analytical approach includes momentum flux (Φ_M) and kinetic energy models. Eq. (2.20) shows the momentum flux [26] for separated flows.

$$\Phi_M = \dot{m}^2 \left[\frac{x^2 v_g}{\alpha} + \frac{(1-x)^2 v_l}{1-\alpha}\right] \tag{2.20}$$

We assume that void fraction can be obtained by the minimum momentum flux. Therefore, we can take the derivative of Eq. (2.20) with respect to α, and then compare the result with Eq. (2.19) to obtain the slip factor as follows.

$$S = \left(\frac{\rho_l}{\rho_g} \right)^{1/2} \tag{2.21}$$

For the annular flow, Zivi [27] derived the kinetic energy equation by Eq. (2.22) and assumed void fraction can be obtained by the minimum kinetic energy. Therefore, we can take the derivative of Eq. (2.22) with respect to α, and then compare the result with Eq. (2.19) to obtain the slip factor by (2.23).

$$KE = \frac{1}{2}\rho_g \frac{\dot{m}^2 x^2}{\alpha^2 \rho_g^2} \frac{\dot{m} x A_x}{\rho_g} + \frac{1}{2}\rho_l \frac{\dot{m}^2 (1-x)^2}{(1-\alpha)^2 \rho_l^2} \frac{\dot{m}(1-x) A_x}{\rho_l} \tag{2.22}$$

$$S = \left(\frac{\rho_l}{\rho_g} \right)^{1/3} \tag{2.23}$$

Table 2.1 summarizes selected analytical void fraction models where e denotes the entrainment fraction (droplet mass flow rate divided by total liquid mass flow rate).

Table 2.1. Summary of selected analytical void fraction models.

Model	Correlation	Condition
Momentum flux [26]	$\alpha = \left[1 + \frac{1-x}{x} \left(\frac{\rho_g}{\rho_l} \right)^{1/2} \right]^{-1}$	Separated flows
Zivi [27]	$\alpha = \left[1 + \frac{1-x}{x} \left(\frac{\rho_g}{\rho_l} \right)^{2/3} \right]^{-1}$	Annular flow with no entrainment
Zivi [27]	$\alpha = \left\{ 1 + e\left(\frac{1-x}{x} \right) \frac{\rho_g}{\rho_l} + (1-e)\left(\frac{1-x}{x} \right) \left(\frac{\rho_g}{\rho_l} \right)^{2/3} \left[\frac{1 + e\left(\frac{1-x}{x} \right) \frac{\rho_g}{\rho_l}}{1 + e\left(\frac{1-x}{x} \right)} \right]^{1/3} \right\}^{-1}$	Annular flow with liquid entrainment

Empirical slip models can be obtained by experimental data. Smith [28] proposed a slip model by Eq. (2.24) for the separated flow by assuming identical momentum fluxes in each phase.

$$S = e + (1-e)\left[\frac{\frac{\rho_l}{\rho_g} + e\left(\frac{1-x}{x}\right)}{1 + e\left(\frac{1-x}{x}\right)}\right]^{1/2} \tag{2.24}$$

Chisholm [29] determined the slip factor for annular flow by Eq. (2.25); this model satisfied thermodynamic limits. As the steam quality approaches zero, only tiny bubbles exist in the system. We can assume the bubbles and liquids move together since the buoyancy is negligible. Therefore, the slip factor is one. When the steam quality is equal to 1, the Chisholm model agrees with the analytical slip model Eq. (2.21). Table 2.2 summarizes void fraction models by empirical slip factors.

$$S = \left[1 - x\left(1 - \frac{\rho_l}{\rho_g}\right)\right]^{1/2} \tag{2.25}$$

Table 2.2. Summary of void fraction models using empirical slip factors.

Model	Correlation	Condition
Smith [26, 28]	$\alpha = \left[1 + 0.79\left(\frac{1-x}{x}\right)^{0.78}\left(\frac{\rho_g}{\rho_l}\right)^{0.58}\right]^{-1}$	Separated flows
Chisholm [29]	$\alpha = \left\{1 + \frac{1-x}{x}\frac{\rho_g}{\rho_l}\left[1 - x\left(1 - \frac{\rho_l}{\rho_g}\right)\right]^{1/2}\right\}^{-1}$	Annular flow

Zuber and Findlay [25] derived the void fraction based on the drift-flux model given by Eq. (2.26) where v_{gj} and C_0 are the drift velocity and distribution parameter. This expression correlates the void fraction as the function of the mass flux while the analytical void fraction model does not include this effect. Furthermore, the distribution parameter allows the Zuber-Findlay (ZF) model to account for the effect of non-uniform flow distributions. This parameter (C_0) [1] needs

to be adjusted for different system characteristics such as the pressure, geometry, and mass flow rate.

$$\alpha = \left[C_0 \left(1 + \frac{1-x}{x} \frac{\rho_g}{\rho_l} \right) + \frac{\rho_g v_{gj}}{xG} \right]^{-1} \tag{2.26}$$

Ishii [30] showed that the drift-flux model can be used independently to flow regimes. However, the model should be applied only when the drift velocity is significantly larger than the sum of the superficial velocities of the liquid and vapor [26]. Table 2.3 lists the selected drift-flux models where d_i, P_r, and σ denote the inner diameter, the reduced pressure, and surface tension.

Table 2.3. Summary of the selected drift flux models for different flow regimes.

Model	Correlation	Condition
Zuber-Findlay [25]	$C_0 = 1.13$ $v_{gj} = 1.41 \left[\frac{\sigma g (\rho_l - \rho_g)}{\rho_l^2} \right]^{0.25}$	High-pressure steam
Zuber et al. [31]	$C_0 = \begin{cases} 1 - 0.5 P_r & , \ d_i > 50 \ mm \ \& \ P_r > 0.5 \\ 1.2 & , \ P_r < 0.5 \\ 1.2 - 0.4(P_r - 0.5), & d_i < 50 \ mm \ \& \ P_r > 0.5 \\ 1.4 - 0.4 P_r & , \ \text{rectangular channels} \end{cases}$ $v_{gj} = 1.41 \left[\frac{\sigma g (\rho_l - \rho_g)}{\rho_l^2} \right]^{0.25}$	Vertical upward flow in bubbly flow regime
Zuber et al. [31]	$C_0 = 1.2$ $v_{gj} = 0.35 \left[\frac{g(\rho_l - \rho_g) d_i}{\rho_l^2} \right]^{0.5}$	Slug flow
Ishii et al. [32]	$C_0 = 1.0$ $v_{gj} = 23 \frac{\rho_l - \rho_g}{\rho_l} \left[\frac{\mu_l v_l (1-\alpha)}{\rho_g d_i} \right]$	Annular flow

Through the tables in this section, there are lots of correlations for void fraction models in each flow regime, and model form uncertainty becomes a critical issue. However, it is difficult to

quantify the source of errors when system includes flow regime transition. This difficulty arouses the interest in DDM because DDM have the potential to build models from total data that are valid for a range of flow regimes.

2.3. System simulation

This section introduces system codes which can be used to generate training data for data-driven modeling framework demonstrations. Simulation platforms are also introduced, and they can be used for demonstrations of data-driven NSTH models.

2.3.1. TRACE

The TRAC/RELAP Advanced Computational Engine (TRACE) is the USNRC state-of-the-art system code that aims at analyzing large/small break LOCAs (Loss-of-Coolant Accidents) and anticipated transients for light water reactors. It inherits and enhances the features of three codes: TRAC-P (Transient Reactor Analysis Code for PWR), TRAC-B (Transient Reactor Analysis Code for BWR), and RELAP (Reactor Excursion and Leak Analysis Program). The two-phase modeling is based on the system-level two-fluid model [1] that requires closure relations to catch the sub-grid-scale physics such as drag forces, heat transfers, and interfacial mass/energy transfers. TRACE is used to manufacture training data for demonstrations of the data-driven modeling framework.

2.3.2. Dymola

Dymola [33] is the system modeling and simulation environment using the Modelica programming language [34]. The goal of Dymola is to separate physical models and numerical solvers so that people from diverse backgrounds can tightly collaborate with each other. The

physicists can focus on model development while mathematicians can work on verification of numerical solvers.

Modelica is an equation-based, object-oriented, and multi-physics programming language for complex systems such as power, hydraulic, control, mechanical, and thermal systems. The modeling language has been widely used in the automobile industries [35] as well as in the aircraft system designs [36]. In nuclear engineering, Souyri [37] implemented an EDF (Electricité de France) 1300 MW PWR for system dynamics analysis. Modelica includes four features as follows.

- The equation-based feature ensures acausal coding that provides flexible and reusable models because the equations do not relate to the direction of data flow.
- The multi-physics modeling capability allows different physical objects to be tightly coupled such as models of control, power, thermodynamic, and mechanical systems.
- Modelica is nonproprietary and object-orientated language. When system includes multiple models, the Modelica compiler can remove redundant equations and rearrange the solution matrix for numerical solvers. Figure 2.2 shows the model translation process inside Dymola.
- The models are highly modularized and easy to maintain so that the programming language is suitable for constructing the architecture of complex physical system.

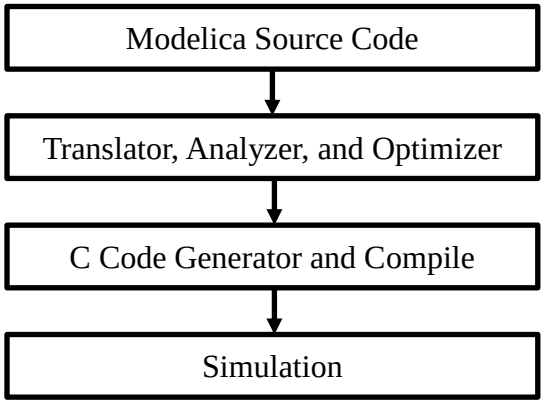


Figure 2.2. Modelica translation process.

2.3.3. OpenFOAM

OpenFOAM [38, 39] stands for open source field operation and manipulation. It includes various numerical solvers written in C++, and it has been widely used for CFD simulation. The code package is nonproprietary and allows users to customize solvers for a specific application. Therefore, we implement data-driven NSTH models in OpenFOAM solvers to evaluate performance of the proposed machine learning frameworks in the dissertation.

2.4. Machine learning for DDM of NSTH

This section provides the review of thermal fluid data which can be used to train machine learning models. Machine learning methods are also reviewed to determine the optimal algorithm for the development of data-driven NSTH models.

2.4.1. Thermal fluid data

Figure 2.3 provides an overall characterization of thermal fluid data [40] by data type, data source, and data quality. The global data are system conditions and integrated variables such as system pressure, mass flow rate, pressure drop, and total heat input. The local data are time series data at specific locations. The field data are measurements of field variables resolved in space and in time. Traditionally, experiments are a primary source of data, including so-called integral effect tests (IETs) and separate effect tests (SETs). As the name suggests, SETs and IETs are designed to investigate isolated phenomena and complex (tightly coupled) phenomena, respectively. Increasingly, appropriately validated numerical simulations become a credible source of data. This includes high-fidelity numerical simulations (e.g., DNS, and other CFD methods), as well as system-level simulation using computer models in parameter domains that are extensively calibrated and validated. It is noted that datasets vary by their quality regarding the quantity and

uncertainty. The amount of data affects the performance of inverse modeling since sufficient data can reduce the model parameter uncertainty in the domain of interest. Within a narrow context of ML for thermal fluid simulation, the data quality can be characterized by the amount of relevant and adequately evaluated data (i.e., data quantity) and associated uncertainty (including measurement uncertainty and other biases, e.g., scaling, processing).

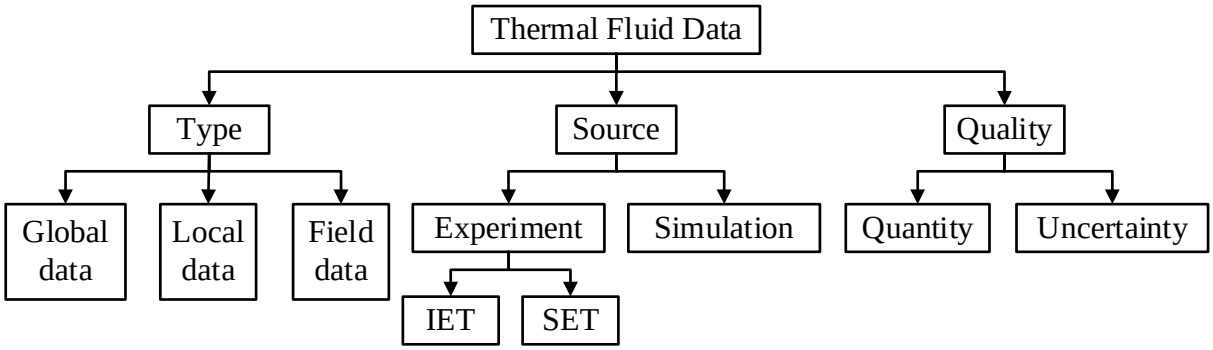


Figure 2.3. Hierarchy of thermal fluid data.

2.4.2. Machine learning (ML)

Machine learning (ML) can be used to develop closure models by learning from the available, relevant, and adequately evaluated data[2] (ARAED) with nonparametric models. While the concept of ML is not new, the past decade has witnessed a significant growth of capability and interest in machine learning thank to advances in algorithms, computing power, affordable memory, and abundance of data. There is a wide range of applications of machine learning in different areas of engineering practice. In a narrow context of the present study, machine learning is defined as the capability to create effective surrogates from a substantial amount of data obtained from measurements and simulations.

[2] In this study, assumption is made that the data required for ML are available, and their relevance and applicability has been assessed.

Figure 2.4 depicts a workflow of thermal fluid closure development using ML. The objective is to construct a function to represent the unknown model that correlates inputs and targets. Since supervised learning [41] is interested, inputs and targets are essential that can be obtained from data. The $\boldsymbol{X}$ denotes the flow feature space as inputs. The $\boldsymbol{Y}$ presents the response space as targets that are associated with flow features. The subscript k denotes the k^{th} measurement at a certain location. After collecting all relevant datasets, machine learning models (ML) are generalized by a set of nonlinear functions with hyperparameters to represent a thermal fluid closure. Based on different machine learning methods, various algorithms are employed to seek an optimal solution that allows a ML-based model to fit the observed data. Based on distinct learning purposes, Domingos [42] classified machine learning methods into five tribes including symbolists, evolutionaries, analogizers, connectionists, and Bayesians. Table 2.4 lists the five tribes in ML with their learning algorithm and challenges. Ling & Templeton [43] evaluated the predictability of various machine learning algorithms for predicting the averaged Navier-Stoke uncertainty in a high Reynolds region.

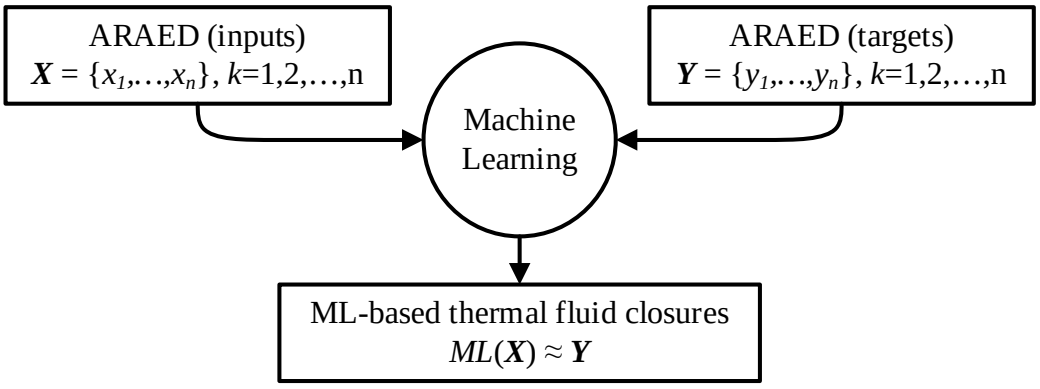


Figure 2.4. Workflow of thermal fluid closure development using machine learning.

Table 2.4. Summary of five tribes in ML and their master algorithm with applications [42].

Tribe	Problem	Master Algorithm
Symbolists	Knowledge Composition	Inverse Deduction
Evolutionaries	Structure Discovery	Genetic Programming
Analogizers	Similarity	Support Vector Machine
Connectionists	Credit Assignment	Backpropagation
Bayesians	Uncertainty	Probabilistic Inference

Deep learning (or deep neural networks) belongs to the connectionists tribe in Table 2.4. Deep neural networks can use nonparametric models to capture any measurable information [44]. Therefore, the dissertation focuses on using deep learning to develop data-driven NSTH models because deep learning provides flexible model structures that are not limited to specific model forms.

2.4.3. Deep Learning (DL)

Deep learning (DL) belongs to a branch of machine learning. A breakthrough has been made since Hinton [45] first introduced a fast algorithm to train neural networks (NNs) with multilayer perceptrons. In general, any NN with more than two layers is referred to as deep learning [46]. Deep neural networks contain numerous hyperparameters and adaptive model forms to achieve pattern recognition or regression for complex datasets. Hornik [44] showed that multilayer NNs are compatible with the universal approximation theorem. There is no theoretical limit for multilayer NNs to capture the properties of any measurable information. Notably, hierarchical structures of deep learning deem appropriate for describing complex models that involve multiple scales.

Deep neural networks include lots of variations such as feedforward neural networks (FNNs) [44], convolutional neural networks (CNNs) [47], recurrent neural networks (RNNs) [48], self-organizing map [49], and Boltzmann machine [50]. Table 2.5 summarizes the type of each neural network and their relevant problem domains [46]. Table 2.6 defines the acronyms. The

present work focuses on applying FNNs and CNNs to achieve data-driven modeling because building models from data belongs to regression problems.

Table 2.5. Neural networks and its applicable problem domains (adopted after Heaton [46]).

	Clust	Regis	Classif	Predict	Robot	Vision	Optim
Feedforward		✓✓✓	✓✓✓	✓✓	✓✓	✓✓	
Convolutional Network		✓	✓✓✓		✓✓✓	✓✓✓	
Recurrent Network		✓✓	✓✓	✓✓✓	✓✓	✓	
Self-organizing Map	✓✓✓				✓	✓	
Boltzmann Machine			✓				✓✓

Table 2.6. Acronyms of the problem domains in Table 2.5.

Acronyms	Meaning
Clust	Unsupervised clustering problems
Regis	Regression problems
Classif	Classification problems
Predict	Prediction problems
Robot	Robotics, using sensors and motor control
Vision	Computer vision problems
Optim	Optimization problems

2.4.3.1. Feedforward neural networks

Feedforward neural networks (FNNs) belong to supervised learning, and they require inputs and targets from data during the training. Figure 2.5 depicts a structure of a typical three-layer FNN with one input layer, two hidden layers (HLs), one output layer, and three hidden units (HUs) in each hidden layer. Data flow goes straight from the input layer to the output layer. Neural networks can take arbitrary inputs and outputs and build a correlation based on data. The layer of neural networks is counted by the summation of numbers of hidden layers and the output layer. For instance, Figure 2.5 shows a structure of a three-layer neural networks.

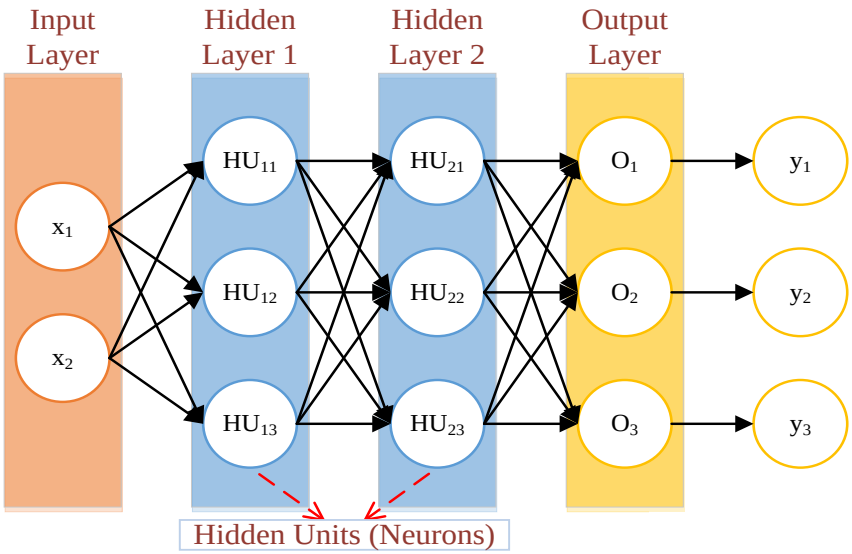


Figure 2.5. A three-layer neural network.

Eq. (2.27)-Eq. (2.29) show the mathematic formulation of feedforward neural networks. Eq. (2.27) gives the input vector (**x**) with total elements equal to *n*. Eq. (2.28) shows the model inside HU where *i* and *j* are the ith number of inputs, and jth number of hidden units. Furthermore, we denote weights and biases by **w** and *b* for each hidden unit and use σ to represent nonlinear activation functions. Finally, Eq. (2.29) gives the output ($\hat{y}$) of neural networks where *m* denotes the mth output layer (*o*).

$$\mathbf{x} = [x_1, x_2, \ldots, x_n] \tag{2.27}$$

$$HU_j(\mathbf{x}) = \sigma\left(\sum_{i=1}^{n} w_{ji} x_i + b_j\right) \tag{2.28}$$

$$\hat{y}(\mathbf{HU}) = \sum_{i=1}^{m} w_{oji} HU_i + b_o \tag{2.29}$$

Eq. (2.30) defines a loss function (Euclidean loss) that can be used to optimize parameters of neural networks. For regression problems, the L2 squared norm (or the so-called Euclidean loss) is commonly used. In Eq. (2.30), *N* and *y* are the total data and training target.

$$L = \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2 \tag{2.30}$$

Overfitting is a critical issue that models fit training datasets well, but they show no predictability. To prevent overfitting, we can define a loss function using L2 regularization by Eq. (2.31) with the regularization strength (λ), weights (w_i), and total weights (W). The goal of training is to find values of **w** such that the model is consistent with data.

$$L^* = L + \frac{\lambda}{2}\sum_{i=1}^{W} w_i^2 \tag{2.31}$$

2.4.3.2. Convolutional neural networks

Convolutional neural networks (CNNs) [47] include convolutional layers to reduce model parameters, and they are efficient in training. The activation function, rectified linear unit (ReLU) [51], can accelerate the training of CNNs. The input is a matrix, and the output can be either a matrix or one-hot vector [52]. Figure 2.6 depicts an architecture of CNNs. The model includes three convolutional layers and three fully connected layers. The input and output are both field data. After the first convolutional layer, eight feature maps are generated, and each feature map detects the patterns from the temperature field data. The second convolutional layer takes the inputs from the previous layer, and it outputs 12 feature maps. The third convolutional layer receives the inputs from the previous layer, and it delivers 24 feature maps to the fully connected layer. After fully connected layers, we can obtain the output from CNNs.

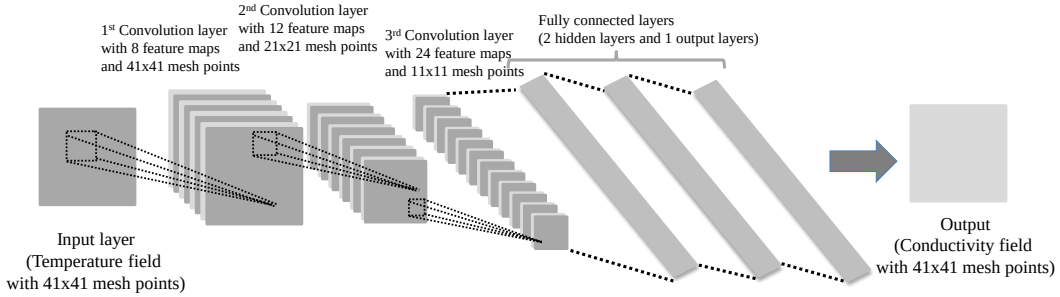


Figure 2.6. Architecture of CNN-based conductivity model (adopted after LeCun) [47].

2.4.3.3. Activation functions for neural networks

Activation functions are non-linear regression functions inside hidden units. Table 2.7 gives activation factions that are widely used.

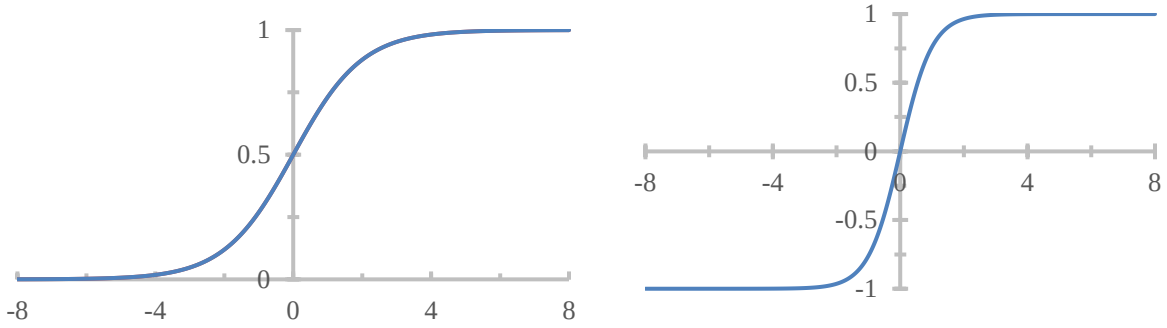
Table 2.7. Common activation functions in a neural network.

Activation Function	Formula
Sigmoid	$f(x) = \left(1 + e^{-x}\right)^{-1}$
Hyperbolic Tangent (Tanh)	$f(x) = 2 \cdot sigmoid\left(2x\right) - 1$
Rectified Liner Unit (ReLU)	$f(x) = \max\left(0, x\right)$

The sigmoid non-linearity can constrain outputs to return values between zero and one. Eq. (2.32) gives the derivative of the sigmoidal function (σ), and the derivative is easy for implementation.

$$\frac{d\sigma}{dx} = \sigma(x)\left[1 - \sigma(x)\right] \tag{2.32}$$

Figure 2.7(a) depicts the plot for the sigmoid non-linearity. Both functions saturate at tails, and this property can cause the gradient vanishing issue when training neural networks. When the gradient is equal to zero, weights and biases cannot be updated. Therefore, parameter initialization needs to be careful while using the sigmoidal activation.



(a) (b)

Figure 2.7. Non-linearity of (a) the sigmoid, and (b) tanh.

Another drawback of the sigmoidal function is that outputs are not zero-centered [53], which may result in zig-zagging dynamics for calculation of weights and biases during training.

The hyperbolic tangent function (σ) can constrain outputs to return values between negative and positive one. It is also easy to implement into the learning algorithm because its derivative can be represented by itself as given in Eq. (2.33). It also has the issue of saturation as the sigmoid non-linearity shown in Figure 2.7(b). However, the hyperbolic tangent function is easier to train than the training of the sigmoidal function because the gradient of *tanh* is larger than the sigmoid as depicted by Figure 2.8. In the meanwhile, the output of tanh is zero-centered. Therefore, the activation function, *tanh*, is preferred over the sigmoid.

$$\frac{d\sigma}{dx} = 1 - \sigma^2(x) \tag{2.33}$$

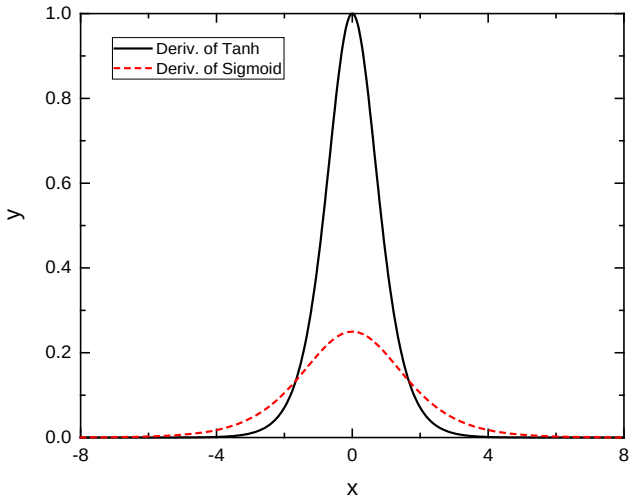


Figure 2.8. The comparison of derivatives of the tanh and sigmoid.

The rectified linear unit (ReLU) [54] has become popular because it helps learning algorithms to converge faster than the previous two activation functions [55]. It can avoid the vanishing gradient issue. The drawback of ReLU is that some hidden units may never activate if

gradients are negative. When the learning rate of learning algorithms is inappropriate (too high), hidden units may not activate during the training [53].

2.4.3.4. Backpropagation algorithm

The backpropagation algorithm [56] can be used to find weights and biases for neural networks. Figure 2.9 depicts a simple neural network for explanation of the backpropagation algorithm where σ and o denote the sigmoidal function and outputs of each hidden layer. We use s_1 and s_2 to present the inputs of the first and second hidden layers respectively. Finally, the model output ($\hat{y}_i$) can be expressed by $w_3 o_2$. Eq. (2.34) gives the loss function of the neural network in Figure 2.9 where y_i denotes the training data.

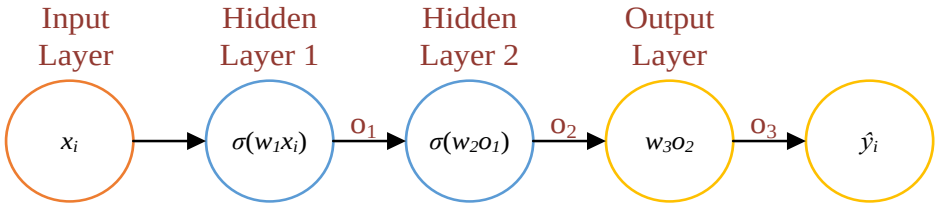


Figure 2.9. The three-layer NN with single HU in each layer.

$$L = \frac{1}{2}\left(\hat{y}_i - y_i\right)^2 = \frac{1}{2}\left\{w_3\sigma\left[w_2\sigma\left(w_1 x_i\right)\right] - y_i\right\}^2 \tag{2.34}$$

Eq. (2.35) calculates the derivative of Eq. (2.34) that propagates the error ($\hat{y} - y$) to the output layer.

$$\frac{\partial L}{\partial w_3} = \left\{w_3\sigma\left[w_2\sigma\left(w_1 x_i\right)\right] - y_i\right\}\sigma\left[w_2\sigma\left(w_1 x_i\right)\right] = \left(\hat{y}_i - y_i\right)o_2 \tag{2.35}$$

By using the chain rule, Eq. (2.36) shows how we can propagate the error to the second hidden layer. Eq. (2.37) shows the error propagation to the first hidden layer. After we obtain the error for each hidden layer, the stochastic gradient decent (SGD) [57] method is commonly used

to update the weights. Eq. (2.38) gives the formula for SGD where η is the step size or learning rate in machine learning.

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial o_3}\frac{\partial o_3}{\partial o_2}\frac{\partial o_2}{\partial s_2}\frac{\partial s_2}{\partial w_2} = \left(\hat{y}_i - y_i\right) w_3 \sigma\left(w_2 o_1\right)\left[1-\sigma\left(w_2 o_1\right)\right]\sigma\left(w_1 x_i\right) \tag{2.36}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o_3}\frac{\partial o_3}{\partial o_2}\frac{\partial o_2}{\partial s_2}\frac{\partial s_2}{\partial o_1}\frac{\partial o_1}{\partial s_1}\frac{\partial s_1}{\partial w_1} = \left(\hat{y}_i - y_i\right) w_3 \sigma\left(w_2 o_1\right)\left[1-\sigma\left(w_2 o_1\right)\right] w_2 \sigma\left(w_1 x_i\right)\left[1-\sigma\left(w_1 x_i\right)\right] x_i \tag{2.37}$$

$$w^{n+1} = w^n - \eta \frac{\partial L}{\partial w^n} \tag{2.38}$$

2.4.3.5. Uncertainty quantification for DL

"*Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful*," George E.P. Box, page 74 of [58].

The goal of uncertainty quantification is to evaluate and minimize uncertainties associated with experiments and models. Since models usually include assumptions, reality may not be fully represented by models. Therefore, it is important to know how uncertain models are, so we can have confidence to apply models to predict quantities of interest. Most importantly, we are interested in understanding how the uncertainty propagates through models in system simulation and how it affects the modeling results. The sources of uncertainty can be classified in to four categories as follows.

Data uncertainty. It can be caused by noisy data and measurement errors that are not directly related to models.

Model parameter uncertainty. Models usually include tunable parameters to fit data. However, for deep learning models, the model parameters are weights and biases that are determined by optimizers based on data.

Model form uncertainty. Inferring models from data is an inverse process. Solutions can be infinite and result in lots of different model forms.

Numerical uncertainty. When we run computer simulations, we solve partial differential equations (PDEs) in discretized forms. The selection of mesh sizes can affect the accuracy of results, and an optimal mesh size is problem-dependent. Roundoff error is another source of uncertainty for numerical simulation. In the meanwhile, some deep learning frameworks calculate the gradient numerically, and some frameworks compute the gradient analytically. The difference in calculating gradient can induce uncertainty for deep learning models.

The model parameter uncertainty and model form uncertainty of deep learning can be considered together as model uncertainty which can be estimated by neural networks with dropout [59]. Dropout [59] is the stochastic regularization technique that was originally used to avoid over-fitting in deep learning. Recent research [60] proves that neural networks with dropout are Bayesian neural networks.

2.4.3.6. Comparison of deep learning frameworks

There are lots of deep learning frameworks with different programming languages on the market. All major deep learning frameworks run on GPU to leverage computing power. We review the following deep learning frameworks and select a tool to implement data-driven NSTH models in this study.

Tensorflow

Tensorflow [57] is maintained by Google, and it can run on heterogeneous systems ranging from single GPU to multiple GPUs. It provides a wide variety of algorithms to train deep neural networks, and it is widely used in the fields of natural language processing and computational drug discovery. Tensorflow also allows users to define functions in tensor forms that support indexing,

slicing, cloning, and resizing, and then Tensorflow automatically computes the derivatives of those functions. Therefore, users can deploy customized models including PDEs. It provides Python and C++ APIs that allow users to connect deep learning models with existing statistical or scientific libraries. Tensorflow also contains a graphic system to visualize the source tree of deep learning models.

Theano

Theano [61] is an academic project, and Tensorflow is inspired by this project. The Theano framework includes wide varieties of algorithms to train deep neural networks, and it can do tensor calculations. The main difference between Theano and other tools is that Theano computes the analytical solutions of derivatives. Theano is expected to construct accurate models based on this feature.

Torch

Torch [62] is initially developed at New York University, and it supports tensor computation. Torch includes machine learning libraries, and it uses a programming language, Lua [63], with underlying C implementation. Facebook AI Research [64] maintains a deep learning module for Torch.

Caffe

Caffe [65] is a deep learning framework which is developed at Berkeley Vision and Learning Center, and it is recognized for offering "Model Zoo" that is a repository including pre-trained deep neural networks. This feature makes it attractive to industries, especially for computer vision companies. Caffe provides configuration files to allow users to assemble the existing models for specific applications.

Microsoft Cognitive Toolkit (MCT)

Microsoft Cognitive Toolkit (MCT) [66] is the deep learning framework from Microsoft, previously known as Computational Network Toolkit. The goal of MCT is to provide a fast and easy configurable machine learning system for developers to write less code. Therefore, MCT provides configuration files for users to assemble ML models. It is possible to embed MCT into a python or C++ code to enable the deep learning capability.

The above deep learning frameworks can be classified into two categories: configuration file and programmatic generation. Frameworks, which belong to the configuration file category, only allow users to specify a configuration file to use deep neural networks. Therefore, those tools are not preferable in this work since we require the capability of programming customized models.

On the other hand, deep learning frameworks in programmatic generation category let users implement models, and then those tools provide necessary algorithms to train deep neural networks. With this feature, we can implement data-driven NSTH models. Table 2.8 summarizes deep learning frameworks into two categories. We decided to work with Tensorflow because it provides Python and C++ APIs that are flexible for code coupling.

Table 2.8. Two categories of deep learning packages.

Configuration File	Programmatic Generation
Caffe MCT	Tensorflow Theano Torch

2.5. Contemporary works of using ML methodologies in thermal fluid simulation

Insofar, based on a best-effort review of the literature, contemporary work can be grouped into two distinct strategies for employing machine learning in the field of NSTH simulation, namely in algorithm implementation and physics identification. The first approach assumes the physics of fluids is known and applies machine learning to improve solution performance. For

example, Tompson *et al.* [67] utilized convolutional neural networks (CNNs) to accelerate Eulerian fluid simulations. Ladický *et al.* [68] applied regression forests to accelerate smoothed particle hydrodynamics simulations. The other strategy employs machine learning methodologies to a body of data to capture underlying correlations including recognizing governing equations. Brunton, Proctor & Kutz [69] utilized sparse regression with time-series data to recover the Navier-Stokes equations. Not in the thermal fluid domain, but of relevance is the work of Mills, Spanner & Tamblyn [70]. They showed that CNNs with millions of training data recovered the effective form of the Schrödinger equation.

Aside from extracting governing equations from data, ML has been applied to construct surrogates of closure relations. Limited work in this direction relies on supervised learning with the training data from either DNS or large eddy simulation (LES). The applications include both single-phase and two-phase flow problems. Ma, Lu & Tryggvason [71-73] utilized artificial neural networks (ANNs) to obtain closures for stream stress and surface tension force. They implemented ANN-based closures in a two-fluid model for simulation of isothermal bubbly flow in a vertical box channel.

Parish & Duraisamy [74] proposed the field inversion and machine learning framework that used a Gaussian process to assimilate data. They demonstrated that source terms of the heat conduction equation could be inferred from data to reconstruct spatial temperature profiles. The FIML was also applied to the k-ω turbulence model [24] for assimilating DNS data to reduce model form errors. The modified turbulence model was used to improve Reynolds-averaged Navier-Stokes (RANS) equations for simulating a single-phase planar channel flow. In a more recent work, Zhang & Duraisamy [18] replaced Gaussian functions by feedforward neural networks (FNNs) to enable spatiotemporal modifications. Tracy, Duraisamy & Alonso [75]

utilized an FNN-based closure to learn the results from the Spalart–Allmaras [21] turbulence model.

Wu *et al.* [76] and Wang *et al.* [77, 78] used random forest regression to build a discrepancy field of Reynold stress between DNS and RANS simulation results. Then they used a modified Reynolds stress field to improve the prediction of the RANS model for test flows. Ling, Kurzawski & Templeton [19] trained Reynolds stress closures by tensor basis deep neural networks with DNS and LES data. The inputs of deep neural networks were the mean strain and rotation rate tensors obtained from RANS simulation using eddy viscosity models. The results indicated that deep neural networks improved RANS simulation for flows in different geometries with various Reynolds numbers.

The current reference works demonstrate several ways to implement ML-based models in conservation equations. However, those works do not discuss the implementation from perspectives of thermal fluid data that include data type, data source, and data quality. Assumptions behind experiments affect the selection of appropriate frameworks. This work includes a classification system that provides a comprehensive overview of using machine learning to maximize predictive capabilities of NSTH simulation. Such a classification system allows machine learning frameworks to become transparent regarding assumptions, workflows, and knowledge and data requirements. Machine learning frameworks are solution algorithms to achieve data-driven modeling of NSTH. With a classification system, we can select appropriate frameworks based on the available data and knowledge.

2.6. Summary

In this chapter, we reviewed the essential elements of the data-driven modeling framework including thermal-hydraulics models, simulation platforms, and machine learning, especially deep learning. Closure relations are essential to close conservation equations. The traditional approach of developing closure models requires lots of efforts that involve decades of work. The models are usually limited to certain flow regimes or conditions. Deep learning potentially provides a flexible way to construct a surrogate directly from data. With sufficient training data, deep learning has the potential to figure out underlying correlations behind data. Simulation platforms such as Dymola and OpenFOAM include lots of numerical solvers that allow us to quickly deploy and test NSTH models for prediction. Based on data and knowledge requirements, we propose five different frameworks to embed DL-based closure relations in NSTH simulation in CHAPTER 3. The approach relaxes the structure of nuclear system codes and allows the codes to assimilate newly observed data that can potentially reduce uncertainty in prediction.

CHAPTER 3. FORMULATION OF THE FRAMEWORK

3.1. Introduction of data-driven frameworks for closure development

Data-driven modeling (DDM) has been previously employed to develop NSTH models. However, the traditional approach requires extensive efforts to gain insights and mechanistic understanding through analysis of data to represent data in a compact form. Figure 3.1 depicts the traditional framework for developing closure models. Starting from the knowledge base, we can design experiments to obtain relevant data, perform data analysis and research to derive sought-after closure models for thermal fluid simulation. This approach may take years to decades. The long time needed for developing new closure relations limits the pace of model applications while dealing with newly designed systems, including new geometries and new coolants. The obtained closure models are implemented into thermal fluid simulation for applications and assessments. Once the closure models are tested and evaluated, these models are stored in the knowledge base.

