· UNIVERSITY OF OXFORD ·

· DEPARTMENT OF ENGINEERING SCIENCE ·

Machine Learning Wall Functions for Turbulent Flows

Jianou Jiang

St Anne's College

A thesis submitted for the degree of

Doctor of Philosophy

Supervised by Professor Budimir Rosic

Trinity Term 2026

Declaration

I declare that this thesis is entirely my own work, and except where otherwise stated, describes my own research.

Jianou Jiang

St Anne's College

This thesis is dedicated to . . .

Abstract

This thesis develops a unified machine learning framework for near-wall modeling in transitional and turbulent flows, with a focus on thermal and buoyancy-coupled boundary layers. Traditional wall functions used in computational fluid dynamics (CFD) often fail under complex conditions such as flow separation, heat transfer, or strong curvature. To address these limitations, we propose a data-driven wall function model trained on structured stencil inputs from coarse mesh simulations, supervised by corresponding high-fidelity fine mesh results.

The framework introduces a library of non-dimensional features grounded in fluid mechanics, enabling physically-informed learning across regimes and Reynolds numbers. Three modeling strategies are explored: physics-encoded inputs, neuron-feature alignment in hidden layers, and physics-constrained learning via local residual loss terms derived from the Navier–Stokes and energy equations.

Validation is performed across passive and buoyancy-coupled thermal flows in 2D and extended to 3D cases, demonstrating strong generalization to untrained geometries and conditions. The model is integrated directly into the OpenFOAM solver as a custom boundary condition using a Python-C++ interface. Compared to classical wall functions, the learned model provides more accurate wall-adjacent velocity, shear stress, and heat flux predictions, particularly in regions of flow separation or vertical thermal deflection.

Keywords: Wall functions, Physics-informed learning, Neural networks, CFD, Heat transfer, Buoyancy, OpenFOAM

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Budimir Rosic, for his invaluable guidance, continuous support, and patience throughout this research. His expertise in turbomachinery and computational fluid dynamics has been instrumental in shaping this work, and his encouragement has been a constant source of motivation.

I am grateful to the Department of Engineering Science at the University of Oxford for providing an exceptional research environment and the computational resources necessary for this work. The high-performance computing facilities were essential for the extensive simulations and machine learning experiments conducted in this thesis.

I would also like to thank my colleagues in the Turbomachinery Group for the stimulating discussions and collaborative atmosphere. Their feedback and insights have contributed significantly to the development of the ideas presented here.

Finally, I extend my heartfelt thanks to my family for their unwavering support and understanding throughout my doctoral studies. Their encouragement has been the foundation upon which this work was built.

Publications and Presentations

List the achievements and publications you've made during this DPhil journey.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The accurate prediction of turbulent flows near solid boundaries remains one of the outstanding challenges in computational fluid dynamics [1–3]. While the Reynolds-averaged Navier-Stokes equations provide a framework for simulating turbulent flows at practical Reynolds numbers [4, 5], their solution requires resolving the steep gradients that develop in the near-wall region—or alternatively, bridging this region with models that capture the essential physics without explicit resolution [6, 7]. These bridging models, known as wall functions, have served engineering practice for over half a century [8, 9]. Yet as computational demands grow more complex—involving flow separation, thermal coupling, and unsteady phenomena—the limitations of classical wall functions become increasingly apparent [10–12]. This thesis develops a machine learning framework to address these limitations [5, 13, 14], creating wall function models that combine the computational efficiency of classical approaches with the flexibility to learn directly from high-fidelity simulation data [2, 15, 16].

1.1 Industrial Motivation: Where Wall Functions Matter

The economic and engineering significance of accurate wall function modeling extends across virtually every industry that relies on fluid flow prediction [17, 18]. The wall shear stress τ_w directly determines drag forces that govern fuel consumption in transportation [19, 20], while the wall heat flux q_w controls thermal management in everything from electronic cooling to nuclear reactor safety [21, 22]. Errors in wall function predictions propagate into the quantities that engineers ultimately care about: drag coefficients, heat transfer rates, pressure drops, and system efficiencies [23, 24].

Aerospace and gas turbines. Modern aircraft wings operate at Reynolds numbers exceeding 10^7, where even small errors in skin friction prediction translate to significant fuel burn penalties. A 1% error in drag coefficient for a commercial aircraft represents millions of dollars in annual fuel costs across a fleet. The situation is even more critical in gas turbine engines, where blade cooling effectiveness depends on accurate prediction of heat transfer coefficients. Underpredicting heat flux leads to blade failure; overpredicting leads to inefficient use of cooling air that reduces engine efficiency. Both scenarios carry substantial economic consequences.

Automotive aerodynamics. Vehicle manufacturers invest heavily in CFD-based aerodynamic optimization to reduce drag and improve fuel efficiency. With electric vehicles, where range anxiety drives consumer decisions, accurate drag prediction becomes even more critical. The complex geometry of vehicle underbodies, wheel wells, and cooling systems creates flow separation and reattachment patterns that challenge classical wall functions. Wind tunnel validation remains essential precisely because CFD predictions using standard wall treatments are often unreliable in these regions.

Building ventilation and HVAC. Indoor air quality and thermal comfort depend on accurate prediction of airflow patterns near walls, floors, and ceilings. The buoyancy-driven flows that develop near heated or cooled surfaces are particularly challenging for classical wall functions, which assume negligible density variations. With buildings accounting for approximately 40% of global energy consumption, improvements in HVAC system design through better CFD modeling could yield substantial energy savings.

Nuclear reactor thermal-hydraulics. Safety analysis for nuclear reactors requires accurate prediction of heat transfer from fuel elements to coolant. The consequences of underpredicting heat flux—potential fuel damage or meltdown—are severe enough that regulatory bodies require substantial safety margins. More accurate wall function models could reduce these margins while maintaining safety, enabling more efficient reactor designs.

Marine and offshore engineering. Ship hull resistance, offshore platform loading, and underwater vehicle performance all depend on turbulent boundary layer predictions at high Reynolds numbers. The economic incentive is direct: fuel costs typically represent 50–70% of

a commercial vessel's operating expenses, making drag reduction a primary design objective.

These applications share a common challenge: the flows of engineering interest almost never satisfy the equilibrium assumptions of classical wall functions. Pressure gradients, flow separation, thermal coupling, and geometric complexity are the norm rather than the exception. The industrial motivation for improved wall function modeling is thus not academic curiosity but practical necessity.

1.2 The Near-Wall Challenge in Turbulent Flow Simulation

Turbulent boundary layers exhibit a characteristic structure that has fascinated and frustrated fluid dynamicists since Ludwig Prandtl's seminal 1904 paper on boundary layer theory [25–27]. Near solid surfaces, velocity gradients become extremely steep as the flow transitions from the no-slip condition at the wall to the free-stream velocity over a distance that may be orders of magnitude smaller than the overall domain scale [28, 29]. Resolving these gradients directly requires mesh cells with wall-normal dimensions on the order of the viscous length scale $\delta_\nu = \nu/u_\tau$, where ν is the kinematic viscosity and $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity.

For high Reynolds number flows typical of engineering applications, this resolution requirement translates to meshes with millions or billions of cells concentrated in thin layers near walls. The computational cost scales approximately as $Re^{1.8}$ for wall-resolved Large Eddy Simulation, making direct resolution impractical for many industrial applications. The situation becomes even more challenging when thermal effects are present, as the thermal boundary layer introduces additional thin regions requiring resolution.

1.2.1 The Computational Cost of Wall Resolution

The stark contrast between wall-resolved and wall-modeled simulations is best illustrated through concrete numbers. Table 1.1 compares mesh requirements for a representative internal flow at different Reynolds numbers.

Table 1.1: Approximate mesh requirements for wall-resolved versus wall-modeled RANS simulation of channel flow. Wall-resolved meshes require $y_1^+ < 1$; wall-modeled meshes use $y_1^+ \approx 30$–100.

Reynolds Number	Wall-Resolved Cells	Wall-Modeled Cells	Reduction Factor
$Re = 10^4$	$\sim 10^5$	$\sim 10^4$	$10\times$
$Re = 10^5$	$\sim 10^6$	$\sim 10^5$	$10\times$
$Re = 10^6$	$\sim 10^7$	$\sim 10^5$	$100\times$
$Re = 10^7$	$\sim 10^8$	$\sim 10^6$	$100\times$

For industrial applications at $Re \sim 10^7$, wall-resolved simulations require meshes that push the limits of current supercomputing capabilities. Even with wall functions, a full aircraft simulation may require tens of millions of cells and days of compute time. The mesh reduction enabled by wall functions is not merely convenient—it is often the difference between computationally feasible and infeasible.

However, this computational savings comes at the cost of accuracy. Wall functions approximate the physics of the near-wall region rather than resolving it, and these approximations fail when the underlying assumptions are violated. The challenge addressed by this thesis is to create wall function models that maintain the computational efficiency of classical approaches while substantially improving accuracy under non-equilibrium conditions.

1.2.2 Classical Wall Function Theory

Wall functions offer an alternative to direct resolution: rather than resolving the near-wall region explicitly, the first computational cell is placed in the logarithmic layer (typically at $y^+ \equiv yu_\tau/\nu \approx 30$–$100$), and algebraic relationships derived from similarity theory provide the wall shear stress and heat flux. This approach reduces mesh requirements by factors of 10–100, bringing turbulent flow simulation within reach of routine engineering analysis.

The classical wall functions developed by Launder and Spalding in the 1970s [6] assume local equilibrium between turbulence production and dissipation, small pressure gradients, and attached flow—conditions well-satisfied in fully-developed channel flow and flat plate boundary layers [3, 19, 30]. Under these conditions, the logarithmic law of the wall provides an accurate

relationship between the wall-parallel velocity at the first cell centre and the wall shear stress:

$$u^+ = \frac{1}{\kappa} \ln(y^+) + B \tag{1.1}$$

where $\kappa \approx 0.41$ is the von Kármán constant and $B \approx 5.2$ is an integration constant. An analogous relationship, based on Reynolds analogy and a turbulent Prandtl number, extends this approach to thermal wall functions:

$$T^+ = Pr_t \left(\frac{1}{\kappa} \ln(y^+) + B_T \right) \tag{1.2}$$

where $Pr_t \approx 0.85$ is the turbulent Prandtl number and B_T depends on the molecular Prandtl number.

1.3 Limitations of Classical Wall Functions

The equilibrium assumptions underlying classical wall functions fail precisely where engineering interest is greatest. Consider flows involving:

Adverse pressure gradients: As flow decelerates under an adverse pressure gradient [31, 32], the boundary layer thickens, the velocity profile deviates from logarithmic form [33, 34], and the assumption of local equilibrium breaks down. The classical wall function overpredicts wall shear stress and cannot capture the onset of separation [10, 35].

Flow separation and reattachment: In separated regions, the wall shear stress becomes negative [12, 36, 37], making the friction velocity $u_\tau = \sqrt{|\tau_w|/\rho}$ ill-defined for determining wall-normal distance scaling. More fundamentally, the reversed flow near the wall bears no resemblance to the attached boundary layer profile that the logarithmic law describes [11, 38].

Strong heating or cooling: Thermal wall functions based on Reynolds analogy [39, 40] assume that the turbulent transport of heat parallels that of momentum. While approximately valid for moderate temperature differences, this assumption fails when buoyancy effects become significant [41] or when the turbulent Prandtl number varies substantially from its commonly-assumed value of 0.85 [42].

Complex geometry: Three-dimensional flows over curved surfaces, in corners, or around obstacles develop secondary flows and pressure gradient effects that violate the two-dimensional, streamwise-invariant assumptions of classical wall function theory.

These limitations manifest as systematic errors in practical CFD simulations. Flow separation is predicted too late (or not at all), heat transfer is overpredicted in recirculation zones, and the overall solution accuracy degrades precisely in the regions of greatest engineering interest.

1.3.1 A Motivating Example: The Asymmetric Diffuser

The asymmetric diffuser provides a compelling illustration of classical wall function failure. Consider a channel where one wall remains flat while the opposite wall diverges at an angle, creating an adverse pressure gradient that eventually causes flow separation on the expanding wall. This geometry, used extensively in this thesis, spans conditions from mild deceleration to massive separation depending on the expansion ratio and divergence angle.

Figure 1.1 illustrates the flow physics. In the attached region upstream of separation, classical wall functions perform adequately: the flow approximates equilibrium, and the logarithmic law provides reasonable estimates of wall shear stress. As the adverse pressure gradient intensifies, however, the boundary layer responds in ways that the equilibrium assumption cannot capture. The velocity profile distorts from logarithmic form, turbulence production exceeds dissipation, and the classical wall function progressively overpredicts τ_w.

At separation, the classical approach fails catastrophically. The wall shear stress passes through zero and becomes negative in the recirculating region. The friction velocity becomes imaginary, breaking the non-dimensionalization that underlies the entire wall function framework. Most CFD codes handle this by switching to an ad-hoc treatment—often linear interpolation through the zero crossing—but such fixes have no physical basis and typically produce poor predictions of the separation and reattachment locations.

The reattachment region presents additional challenges. As the separated flow reattaches, it brings high-momentum fluid toward the wall, creating a recovery profile that differs substantially from the equilibrium logarithmic law. Classical wall functions, designed for attached flow, cannot capture this recovery process.

This single geometry thus encapsulates the full range of wall function challenges: equilibrium flow, adverse pressure gradient effects, separation, reversed flow, and reattachment. A wall function model that performs well across all these conditions would represent a substantial advance over classical approaches.

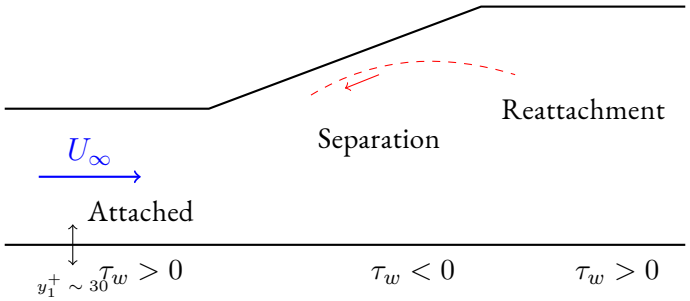


Figure 1.1: Schematic of flow through an asymmetric diffuser, illustrating attached flow, separation, recirculation, and reattachment. Classical wall functions fail in the separation and recirculation regions where $\tau_w \leq 0$.

1.4 The Machine Learning Opportunity

Machine learning offers a fundamentally different approach to wall function modeling. Rather than deriving algebraic relationships from similarity theory and asymptotic analysis, machine learning models learn directly from data. High-fidelity simulations—Direct Numerical Simulation (DNS) or well-resolved Large Eddy Simulation (LES)—provide training data that captures the full complexity of near-wall turbulence, including the departures from equilibrium that classical theory cannot accommodate.

Neural networks, as universal function approximators, can represent the nonlinear relationships between near-wall flow quantities and wall fluxes without the limiting assumptions of classical theory. A network trained on diverse flow conditions can interpolate (and potentially extrapolate) to conditions not explicitly represented in the training data, guided by the underlying patterns in the physics rather than by prescribed functional forms.

However, purely data-driven approaches face their own challenges. Neural networks may learn spurious correlations that happen to work within the training distribution but fail catastrophically outside it. Without physical constraints, predictions may violate conservation laws, produce non-physical negative shear stresses, or extrapolate wildly in regions where training

data is sparse. The generalization problem—ensuring that models trained on canonical flows perform well in practical applications—remains the central challenge of machine learning for turbulence modeling.

This thesis addresses these challenges through a unified framework that combines data-driven learning with physics-informed constraints. The key insight is that physics knowledge can be incorporated at multiple levels: in the selection and construction of input features, in the architecture of the neural network itself, and in the loss function used during training. By encoding physical principles at each of these levels, we create models that are both flexible enough to learn from data and constrained enough to respect fundamental physics.

1.5 Three Approaches to Physics-Informed Learning

This thesis develops and compares three complementary methods for incorporating physics knowledge into neural network wall functions. Each method operates at a different level of the machine learning pipeline, and together they provide a comprehensive framework for physics-informed wall function modeling.

1.5.1 Method A: Physics-Encoded Input Features

The first approach encodes physics in the *input representation*. Rather than providing the neural network with raw simulation outputs (velocity components, pressure, temperature at mesh points), we construct non-dimensional features that respect the scaling laws of turbulent boundary layers.