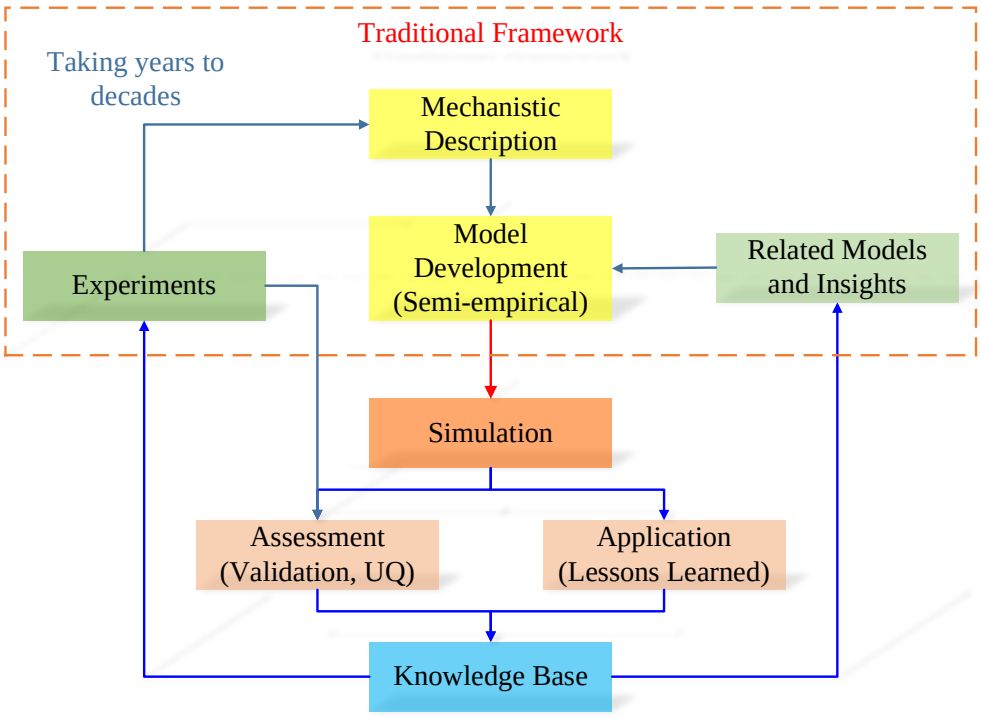


Figure 3.1. Traditional framework for developing sub-grid-scale (SGS) physics models.

As an alternative to the traditional framework, Figure 3.2 depicts the data-driven modeling framework with machine learning. After data are collected from experiments, we can use machine learning to figure out underlying correlations behind data. The use of machine learning has the potential to shorten the model development phase. For instance, deep learning uses non-parametric models to capture trends of data such that DL-based models are not limited to fixed model forms. Based on different approaches to use machine learning in NSTH simulation, a system is established in Section 3.2 to classify machine learning frameworks which can accomplish data-driven modeling of NSTH.

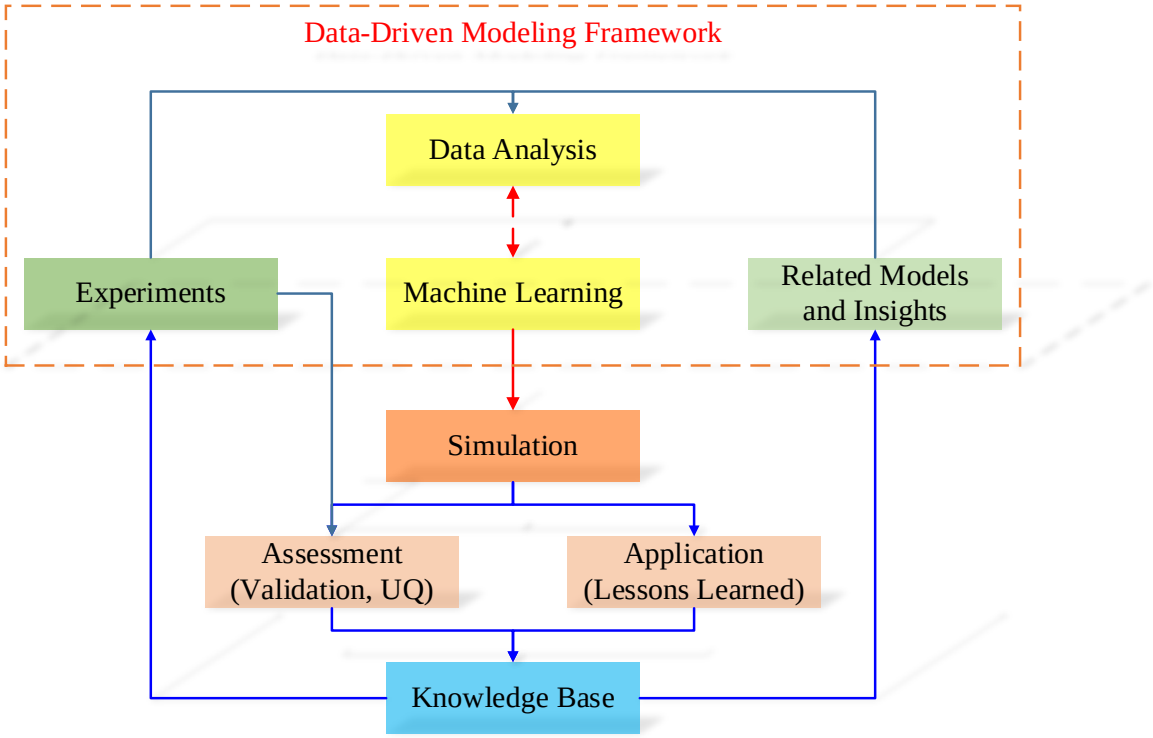


Figure 3.2. The data-driven modeling framework for system simulations.

3.2. Classification of machine learning in NSTH

Machine learning strategies employed in NSTH simulation include various frameworks to leverage values of data from advanced validation experiments and high-fidelity numerical

simulations such as DNS, LES, or RANS. Several recent studies aim at closing the mass-momentum-energy conservation equation by ML-based closures, while others on extracting governing equations from data. All frameworks have the goal to represent underlying correlations behind data and to capture data in compact forms for simulation. Based on distinct strategies of incorporating machine learning into NSTH simulation, we propose a classification into five frameworks including physics-separated ML (PSML or Type I ML), physics-evaluated ML (PEML or Type II ML), physics-integrated ML (PIML or Type III ML), physics-recovered (PRML or Type IV ML), and physics-discovered ML (PDML or Type V ML).

Type I ML is physics-separated because it requires the separation of scales [14, 79]. Closure models are independently built upon data, and then they are implemented in conservation equations. Type II ML is physics-evaluated. The framework includes simulation based on prior knowledge. When discrepancies occur between observations and simulation, the observed data become references to inform simulation to achieve data-model consistency. Type III ML is physics-integrated since there is no need to separate scales. Instead, ML-based closure models are embedded and trained in conservation equations. Type IV ML is physics-recovered because it aims at recovering the form of governing equations from data. Type V ML is physics-discovered. It is end-to-end ML that ultimately relies on learning algorithms to figure out hidden physics from a considerable amount of data.

Machine learning frameworks are solution algorithms to allow NSTH simulation to leverage the value of data. Figure 3.3 depicts the hierarchy of machine learning frameworks based on the goal structuring notation (GSN) [80, 81]. GSN can present the logic of argumentation by a graphic notation. Figure 3.4 gives definitions of principal components defined by GSN. Figure 3.3 shows that the top goal (G_{top}) is to maximize predictive capabilities of NSTH simulation by

machine learning. Then there are three sub-goals following by the top goal about how to achieve the top goal. First, machine learning can assimilate data to construct closure relations. Based on different data sources and assumptions, solution algorithms to the first sub-goal (G_1) can be found by Type I, Type II, and Type III ML. Second, since a system can be nonlinear and include multiscale dynamics, we may not want to assume governing equations are known. Instead, we rely on ML to recover the form of equations by assuming that a thermal fluid process can be effectively captured by a PDE model. Type IV ML is the solution to the second sub-goal (G_2). Third, if data are immense enough, ML is expected to discover hidden physics directly through data. Type V ML is the solution to the third sub-goal (G3). It is noted that the first two sub-goals lead to solutions that converge to conservation equations. Section 3.2.2-3.2.6 introduce each framework in detail.

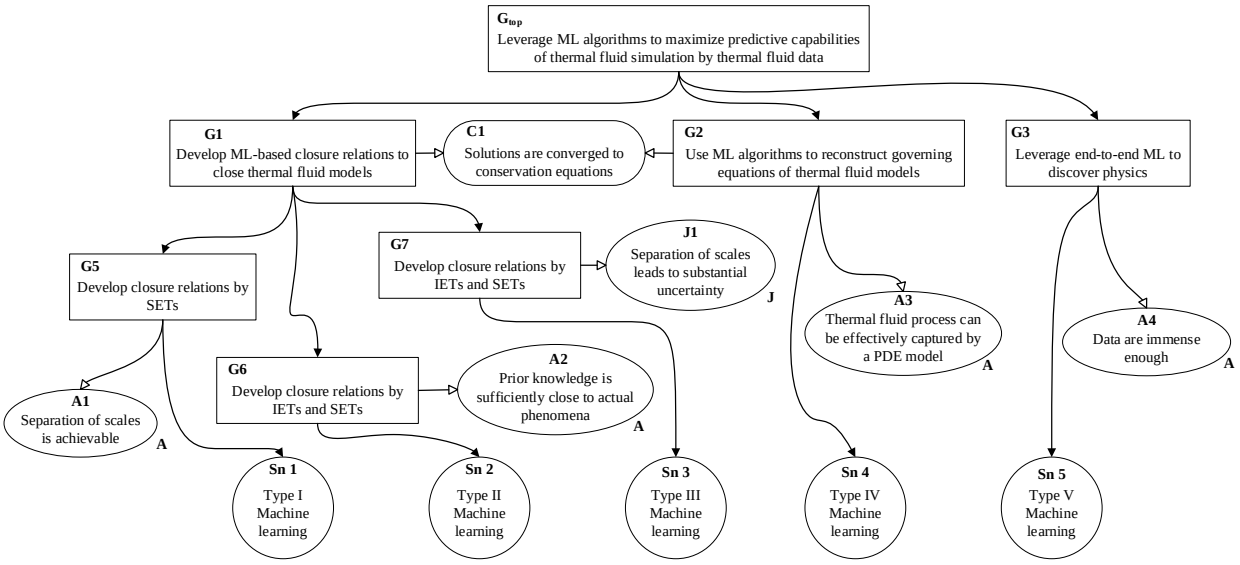


Figure 3.3. Hierarchy of machine learning (ML) frameworks for thermal fluid simulation.

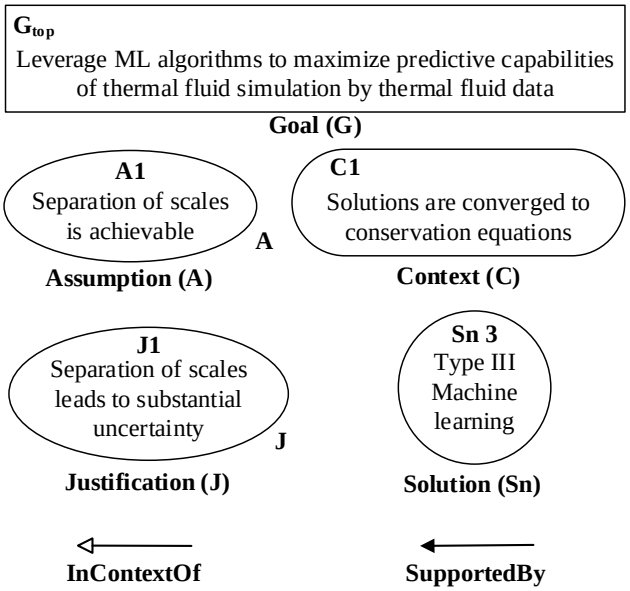


Figure 3.4. Principal components of the ML framework hierarchy using the notation by GSN.

3.2.1. Criteria for classifying ML frameworks for thermal fluid simulation

Each framework has its distinct goal and approach to leverage data. Since we classify five frameworks, we build the classification system based on four conditions. First, we examine whether solutions are converged meaning that solutions conserve the mass-momentum-energy balance in a control volume. Second, we check if the framework focuses on developing fluid closures. Third, we distinguish Type III ML from other frameworks because it inherently ensures data-model consistency. Finally, the last condition is about the separation of scales. Accounting for all four conditions, we categorize five distinct types of machine learning frameworks for NSTH simulation based on the following four criteria:

Criterion 1: Is a PDE involved in thermal fluid simulation?

The first criterion examines whether a conservation equation (partial differential equation, PDE) is involved in thermal fluid simulation. Type V ML relies on machine learning to discover the underlying physics directly from data and to deliver equivalent surrogates of governing equations. Type V ML is an extreme case when there is no prior knowledge, and we must purely

depend on the observed data. By this criterion, we can distinguish Type V ML from the other four ML frameworks.

Criterion 2: Is the form of PDE models given?

The second criterion inspects if the form of conservation models (partial differential equation models) is known. Type IV ML does not introduce biases in selecting physics models; instead, it recovers the exact form of conservation models based on data. Therefore, we can distinguish Type IV ML from Type-I, Type II, and Type III ML.

Criterion 3: Is a PDE involved in the training of closure relations?

A partial differential equation is involved in Type I, Type II, and Type III ML. Therefore, the goal is to develop closure models in nonparametric forms to close conservation equations. Criterion 3 checks whether conservation equations are involved in the training of ML-based closures. Traditionally, the assumptions [82, 83] of scale separation and physics decomposition are essential to develop closure models. The former allows us to set up SETs for various scales while the latter decomposes closure relations into different physics within the same scale. However, in many thermal fluid processes, the physics (physical mechanisms) is tightly coupled. Type III ML avoids these two assumptions by training closure models that are embedded in PDEs. By this criterion, we can distinguish Type III ML from Type I and Type II ML.

Criterion 4: Is a scale separation assumption required for model development?

This criterion tests whether the model development requires the separation of scales. This hypothesis isolates closure relations from conservation equations so that the models can be separately built and calibrated by SETs. The scale separation is essential for Type I ML because it only relies on data to construct closure models. However, the data by SETs may have been distorted, while IETs are designed to capture (a selected set of) multi-physics phenomena.

Table 3.1 summarizes the criteria to classify the five distinct types of ML frameworks for NSTH simulation.

Table 3.1. Criteria for the ML framework classification.

Classification Criteria	Type I ML (PSML)	Type II ML (PEML)	Type III ML (PIML)	Type IV ML (PRML)	Type V ML (PDML)
1. Is a PDE involved in thermal fluid simulation?	Yes	Yes	Yes	Yes	No
2. Is the form of PDE models given?	Yes	Yes	Yes	No	No
3. Is a PDE involved in the training of closure relations?	No	No	Yes	No	No
4. Is a scale separation assumption required for the model development?	Yes	No	No	No	No

3.2.2. Type I machine learning, physics-separated machine learning (PSML)

Type I ML or so-called physics-separated ML (PSML) aims at developing closure models by using SET data. Type I ML assumes that conservation equations and closure relations are scale separable, for which the models are local. Type I ML requires a thorough understanding of the system so that SETs can be designed to support model developments. Figure 3.5 depicts the hierarchical decomposition of system simulation that allows physics models to be scale separable. The system can be divided into various sub-systems such as a reactor core, steam generator, reactor coolant system, and emergency core cooling system. The foundations of those sub-systems are multiphase models that require closure relations based on sub-grid-scale physics. Figure 3.6 illustrates the workflow about how we can obtain closure models to close conservation equations where the models are separately developed by using SET data. Therefore, we can apply ML-based closures to assimilate data to achieve data-driven thermal fluid simulation.

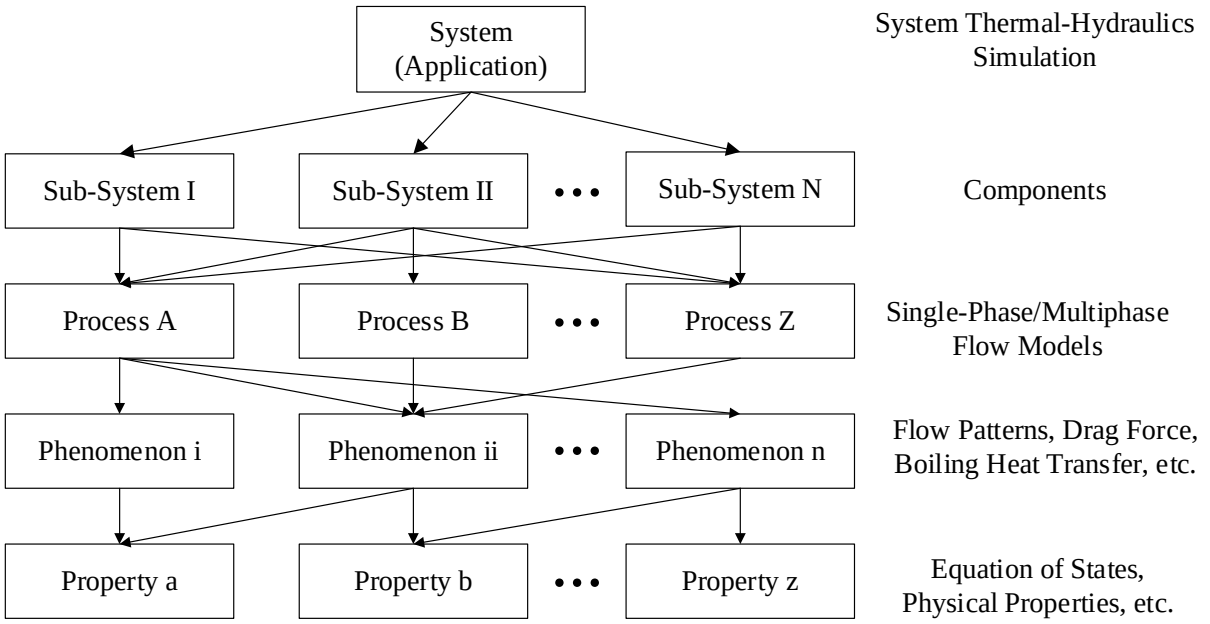


Figure 3.5. Hierarchical decomposition of system thermal-hydraulics simulation.

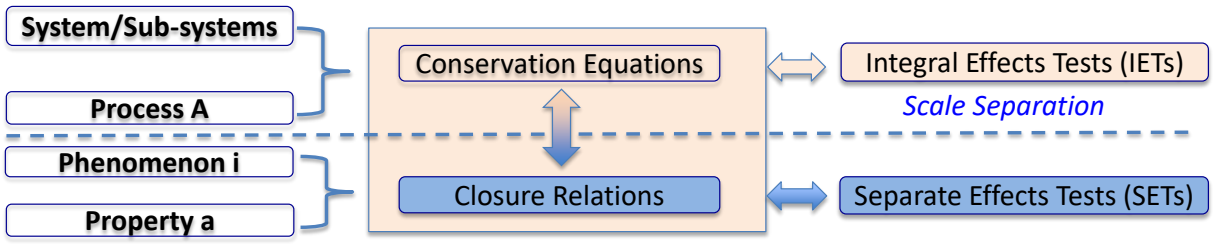


Figure 3.6. Closure development requires a scale separation assumption.

Figure 3.7 depicts the architecture of Type I ML framework, and it is forward data-driven modeling. The procedure includes the following elements:

Element 1. Assume a scale separation is achievable such that closure models can be built from SETs. From either high-fidelity simulations or experiments, collect training data, (x_k, y_k).

Element 2. Preprocess data from element 1 to ensure that data from multi-sources have the same dimension and manipulation such as the selection of averaging methods.

Additionally, consider normalizing data so that we can approximately equalize the importance for each data source. For large datasets, employing principal component analysis [84] can be helpful to reduce the dimension of data.

Element 3. Compute flow features or system characteristics, $\boldsymbol{X}$, as training inputs for element 5.

Element 4. Calculate the corresponding outputs ($\boldsymbol{Y}$) of the desired closures from data as training targets that can supervise ML algorithms to learn from data.

Element 5. Utilize ML algorithms to build a correlation between inputs and targets. After the training, output the ML-based closure model, $ML(\boldsymbol{X})$, to element 6.

Element 6. Constrain the ML-based closure, $g(ML(\boldsymbol{X}))$, to satisfy model assumptions and to ensure the smoothness of model outputs since it needs to be solved with PDEs. It is noted that this element is not essential if assumptions are not applicable.

Element 7. Implement the ML-based closure into conservation equations, and solve PDEs for predictions with the embedded ML-based closure that is iteratively queried.

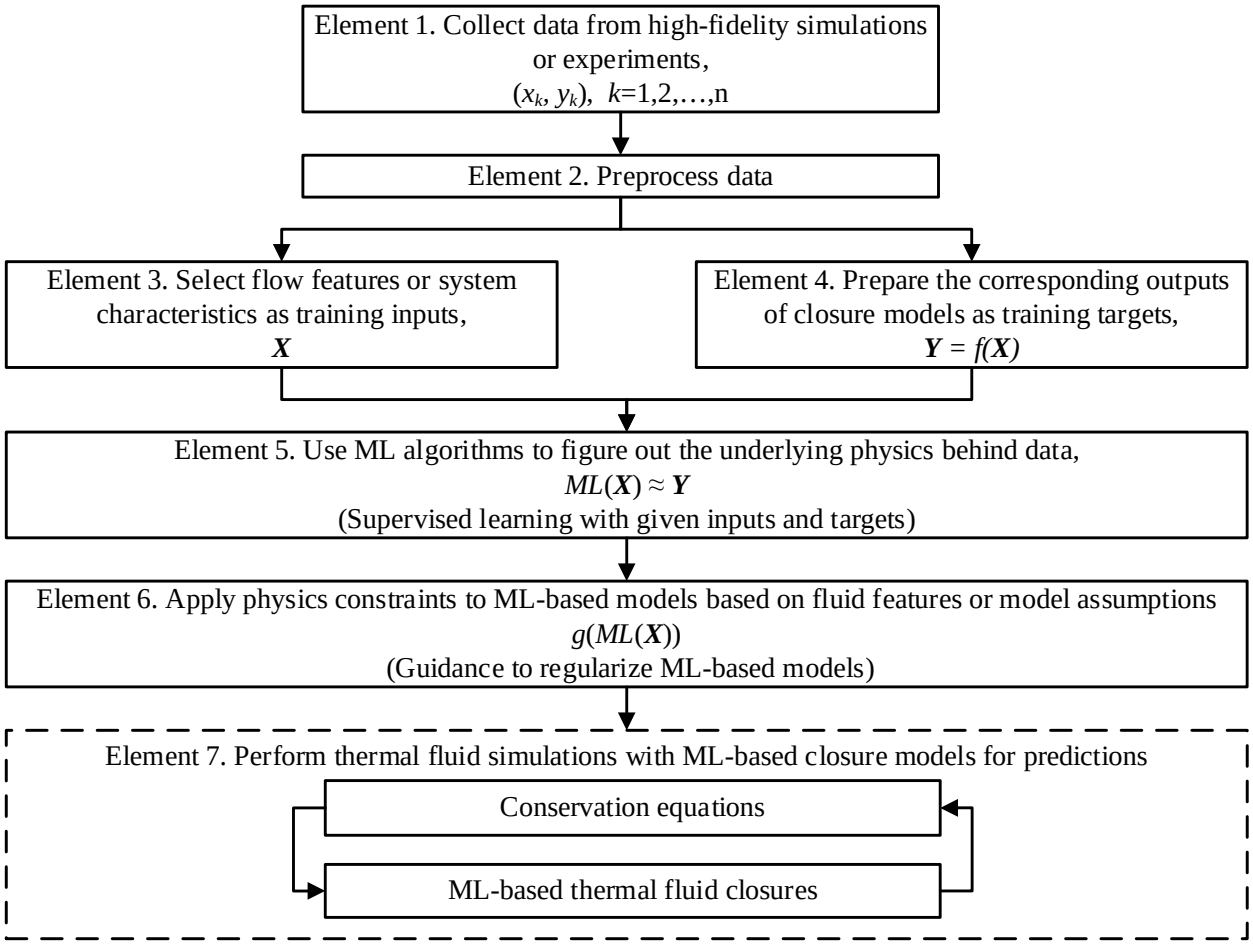


Figure 3.7. Overview of Type I ML framework with a scale separation assumption.

Type I ML satisfies the criteria from Table 3.1 except the third criterion. The quality of SET data largely controls the performance of closure models obtained by Type I ML. While the experimental uncertainty in each SET may be controlled and reduced, the process uncertainty (dominated by design assumptions) is irreducible. We note that PDEs and closure relations are decoupled in Type I ML. It can cause model biases between conservation equations and closure relations. It is noted that inferring model parameters from data belong to inverse problems which are ill-posed [85]. For ML models, a small change in inputs can result in large uncertainty in outputs. While implementing ML-based closures in PDEs, the uncertainty can lead to a discontinuity that fails numerical simulation. For more practices related to Type I ML, readers are

referred to Ma *et al.* [71-73], Parish & Duraisamy [74], Zhang & Duraisamy [18], Tracy *et al.* [75, 86], Singh & Duraisamy [87], and Chang & Dinh [88, 89].

3.2.3. Type II machine learning, physics-evaluated machine learning (PEML)

Type II ML or so-called physics-evaluated machine learning (PEML) focuses on reducing the uncertainty for conservation equations. It requires prior knowledge on selecting closure models to predict thermal fluid behaviors. Type II ML utilizes high-fidelity data to inform low-fidelity simulation. Comparing to high-fidelity models, ROMs can efficiently solve engineering design problems within an affordable time frame. However, ROMs may produce significant uncertainty in predictions. Type II ML can improve the uncertainty of low-fidelity simulation by reference data. Since the physics of thermal fluids is nonlinear, ML algorithms are employed to capture the underlying correlation behind high-dimensional data. The framework requires training inputs such as flow features that represent the mean flow properties. Training targets are the responses that correspond to input flow features.

Figure 3.8 depicts the framework of Type II ML, and it includes the following procedures:

Element 1. Perform low-fidelity simulation ($\boldsymbol{\Psi_L}$) to generate data for calculating input flow features.

Element 2. Perform high-fidelity simulation ($\boldsymbol{\Psi_H}$) with identical system characteristics in element 1. High-fidelity data are used to compute targets in element 5.

Element 3. Average high-fidelity data to match the dimension of low-fidelity data. The averaging method should preserve the consistency between high-fidelity and low-fidelity simulations. Additionally, normalizing data can accelerate the training of ML. For large datasets, principal component analysis [84] can reduce the dimension of data.

Element 4. Calculate flow features, $\boldsymbol{X}(\boldsymbol{\Psi_L})$, as training inputs to element 6.

Element 5. Compute targets, $f(\boldsymbol{X}(\boldsymbol{\Psi}_H))$, as the responses to input flow features by high-fidelity data. Targets can also be discrepancy/error, $\boldsymbol{\varepsilon}(\boldsymbol{\Psi}_H, \boldsymbol{\Psi}_L)$, between high-fidelity and low-fidelity data.

Element 6. Use an ML algorithm to represent the underlying correlation between flow features and discrepancy/error of flow properties. After the training, output an ML-based discrepancy/error model, $ML(\boldsymbol{X}(\boldsymbol{\Psi}_L))$, to element 8.

Element 7. Execute new low-fidelity simulation ($\boldsymbol{\Psi'}_L$) under predicting conditions. Then use the solution to obtain flow features as inputs to element 8.

Element 8. Use flow features from element 7 as inputs to query values from the ML-based model, $ML(\boldsymbol{X}(\boldsymbol{\Psi'}_L))$. Output values of a fluid closure in a fixed field to element 9.

Element 9. Implement the results from element 8 in the low-fidelity model ($\boldsymbol{\Psi}_L$) for predictions.

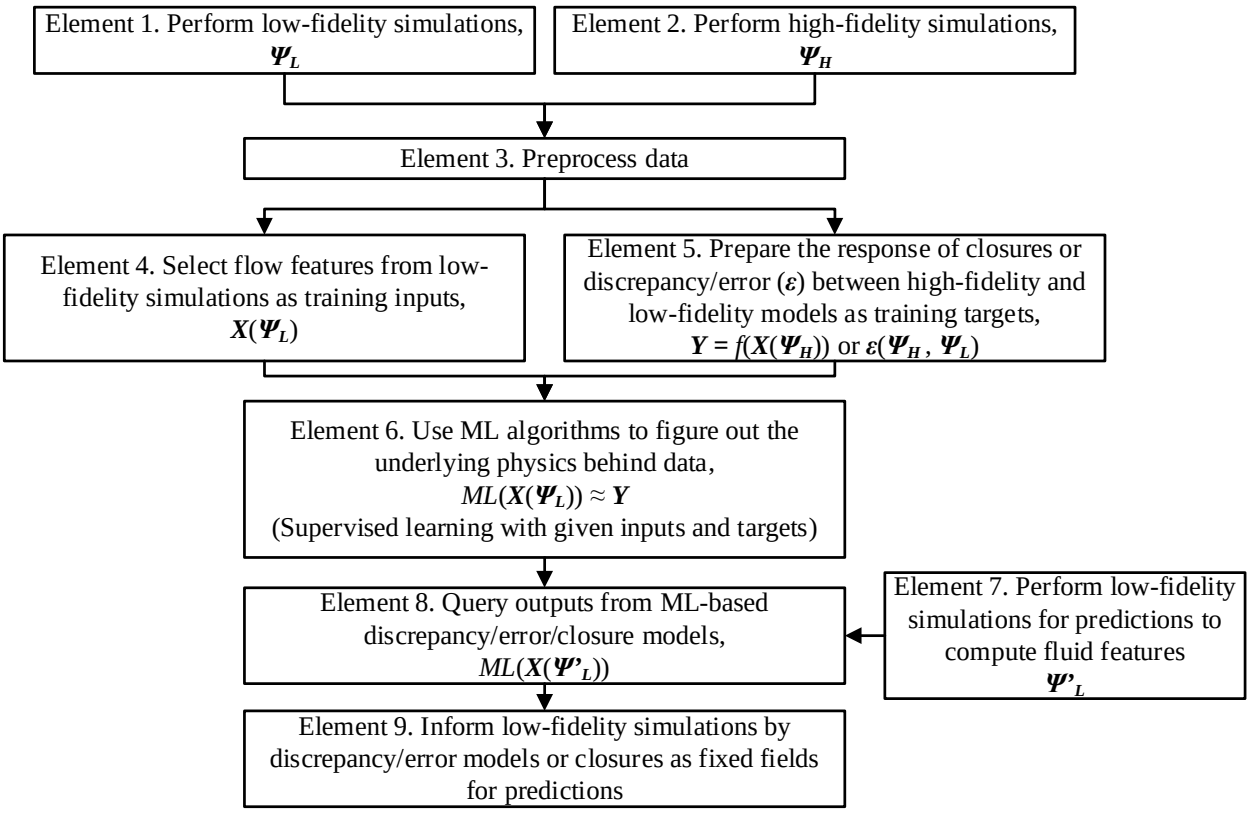


Figure 3.8. Overview of Type II ML framework.

Type II ML satisfies the first two criteria in Table 3.1. We note that PDEs and closure relations are loosely coupled in Type II ML because PDEs are only used for calculating input flow features. The framework provides a one-step solution to improve low-fidelity simulation. Model uncertainty is not accumulated in Type II ML because numerical solvers do not interact with ML models. However, for Type II ML there exists an open question about what the magnitude of initial errors can be before it is too late to bring a prior solution to a reference solution. For more detailed exampless of Type II ML, readers are referred to Ling & Templeton [43], Ling, *et al.* [90], Wu *et al.* [76], Wang *et al.* [77, 78], Ling *et al.* [19], and Zhu & Dinh [91].

3.2.4. Type III machine learning, physics-integrated machine learning (PIML)

To the best knowledge of the author, Type III ML or so-called physics-integrated ML (PIML) is introduced and developed for the first time in this work. Type III ML aims at developing closure relations to close thermal fluid models without a scale separation assumption. Closure models are embedded and trained in system dynamics. Training data can be obtained from SETs and IETs. Notably, Type III ML can lead the paradigm shift of using ML in thermal fluid simulation because it allows the direct use of field data from IETs. Figure 3.9 shows the framework of Type III ML. Inputs for Type III ML do not directly come from observations; instead, they are solutions of PDEs. Type III ML involves the following elements:

Element 1. Collect data, (x_k, y_k), from high-fidelity simulations or experiments that are used to compute targets for the training.

Element 2. Preprocess the data from element 1 to ensure that data from multi-sources are consistent with conservation equations regarding the dimension and manipulation such as the selection of averaging methods. Additionally, consider normalizing data so that we can

approximately equalize the importance for each data source. For large datasets, employ principal component analysis [84] can reduce the dimension of data.

Element 3. Prepare training targets (***Y***) from data that corresponds to PDE solutions.

Element 4. For the initial step of the sub-framework, calculate flow features (***X***) from data as training inputs for element 5. After that, flow features are computed based on PDE solutions from element 6.

Element 5. Adjust model parameters of an ML-based closure, *ML*(***X***), using an ML algorithm. Then output the ML-based closure to element 6.

Element 6. Solve conservation equations with the ML-based closure that is iteratively queried during a solution scheme.

Element 7. Check if the solution from element 6 converges to the target values within a tolerance interval. If the convergence test does not pass, go to element 4 and continue the loop in the sub-framework. If the result is converged, output the conservation model with the embedded ML-based closure to element 8. The selection of tolerance intervals is case-dependent.

Element 8. Solve the PDE model from element 7 with new system characteristics for predictions.

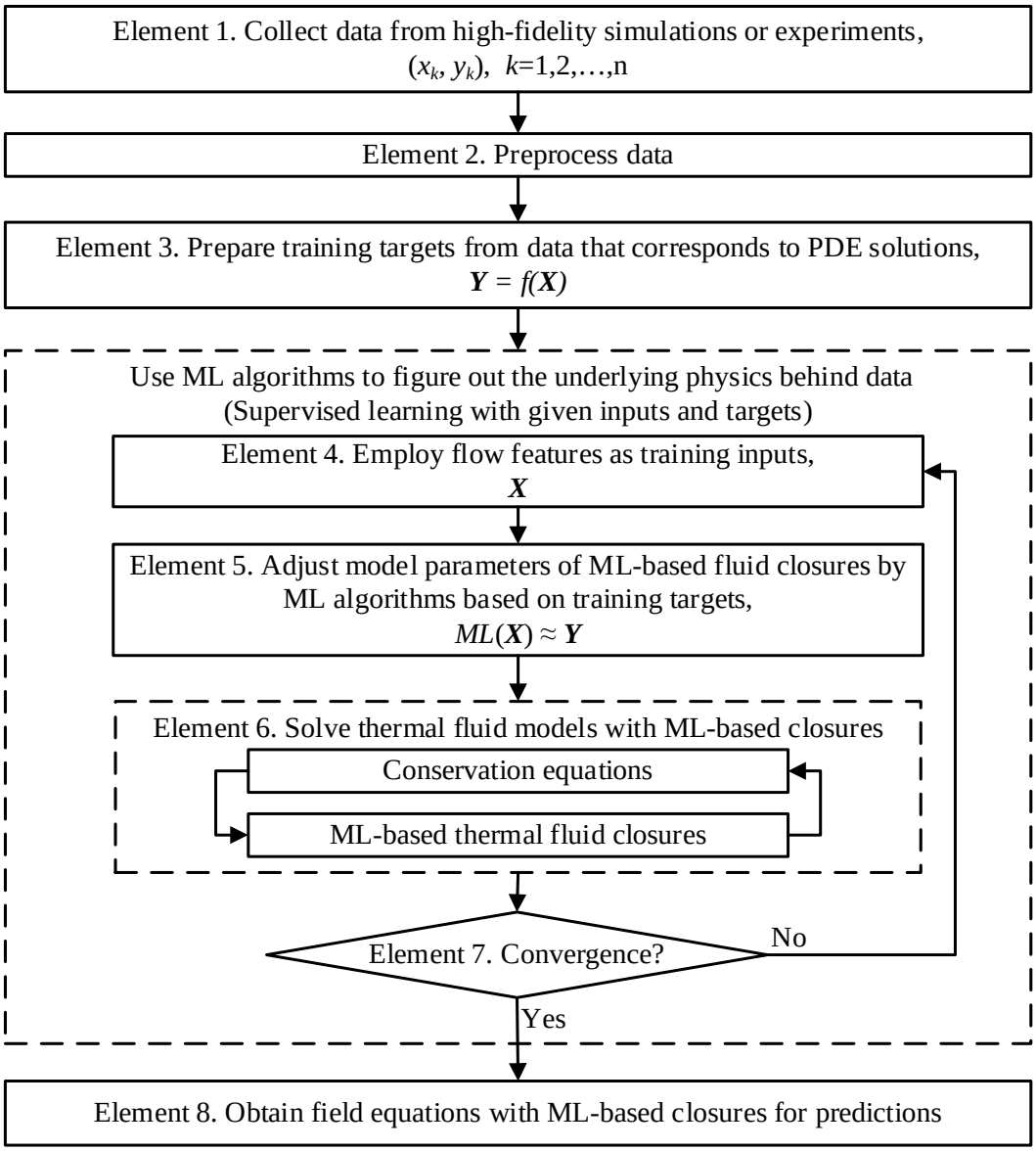


Figure 3.9. Overview of Type III ML framework.

Type III ML satisfies most criteria in Table 3.1 except for the fourth criterion. We note that PDEs and closure relations are tightly coupled in Type III ML. It is a challenging problem. Such tightly coupled multiscale problems require that numerical solutions (of the governing PDE system) are realized (hence evolving datasets for training) whenever ML algorithms tune model parameters. Therefore, Type III ML is computationally expensive. The research on Type III ML

methodology promises a high-potential impact in complex thermal fluid problems where the separation of scales or physics decomposition may involve significant errors.

3.2.5. Type IV machine learning, physics-recovered machine learning (PRML)

Type IV ML or so-called physics-recovered ML (PRML) aims at recovering the exact form of PDEs. Figure 3.10 depicts the framework of Type IV ML. It requires no assumption about the form of governing equations. Instead, the framework requires to construct a candidate library that includes components of governing equations such as time derivative, advection, diffusion, and higher order terms. For instance, Eq. (3.1) shows the equation that we want to recover from data. We can assume a model given in Eq. (3.2) where $\boldsymbol{\Theta}(\boldsymbol{X})$ is a library including candidate terms and it is defined by Eq. (3.3). The goal is to find the vector (ξ) such that ξ can make Eq. (3.2) identical to Eq. (3.1). Eq. (3.4) gives a solution which can be found by the sparse regression method [69].

$$\frac{\partial y}{\partial t} = A\frac{\partial^2 y}{\partial x^2} + B\frac{\partial y}{\partial x} + C \tag{3.1}$$

$$\frac{\partial y}{\partial t} = \Theta(\mathbf{X})\xi \tag{3.2}$$

$$\Theta(\mathbf{X}) = \left[\frac{\partial^4 y}{\partial x^4}, \frac{\partial^3 y}{\partial x^3}, \frac{\partial^2 y}{\partial x^2}, \frac{\partial y}{\partial x}, 1\right] \tag{3.3}$$

$$\xi = [0, 0, A, B, C]^T \tag{3.4}$$

The procedure of Type IV ML contains the following elements:

Element 1. Collect time series data ($\boldsymbol{\omega_t}$) from either validated simulations or experiments.

Element 2. Build a library, $\boldsymbol{\Theta}(\boldsymbol{X})$, for candidate terms in governing equations.

Element 3. Reconstruct governing equations using the time derivative term ($\partial y/\partial t$) and the optimal combination of candidate terms by sparse regression [69] with a sparse vector ($\boldsymbol{\xi}$) that follows Occam's razor [92].

Element 4. Solve the recovered governing equation with new system characteristics for predictions.

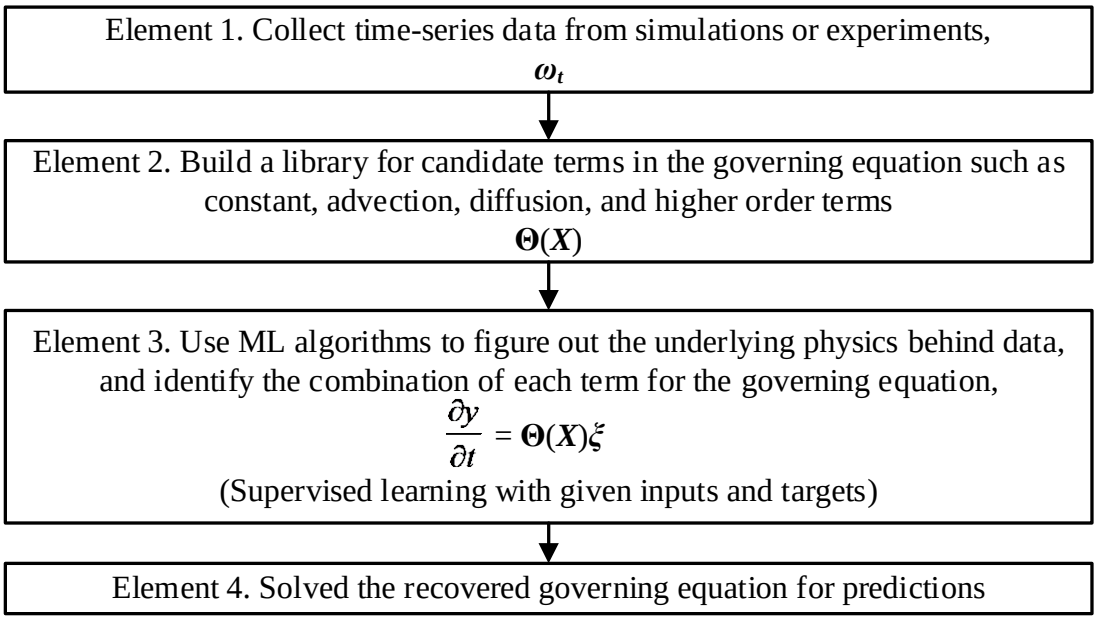


Figure 3.10. Overview of Type IV ML framework.

Type IV ML only satisfies the first criterion in Table 3.1. The challenge of Type IV ML can be the recovery of closure relations in thermal fluid models. Closure models are usually complex, and they are hard to be represented by each derivative term. Therefore, it is an open question about how to apply Type IV ML for complex flow system such as turbulence modeling. For more practices related to Type IV ML, readers are referred to Brunton *et al.* [69].

3.2.6. Type V machine learning, physics-discovered machine learning (PDML)

Type V ML or so-called physics-discovered ML (PDML) is the extreme case. Type V ML is used for either condition. First, it assumes no prior knowledge of physics. Second, it assumes existing models and modeling tools are not trustworthy or not applicable for thermal fluid systems under consideration. More generally, Type V ML is “equation-free” and instrumental in the search

for a new modeling paradigm for complex thermal-fluid systems. Type V ML does not involve conservation equations nor satisfy any criterion in Table 3.1. Instead, it wholly relies on data to discover the effective predictive models. However, such situation rarely occurs because there are usually physics principles or hypotheses that can be postulated to reduce the dimension of problems. For the discussion related to Type V ML, readers are referred to Mills *et al.* [70], and Hanna *et al.* [93].

3.2.7. Knowledge and data requirements for ML frameworks in NSTH

In the present context of ML, knowledge refers to a body of theoretical and empirical evidence that is available and trustworthy for understanding and description of physical mechanisms that underlie thermal fluid processes under consideration. This knowledge can guide selecting model forms, including conservation equations and corresponding closure relations, designing experiments, and performing high-fidelity simulations. The data requirements refer to characteristics of the body of data (e.g., types, amount, quality) needed to enable NSTH simulation with the required accuracy. In other words, the required data must be sufficient to complement the "knowledge" for building closure models and recovering/discovering the physics.

The form of PDEs are known for Type I, Type II, Type III ML, and the focus is to build closure relations. In traditional modeling approaches, closure models are local, relating a group of (local) source terms (i.e., sub-grid-scale interactions) to a group of (local) flow features. Even when in engineering literature, source terms are expressed regarding global parameters (like flow rate, system pressure), they are used as surrogates for local-valued parameters (through the assumptions that equate global and local conditions).

Type I ML build closure relations independently from PDEs, but it requires a thorough or assumed understanding of the physics that is essential to set up SETs for acquiring data. Globally

measured data or locally measured data (using point instruments) are very small amount of data. In such case, complicated ML-based closures are not necessarily the best choice. Therefore, among the frameworks, Type I ML exhibits a minimal data requirement with a maximal knowledge requirement.

Type-II ML assumes prior knowledge of physics that guide the selection of closure relations for NSTH simulation. However, the use of prior models yields uncertainty in thermal fluid analyses. This uncertainty (or error) can be inferred by comparing the model prediction to reference solutions from high-fidelity simulations, high-resolution experiments as well as data obtained in IETs that include multi-physics phenomena. Correspondingly, Type II ML requires larger data quantities but less knowledge than Type I ML.

Type III ML trains closure relations that are embedded in conservation equations without invoking a scale separation assumption. IET data can be directly adapted into simulation by applying Type III ML. While the term ML is broad, in the present work ML refers to the use of non-parametric models or even narrower, use of DNNs. This means no prior knowledge of model forms of closure relations. Thus, Type III ML requires less knowledge than Type II ML (which "best-estimated" closure models on the basis of past data). Consequently, Type III ML requires a large body of data to represent models than that of Type II ML.

Type IV ML intends not to make any bias on selecting conservation equations; instead, it recovers the exact PDE form from data. It assumes less prior knowledge but requires more extensive training data than the previous three frameworks.

Type V ML is an extreme case that makes no assumption about prior knowledge or reference solutions for thermal fluid systems under consideration. The aim is to apply ML methods to learn from data, and to establish a data-driven predictive capability. For NSTH simulation, it

means discovering the effective model form of conservation equations and closure relations. Accordingly, among the frameworks, Type V ML is the most stringent with respect to data requirements (types, quantity, and quality).

Figure 3.11 depicts the domain of ML frameworks regarding prior knowledge and data requirements.

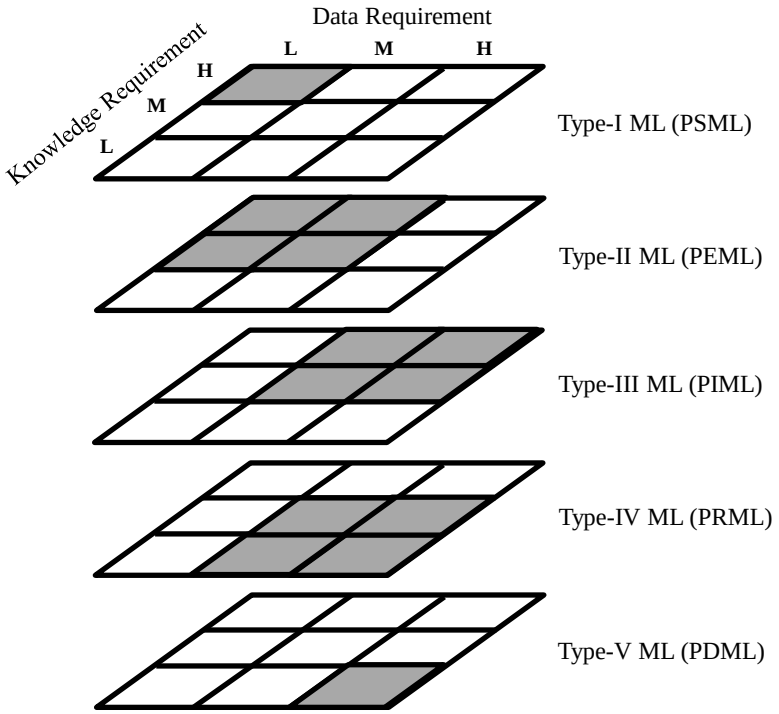


Figure 3.11. The domain of various ML frameworks.

3.3. Contemporary works

The contemporary works summarized in Section 2.5 can be classified using five types of ML frameworks. Figure 3.12 summarizes the contemporary works of using ML in NSTH. While there is a growing interest and recognition of potential capability of ML in NSTH simulation, development in this area is still in its infancy. There is no two-phase flow application in thermal-hydraulics yet identified from the literature surveys. In this work, we classify five ML frameworks for DDM of NSTH. Selections of frameworks are based on knowledge and data requirements.

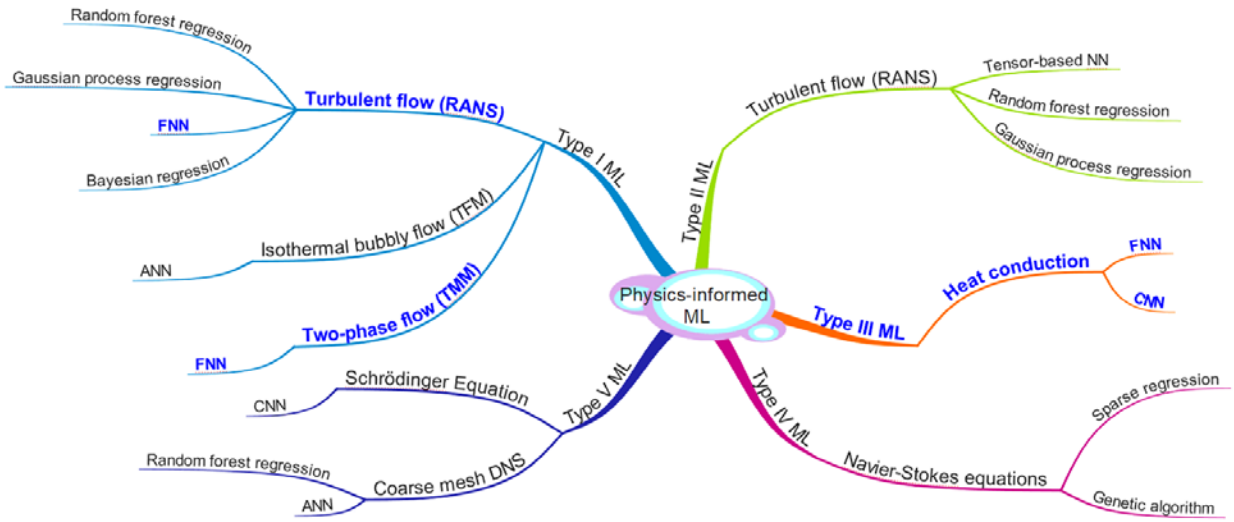


Figure 3.12. Contemporary work of using ML in thermal fluid simulation.

3.4. Evaluation and implementation of machine learning frameworks

3.4.1. Method of manufactured data (MMD)

To evaluate the proposed ML frameworks, this dissertation work proposes a method of manufactured data (MMD) as surrogates for actual datasets (such as experimental measurements and simulation results) in real-life applications. MMD applies a computer code with high-fidelity models to generate numerical solutions, of which datasets with distinct characteristics are selected for training and testing purposes. The "high fidelity" here refers to models which have been subjected to extensive adjustments and assessed to be trustworthy for conditions under consideration. Both training and testing datasets can be generated with different degrees of detail (homogenization, amount), controlled uncertainty ("manufactured errors and biases"), and other qualities. The testing datasets are typically broader than training datasets because they will serve as the benchmarks to evaluate predictive capability of the trained models.

3.4.2. Requirements of well-posedness

Inferring models from data belongs to inverse problems which are ill-posed. According to Hadamard's [85] definition, well-posed problems should satisfy three requirements: a solution should exist, the solution should be unique, and the solution should continuously depend on data. Based on Hadamard's definition, we define the well-posed criteria for PDE-constrained machine learning (PDE-ML) problems as follows.

(a) A solution should exist.

(b) The acceptable solutions should be limited in an acceptable interval.

(c) A small change in the inputs should only cause a small change in the outputs.

In NSTH simulation, there are lots of closure relations which represent the data from legacy experiments in compact forms. To leverage the values from those experiments, requirement (c) can be extended as follows.

(d) The output behavior of ML-based models should change continuously with insights.

The "insight" refers to the current knowledge to the problem of interest. For instance, the friction factor should be proportional to the inverse of Reynolds number for laminar flow. Based on requirement (d), we define the model complexity (MC) by Eq. (3.5) where I is insights. Models (M) are machine learning models, and we use deep neural networks in this study. Parameters (Par) are governing parameters such as flow properties. Eq. (3.6) defines model-insight consistency (MIC) which can be used to systematically analyze the requirement of well-posedness. MIC does not relate to the model accuracy; instead, it serves as a criterion to select an optimal DL-based model that is well-posed. We refer both MC and MIC numbers as the physics-informed regulation parameters.

$$MC = \frac{|\frac{\partial M}{\partial Par}| - |\frac{\partial I}{\partial Par}|}{|\frac{\partial I}{\partial Par}|} \tag{3.5}$$

$$MIC = 1 - MC \tag{3.6}$$

3.4.3. Search for well-posed PDE-constrained ML models

Based on the physics-informed regulation parameters (model complexity and model-insight consistency), we propose the physics-constrained deep learning (PCDL) strategy to obtain well-posed deep learning models for NSTH simulation. PCDL includes a two-step process: policy and value networks. The policy network can pre-identify the numerical stability criteria of PDE solutions. The value network utilizes those criteria to search for deep learning models which are well-posed.

Figure 3.13 depicts information flow of the policy network, which can search for the numerical stability criteria of PDE solutions. Initially, we arbitrarily assign hyperparameters for deep neural networks such as numbers of hidden units and hidden layers and learning rate. Then we start to train deep neural networks with random weight initialization. After the training, we implement DL-based closure relations in conservation equations for system simulation. If the solution is diverged, we record the parameter set and simulation results which become inputs for the value network. If the solution is converged, we need to adjust model hyperparameters until the unsuccessful simulation is found.

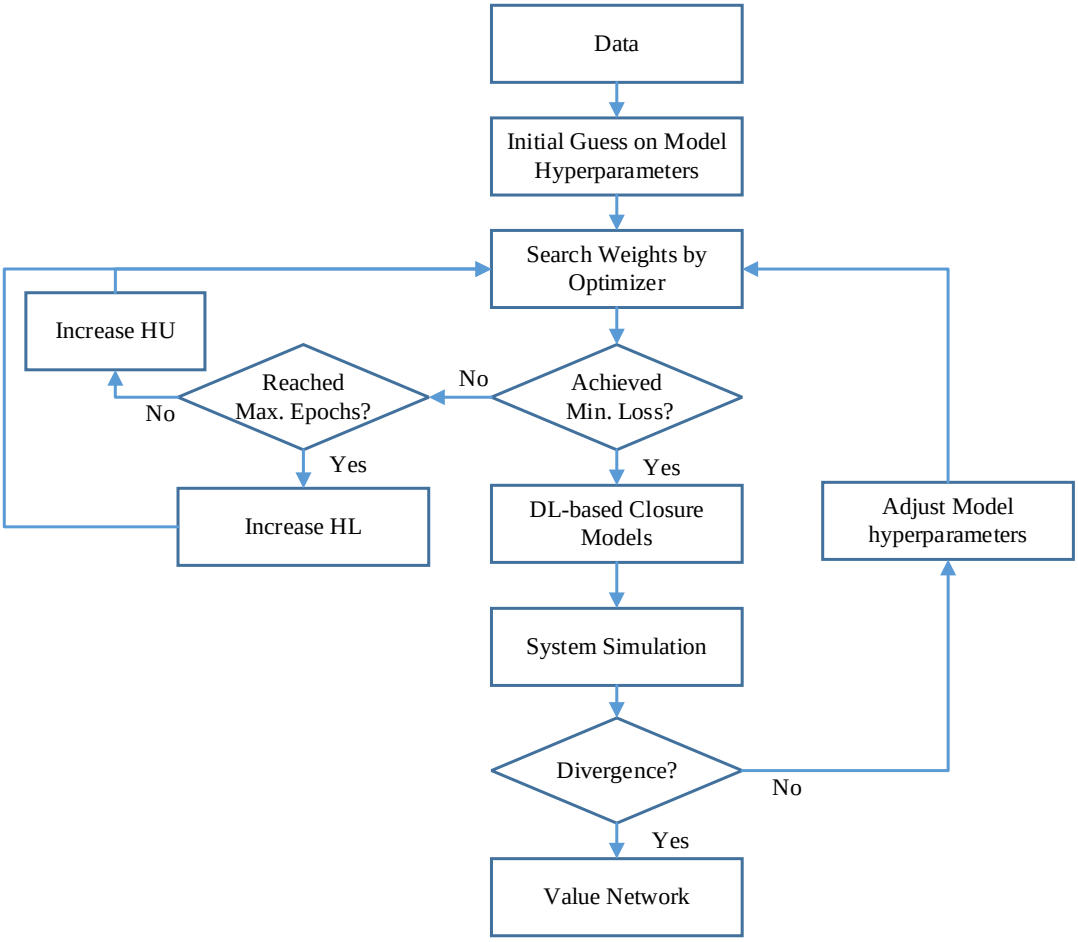


Figure 3.13. Policy Network for numerical stability criteria of coupled PDE-DL simulation.

The value network takes inputs from training datasets, insights, and outputs of the policy network. By using insights and the outputs of the policy network, we can calculate model complexity given by Eq. (3.5) from the unsuccessful simulation. Therefore, we can calculate model-insight consistency from model complexity, and model-insight consistency will become the screening criterion to search for the optimal DL-based closure model.

Figure 3.14 illustrates information flow of the value network, which can search for the well-posed deep learning model for NSTH simulation. Initially, we train multiple deep neural networks with different hyperparameters. After the training, DL-based closures are implemented in conservation equations. Based on model-insight consistency, we can find well-posed deep

learning models. Then we test those models with testing data to figure out the optimal model with the maximum predictive capability. The process can be iteratively repeated until we find the optimal hyperparameter set for DL-based closures. The case study in CHAPTER 4 demonstrates how to use the physics-constrained deep learning strategy to obtain the DL-based friction model that is well-posed.

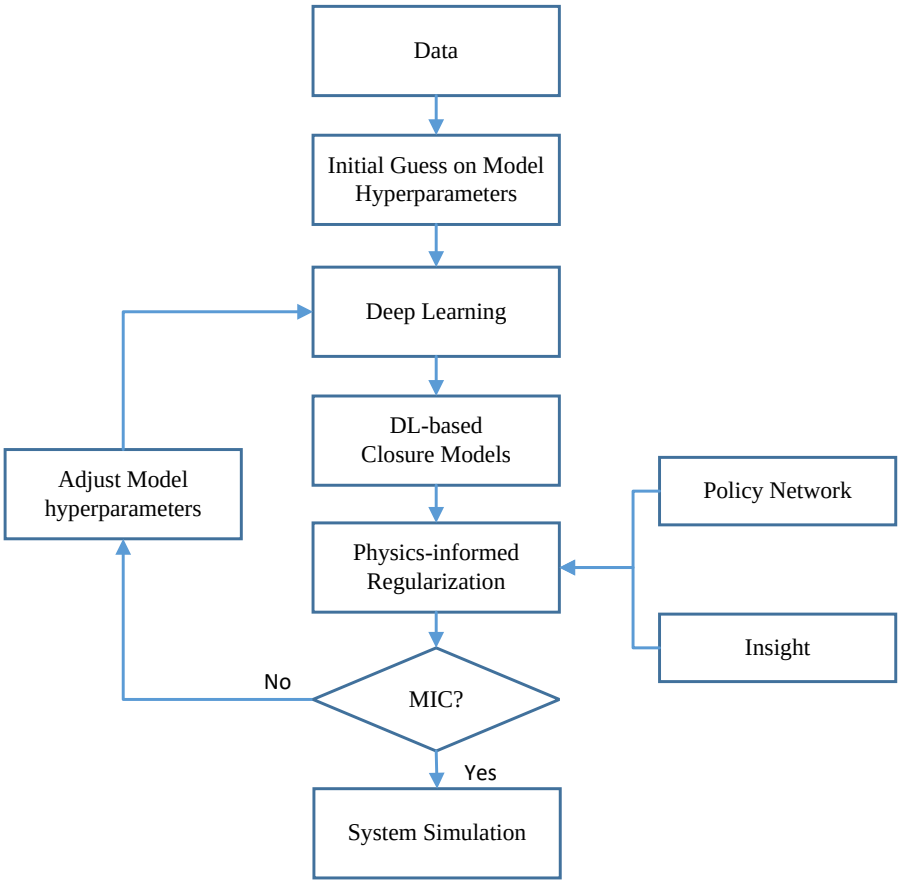


Figure 3.14. Value network for development of well-posed DL-based closure models.

3.4.4. Data quantity requirements

The well-posed requirement (c) in Section 3.4.2 requires that a small change in the inputs should only cause a small change in the outputs. When training data are insufficient, DL-based models can be sensitive to the inputs such that a small change in the inputs can result in a large

change in the outputs. To investigate in this sensitivity issue, we use principal component analysis (PCA) [94] to correlate the eigenvalues of training datasets to the relative output error.

Eq. (3.7) defines the fraction of variance explained by a single principle component where λ and i are the eigenvalue and i^{th} principal component.

$$ev_i = \frac{\lambda_i}{\sum_i \lambda_i} \tag{3.7}$$

We assume the minimum principal components should at least explain 95% of the total variance from raw data. Eq. (3.8) defines the recovery factor (RF) to quantitatively analyze the constraint of required datasets for achieving robust PDE-DL simulation. Recovery factor is scaled to return the value between zero and one. It is one when all the datasets are used for the training.

$$RF = \max\left(20\left(\sum_i ev_i - 0.95\right), 0\right) \tag{3.8}$$

Eq. (3.9) defines the relative error (ε) for sensitivity analysis where $\hat{y}$ and y are the model output and data.

$$\varepsilon = \frac{|\hat{y} - y|}{y} \tag{3.9}$$

Figure 3.15 depicts the workflow to determine the minimum training datasets for well-posed deep learning models. Initially, we apply PCA to the raw data and select several candidate datasets with different amount of training data. For each candidate dataset, we calculate RFs and train the corresponding deep learning models. After obtaining well-trained deep learning models, we can perturb model inputs by a small error. Then we can check if the error is amplified by deep learning models. If the error is amplified, we abandon the model and test other models which are trained with more datasets. Until we find a deep learning model that is satisfy the well-posed requirement (c) in Section 3.4.2, we record its recovery factor number and this number reflects the

minimum training datasets for a well-posed deep learning model. If we cannot find any model that satisfy the well-posed requirement, we need to increase the quantity of training data. Then we repeat the processes in Figure 3.15 until a well-posed deep learning model is found.

The case study in CHAPTER 5 demonstrates how to decide the minimum datasets for a well-posed DL-based closure model using the workflow in Figure 3.15.

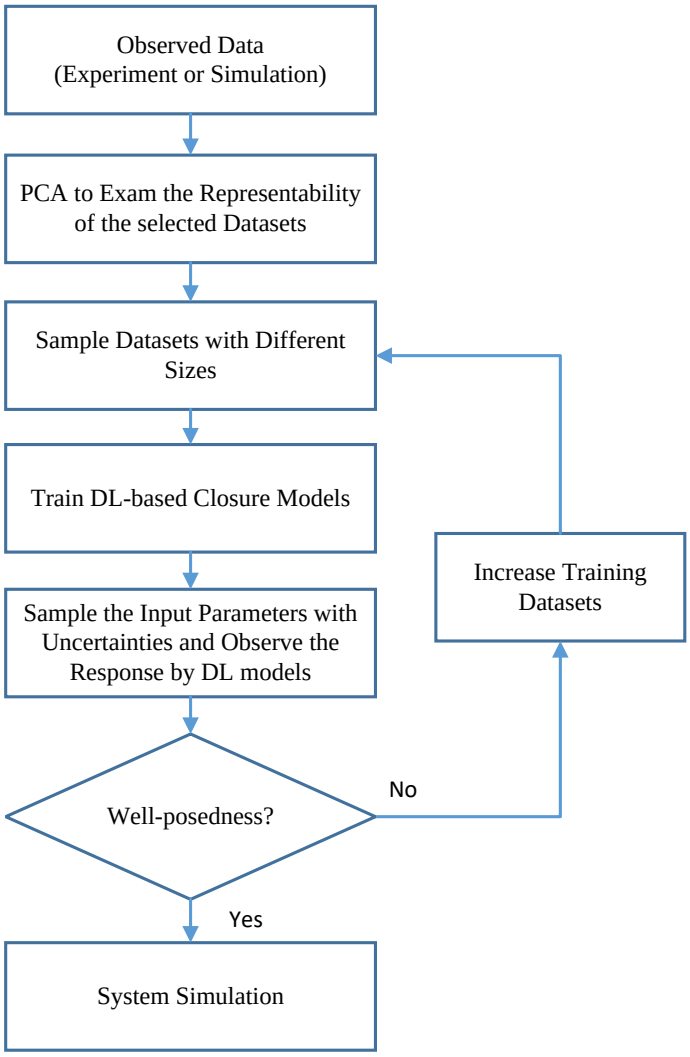


Figure 3.15. Workflow for data requirement of DL-based fluid closures.

3.5. Summary

Five ML frameworks are formulated in this chapter for data-driven modeling of NSTH. Each machine learning framework can leverage values of data from advanced validation

experiments or high-fidelity numerical simulations. The selection of framework is based on data and knowledge requirements. The framework with deep learning can shorten model development time and extend the applicability of NSTH simulation. To evaluate machine learning frameworks for NSTH simulation, we use manufactured data that allow us to manipulate different data qualities for framework comparison.

Although a classification system is established in this chapter, there are still challenges about how to implement machine learning frameworks in NSTH simulation. Starting from CHAPTER 4, several case studies are formulated to investigate in the critical questions of PDE-DL simulation. Those questions are listed as follows.

(a) What is data? How frequent should data be sampled? How much data are sufficient for a well-posed DL-based closure?

(b) What are flow features? How to select flow features?

To leverage values of a substantial amount of data, the next generation NSTH code is expected to be multi-dimensional. Therefore, the case studies are formulated to include system-level simulation and CFD simulation.

CHAPTER 4. CASE STUDY A: REQUIREMENTS OF WELL-POSEDNESS

4.1. Introduction

In this case, we show that DL-based friction closure can cause the modeling of single-phase water to diverge if multilayer neural networks are used, and how to employ the physics-constrained deep learning (PCDL) strategy to build the well-posed DL-based closures for PDE simulation. We refer to this process as PDE-constrained machine learning or PDE-ML. The isothermal single-phase flow simulation is the simplest application that it only requires the wall friction drag. To address the issue of PDE-DL simulation, we investigate the conditions by which the DL-based closure models work compatibly, stably, and effectively with PDE-constrained forward prediction problems. To develop the technical basis for PDE-ML, numerical experiments are performed for manufactured problems of increasing complexity. The case study belongs to Type I ML.

4.2. Objective

The objective of this case study is to investigate whether model-insight consistency (MIC) can be used as the screening criterion to find well-posed DL-based closures. The proposed notion of MIC and PCDL strategy are defined in Section 3.4.2.

4.3. Problem formulation

We examine the PCDL strategy by using the known solution for the wall friction problem in the Modelica Standard Fluid Library (MSFL) to manufacture experimental data and construct a DL-based friction closure by PCDL from globally measured data at inlet and outlet. Figure 4.1 depicts the layout of the numerical experiment. The flow properties such as the pressure drop and mass flow rate are recorded from “testSection” pipe.

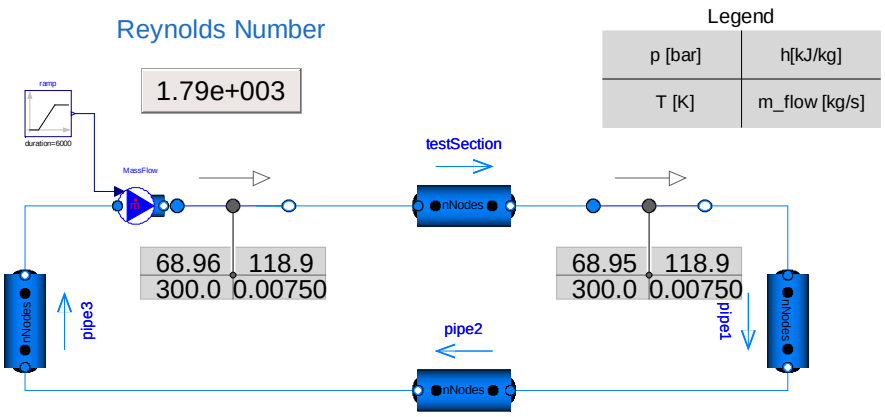


Figure 4.1. Experiment simulation layout in Modelica.

Figure 4.2 shows a controller with a ramp function for varying the mass flow rate. To simplify the problem, we deal with laminar flow and control the Reynolds numbers ranging from 100 to 2000. We uniformly sample 462 points for the case where Reynolds numbers are within [300, 1800] at the outlet of the testSection pipe. The manufactured datasets allow us to test the performance of DL-based friction closure in the extrapolation region since the range of Reynolds number in the training dataset is smaller than the Reynolds number in the actual application.

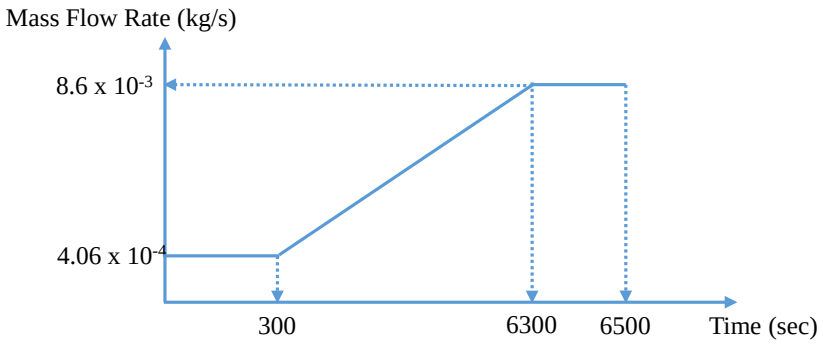


Figure 4.2. A controller for varying the mass flow rate.

Then the generalized DL-based friction closure is implemented into MSFL, and we test if the baseline solution can be reconstructed. The training target, friction factor, is obtained by Eq.

(4.1) assuming the flow is fully-developed. The governing parameter (Par) in this case is Reynolds number, and we select it as the input for training DL models. Table 4.1 summarizes the experimental condition, and we uniformly sample 462 points from the training dataset.

$$\xi = 2\frac{\Delta P D_{hyd}\rho}{G^2 L} \tag{4.1}$$

Table 4.1. Experimental conditions.

Parameters	Values
Pipe length (m)	2
Pipe diameter (m)	6.25×10^{-3}
Flow rate range (kg/s)	4.06×10^{-4} – 8.6×10^{-3}
Re range for experiment	100 – 2000
Re range for training	300 – 1800

4.4. Theoretical treatment

We assume that the global model is valid for the local application and this is a reasonable assumption because this case models single-phase laminar flow in a smooth pipe. Therefore, the friction factor can be calculated by Eq. (4.1) using the measurement data from the numerical experiment. It is the global friction model since it uses the full pipe length to calculate the friction model.

The insight, in this case, is the analytical friction factor by assuming the flow was parabolic [95]. Figure 4.3 shows a diagram of fully-developed laminar flow in a cylindrical channel with the pip length and radius equal to L and R. Eq. (4.2) shows the force balance where τ and P are the viscous stress and pressure. Eq. (4.2) can be rearranged into Eq. (4.3) to present the relation between the viscous stress and pressure difference. At r = D/2, Eq. (4.4) shows the wall shear stress. Eq. (4.5) shows the shear stress of a Newtonian fluid where u denotes the velocity in the axial direction. We can take Eq. (4.3) into Eq. (4.5), and integrate Eq. (4.6) to obtain the axial velocity profile given in Eq. (4.7). Because the pipe is axisymmetric along the centerline, we can derive the

volumetric flow rate from Eq. (4.8). Then we can further rearrange the equation into Eq. (4.9) that gives the relation between the pressure difference and average velocity (v). Finally, we take Eq. (4.9) into Eq. (4.1) to obtain the analytical form of the friction factor for the laminar fully developed flow. Eq. (4.10) is used to examine whether the DL-based friction model changes continuously with the insight for the problem of interest.

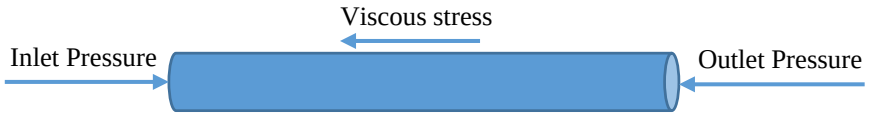


Figure 4.3 Diagram of laminar flow in a cylindrical pipe.

$$p_{in}\pi r^2 = p_{out}\pi r^2 + \tau 2\pi r L^2 \tag{4.2}$$

$$\tau = \frac{r\Delta P}{2L} \tag{4.3}$$

$$\tau_w = \frac{R\Delta P}{2L} \tag{4.4}$$

$$\tau = -\mu\frac{du}{dr} \tag{4.5}$$

$$\int du = \int -(\frac{\Delta P}{2\mu L})rdr \tag{4.6}$$

$$u(r) = \frac{R^2\Delta P}{4\mu L}\left[1-\left(\frac{r}{R}\right)^2\right] = \frac{\tau_w R}{2\mu}\left[1-\left(\frac{r}{R}\right)^2\right] \tag{4.7}$$

$$Q = \int_0^R u(r)dA = \int_0^R u(r)2\pi r dr = \frac{\pi R^4\Delta P}{8\mu L} \tag{4.8}$$

$$\Delta P = \frac{8\mu L Q}{\pi R^4} = \frac{8\mu L v}{R^2} \tag{4.9}$$

$$\xi = \frac{64}{\mathrm{Re}} \tag{4.10}$$

4.5. Implementation

4.5.1. 1D area-averaged mass-momentum conservation equation

Eq. (4.11) and Eq. (4.12) give the 1D area-averaged mass-momentum conservation equations in the discretized form. According to staggered grid configuration, the velocity is solved at the boundary, and the pressure is solved at the cell center where *j* presents the jth cell. Eq. (4.13) shows how to solve the nonlinear term in Eq. (4.12) using Newton-Raphson method. The time discretization will use Radau IIA [96] solver in the Dymola platform.

$$V_j \frac{\rho_j^{t+\Delta t} - \rho_j^t}{\Delta t} + \rho_{j+1/2}^t v_{j+1/2}^{t+\Delta t} A_{j+1/2} - \rho_{j-1/2}^t v_{j-1/2}^t A_{j-1/2} = 0 \tag{4.11}$$

$$\rho_{j+1/2} \frac{v_{j+1/2}^{t+\Delta t} - v_{j+1/2}^t}{\Delta t} + (\rho v)_{j+1/2}^t \frac{v_{j+1/2} - v_{j-1/2}}{\Delta z} = -\frac{P_{j+1}^{t+\Delta t} - P_j^{t+\Delta t}}{\Delta z} - \frac{f_{j+1/2}^t \rho_{j+1/2}^t}{D_{hyd}} \left(v_{j+1/2}^{t+\Delta t}\right)^2 - \rho_{j+1/2}^t g \tag{4.12}$$

$$f(v^{k+1}) = (2v^{k+1} - v^k)\,|v^k| \tag{4.13}$$

4.5.2. Deep neural networks model

Eq. (4.14) shows the sigmoidal function as the activation for the DL model. Eq. (4.15) defines the structure of the first hidden layer of a neural network where *j* denotes the jth number of hidden units. To acquire the scalability, the Reynolds number is selected as the input. Eq. (4.16) defines the hidden layer starting from the second to the last hidden layer where *i* is the number of inputs from previous hidden layer and *k* indicates the kth layer number. Starting from the second hidden layer, the total number of inputs depends on the number of hidden units from the previous layer. Eq. (4.17) gives the output layer of deep neural neworks which is the linear combination of the last hidden layer, and *K* denotes the total number of hidden layer. We can replace the friction factor in Eq. (4.12) by Eq. (4.17) to achieve PDE-constrained prediction using the DL-based friction closure.

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{4.14}$$

$$HU_{1j}(\mathrm{Re}) = sigmoid(w_{1j}\,\mathrm{Re} + b_{1j}) \tag{4.15}$$

$$HU_{kj}(HU_{k-1}) = sigmoid(\sum_i w_{kji} HU_{k-1,i} + b_{kj}) \tag{4.16}$$

$$\xi_{DL}(HU_{K-1}) = \sum_i w_{oji} HU_{K-1,i} + b_o \tag{4.17}$$

After collecting the data from the experiment, we construct DL-based closure models through the two-step physics-constrained deep learning (PCDL) strategy. First, the policy network is employed to search for the unsuccessful conditions for the simulation. Table 4.2 gives the model parameters of deep neural networks. Then we construct nine DL-based friction closures by (n+1)-layer NN models using Eq. (4.17) where n varied from one to three and n represents the total number of HL. To simplify the case, we let the number of hidden units be the same in each hidden layer. After defining deep neural networks with different model parameters, we use Tensorflow to optimize the weights in each layer to obtain the working models that we can implement in the system code for further evaluation.

Table 4.2. Parameters for DL-based wall friction models.

Parameters	Values
Activation function	Sigmoid
Number of HU in each HL	5, 25, 100
Number of HL	1, 2, 3
Number of inputs	1
Number of outputs	1

4.6. Data processing and results

The first step of PCDL is to identify when the PDE-DL simulation failed. Following the workflow of the policy network given by Figure 3.13, Table 4.3 records the performance of PDE-DL modeling with different deep neural networks, and this information serves as the input for the value network to search the optimal DL-based friction closure. According to Table 4.3, the instability issue occurs as the number of hidden units increased. There is no numerical issue for the model with small hidden units and large hidden layers. In the meantime, large hidden layers help the model to accurately catch the pattern from complex data.

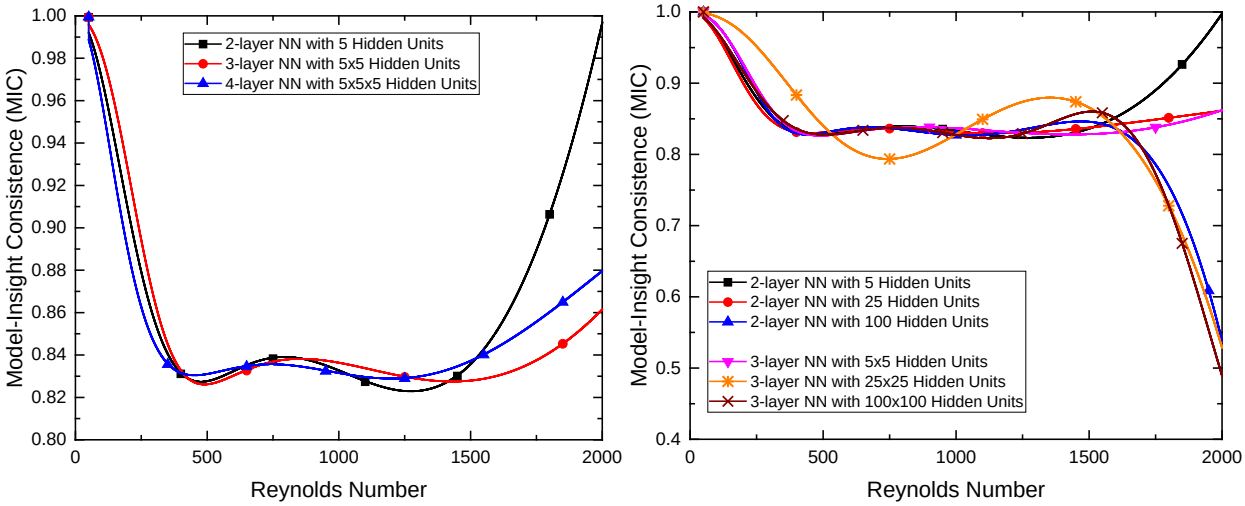
Table 4.3. Performance record from the policy network.

HU \ DL	2-layer	3-layer	4-layer
5	Success	Success	Success
25	Success	Failure	Failure
100	Failure	Failure	Failure

The insight, in this case, is the analytical friction factor given by Eq. (4.10), and we compute the model-insight consistency (MIC) number given by Eq. (3.5) and Eq. (3.6). Figure 4.4(a) depict the MIC plot for three deep neural networks with the same hidden units but different hidden layers, and all three models are successfully solved with conservation equations without numerical stability issues. Figure 4.4(b) shows that some PDE-DL simulations fail in the predicting domain with high Reynolds number (Re > 1800). PDE-ML simulations with complex structures start to fail when the MIC number is below 0.7. The value of 0.7 is recommended as the screening criterion in this case. Also, we resample 4620 points from the original experiment outcome and retrain the DL model. Figure 4.5 shows that the MIC is significantly increased with large datasets.

In the meanwhile, we compare the performance of two activation functions, sigmoid and ReLU (rectified linear unit), using the same hyperparameters for model construction. ReLU

functions is defined as *max*(0, x) where x is the input. Under the same training condition, sigmoidal function yields better smoothness and fitting for this problem. Therefore, the value network focuses on searching the optimal DL-based friction closure using sigmoid function along with the insight and screening MIC number from the outcome of the policy network.



(a) (b)

Figure 4.4. (a) Model-insight consistency (MIC) for ML-based friction models with different hidden layers, and (b) model-insight consistency (MIC) for ML-based friction models with different hidden units and layers.