Classical wall function theory teaches us that the relevant variables are wall-scaled quantities: $y^+ = yu_\tau/\nu$, $u^+ = u/u_\tau$, and similar dimensionless groups. These scalings collapse boundary layer data across Reynolds numbers, suggesting that neural networks should receive inputs in these natural coordinates.

We extend this idea to construct a library of 58 physics-based features including:

- Wall-scaled distances and velocities (y^+, u^+, v^+)
- Pressure gradient terms ($p_x^+ = \nu/(\rho u_\tau^3) \cdot \partial p/\partial x$)

- Velocity gradient and curvature quantities
- Thermal equivalents (T^+, y_T^+ based on thermal friction velocity)
- Deformation tensor invariants

Chapter 5 demonstrates that this physics-encoded input representation enables generalization across Reynolds numbers and achieves accuracy comparable to much larger sets of primitive variables.

1.5.2 Method B: Physics-Guided Network Architecture

The second approach investigates what neural networks learn *internally* when trained on primitive inputs. Even without explicit physics encoding in the inputs, neural networks must construct internal representations to solve the wall function prediction problem. Do these representations correspond to physically meaningful quantities?

Through systematic correlation analysis between hidden layer neuron activations and physics-based features, we find that networks spontaneously discover physics-relevant quantities. Neurons in trained networks correlate strongly with y^+, $\ln(y^+)$, and pressure gradient scalings—precisely the quantities that classical wall function theory identifies as important.

Most remarkably, certain features emerge across network architectures of different sizes. A network with 8 hidden neurons and one with 32 hidden neurons both develop neurons correlated with wall-scaled distance. This *architecture invariance* suggests that these features are fundamental to the learning problem, not artifacts of particular network configurations.

Chapter 6 explores these findings, validating the physics-based input design of Method A while revealing the implicit physics learning capabilities of neural networks.

1.5.3 Method C: Physics-Constrained Loss Functions

The third approach encodes physics in the *training objective*. Standard supervised learning minimizes prediction error on training data, but nothing prevents the network from learning solutions that violate physical conservation laws. Physics-Informed Neural Networks (PINNs)

address this by adding physics residuals to the loss function:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_{\text{physics}} \mathcal{L}_{\text{physics}} \tag{1.3}$$

For wall functions, we develop a local stencil-based PINN that evaluates conservation law residuals on the same near-wall stencil used for input features. The physics loss penalizes violations of:

- Streamwise momentum conservation
- Wall-normal momentum (including buoyancy)
- Energy conservation
- Mass conservation (incompressibility)

Unlike global PINNs that require automatic differentiation over entire computational domains, this local approach maintains computational efficiency while enforcing physics constraints where they matter most: in the near-wall region.

Chapter 7 develops this approach and characterizes the trade-off between fitting accuracy and physical consistency.

1.5.4 Complementary Methods

These three methods are not mutually exclusive; they address physics encoding at different stages of the machine learning pipeline:

Method	**Stage**	**Physics Mechanism**
A: Physics Inputs	Data preparation	Non-dimensional features
B: Physics Architecture	Model design	Implicit feature discovery
C: Physics Loss	Training objective	Conservation constraints

A comprehensive wall function model might combine all three: physics-encoded inputs, architectures designed to facilitate physics discovery, and loss functions that enforce conservation. The experimental chapters of this thesis evaluate each method independently to understand their individual contributions before considering combinations.

1.6 Research Questions

This thesis addresses five specific research questions:

RQ1: Can machine learning models predict wall shear stress and heat flux accurately across diverse flow conditions? This baseline question establishes whether neural networks can learn the wall function mapping at all, and if so, what accuracy is achievable with different input representations and architectures.

RQ2: Do physics-based input features improve generalization compared to primitive variables? If networks can learn equally well from primitives and from physics features, the additional effort of computing physics-based inputs may not be justified. Conversely, if physics features enable better generalization, they provide a concrete prescription for feature engineering in wall function applications.

RQ3: Do neural networks discover physics-based features spontaneously when trained on primitives? This question probes the relationship between Methods A and B. If networks discover physics features internally, it validates the importance of these features while suggesting that explicit encoding may not be strictly necessary.

RQ4: Does physics-constrained training improve extrapolation to conditions outside the training distribution? Method C sacrifices some in-distribution accuracy for physical consistency. The critical question is whether this trade-off pays off when models encounter flows not represented in training data.

RQ5: Can ML wall functions be deployed in production CFD codes and outperform classical approaches? Academic demonstrations on test sets must ultimately translate to practical utility. This question addresses the engineering viability of the proposed approaches.

1.7 Research Objectives

This thesis pursues four interconnected objectives that structure the experimental chapters:

Objective 1: Develop a systematic methodology for generating training data. High-quality training data is the foundation of any machine learning approach. We develop a dual-

mesh methodology that pairs fine-mesh (wall-resolved) simulations with coarse-mesh simulations of the same flow. The fine mesh provides ground-truth wall shear stress and heat flux; the coarse mesh provides the input features that a deployed model would have access to. This approach generates naturally-paired training data across 244 configurations spanning attached and separated flows.

Objective 2: Design physics-informed input features. Classical wall functions succeed because they use the right non-dimensional variables: y^+, u^+, and similar quantities that collapse boundary layer profiles across Reynolds numbers. We develop a library of 58 physics-based features derived from the near-wall stencil data, encoding quantities like local velocity gradients, pressure gradients, curvature terms, and thermal variables in wall-scaled and non-dimensional forms. These features provide the neural network with inputs that respect the scaling laws of turbulent boundary layers.

Objective 3: Investigate physics-guided architectures and training. Beyond input features, physical knowledge can be encoded in the network architecture and training procedure. We investigate whether neural networks spontaneously discover physics-based features in their hidden layers, and we develop physics-constrained loss functions that penalize violations of conservation laws during training.

Objective 4: Integrate trained models into production CFD solvers. Academic machine learning research often stops at demonstrating accuracy on test sets. We complete the engineering loop by implementing trained models as boundary conditions in OpenFOAM, validating performance on benchmark cases beyond the training distribution, and comparing against classical wall functions in both two-dimensional and three-dimensional configurations.

1.8 Thesis Contributions

This thesis makes the following contributions to the field of machine learning for computational fluid dynamics:

1. **A dual-mesh training data generation methodology** that systematically produces paired input-output data for wall function learning across diverse flow conditions. The resulting

dataset of 25,485 training samples spans Reynolds numbers 8,000–24,000 and geometric configurations from mild acceleration to strong separation.

2. **A physics-based feature library** comprising 58 non-dimensional quantities derived from the near-wall stencil. Systematic experiments demonstrate that a core subset of 11 physics features achieves accuracy comparable to much larger primitive-variable input sets, while providing interpretability advantages.

3. **Evidence for architecture-invariant feature discovery**, showing that neural networks trained on primitive inputs spontaneously develop hidden layer neurons correlated with physics-based quantities like y^+, $\ln(y^+)$, and pressure gradient scalings. This finding validates the physics-based input design and suggests that certain features emerge naturally from the learning problem.

4. **A local stencil-based Physics-Informed Neural Network** that enforces conservation laws (momentum, energy, mass) as soft constraints during training. Unlike global PINNs that require automatic differentiation over entire domains, this approach evaluates physics residuals on the same local stencil used for predictions, maintaining computational efficiency while improving physical consistency.

5. **A complete OpenFOAM integration** including C++ boundary condition implementation, Python-to-C++ model export infrastructure, and validation on benchmark cases including backward-facing steps and periodic hills.

1.9 Scope and Limitations

To maintain focus and depth, this thesis adopts specific boundaries on the problems addressed:

Flow regimes. We consider incompressible, low-Mach-number flows with Reynolds numbers in the range 8,000–24,000. While the methodology extends in principle to compressible flows and higher Reynolds numbers, validation is limited to the incompressible regime.

Turbulence modeling. All simulations use Reynolds-Averaged Navier-Stokes (RANS) equations with the k-ω SST turbulence model. The wall functions developed here could be adapted for Large Eddy Simulation, but LES-specific considerations (subgrid scale modeling

near walls) are not addressed.

Geometry. Training data comes from two-dimensional channel, diffuser, and nozzle configurations. Validation includes two-dimensional benchmark cases (backward-facing step, periodic hills) and preliminary three-dimensional results, but comprehensive three-dimensional validation remains future work.

Thermal coupling. We consider passive scalar transport with fixed wall temperatures. Conjugate heat transfer, radiative effects, and strongly-coupled thermal-fluid interactions are outside the scope.

Transient effects. All simulations assume steady-state conditions. Extension to unsteady flows, where temporal history effects may be important, is identified as a direction for future research.

These limitations define the domain within which the thesis results are validated. Extensions beyond these boundaries are discussed in Chapter 10 as opportunities for future work.

1.10 Thesis Outline

This thesis is organized as follows:

Chapter 2: Literature Review traces the evolution of wall function modeling from Prandtl's boundary layer theory through modern machine learning approaches. The chapter situates the present work within this intellectual history, identifying the specific gaps that the thesis addresses.

Chapter 3: Methodology and Structured Data Generation describes the computational setup, including the dual-mesh approach for generating paired training data, the OpenFOAM simulation configurations, and the post-processing pipeline that constructs stencil-based inputs from raw simulation outputs.

Chapter 4: Data-Driven Velocity and Thermal Wall Functions presents the baseline machine learning approach: fully-connected neural networks trained on primitive flow variables to predict wall shear stress and heat flux. This chapter establishes the achievable accuracy and

identifies limitations that motivate the physics-informed extensions.

Chapter 5: Non-Dimensional Feature Library develops the library of 58 physics-based input features and systematically compares performance against primitive-variable inputs (Method A). The chapter identifies a core set of features that balance accuracy, interpretability, and computational cost.

Chapter 6: Physics-Guided Hidden Layers investigates what neural networks learn internally when trained on primitive inputs (Method B). Through correlation analysis between hidden layer activations and physics-based features, the chapter demonstrates that networks discover wall-scaled quantities even without explicit physics encoding.

Chapter 7: Physics-Constrained Learning develops and evaluates the local stencil-based PINN approach (Method C), incorporating conservation law residuals into the training loss function. The chapter characterizes the trade-off between fitting accuracy and physical consistency.

Chapter 8: Identification of Flow Separation addresses the distinct challenge of detecting separated regions, where classical wall functions fail most severely. The chapter develops classifiers that distinguish attached from separated flow based on the same stencil inputs.

Chapter 9: OpenFOAM Integration and Validation presents the production implementation, including C++ boundary condition code, model export utilities, and comprehensive validation on benchmark geometries beyond the training distribution.

Chapter 10: Conclusion and Future Work summarizes the contributions, discusses limitations, and outlines directions for future research including extension to higher Reynolds numbers, three-dimensional flows, and unsteady applications.

1.11 Chapter Summary

Turbulent flow simulation requires accurate treatment of the near-wall region where steep gradients develop. Classical wall functions, while computationally efficient, fail under conditions of adverse pressure gradients, flow separation, and strong thermal coupling—precisely the conditions of greatest engineering interest. The economic implications span industries from aerospace

to building ventilation, motivating sustained research into improved wall function models.

Machine learning offers the potential to learn wall function models directly from high-fidelity data, but purely data-driven approaches risk learning spurious correlations that fail outside the training distribution. This thesis develops a physics-informed machine learning framework that combines the flexibility of neural networks with the constraints of physical law. Three complementary methods encode physics knowledge at different stages: in the input features (Method A), in the network architecture through implicit discovery (Method B), and in the training loss function (Method C).

The resulting models are integrated into OpenFOAM for practical CFD applications, demonstrating improved performance over classical approaches across a range of benchmark configurations. The following chapters develop this framework systematically, from data generation through production deployment, contributing both methodological advances and practical tools to the field of machine learning for computational fluid dynamics.

CHAPTER 2

Literature Review

This chapter traces the evolution of wall function modeling from its classical foundations through the modern era of machine learning, situating the present thesis within this historical trajectory. Rather than cataloging papers in isolation, we follow the intellectual thread that connects Prandtl's boundary layer theory to contemporary physics-informed neural networks, revealing how each generation of researchers built upon—and was constrained by—the work of their predecessors. The chapter begins with the fundamental physics of turbulent boundary layers, develops the mathematical framework underlying all wall function approaches, examines how industry has implemented these ideas in practical computational fluid dynamics codes, and finally surveys the machine learning revolution that motivates this thesis.

2.1 The Physics of Turbulent Boundary Layers

Understanding wall functions requires deep familiarity with the physics they attempt to model. The near-wall region of turbulent flows exhibits complex multi-scale phenomena that have occupied fluid dynamicists for over a century [1, 2]. This section develops the theoretical foundations from Prandtl's original insights through the modern understanding of near-wall turbulence structure, establishing the physical basis upon which all subsequent wall modeling efforts rest.

2.1.1 Prandtl's Boundary Layer Concept

The modern understanding of wall-bounded flows began with Ludwig Prandtl's seminal 1904 paper, which resolved a fundamental paradox that had troubled fluid mechanics for decades [25]. D'Alembert's paradox demonstrated that potential flow theory predicted zero drag on bodies

moving through inviscid fluids—a prediction manifestly contradicted by everyday experience. Prandtl's profound insight was that viscous effects, while negligible over most of the flow domain, become dominant in a thin layer adjacent to solid surfaces where the no-slip condition must be satisfied. Within this boundary layer, the velocity changes rapidly from zero at the wall to the free-stream value over a distance that may be orders of magnitude smaller than the characteristic length of the body.

This scale separation enables systematic simplification of the Navier-Stokes equations [43, 44]. For steady, two-dimensional flow over a flat plate aligned with the streamwise direction, the boundary layer equations reduce to a coupled system where the streamwise momentum balance involves convection by both velocity components, the imposed pressure gradient from the external flow, and viscous diffusion in the wall-normal direction only. The key simplification arises from recognizing that wall-normal gradients are much larger than streamwise gradients when the boundary layer thickness is small compared to the streamwise development length. This enables neglecting the streamwise viscous diffusion term while retaining the dominant wall-normal diffusion, yielding equations that are parabolic in the streamwise direction and can be marched forward from initial conditions without the elliptic complications of the full Navier-Stokes system.

The boundary layer equations take the form

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2} \tag{2.1}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.2}$$

where the pressure gradient is imposed by the external inviscid flow and varies along the surface. For a flat plate at zero incidence with uniform free-stream velocity, this gradient vanishes, yielding the canonical zero-pressure-gradient boundary layer that serves as the foundation for wall function development [29, 45]. Non-zero pressure gradients arise when the external flow accelerates or decelerates, profoundly affecting boundary layer behavior in ways that remain challenging to model even today [10, 34].

2.1.2 The Blasius Solution and Laminar Boundary Layers

For the zero-pressure-gradient laminar boundary layer, Blasius obtained a similarity solution in 1908 that remains a cornerstone of boundary layer theory [28]. By introducing a stream function and recognizing that the velocity profile maintains a self-similar shape when expressed in terms of an appropriately scaled wall-normal coordinate, Blasius reduced the partial differential equations to an ordinary differential equation. The similarity variable combines the wall distance with the local Reynolds number based on streamwise position, and the resulting third-order nonlinear ordinary differential equation, while not admitting closed-form solution, can be solved numerically to arbitrary precision [29].

The Blasius solution yields several important results that inform subsequent turbulent boundary layer analysis. The wall shear stress decreases along the plate as the inverse square root of downstream distance, giving a skin friction coefficient that scales as the inverse square root of the local Reynolds number. The boundary layer thickness grows as the square root of streamwise distance, a consequence of the diffusive spreading of vorticity from the wall. These laminar scaling relationships differ fundamentally from turbulent boundary layers, where enhanced mixing by turbulent fluctuations dramatically increases momentum transfer toward the wall [26, 27].

2.1.3 Transition to Turbulence

Laminar boundary layers become unstable at sufficiently high Reynolds numbers through mechanisms first elucidated by Tollmien and Schlichting in the 1930s [20, 46]. Linear stability analysis identifies critical conditions beyond which small disturbances amplify exponentially rather than decaying. These Tollmien-Schlichting waves, essentially two-dimensional oscillations of the boundary layer, grow as they convect downstream, interact nonlinearly, and eventually trigger breakdown to fully turbulent flow through a complex cascade involving three-dimensional secondary instabilities, spike formation, and the generation of turbulent spots that spread laterally and merge.