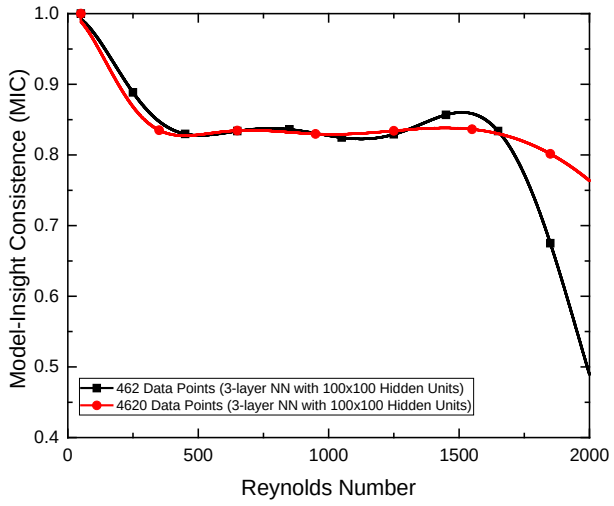
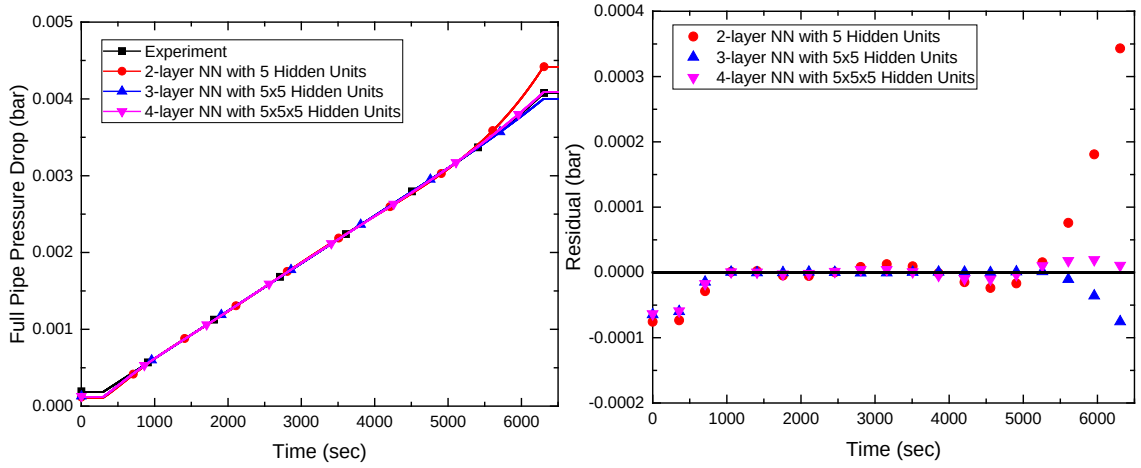


Figure 4.5. Model-insight consistency (MIC) for friction models with different datasets.

Based on the workflow by Figure 3.14, the value network finds that the four-layer neural network gives the optimal result. Figure 4.6(a) depicts the results of the full pipe pressure drop by using DL-based friction closures to replace the friction model in Eq. (4.12). When we fix the number of hidden units equal to five in each hidden layer, all DL-based friction closures successfully achieve PDE-DL simulation. However, the two-layer neural network is found to perform poorly in both prediction and training domains when it is compared to three-layer and four-layer neural networks. Figure 4.6(b) shows the residual plot where the residual is defined by Eq. (4.18). All model errors, relatively small in training, escalate in the extrapolation domain as expected.

$$\text{Residual} = \Delta P_{DL} - \Delta P_{Experiment} \tag{4.18}$$

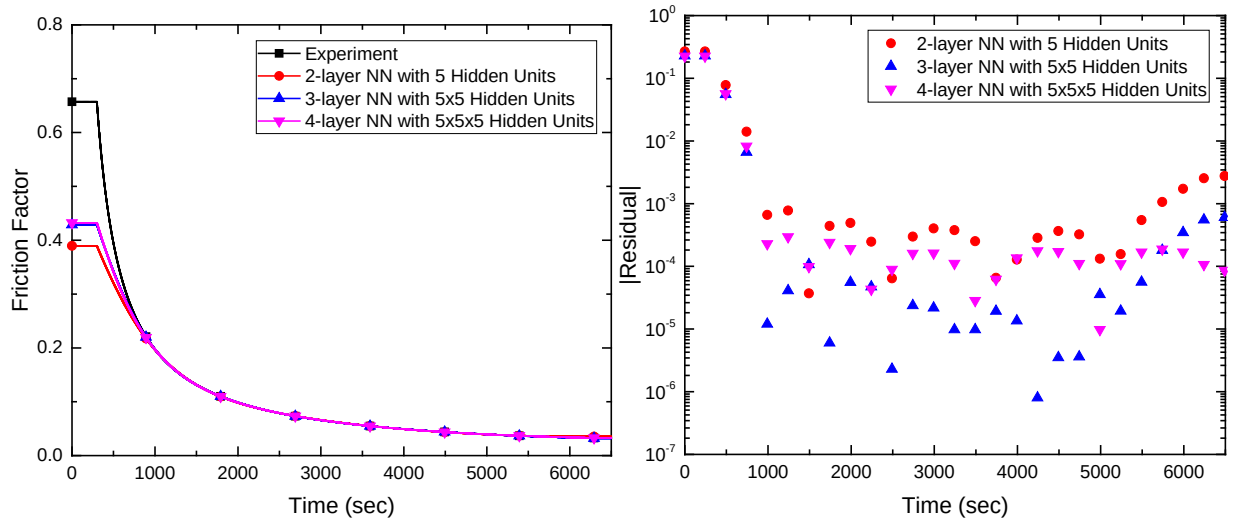


(a) (b)

Figure 4.6. (a) Full pipe pressure drops in both training and extrapolation domains by DL-based friction models, and (b) residual of full pipe pressure drops in both training and extrapolation domains by DL-based friction models.

Figure 4.7(a) shows the results of friction factors by Eq. (4.1) for three different deep neural networks. Although all models deviate from the experiment data in the low-Re extrapolation domain, there is no significant impact for calculating the pressure drop. Figure 4.7(b) illustrates that the pressure drop calculation is sensitive to the predictive friction factor in the high-Re domain.

As a result, the four-layer neural network tends to exhibit good predictability in the high-Re extrapolation domain. This suggests that the neural network with small hidden units and large hidden layers yields not only a robust simulation model but also an accurate result.



(a) (b)

Figure 4.7. (a) Friction factors in both training and extrapolation domains by DL-based friction models, and (b) residual of friction factors in both training and extrapolation domains by DL-based friction models.

4.7. Analysis and lessons learned

Based on the case study, we observe the following features for using the physics constrained deep learning strategy to stabilize coupled PDE-DL solutions.

i. The model of friction factor is a smooth curve with non-negative values, and selecting the activation function, sigmoid, produces stable and more accurate results than choosing ReLU function.

ii. A large number of hidden units is expected to provide a good fitting in the training domain. However, it results in high model complexity and causes the instability while solving the coupled PDE-DL problems.

iii. Increasing hidden layers will not increase the model complexity, and it allows better prediction than neural networks with less number of layers.

iv. If a large number of hidden units is needed, a sufficient quantity of data must be provided to control the model complexity.

This case studies a simple physics using a small dataset (hundreds of data points). The result indicates that the PDE-DL simulation can be numerically stable by the combination of a small number of hidden units with a large number of hidden layers. For the case with large datasets, there is more freedom to adjust neural networks' hyperparameters that potentially enable deep learning models to capture complex physics.

4.8. Summary

The case study of the system-level fluid simulation introduces a notion of model-insight consistency (MIC). MIC is a screening criterion that has the potential to guide the search for the optimal DL model for Type I ML problems. The insight refers to the best knowledge of the problem of interest. It also potentially regularizes DL-based models to prevent them from outputting physically unreasonable values or unphysical oscillations. Guided by Occam's razor principle, the optimal model should be the deep neural network with the simplest structure that captures the insights and the data within an uncertainty range. This case study shows that model-insight consistency is indicative to evaluate the potential performance of deep learning models. It is noted that this case study is limited to modeling the fully developed laminar flow. The case study only includes a simple closure model with a single scaling parameter, Reynolds number. A broader case study is a must to characterize the usefulness of model-insight consistency in more complex

processes. For instance, the “insight” is subject to multiple scaling parameters and various sources of uncertainty such as pressure loss due to spacer grids.

CHAPTER 5. CASE STUDY B: REQUIREMENTS OF DATA QUANTITY

5.1. Introduction

We use Type I ML to demonstrate how to construct a closure relation to close two-phase mixture models (TMMs) [12]. TMMs are convenient to deal with the phase appearance and disappearance [12], and it can consistently increase the fidelity by increasing field equations. Figure 2.1 summarizes family of two-phase mixture models and its applicable problems. For example, the three-equation TMM assumes that the velocity, temperature, and pressure are homogeneous. It is not valid when the system includes the subcooled liquid. We can add field equations to extend applicable domains of mixture models. However, we need more closure relations to close field equations.

Nuclear system thermal-hydraulics (NSTH) simulation involves flow regime transitions, and this requires distinct closure models for each flow regime. The transitions complicate the scaling analysis since each experiment is valid in a particular domain and includes distinct uncertainties. Scalabrin, Condosta & Marchi [97, 98] utilized artificial neural networks to build heat transfer models applied to a range of flow regimes for boiling flows. In this demonstration, we explore the hypothesis if the data-driven approach by using deep learning can construct the slip closure that is valid over a range of flow regimes in a vertical boiling channel. We start the investigation by using the three-equation TMM to predict boiling channel flow.

5.2. Objectives

The objectives of this case include two parts. First, we use Type I ML to demonstrate how to construct a closure model to close two-phase mixture models [12]. Second, we demonstrate how to apply the developed strategy in Section 3.4.4 to find a reliable and accurate DL-based slip

closure for nuclear system thermal-hydraulics simulation. The demonstration includes scaling applications that are outside the training domain.

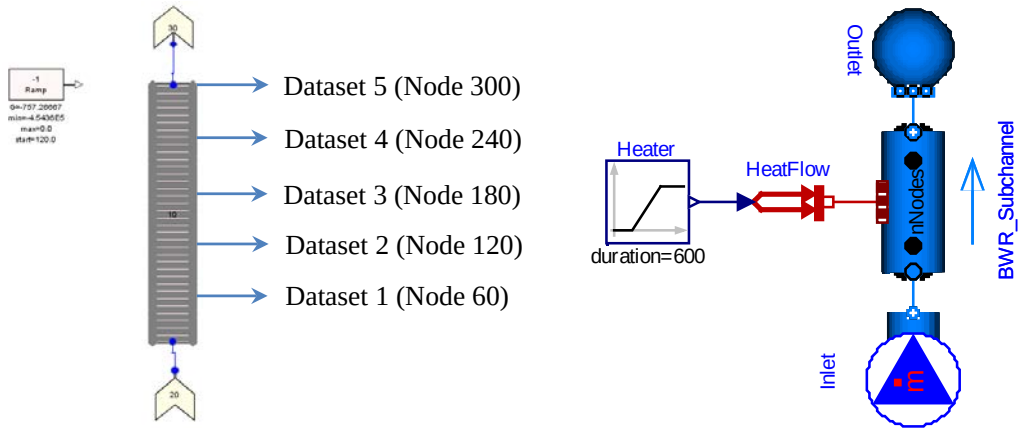
5.3. Problem formulation

The three-equation two-phase mixture model (TMM) only requires the void fraction and wall friction closures for two-phase flow simulations, and it is an ideal case for testing the performance of machine learning methods in two-phase flow simulation because of its simplicity. We further assume the model form uncertainty for the wall drag can be recovered by using a DL-based void fraction model. A BWR subchannel boiling case is selected for this task, and Table 5.1 gives the operating characteristics. In the meanwhile, Zuber-Findlay model [25] is widely used in predicting the void fraction, and we implement it into the three-equation TMM to check if the DL-based closure can be as successful as Zuber-Findlay model.

We use TRACE to generate high-resolution data for training a DL-based slip closure. Figure 5.1(a) illustrates the experiment layout by TRACE including a vertical heating pipe, fixed flow source, pressure boundary, and ramped heater. The ramped heater is off for the first 120 seconds, and then it gradually heats the pipe from 120 seconds to 720 seconds. The heater remains at its nominal power for the last 120 seconds. Figure 5.1(a) also illustrates the five locations where we sample the datasets for training a DL-based slip closure. Initially, we will use fixed datasets for DL training to see if the model can correctly predict the BWR subchannel case. In the meanwhile, the TMM is implemented in Dymola using Modelica language with hierarchical structures that the void fraction module can be switched to test the performance between drift flux and slip models. Figure 5.1(b) depicts the simulation layout in Dymola, and it has the same configuration as TRACE layout including a vertical heating pipe, fixed flow source, pressure boundary, and ramped heater.

Table 5.1. A BWR operating characteristics.

Parameters	Values
Channel heat flux (J/sec-m^2)	4.5436 x 10^5
Outlet pressure (bar)	67.73
Coolant mass flux (kg/sec-m^2)	1925.85
Inlet temperature (K)	550.93
Heated diameter (m)	0.0125
Heated length (m)	3.7084
Flow area (m^2)	1.41096 x 10^{-4}
TRACE nodalization	300
TMM nodalization	48



(a) (b)

Figure 5.1. (a) TRACE layout with 5 sampling locations for train the slip closure by deep neural networks, and (b) the experimental layout by Modelica for a BWR subchannel simulation.

We run TRACE with the BWR operating characteristics to obtain the training datasets, and we define the inputs for the deep neural networks including the local pressure and Reynolds numbers for both mixture and vapor. The Reynolds number is dimensionless and can predict the simulation flow structures with different fluid flow conditions. Therefore, we assume the DL-based closure can obtain predictability by using Reynolds numbers as inputs. The training target is slip factor which is the ratio of the vapor velocity to the liquid velocity. TRACE requires 300 axial nodes to achieve the converged solution, and we uniformly sampled the data from 5 nodes

as depicted in Figure 5.1(a). Then we will formulate the task that includes two parts. First, we will examine how good the DL-based slip model can work with the three-equation TMM. Second, we will demonstrate how to use the strategy introduced in Section 3.4.4 to achieve the well-posedness of two-phase simulation using the DL-based closure.

The first part of this task:

i. Manufacture the required data by running TRACE simulations using the BWR characteristics in Table 5.1 and modeling layout in Figure 5.1.

ii. Collect five datasets as shown in Figure 5.1(a) from TRACE results. Then use Tensorflow [57] to implement closures by deep neural networks, which can figure out underlying correlations behind data. Finally, utilize the Adam [99] algorithm to optimize DL-based closure models.

iii. Take the DL-based slip model from Tensorflow and add the constraint to the model based on the assumption of the slip model. Then we implement the DL-based local slip model into TMM using Modelica [14] language. In the meanwhile, the Zuber-Findlay correlation is also implemented in TMM to ensure the results are correct. This study is a high-pressure steam heating case, and the Zuber-Findlay correlation has been successfully applied in this type of problems. Therefore, the results of TMM-DL (TMM with DL-based slip closure) should show the same trend as the outcomes by TMM-ZF (TMM with the Zuber-Findlay correlation).

iv. Check if the results by TMM-DL is consistent with the original TRACE data Then check if the results show the same trend as TMM-ZF. After the comparison, we apply TMM-DL to predict the applications including four cases with different system

characters: (1) 200% power, (2) 80% mass flow rate, (3) 200% hydraulic diameter, and (4) 110% system pressure. The results by TMM-DL are compared to TMM-ZF and TRACE.

v. Compare outlet void fractions from TRACE and TMM results with Eq. (5.1), defining the relative error between TMM and TRACE for evaluating the performance of PDE-constrained forward prediction.

$$relative\,error = \frac{DATA_{TMM} - DATA_{TRACE}}{DATA_{TRACE}} \tag{5.1}$$

vi. Compare the calculated Reynolds number by TMM-DL with the training datasets to ensure that the simulation is extended into the extrapolation domain which is away from the original training domain.

The second part of this task:

i. Resample the training datasets by TRACE simulation for different locations with different total numbers of datasets including the cases of one dataset, three datasets, five datasets, and 300 datasets.

ii. Use principal component analysis (PCA) to analyze the representability of each dataset as defined in Section 3.4.4, and construct the DL-based closures by deep neural networks for the four cases in the previous step.

iii. Assign a small uncertainty to DL-based models and observe their responses. According to the Hardarmad's philosophy [85], we assume the model should continuously depend on the data meaning that a small change in the data should only result in a small change in the outputs.

The above steps show how to determine the required number of datasets that can accomplish a reliable and accurate two-phase simulation using DL-based fluid dynamics closures.

5.4. Implementation

5.4.1. Implementation of the three-equation mixture model

We implement the three-equation TMM into the Modelica Standard Fluid Library [100]. Eq. (5.2)-(5.4) give the mass-energy-momentum conservation equation where ρ, u, h, and v, are the mixture density, internal energy, enthalpy, and velocity. The l, v, A, P, α, τ_w, and P_w denote the liquid, vapor, area, pressure, void fraction, wall shear, and wetted perimeter. Eq. (5.5)-(5.6) are two-phase correction terms for the internal energy equation. Eq. (5.7) is the two-phase correction for the momentum equation. The mixture model requires a closure model for wall friction, but we do not implement two-phase correction terms for it due to the use of a fixed mass flow rate source. We further assume that there is no heat transfer, and the heat directly deposits into the fluid. As a result, we only need to develop the DL-based void-quality-slip closure to close the TMM, and we refer to this model as the TMM-DL

$$A\frac{\partial \rho}{\partial t}+\frac{\partial \rho v A}{\partial z}=0 \tag{5.2}$$

$$A\frac{\partial \rho u}{\partial t}+\frac{\partial \rho u v A}{\partial z}=-P\frac{\partial v A}{\partial z}+q'(z)+E_{1,2\Phi}+E_{2,2\Phi} \tag{5.3}$$

$$\rho\frac{\partial v}{\partial t}+\rho v\frac{\partial v}{\partial z}=-\frac{\partial P}{\partial z}+\frac{\tau_w P_w}{A}-\rho g+M_{2\Phi} \tag{5.4}$$

$$E_{1,2\Phi}=-\frac{\partial}{\partial z}\left\{\frac{\alpha_l\alpha_v\rho_l\rho_v}{\rho}\left(u_v-u_l\right)\left(v_v-v_l\right)\right\}A \tag{5.5}$$

$$E_{2,2\Phi}=-P\frac{\partial}{\partial z}\left\{\frac{\alpha_l\alpha_v\rho_l\rho_v}{\rho}\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)\left(v_v-v_l\right)\right\}A \tag{5.6}$$

$$M_{2\Phi} = -\frac{\partial}{\partial z}\left\{\frac{\alpha_l \alpha_v \rho_l \rho_v}{\rho}\left(v_v - v_l\right)^2\right\} A \tag{5.7}$$

Eq. (5.8) gives the essential wall friction correlation by the Swamee-Jain approximate solution for the Colebrook-White equation [101] where f, ε, D_{hyd}, and $\mathrm{Re}_{2\Phi}$ are the friction factor, surface roughness, hydraulic diameter, and two-phase mixture Reynolds number ($\mathrm{Re}_{2\Phi}$).

$$f = 0.25\left[\log\left(\frac{\varepsilon}{3.7 D_{hyd}} + \frac{5.74}{\mathrm{Re}_{2\Phi}^{0.9}}\right)\right]^{-2} \tag{5.8}$$

5.4.2. Implementation of slip closures

5.4.2.1. Implementation of classic slip closure

The Zuber-Findlay correlation [25] is a drift-flux void fraction model that is very successful to predict high-pressure steam flow. We code it into TMM to ensure that the implementation is correct. Eq. (5.9) shows the Zuber-Findlay correlation where the drift velocity (v_{gj}) is given by Eq.(5.10). It can work with or without flow patterns [30], and it is valid when the total volumetric flux is significantly smaller than the drift velocity [26]. Under high pressure steam conditions, the distribution parameter (C_0) is suggested to be 1.13 [102]. However, C_0 still needs to be adjusted under different system characteristics such as pressure, geometry, and perhaps mass flow rate [1]. We refer to the two-phase mixture model using the Zuber-Findlay correlation as TMM-ZF.

$$\alpha = \left[C_0\left(1 + \frac{1-x}{x}\frac{\rho_g}{\rho_l}\right) + \frac{\rho_g v_{gj}}{Gx}\right]^{-1} \tag{5.9}$$

$$v_{gj} = 1.41\left[\frac{\sigma g\left(\rho_l - \rho_g\right)}{\rho_l^2}\right]^{0.25} \tag{5.10}$$

5.4.2.2. Implementation of deep NN-based slip closure

Eq. (5.11) shows the void-quality-slip closure [1] where x is the fluid quality. The TMM requires a slip model that is the ratio of v_g to v_l to solve for void fraction distributions. We construct DL-based slip models using deep FNNs (DFNNs) with a four-layer structure. Eq. (5.12) gives the formulation of slip models with three input parameters: the local pressure, local two-phase Reynolds number, and vapor Reynolds number. Eq. (5.13) defines the local two-phase Reynolds number [103] where $G_{2\Phi}$ is the two-phase mass flux and μ is the dynamic viscosity. Eq. (5.14) defines the vapor Reynolds number. FDNNs use the sigmoidal activation functions with five hidden units in each hidden layer, and the model is implemented by Tensroflow [57] using the Adam [99] optimizer.

$$\alpha = \left(1 + \frac{1-x}{x}\frac{\rho_v}{\rho_l}S\right)^{-1} \tag{5.11}$$

$$S = DFNN(\mathbf{x}), with \quad \mathbf{x} = \left(P_{local}, \mathrm{Re}_{2\Phi,local}, \mathrm{Re}_{v,local}\right) \tag{5.12}$$

$$\mathrm{Re}_{2\Phi,local} = \frac{G_{2\Phi}D_{hyd}\left[x^2 + (1-x)^2(\rho_v/\rho_l)\right]}{\mu_v + \mu_l(1-x)(\rho_v/\rho_l)} \tag{5.13}$$

$$\mathrm{Re}_{v,local} = \frac{\rho_v v_v D_{hyd}}{\mu_v} \tag{5.14}$$

Table 5.2 gives parameters of the DL-based slip closure with the activation function, sigmoid. Eq. (5.15) shows the structure of the first hidden layer in the DL model where j is the j^{th} hidden unit and i is the i^{th} training input. Eq. (5.16) gives the second and third hidden layer where k indicates the layer number. Starting from the second hidden layer, the total number of inputs depends on the number of hidden units from the previous layer. Eq. (5.17) shows the output layer of the DFNN which is a linear combination of the hidden units from the last hidden layer.

Table 5.2. Deep neural networks' hyperparameters.

Parameters	Values
Activation function	Sigmoid
Number of HU in each HL	5
Number of HL	3
Number of inputs	3
Number of outputs	1

$$HU_{1j}(\mathbf{x}) = sigmoid(\sum_{i=1}^{3} w_{1ji} x_i + b_{1j}) \tag{5.15}$$

$$HU_{kj}(HU_{k-1}) = sigmoid(\sum_{i=1}^{5} w_{kji} HU_{k-1,i} + b_{kj}) \tag{5.16}$$

$$DFNN_{out}(HU_{K-1}) = \sum_{i=1}^{5} w_{oji} HU_{K-1,i} + b_o \tag{5.17}$$

5.4.3. Implementation of two-phase flow modeling by Type I ML

Algorithm CHAPTER 5.1 shows the procedure of utilizing Type I ML to develop slip closures. Training data are obtained from the two-fluid model (TFM) [1]. We assume that there are invisibly tiny bubbles moving with the liquid when single-phase flows present. Therefore, we need to constrain the DL-based slip to have the minimum output equal to one. Then we implement the DL-based slip closure into the TMM for predictions. Eq. (5.18) defines the relative error to evaluate the performance of Type I ML.

Algorithm CHAPTER 5.1 Type I ML for the system-level TFS.

Input: training inputs (P, $\mathrm{Re}_{2\Phi}$, and Re_v) by the TFM (*element 3* in Figure 3.7)

training target ($S = v_g / v_l$) (*element 4* in Figure 3.7)

Output: TMM with DL-based slip closure for predictions (*element 7* in Figure 3.7)

1: **for** i < maximum_iteration **do** (*element 5* in Figure 3.7)

2: // Build a slip model using DFNNs

3: $S(P, \mathrm{Re}_{2\Phi}, \mathrm{Re}_v) \leftarrow DFNN$

4: **for** all inputs $\in$ training datasets **do**

5: Update the parameters for each layer in DFNN with inputs

6: Constrain the value of *DFNN* based on the model assumption: (*element 6* in Figure 3.7)

$DFNN \leftarrow \max(1, DFNN)$

7: Solve TMM with DL-based slip closure: (*element 7* in Figure 3.7)

$$\alpha = \left[1 + \frac{1-x}{x}\frac{\rho_g}{\rho_l} S(R, \mathrm{Re}_{2\Phi}, \mathrm{Re}_v)\right]^{-1}$$

$$\varepsilon_r = \frac{\alpha_{TMM} - \alpha_{TFM}}{\alpha_{TFM}} \tag{5.18}$$

5.5. Manufacturing synthetic data for Type I ML

We use the two-fluid model implemented by USNRC TRACE [10] to provide the synthetic data for building the void-quality-slip closure. We generate one baseline training dataset using the boiling channel characteristics given in Table 5.3. Then we create another eight validating datasets with distinct system characteristics.

Table 5.3. Characteristics of the boiling channel.

Parameters	Values
Channel heat flux (J/sec-m^2)	4.5436 x 10^5
Outlet pressure (bar)	67.73
Coolant mass flux (kg/sec-m^2)	1925.85
Inlet temperature (K)	550.93
Heated diameter (m)	0.0125
Heated length (m)	3.7084
Flow area (m^2)	1.41096 x 10^{-4}

5.6. Results analysis by using TMM-DL to predict various system characteristics

5.6.1. Using TMM-DL to predict various system characteristics

Figure 5.2 and Figure 5.3 depict the void fraction and slip factor comparisons between the TMM-DL and TFM at the outlet for various system characteristics. When the system reaches the steady state at 720 seconds, a heater maintains a constant heat flux. Figure 5.2 also shows the results by TMM-ZF. Without tuning the distributed parameter, TMM-ZF yields a substantial discrepancy when we compare the results to TFM.

For the baseline case in Figure 5.2, the relative error between the TMM-DL and TFM is 6.8% at the steady state. As we double the power or reduce the power by half, the relative errors were below 5%. When we increase the mass flow rate (MFlow) by 20%, the relative error becomes 8.5%. However, the error goes down to 4.7% as we decrease the MFlow to 80%. The relative error is 0.7% for the case with the hydraulic diameter (D_{hyd}) increased by the factor of 2, but the slip factor deviates about 10% from TFM results. The error becomes 3.2% when we set the D_{hyd} equal to half of the baseline case. For the 110% pressure case, the relative error is 4.2%. Finally, the significant error (11.5%) occurs for the 95% pressure case.

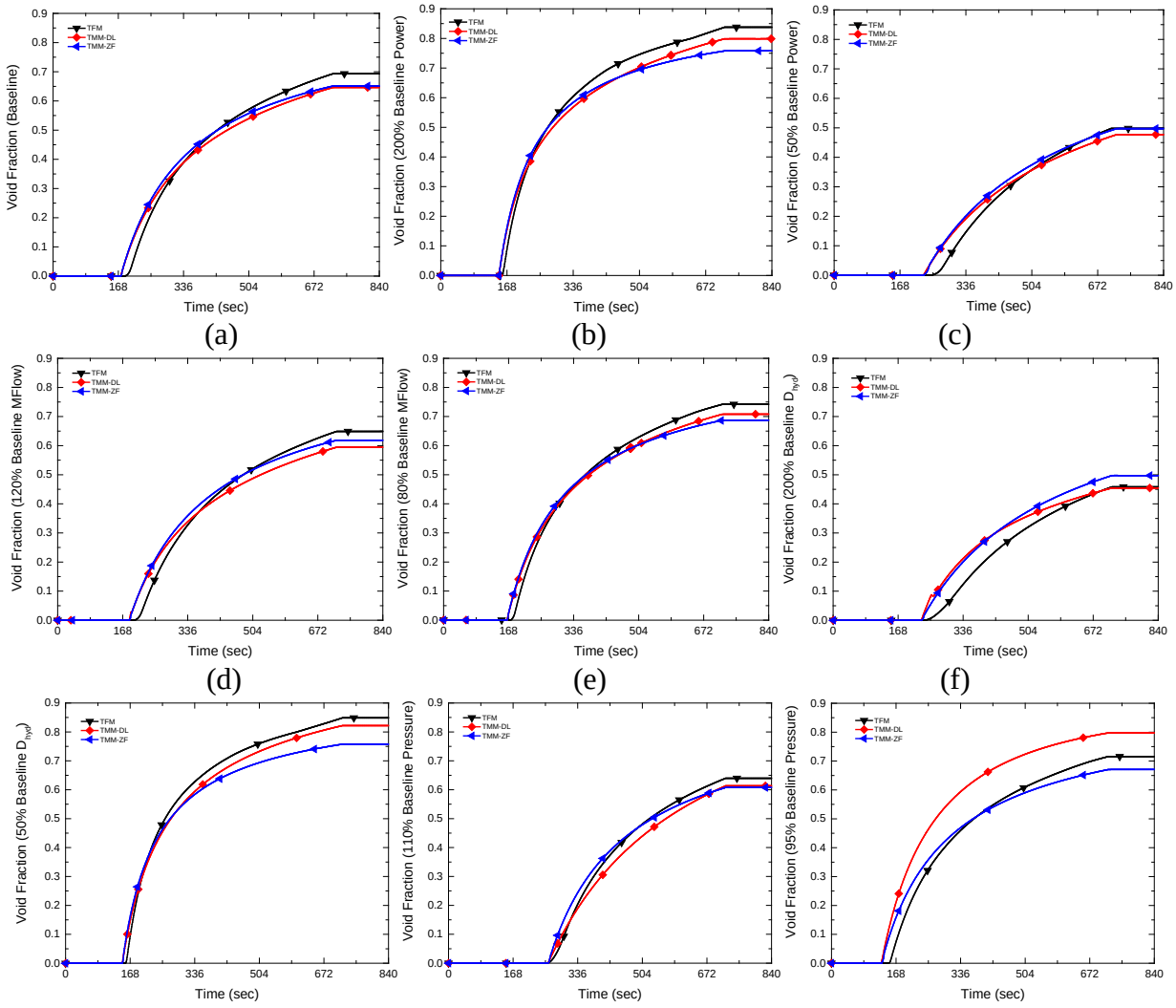


Figure 5.2. Comparison of void fraction at the pipe outlet for TFM, TMM-DL, and TMM-ZF for various system characteristics such as (a) the baseline, (b) 200% baseline power, (c) 50% baseline power, (d) 120% baseline mass flow rate (MFlow), (e) 80% baseline MFlow, (f) 200% baseline D_{hyd}, (g) 50% baseline D_{hyd}, (h) 110% baseline pressure, and (i) 95% baseline pressure.

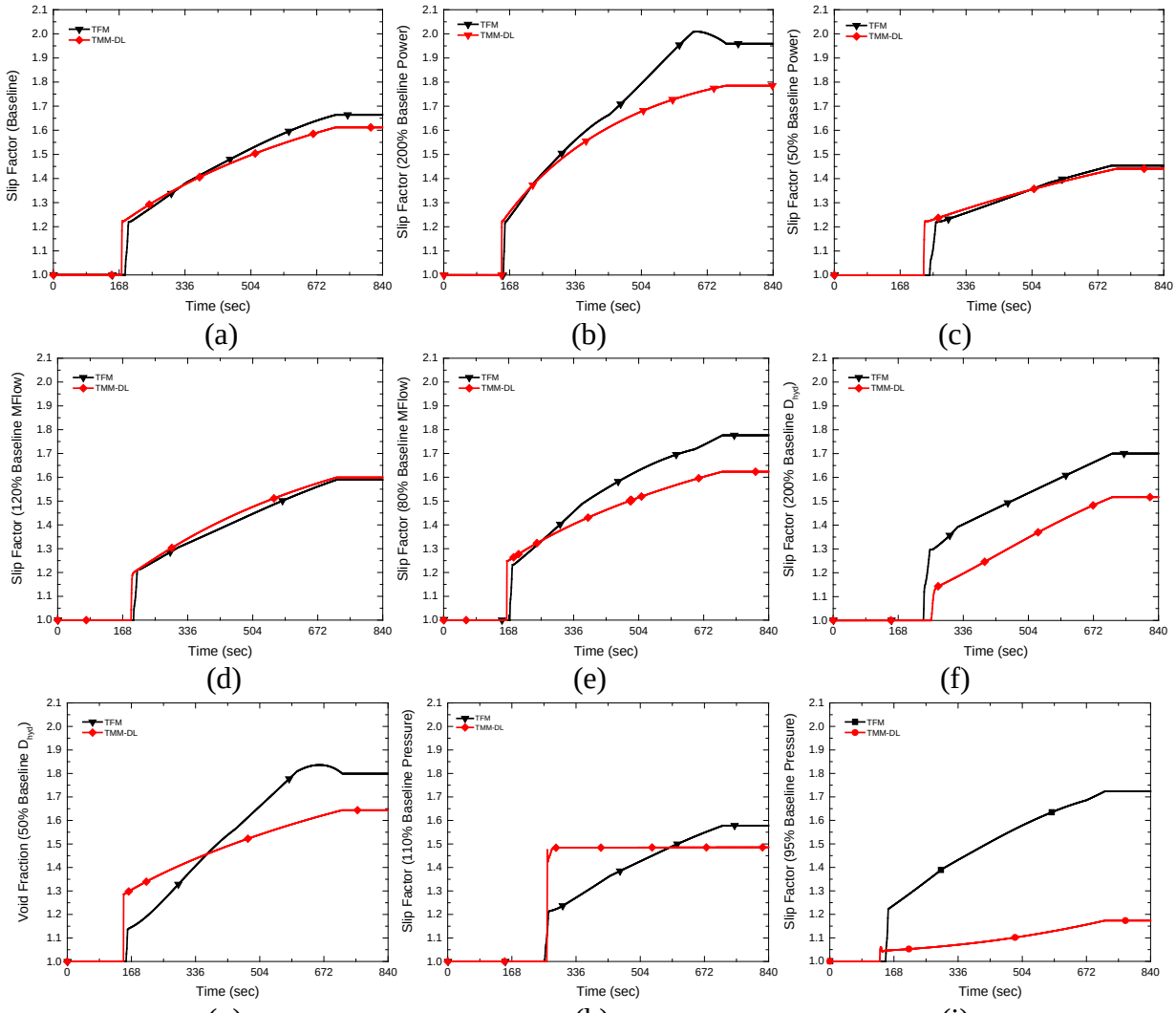


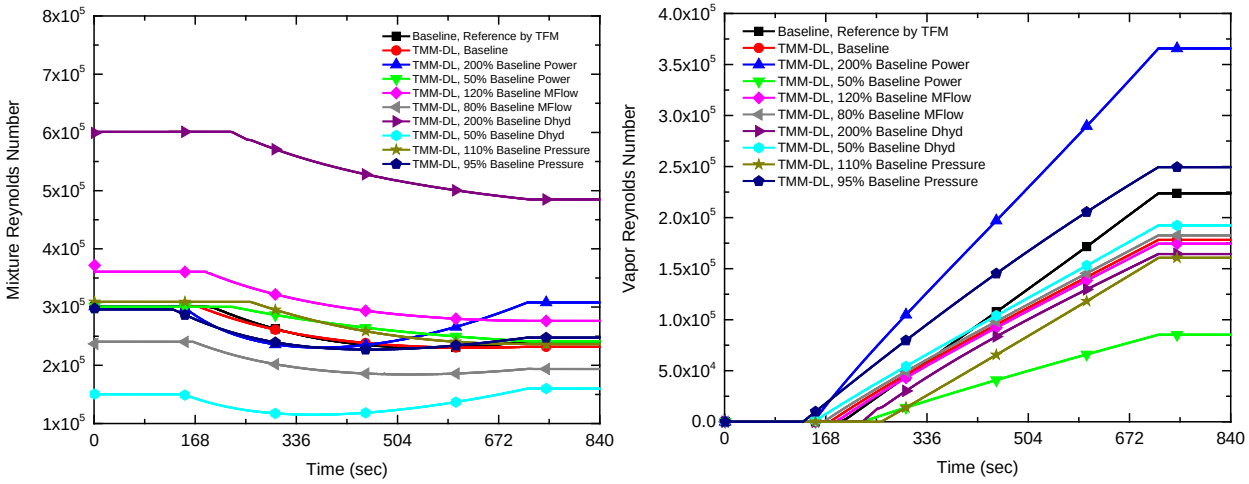
Figure 5.3. Comparison of slip factor at the pipe outlet between TFM and TMM-DL for various system characteristics such as (a) the baseline, (b) 200% baseline power, (c) 50% baseline power, (d) 120% baseline mass flow rate (MFlow), (e) 80% baseline MFlow, (f) 200% baseline D_{hyd}, (g) 50% baseline D_{hyd}, (h) 110% baseline pressure, and (i) 95% baseline pressure.

Table 5.4 gives the relative errors of void fractions between the TFM and TMM-DL at the steady-state. The case with 200% baseline Dhyd yields the minimum relative error while the case with 95% baseline pressure causes the maximum error. Figure 5.4 depicts the inputs, $Re_{2\Phi}$ and Re_v, of the DL-based slip closure for each test at a different time to ensure that the application is beyond the training domain. At the steady state, the $Re_{2\Phi}$ for all cases are outside the training domain. However, the steady-state Re_v numbers for most cases are within the training domain

except for the 200% baseline power and 95% baseline pressure cases. Figure 5.4 indicates that the input set ($Re_{2\Phi}$, Re_v) are beyond the training dataset. Therefore, the DL-based closure has scaling capability for this problem.

Table 5.4. Relative errors of void fractions between TMM-DL and TFM results.

Case	Relative Errors of Void Fraction
Baseline	-6.8%
200% Baseline Power	-4.7%
50% Baseline Power	-4.2%
120% Baseline MFlow	-8.5%
80% Baseline MFlow	-4.7%
200% baseline D_{hyd}	-0.7%
50% baseline D_{hyd}	-3.2%
110% Baseline Pressure	-4.2%
95% Baseline Pressure	11.5%



(a) (b)

Figure 5.4. Comparison of the inputs, (a) $Re_{2\Phi}$ and (b) Re_v, for the DL-based slip model with simulations under different system characteristics including the baseline, 200% baseline power, 50% baseline power, 120% baseline mass flow rate (MFlow), 80% baseline MFlow, 200% baseline D_{hyd}, 50% baseline D_{hyd}, 110% baseline pressure, and 95% baseline pressure to ensure that the applications is beyond the training domain.

5.6.2. Exploring DL uncertainty by different data quantities

Inferring models from data belongs to inverse problems which are ill-posed. We use the recovery factor defined in Section 3.4.4 to explore whether a DL-based closure satisfies the data quantity requirement. Table 5.5 gives the recovery factors by Eq. (3.8) for cases with various numbers of training datasets. The recovery factor implies how good the selected training datasets can represent the reference data which include 300 datasets in this problem. Table 5.5 indicates that the one dataset case does not represent the initial training datasets well.

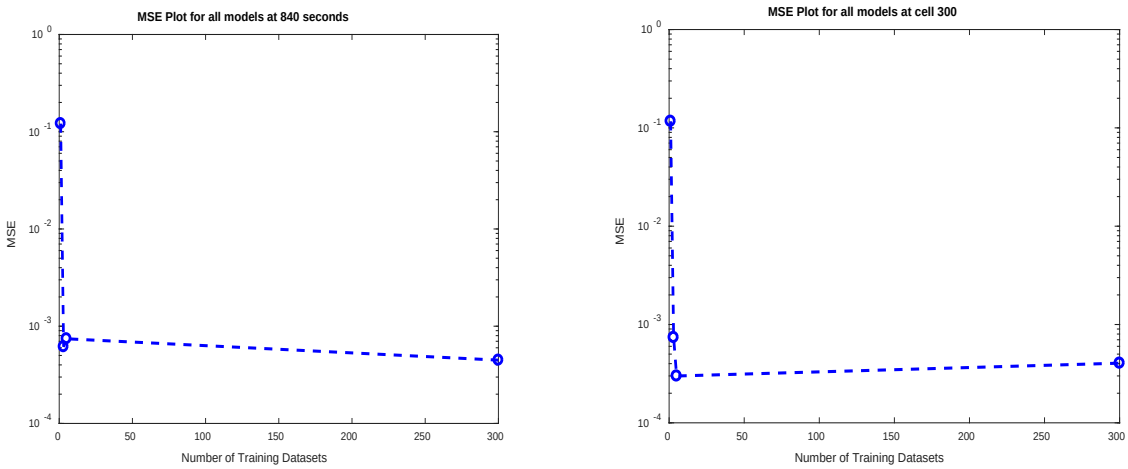
Table 5.5. Recover factors for different training datasets of inputs and target.

	Pressure	ReMix	Re	Slip Factor
1 Dataset	0.99	0.18	0.94	0.29
3 Datasets	0.99	0.99	0.99	0.71
5 Datasets	1.00	0.99	1.00	0.88
300 Datasets	1.00	1.00	1.00	1.00

To explore which model is well-posed, we add uncertainties to the datasets in Table 5.5, and we observe the response of the outputs from DL models. We sample 1000 random variables from $N(0.01, 0.001)$. Then we generate the new inputs with uncertainties by Eq. (5.19) where $\mathbf{x}$ is the input vector and ε is random numbers. Since the training data are from the transient problem, the following figures will show two cases: (i) figures for all spatial cells at steady state, and (ii) figures for the outlet cell (cell 300) for all time steps. We compare the mean square error and relative error between the model with training inputs and the model with uncertain inputs.

$$\mathbf{x}_{new} = \mathbf{x}(1+\varepsilon) \tag{5.19}$$

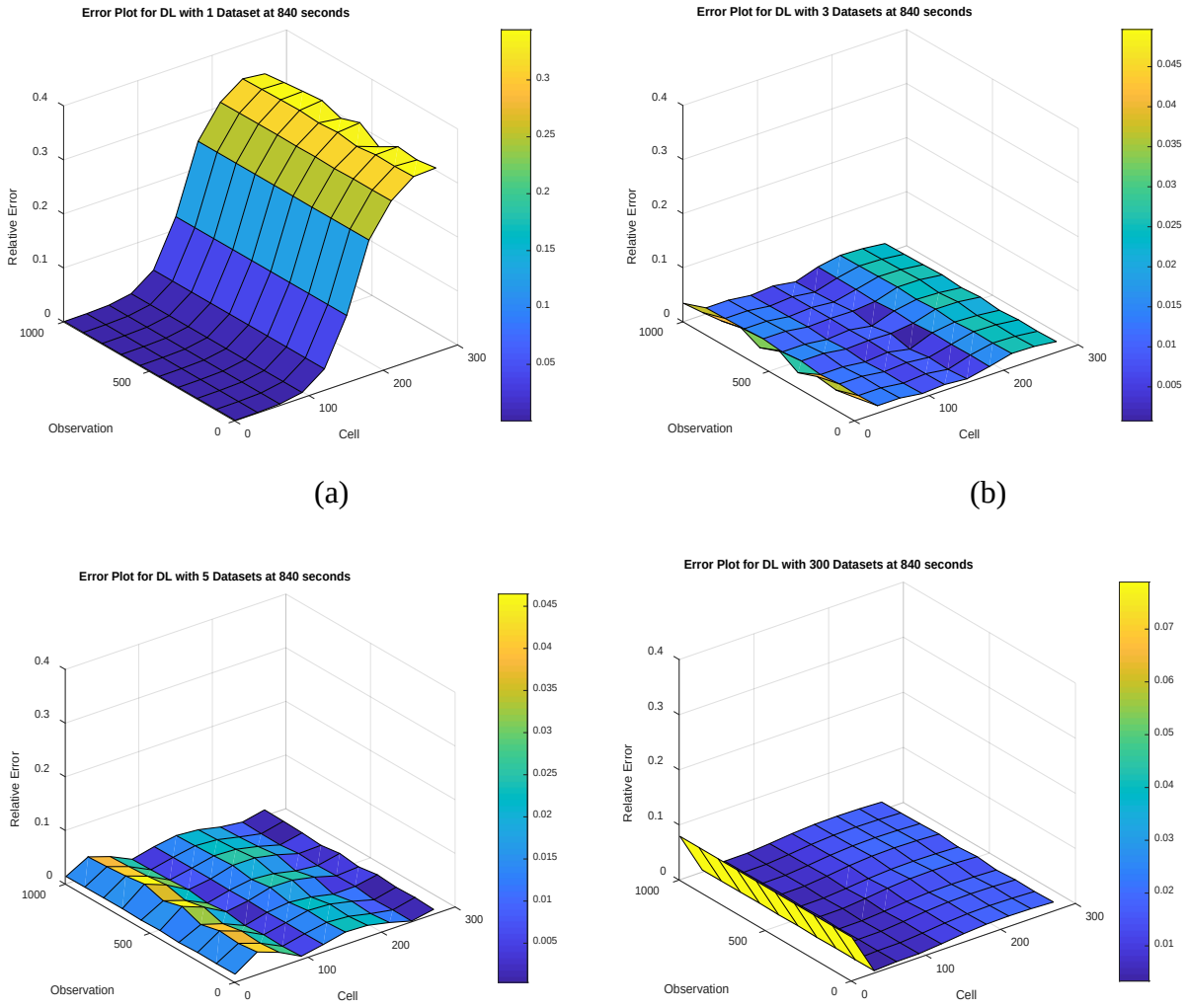
Figure 5.5(a) shows the mean square error (MSE) of four DL-based slip models trained by different numbers of datasets for all spatial cells at the steady state. Figure 5.5(b) depicts the MSE for the outlet cell for all time steps. Both figures indicate that the DL-based slip model by one dataset yields the largest error versus other cases.



(a) (b)

Figure 5.5. Mean square errors of all DL-based slip models by different training dataset by comparing (a) all vertical cells at steady state and (b) outlet cell (cell 300) for all time steps.

Figure 5.6 shows the 3D relative error plots of all DL-based slip models at steady state, and Figure 5.7 depicts the 3D relative error plots of the outlet cell for all time steps. From these figures, when we assign a small uncertainty (1%) to the inputs, the DL-based slip model trained by one dataset results in a large relative error (~35%) to the training data. Therefore, we consider that the DL-based slip model trained by one dataset is not well-posed.



(a) (b)

(c) (d)

Figure 5.6. 3D plot showing the relative error of all spatial cells at steady state between DL-based slip models using the training inputs and the inputs with uncertainties for the DL-based slip model trained by (a) 1 dataset, (b) 3 datasets, (c) 5 datasets, and (d) 300 datasets.

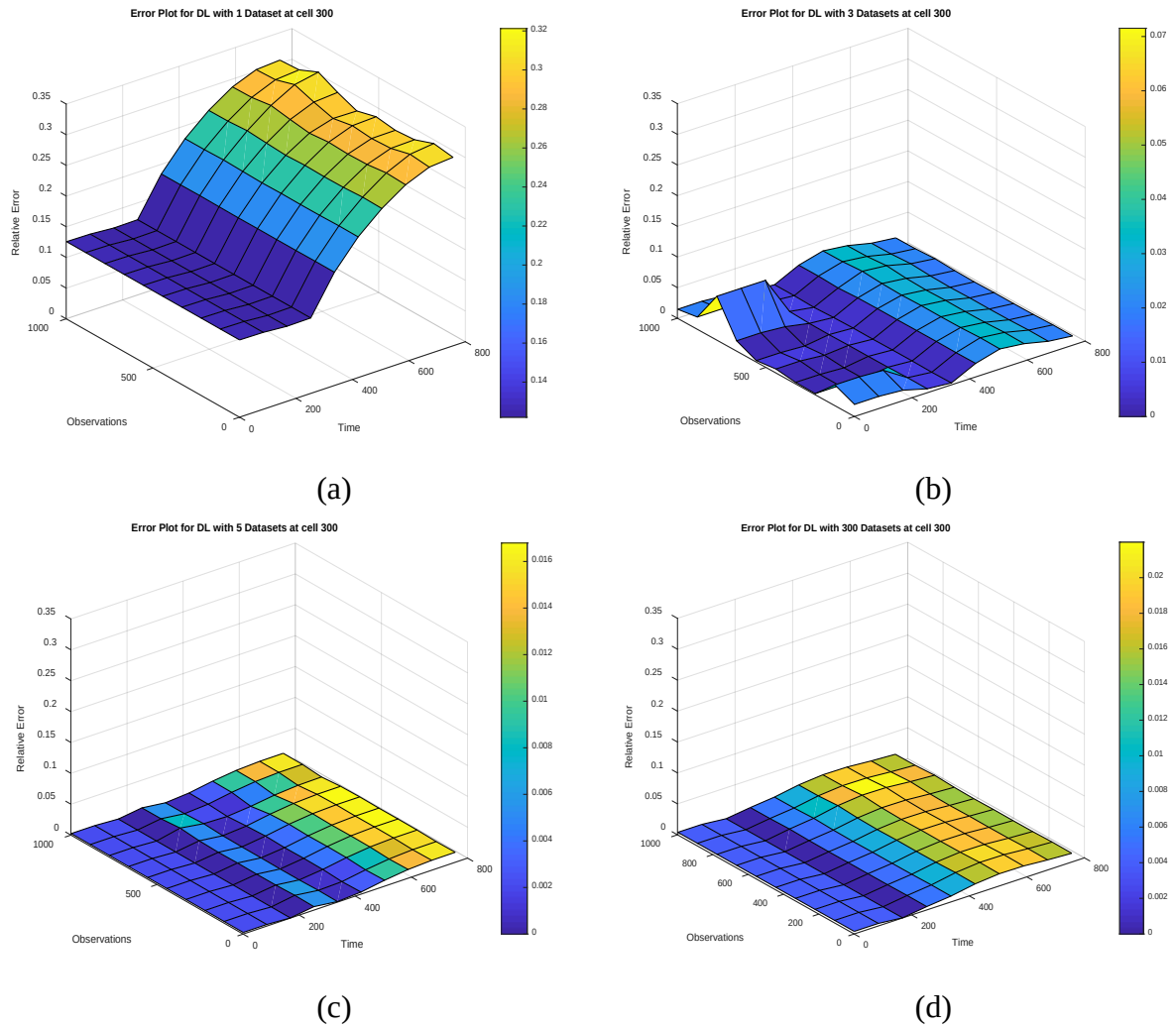
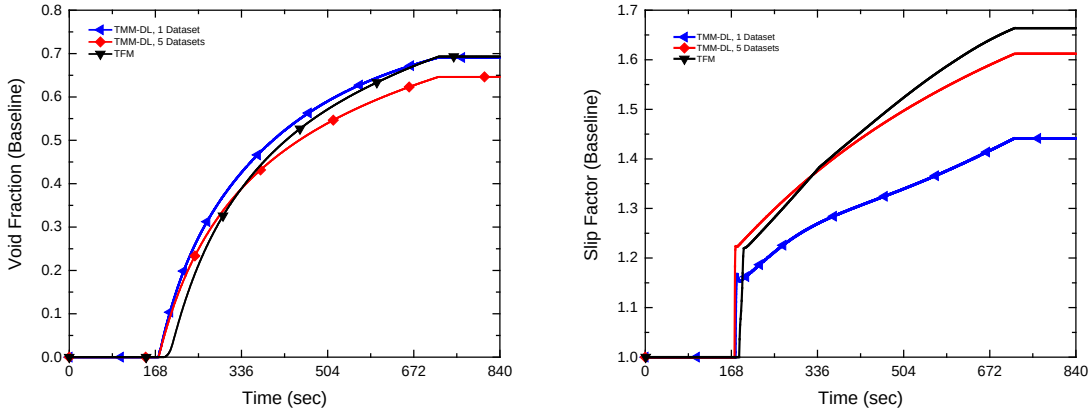


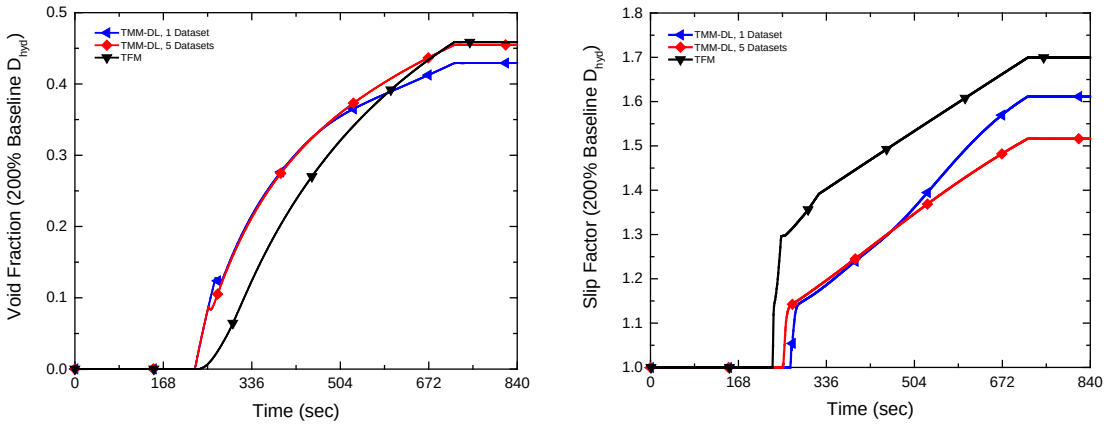
Figure 5.7. 3D plot showing the relative error of outlet cell (cell 300) for all time steps between DL-based slip models using the training inputs and the inputs with uncertainties for the DL-based slip model trained by (a) 1 dataset, (b) 3 datasets, (c) 5 datasets, and (d) 300 datasets.

Figure 5.8-Figure 5.10 depict the void fraction comparison at pipe outlet for TMM-DL trained by one dataset, TMM-DL trained by five datasets, and TFM for three different system characteristics including original experiment conditions, 200% hydraulic diameter, and 200% power. Although one-dataset TMM-DL agrees with the baseline, its prediction is unstable because it performs worse than five-dataset TMM-DL for 200% hydraulic diameter. Furthermore, one-dataset TMM-DL fails the PDE simulation for the 200% power case. Whenever the uncertainty for inputs exceeds its tolerance limit, the TMM-DL cannot make a successful simulation.



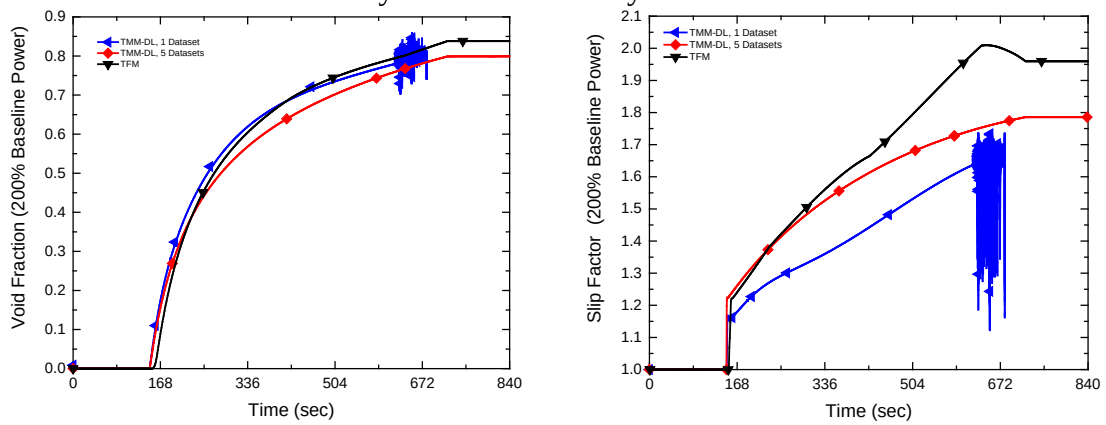
(a) (b)

Figure 5.8. Comparison of outlet void fractions at the outlet for TMM-DL trained by one dataset, TMM-DL trained by five datasets, and TFM with baseline conditions.



(a) (b)

Figure 5.9. Comparison of outlet void fractions at the outlet for TMM-DL trained by one dataset, TMM-DL trained by five datasets, and TFM with original experiment parameters but increasing the hydraulic diameter by factor of 2.



(a) (b)

Figure 5.10. Comparison of outlet void fractions at the outlet for TMM-DL trained by one dataset, TMM-DL trained by five datasets, and TFM with original experiment parameters but increasing the power by factor of 2.

5.7. Lessons learned

Type I ML can achieve the cost-effective development of the DL-based slip closure that is ubiquitous across flow regimes from the single-phase flow to flow boiling. The information of flow regime transitions implicitly inherits from the data. For this case study, the DL-based slip closure exhibits predictability beyond the training domain. Specifically, the mixture model can make predictions within a reasonable uncertainty range by comparing the results to the TFM for various system characteristics outside the training domain. However, further investigations are needed for a broader range of system conditions as well as for datasets generated by experiments. Presently synthetic data are limited to the TRACE simulation that is based on certain assumptions and models. It is noted that caution must be exercised in applying the synthetic data because they may have been biased by the previous calibration of models. The analysis conducted for the example of the TMM-DL modeling suggests that the performance of data-driven models may be affected by model biases. Therefore, the evaluation may be hampered by various sources of uncertainties including model forms and numerical errors. For instance, the drag force model in the TRACE TFM is inherited from the drift-flux model [10], whereas the tested mixture model is based on the phasic velocity slip formulation.

5.8. Summary

The two-phase flow case study also demonstrates how to employ the hypothesis and recovery factor defined in Section 3.4.4 with Type I ML framework. The case study suggests that RF has the potential to be a screening criterion to find a robust DL model that satisfies the requirement of well-posedness. The hypothesis states that the output of DL models should continuously depend on the inputs. This means a small change in the inputs should only result in

a small change in the output. Otherwise, the DL-based closures become ill-posed and cannot be used for Type I ML problems. It is found that the DL-based slip model (closure relation) exhibits prediction beyond the training dataset. Specifically, the three-equation two-phase mixture model can predict test cases within reasonable uncertainty ranges. However, further investigations are needed for a broader range of system conditions as well as for training data collected from experiments.

CHAPTER 6. CASE STUDY C: FRAMEWORK COMPARISON

6.1. Introduction

The heat conduction case study is formulated to demonstrate how to employ Type I, Type II, Type III, and Type V ML to build ML-based thermal conductivity and to compare results by each framework. Chanda *et al.* used ANN with genetic algorithm [104] to solve inverse modeling for heat conduction problems. In this work, deep learning (DL) [4] is selected as the ML methodology in this task. Principally, any neural network (NN) with more than two layers (one hidden layer with an output layer) is considered to be DL [46]. Hornik [44] proved that multilayer NNs are universal approximators and can capture the properties of any measurable information. This capability makes DL attractive for the closure development in thermal fluid simulation. Notably, we implement NN-based thermal conductivity by FNNs and convolutional neural networks (CNNs) to evaluate the performance of closure relations by distinct NNs.

6.2. Objectives

Type III ML is first defined and introduced in this dissertation. The first objective is to demonstrate how to use Type III ML to achieve data-driven modeling. The results are compared to Type-I, Type II, and Type-V ML to show the advantage of Type III ML. The second objective is to show that the solution is numerically stable while solving the PDE with CNN-based closure models.

6.3. Problem formulation

We formulate the synthetic task using a 2D (two-dimensional) heat conduction model given by Eq. (6.1) where $k(T)$ is a nonlinear thermal conductivity. To generate training data, Eq.

(6.2) shows a temperature-dependent model for $k(T)$ where c, σ, and μ are constant parameters. Table 6.1 gives two parameter sets (baseline and prior sets) to generate data. While demonstrating ML frameworks, $k(T)$ becomes NN-based thermal conductivity.

$$\frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right]+\frac{\partial}{\partial y}\left[k(T)\frac{\partial T}{\partial y}\right]=0 \tag{6.1}$$

$$k(T)=\frac{c}{\sigma\sqrt{2\pi}}e^{-\frac{(T-\mu)^2}{2\sigma^2}} \tag{6.2}$$

Table 6.1. Two parameter sets for the thermal conductivity model.

Dataset	c (W/m)	σ (K)	μ (K)
Baseline set for producing synthetic data	$7.2\text{x}10^4$	300	1200
Prior set for producing inputs required by Type II ML	$7.2\text{x}10^4$	600	2100

Two numerical experiments are designed to emulate IETs and SETs for manufacturing synthetic data by solving Eq. (6.1) using parameters sets in Table 6.1. IETs provide field data, for instance, 2D temperature fields by an infrared camera. SETs offer global data such as a 1D measurement by thermocouples. Synthetic data are used for training and validating NN-based thermal conductivity. Type I ML can only use SET data because of a scale separation assumption. Type II ML can only use SET data because the goal is to improve the prior thermal conductivity by the baseline. Type III and Type V ML use field data. We compare Type I and Type II ML using training data from SETs. Then Type III and Type V ML are compared by field data from IETs.

6.4. Manufacturing synthetic data for ML frameworks

Numerical solutions assume piecewise-linear temperature between grids [105] with Dirichlet type boundaries. We further assume conductivity profiles are also piecewise-linear between mesh points.

6.4.1. Manufacturing IET data

IETs are measurements of temperature fields. Synthetic IET data are generated by Eq. (6.1) with the baseline set in Table 6.1. Figure 6.1 illustrates the layout of IET experiments with four constant boundary temperatures. We change T_{west} for various observations and fix the boundary temperature (1300K) at the east side. The T_{north} and T_{south} are linearly dependent on the west boundary condition. We prepare three training datasets by including distinct data quantities and three validating datasets by changing T_{west}. Table 6.2 gives the metadata of each training or validating dataset. All observations are uniformly sampled within a given temperature range.

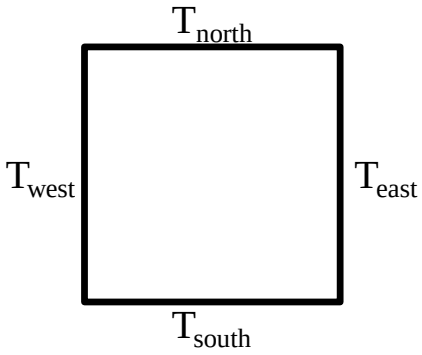


Figure 6.1. Schematic of integral effects tests (IETs) for measuring temperature fields.

Table 6.2. Summary of IET training and validating datasets.

Dataset	Data Quantity	Temperature Range at T_{west}	Description
T1	11 observations	[1000K, 1100K]	Training dataset
T2	100 observations	[1000K, 1100K]	Training dataset
T3	1000 observations	[1000K, 1100K]	Training dataset
P1	1000 observations	[1000K, 1100K]	Validating dataset
P2	1000 observations	[900K, 1000K]	Validating dataset
P3	1000 observations	[800K, 900K]	Validating dataset

6.4.2. Manufacturing SET data

SETs are global measurements by thermocouples. Figure 6.2 depicts the layout of SETs for obtaining mean temperature and heat conductivity data. A heater is on top of the sample to

maintain a constant temperature (T_H). Thermal insulation is installed on the outside surface. The coolant at the bottom removes the heat with a constant heat transfer coefficient. Eq. (6.3) calculates the temperature profiles within the sample using the parameter sets in Table 6.1. Eq. (6.4) calculates the observed heat conductivity (k_{obs}), and the mean temperature is obtained by arithmetic averaging T_H and T_C.

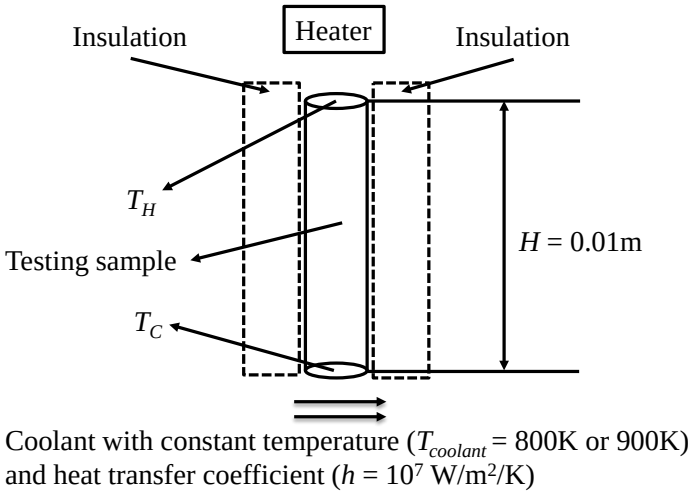


Figure 6.2. Schematic of separate effects tests (SETs) for measuring thermal conductivity as the function of sample's mean temperature.

$$\frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right]=0 \tag{6.3}$$

$$k_{obs}\frac{T_H-T_C}{H}=h(T_C-T_{coolant}) \tag{6.4}$$

We generate two training datasets with two coolant temperatures to explore the effect by different data qualities. Table 6.3 shows the metadata of SET datasets. A large temperature gradient across the testing sample increases the nonlinearity of temperature profiles. For each training set, we uniformly sample 41 T_H from Eq. (6.5) to keep mean temperatures in SETs within the same range as IETs.

Table 6.3. Summary of SET training datasets.

Dataset	Data Quantity	Data Quality	$T_{coolant}$ (K)	Description
S1	41 observations	Low	800	Training dataset
S2	41 observations	High	900	Training dataset

$$\left(T_{H,\max}, T_{H,\min}\right) = \left(2T_{IET,\max} - T_{coolant}, 2T_{IET,\min} - T_{coolant}\right) \tag{6.5}$$

6.5. Implementation

6.5.1. Implementation of the heat conduction task by different ML frameworks

We present Type I ML in Algorithm 6.1. SET data are generated by Eq. (6.3) with the baseline set in Table 6.1. Inputs and targets are temperatures and thermal conductivities. After the training, FNN-based thermal conductivity is implemented in Eq. (6.1) for predictions.

Algorithm 6.1. Type I ML for 2D heat conduction problem with Dirichlet BC.

Input: Training inputs (T_{data}, *element 3* in Figure 3.7) and training targets (k_{data}, *element 4* in Figure 3.7) from the SET (*element 1* in Figure 3.7)
Output: Temperature fields for predictions (*element 7* in Figure 3.7)
1: **for** all epochs < maximum_epoch **do** (*element 5* in Figure 3.7)
2: // Build a conductivity model using FNNs
3: $k(T) \leftarrow FNN(T)$
4: **for** all inputs $(T_{data}, k_{data}) \in$ training datasets **do**
5: Update hyperparameters for each layer in FNNs
6: Implement $k(T)$ into Eq. (6.1) (*element 7* in Figure 3.7)
7: Solve Eq. (6.1) with Dirichlet boundaries (*element 7* in Figure 3.7)

Algorithm 6.2 outlines the application of Type II ML. SET data are generated by Eq. (6.3) with prior and baseline sets in Table 6.1. Targets are thermal conductivities calculated by the baseline set. However, inputs are mean temperatures by the prior set. After the training, FNN-based thermal conductivity can be queried by new input temperature fields from solutions with new boundary conditions. Once thermal conductivities are obtained, they are implemented as fixed

fields into Eq. (6.1) to obtain temperature profiles for predictions. Therefore, Type II ML first solves the heat conduction equation with an initially guessed parameter set, and then improves the solution by the FNN-based closure.

Algorithm 6.2. Type II ML for 2D heat conduction problem with Dirichlet BC.

Input: Training inputs (T_{prior}) (*element 4* in Figure 3.8) and training targets ($k_{baseline}$, *element 5* in Figure 3.8) from the SET (*element 2* in Figure 3.8)
Output: Temperature fields for predictions (*element 10* in Figure 3.8)
1: Solve Eq. (6.3) with the prior set in Table 6.1 (*element 1* in Figure 3.8)
2: Use temperature profiles (T_{prior}) as inputs for training NNs (*element 4* in Figure 3.8)
3. **for** all epochs < maximum_epoch **do** (*element 6* in Figure 3.8)
4: // Build surrogates for thermal conductivities using FNNs
5: $k(T) \leftarrow FNN(T)$
6: **for** all inputs $(T_{prior}, k_{baseline}) \in$ training datasets **do**
7: Update weights and biases for each layer in FNNs
8: Solve Eq. (6.1) again with the prior set in Table 6.1 and new boundaries (*element 7* in
9: Figure 3.8)
10: Query $k(T'_{prior})$ by new temperature fields (T'_{prior}) (*element 8* in Figure 3.8)
11: Implement the thermal conductivities as fixed fields into Eq (6.1), and solve the equation to
12: obtain temperature profiles for predictions (*element 9* in Figure 3.8)

Algorithm 6.3 outlines the procedure of Type III ML. IET field data are generated by Eq. (6.1) with the baseline set in Table 6.1. Targets are baseline temperature fields. Inputs are the solution by Eq. (6.1) with NN-based thermal conductivity. During the training, inputs are iteratively updated due to the change of weights and biases in NNs. After the training, the model is ready for predictions. Type III ML ensures data-model consistency because PDEs are involved in the training.

Algorithm 6.3. Type III ML for 2D heat conduction problem with Dirichlet BC.

Input: Training targets ($T_{baseline}$, *element 3* in Figure 3.9) and training inputs (T_{PDE}, *element 4* in Figure 3.9)
Output: Temperature fields for predictions
1: **for** all outer < maximum_outer_iteration **do**
2: **for** all epoch < maximum_epoch **do** (*element 5* in Figure 3.9)
3: // Build conductivity models using both FNNs and CNNs
4: $k(T) \leftarrow FNN(T),\ CNN(T)$
5: **for** all inputs $T_{PDE} \in$ PDE solutions **do**
6: Update hyperparameters for each layer in FNNs and CNNs
7: Solve Eq. (6.1) using NN-based thermal conductivity for predictions (*element 6* in
8: Figure 3.9)

Algorithm 6.4 shows the implementation of Type V ML using IET data. The data are generated by Eq. (6.1) with prior and baseline sets in Table 6.1. Inputs are temperate profiles by the prior set. Outputs are the discrepancies (δT) between prior and baseline temperature fields. After the training, discrepancy fields are queried by new temperature fields with the prior set and new boundary conditions. Then the improved temperature fields are obtained by adding discrepancy fields into prior temperature fields. Therefore, the predicted temperature fields by Type V ML are not constrained by the governing equation.

Algorithm 6.4 Type V ML for 2D heat conduction problem with Dirichlet BC.

Input: baseline temperature ($T_{baseline}$) and training inputs (T_{prior})
Output: Temperature fields for predictions

1: Solve Eq. (6.1) with the prior set in Table 6.1
2: Use temperature profiles (T_{prior}) as inputs for training NNs
3: Compute discrepancies between baseline temperature fields ($T_{baseline}$) and prior temperature
4: fields (T_{prior})
5. $\delta T = T_{baseline} - T_{prior}$
6. **for** all epochs < maximum_epoch **do**
7: // Build surrogates for thermal conductivities using FNNs
8: $\delta T \leftarrow FNN(T_{prior})$
9: **for** all inputs $(T_{prior}, \delta T) \in$ training datasets **do**
10: Update weights and biases for each layer in FNNs
11: Solve Eq. (6.1) again with the prior set in Table 6.1 and new boundaries
12: Query $\delta T(T'_{prior})$ by new temperature fields (T'_{prior})
13: Improve temperature profiles for predictions by $\delta T(T'_{prior})$

Eq. (6.6) defines the root-mean-square error (RMSE) to evaluate the performance of each ML framework where *i* denotes the ith observation, *N* is the total number of data points in the ith observation, and *j* presents the jth solution. For each validating dataset, we calculate mean RMSE by using arithmetic averaging.

$$RMSE_i = \sqrt{\frac{\sum_j \left(T_{Model,j} - T_{data,j}\right)^2}{N_i}} \tag{6.6}$$

6.5.2. Implementation of NN-based thermal conductivity model

6.5.2.1. FNN-based thermal conductivity model

We use FNNs and CNNs to reconstruct thermal conductivity from training data. Eq. (6.7) gives the formulation of FNN-based thermal conductivity where *T*, **x**, and *i* are the temperature, input vector, and ith training input. The sigmoidal activation function, $1/(1+e^{-x})$, is selected. Eq. (6.8) shows the structure of the first hidden layer (HL) where *j* is the jth hidden units (HUs) and N_{in}

is the total number of inputs from the input layer. The weight (w) and bias (b) are parameters to be learned based on training data. Eq. (6.9) presents the general structure of HLs where k and N_{HUk} are the layer number and total number of HUs in the kth layer. Starting from the second HL, the number of inputs depends on the quantity of HUs from the previous HL. Eq. (6.10) shows the output layer as a linear combination of HUs from the last HL where L is the total layer number of FNNs. For this demonstration, we use three-layer FNNs with ten HUs in each HL, and we fix this structure for all types of ML learning frameworks.

$$k_{DNN} = FNN(\mathbf{x}), with \quad x_i = (T_i) \tag{6.7}$$

$$HU_{1j}(\mathbf{x}) = sigmoid\left(\sum_{i=1}^{N_{in}} w_{1ji} x_i + b_{1j}\right) \tag{6.8}$$

$$HU_{kj}(\mathbf{x}) = sigmoid\left(\sum_{i=1}^{N_{HU_k}} w_{kji} HU_{k-1,i} + b_{kj}\right) \tag{6.9}$$

$$FNN(\mathbf{x}) = \sum_{i=1}^{N_{HU_{L-1}}} w_{o,i} HU_{L-1,i} + b_0 \tag{6.10}$$

6.5.2.2. CNN-based thermal conductivity model

Figure 6.3 depicts the architecture [47] of CNN-based thermal conductivity that includes three convolutional layers and three fully connected layers. We use the ReLU [51] activation for layers in CNNs to accelerate the training. Inputs are temperature fields. After the first convolutional layer, eight feature maps are generated, and each feature map detects the patterns from temperature fields. The second convolutional layer takes inputs from the previous layer, and it outputs 12 feature maps. The third convolutional layer receives inputs from the previous layer, and it delivers 24 feature maps to fully connected layers. Finally, we obtain thermal conductivity fields from CNN's outputs.

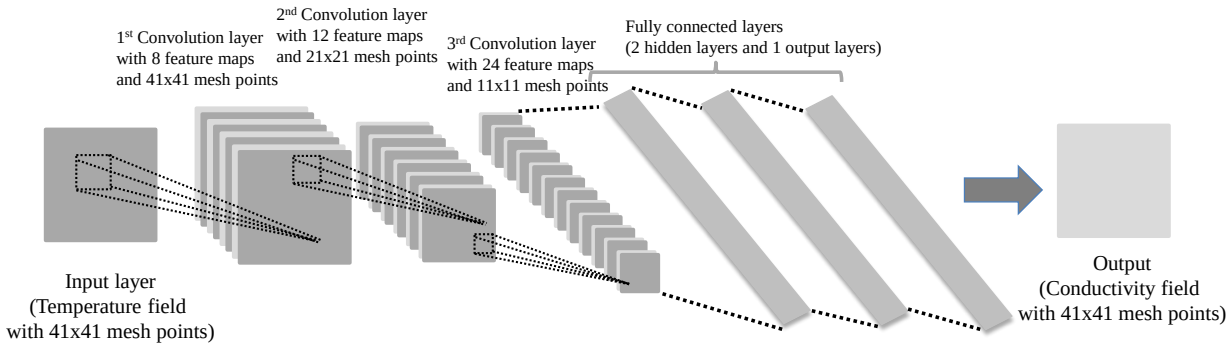


Figure 6.3. Architecture of CNN-based thermal conductivity (adopted after LeCun) [47].

Learning is an optimization process, and we need to define a cost function based on distinct types of data to inform ML algorithms to tune NN hyperparameters. Eq. (6.11) defines the cost function where N, $y_{i,data}$, and $y_{i,model}$ are the total number of training data, ith training data, and ith model solution. To prevent overfitting, we add a regularization term in Eq. (6.11) where i, and N_L denote the ith layer and total layer number. λ is the regularization strength, and $\mathbf{W}$ is the matrix of total weights in ith layer. We implement NN-based thermal conductivity using Tensorflow [57] which is the DL framework developed by Google. Weights and biases of NNs are tuned based on data using the Adam [99] algorithm.

$$E = \frac{1}{2N}\left[\sum_{i=1}^{N}\left(y_{i,\text{model}} - y_{i,\text{data}}\right)^2 + \sum_{i=1}^{N_L}\lambda_i \left\|\mathbf{W}_i\right\|^2\right] \tag{6.11}$$

6.6. Results analysis

6.6.1. Comparing results by Type I and Type II ML using SET data

We define the notation NN-A as NNs trained by a dataset A where NNs can be either CNNs or FNNs. Figure 6.4 depicts the averaged RMSE by comparing validating datasets to Type I and Type II ML results. When we used the low-quality dataset (S1) to train FNNs, both Type I and Type II ML provided poor predictions. However, Type II ML does not query values from FNN-

based thermal conductivity. The error does not accumulate during each iteration while solving PDEs. When predictions are away from the training domain, Type II ML exhibits better performance than Type I ML. When we trained FNNs by high-quality dataset (S2), both frameworks reduce errors in predictions, but Type I ML displays better predictability than Type II ML.

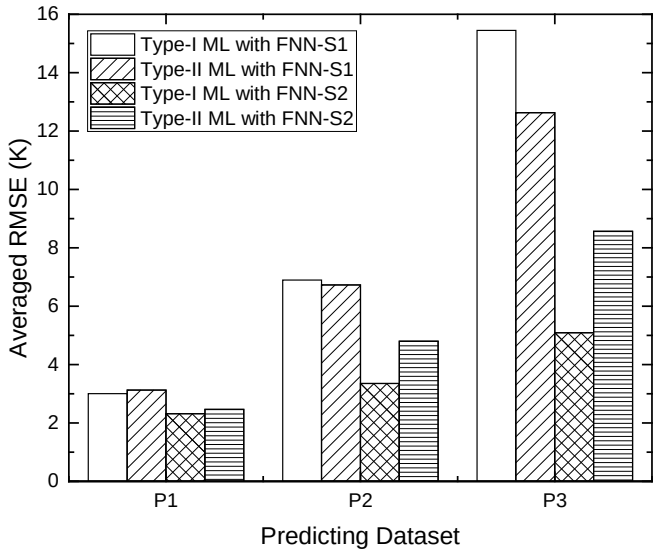
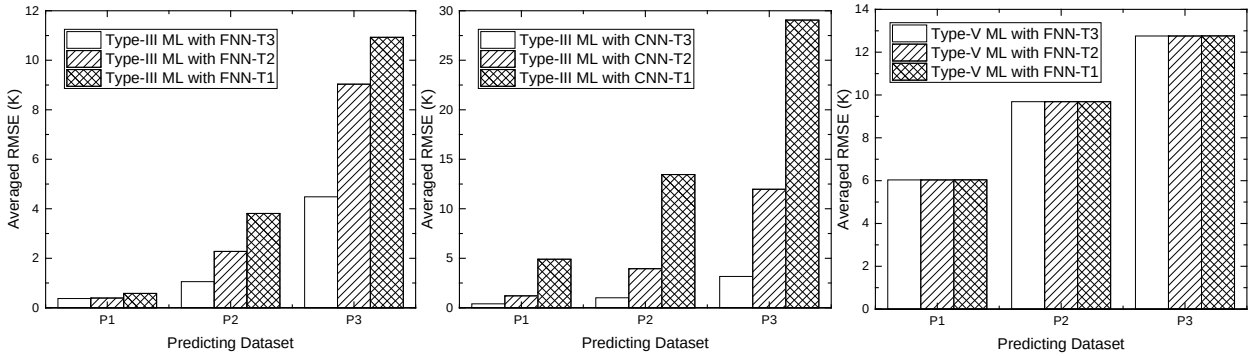


Figure 6.4. Averaged RMSE by comparing the validating datasets, P1, P2, and P3, to Type I and Type II ML results using the FNN with the training datasets, S1 and S2.

6.6.2. Comparing the results by Type III and Type V ML using IET data

Figure 6.5 illustrates the averaged RMSE by comparing validating datasets to Type III and Type V ML results. Figure 6.5(a) shows results by using Type III ML with FNN-based thermal conductivity. The results are improved as we increase the quantity of training data. Figure 6.5(b) indicates that Type III ML with the CNN-T3 yields a lower error than the FNN-T3 result. However, CNN-based closures require more training data than FNN-based closures to exhibit good predictability. CNNs are efficient in the training. We performed simulations on NVIDIA TITAN Xp, and the training of Type III ML with the FNN-T3 took approximately 43.7 hours. On the

contrary, Type III ML with the CNN-T3 only took about 1.2 hours to achieve a converged solution. Figure 6.5(c) presents the averaged RMSE by Type V ML with FNNs. The RMSE cannot be improved with increasing the quantity of training data. The results imply that targets do not uniquely depend on inputs. Bishop [106] recognized this issue and solved it by mixture density networks.



(a) (b) (c)

Figure 6.5. Averaged RMSE by comparing the validating datasets, P1, P2, and P3, to the results obtained by (a) Type III ML using the FNN, (b) Type III ML using the CNN, and (c) Type V ML using the FNN with training datasets, T1, T2, and T3.

6.7. Lessons learned

When SET data are employed to Type I and Type II ML, the data quality strongly affects the accuracy of predictions. Type I ML requires one to solve PDEs and query values from FNN-based closures for each iteration. If the data quality is low, errors accumulate in Type I ML, and Type II ML is more appropriate because Type II solves PDEs with fixed-field closures. When the data quality is high, Type I ML exhibits better performance than Type II ML. Type III ML trains a closure model that is embedded in PDEs by using IET data. The results are more accurate than Type I and Type II using SET data. Type III training requires more data than the other two frameworks. When CNNs are used in Type III ML, the increase of training data significantly

reduces the error in predictions. In the meanwhile, CNNs can accelerate the training in Type III ML by about 36 times faster than the training using FNNs. Type V ML results show an identifiability issue that indicates the selected input is not appropriate or different NNs should be used such as mixture density networks.