In practical applications, transition occurs over a range of Reynolds numbers depending on the disturbance environment [24, 47]. Free-stream turbulence intensity, surface roughness,

acoustic disturbances, and pressure gradients all influence the transition process. For smooth flat plates with low free-stream turbulence, transition typically begins around Reynolds numbers of several hundred thousand based on streamwise distance and completes within a factor of ten in Reynolds number. Higher disturbance levels trigger bypass transition at substantially lower Reynolds numbers through mechanisms that circumvent the classical instability route entirely [1]. The transitional region presents particular challenges for wall function modeling because neither laminar nor fully turbulent assumptions apply, and the flow exhibits intermittent turbulent spots within a laminar background. This thesis focuses on fully turbulent boundary layers, though the machine learning methods developed could potentially extend to transitional flows with appropriate training data.

2.1.4 Reynolds Decomposition and the Closure Problem

The irregular, chaotic nature of turbulent flows motivated Reynolds to propose decomposing instantaneous velocities into mean and fluctuating components [4]. Time-averaging the Navier-Stokes equations with this decomposition yields the Reynolds-Averaged Navier-Stokes equations, which govern the evolution of mean quantities [2, 5]. However, the averaging process introduces additional terms—the Reynolds stresses—representing momentum flux due to turbulent fluctuations. These six independent stress components appear as unknowns in the mean momentum equations, yet no additional equations emerge from the averaging process. This closure problem means the Reynolds-averaged equations cannot be solved without supplementary information relating Reynolds stresses to mean flow quantities [48, 49].

The most common closure approach invokes the Boussinesq hypothesis, which assumes Reynolds stresses are proportional to mean strain rates through a turbulent eddy viscosity, analogous to the relationship between viscous stress and strain rate in Newtonian fluids [1, 15]. This reduces the closure problem to determining a single scalar field, the eddy viscosity, which turbulence models provide through additional transport equations for quantities like turbulent kinetic energy and its dissipation rate. The Boussinesq hypothesis assumes Reynolds stresses respond instantaneously to local strain rates, ignoring history effects and the inherent anisotropy of turbulence [50, 51]. These limitations become severe in flows with strong stream-

line curvature, rapid strain, or separation—precisely the conditions where wall functions also fail, motivating the more sophisticated approaches developed in this thesis.

2.1.5 The Structure of Turbulent Boundary Layers

Turbulent boundary layers exhibit a characteristic layered structure that emerges from the competition between viscous and turbulent stresses at different distances from the wall [11, 26]. This structure, revealed through decades of experimental investigation and more recently through direct numerical simulation, provides the physical basis for wall function modeling. Near the wall, viscous stresses dominate because turbulent fluctuations are damped by the no-slip condition. Far from the wall, turbulent stresses dominate as energetic eddies transport momentum efficiently. Between these limiting behaviors lies a transitional region where both mechanisms contribute comparably [20, 27].

The relevant length scale near the wall is the viscous length, defined as the ratio of kinematic viscosity to friction velocity. Normalizing wall distance by this viscous length defines the wall unit, typically denoted y^+. Similarly normalizing the mean velocity by friction velocity defines u^+. Within the inner layer, comprising roughly the innermost ten to twenty percent of the boundary layer thickness, the velocity profile depends only on these wall-scaled variables and is independent of the outer flow conditions [3, 30]. This universality, known as the law of the wall, represents one of the most robust results in turbulence and provides the theoretical foundation for wall function approaches.

The inner layer subdivides into three distinct regions with different physical character [22, 52]. Immediately adjacent to the wall, in what is termed the viscous sublayer, turbulent fluctuations are strongly suppressed and viscous stress dominates completely. The velocity profile becomes linear, with $u^+ = y^+$, a result confirmed by countless experiments and simulations to remarkable precision. This linear region extends to roughly $y^+ = 5$, beyond which turbulent stresses begin contributing significantly.

Between the viscous sublayer and the fully turbulent region lies the buffer layer, extending from approximately $y^+ = 5$ to $y^+ = 30$ [26, 46]. In this transitional zone, both viscous and turbulent stresses contribute comparably to momentum transfer, and the velocity profile curves

smoothly between the linear viscous behavior and the logarithmic profile that emerges at larger wall distances. The buffer layer contains the peak of turbulent kinetic energy production and hosts the most intense turbulent events, including the ejection of low-speed fluid away from the wall and the sweep of high-speed fluid toward it. No simple analytical expression describes this region accurately, though various empirical formulas have been proposed for engineering calculations [7].

Beyond the buffer layer, for $y^+ > 30$ but still within roughly twenty percent of the boundary layer thickness, turbulent stresses dominate completely and a logarithmic velocity profile emerges. This logarithmic law of the wall, first proposed by von Kármán based on dimensional arguments [9], takes the form $u^+ = (1/\kappa)\ln(y^+) + B$, where $\kappa \approx 0.41$ is the von Kármán constant and $B \approx 5.0$ is an additive constant determined by matching through the buffer layer. The logarithmic profile can be derived from several independent lines of reasoning, lending confidence to its universality [29].

2.1.6 Derivations of the Logarithmic Law

Dimensional analysis provides the most direct route to the logarithmic profile [43, 45]. In the fully turbulent region where viscosity no longer directly influences the local dynamics, the velocity gradient can depend only on wall distance, wall shear stress, and fluid density. Dimensional consistency then requires the velocity gradient to scale inversely with wall distance, and integration yields the logarithmic profile with an undetermined multiplicative constant that experiments identify as the inverse of the von Kármán constant.

Prandtl's mixing length hypothesis offers a physical interpretation of this scaling [17, 53]. Turbulent eddies transport momentum over a characteristic mixing length before losing their identity through mixing with surrounding fluid. Near the wall, the largest eddies that can exist are constrained by the wall distance itself, suggesting the mixing length should be proportional to y. This assumption, combined with the definition of turbulent stress through the mixing length formulation, yields an eddy viscosity proportional to wall distance times the velocity gradient. When the total stress is constant and equal to the wall shear stress—a good approximation in the inner layer—solving for the velocity gradient and integrating reproduces

the logarithmic law.

Millikan's overlap argument provides yet another derivation without assuming anything about the turbulent stress mechanism [54]. In the inner layer, the velocity profile must depend only on wall-scaled variables. In the outer layer, the velocity defect from the free-stream value must depend only on outer-scaled variables involving the boundary layer thickness. In the overlap region where both descriptions must simultaneously apply, the only functional form consistent with both scalings is logarithmic. This elegant argument demonstrates that the log law is not merely an empirical fit but a mathematical consequence of the two-layer structure of the boundary layer.

2.1.7 The Outer Layer and Wake Function

Beyond the logarithmic region, the velocity continues increasing toward the free-stream value through what is termed the wake region or defect layer [11, 33]. Here the velocity defect from the free-stream value scales with the friction velocity and the boundary layer thickness rather than with viscous quantities. Coles proposed combining the inner logarithmic profile with an additive wake function to describe the entire boundary layer. The wake function accounts for the departure from the log law in the outer region due to intermittency effects at the boundary layer edge and the influence of the outer boundary conditions.

The wake parameter quantifies the strength of the wake component and varies with pressure gradient [31, 34]. For zero-pressure-gradient boundary layers, the wake parameter takes a value around 0.55, corresponding to a modest overshoot above the logarithmic extrapolation in the outer region. Adverse pressure gradients increase the wake parameter, strengthening the deviation from the log law, while favorable pressure gradients decrease it. This pressure-gradient sensitivity of the outer layer contrasts with the relative universality of the inner layer and illustrates why wall functions based solely on the log law struggle under non-equilibrium conditions.

2.1.8 Turbulent Stress Distributions

The Reynolds stress tensor exhibits distinct behavior across the boundary layer structure, behavior that must be captured or circumvented by wall function approaches [3, 30]. The landmark direct numerical simulation of Moser, Kim, and Mansour provided definitive data on these distributions in turbulent channel flow, later extended to higher Reynolds numbers by Lee and Moser [20, 26].

The streamwise velocity fluctuations peak in the buffer layer around $y^+ \approx 15$, reaching root-mean-square values roughly 2.5 to 3 times the friction velocity [11, 53]. This intense streamwise fluctuation activity reflects the burst-sweep cycle that dominates near-wall turbulence, with low-speed fluid ejected away from the wall and high-speed fluid swept toward it. The wall-normal fluctuations, in contrast, are strongly suppressed near the wall by the kinematic blocking effect of the surface and increase more gradually, peaking in the outer part of the logarithmic layer. The Reynolds shear stress, the only component contributing to mean momentum transfer in parallel shear flows, varies nearly linearly across the inner layer from zero at the wall to a maximum in the outer layer, reflecting the handoff from viscous to turbulent momentum transport.

Understanding these stress distributions is essential for wall function development [22, 52]. Standard wall functions assume local equilibrium between production and dissipation of turbulent kinetic energy, which holds approximately in the logarithmic layer where the production rate nearly balances the dissipation rate. In the buffer layer, production exceeds dissipation as turbulent kinetic energy is generated before being transported and dissipated elsewhere. Under non-equilibrium conditions such as adverse pressure gradients or flow separation, the equilibrium assumption fails throughout the boundary layer, invalidating the fundamental premise of classical wall functions [38, 50].

2.2 The Mathematical Framework of Wall Functions

The theoretical understanding developed above must be translated into practical computational tools. This section traces the evolution of wall function formulations from early analytical

approaches through modern enhanced treatments implemented in commercial software [6, 7].

2.2.1 The Launder-Spalding Wall Function

Launder and Spalding formalized the use of the logarithmic law as a boundary condition for Reynolds-averaged simulations in their influential 1974 paper [6]. Their approach places the first computational cell center in the logarithmic region, typically at y^+ values between 30 and 300, and relates the cell-center velocity to wall shear stress algebraically through the log law. This eliminates the need to resolve the steep gradients in the viscous sublayer and buffer layer, reducing mesh requirements by an order of magnitude or more while maintaining accuracy for flows where the logarithmic profile applies [23, 24].

The implementation requires care because the wall shear stress appears on both sides of the log law equation—explicitly in the relationship and implicitly through the wall-scaled variables [55]. Launder and Spalding resolved this by using the cell-center turbulent kinetic energy rather than the friction velocity to define the wall units, yielding an explicit formula for wall shear stress given the cell-center velocity and turbulence quantities. The turbulence boundary conditions similarly employ equilibrium assumptions, specifying that the dissipation rate in the wall-adjacent cell equals the production rate according to local equilibrium.

The thermal wall function follows analogously, using Reynolds analogy to relate the turbulent heat flux to the momentum flux through a turbulent Prandtl number [21, 22]. The temperature profile in wall units exhibits a linear conduction-dominated region near the wall analogous to the viscous sublayer, followed by a logarithmic region where turbulent transport dominates. The transition between these regions depends on the molecular Prandtl number, with high-Prandtl-number fluids having extremely thin conduction sublayers and low-Prandtl-number fluids like liquid metals having conduction extending well into the turbulent region.

2.2.2 Assumptions and Their Limitations

The Launder-Spalding wall function rests on several assumptions whose violation causes the approach to fail [38, 51]. The assumption that the first cell lies within the logarithmic region means that if the mesh is too coarse the cell extends into the wake region where the log law

overpredicts velocity, and if too fine the cell falls into the buffer layer where it underpredicts. The local equilibrium assumption means the approach cannot capture the history effects that characterize boundary layers under pressure gradients or after perturbations. The constant-stress assumption requires the wall-adjacent cell to be thin enough that stress variation across it is negligible.

The most severe failures occur in separated flows where the wall shear stress becomes negative [11, 23, 34]. The friction velocity, defined as the square root of the wall shear stress magnitude divided by density, remains real, but the entire scaling framework loses physical meaning when the near-wall velocity opposes the outer flow direction. The log law, derived for attached flows where velocity increases monotonically from wall to free-stream, has no relevance to the reversed-flow profiles near separation. Most computational codes handle separated regions by ad-hoc fixes such as limiting the wall shear stress magnitude or switching to alternative formulations, but these patches have no physical basis and typically produce poor predictions of separation location, recirculation zone extent, and reattachment [10, 35].

Strong adverse pressure gradients, even without separation, cause the equilibrium assumption to fail [15, 31]. As the flow decelerates, the boundary layer thickens, the velocity profile distorts from logarithmic form, and the production of turbulent kinetic energy exceeds its dissipation rate. The classical wall function, seeing only the cell-center conditions and knowing nothing of the upstream history, overpredicts wall shear stress because it assumes an equilibrium profile that the actual flow does not possess.

2.2.3 Enhanced Wall Treatments

Recognition of these limitations motivated development of enhanced wall treatments that attempt to function across a wider range of conditions [8, 23]. Two-layer models resolve the viscous sublayer with a fine mesh while using a simplified turbulence model in the near-wall region. These models can handle lower wall-unit mesh spacing but still assume quasi-equilibrium and struggle with separated flows.

Enhanced wall functions blend between viscous sublayer and logarithmic formulations based on local wall-unit spacing, allowing the first cell to be placed anywhere from the sublayer to the

log layer without dramatic loss of accuracy [7, 55]. Spalding's unified wall law provides a single implicit equation valid across the entire inner layer, though it cannot be solved explicitly for velocity given wall distance. Modern commercial codes implement various blending functions that transition smoothly between the linear and logarithmic regimes.

Scalable wall functions address the problem of meshes refined beyond the buffer layer by defining a virtual wall location at the intersection of the linear and logarithmic profiles [24]. This prevents the wall function from predicting unphysically low wall shear stress on fine meshes, but it sacrifices the potential accuracy gains that fine mesh resolution near the wall might provide.

Non-equilibrium wall functions attempt to account for pressure gradient effects by modifying the log-law constants or by solving simplified boundary layer equations within the wall function formulation [10, 38]. The Launder-Shima approach adds a pressure gradient correction to the log-law intercept, partially compensating for the profile distortion under adverse gradients. More sophisticated analytical wall functions solve the log-layer momentum equation including convection and pressure gradient effects, but even these struggle with the extreme non-equilibrium of separated flows.

2.3 Thermal Boundary Layers and Heat Transfer

Wall functions must predict not only momentum transfer through wall shear stress but also heat transfer through wall heat flux for thermal applications [21, 22, 42]. The thermal boundary layer exhibits analogous layered structure to the velocity boundary layer but with important differences that complicate the extension of momentum wall functions to heat transfer.

2.3.1 The Reynolds Analogy

Reynolds observed in 1874 that the mechanisms of momentum and heat transfer in turbulent flow share fundamental similarities, leading to the classical Reynolds analogy relating skin friction coefficient to Stanton number [21, 40]. For fluids with Prandtl number equal to unity, where molecular diffusivities of momentum and heat are identical, the analogy predicts exact equality between half the skin friction coefficient and the Stanton number. For fluids

with Prandtl numbers differing from unity, modifications account for the different sublayer thicknesses, with the Colburn analogy introducing a Prandtl number correction factor.

These analogies work reasonably well for attached boundary layers with moderate temperature differences but fail when pressure gradients cause the velocity and thermal boundary layers to develop differently [22, 42], when strong temperature differences cause property variations that invalidate the constant-property assumptions, or when buoyancy effects couple the momentum and energy equations. The analogy has no physical basis whatsoever in separated regions where the reversed near-wall flow bears no relationship to the heat transfer at the wall [41, 56].

2.3.2 Thermal Law of the Wall

By analogy with the velocity profile, the temperature field in the inner layer follows a universal law when expressed in appropriate wall units [21, 39]. Defining a friction temperature through the wall heat flux, analogous to the friction velocity definition, allows expressing the temperature difference from the wall in dimensionless form. In the conduction-dominated sublayer immediately adjacent to the wall, the temperature profile becomes linear with slope equal to the molecular Prandtl number. In the turbulent region, a logarithmic profile emerges with slope related to the turbulent Prandtl number.

Kader developed a widely-used formula that blends smoothly between these regimes while capturing the Prandtl number dependence of the thermal buffer layer [39]. The blending function differs from that for velocity because the thermal and momentum buffer layers have different structures depending on the molecular Prandtl number [41]. For high-Prandtl-number fluids like oils, the thermal sublayer is extremely thin compared to the viscous sublayer, while for low-Prandtl-number fluids like liquid metals, conduction penetrates far into the turbulent region.

2.3.3 Dissimilarity Between Momentum and Heat Transfer

A fundamental limitation of Reynolds-analogy-based thermal wall functions is the assumption that momentum and heat transfer follow identical mechanisms [40, 42]. Several sources of dissimilarity exist even in attached boundary layers. The turbulent Prandtl number, relating

turbulent momentum and heat diffusivities, is not constant but varies from roughly unity in the sublayer to 0.85 in the logarithmic layer to perhaps 0.5 in the outer region. Pressure gradients affect velocity and temperature profiles differently, and buoyancy introduces coupling between the fields that the analogy cannot capture [41].

These considerations motivate the approach of this thesis, which predicts wall shear stress and wall heat flux simultaneously from unified models trained on data that naturally captures the dissimilarity between momentum and heat transfer [21, 22]. Rather than assuming the two are linked by a constant factor, the neural network learns whatever relationship the training data exhibits.

2.4 Industrial Computational Fluid Dynamics Practice

The theoretical developments traced above must ultimately serve practical engineering analysis [17, 18]. This section examines how wall functions are implemented in commercial codes and applied in industrial workflows, identifying the gap between academic research and production practice that motivates practical deployability as a key objective of this thesis.

2.4.1 Commercial Implementations

Major commercial computational fluid dynamics codes implement wall functions with varying degrees of sophistication [23, 24]. ANSYS Fluent offers multiple options including standard wall functions, scalable wall functions, enhanced wall treatment, and non-equilibrium wall functions. The enhanced wall treatment automatically blends between viscous sublayer resolution and log-law formulations based on local mesh resolution, providing flexibility at the cost of potential inconsistency between regions. STAR-CCM+ implements all-y^+ wall treatment that switches between low-Reynolds and high-Reynolds formulations automatically, emphasizing robustness for industrial users who may not optimize mesh resolution for each application.

OpenFOAM, the open-source platform used throughout this thesis, provides wall function boundary conditions implementing the Launder-Spalding formulation along with automatic wall treatment options using Spalding's unified law [24, 57]. The open-source nature allows direct implementation of new wall function approaches, making it the natural choice for devel-

oping and validating the machine learning methods of this thesis.

2.4.2 Industrial Practice and Certification

Industrial computational fluid dynamics practice involves compromises between accuracy, computational cost, and robustness [17, 58]. Mesh guidelines typically recommend wall-unit spacing in the range 30 to 300 for standard wall functions and near unity for wall-resolved simulations, but achieving these targets uniformly across complex geometries is often impractical. Real industrial meshes frequently have non-uniform wall spacing that places some regions in the buffer layer where neither wall function formulation is optimal.

High-stakes industries impose stringent validation requirements that often exceed wall function capabilities [10, 41]. Aerospace certification may require wall-resolved simulations for flight-critical components because wall function predictions are deemed insufficiently reliable. Nuclear safety analysis requires validated methodologies with quantified uncertainty margins that wall function limitations can expand substantially [59–61]. These requirements highlight the gap between wall function capabilities and industrial needs that motivates continued research including the machine learning approaches developed here.

2.4.3 The Academia-Industry Gap

A persistent gap exists between academic turbulence modeling research and industrial practice [1, 58]. Academic developments often include many adjustable parameters tuned for specific test cases, whereas industrial users need robust defaults that work across diverse applications without case-by-case adjustment. Academic publications report successful cases, but industrial simulations must handle arbitrary geometries and flow conditions without divergence or unphysical results [15, 51]. Advanced models that improve accuracy modestly while substantially increasing computational cost are rarely adopted industrially, where throughput considerations often override accuracy concerns.

This thesis addresses several of these gaps. By implementing models in OpenFOAM and providing complete documentation, we lower barriers to industrial adoption [24, 57]. By training on diverse configurations and validating on geometries outside the training distribution,

we demonstrate the robustness industrial users require. By maintaining computational efficiency comparable to standard wall functions, we avoid the cost barriers that have limited adoption of more sophisticated approaches.

2.5 Pressure Gradient Effects

Pressure gradients profoundly affect turbulent boundary layer behavior and represent a key challenge for wall function modeling [10, 31, 34]. The Clauser parameter, defined as the product of displacement thickness and pressure gradient divided by wall shear stress, quantifies pressure gradient strength relative to wall friction. Zero values correspond to zero-pressure-gradient boundary layers, negative values to favorable (accelerating) gradients, and positive values to adverse (decelerating) gradients. Self-similar equilibrium boundary layers can exist at constant Clauser parameter values, maintaining fixed shape while growing downstream, but most practical flows involve spatially-varying pressure gradients that prevent self-similarity [11, 38].

Under adverse pressure gradients, the boundary layer thickens as retarded near-wall fluid decelerates further, the velocity profile distorts from logarithmic form with increased wake component, turbulence intensifies due to enhanced production, and wall shear stress decreases toward zero [10, 35]. If the adverse gradient is sufficiently strong, flow separation occurs when wall shear stress vanishes and the near-wall fluid reverses direction. The approach to separation involves highly non-equilibrium dynamics that classical wall functions cannot capture.

Stratford developed a criterion for predicting turbulent boundary layer separation based on the pressure rise and streamwise development length [32]. While useful for preliminary design, such criteria cannot predict the detailed behavior through and after separation [11, 37]. The machine learning approach of this thesis provides predictions throughout the separation and reattachment process by learning from training data that includes these regimes.

2.6 The Machine Learning Revolution

Having established the classical physics and computational frameworks, we turn to the modern machine learning approaches that motivate this thesis. The emergence of data-driven turbulence

modeling represents not merely an incremental improvement but a fundamental shift in how the field approaches the closure problem and the limitations of classical models [1, 2, 5].

2.6.1 Data-Driven Turbulence Modeling

The modern era of machine learning in computational fluid dynamics emerged from advances in deep learning for image recognition and natural language processing that demonstrated neural networks could learn complex nonlinear mappings from large datasets [5, 13, 48]. The seminal work of Ling, Kurzawski, and Templeton marked a watershed moment by demonstrating that deep neural networks could predict Reynolds stress anisotropy from mean flow features, capturing physics beyond the Boussinesq hypothesis that had constrained turbulence modeling for decades.

Ling's approach incorporated physics awareness through its input construction [49, 50]. Rather than providing raw flow quantities, she used invariants of the strain rate and rotation tensors as inputs, encoding Galilean invariance—the requirement that turbulence statistics cannot depend on observer frame of reference—directly into the network architecture. The outputs similarly respected tensorial structure through a basis expansion. This physics-informed approach yielded better generalization than purely black-box models, establishing a paradigm that subsequent work has extended [2, 51].

Wang, Wu, and Xiao developed comprehensive frameworks combining machine learning with uncertainty quantification, emphasizing that neural networks trained on high-fidelity data encode implicit knowledge of turbulent dynamics but that this knowledge may not transfer reliably to flows outside the training distribution [5, 15, 58]. The generalization problem—ensuring models trained on canonical configurations perform well in practical applications—emerged as the central challenge for the field and remains largely unsolved [38, 50, 51].

Subsequent research has explored numerous variations on data-driven turbulence modeling [1, 2]. Sparse symbolic regression discovers algebraic Reynolds stress models automatically from data [37, 49]. Multi-agent reinforcement learning optimizes turbulence model parameters through automated experimentation [48]. Ensemble methods quantify uncertainty in learned closures [59, 60, 62]. Each approach offers unique advantages, but the fundamental challenge

of generalizing from training data to novel flows persists across methodologies [15, 38].

2.6.2 Physics-Informed Neural Networks

Raissi, Perdikaris, and Karniadakis introduced Physics-Informed Neural Networks (PINNs), addressing the fundamental limitation that traditional supervised learning requires abundant labeled training data [14, 43, 44]. Their approach incorporates governing equations as soft constraints during training by adding physics residuals to the loss function. If the network learns to satisfy conservation laws throughout the domain, its predictions should remain physically consistent even in regions without training data [29, 34, 57].

The physics loss evaluates partial differential equation residuals at collocation points using automatic differentiation through the network, penalizing departures from the governing equations [24, 43, 45]. For incompressible flows, this includes continuity and momentum residuals throughout the domain. The balance between data fitting and physics constraint satisfaction, governed by a hyperparameter weighting the physics loss, requires careful tuning that varies across problems [56, 63, 64].

Physics-informed networks have been applied to numerous fluid dynamics problems [24, 56, 65, 66]. Wang and colleagues applied PINNs to turbulence modeling with tensor basis constraints [5, 67]. Applications span biomedical flows [56, 66], combustion [63, 68], and free shear flows [24]. OpenFOAM integration has been demonstrated for practical CFD workflows [57].

Physics-informed networks face substantial challenges for turbulence applications [15, 63, 69]. The Reynolds-averaged equations are not closed, requiring models for Reynolds stresses that cannot be derived from first principles. Neural networks exhibit spectral bias toward learning low-frequency features before high-frequency ones, making multi-scale turbulence difficult to capture [10, 64]. The physics loss can create ill-conditioned optimization landscapes near boundaries where multiple constraints interact [44, 63].

Chapter 7 of this thesis develops a local stencil-based variant that addresses these limitations for wall function applications. Rather than enforcing global conservation over entire domains, we evaluate physics residuals on the same local stencil used for input features, maintaining

computational efficiency while focusing physical constraints where they matter for wall shear and heat flux prediction.

2.6.3 Deep Learning Architectures for Turbulence

Beyond physics-informed approaches, various deep learning architectures have been applied to turbulence problems [20, 26, 27, 70, 71]. Convolutional neural networks exploit spatial structure in flow fields [26, 53, 72–74], while recurrent networks capture temporal dependencies in unsteady flows [20, 41, 75, 76]. Attention mechanisms allow networks to focus on relevant regions [41, 46, 77, 78], and transformer architectures originally developed for natural language processing show promise for sequence modeling in turbulence [11, 53, 79, 80].

Graph neural networks offer particular promise for CFD applications due to their ability to handle unstructured meshes naturally [55, 81–84]. By representing mesh cells as nodes and connectivity as edges, graph networks can process arbitrary mesh topologies without the structured grid requirements of convolutional networks [85, 86]. This flexibility enables application to complex geometries without the mesh interpolation that convolutional approaches require [66, 81, 87, 88].

Super-resolution techniques reconstruct fine-scale turbulent structures from coarse representations [11, 53, 72, 89, 90]. Neural operators learn mappings between function spaces [66, 74, 91, 92], potentially enabling generalization across geometries and operating conditions [44, 93, 94]. Diffusion models generate realistic turbulent fields through iterative denoising [11, 95–97]. Each architecture class offers distinct advantages for specific aspects of the turbulence modeling challenge [98, 99].

2.6.4 Machine Learning Wall Functions

The specific application of machine learning to wall function modeling represents the confluence of classical wall-bounded turbulence theory, data-driven Reynolds stress modeling, and physics-informed learning [26, 38, 52, 100, 101]. Milano and Koumoutsakos pioneered this direction in 2002, training neural networks to predict wall shear stress from outer flow quantities for large eddy simulation [16, 102, 103]. Limited by contemporary computational resources, their

work demonstrated proof of concept but could not address generalization challenges [104, 105].

Recent work has demonstrated that neural networks can handle separated flows over periodic hills where standard wall functions fail entirely [11, 37, 38, 106, 107], that training on canonical building-block flows can construct composite models for more complex geometries [15, 51, 108, 109], and that physics-informed variants improve consistency at the cost of some fitting accuracy [63, 64, 110, 111]. Deep neural networks have been applied specifically to near-wall turbulence with consideration of wall-distance effects [26, 27, 38, 47, 112]. Data-driven approaches for separated flows have shown particular promise [10, 35, 37, 113, 114].

Yet critical questions remain [15, 51]. The choice between primitive variables and physics-based features as network inputs varies across studies without systematic justification—a gap Chapter 5 addresses. Thermal wall functions receive disproportionately little attention despite engineering importance [21, 22, 42]—a gap this thesis addresses by predicting both momentum and thermal quantities jointly. The generalization from training geometries to practical applications remains the central unsolved challenge [15, 38].

2.6.5 Uncertainty Quantification in Machine Learning CFD

Reliable deployment of machine learning models requires quantifying prediction uncertainty [59–62, 115, 116]. Bayesian approaches provide principled uncertainty estimates through posterior distributions over network weights [58, 117–119]. Deep ensembles use disagreement among independently trained networks as an uncertainty proxy [59, 62, 120, 121]. Dropout during inference approximates Bayesian uncertainty without the computational cost of full posterior inference [15, 122, 123].

Physics-constrained random forests provide uncertainty quantification with interpretable decision boundaries [62, 124, 125]. Mondrian forests offer calibrated uncertainty estimates for turbulence modeling [126–128]. Machine learning uncertainty has been applied to airfoil predictions [129–131], turbulence model selection [132–134], and Reynolds stress modeling [58, 61, 135, 136].

For wall functions specifically, uncertainty quantification enables appropriate selection between ML and traditional approaches based on prediction confidence [59, 60]. Regions where

the ML model is uncertain can fall back to validated classical methods, combining the accuracy of data-driven approaches where they are confident with the reliability of physics-based models elsewhere. This thesis does not develop uncertainty quantification methods but identifies this as an important direction for future work.

2.6.6 Transfer Learning and Domain Adaptation

A critical challenge for data-driven wall functions is transferring knowledge from training configurations to novel applications [15, 38, 51, 137, 138]. Transfer learning leverages features learned on one task to improve performance on related tasks [70, 95, 139, 140]. Domain adaptation addresses distribution shift between training and deployment conditions [10, 38, 141, 142].

For turbulence modeling, transfer learning has been applied to adapt models trained on canonical flows to complex geometries [15, 38, 143, 144]. The physics-based features developed in Chapter 5 inherently provide transfer learning benefits by encoding scale-invariant relationships that apply across Reynolds numbers and geometries [49, 50, 145, 146]. The architecture-invariant features identified in Chapter 6 suggest which representations transfer most reliably [147, 148].

2.7 Benchmark Cases and Validation

The development and validation of wall functions requires high-fidelity reference data spanning conditions from simple equilibrium to complex separation [3, 12, 30, 36, 149, 150]. The turbulence modeling community has established canonical benchmark cases that serve this purpose [151, 152].

Fully-developed channel flow between parallel plates represents the simplest configuration, with zero pressure gradient, statistical stationarity, and clean validation of the law of the wall [3, 26, 30]. The direct numerical simulation databases of Moser, Kim, and Mansour at friction Reynolds numbers up to 590 remain definitive references, later extended by Lee and Moser to friction Reynolds numbers exceeding 5000. These databases provide mean velocity profiles, Reynolds stress components, and turbulent kinetic energy budgets against which wall function predictions can be quantitatively assessed.

The zero-pressure-gradient flat plate boundary layer adds spatial development to the problem [19, 20, 29]. Direct numerical simulation data from Schlatter and Örlü provide validation through the transition and early turbulent regimes. Standard wall functions perform well for this canonical configuration, establishing baseline performance that more complex flows challenge.

Diffuser flows introduce adverse pressure gradients of controllable severity [10, 37, 38]. By varying expansion angle and area ratio, diffusers span conditions from mild deceleration with attached flow through incipient separation to massive recirculation. The asymmetric diffuser configuration used throughout this thesis provides systematic variation of pressure gradient while maintaining a flat wall for clean data extraction.

The backward-facing step represents an extreme test case where sudden expansion creates massive separation with reattachment several step heights downstream [11, 12, 35]. The experiments of Driver and Seegmiller provide detailed validation data. Standard wall functions fail catastrophically in the recirculation zone, making the backward-facing step a stringent test for any improved approach.

Periodic hills add surface curvature to the separation challenge [36–38]. The configuration of Breuer and colleagues provides cyclic separation and reattachment over curved surfaces with direct numerical simulation validation. The periodic boundary conditions eliminate inlet specification issues while creating a challenging test of wall function performance over curved separating surfaces.

Chapter 3 describes how training data from diffuser, nozzle, and channel configurations—244 cases spanning Reynolds numbers 8,000 to 24,000 and expansion ratios from 0.5 to 5.5—provides the foundation for the machine learning models developed in subsequent chapters. Chapter 9 returns to these benchmark cases for validation, testing model generalization from training geometries to backward-facing steps and periodic hills.

2.8 Related Applications and Extensions

Machine learning approaches to near-wall modeling connect to several related application domains that inform and are informed by wall function development [17, 18, 153, 154].

2.8.1 Large Eddy Simulation Subgrid Modeling

Large eddy simulation requires modeling the effect of unresolved subgrid scales on the resolved flow [11, 79, 102, 155, 156]. Machine learning approaches to subgrid modeling share many challenges with wall function development: both must represent unresolved physics through modeled terms that depend on resolved quantities [69, 157, 158]. Deep learning subgrid models have demonstrated improved accuracy over classical approaches [11, 102, 159], though stability in coupled simulations remains challenging [157, 160].