6.8. Summary

The case study indicates a preference for Type III ML. It can effectively utilize the field data, potentially generating more robust predictions than Type I, Type II, and Type V ML. Table 6.4 summarizes the properties of each ML framework based on the lessons learned in this study. The data quality needs to be high when NN-based closures are iteratively queried by Eq. (6.1). When the framework uses training data by IETs, the data quantity should be high to achieve predictions. The physics is conserved when the solution is constrained by Eq. (6.1). In this case study, Type V ML may require more input features than other ML frameworks. The selected input feature for Type V ML is insufficient to make the output uniquely depend on it.

Table 6.4. Properties of each ML framework for the heat conduction demonstration.

	Type I ML	Type II ML	Type III ML	Type V ML
Training data type	SET	SET	IET	IET
Data quantity requirement	Low	Low	High	High
Data quality requirement	High	Low	High	Low
Are NN-based closures iteratively queried while solving Eq. (6.1)?	Yes	No	Yes	No
Are solutions constrained by Eq. (6.1)?	Yes	Yes	Yes	No
Note	Type I is preferable when SET data quality is high.	Type II is preferable when SET data quality is low.	CNN-based closures are preferable.	There is an identifiability issue for the training.

CHAPTER 7. CASE STUDY D: TURBULENT FLOW MODELING

7.1. Problem formulation

Reynolds-averaged Navier-Stokes (RANS) equations obtained by temporal averaging of Navier-Stokes equations require Reynolds stress to close the model. The linear eddy viscosity model (LEVM) has been widely used to represent Reynolds stress that leads to various mechanistic turbulence models [21] such as the Spalart-Allmaras, k-ε, and k-ω models. The models have been extensively studied, evaluated and calibrated for different flow characteristics with different degrees of accuracy. Consequently, performance of different models is limited in their calibration domain and exhibit high uncertainty in prediction regimes [86, 107].

With advanced computing power, "first-principles" DNS and high-resolution LES have been used to generate high-fidelity turbulence data to inform turbulence modeling. Although not so named, Type I and Type II ML previously have been formulated and applied for data-driven turbulence modeling; e.g., in the work of Zhang & Duraisamy [18] and Ling, Kurzawski & Templeton [19]. Their implementation is analyzed with respect to the proposed frameworks in Sections 3.2.2 and 3.2.3, respectively.

7.2. Objectives

The objectives are to show how to employ Type I and Type II ML for data-driven turbulence modeling. Although Type I and Type II ML have been previously demonstrated in the literature, the reference work does not present a clear workflow of the frameworks. Through this case study, we can also show the limitation of existing published work on data-driven turbulence modeling.

7.3. Implementation

7.3.1. Implementation of turbulent flow modeling by Type I ML

Zhang & Duraisamy [18] used the spatiotemporal function to modify the *k*-ω model that can inform RANS simulation by assimilating data from DNS. Figure 7.1 depicts the application of Type I ML framework for data-driven turbulence modeling as proposed by Zhang & Duraisamy [18]. In correspondence with the structure described in Figure 3.7, the procedure includes the following elements:

Element 1. Assume scale separation is achievable such that DNS data ($\boldsymbol{\Psi_{DNS}}$) can be used to obtain a spatiotemporal function ($\boldsymbol{\alpha}$) in the *k*-ω model. Then collect RANS data ($\boldsymbol{\Psi_{RANS}}$) for computing candidates of flow features ($\boldsymbol{Q}$).

Element 2. Average DNS data ($\boldsymbol{\Psi_{DNS}}$) to match the dimension of RANS data ($\boldsymbol{\Psi_{RANS}}$). Then scale the flow features ($\boldsymbol{Q}$) from element 1 as inputs for element 3

Element 3. Select flow features ($\boldsymbol{Q}$) through the hill-climbing feature selection, and use the results as training inputs for element 5.

Element 4. Compute the training targets, spatiotemporal factors ($\boldsymbol{\alpha}$), by solving the inverse problem using the turbulence kinetic energy equation and averaged $\boldsymbol{\Psi_{DNS}}$.

Element 5. Utilize an NN algorithm to capture the underlying correlation between flow features ($\boldsymbol{Q}$) and spatiotemporal factors ($\boldsymbol{\alpha}$). After the training, output the FNN-based spatiotemporal model, $FNN(\boldsymbol{Q(\Psi_{RANS})})$, to element 6.

Element 6. The $g(ML(\boldsymbol{Q}))$ is equal to $ML(\boldsymbol{Q})$ since there is no assumption made in the reference.

Element 7. Implement the FNN-based spatiotemporal model into the *k*-ω model, and solve RANS equations for predictions.

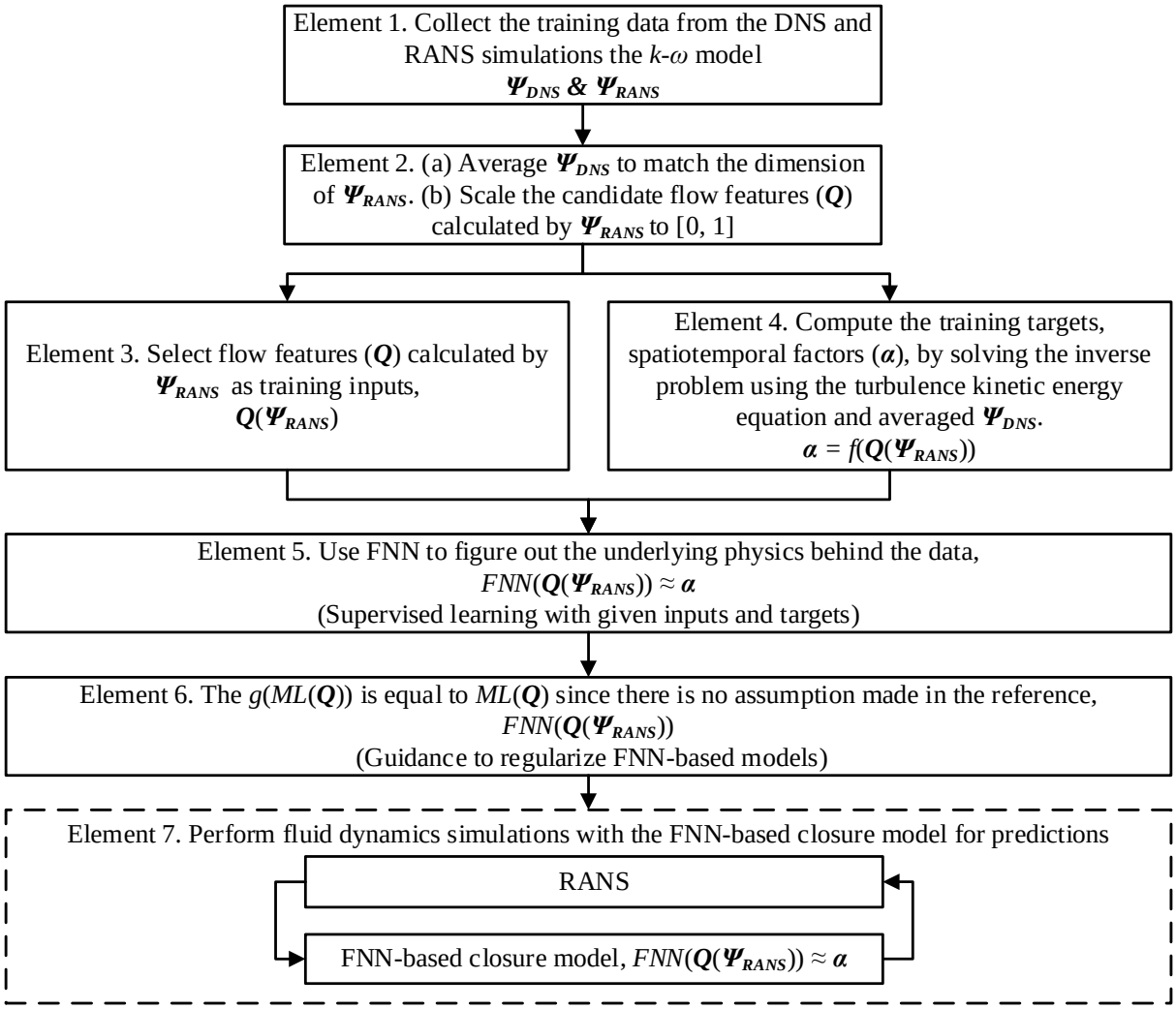


Figure 7.1. Type I ML for turbulence modeling as proposed by Zhang & Duraisamy [18].

The study simulated 1D channel flow and used training datasets with friction Reynolds numbers (Re_τ) [108] ranging from 180 to 4200. The result indicated that the reconstructed function (α) could be applied to the testing case with Re_τ equal to 2000.

7.3.2. Implementation of turbulent flow modeling by Type II ML

Ling, Kurzawski & Templeton [19] utilized the ML-based Reynolds stress anisotropy tensors by tensor basis neural networks (TBNNs) to close RANS equations. Figure 7.2 depicts the application of Type II ML framework for data-driven turbulence modeling as proposed by Ling,

Kurzawski & Templeton [19]. In correspondence with the structure described in Figure 3.8, the procedure includes the following elements:

Element 1. Perform RANS simulations with the *k*-ε model. The results ($\boldsymbol{\Psi_{RANS}}$) are prior solutions for training.

Element 2. Perform DNS simulations with identical system characteristics in element 1. The results ($\boldsymbol{\Psi_{DNS}}$) are baseline solutions for training.

Element 3. Average $\boldsymbol{\Psi_{DNS}}$ to match the dimension of $\boldsymbol{\Psi_{RANS}}$.

Element 4. Select five tensor invariants ($\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_5]$) as training inputs for the input layer and ten isotropic basis tensors ($\boldsymbol{T} = [T_1, ..., T_{10}]$) as inputs for the tensor input layer by $\boldsymbol{\Psi_{RANS}}$. $\boldsymbol{\lambda}(\boldsymbol{\Psi_{RANS}})$ and $\boldsymbol{T}(\boldsymbol{\Psi_{RANS}})$ are training inputs to element 6. It is noted that the $\boldsymbol{\lambda}$ and $\boldsymbol{T}$ can be computed from the non-dimensionalized strain rate ($\boldsymbol{S}$) and rotation rate tensors ($\boldsymbol{R}$).

Element 5. Compute Reynolds stress anisotropy tensors ($\boldsymbol{b}$) by averaged $\boldsymbol{\Psi_{DNS}}$ as the training targets that can supervise NN algorithms to learn from data.

Element 6 Use NN algorithms to represent the underlying correlation of the non-dimensionalized strain rate ($\boldsymbol{S}$), rotation rate tensors ($\boldsymbol{R}$) and Reynolds stress anisotropy tensors ($\boldsymbol{b}$). After the training, output the TBNN-based Reynolds stress anisotropy tensor, $TBNN(f(\boldsymbol{\lambda}(\boldsymbol{\Psi_{RANS}})), \boldsymbol{T}(\boldsymbol{\Psi_{RANS}}))$, to element 8.

Element 7. Execute a new RANS (*k*-ε) simulation ($\boldsymbol{\Psi'_{RANS}}$) with different system characteristics. Then use the solution to obtain $\boldsymbol{\lambda}$ and $\boldsymbol{T}$ as inputs to element 8.

Element 8. Use $\boldsymbol{\lambda}$ and $\boldsymbol{T}$ from element 7 as inputs to query values from the TBNN-based Reynolds stress anisotropy tensor model, $TBNN(f(\boldsymbol{\lambda}(\boldsymbol{\Psi_{RANS}})), \boldsymbol{T}(\boldsymbol{\Psi_{RANS}}))$. Output the Reynolds stress anisotropy tensor as fixed fields to element 9.

Element 9. Implement the results from element 8 into the RANS solver, SIERRA Fuego [109], for predictions.

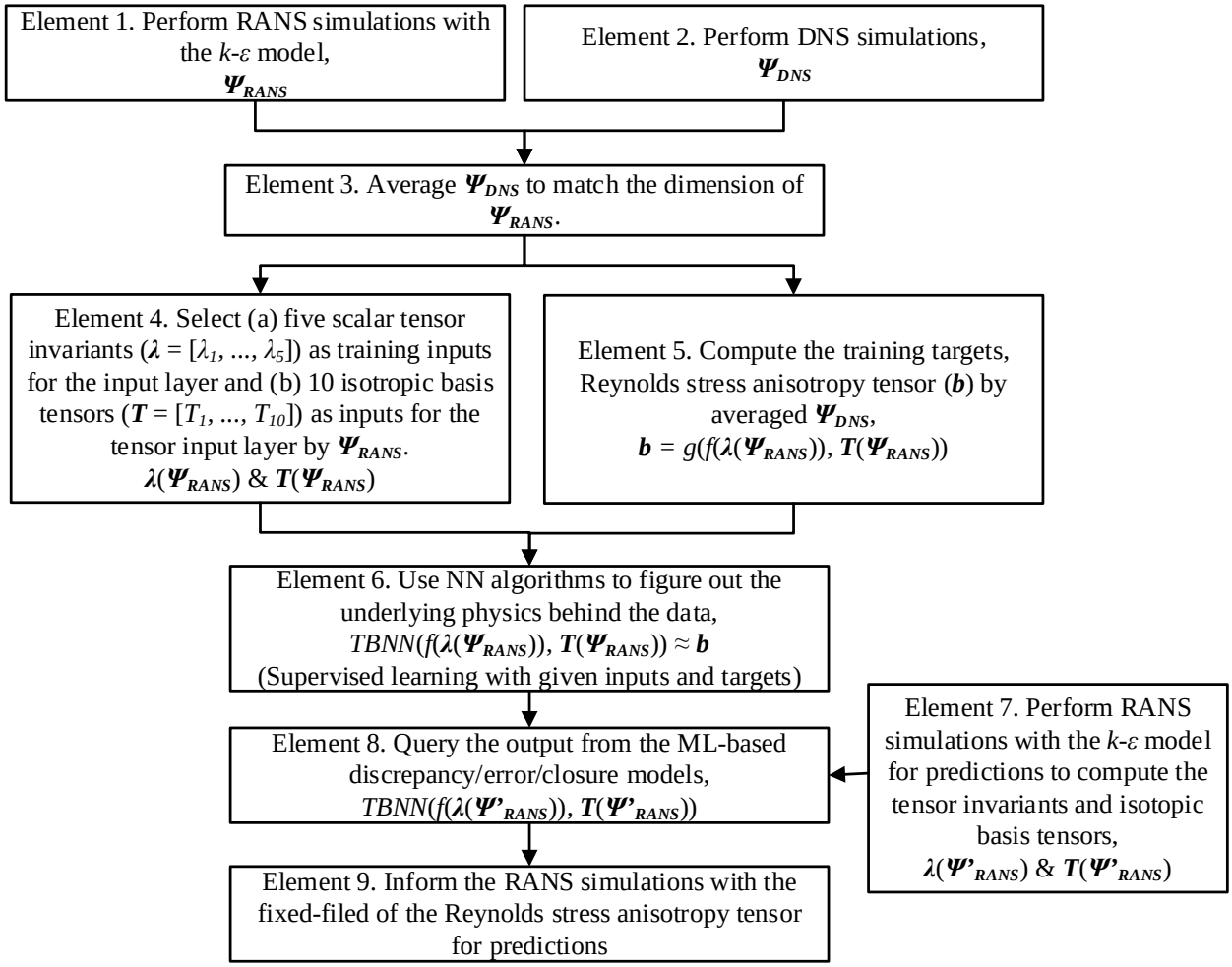


Figure 7.2. Type II ML for turbulence modeling as proposed by Ling *et al.* [19].

Two flow cases had been tested including turbulent duct flow (Re_b = 2000) and flow over a wavy wall (Re = 6850). The TBNN was trained by six cases with various Reynolds number given in Table 7.1. The results indicated that the TBNN with embedded Galilean invariance could be used for Reynolds stress anisotropy predictions which is better than generic NNs. Notably, the TBNN yields more accurate predictions than the LEVM.

Table 7.1. High-fidelity simulations for training the TBNN.

Duct flow [110]	Channel flow [111]	Inclined jet in cross-flow [112]	Perpendicular jet in cross-flow [113]	Flow around a square cylinder [114]	Flow through a converging-diverging channel flow [115]
$Re_b = 3500$	$Re_\tau = 590$	$Re_{jet} = 5000$	$Re_{jet} = 5000$	$Re = 21400$	$Re_\tau = 600$

Kutz [3] suggested that DNNs have the potential to bring a paradigm shift in modeling of complex flows thanks to their capability to capture multiscale features from data. He indicated that ROMs for fluids based on the singular value decomposition have difficulties to capture transient and multiscale phenomena as well as invariances due to scaling. On the contrary, DNNs can capture multi-scale features [116] through its hierarchy. Although DNNs can predict trends in data well, it is a challenge for DNNs to generate readily interpretable physical models.

7.4. Summary

Zhang & Duraisamy's [18] work belongs to Type I ML. They showed that the spatiotemporal function allowed the k- ω model to assimilate data. However, the method requires Boussinesq hypothesis that limits the function form to eddy viscosity models. It requires an extensive demonstration to show the applicability of the method regarding flows in a different geometry and regime.

Ling, Kurzawski & Templeton's [19] work belongs to Type II ML. They demonstrated that TBNN captured the invariant of Reynolds stress modeling for various flows. The work used fixed fields of the DL-based Reynolds stress to close RANS equations. The authors also mentioned that the stress model should be iteratively queried while solving RANS equations. The demonstrations use steady-state cases to show that the TBNN can improve RANS predictions in different geometries and at distinct Reynolds numbers. For Type II ML there exists an open question about what the magnitude of errors can be before it is too late to bring a prior solution to a baseline.

CHAPTER 8. CASE STUDY E: DATA-DRIVEN TURBULENCE MODELING

8.1. Introduction

Reynolds-averaged Navier-Stokes (RANS) equations are widely used in fluid engineering simulation and analysis due to its computational efficiency. Reynolds stress is essential to close RANS equations. Linear eddy viscosity models (LEVMs) are attractive to represent Reynolds stress due to their computational efficiency. LEVMs include Spalart-Allmaras [22], k-ε [23], and k-ω [24] models that require extensively evaluated and calibrated for different flow characteristics. Consequently, performance of different models are limited in their calibration domains and exhibit different degrees of uncertainty in prediction. Tracey, Duraisamy & Alonso [86] demonstrated that the Menter's k-ω model [117] yielded large uncertainty for the calculation of Reynolds stress anisotropy.

The growing interest in machine learning (ML), especially deep learning (DL), application for science and engineering leads to data-driven modeling of Reynolds stress. Deep learning (DL) [4] belongs to a branch of machine learning (ML), and it is a universal approximator [44] that can capture underlying correlations behind data. DL (or deep neutral networks, DNNs) with its hierarchical model structure can leverage values of large datasets from relevant experiments and simulations without limiting to a single data source. Such feature can achieve total data-model integration [14] that is capable of constructing fluid closures over a range of flow regimes. Based on data and knowledge requirements, Chang & Dinh [118] classified five types of ML frameworks for using ML in thermal fluid simulation. The present work employs Type I (physics-separated) and Type II (physics-evaluated) ML frameworks [118] for the development of DL-based Reynolds stress.

Type I ML [118] requires a scale separation assumption [79, 82] such that closure relations can be derived separately from conservation equations using experimental data and ML models. Then simulation solves conservation equations with embedded ML-based closures. Closure relations by Type I ML are iteratively queried during simulation. Previous Type I ML applications included system-level flow modeling and Reynolds-averaged turbulence modeling. Chang & Dinh [88, 89] employed DL-based closures to model system pressure drop and boiling channel flow. Ma, Lu & Tryggvason [71] used neural networks (NNs) to surrogate fluid closures from simulating isothermal bubbly flow. Tracy, Duraisamy & Alonso [75] and Zhang & Duraisamy [18] used shallow NNs to achieve data-driven turbulence modeling. Although Type I ML has been employed for flow simulation, previous works do not investigate in data requirement for developing DL-based closures with predictive capabilities. In the present work, we demonstrate a method to quantify the predictive capability of DNNs based on training datasets with different qualities.

Type II ML [118] requires knowledge on selections of simulation models as low-fidelity models, which are efficient for computation. Model uncertainty can be reduced by high-fidelity simulation such as direct numerical simulation and large eddy simulation. Closure relations by Type II ML can be built to correlate the inputs (mean flow properties by low-fidelity simulation) and targets (quantities of interest by high-fidelity simulation). To employ Type II ML in thermal fluid simulation, we need to run low-fidelity simulation with embedded mechanistic closures to obtain mean flow properties as model inputs. Then we use the inputs to query outputs from ML-based closures. The outputs, fixed values, are implemented in the low-fidelity model to replace the mechanistic closure. The Type II ML approach is similar to the strategy to leverage high-fidelity data proposed by Lewis *et al.* [15]. Previous Type II ML works included Reynolds-averaged turbulence modeling. Ling, Kurzawski & Templeton [19] used DL-based Reynolds stress with

embedded Galilean invariance to close RANS, and they demonstrated predictive capabilities of the DL-based stress for flows in different geometries. However, they focused on steady flow applications. In the percent work, we demonstrate that DL-based Reynolds stress can be used for unsteady flow simulation.

Kutz [3] addressed several open challenges for applications of DL-based closures such as requirements of training data. Ling & Templeton [43] applied the Mahalanobis to indicate the similarity between training and testing data. In the present paper, Type I and Type II ML [118] are studied to investigate the requirements of DL-based Reynolds stress development. The case study in the present work is formulated based on: (i) How large training datasets should be to train DL-based closures? (ii) What are the necessary and sufficient flow features? These questions are fundamental to define data requirements of thermal fluid simulation with embedded DL-based closures. We define the RANS simulation with embedded DL-based closures as RANS-DL. The present paper investigates how to apply RANS-DL to accomplish data-driven turbulence modeling of computational fluid dynamics by assimilating available, relevant, and adequately evaluated data.

The case study is structured to include objectives (Section 8.2), assumption testing (Section 8.3), formulation of the case study (Section 8.4), flow features coverage mapping (Section 8.5), implementation of ML frameworks (Section 8.6), results (Section 8.7), lessons learned (Section 8.8), and summary (Section 8.9).

8.2. Objectives

The objectives include three aspects. First, we want to test whether DL can capture the hidden physics without knowing the complete flow history. Second, we investigate what is the

essential and necessary flow features for the DL-based Reynolds stress to reconstruct the reference solution. Finally, we explore the limits of Type I and Type II ML in the case study.

8.3. Assumption testing

Data-driven modeling by DL requires a substantial amount of data. To investigate the data requirement, we formulate three tests to investigate how to use DL to close RANS equations. The first test is to find the essential datasets to reconstruct the history of flow transients by RANS-DL. The second test is to determine necessary flow features as inputs for DL-based closures. Finally, the last test is to examine the applicability of Type I and Type II ML for unsteady flow simulation.

8.3.1. Assumption testing on the data requirement

We assume that DL can discover hidden time derivatives from spatially distributed velocity fields collected from different flow patterns. Therefore, we can sample data from various simulation time steps and train DNNs using total data. The assumption testing includes training data obtained from reference solutions by RANS simulation. The success criterion depends on whether RANS-DL can reconstruct reference solutions.

8.3.2. Assumption testing on the flow feature selection

We assume that the sufficient and necessary flow features can be defined by spatial derivatives of velocity fields. DL belongs to supervised learning [41] which requires inputs and targets for training. For DL-based Reynolds stress, the inputs are flow features that represent mean flow properties, and the target is the Reynolds stress tensor. We select input flow features based on the incompressible momentum equation [20] given by Eq. (7.1) where $\bar{u}$ is the mean velocity and i, j, and k denote directions. D/Dt, ρ, $\bar{p}$, $\bar{\tau}_{ij}$, μ, and δ_{ij} are the material derivative, fluid density, mean pressure, Reynolds stress tensor, molecular viscosity, and Kronecker delta. We can

manipulate Eq. (7.1) into Eq. (7.2) that shows the dependency of Reynolds stress. Eq. (7.3) gives the derivative of Reynolds stress as a function of several bases from Eq. (7.2).

$$\rho \frac{D\overline{u}_i}{Dt} = -\frac{\partial \overline{p}}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial \overline{u}_k}{\partial x_k}\right] + \frac{\partial \overline{\tau}_{ij}}{\partial x_j} \tag{7.1}$$

$$\frac{\partial \overline{\tau}_{ij}}{\partial x_j} = \rho \frac{D\overline{u}_i}{Dt} + \frac{\partial \overline{p}}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial \overline{u}_k}{\partial x_k}\right] \tag{7.2}$$

$$\frac{\partial \overline{\tau}_{ij}}{\partial x_j} = f\left(\rho \frac{D\overline{u}_i}{Dt}, \frac{\partial \overline{p}}{\partial x_j}, \frac{\partial}{\partial x_j}\left(\mu\frac{\partial \overline{u}_i}{\partial x_j}\right), \frac{\partial}{\partial x_j}\left(\mu\frac{\partial \overline{u}_j}{\partial x_i}\right), \frac{\partial}{\partial x_j}\left(\mu\delta_{ij}\frac{\partial \overline{u}_k}{\partial x_k}\right)\right) \tag{7.3}$$

Based on the first assumption testing, time derivatives are not selected as training inputs since data are steady for each dataset. The merged PISO-SIMPLE algorithm [119, 120] is implemented for pimpleFoam that use the projection method [121, 122] to solve RANS equations. Since pressure is separately solved from the momentum equation, we further assume that the pressure term can be excluded from training inputs. As a result, the essential flow features for DL-based Reynolds stress can be represented by remaining spatial derivatives of velocities. Eq. (7.4) uses the matrix form to show Reynolds stress ($\boldsymbol{\tau}$) as a dyadic product between the derivative operator and velocity ($\mathbf{U}$). The dyadic product in Eq. (7.4) results in nine velocity derivatives as input flow features. Targets are the Reynolds stress symmetry tensor that includes six stress components. The assumption testing is to examine whether those nine flow features are sufficient and necessary for surrogating Reynolds stress by DNNs.

$$\boldsymbol{\tau}_{turb} = f\left(\left(\nabla \otimes \mathbf{U}\right)^T\right) \tag{7.4}$$

8.3.3. Assumption testing on Type I and Type II ML

Chang & Dinh proposed Type I and Type II ML frameworks [118] to build closure relations for thermal fluid simulation. We implement these two frameworks for data-driven

turbulence modeling using DL-based Reynolds stress in Section 8.6 to explore performance of each framework. Closure relations are iteratively queried in Type I ML while solving conservation equations. Type II ML solves conservation equations with fixed closure relations. Therefore, we assume Type I ML is capable of simulating unsteady flow while Type II ML has limitations in transient problems. The goal of Type I ML is to reproduce transient solutions by RANS. Type II ML aims at exploring what the magnitude of errors can be before it is too late to bring solutions to the quasi-steady state from a transient state. The assumption testing is to evaluate whether the goals for each ML framework is achieved.

8.4. Formulation of the case study

8.4.1. Numerical experiment

The numerical experiment is formulated to evaluate performance of RANS simulation with embedded DL-based Reynold stress (RANS-DL). The RANS simulation using the k-ε model serves as reference solutions that are used to train DL-based Reynolds stress. Figure 8.1 depicts the simulation configuration created by Pitz and Daily [123] which is used to explore the requirements for data-driven turbulence modeling. The 2D geometry includes a backward-facing step and converging nozzle. System characteristics are summarized in Table 8.1. This geometry configuration is complex enough since unsteady flow is affected by the turbulence mixing layer growth, entrainment rate, and reattachment length. The k-ε model has been validated [124] for this geometry. The pimpleFoam solver [39] in OpenFOAM [38] is used to generate data for the development of DL-based Reynolds stress.

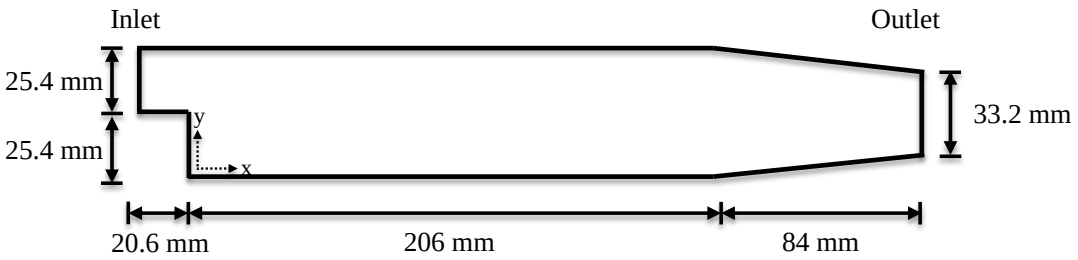


Figure 8.1. Geometry configurations of RANS simulation.

Table 8.1. System characteristics for RANS simulation.

Initial Conditions	
Velocity	0 m/s
Pressure	0 bar
Boundary Conditions	
Inlet velocity	(10, 0, 0) m/s
Inlet pressure	Zero gradient
Outlet velocity	Zero gradient
Outlet pressure	0 bar
Transport Properties	
Kinematic viscosity	10^{-5} m^2/s

8.4.2. Training data

Training data are generated by RANS simulation with the k-ε model. The equations are solved by pimpleFoam using fixed time step, 2.4×10^{-5} sec. Four datasets are created and listed in Table 8.2. The first three datasets involve millions of data points, and the last dataset has hundreds of thousands of data points. The data in T10A and T10B are uniformly sampled from ten various times, and sampling time ranges are given in Table 8.2. T10A includes less transient details than T10B because the data are sampled from a coarse time interval in T10A. The baseline dataset includes data sampled from twenty separate times, and it is used to evaluate performances of RANS-DL.

Table 8.2. Generated datasets sampled at various times with distinct flow patterns.

Dataset	Data Quantity		Total datasets sampled from various times (sec)	Description
	Inputs ($x10^6$)	Targets ($x10^5$)		
T1	0.11	0.76	0.1	Training dataset
T10A	1.15	7.65	[0.01, 0.1]	Training dataset
T10B	1.15	7.65	[0.010024, 0.01024]	Training dataset
Baseline	2.30	15.30	[0.010024, 0.01048]	Validating dataset
QSS	0.11	0.76	1	Validating dataset

The QSS (quasi-steady-state) dataset is sampled from RANS simulation. The QSS solution is checked by the mean square error (MSE) defined by Eq. (7.5) where N, i, y, and y_{ref} are the total data points, ith data point, solution at the current time step (t^n) and previous time step (t^{n-1}). Figure 8.2 depicts MSE analysis for the simulation running from 0.1 to 1 sec. Based on the result, the QSS dataset is sampled at t = 1 sec.

$$MSE = \frac{\sum_{i=1}^{N}\left(y_i - y_{ref,i}\right)^2}{N} \tag{7.5}$$

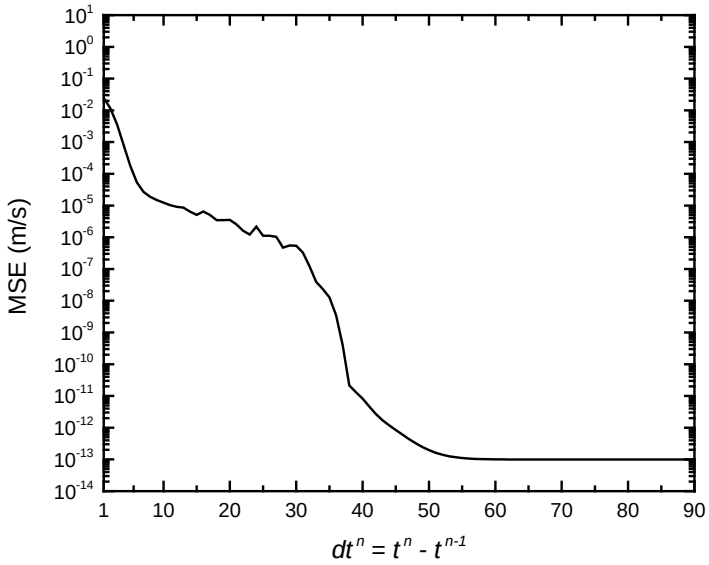


Figure 8.2. MSE analysis to check whether the quasi-steady-state condition is achieved.

8.5. Flow features coverage mapping

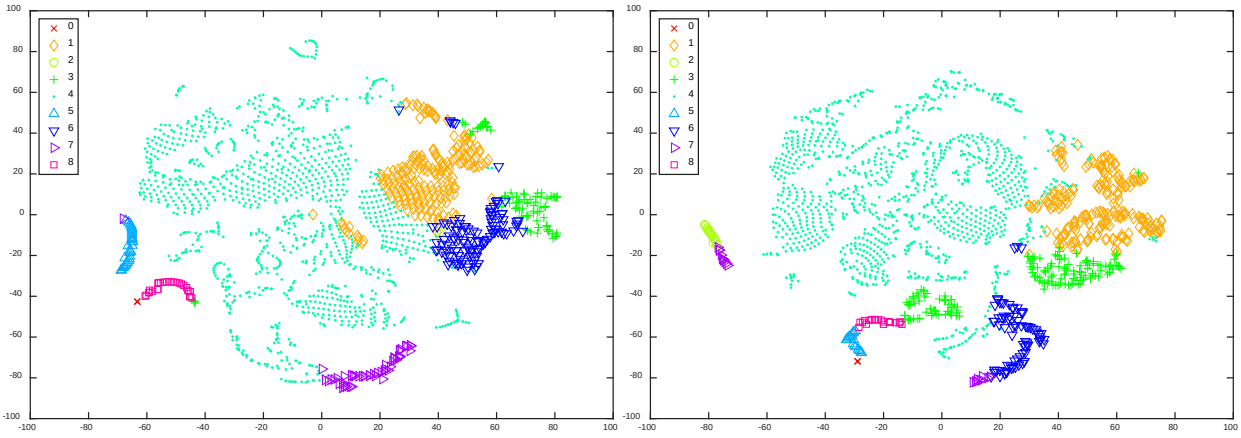
Flow features coverage mapping (FFCM) is a 2D graph that shows distributions of high-dimensional flow features. It can be used to quantify whether physics is sufficiently covered by training data. If the mapping between training and applications shows similar distributions, we are confident in predictive results by RANS-DL. The discrepancy between two FFCM can be quantified by Eq. (7.5) using MSE for point-by-point comparisons.

FFCM is obtained by a two-step approach. First, k-mean clustering [125, 126] is employed to classify flow features based on their similarities by computing distances between centroids of clusters and data points. Data points are assigned to a cluster if the minimum distance is achieved. Then centroids of clusters are updated based on data points within a cluster. The process is iteratively repeated until convergence is reached. The clustering results are multidimensional because our selected flow features include nine components of spatial derivatives of velocity fields.

Second, we use t-SNE (t-distributed stochastic neighbor embedding) [127] to visualize the clustering result that is flow features coverage mapping. t-SNE is a method for dimensionality reduction, and it can project high-dimensional data in a 2D or 3D graph while preserving characteristics of data points. t-SNE first calculates pairwise conditional probabilities using Gaussian kernels for high-dimensional data such that similar points have high probabilities while dissimilar points have low probabilities. Then t-SNE uses a *t* distribution to measure pairwise similarities of low-dimensional data points. Positions of low-dimensional points are calculated by minimizing Kullback-Leibler divergence [128] between *t* and Gaussian distributions in low-dimensional and high-dimensional spaces. A *t* distribution has fat tails at both ends that ensure dissimilar points in low-dimensional space to be placed away from similar points. Therefore, t-

SNE can embed high-dimensional data in a low-dimensional space. By k-mean clustering and t-SNE, we can build FFCM to quantify similarities of flow features between RANS-DL and training datasets. FFCM can be used to evaluate whether the training of DL-based Reynolds stress is sufficient.

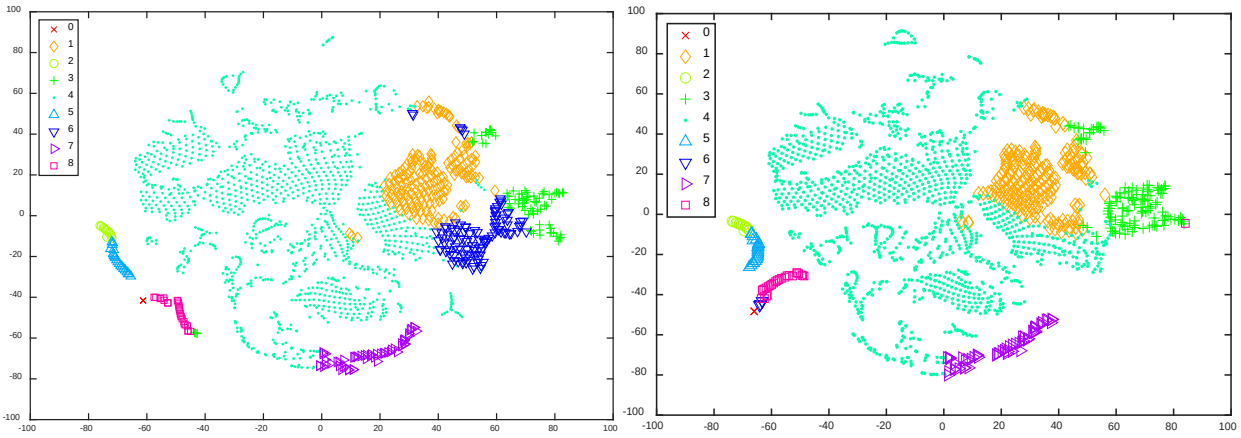
Figure 8.3 depicts FFCM by T10A data at two times. The discrepancy can be quantified by computing the Euclidean distance. Eq. (7.6) gives the mean Euclidean distance (d) where N and i denote the total data and ith data point. (x, y) and $(\hat{x}, \hat{y})$ are coordinates from two distinct mapping. We can observe significant differences between Figure 8.3(a) and Figure 8.3(b), and their distance is 43.67. Figure 8.4 shows flow features coverage mapping by T10B data at two times, and the distance between Figure 8.4(a) and Figure 8.4(b) is 11.31. The results indicate that a sharp transient happens between two consecutive time intervals in T10A. It is because the sampling interval is coarser in T10A than the interval in T10B. Figure 8.3 and Figure 8.4 serve as the references that are used to examine physics coverages of DL-based Reynolds stress in applications. If features mapping in applications has a similar distribution as the references, we expect a good prediction of velocity fields by RANS-DL.



(a) (b)

Figure 8.3. Visualization of flow features coverage mapping (FFCM) using t-SNE at (a) t = 0.01 sec and (b) t = 0.02 sec from T10A dataset. The flow features are clustered by k-means clustering with variously labeled colors.

$$d = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2\right]} \tag{7.6}$$



(a) (b)

Figure 8.4. Visualization of flow features coverage mapping (FFCM) using t-SNE at (a) t = 0.010096 sec from T10B dataset and (b) t = 0.010456 sec from the V10 dataset. The flow features are clustered by k-means clustering with variously labeled colors.

8.6. Implementation of ML frameworks

8.6.1. Implementation of NN-based Reynolds stress model

We use DL (or DNNs) to surrogate Reynolds stress due to the nonparametric modeling feature of DNNs. This feature allows the model form of DNNs to be adaptive based on various data quantities. Figure 8.5 depicts a structure of DNNs including nine input flow features and six output Reynolds stress components. We use Tensorflow [57] to design a ten-layer DNN with 512 hidden units (HUs) for DL-based Reynolds stress. The activation function (σ), rectified linear units (ReLU) [51], is used in each hidden layer (HL). Eq. (7.7) defines a cost function by Euclidean loss where N, $y_{i,data}$, and $y_{i,model}$ are the total number of training data, i^{th} training data, and i^{th} model solution. DNNs' parameters such as weights and biases are tuned based on data using Adam [99] algorithm.

$$E = \frac{1}{2N}\left[\sum_{i=1}^{N}\left(y_{i,\text{model}} - y_{i,\text{data}}\right)^2\right] \tag{7.7}$$

Large data can increase the difficulty of training DNNs. In fact, neural networks with many HLs may suffer from gradient vanishing or explosion issues that slow the learning process. Batch normalization (BN) [129] is a method that can reduce internal covariate shifts in DNNs to prevent those issues. Therefore, BN is implemented in each HL to accelerate the speed of training. Figure 8.6(a) depicts the comparison of Euclidean loss by training DNNs with different data. T10A010 denotes that training data are from T10A at t = 0.1 sec while T10A includes data from all times. T10A010 only involves one-tenth data points of T10A. Figure 8.6(a) shows that the learning using T10A is much slower than T10A010. Figure 8.6(b) reveals that the learning becomes fast while implementing BN in DNNs. Figure 8.7 depicts a model-data plot to show that DL-based Reynolds stress is well-trained by T10B dataset since model outputs agree with data.