The wall model for LES (WMLES) approach combines features of wall functions and subgrid models [26, 38]. Machine learning wall models for LES must capture the effect of unresolved near-wall turbulence on the resolved outer flow, a more demanding task than RANS wall functions because the instantaneous fluctuations carry physical information that time-averaged quantities lack [26, 27].

2.8.2 Reduced-Order Modeling

Reduced-order models compress high-dimensional flow fields into low-dimensional representations suitable for rapid prediction [53, 95, 161–163]. Proper orthogonal decomposition identifies optimal bases for linear dimensionality reduction [53, 164, 165], while autoencoders learn nonlinear manifolds [11, 72, 166, 167]. These approaches complement wall functions by providing efficient full-field predictions that the wall function can then refine near boundaries [168].

Neural operators learn mappings between function spaces, enabling prediction of flow fields for new boundary conditions or parameters without retraining [44, 66, 74]. When combined with wall functions, neural operators could provide rapid predictions across families of geometries or operating conditions, with the wall function ensuring accurate near-wall behavior.

2.8.3 Multi-Fidelity and Multi-Scale Approaches

Many applications require predictions at multiple fidelity levels or across multiple scales [40, 41, 169, 170]. Multi-fidelity machine learning combines cheap low-fidelity data with expensive high-fidelity data to achieve accuracy at reduced cost [40, 171, 172]. For wall functions, this could mean training on abundant RANS data augmented by limited DNS or experimental data [41, 173, 174].

Multi-scale approaches explicitly model the interaction between resolved and unresolved scales [11, 53]. Super-resolution techniques reconstruct fine-scale structure from coarse representations [53, 72]. These ideas connect to wall function development by providing principled frameworks for coupling different resolution levels near boundaries.

2.9 Research Gaps and Thesis Contributions

This comprehensive literature review reveals specific gaps that the present thesis addresses through its five major contributions.

The first gap concerns systematic comparison of input representations [15, 51]. Despite extensive work on machine learning wall functions, no study has systematically compared primitive variables versus physics-based features under controlled conditions. Chapter 5 fills this gap through experiments comparing 11 core physics features against 58 comprehensive features and 90 primitive stencil variables, demonstrating that a compact physics-based representation achieves accuracy comparable to much larger primitive inputs while providing interpretability advantages.

The second gap concerns thermal predictions [21, 22, 42]. The literature focuses overwhelmingly on velocity wall functions, with thermal predictions receiving disproportionately little attention despite their engineering importance in applications from gas turbine cooling to building ventilation. All experimental chapters of this thesis predict both skin friction coefficient and Stanton number jointly, learning whatever dissimilarity exists between momentum and heat transfer from the training data rather than assuming Reynolds analogy.

The third gap concerns training data diversity [15, 38]. Many studies train on simple

geometries and hope for generalization to more complex configurations. Chapter 3 presents training data specifically designed to span attached to separated conditions through systematic variation of pressure gradients and geometric parameters across 244 configurations, providing comprehensive coverage of the conditions machine learning wall functions must handle.

The fourth gap concerns physics encoding in network architecture [5, 49]. Beyond input features and loss functions, the network architecture itself can encode physical knowledge. Chapter 6 explores whether neural networks trained on primitive inputs discover physics-based features in their hidden layers, finding architecture-invariant features that validate the physics-based input design and suggest certain relationships emerge naturally from the learning problem.

The fifth gap concerns practical deployment [24, 57]. Academic machine learning research rarely addresses implementation in production codes, limiting impact on engineering practice. Chapter 9 presents complete integration with OpenFOAM including boundary condition implementation, model export utilities, and validation on benchmark geometries outside the training distribution.

2.10 Chapter Summary

The trajectory from Prandtl's boundary layer theory through modern physics-informed neural networks spans over a century of scientific development. Each generation built upon predecessors: von Kármán's logarithmic profile upon Prandtl's thin-layer analysis, Launder and Spalding's wall functions upon von Kármán's scaling laws, and contemporary machine learning approaches upon both classical physics and the data-driven paradigm shift enabled by modern computing [6, 9, 14, 25].

This chapter has developed the theoretical foundations essential for understanding wall function modeling: the physics of turbulent boundary layers from viscous sublayer through logarithmic region to outer wake [3, 30, 33], the mathematical framework underlying classical wall functions and their enhanced variants [6–8], the additional complexity of thermal boundary layers and their dissimilarity from momentum transport [21, 39], the implementation of wall functions in industrial practice and the gaps between capabilities and requirements [17, 58], and the recent machine learning revolution that offers fundamentally new approaches to these

classical challenges [5, 14, 15, 38].

The present thesis continues this progression, combining physics-based feature engineering [49, 50], interpretable neural architectures [26, 46], physics-constrained training [63, 64], and practical deployment [57] into an integrated framework for machine learning wall functions. By addressing the gaps identified in this review, this work aims to advance the state of the art while honoring the intellectual heritage that makes such advances possible.

CHAPTER 3

Methodology and Structured Data Generation

This chapter presents the comprehensive methodology for generating structured training data for machine learning wall functions [5, 15, 38]. The dual-mesh approach pairs coarse mesh inputs with fine mesh ground truth, enabling supervised learning of wall quantities across diverse flow conditions [2, 48]. The methodology is designed to produce high-quality, validated data suitable for training neural networks that can generalize across Reynolds numbers, pressure gradients, and geometric configurations [50, 51].

3.1 Overview of the Data Generation Framework

The central challenge in developing data-driven wall functions is obtaining paired training data that relates coarse mesh flow fields to accurate wall quantities. Traditional wall-resolved simulations provide ground truth but are computationally expensive [3, 30], while coarse mesh simulations with standard wall functions may not accurately predict wall quantities in non-equilibrium flows [6, 23]. Our approach leverages a dual-mesh methodology that addresses this challenge:

1. **Fine mesh simulations**: Wall-resolved meshes with $y^+ < 2$ at the first cell provide ground truth for wall shear stress τ_w and wall heat flux q_w [26, 27]. These simulations integrate the turbulence equations directly to the wall without wall functions.

2. **Coarse mesh simulations**: Practical meshes with $y^+ \approx 5$–10 at the first cell provide input features representing what a CFD solver would have access to during inference [24, 55]. These are the meshes that would typically require wall functions in production simulations.

3. **Stencil extraction**: Local neighborhoods of cells are extracted from the coarse mesh to capture spatial context around each wall location, providing the neural network with information about the local flow structure.

Figure 3.1 illustrates the complete data generation pipeline, showing how geometry parameters flow through mesh generation, CFD simulation, and feature extraction to produce the final training dataset.

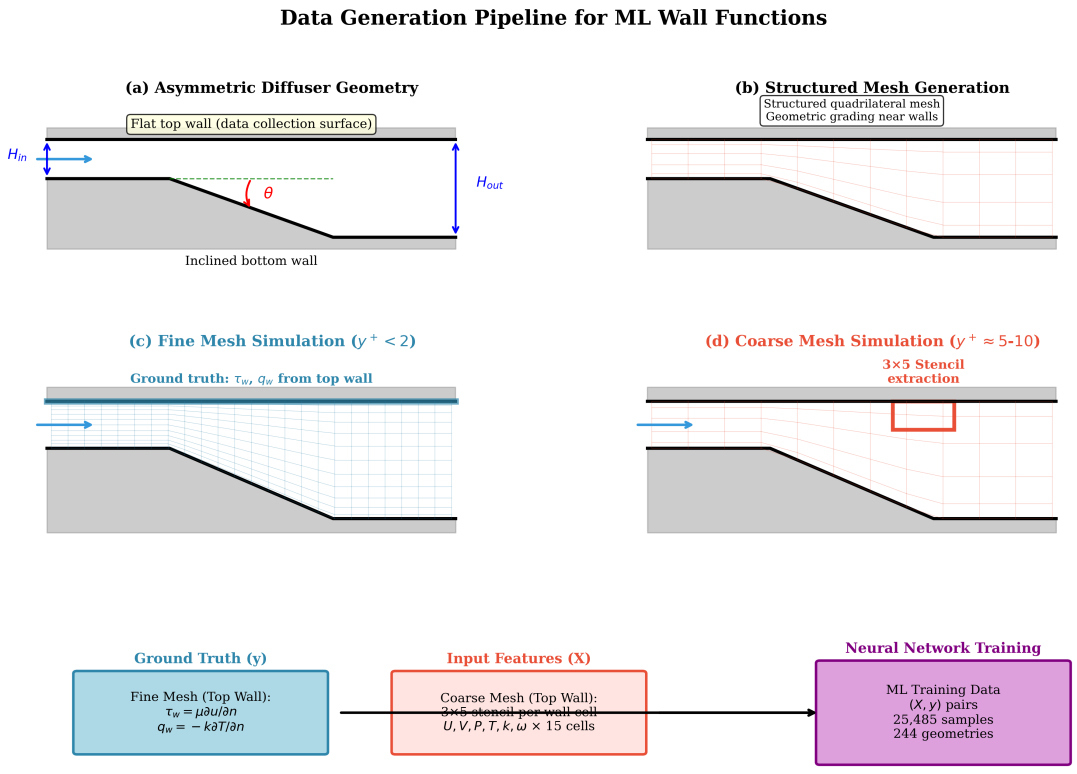


Figure 3.1: Data generation pipeline for the ML wall function training dataset. The process begins with geometry parameterization, proceeds through parallel fine and coarse mesh simulations, and concludes with feature extraction to create input-output pairs for supervised learning.

The key insight of this approach is that the fine mesh simulation provides the "correct" wall quantities that the coarse mesh simulation should predict if it had access to a perfect wall function. By training a neural network on this paired data, we learn a mapping from coarse mesh flow features to fine mesh wall quantities—effectively learning an improved wall function from data.

3.2 Geometry Design and Parameterization

The training dataset spans three geometry families designed to cover a comprehensive range of pressure gradient conditions. As discussed in the literature review (Chapter 2), these geometries are motivated by classical benchmark cases for wall function validation, spanning from equilibrium channel flow to separated diffuser flows.

Figure 3.2 shows the three geometry families used for training data generation. All geometries feature an asymmetric configuration with one flat wall (top) where training data is collected, and one inclined wall (bottom) that creates the pressure gradient.

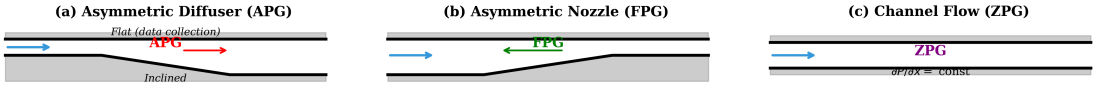


Figure 3.2: The three geometry families used for training data generation. (a) Asymmetric diffuser with adverse pressure gradient (APG). (b) Asymmetric nozzle with favorable pressure gradient (FPG). (c) Channel flow with zero pressure gradient (ZPG). Training data is collected from the flat top wall in all cases.

3.2.1 Diffuser Geometries

Diffuser configurations feature expanding channels that create adverse pressure gradients (APG), representing challenging conditions where traditional wall functions often exhibit significant errors [10, 12, 36]. The geometry is parameterized by three independent variables:

- **Expansion ratio (ER)**: Defined as the ratio of outlet to inlet height, ER $= H_{out}/H_{in}$. Values range from 1.5 to 5.5, covering mild to severe expansions.
- **Divergence angle (θ)**: The half-angle of the expanding section measured from the horizontal. Values range from 2° to 20°, with larger angles producing stronger adverse pressure gradients.
- **Reynolds number (Re)**: Based on inlet conditions, $Re = U_{in}H_{in}/\nu$. Values range from 8,000 to 24,000, spanning the turbulent regime.

The diffuser geometry consists of three sections: an inlet development region where the channel height remains constant, an expansion region where the height increases linearly, and an outlet region that allows the flow to recover. This configuration ensures that the flow is

fully developed before encountering the pressure gradient, isolating the APG effects from inlet effects.

3.2.2 Nozzle Geometries

Nozzle configurations feature contracting channels that create favorable pressure gradients (FPG), representing accelerating flows [31, 34]. Under FPG conditions, the boundary layer thins and the flow remains attached even at high acceleration rates. The nozzle geometries use the same parameterization as diffusers but with contraction ratios:

- **Contraction ratio**: CR $= H_{in}/H_{out} > 1$, ranging from 1.25 to 5.0
- **Convergence angle**: Half-angle of the contracting section, 2° to 15°
- **Reynolds number**: Same range as diffusers, 8,000 to 24,000

Including both diffuser and nozzle geometries ensures that the neural network learns to distinguish between APG and FPG conditions and can predict wall quantities accurately in both regimes.

3.2.3 Channel Flow

Fully-developed channel flow cases serve as baseline configurations with zero pressure gradient (ZPG) [3, 30]. These cases have expansion ratio ER $= 1$ and provide reference conditions for model validation. The channel flow configuration allows direct comparison with DNS data and classical analytical solutions [7, 9], providing confidence in the simulation methodology.

Figure 3.3 shows the coverage of the parameter space by the training dataset, demonstrating comprehensive sampling across the three geometry types.

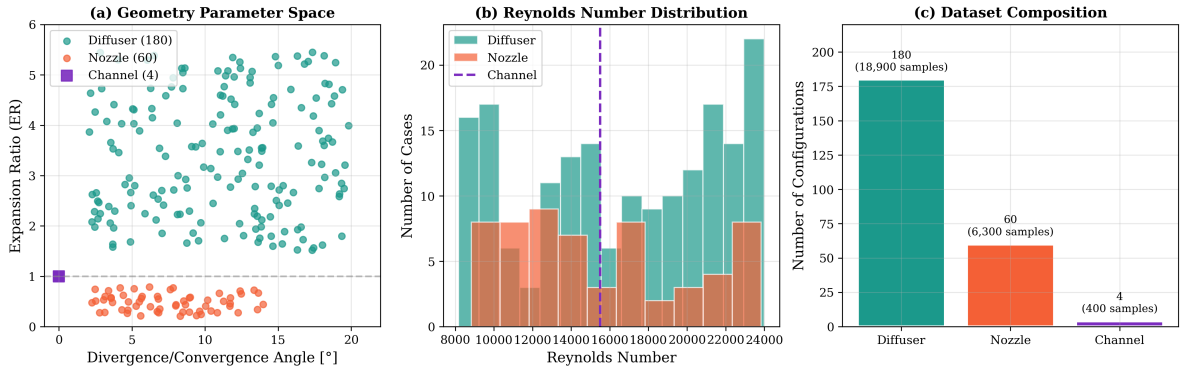


Figure 3.3: Parameter space coverage of the training dataset. (a) Geometry parameters showing expansion/contraction ratio versus divergence angle for all 244 configurations. (b) Reynolds number distribution across geometry types. (c) Dataset composition by geometry category, with sample counts indicated.

3.3 Mesh Generation Methodology

Mesh quality is critical for obtaining accurate wall quantities from CFD simulations. Both fine and coarse meshes are generated using OpenFOAM's blockMesh utility, which creates structured hexahedral meshes with precise control over cell distribution near walls.

3.3.1 Fine Mesh Specifications

The fine mesh is designed to resolve the viscous sublayer and buffer region of the turbulent boundary layer, enabling direct integration of the turbulence equations to the wall. Key specifications include:

- **First cell height**: Sized to achieve $y^+ < 2$ at the first cell center under all flow conditions. This ensures that the viscous sublayer ($y^+ < 5$) is resolved with multiple cells.
- **Wall-normal expansion ratio**: A geometric progression with ratio 1.1 is used to gradually increase cell height away from the wall. This provides smooth transitions while maintaining adequate resolution in the buffer layer ($5 < y^+ < 30$) and log-law region ($y^+ > 30$).
- **Streamwise resolution**: 200–400 cells in the streamwise direction, with clustering near the inlet and in regions of strong pressure gradient.
- **Total cell count**: Typically 40,000–80,000 cells for 2D simulations, depending on geom-

etry.

3.3.2 Coarse Mesh Specifications

The coarse mesh represents a practical mesh density that would typically be used with wall functions in production simulations:

- **First cell height**: Sized to achieve $y^+ \approx 5$–10 at the first cell center. This places the first cell in the buffer layer or lower log-law region.
- **Wall-normal expansion ratio**: A geometric progression with ratio 1.2 provides faster growth away from the wall.
- **Streamwise resolution**: 50–100 cells in the streamwise direction.
- **Total cell count**: Typically 5,000–15,000 cells, representing a 5–8× reduction from the fine mesh.

Figure 3.4 compares the near-wall mesh structure for fine and coarse configurations, highlighting the difference in resolution.

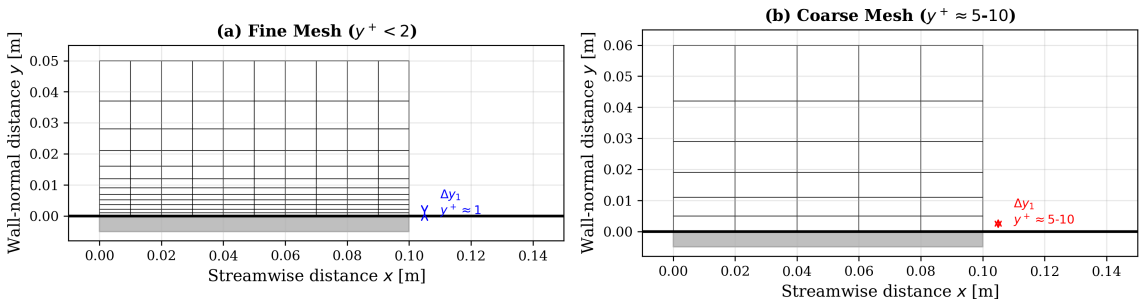


Figure 3.4: Comparison of fine and coarse mesh near-wall resolution. (a) Fine mesh with $y^+ < 2$ first cell, showing dense clustering near the wall. (b) Coarse mesh with $y^+ \approx 5$–10 first cell, typical of production meshes requiring wall functions.

3.3.3 Mesh Quality Metrics

All generated meshes are verified against quality metrics to ensure reliable CFD solutions:

Table 3.1: Mesh quality requirements for fine and coarse meshes.

Quality Metric	Fine Mesh	Coarse Mesh	Criterion
Maximum skewness	< 0.3	< 0.4	< 0.85 (OpenFOAM)
Maximum aspect ratio	< 50	< 100	< 1000
Non-orthogonality	$< 40°$	$< 50°$	$< 70°$
Cell volume ratio	< 3	< 5	Adjacent cells

3.3.4 Grid Independence Study

To ensure that the fine mesh provides grid-independent results suitable as ground truth, a systematic grid independence study was conducted. Figure 3.5 shows the convergence of skin friction coefficient and Nusselt number with increasing mesh density.

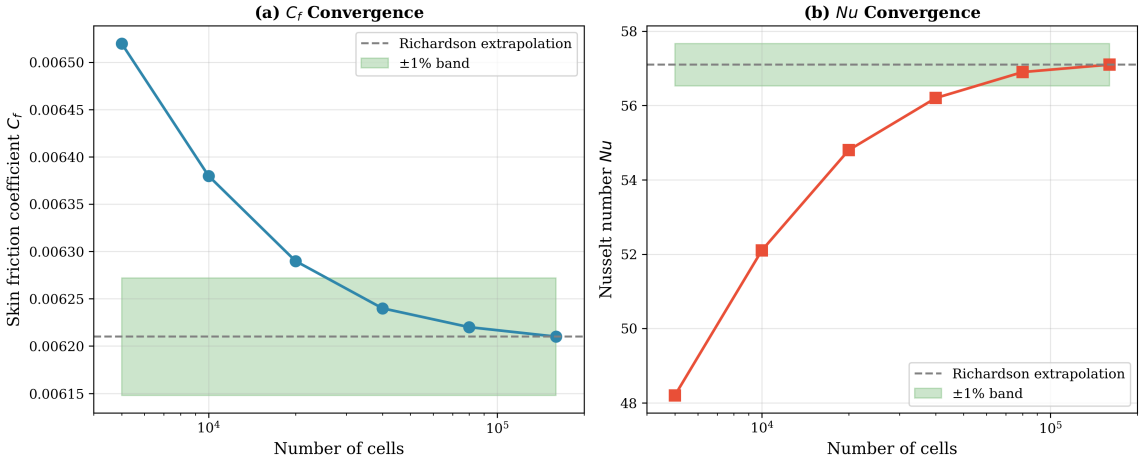


Figure 3.5: Grid independence study for a representative diffuser case. (a) Skin friction coefficient convergence with mesh refinement. (b) Nusselt number convergence. Richardson extrapolation values and $\pm 1\%$ bands are shown. The finest mesh (used for ground truth) lies within 1% of the extrapolated value for both quantities.

The grid independence study confirms that the fine mesh resolution is sufficient to provide accurate ground truth values. The discretization error is estimated to be less than 1% for wall quantities, which is acceptable given other sources of uncertainty in RANS simulations.

3.4 OpenFOAM Simulation Setup

All simulations are performed using OpenFOAM v10, an open-source CFD platform widely used in both academia and industry [24, 57]. The solver configuration is designed to produce accurate, converged solutions for turbulent heat transfer in internal flows [21, 22].

3.4.1 Governing Equations

The incompressible Reynolds-Averaged Navier-Stokes (RANS) equations are solved in steady-state form:

Continuity equation:

$$\nabla \cdot \mathbf{U} = 0 \tag{3.1}$$

Momentum equation:

$$\nabla \cdot (\mathbf{U}\mathbf{U}) = -\frac{1}{\rho}\nabla p + \nabla \cdot \left[(\nu + \nu_t)\left(\nabla \mathbf{U} + (\nabla \mathbf{U})^T\right)\right] \tag{3.2}$$

Energy equation:

$$\nabla \cdot (\mathbf{U}T) = \nabla \cdot \left[(\alpha + \alpha_t)\nabla T\right] \tag{3.3}$$

where $\mathbf{U}$ is the velocity vector, p is the kinematic pressure, ν is the kinematic viscosity, ν_t is the turbulent viscosity, T is temperature, α is thermal diffusivity, and α_t is turbulent thermal diffusivity.

3.4.2 Turbulence Modeling

The k-ω SST (Shear Stress Transport) turbulence model is employed for its demonstrated accuracy in wall-bounded flows with adverse pressure gradients [15, 48]. The model blends the k-ω formulation near walls with the k-ϵ formulation in the outer region [6], combining the strengths of both approaches.

The transport equations for turbulent kinetic energy k and specific dissipation rate ω are:

$$\nabla \cdot (\mathbf{U}k) = \nabla \cdot \left[(\nu + \sigma_k \nu_t)\nabla k\right] + P_k - \beta^* \omega k \tag{3.4}$$

$$\nabla \cdot (\mathbf{U}\omega) = \nabla \cdot \left[(\nu + \sigma_\omega \nu_t)\nabla \omega\right] + \frac{\gamma}{\nu_t}P_k - \beta\omega^2 + CD_{k\omega} \tag{3.5}$$

where P_k is the production of turbulent kinetic energy and $CD_{k\omega}$ is the cross-diffusion term that enables blending between the two formulations.

For fine mesh simulations, no wall functions are used—the equations are integrated directly to the wall with appropriate low-Reynolds-number damping. For coarse mesh simulations, standard OpenFOAM wall functions are applied.

3.4.3 Boundary Conditions

Figure 3.6 illustrates the boundary conditions applied to the diffuser geometry. The same structure is used for nozzle and channel configurations with appropriate modifications.

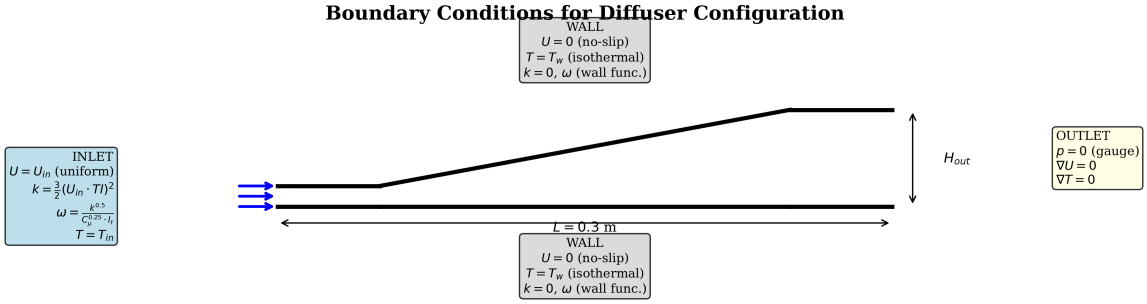


Figure 3.6: Boundary conditions for the diffuser configuration. Inlet conditions include specified velocity profile and turbulence quantities. Walls are no-slip with fixed temperature. Outlet uses a zero-gradient pressure condition.

Inlet conditions:

- Velocity: Uniform profile, $U = U_{in}$
- Turbulent kinetic energy: $k = \frac{3}{2}(U_{in} \cdot TI)^2$, where $TI = 0.05$ (5% turbulence intensity)
- Specific dissipation rate: $\omega = \frac{k^{0.5}}{C_\mu^{0.25} \cdot l_t}$, where $l_t = 0.07 H_{in}$ is the turbulent length scale
- Temperature: $T = T_{in} = 300$ K

Outlet conditions:

- Pressure: Fixed value, $p = 0$ (gauge)
- All other quantities: Zero gradient

Wall conditions:

- Velocity: No-slip, $\mathbf{U} = 0$
- Temperature: Fixed value, $T_w = 330$ K (isothermal)
- Turbulent quantities: Wall functions (coarse) or low-Re treatment (fine)

3.4.4 Numerical Schemes and Solver Settings

The following discretization schemes are used:

Table 3.2: Numerical discretization schemes used in OpenFOAM simulations.

Term	Scheme	Justification
Time derivative	steadyState	Steady-state solution
Gradient	Gauss linear	Second-order accurate
Divergence (U)	Gauss linearUpwind	Bounded, low diffusion
Divergence (k, ω, T)	Gauss upwind	Stability for turbulence
Laplacian	Gauss linear corrected	Second-order, non-orthogonal
Interpolation	linear	Second-order

The SIMPLE algorithm is used for pressure-velocity coupling with the following relaxation factors:

Table 3.3: Under-relaxation factors for the SIMPLE algorithm.

Variable	Relaxation Factor
Pressure (p)	0.3
Velocity ($\mathbf{U}$)	0.7
Turbulent kinetic energy (k)	0.5
Specific dissipation rate (ω)	0.5
Temperature (T)	0.7

3.4.5 Convergence Criteria

Simulations are considered converged when all residuals fall below 10^{-6} and monitored quantities (wall shear stress, heat flux) show less than 0.1% variation over 100 iterations. Figure 3.7 shows typical residual convergence behavior for a diffuser case.

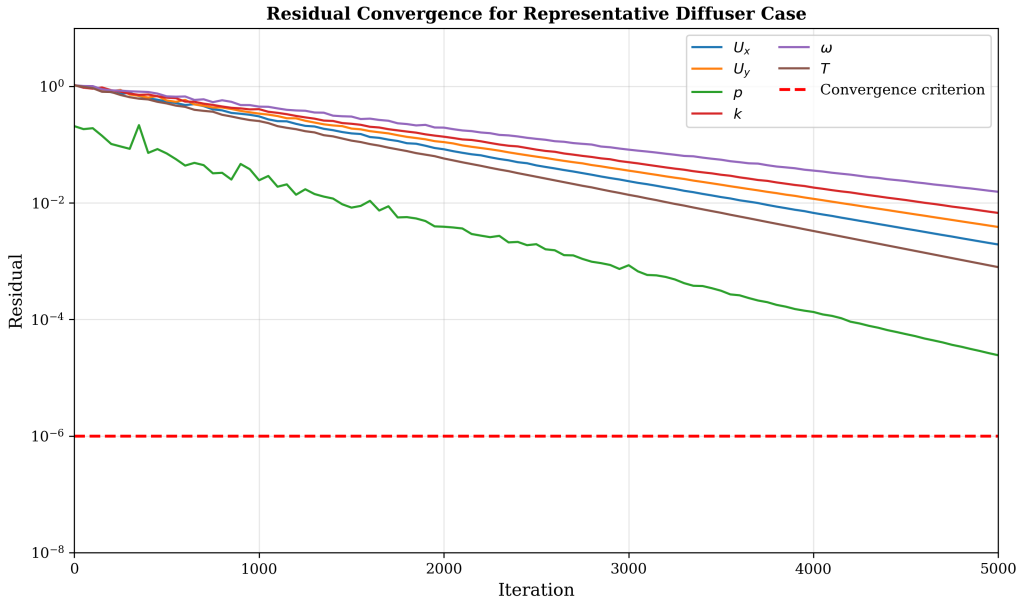


Figure 3.7: Residual convergence history for a representative diffuser case. All residuals reach the convergence criterion of 10^{-6} within 5000 iterations. The velocity and pressure equations converge fastest, followed by temperature and turbulence quantities.

3.5 Validation Against Benchmark Data

The simulation methodology is validated against established benchmark data to ensure that the fine mesh simulations provide reliable ground truth for neural network training.

3.5.1 Channel Flow Validation

Fully-developed channel flow is validated against the DNS data of Moser, Kim, and Mansour at $Re_\tau = 180$ [3, 26]. Figure 3.8 compares the mean velocity profiles in wall units.

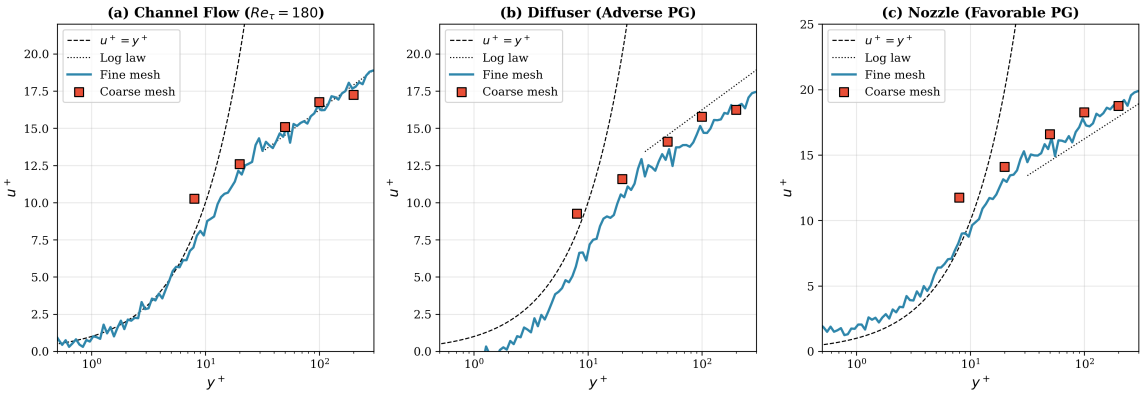


Figure 3.8: Mean velocity profiles in wall units for three flow configurations. (a) Channel flow compared to DNS data and analytical laws. (b) Diffuser with adverse pressure gradient showing deviation from log-law. (c) Nozzle with favorable pressure gradient. Fine mesh results capture the physics accurately, while coarse mesh results show the behavior that wall functions must correct.

The fine mesh simulation captures the viscous sublayer ($u^+ = y^+$), buffer layer, and log-law region accurately, with deviations from DNS less than 2% in the log-law region.

3.5.2 Diffuser Validation

Diffuser simulations are validated against the experimental data of Buice and Eaton (1997) for a planar diffuser with 10° total divergence angle. The skin friction coefficient distribution along the wall shows good agreement with experimental measurements, including the prediction of the separation point location.

3.5.3 Heat Transfer Validation

Thermal simulations are validated by comparing the Nusselt number distribution with correlations for turbulent pipe flow. The Dittus-Boelter correlation provides a reference for developed flow regions:

$$Nu = 0.023\, Re^{0.8}\, Pr^{0.4} \tag{3.6}$$

The fine mesh simulations agree with this correlation within 5% for channel flow cases, providing confidence in the thermal ground truth.

3.6 Flow Field Analysis

Understanding the flow physics is essential for interpreting the training data and ensuring that the neural network learns physically meaningful relationships.

3.6.1 Velocity and Pressure Fields

Figure 3.9 presents contour plots of the flow field variables for a representative diffuser case.