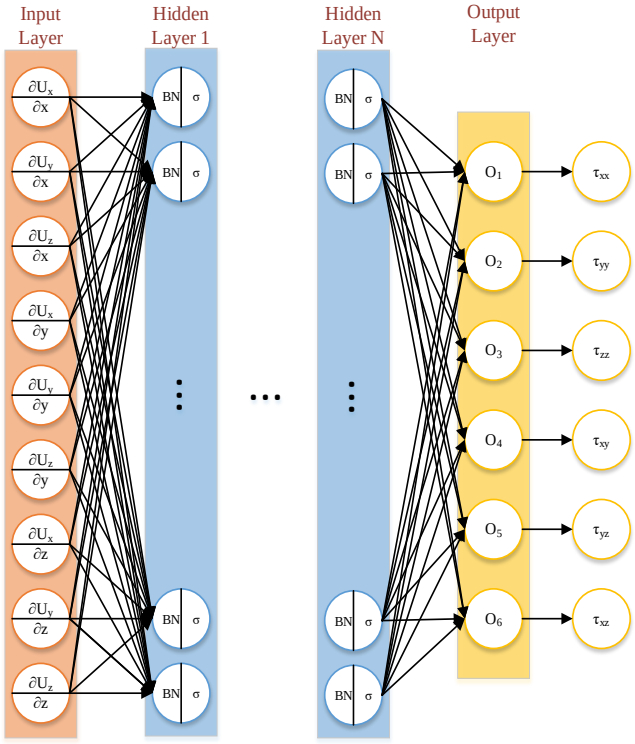


Figure 8.5. Structure of a DNN as a surrogate of Reynolds stress.

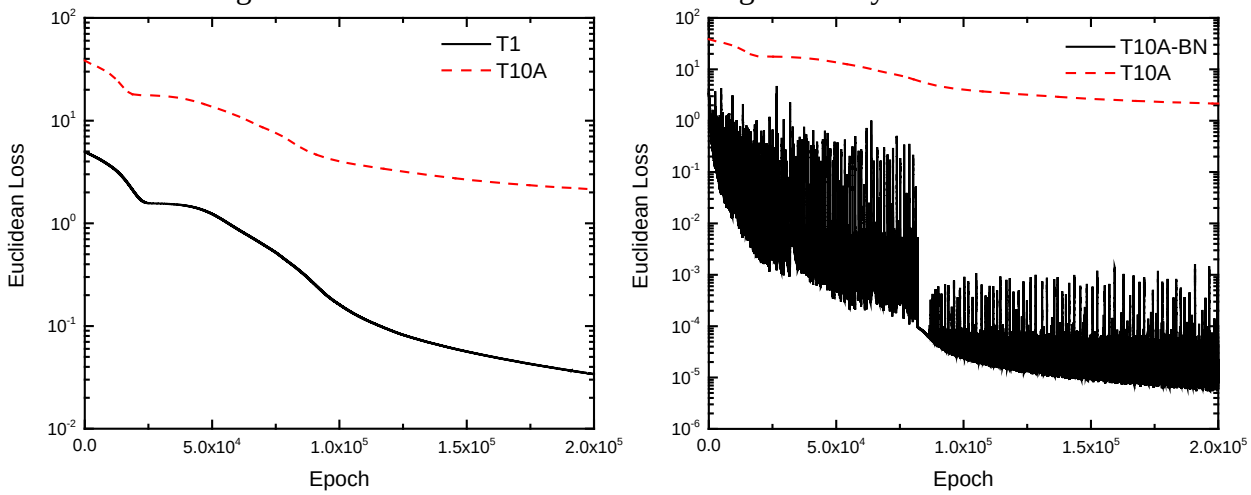


Figure 8.6. (a) Comparison of Euclidean loss between DNNs using training datasets T1 and T10A. (b) Comparison of the loss between DNNs using training datasets T10A and T10A-BN.

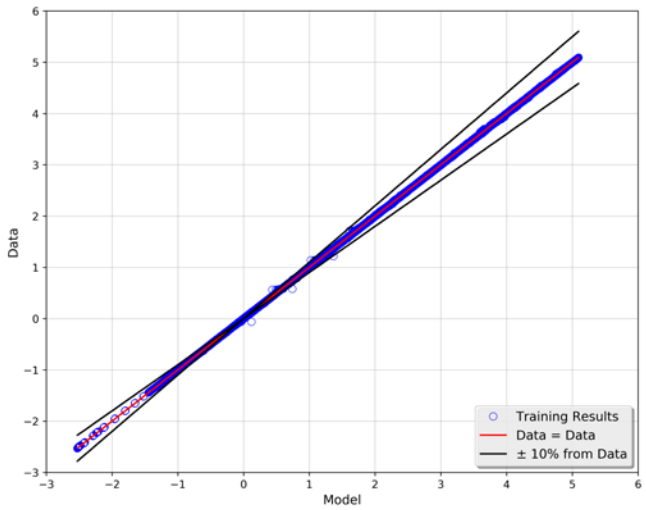


Figure 8.7. Model-data plot for the DNN trained by T10B data.

8.6.2. Implementation of Type I ML for data-driven turbulence modeling

The goal of Type I ML [118] is to build DL-based Reynolds stress that allows RANS-DL to reconstruct the results by baseline solutions. Type I ML requires a scale separation assumption [14, 79] that allows closure relations to be separately trained from conservation equations. Figure

8.8 depicts Type I ML framework for the development of DL-based Reynolds stress. The procedure includes the following elements:

Element 1. Assume the separation of scales is achievable such that Reynolds stress can be calculated from RANS data ($\boldsymbol{\Psi_{RANS}}$) using Boussinesq hypothesis with the *k*-ω model. Transient data ($\boldsymbol{\Psi_{RANS}}$) are given in Table 8.2.

Element 2. Compute a dyadic product between the gradient operator ($\boldsymbol{\nabla}$) and velocity fields ($\mathbf{U}$) from $\boldsymbol{\Psi_{RANS}}$ that results in nine velocity derivatives as flow features ($\boldsymbol{Q}$).

Element 3. Select flow features ($\boldsymbol{Q}$) calculated by element 2 as training inputs for element 5.

Element 4. Compute training targets, Reynolds stress ($\boldsymbol{\tau}$), by solving the linear eddy viscosity model using the velocity fields, turbulence kinetic energy, and dissipation rate from $\boldsymbol{\Psi_{RANS}}$. The results become targets for element 5.

Element 5. Utilize Adam algorithm [99] to capture underlying correlations between flow features ($\boldsymbol{Q}$) and Reynolds stress ($\boldsymbol{\tau}$) by DNNs. After the training, output the DL-based Reynolds stress, $DNN(\boldsymbol{Q(\Psi_{RANS})})$, to the next element.

Element 6. Constrain the output of $DNN(\boldsymbol{Q(\Psi_{RANS})})$ by $g(DNN(\boldsymbol{Q(\Psi_{RANS})}))$ to satisfy the property of 2D simulations, i.e., Reynolds stress components should be zero in x-z and y-z directions.

Element 7. Implement DL-based Reynolds stress in pimpleFoam solver. Then solve the RANS with the embedded DL-based closure that is iteratively queried. The baseline dataset in Table 8.2 is used to evaluate the performance of RANS-DL.

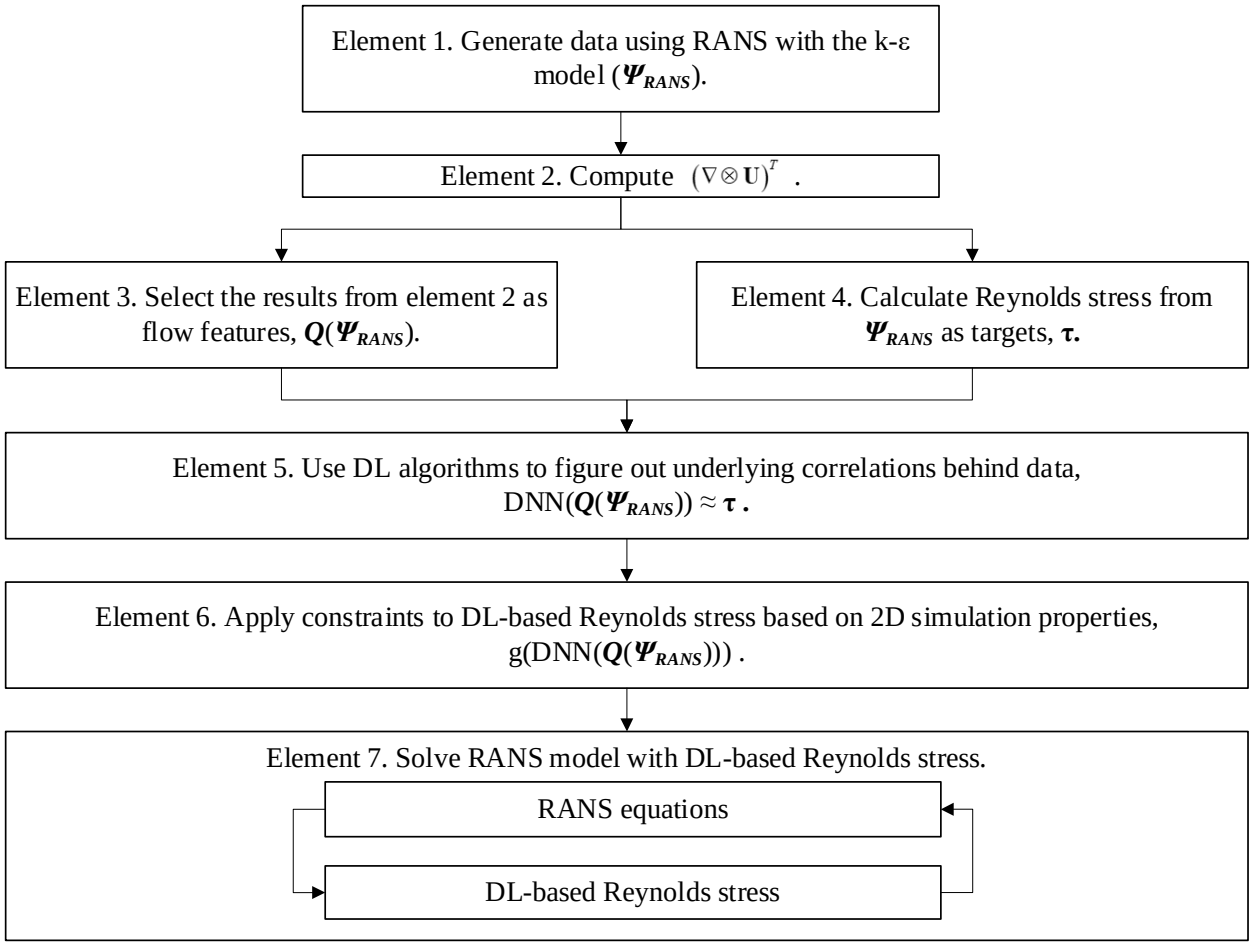


Figure 8.8. Type I ML for Reynolds-averaged turbulence modeling.

8.6.3. Implementation of Type II ML for data-driven turbulence modeling

The goal of Type II ML [118] is to use the reference Reynolds stress to bring solutions to the quasi-steady state (QSS) from various transient states. The reference Reynolds stress is calculated by QSS dataset in Table 8.2. Figure 8.9 depicts the workflow of Type II ML for data-driven turbulence modeling. The procedure involves the following elements:

Element 1. Solve RANS equations with the *k-ε* model until the solution ($\Psi_{RANS,\omega}$) reaches the quasi-steady state. QSS dataset is given in Table 8.2, and it serves as the reference that can be used to compute raining targets and to evaluate whether RANS-DL achieves the

goal. The goal is to test if Type II ML can bring solutions from various transient states to the quasi-steady state.

Element 2. Perform RANS simulations with the *k-ε* model to obtain solutions ($\boldsymbol{\Psi_{RANS}}$) at various transient states.

Element 3. Compute a dyadic product between the gradient operator ($\boldsymbol{\nabla}$) and velocity fields ($\mathbf{U}$) from $\boldsymbol{\Psi_{RANS}}$. The results include nine velocity derivatives.

Element 4. Select the nine spatial velocity derivatives from element 3 as flow features ($\boldsymbol{Q}$) which become training inputs for element 6.

Element 5. Compute reference Reynolds stress ($\boldsymbol{\tau}$) by Boussinesq hypothesis with the *k-ε* model and $\boldsymbol{\Psi_{RANS,\ \omega}}$. The results become targets for element 6 that can supervise DL algorithms to learn from data.

Element 6. Utilize DL to correlate flow features ($\boldsymbol{Q}$) from various transient states to the reference Reynolds stress ($\boldsymbol{\tau}$). After the training, output DL-based Reynolds stress, $DNN(\boldsymbol{Q(\Psi_{RANS})})$, to element 8.

Element 7. Execute RANS simulations with the *k-ε* model ($\boldsymbol{\Psi'_{RANS}}$), and stop simulations at a particular transient state. Then use the solution to compute new flow features ($\boldsymbol{Q'}$) as inputs to element 8.

Element 8. Query the values of DL-based Reynolds stress by new flow features ($\boldsymbol{Q'}$). Then output fixed Reynolds stress fields to element 9.

Element 9. Implement fixed Reynold stress fields in pimpleFoam solver to close RANS equations.

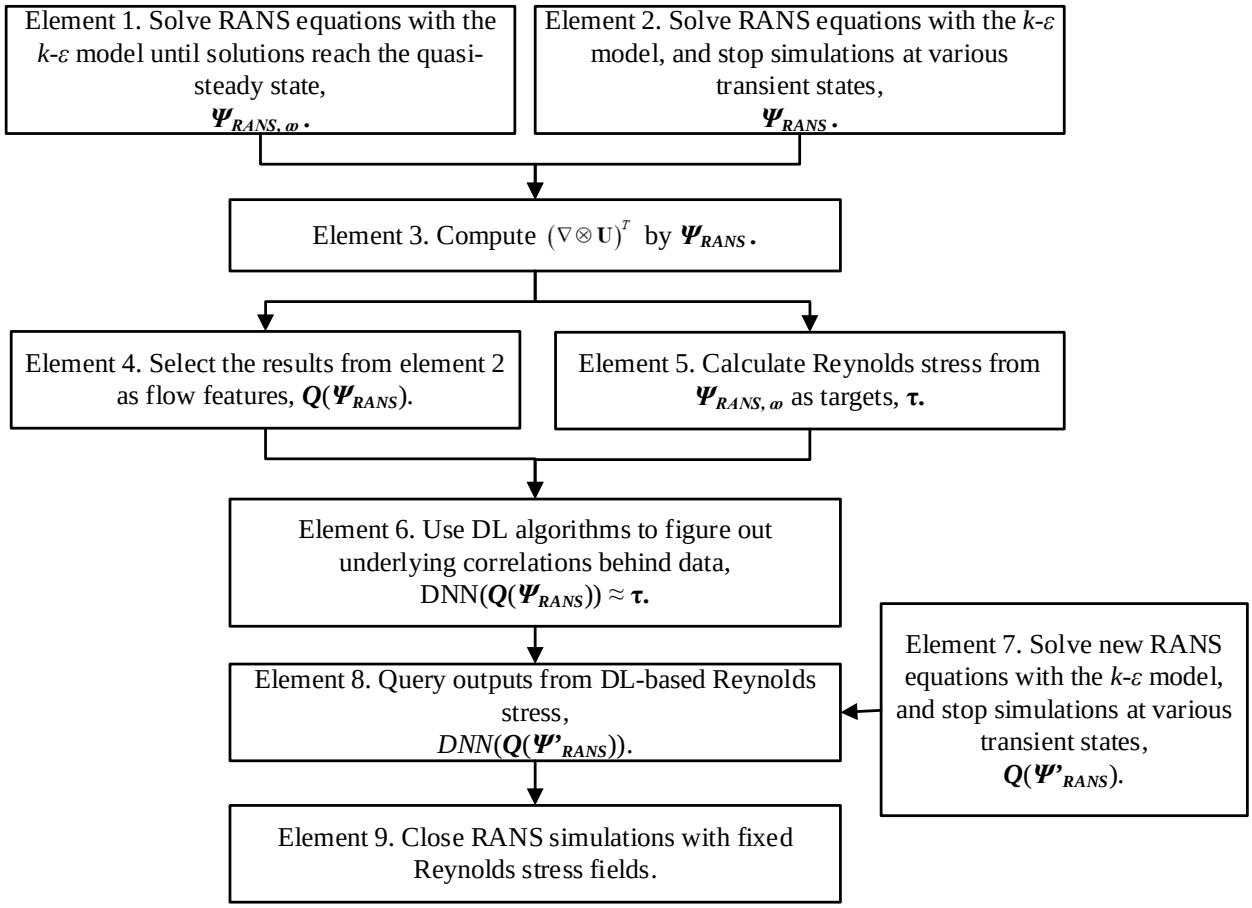


Figure 8.9. Type II ML for data-driven turbulence modeling using RANS model with DL-based Reynolds stress.

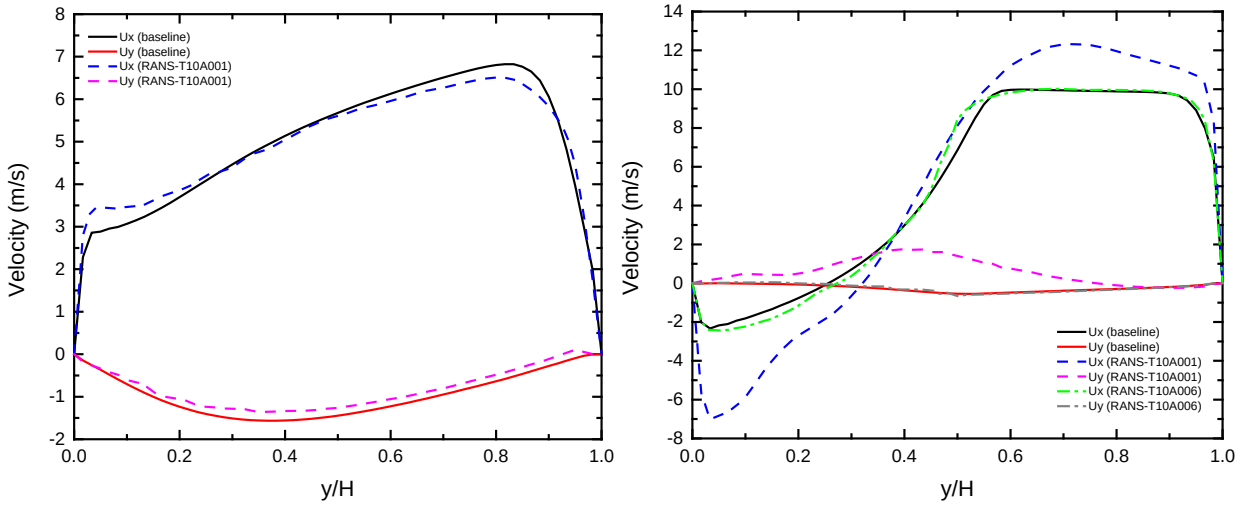
Table 8.2 shows that Type II ML only includes one-tenth data points of the data used in Type I ML. Since Figure 8.7 demonstrates that DL can successfully infer a surrogate that fits large data by T10B, the challenge of Type II ML is not subject to performance of DL. Instead, the challenge is whether Type-II ML can bring solutions to the quasi-steady state from an arbitrary transient state. To investigate this limitation, we directly explore the problem from element 7. We assume that DNNs can output ideal fields of reference Reynolds stress without uncertainty. No matter what flow features are inputted, DL-based Reynolds stress can always deliver the reference stress field. Therefore, we can implement the reference stress field in RANS equations and evaluate the performance of Type II ML while simulating unsteady flow.

8.7. Results

To explore the assumption testing, we analyze results into three sections. The first section shows that errors of RANS-DL by Type I ML are accumulated along with the simulation time. The second section focuses on testing whether RANS-DL by Type I ML can recover the baseline solutions for unsteady flow. The last section aims at testing if RANS-DL by Type II ML can find solutions to the quasi-steady state from a transient state.

8.7.1. Error accumulation along with time during simulation

The case is formulated to analyze how errors propagate when training data do not sufficiently cover the flow features in applications. We use T10A to train DL-based Reynolds stress, and implement the stress in RANS equations. Then the simulation is started at t = 0.1 sec using initial conditions obtained from the baseline. Figure 8.10(a) depicts velocity profiles at t = 0.015 sec by RANS-T10A001 which stands for RANS-DL starting at t = 0.01 sec. The trend of RANS-T10A001 velocity (dash line) agrees with the baseline (solid line). At t = 0.065 sec, Figure 8.10(b) shows that the uncertainty of RANS-T10A006 (dash-dot line) is much smaller than the uncertainty of RANS-T10A001 while comparing results to the baseline. RANS-T10A006 represents RANS-DL starting at t = 0.06 sec. Figure 8.10 indicates that T10A does not contain enough data to allow DL to capture all transient behaviors. Initially, there are strong transients in unsteady flow simulation, and errors caused by DL-based Reynolds stress grow along with the time.

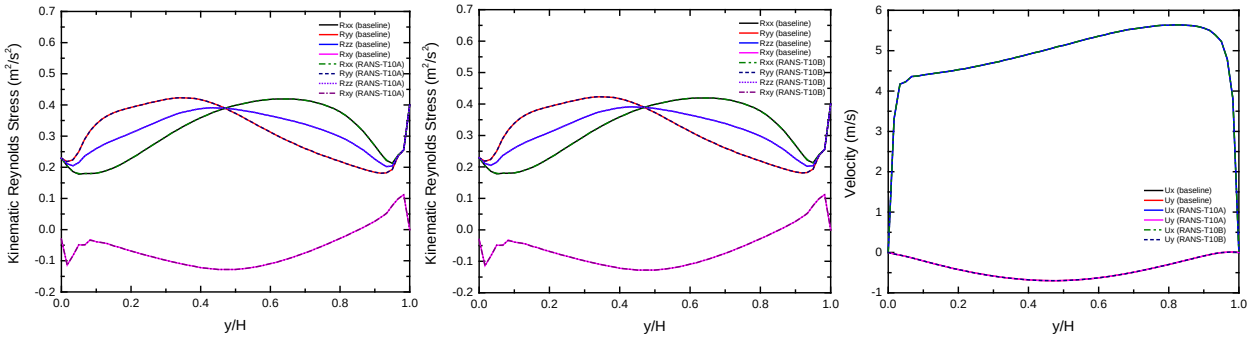


(a) (b)

Figure 8.10. (a) Comparison of velocities between the baseline and RANS-T10A at x = 0.07 m and t = 0.015 sec with initial conditions from the baseline at t = 0.01 sec. (b) Comparison of velocities of the baseline, RANS-T10A001, and RANS-T10A006 at x = 0.07 m and t = 0.065 sec with initial conditions from the baseline at t = 0.01 and 0.06 sec for RANS-T10A001 and RANS-T10A006.

8.7.2. Exploration of data requirements to reconstruct RANS solutions

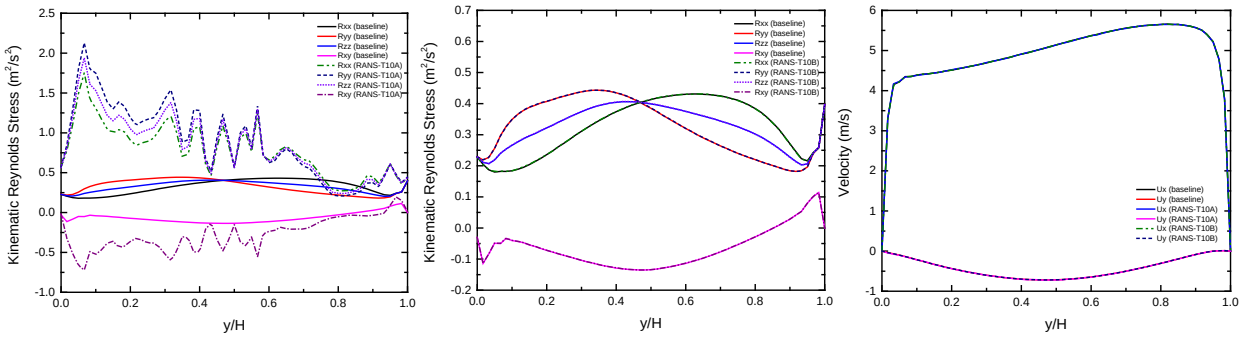
This task is formulated to compare the performance of RANS-DL with the stress closures trained by T10A and T10B. Figure 8.11 depicts initial Reynolds stress and velocities of the baseline, RANS-T10A, and RANS-T10B at t = 0.01 sec.



(a) (b) (c)

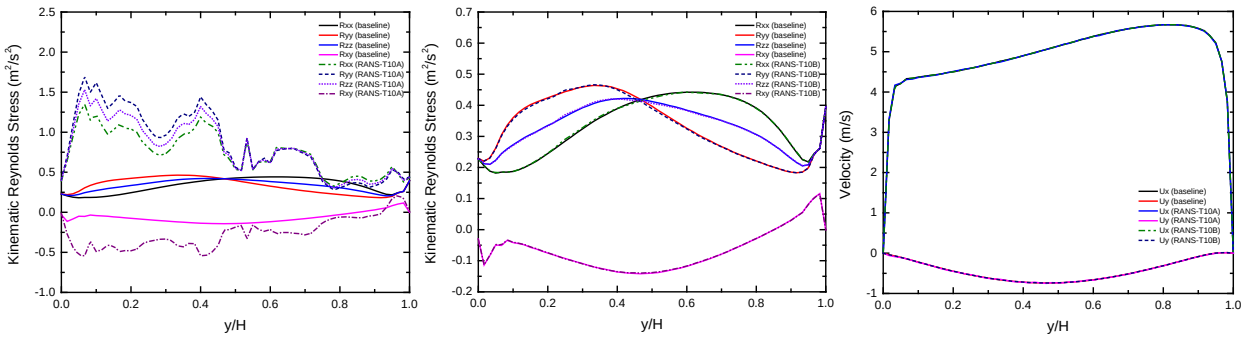
Figure 8.11. Comparison of initial kinematic Reynolds stress (a) between the baseline and RANS-T10A (b) and between the baseline and RANS-T10B at t = 0.01 sec. (c) Comparison of the initial velocities for the baseline, RANS-T10A, and RANS-T10B at t = 0.01 sec.

Figure 8.12-Figure 8.15 illustrate the results by RANS-DL at t = 0.01012, 0.01024, 0.01036, and 0.01048 sec. The first two times are within the training domain of T10B while the last two times are in extrapolation domains. For T10A, all simulation times are in extrapolation domains because its data are sampled from a coarse time interval. Therefore, RANS simulation using DL-based Reynolds stress by T10A (RANS-T10A) yields large uncertainty than RANS-T10B. Figure 8.14(b) shows that RANS-T10B starts to deviate from the baseline when the simulation is outside of the training domain. Although the simulation time is too short to make significant changes in velocity profiles, Figure 8.15(c) depicts that the velocity of RANS-T10A is different from the baseline at the bottom location.



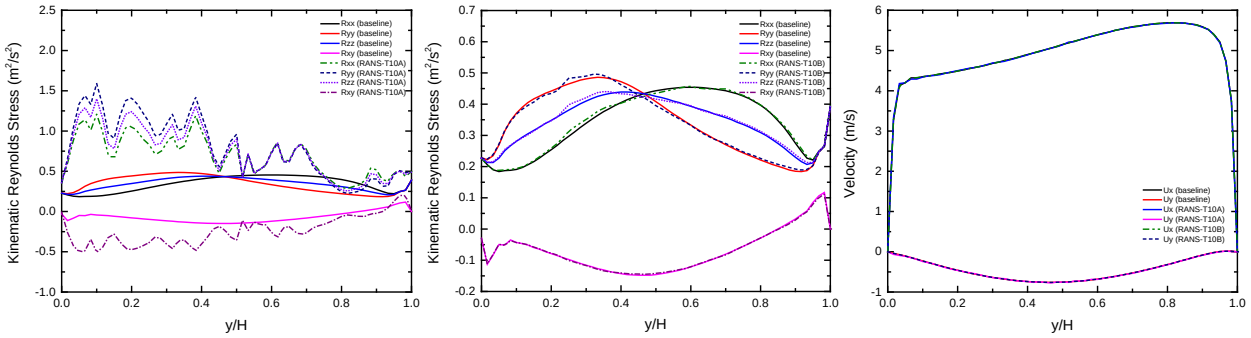
(a) (b) (c)

Figure 8.12. Comparison of kinematic Reynolds stress (a) between the baseline and RANS-T10A and (b) between the baseline and RANS-T10B at t = 0.01012 sec and x = 0.07 m. (c) Comparison of the velocity of the baseline, RANS-T10A, and RANS-T10B.



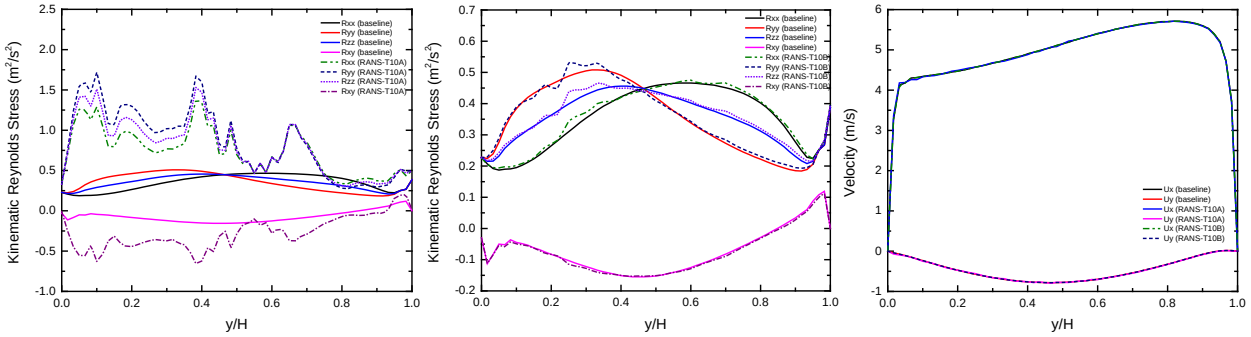
(a) (b) (c)

Figure 8.13. Comparison of kinematic Reynolds stress (a) between the baseline and RANS-T10A and (b) between the baseline and RANS-T10B at t = 0.01024 sec and x = 0.07 m. (c) Comparison of the velocity of the baseline, RANS-T10A, and RANS-T10B.



(a) (b) (c)

Figure 8.14. Comparison of kinematic Reynolds stress (a) between the baseline and RANS-T10A and (b) between the baseline and RANS-T10B at t = 0.01036 sec and x = 0.07 m. (c) Comparison of the velocity of the baseline, RANS-T10A, and RANS-T10B.



(a) (b) (c)

Figure 8.15. Comparison of kinematic Reynolds stress (a) between the baseline and RANS-T10A and (b) between the baseline and RANS-T10B at t = 0.01048 sec and x = 0.07 m. (c) Comparison of the velocity of the baseline, RANS-T10A, and RANS-T10B.

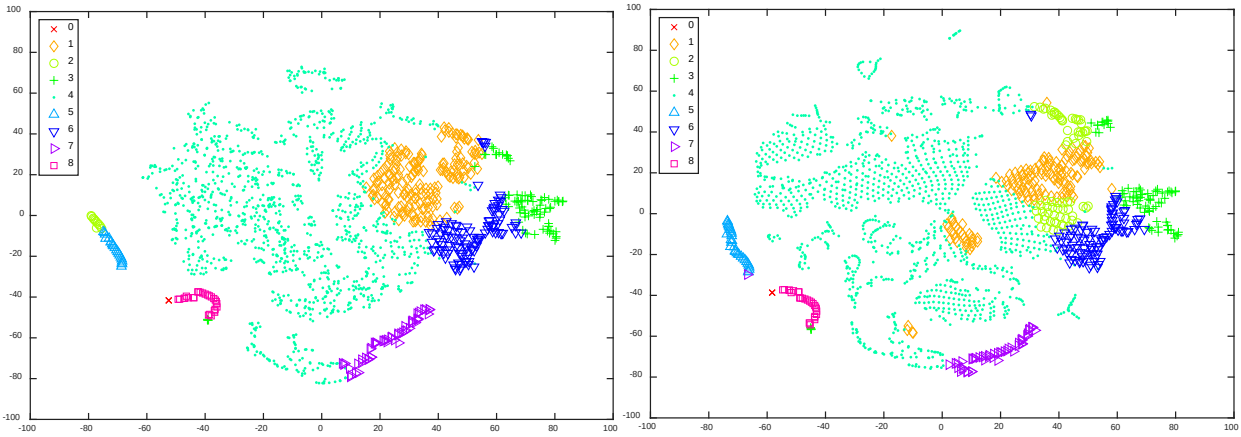
8.7.2.1. Visualization of the coverage of the flow features in applications by FFCM

We can use flow features coverage mapping (FFCM) to quantify the coverage of flow features in training datasets. Figure 8.16 depicts FFCM for RANS-T10A and RANS-T10B at t = 0.01012 sec. For RANS-T10A, we compare Figure 8.16(a) to Figure 8.3(a). Flow features in Figure 8.16(a) exhibit different distributions than the features in Figure 8.3(a). The discrepancy between two figures can be quantified by the Euclidean distance which is 35.62. The result indicates that T10A dataset is insufficient to cover the transient details in Figure 8.12(a). For RANS-T10B, we compare Figure 8.16(b) to Figure 8.4(a). The two figures have similar

distributions since the distance is 4.05 that is much smaller than the distance by RANS-T10A. The result indicates that T10B sufficiently covers the transient details so that RANS-T10B agrees with the baseline in Figure 8.12(b).

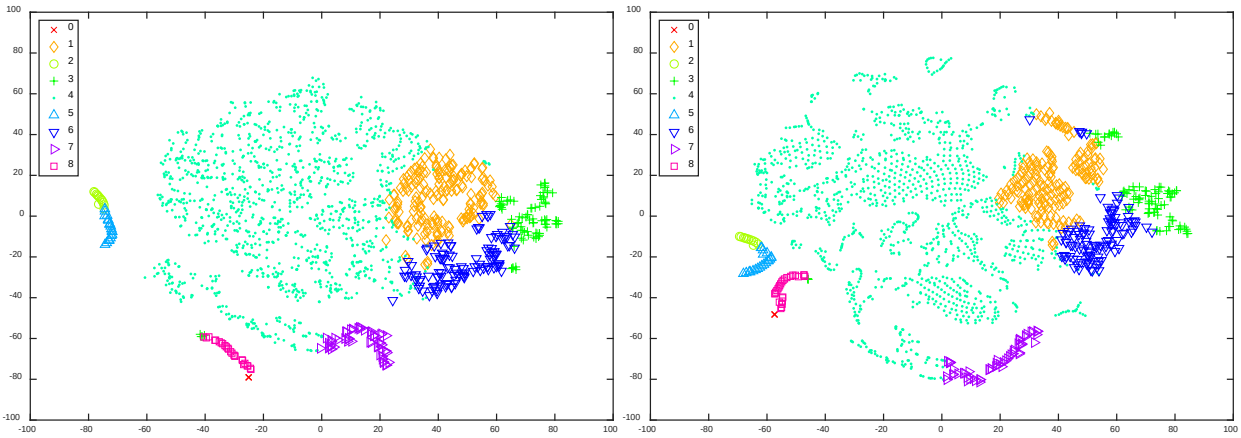
Figure 8.17 shows FFCM for RANS-T10A and RANS-T10B at t = 0.01048 sec which is outside of the training domain. Figure 8.17(a) shows FFCM for RANS-T10A, and the result is dissimilar to Figure 8.3(a) which is FFCM by training data, T10A. Figure 8.17(b) depicts that the mapping for RANS-T10B deviates from Figure 8.4(b) because the simulation is outside of the training domain. However, the distance is 10.57 which is still smaller than RANS-T10A results. Figure 8.15 shows that the performance of RANS-T10B is better than the performance of RANS-T10A.

Table 8.3 summarizes the distances between distinct FFCM. Table 8.3 indicates that the distance between RANS-T10B and T10B is much smaller than the distance between RANS-T10A and T10A. The result implies that T10B covers more transient details than T10A. Therefore, RANS-T10B shows good predictive capabilities in Figure 8.12-Figure 8.15. The analysis by FFCM indicates that RANS-DL can make inferences from training data for prediction when training data sufficiently cover the physics in applications.



(a) (b)

Figure 8.16. Visualization of flow features coverage mapping (FFCM) using t-SNE for (a) RANS-T10A and (b) RANS-T10B at t = 0.01012 sec. The flow features are clustered by k-means clustering with variously labeled colors.



(a) (b)

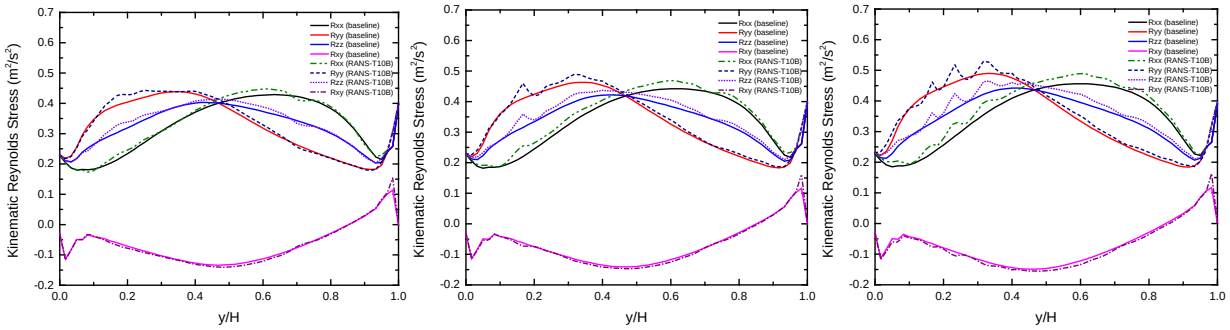
Figure 8.17. Visualization of flow features coverage mapping (FFCM) using t-SNE for (a) RANS-T10A and (b) RANS-T10B at t = 0.01048 sec. The flow features are clustered by k-means clustering with variously labeled colors.

Table 8.3. Summary of Euclidean distances between different FFCM.

Simulation time (sec)	*d* between RANS-T10A and T10A (0.01 sec)	*d* between RANS-T10B and T10B
0.010096	35.62	4.05
0.010456	38.91	10.57

8.7.2.2. Evaluation of RANS-DL using half of the solver time step

This task is formulated to solve RANS-T10B using half of the solver time step ($1.2x10^{-5}$ sec). Figure 8.18 illustrates the comparison between RANS-T10B and the baseline for kinematic Reynolds stress and velocities at three times: 0.010096, 0.01024, and 0.010384 sec. When the solver time step is reduced, DL-based Reynolds stress is not sufficiently trained by those transient conditions. RANS-T10B cannot reproduce the identical solutions as the baseline. However, when RANS-T10B predicts flow transients in the training domain, the discrepancy to the baseline is still smaller than the errors in extrapolation domains. The results indicate that DL-based Reynolds stress can make inferences from the training data.

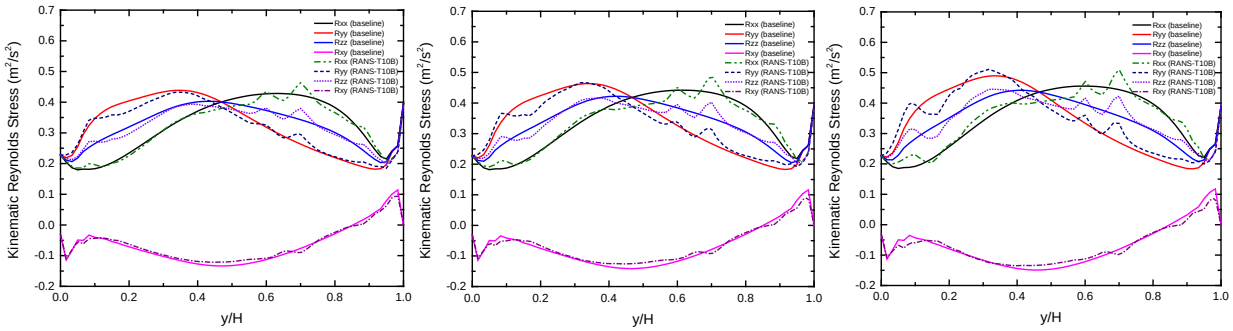


(a) (b) (c)

Figure 8.18. Comparison of kinematic Reynolds stress at x = 0.07 m between the RANS-T10B and baseline at t = (a) 0.010096, (b) 0.01024, and (c) 0.010384 sec with the solver time step size equal to 1.2x10-5 sec.

8.7.2.3. Evaluation of RANS-DL using double the solver time step

In this task, we increase the solver time step to $4.8x10^{-5}$ sec for RANS-T10B. Figure 8.19 depicts RANS-T10B results for three times with the comparison to the baseline. The discrepancy occurs because solutions cannot make time convergence to the same value as the baseline solution.