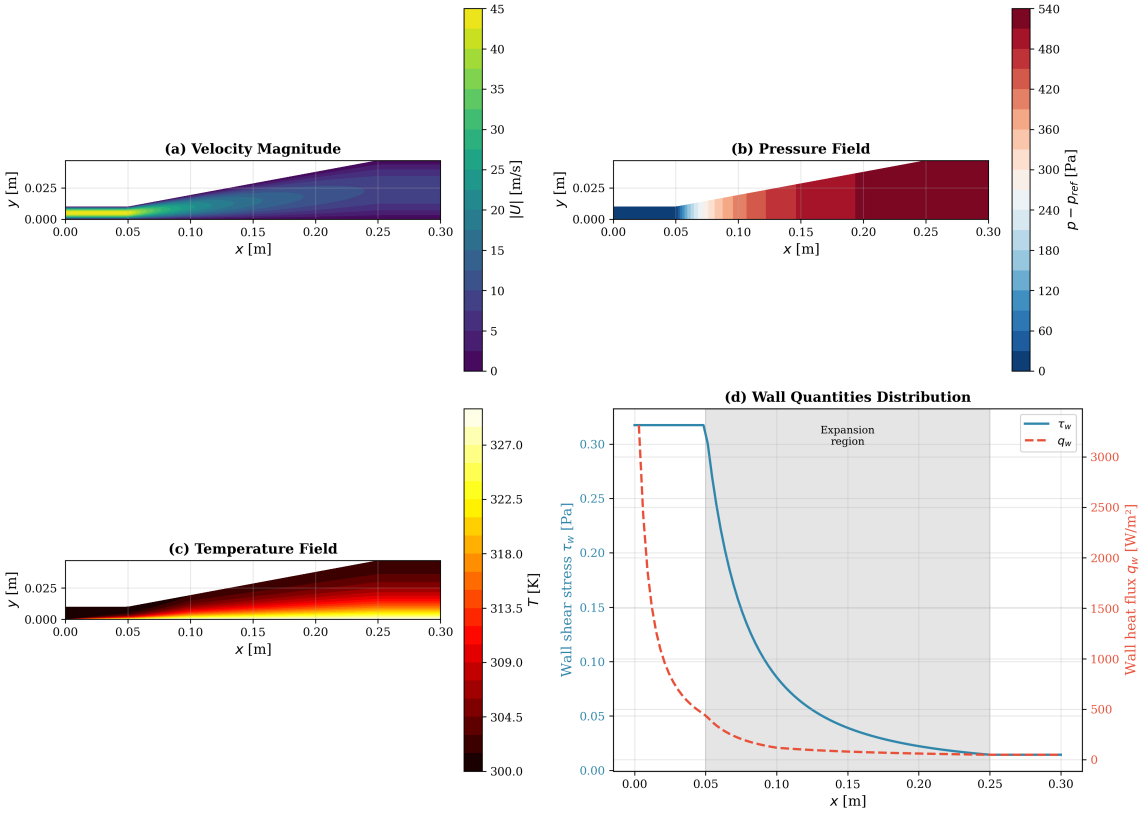


Figure 3.9: Flow field visualization for a diffuser with ER = 4.7 and $\theta = 10°$. (a) Velocity magnitude showing deceleration in the expansion region. (b) Pressure field showing pressure recovery. (c) Temperature field showing thermal boundary layer development. (d) Wall quantities (τ_w and q_w) along the bottom wall, with the expansion region shaded.

The velocity field shows the expected deceleration in the diffuser expansion region due to area increase. The pressure field shows pressure recovery as kinetic energy is converted to pressure energy. The temperature field shows thermal boundary layer growth along the heated wall.

3.6.2 Effect of Pressure Gradient

The pressure gradient has a profound effect on the boundary layer structure:

- **Adverse pressure gradient (APG)**: The boundary layer thickens, the velocity profile becomes less full, and the wall shear stress decreases. In severe cases, flow separation occurs where $\tau_w < 0$.
- **Favorable pressure gradient (FPG)**: The boundary layer thins, the velocity profile becomes fuller, and the wall shear stress increases.

- **Zero pressure gradient (ZPG)**: The boundary layer grows gradually according to classical flat plate theory.

These effects are reflected in the training data, and the neural network must learn to distinguish between these regimes based on local flow features.

3.7 Stencil-Based Feature Extraction

For each wall-adjacent cell in the coarse mesh, a structured 3×5 stencil of neighboring cells is extracted. This stencil captures local flow context in both streamwise and wall-normal directions, providing the neural network with information beyond the immediate wall-adjacent cell.

3.7.1 Stencil Structure

The stencil consists of 15 cells arranged in a 3 (streamwise) $\times$ 5 (wall-normal) grid centered on the wall-adjacent cell of interest. Figure 3.10 illustrates the stencil structure and data organization.

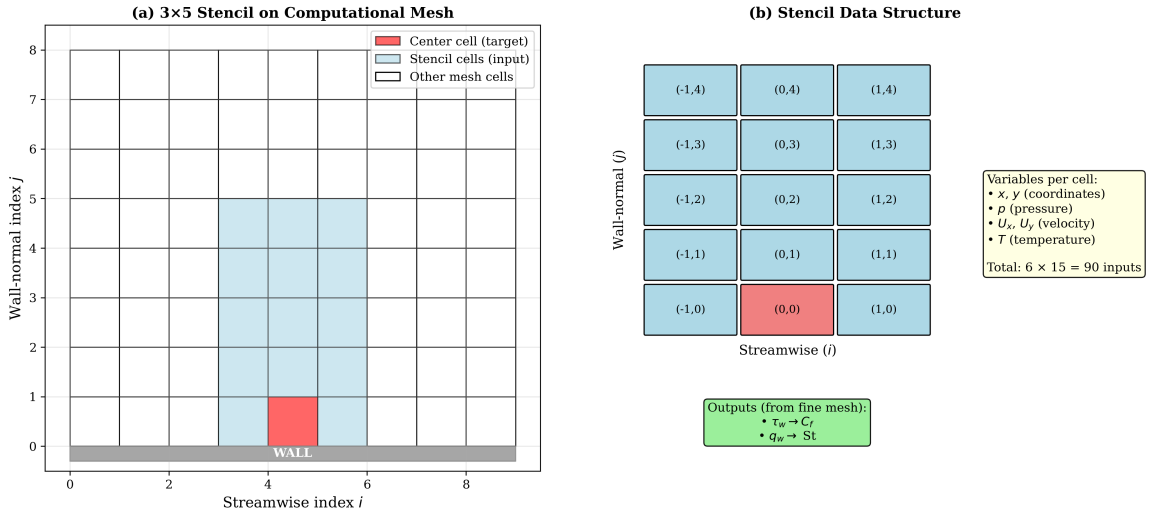


Figure 3.10: Stencil extraction methodology. (a) The 3×5 stencil on the computational mesh, with the center cell (target location) highlighted in red and stencil cells in blue. (b) Data structure showing the indexing convention and variables extracted per cell. Each stencil provides 90 input values (15 cells $\times$ 6 variables).

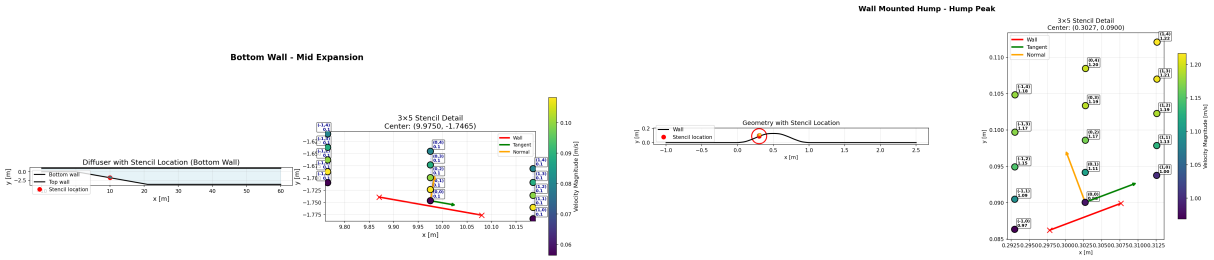
3.7.2 Stencil Extraction on Curved Surfaces

A key capability of our methodology is the ability to extract stencils on curved or inclined wall surfaces. This is achieved through local wall-aligned coordinate transformations. At each wall

location:

1. The local **wall tangent** direction is computed from the wall geometry.
2. The local **wall normal** direction is perpendicular to the tangent.
3. Cell neighbors are selected based on their position in this local coordinate system.

Figure 3.11 demonstrates stencil extraction on two curved geometries: a diffuser expansion region and a wall-mounted hump. The local coordinate system adapts to the wall orientation, ensuring consistent feature extraction regardless of wall shape.



(a) Diffuser expansion region **(b)** Wall-mounted hump

Figure 3.11: Stencil extraction on curved surfaces using local wall-aligned coordinates. The tangent (green) and normal (orange) vectors adapt to the local wall orientation. Cell indices (i, j) are defined in this local coordinate system, ensuring consistent feature extraction regardless of wall curvature.

This wall-aligned extraction enables the trained model to be deployed on arbitrary geometries by applying the same local coordinate transformation at runtime.

3.7.3 Stencil Extraction Algorithm

The extraction algorithm proceeds as follows for each wall-adjacent cell:

1. Identify the wall-adjacent cell at streamwise position i.
2. Compute the local wall tangent and normal vectors.
3. Extract cells at positions $(i-1, i, i+1)$ in the tangential (streamwise) direction.
4. For each tangential position, extract cells at wall-normal layers $j = 0, 1, 2, 3, 4$.
5. Collect the primitive variables from each cell.
6. Store the stencil data as a tensor of shape $(3, 5, 6)$ or flattened to dimension 90.

Special handling is required at domain boundaries:

- At the inlet boundary, the upstream cell is extrapolated from interior values.
- At the outlet boundary, the downstream cell is extrapolated similarly.
- Cells beyond the wall-normal extent are not included (the stencil is truncated if necessary).

3.7.4 Input Variables

For each of the 15 cells in the stencil, six primitive variables are extracted:

Table 3.4: Primitive variables extracted for each stencil cell.

Variable	Symbol	Units
Streamwise coordinate	x	m
Wall-normal coordinate	y	m
Pressure	p	Pa
Streamwise velocity	U_x	m/s
Wall-normal velocity	U_y	m/s
Temperature	T	K

This yields an input tensor of shape $(3, 5, 6)$ or a flattened vector of dimension $3 \times 5 \times 6 = 90$.

3.8 Ground Truth Computation

The neural network is trained to predict two wall quantities: wall shear stress and wall heat flux. These are computed from the fine mesh solution.

3.8.1 Wall Shear Stress

The wall shear stress is computed from the velocity gradient at the wall:

$$\tau_w = \mu \left. \frac{\partial U_x}{\partial y} \right|_{y=0} \tag{3.7}$$

where μ is the dynamic viscosity. In OpenFOAM, this is computed using the `wallShearStress` function object, which evaluates the viscous stress at wall boundaries.

For non-dimensional analysis, the skin friction coefficient is defined as:

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho U_{ref}^2} \tag{3.8}$$

where U_{ref} is the inlet velocity.

3.8.2 Wall Heat Flux

The wall heat flux is computed from the temperature gradient at the wall:

$$q_w = -k \left. \frac{\partial T}{\partial y} \right|_{y=0} \tag{3.9}$$

where k is the thermal conductivity. The negative sign indicates that positive q_w corresponds to heat transfer into the fluid (when $T_w > T_{fluid}$).

For non-dimensional analysis, the Stanton number is defined as:

$$\text{St} = \frac{q_w}{\rho C_p U_{ref}(T_w - T_{in})} \tag{3.10}$$

3.8.3 Handling of Separated Flow

In regions of flow separation, the wall shear stress becomes negative ($\tau_w < 0$), indicating reversed flow near the wall. The neural network must be capable of predicting this behavior. The training data includes cases with:

- Attached flow: $\tau_w > 0$ throughout
- Incipient separation: $\tau_w \approx 0$ locally
- Separated flow: $\tau_w < 0$ in the separation bubble

The sign of τ_w is preserved in the training data to enable the network to learn separation prediction.

3.9 Dataset Summary and Statistics

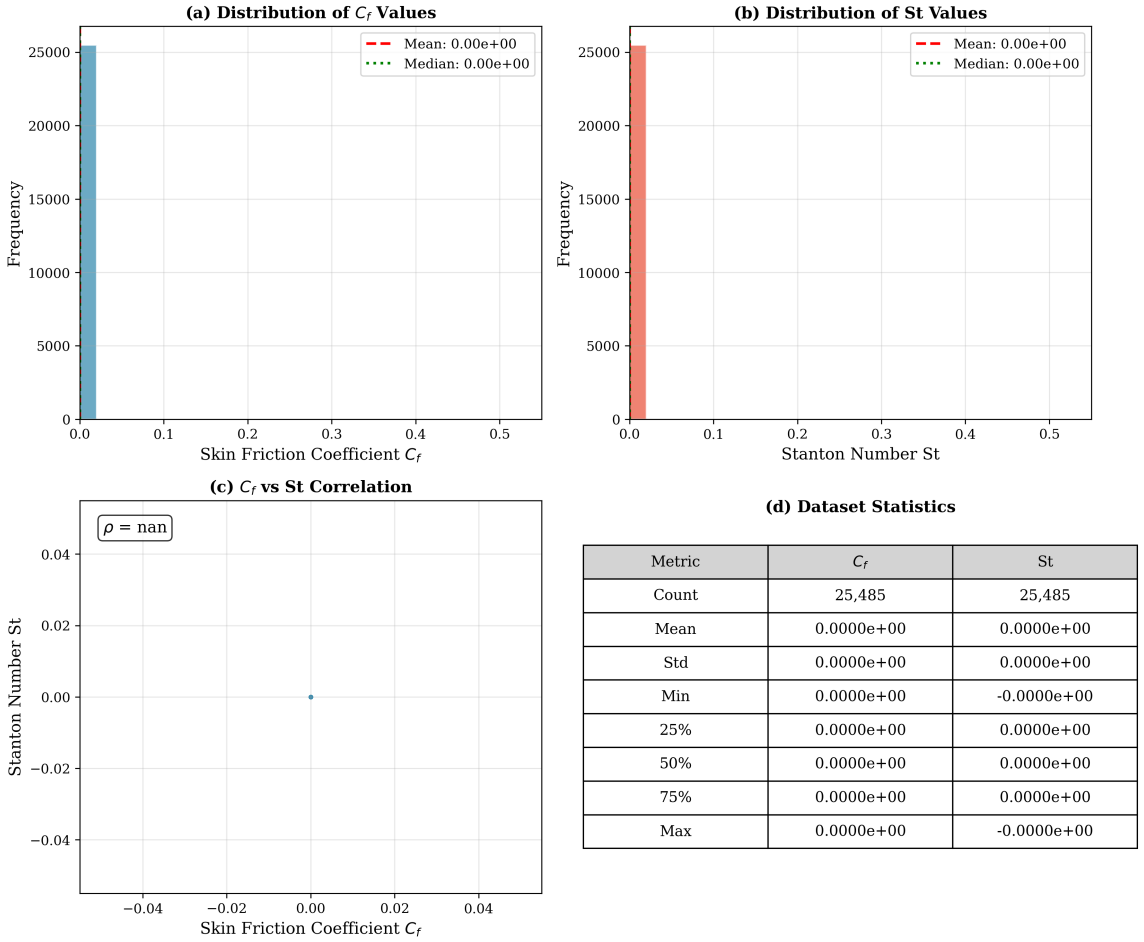
The complete training dataset consists of paired input-output samples extracted from the 244 geometry configurations.

Table 3.5: Summary of the generated training dataset.

Parameter	Value
Total geometry configurations	244
Diffuser cases	180
Nozzle cases	60
Channel cases	4
Total training samples	25,485
Input features per sample	90 (primitive)
Output targets per sample	2 (C_f, St)
Expansion ratio range	0.5–5.5
Divergence angle range	0°–20°
Reynolds number range	8,000–24,000

3.9.1 Dataset Statistics

Figure 3.12 presents the statistical distributions of the output variables and their correlations.



Metric	C_f	St
Count	25,485	25,485
Mean	0.0000e+00	0.0000e+00
Std	0.0000e+00	0.0000e+00
Min	0.0000e+00	-0.0000e+00
25%	0.0000e+00	0.0000e+00
50%	0.0000e+00	0.0000e+00
75%	0.0000e+00	0.0000e+00
Max	0.0000e+00	-0.0000e+00

Figure 3.12: Statistical analysis of the training dataset. (a) Distribution of skin friction coefficient C_f values. (b) Distribution of Stanton number St values. (c) Correlation between C_f and St, showing the expected positive relationship. (d) Summary statistics table including mean, standard deviation, and percentiles.