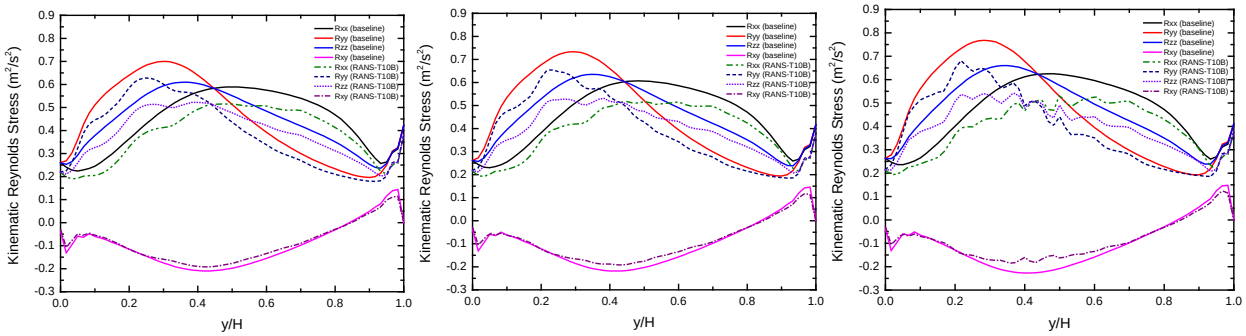


(a) (b) (c)

Figure 8.19. Comparison of kinematic Reynolds stress at x = 0.07 m between RANS-T10B and the baseline at t = (a) 0.010096, (b) 0.01024, and (c) 0.010384 sec with the solver time step size equal to 4.8×10^{-5} sec.

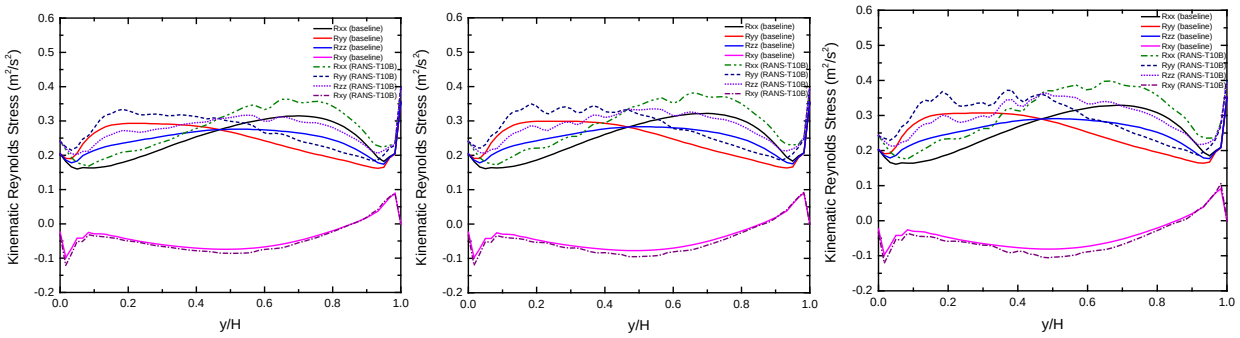
8.7.2.4. Evaluation of RANS-DL by perturbing the inlet velocity

In this task, we solve RANS-T10B with ± 10% perturbations of the inlet velocity. Figure 8.20 and Figure 8.21 show the comparison between RANS-T10B and the baseline with inlet velocities, 11 and 9 m/s. Distinctions between RANS-DL and the baseline are expected since DL-based Reynolds stress is not trained under these two conditions.



(a) (b) (c)

Figure 8.20. Comparison of kinematic Reynolds stress at x = 0.07 m between the baseline and RANS-T10B at t = (a) 0.01012, (b) 0.01024, and (c) 0.01036 sec with the inlet velocity equal to 11 m/s.



(a) (b) (c)

Figure 8.21. Comparison of kinematic Reynolds stress at x = 0.07 m between the baseline and RANS-T10B at t = (a) 0.01012, (b) 0.01024, and (c) 0.01036 sec with the inlet velocity equal to 9 m/s.

8.7.3. Evaluation of the performance of using Type II ML with transient data

The last task is formulated to investigate whether Type II ML can bring RANS-DL to the quasi-steady state from a transient state. Figure 8.22 sketches the streamwise velocity field at the quasi-steady state as the reference solution. Figure 8.23(a) illustrates the simulations with the various start time ranging from t = 0.06 sec to t = 0.4 sec. Figure 8.23(b) gives the results at t = 1 sec. The results reveal that Type II ML can take RANS simulations to the quasi-steady state if initial states are close to the reference solution. Otherwise, RANS-DL by Type II ML can lead to physically unstable solutions.

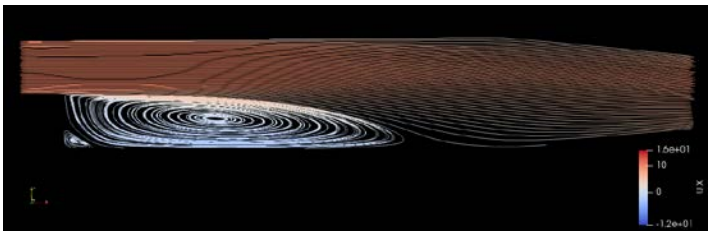


Figure 8.22. Streamwise velocity field for the quasi-steady state by the baseline solution at t = 1 sec.

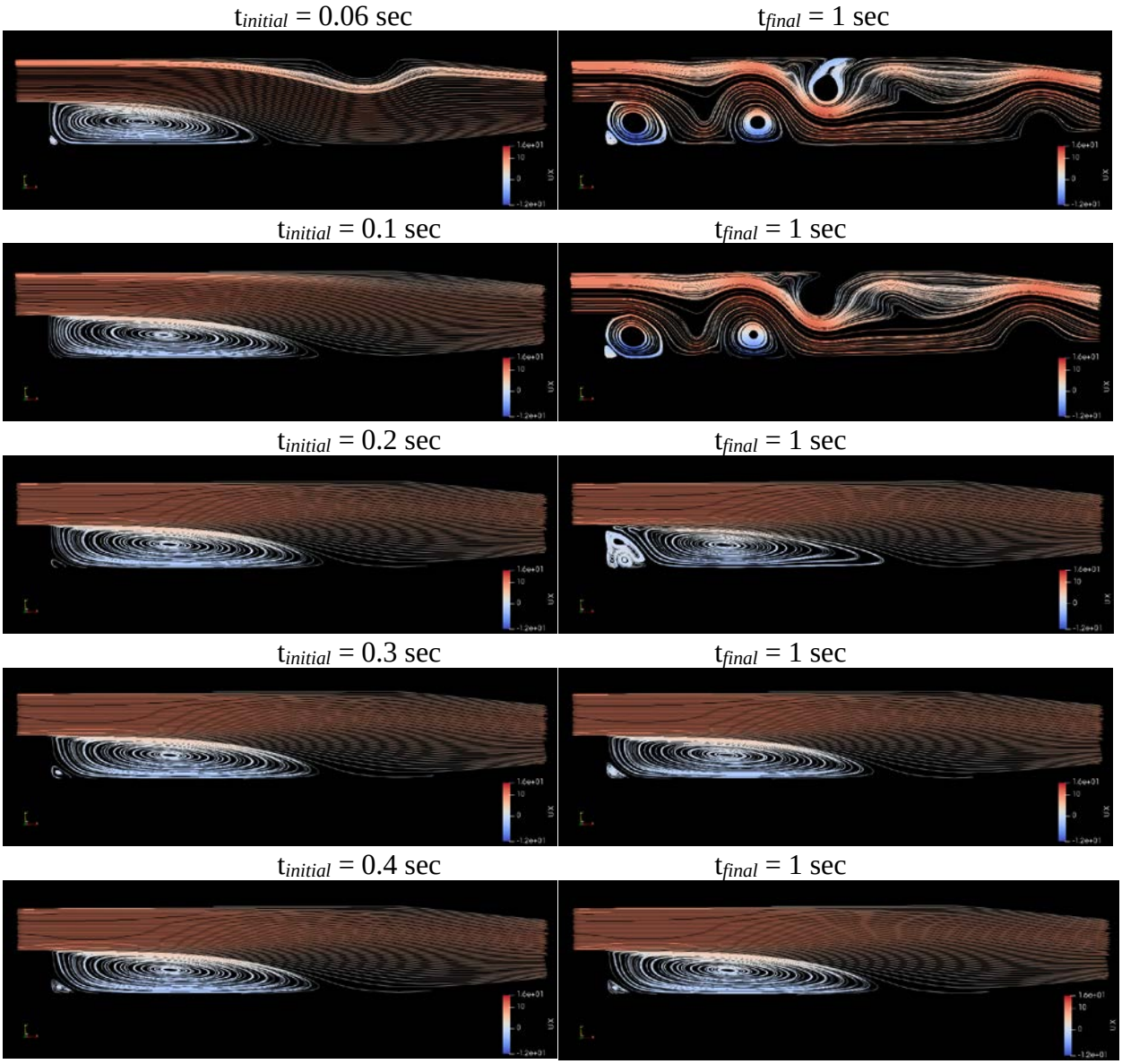
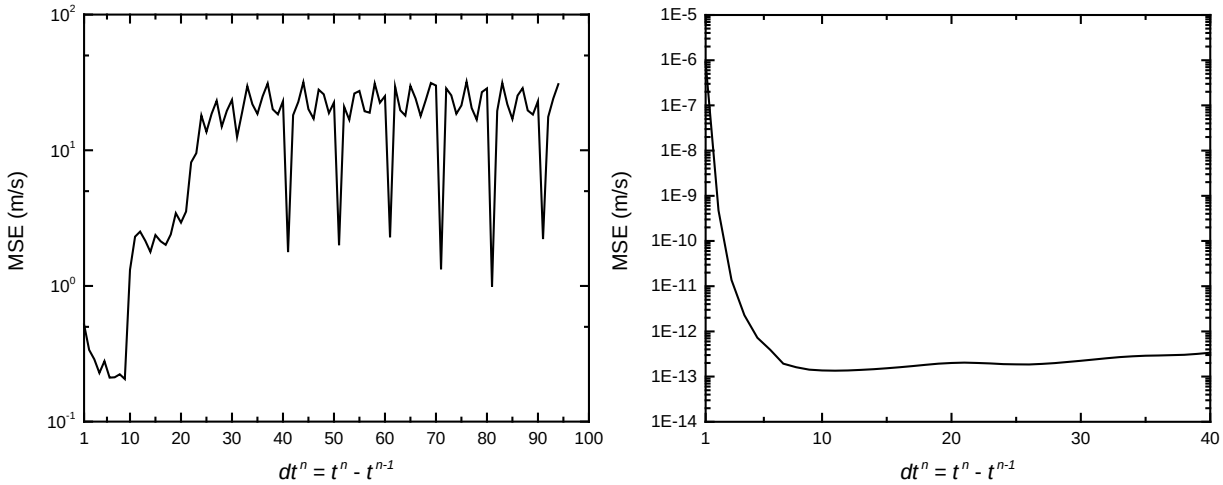


Figure 8.23. (a) Initial streamwise velocity field at different transient steps by the baseline solutions. (b) Final streamwise velocity field at t = 1 sec by the RANS model with the fixed field of the Reynolds stress from the quasi-steady state solution.

Figure 8.24(a) shows MSE analysis for RANS simulation with the initial state at t = 0.06 sec. The MSE is calculated by evaluating the difference (dt^n) between the solutions from two consecutive time steps (t^{n-1} and t^n). When the initial state is far from the quasi-steady state, Type II ML leads to physically unstable solutions. Figure 8.24(b) depicts the result by using the initial

condition at t = 0.6 sec. Since the velocity field is close to the reference value, the solution can reach the quasi-steady state. Figure 8.25 presents initial MSEs and final MSEs by comparing the reference solution to RANS simulations with distinct start times given in Table 8.4. Figure 8.25 indicates that case 6 is the threshold that allows Type II ML to carry solutions from a transient state to the quasi-steady state. It is noted that the initial condition of case 6 is close to the quasi-steady-state solution.



(a) (b)

Figure 8.24. MSE analysis for showing the solution is (a) unstable when the reference Reynolds stress is injected at t = 0.06 sec and (b) the solution is stable when the reference Reynolds stress is injected at t = 0.6 sec.

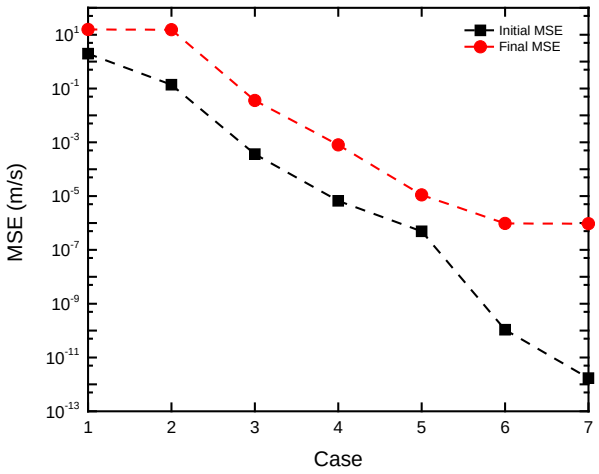


Figure 8.25. MSE analysis for searching the threshold discrepancy between the initial transient and reference velocity fields that can bring the transient solution to the quasi-steady state by Type II ML.

Table 8.4. RANS simulations with different initial states.

Case	1	2	3	4	5	6	7
Start time (sec)	0.06	0.1	0.2	0.3	0.4	0.5	0.6

8.8. Lessons learned

Based on the case study in this work, we observed several properties of Type I and Type II ML. Type I ML can deliver DL-based Reynolds stress to close RANS equations for unsteady flow simulation. Training data only requires spatial derivatives of velocity fields without using time derivative quantities because the time history is embedded in transient data. However, data are required to have sufficient spatiotemporal resolutions to include sufficient transient details that allow DL to discover underlying correlations behind data.

The uncertainty of RANS-DL is accumulated along with simulation time if flow features are in extrapolation domains. This is because the physics is not covered by training data, and the coverage of physics can be quantified by computing the Euclidean distance between two flow features coverage mapping (FFCM). When FFCM shows similar distributions between training and applications, RANS-DL can achieve satisfactory performance in prediction. Therefore, when data is insufficient, DL-based Reynolds stress should not be used to predict flow transients which are far from the training domain.

Type II ML can cause physically unstable solutions when initial states of RANS simulation are far from the quasi-steady-state solution. RANS simulation by Type II ML converges to reference solutions only when initial conditions are close enough to reference solutions. This essence limits the use of Type II ML for unsteady flow simulation.

8.9. Summary

The case study demonstrates data-driven turbulence modeling for transient applications that use RANS equations with DL-base Reynolds stress to replicate transient flow prediction by RANS (k-ε) simulation. The case study also indicates flow features by first-order spatial derivatives of velocity fields are necessary and sufficient to reconstruct the RANS results.

The goal of using DL-based Reynolds stress is to ensure that RANS-DL is globally extrapolating while local variables are within interpolation domains. The results of analysis suggest that DL-based Reynolds stress requires a substantial amount of training data to ensure the predictive capability. Flow features coverage mapping has the potential to quantify values of data that allow us to examine the physics coverage of training datasets.

CHAPTER 9. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

The dissertation research is motivated by the growing interest and development of machine learning models in thermal fluid simulation. The trend is powered by the advent of data-intensive research methods, such as modern thermal fluid experiments and high-fidelity numerical simulations, affordable computing (data processing) power and memory, and progress in machine learning methods, particularly in deep learning (or multilayer neural networks).

Deep learning (DL) has the potential to advance the state-of-the-art in the modeling of nuclear system thermal-hydraulics. Base on the case studies, data-driven modeling with deep learning potentially shortens the model development phase without taking years to decades to understand insights of data. However, there is yet to exist a standard approach to accomplish data-driven modeling of nuclear system thermal-hydraulics (NSTH). The dissertation includes two approaches to investigate how to leverage values of data by machine learning to support NSTH simulation. First, a system is established to characterize different approaches to use machine learning for building data-driven models in NSTH simulation. Framework selection depends on knowledge and data requirements. Second, synthetic examples demonstrate and address the applications and challenges of using deep learning to achieve data-driven modeling of NSTH. With sufficient training data, DL-based closure models work well with conservation equations that can be used for predictions.

The primary contribution of the dissertation is the classification system of machine learning frameworks to illustrate transparent workflows for the development of machine learning models in NSTH simulation. Notably, the development of Type III ML framework can build closure models without the necessity of a scale separation assumption. Section 9.1 highlights the

contributions of the dissertation, and Section 9.2 includes the discussion of recommendations for future work.

9.1. Contributions

1). **The classification of five machine learning frameworks for data-driven modeling of nuclear system thermal-hydraulics to leverage the value of "Big Data," compared to the traditional framework for developing NSTH models.** Five ML frameworks for nuclear system thermal-hydraulics (NSTH) have been introduced in the dissertation including physics-separated ML (PSML or Type I ML), physics-evaluated ML (PEML or Type II ML), physics-integrated ML (PIML or Type III ML), physics-recovered (PRML or Type IV ML), and physics-discovered ML (PDML or Type V ML). With the classification system, it is helpful to select an optimal method to develop data-driven models based on knowledge and data requirements. The frameworks provide the procedures to leverage values of data to support NSTH simulation that potentially extend the application of nuclear codes. Based on the results of case studies, data-driven modeling with deep learning potentially help accelerate the development of thermal-hydraulics models without spending years to decades to understand the data.

2). **The development of Type III ML frameworks to build closure relations without requiring the separation of scales, compared to the traditional development of closure relations by SET data.** Type III (Physics-Integrated Machine Learning) framework is formulated and introduced for the first time in this study. In Type III, conservation equations are involved in the training of machine learning models, thus alleviating the requirement of a scale separation assumption, and potentially reducing the necessity of physics decomposition. Correspondingly, Type III ML framework presents more stringent requirements on modeling and substantially

higher computing resources for training. Based on insights from the case study performed of heat conduction, Type III ML has the highest potential in extracting the value from "big data" in thermal fluid research, while ensuring data-model consistency.

3). **The development of methods to select an optimal deep learning-based closure model to achieve NSTH simulation.** Solving PDE models is complex and requires certain conditions to make the problem well-posed. The case studies of system-level two-phase mixture models demonstrate that the optimal DL-based slip model can be found by using a notion of model-insight consistency. The insight refers to the best knowledge of the problem of interest. The insight also potentially regularizes DL-based models to prevent them from outputting physically unreasonable values or unphysical oscillations. The optimal DL model is defined as models with the maximal predictive capability, given available datasets and insights into machine learning process. Notably, guided by Occam's razor principle, the optimal model should be the deep neural network with the simplest structure that captures the insights and the data within an uncertainty range.

4). **The application of using deep learning-based Reynolds stress to close RANS equations with a substantial amount of training data, compared to the traditional eddy viscosity approach.** Deep learning has the potential to infer a Reynolds stress model directly from data without using the Boussinesq hypothesis of eddy viscosity. The case study of Reynolds-averaged turbulence modeling demonstrates that deep learning can capture the hidden physics behind millions of data points. The result indicates that the DL-based Reynolds stress model can be correlated to the mean flow features by taking the first spatial derivative of the velocity field. By examining flow features coverage mapping, the synthesis example confirms that the selected flow features are necessary and sufficient to preserve the characteristics of Reynolds stress from

data. Flow features coverage mapping can be used to quantify the physics coverage of flow features and has the potential to determine the requirement of data quantity.

9.2. Recommendations for future work

9.2.1. Uncertainty quantification for DL-based closure relations

Uncertainty quantification for deep learning (DL) is a challenge because neural networks include lots of hyperparameters. It is necessary to combine neural networks with Bayesian inference to reflect model uncertainty. In the meantime, experimental uncertainty should also be considered. Future work should formulate case studies to demonstrate how to use Bayesian deep neural networks for thermal fluid simulation.

9.2.2. Uncertainty quantification for PDE constrained DL simulation

DL-based closures tend to accumulate errors during simulation. A regularization method is essential to inform coupled PDE-DL simulation to prevent error amplification.

9.2.3. Challenges on Type III ML

There are technical challenges that need to be addressed before Type III models deliver their promises in practical thermal fluid simulation. The challenges include:

i. Complex interactions of ML-based closures with a system of PDEs (including discontinuities in hyperbolic systems);
ii. Effect of the non-local character of ML-based models on PDE solution methods;
iii. Implementation and effectiveness of multiple closure models, particularly in multiphase and thermal flows;
iv. Useage of training data from IETs and SETs simultaneously.

9.2.4. Assessment of the applicability of a DL generated closure using a code

Numerical solutions by a computer code involve several sources of uncertainty such as discretization error and model form uncertainty. Therefore, it is essential to evaluate whether a DL-based closure created by manufactured solutions from a code is applicable to another code.

9.2.5. Two-phase mixture models with DL-based closures

According to Occam's razor, the best model should be the simplest model that works. Two-phase mixture models have the potential to reduce model form uncertainty of closure relations. It is worthy to explore whether DL-based closures can extend the applicability of two-phase mixture models over a range of flow regimes.

BIBLIOGRAPHY

[1] Ishii M., Hibiki T., Thermo-Fluid Dynamics of Two-Phase Flow, Springer New York, 2010.

[2] Drew D.A., Passman S.L., Theory of Multicomponent Fluids, Springer, 1999.

[3] Kutz J.N., Deep learning in fluid dynamics, Journal of Fluid Mechanics, 814 (2017) 1-4.

[4] LeCun Y., Bengio Y., Hinton G., Deep learning, Nature, 521 (2015) 436-444.

[5] Wulff W., Simulation of two-phase flow in complex systems, Nuclear Technology, 159 (2007) 292–309.

[6] Wulff W., Cheng H.S., Mallen A.N., Modeling and numerical thchniques for high-speed digital simulation of nuclear power plants, in, BNL, 1988.

[7] Wulff W., Critical review of conservation equations for two-phase flow in the U.S. NRC TRACE code, Nuclear Engineering and Design, 241 (2011) 4237-4260.

[8] Wulff W., Computer simulation of two-phase flow in nuclear reactors, Nuclear Engineering and Design, 141 (1993) 303-313.

[9] Team T.R.-D.C.D., RELAP5-3D Code Manual, in, Idaho National Laboratory, 2014.

[10] USNRC, TRACE-V5.0, Theory Manual, TRACE V5.0 User Manual, 2007, TRACE V5.0 Assessment Manual, in, 2007.

[11] R.O. Gauntt R.C., C.M. Erickson, RIG. Gido, R.D. Gasser, S.B. Rodriguez, M.F. Young, MELCOR Computer Code Manuals, in, Sandia National Laboratories, Albuquerque, USA, 2000.

[12] Doster J.M., NE 724 Course Lecture on Two-Fluid Model, in, Department of Nuclear Engineering at NCSU, Raleigh, 2015.

[13] Ng A., Neural Networks and Deep Learning, in, deeplearning.ai on Coursera, 2017.

[14] Dinh N.T., Nourgaliev R., Bui A., Lee H., Perspectives on Nuckear Reactor Thermal Hydraulics, in: NURETH-15, American Nuclear Society, Pisa, Italy, 2013.

[15] Lewis A., Smith R., Williams B., Figueroa V., An information theoretic approach to use high-fidelity codes to calibrate low-fidelity codes, Journal of Computational Physics, 324 (2016) 24-43.

[16] Ackloff R., From Data to Wisdom, Journal of Applied Systems Analysis, 16 (1989) 3-9.

[17] Mell P., Grance T., The NIST Definition of Cloud Computing, in, National Institute of Standards and Technology, Gaithersburg, MD, 2011.

[18] Zhang Z.J., Duraisamy K., Machine Learning Methods for Data-Driven Turbulence Modeling, in, American Institute of Aeronautics and Astronautics, 2015.

[19] Ling J., Kurzawski A., Templeton J., Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics, 807 (2016) 155-166.

[20] Vos R., Farokhi S., Introduction to Transonic Aerodynamics, Springer Netherlands, 2015.

[21] Pope S.B., Turbulent Flows, Cambridge University Press, 2000.

[22] Spalart P., Allmaras S., A One-Equation Turbulence Model for Aerodynamic Flows, in: 30th Aerospace Sciences Meeting and Exhibit, Reno,NV,USA. , 1992.

[23] Wilcox D.C., Formulation of the k-ω Turbulence Model Revisited, AIAA Journal, 46 (2008).

[24] Wilcox D.C., Turbulence Modeling for CFD, 3 ed., DCW Industries, 2006.

[25] Zuber N., Findlay J.A., Average Volumetric Concentration in Two-Phase Flow Systems, J. Heat Transfer, 87 (1965) 453-468.

[26] Thome J.R., Engineering Data Book III, Wolverine Tube, Inc 2010.

[27] Zivi S.M., Estimation of Steady-State Steam Void-Fraction by Means of the Principle of Minimum Entropy Production, Journal of Heat Transfer, 86 (1964) 247-251.

[28] Smith S.L., Void fractions in two-phase flow: a correlation based upon an equal velocity head model, Proc. Instn. Mech. Engrs., 184 (1969) 647-664.

[29] Chisholm D., Pressure gradients due to friction during the flow of evaporating two-phase mixtures in smooth tubes and channels, International Journal of Heat and Mass Transfer, 16 (1973) 347-358.

[30] Ishii M., One-dimensional Drift-flux Model and Constitutive Equations for Relative Motion between Phases in Various Two-phase Flow Regimes, in, Argonne National Lab, 1977.

[31] Zuber N., Staub F.W., Bijwaard G., Kroeger P.G., Steady-state and transient void fraction in two-phase flow systems, in, General Electric Co., 1967.

[32] Ishii M., Chawla T.C., Zuber N., Constitutive equation for vapor drift velocity in two-phase annular flow, AIChE Journal, 22 (1976) 283-289.

[33] Dymola - Dynamic Modeling Laboratory User Manual, in, Dassault Systèmes AB, Lund, 2015.

[34] Fritzson P., Engelson V., Modelica — A unified object-oriented language for system modeling and simulation, in: 12th European Conference Brussels, Belgium, 1988.

[35] Elmqvist H., Mattsson S.E., Otter M., Modelica-a language for physical system modeling, visualization and interaction, in: Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on, 1999, pp. 630-639.

[36] Lind I., Andersson H., Model Based Systems Engineering for Aircraft Systems – How does Modelica Based Tools Fit?, in: Proceedings 8th Modelica Conference, Dresden, Germany, 2011.

[37] Souyri A., Bouskela D., B. Pentori N.K., Pressurized Water Reactor Modelling with Modelica, in: Proc. 5th Intl. Modelica Conference, Vienna, Austria, pp. 127-133.

[38] Weller H.G., Tabor G., Jasak H., Fureby C., A tensorial approach to computational continuum mechanics using object-oriented techniques, Computers in Physics, 12 (1998) 620-631.

[39] OpenFOAM, Open-source Field Operation and Manipulation, Software Package, in, 2015.

[40] Dinh N.T., Validation Data to Support Advanced Code Development, in: NURETH-15, American Nuclear Society, Pisa, Italy, 2013.

[41] Abu-Mostafa Y.S., Magdon-Ismail M., Lin H.-T., Learning From Data, AMLBook, 2012.

[42] Domingos P., The Master Algorithm, Basic Books, 2015.

[43] Ling J., Templeton J., Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty, Physics of Fluids, 27 (2015) 085103.

[44] Hornik K., Stinchcombe M., White H., Multilayer Feedforward Networks are Universal Approximators, Neural Networks, 2 (1989) 359-366

[45] Hinton G.E., Osindero S., Teh Y., A fast learning algorithm for deep belief nets, 18 (2006) 1527-1554

[46] Heaton J., Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks, Heaton Research, Inc., Chesterfield, MO, 2015.

[47] Lecun Y., Bottou L., Bengio Y., Haffner P., Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86 (1998) 2278-2324.

[48] Werbos P.J., Generalization of backpropagation with application to a recurrent gas market model, Neural Networks, 1 (1988) 339-356.

[49] Kohonen T., Self-organized formation of topologically correct feature maps, Biological Cybernetics, 43 (1982) 59-69.

[50] Hinton G.E., Sejnowski T.J., Learning and relearning in Boltzmann machines, MIT Press, 1986.

[51] Nair V., Hinton G.E., Rectified linear units improve restricted boltzmann machines, in, 2010, pp. 807–814.

[52] Harris D., Harris S., Digital Design and Computer Architecture, 2nd ed., Morgan Kaufmann.

[53] Li F.-F., Karpathy A., Johnson J., CS231 Course Lectures on Convolutional Neural Networks for Visual Recognition, in, Stanford University, 2016.

[54] Teh Y.W., Hinton G.E., Rate-coded Restricted Boltzmann Machines for Face Recognition, MIT Press, 2001.

[55] Krizhevsky A., Sutskever I., Hinton G.E., ImageNet Classification with Deep Convolutional Neural Networks, 2012.

[56] Dolhansky B., Artificial Neural Networks: Mathematics of Backpropagation, in, 2014.

[57] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G.S., Davis A., Dean J., Devin M., others, Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467, (2016).

[58] Box G.E.P., Draper N.R., Empirical Model-Building and Response Surfaces, Wiley, 1987.

[59] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, 48 (2014) 1929-1958.

[60] Gal Y., Uncertainty in Deep Learning, in: Department of Engineering, University of Cambridge, Cambridge, UK, 2016.

[61] Team T.D., Theano: A Python framework for fast computation of mathematical expressions, arXiv e-prints, abs/1605.02688 (2016).

[62] Collobert R., Kavukcuoglu K., Farabet C., Torch7: A Matlab-like Environment for Machine Learning, in, BigLearn, NIPS Workshop, 2011.

[63] Ierusalimschy R., Programming in Lua, Lua.org, 2003.

[64] Paszke A., Gross S., Chintala S., Chanan G., Yang E., DeVito Z., Lin Z., Desmaison A., Antiga L., Lerer A., Automatic differentiation in PyTorch, in: Conference on Neural Information Processing Systems, 2017.

[65] Jia Y., Shelhamer E., Donahue J., Karayev S., Long J., Girshick R., Guadarrama S., Darrell T., Caffe: Convolutional Architecture for Fast Feature Embedding, in: Proceedings of the 22nd ACM international conference on Multimedia, ACM, Orlando, Florida, USA, 2014, pp. 675-678.

[66] Agarwal A., Akchurin E., Basoglu C., Chen G., Cyphers S., Droppo J., An Introduction to Computational Networks and the Computational Network Toolkit, in, Microsoft Technical Report, 2014.

[67] Tompson J., Schlachter K., Sprechmann P., Perlin K., Accelerating Eulerian Fluid Simulation With Convolutional Networks, arXiv:1607.03597, (2017).

[68] Ladický L., Jeong S., Solenthaler B., Pollefeys M., Gross M., Data-driven fluid simulations using regression forests, ACM Trans. Graph., 34 (2015) 1-9.

[69] Brunton S.L., Proctor J.L., Kutz J.N., Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences, 113 (2016) 3932–3937.

[70] Mills K., Spanner M., Tamblyn I., Deep learning and the Schrödinger equation, arXiv preprint arXiv:1702.01361, (2017).

[71] Ma M., Lu J., Tryggvason G., Using statistical learning to close two-fluid multiphase flow equations for a simple bubbly system, Physics of Fluids, 27 (2015) 092101.

[72] Ma M., Lu J., Tryggvason G., Using statistical learning to close two-fluid multiphase flow equations for bubbly flows in vertical channels, International Journal of Multiphase Flow, 85 (2016) 336-347.

[73] Tryggvason G., Ma M., Lu J., DNS–Assisted Modeling of Bubbly Flows in Vertical Channels, Nuclear Science and Engineering, 184 (2016) 312-320.

[74] Parish E.J., Duraisamy K., A paradigm for data-driven predictive modeling using field inversion and machine learning, Journal of Computational Physics, 305 (2016) 758-774.

[75] Tracey B.D., Duraisamy K., Alonso J.J., A Machine Learning Strategy to Assist Turbulence Model Development, in, American Institute of Aeronautics and Astronautics, 2015.

[76] Wu J.-L., Wang J.-X., Xiao H., Ling J., Physics-informed machine learning for predictive turbulence modeling: A priori assessment of prediction confidence, arXiv:1607.04563, (2016).

[77] Wang J.X., Wu J.L., Ling J., Iaccarino G., Xiao H., A Comprehensive Physics-Informed Machine Learning Framework for Predictive Turbulence Modeling, arXiv:1701.07102, (2017).

[78] Wang J.-X., Wu J.-L., Xiao H., Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, Physical Review Fluids, 2 (2017) 034603.

[79] Bar-Yam Y., Separation of scales: Why complex systems need a new mathematics, in, NECSI 2015.

[80] Spriggs J., GSN - The Goal Structuring Notation, 1 ed., Springer-Verlag London, 2012.

[81] Kelly T.P., Arguing Safety – A Systematic Approach to Safety Case Management, in: Department of Computer Science, University of York, UK, 1998.

[82] Jun-Ichi Y., Subgrid-scale physical parameterization in atmospheric modeling: How can we make it consistent?, Journal of Physics A: Mathematical and Theoretical, 49 (2016) 284001.

[83] Yano J.-I., What is Scale Separation?: A Theoretical Reflection, in, GAME/CNRM, M´et´eo-France and CNRS (URA 1357), France, 2012.

[84] Jolliffe I.T., Principal Component Analysis, Springer-Verlag New York, 2002.

[85] Hadamard J., Lectures on Cauchy's Problem in Linear Partial Differential Equations, Yale University Press, New Haven, 1923.

[86] Tracey B., Duraisamy K., Alonso J., Application of Supervised Learning to Quantify Uncertainties in Turbulence and Combustion Modeling, in, American Institute of Aeronautics and Astronautics, 2013.

[87] Singh A.P., Duraisamy K., Using field inversion to quantify functional errors in turbulence closures, Physics of Fluids, 28 (2016) 045110.

[88] Chang C.-W., Dinh N., Cetiner S.M., Physics-Constrained Machine Learning for Two-Phase Flow Simulation Using Deep Learning-Based Closure Relation, in: American Nuclear Society Winter Meeting, Washington, DC, 2017, pp. 1749-1752.

[89] Chang C.-W., Dinh N.T., A Study of Physics-Informed Deep Learning for System Fluid Dynamics Closures, in: American Nuclear Society Winter Meeting, Anaheim, CA, 2016, pp. 1785-1788.

[90] Ling J., Jones R., Templeton J., Machine learning strategies for systems with invariance properties, Journal of Computational Physics, 318 (2016) 22-35.

[91] Zhu Y., Dinh N.T., A Data-Driven Approach for Turbulence Modeling, in: NURETH-17, American Nuclear Society, Xi'an, China, 2017.

[92] Rasmussen C.E., Ghahramani Z., Occam's razor, in: Advances in neural information processing systems, 2001, pp. 294-300.

[93] Hanna B.N., Dinh N.T., Youngblood R.W., Bolotnov I.A., Coarse-Grid Computational Fluid Dynamics (CG-CFD) Error Prediction using Machine Learning, under review, (2017).

[94] Konishi S., Introduction to Multivariate Analysis: Linear and Nonlinear Modeling, Chapman and Hall, 2014.

[95] Munson B.R., Young D.F., Okiishi T.H., Fundamentals of Fluid Mechanics, 5th ed., John Wiley & Sons, Inc.

[96] de Swart J.J.B., A simple ODE solver based on 2-stage Radau IIA, Journal of Computational and Applied Mathematics, 84 (1997) 277-280.

[97] Scalabrin G., Condosta M., Marchi P., Flow boiling of pure fluids: local heat transfer and flow pattern modeling through artificial neural networks, International Journal of Thermal Sciences, 45 (2006) 739-751.

[98] Scalabrin G., Condosta M., Marchi P., Modeling flow boiling heat transfer of pure fluids through artificial neural networks, International Journal of Thermal Sciences, 45 (2006) 643-663.

[99] Kingma D.P., Ba J., Adam: A Method for Stochastic Optimization, arXiv:1412.6980, (2014).

[100] Casella F., M. Otter K.P., Richter C., Tummescheit H., The Modelica Fluid and Media Library for Modeling of Incompressible and Compressible Thermo-Fluid Pipe Networks, in: Modelica 2006 Conference, Vienna, 2006.

[101] Swamee P.K., Jain A.K., Explicit Equations for Pipe-Flow Problems, Journal of the Hydraulics Division, 102 (1976) 657-664.

[102] Zuber N., Staub F.W., Bijwaard G., Kroeger P.G., Steady State and Transient Void Fraction in Two-phase Flow Systems, in, General Electric Co., 1967.

[103] Shannak B.A., Frictional pressure drop of gas liquid two-phase flow in pipes, Nuclear Engineering and Design, 238 (2008) 3277-3284.

[104] Chanda S., Balaji C., Venkateshan S.P., Yenni G.R., Estimation of principal thermal conductivities of layered honeycomb composites using ANN–GA based inverse technique, International Journal of Thermal Sciences, 111 (2017) 423-436.

[105] Patankar S.V., Numerical Heat Transfer and Fluid Flow, CRC Press, 1980.

[106] Bishop C.M., Mixture density networks, in, Neural Computing Research Group, Department of Computer Science, Aston University, Birmingham, U.K., 1994.

[107] Craft T.J., Launder B.E., Suga K., Development and application of a cubic eddy-viscosity model of turbulence, International Journal of Heat and Fluid Flow, 17 (1996) 108-115.

[108] Gad-el-Hak M., Bandyopadhyay P.R., Reynolds Number Effects in Wall-Bounded Turbulent Flows, Applied Mechanics Reviews, 47 (1994) 307-365.

[109] Domino S.P., Moen C.D., Burns S.P., Evans G.H., SIERRA/Fuego: a multimechanics fire environment simulation tool, in: American Institute of Aeronautics and Astronautics, 2003.

[110] Pinelli A., Uhlmann M., Sekimoto A., Kawahara G., Reynolds number dependence of mean flow structure in square duct turbulence, Journal of Fluid Mechanics, 644 (2010) 107-122.

[111] Moser R.D., Kim J., Mansour N.N., Direct numerical simulation of turbulent channel flow up to Reτ=590, Physics of Fluids, 11 (1999) 943-945.

[112] Ling J., Ruiz A., Lacaze G., Oefelein J., Uncertainty Analysis and Data-Driven Model Advances for a Jet-in-Crossflow, Journal of Turbomachinery, 139 (2016) 021008-021008-021009.

[113] Ruiz A.M., Lacaze G., Oefelein J.C., Flow topologies and turbulence scales in a jet-in-cross-flow, Physics of Fluids, 27 (2015) 045101.

[114] Ray J., Lefantzi S., Arunajatesan S., Dechant L., Bayesian calibration of a k–ε turbulence model for predictive jet-in-crossflow simulations, in, American Institute of Aeronautics and Astronautics, 2014.

[115] Marquillie M., Ehrenstein U.W.E., Laval J.-P., Instability of streaks in wall turbulence with adverse pressure gradient, Journal of Fluid Mechanics, 681 (2011) 205-240.

[116] Farabet C., Couprie C., Najman L., LeCun Y., Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers, in: Proceedings of the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK, 2012.

[117] Menter F., Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications, AIAA Journal, 32 (1994).

[118] Chang C.-W., Dinh N.T., Classification of Machine Learning Frameworks for Data-Driven Thermal Fluid Models, (2018).

[119] Patankar S.V., Spalding D.B., A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, International Journal of Heat and Mass Transfer, 15 (1972) 1787-1806.

[120] Issa R.I., Solution of the implicitly discretised fluid flow equations by operator-splitting, Journal of Computational Physics, 62 (1986) 40-65.

[121] Chorin A.J., Numerical solutions of the Navier–Stokes equations, Math. Comput., 22 (1968) 745.

[122] Yanenko N.N., The Method of Fractional Steps for Solving Multi-Dimensional Problems of Mathematical Physics in Several Variables, Springer-Verlag, Berlin, 1971.

[123] Pitz R.W., Daily J.W., Combustion in a Turbulent Mixing Layer Formed at a Rearward-Facing Step, AIAA Journal, 21 (1987).

[124] Ahmed U., Prosser R., A posteriori assessment of algebraic scalar dissipation models for RANS simulation of premixed turbulent combustion, Flow, Turbulence and Combustion, (2017).

[125] Lloyd S.P., Least squares quantization in PCM, in, Bell Lab, 1957.

[126] MacQueen J.B., Some methods for classification and analysis of multivariate observations, in: Cam L.M.L., Neyman J. (Eds.) Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, University of California Press, 1967, pp. 281.

[127] Maaten L.v.d., Hinton G., Visualizing Data using t-SNE, Journal of Machine Learning Research, 9 (2008).

[128] Kullback S., Leibler R.A., On Information and Sufficiency, Ann. Math. Statist., 22 (1951) 79-86.

[129] Sergey Ioffe, Szegedy C., Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv:1502.03167, (2015).