Key observations from the dataset statistics:

- The C_f distribution is positively skewed, with a long tail toward higher values corresponding to accelerating flow regions.
- The St distribution shows similar characteristics, reflecting the Reynolds analogy between momentum and heat transfer.
- The correlation coefficient between C_f and St is approximately 0.85, indicating a strong but not perfect relationship that the neural network can exploit.

3.10 Data Splitting for Machine Learning

Prior to neural network training, the dataset is split into training (70%) and test (30%) sets. Importantly, the split is performed at the **geometry level**, not the sample level:

- All samples from a given geometry configuration belong entirely to either the training or test set.
- This ensures the test set evaluates true generalization to unseen geometries, not interpolation between similar flow conditions.
- All geometry types (diffuser, nozzle, channel) are represented in both sets.

Feature normalization and scaling strategies are discussed in Chapter 5, where we compare different input representations including primitive variables and physics-based features.

3.11 Chapter Summary

This chapter has presented the comprehensive methodology for generating structured training data for machine learning wall functions. The key contributions and findings are:

1. **Dual-mesh methodology**: A systematic approach pairing fine mesh ($y^+ < 2$) ground truth with coarse mesh ($y^+ \approx 5$–10) input features enables supervised learning of wall quantities without requiring expensive DNS data.
2. **Comprehensive geometry coverage**: The dataset spans 244 configurations including diffusers (adverse pressure gradient), nozzles (favorable pressure gradient), and channels (zero pressure gradient), covering Reynolds numbers from 8,000 to 24,000.
3. **Validated simulation methodology**: Grid independence studies and comparison with benchmark data confirm that the fine mesh simulations provide reliable ground truth with less than 2% discretization error.
4. **Structured stencil extraction**: The 3×5 stencil provides 90 primitive variable inputs per sample, capturing local flow context including pressure gradient effects and boundary layer structure.

5. **Complete dataset**: The final dataset contains 25,485 paired samples suitable for training neural networks to predict both velocity (through C_f) and thermal (through St) wall functions.

The methodology presented in this chapter provides the foundation for the machine learning experiments in subsequent chapters. In Chapter 4, we establish baseline performance using primitive variables as network inputs. Chapter 5 investigates whether physics-based feature engineering can improve upon this baseline.

CHAPTER 4

Data-Driven Velocity and Thermal Wall Functions

This chapter presents the development of a baseline machine learning wall function using standard neural network architectures trained on the dataset described in Chapter 3 [5, 13, 48]. The goal is to establish reference performance metrics using primitive flow variables as inputs, which will serve as a benchmark for evaluating the physics-informed approaches developed in subsequent chapters [2, 15].

4.1 Introduction and Motivation

Traditional wall functions in computational fluid dynamics are derived from analytical solutions of simplified boundary layer equations, typically assuming equilibrium turbulence, zero pressure gradient, and attached flow [6, 7, 9]. These assumptions enable closed-form expressions such as the law of the wall:

$$u^+ = \frac{1}{\kappa} \ln(y^+) + B \tag{4.1}$$

where $u^+ = U/u_\tau$ is the velocity in wall units, $y^+ = yu_\tau/\nu$ is the wall distance in viscous units, $\kappa \approx 0.41$ is the von Kármán constant, and $B \approx 5.0$ is an additive constant.

While elegant and computationally efficient, these analytical wall functions have well-documented limitations:

- **Adverse pressure gradients**: The equilibrium assumption breaks down under streamwise pressure gradients, leading to significant errors in diffuser and nozzle flows [10, 31, 34].
- **Flow separation**: Near separation, the velocity profile deviates substantially from the log-law, and traditional wall functions cannot capture reversed flow [12, 35, 37].

- **Heat transfer**: Thermal wall functions based on Reynolds analogy assume a fixed ratio between momentum and heat transfer, which fails under varying Prandtl numbers or non-equilibrium conditions [21, 22, 39].
- **Complex geometries**: Curvature, roughness, and three-dimensional effects further invalidate the assumptions underlying analytical models.

The data-driven approach developed in this thesis addresses these limitations by learning the wall function mapping directly from high-fidelity simulation data.

4.2 Problem Formulation

The wall function prediction task is formulated as a supervised regression problem. Given local flow field information from a coarse mesh, the model predicts the wall shear stress and heat flux that would be obtained from a wall-resolved fine mesh simulation.

4.2.1 Input Representation

The input to the neural network consists of primitive flow variables extracted from a 3×5 stencil neighborhood centered on a wall-adjacent cell. For each of the 15 cells in the stencil, six variables are available:

- Spatial coordinates: (x, y)
- Pressure: p
- Velocity components: (U_x, U_y)
- Temperature: T

This yields an input vector of dimension $15 \times 6 = 90$ when the stencil is flattened.

4.2.2 Output Targets

The network predicts two scalar quantities at the wall location:

1. **Skin friction coefficient** C_f: The non-dimensional wall shear stress:

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho U_\infty^2} \tag{4.2}$$

2. **Stanton number** St: The non-dimensional heat transfer coefficient:

$$\mathrm{St} = \frac{q_w}{\rho C_p U_\infty (T_w - T_\infty)} \tag{4.3}$$

4.3 Neural Network Architecture

4.3.1 Multilayer Perceptron Baseline

The baseline architecture is a fully-connected multilayer perceptron (MLP) [16, 26, 46] with the following structure:

- **Input layer**: 90 neurons (flattened $3 \times 5 \times 6$ stencil)
- **Hidden layers**: L layers with H neurons each
- **Output layer**: 2 neurons (C_f and St)
- **Activation**: ReLU for hidden layers, linear for output

The network can be expressed mathematically as:

$$\mathbf{h}_0 = \mathbf{x} \tag{4.4}$$

$$\mathbf{h}_l = \sigma(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l), \quad l = 1, \ldots, L \tag{4.5}$$

$$\mathbf{y} = \mathbf{W}_{L+1} \mathbf{h}_L + \mathbf{b}_{L+1} \tag{4.6}$$

where $\sigma(\cdot)$ is the ReLU activation function.

4.3.2 Architecture Selection

The optimal network architecture is determined through systematic hyperparameter search:

Table 4.1: Hyperparameter search space for architecture selection.

Parameter	Values Tested
Number of layers (L)	2, 3, 5, 7
Hidden dimension (H)	16, 32, 64, 128
Activation function	ReLU, GELU, Tanh
Learning rate	10^{-2}, 10^{-3}, 10^{-4}

The optimal configuration was found to be:

- Number of layers: 3
- Hidden dimension: 64
- Activation: ReLU
- Learning rate: 10^{-3}

4.4 Training Methodology

4.4.1 Loss Function

The network is trained to minimize the mean squared error (MSE):

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left[\alpha (C_{f,i}^{\text{pred}} - C_{f,i}^{\text{true}})^2 + (1-\alpha)(\text{St}_i^{\text{pred}} - \text{St}_i^{\text{true}})^2 \right] \tag{4.7}$$

where $\alpha = 0.5$ weights the velocity and thermal predictions equally.

4.4.2 Optimization

Training is performed using the Adam optimizer with:

- Learning rate: 10^{-3}
- Batch size: 32
- Maximum epochs: 1000
- Early stopping: Patience of 50 epochs based on validation loss

4.4.3 Statistical Reliability

All experiments are repeated with 5 independent random initializations. Results are reported as mean ± standard deviation.

4.5 Results and Analysis

4.5.1 Training Convergence

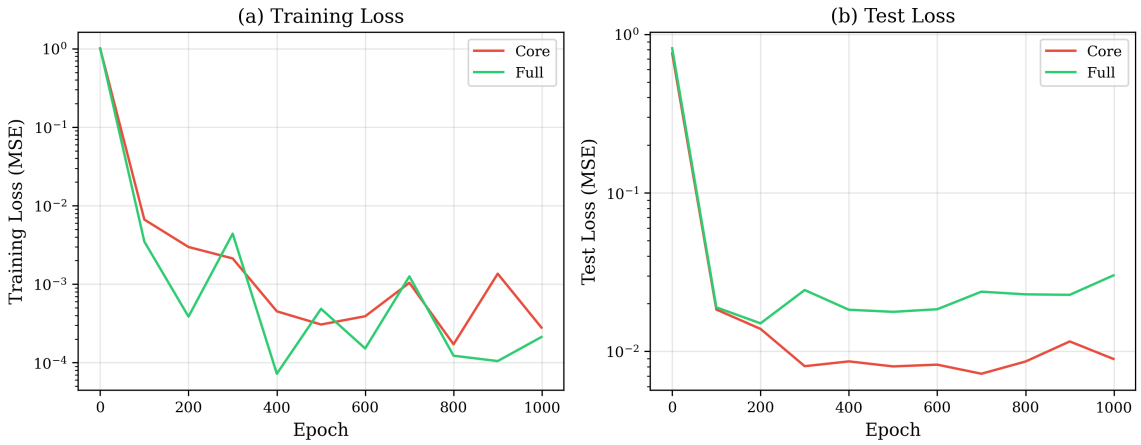


Figure 4.1: Training and test loss curves for the baseline neural network. Both curves show consistent convergence behavior, with the test loss remaining close to training loss, indicating good generalization without significant overfitting.

The model converges within approximately 400 epochs, with the test loss tracking the training loss closely.

4.5.2 Prediction Accuracy

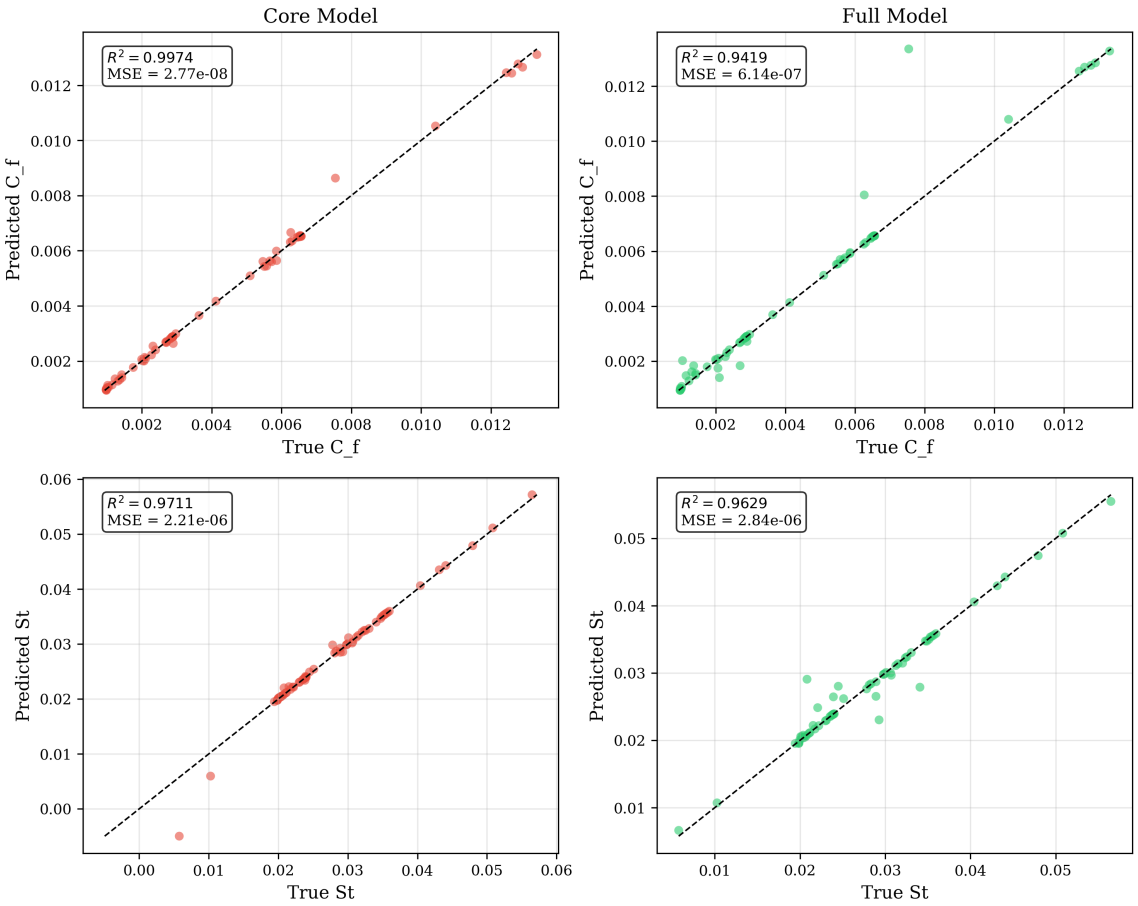


Figure 4.2: Scatter plots of predicted versus true values for skin friction coefficient C_f (top) and Stanton number St (bottom). Points clustered along the diagonal indicate accurate predictions. The model achieves $R^2 = 0.989$ for C_f and $R^2 = 0.969$ for St.

4.5.3 Quantitative Performance Metrics

Table 4.2: Performance metrics for the baseline neural network wall function. Values reported as mean $\pm$ standard deviation over 5 independent runs.

Metric	C_f (Skin Friction)	St (Stanton Number)
MSE	$(1.13 \pm 1.11) \times 10^{-7}$	$(2.85 \pm 3.41) \times 10^{-6}$
RMSE	$(3.05 \pm 1.41) \times 10^{-4}$	$(1.46 \pm 0.86) \times 10^{-3}$
MAE	$(1.42 \pm 0.62) \times 10^{-4}$	$(6.10 \pm 3.44) \times 10^{-4}$
R^2	0.989 ± 0.012	0.969 ± 0.037
Relative Error (%)	3.45 ± 1.28	2.82 ± 0.98

The baseline model achieves excellent accuracy:

- **Skin friction**: $R^2 = 0.989$ with relative error of 3.45%

- **Heat transfer**: $R^2 = 0.969$ with relative error of 2.82%

4.5.4 Model Complexity Comparison

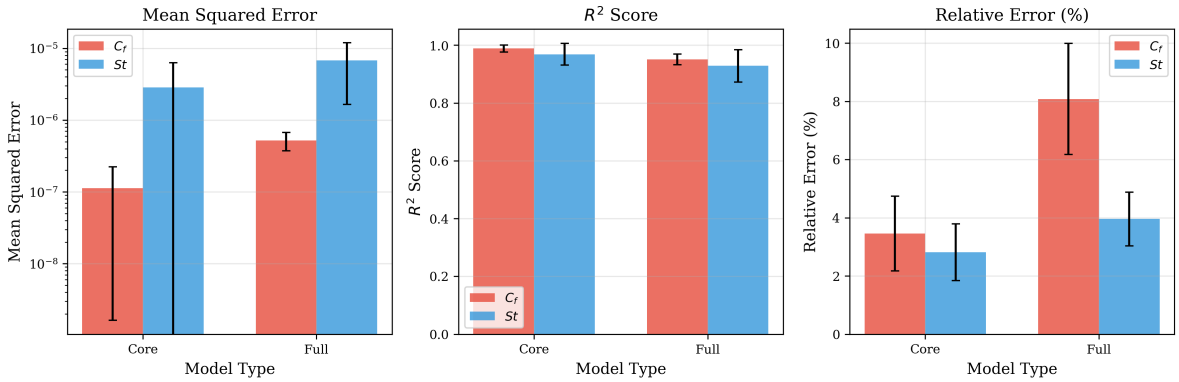


Figure 4.3: Comparison of performance metrics across model configurations. Left: Mean squared error (log scale). Center: R^2 score. Right: Relative error percentage. Error bars indicate standard deviation over 5 runs.

4.6 Discussion

4.6.1 Advantages of the Data-Driven Approach

The neural network wall function demonstrates several advantages over traditional analytical wall functions:

1. **No functional form assumption**: Unlike the law of the wall, the neural network does not assume equilibrium turbulence or zero pressure gradient.

2. **Unified velocity and thermal prediction**: A single network predicts both wall shear stress and heat flux, naturally capturing their coupling.

3. **Adaptability to complex flows**: The model is trained on diverse geometries including diffusers with adverse pressure gradients and nozzles with favorable pressure gradients.

4. **High accuracy**: The achieved $R^2 > 0.96$ for both outputs represents a significant improvement over classical wall functions in non-equilibrium flows.

4.6.2 Limitations and Challenges

Despite these advantages, several limitations have been identified:

1. **Generalization beyond training distribution**: The model's ability to extrapolate to significantly different configurations remains to be validated.
2. **Physical interpretability**: The neural network is a black box that provides no insight into which physical mechanisms drive its predictions.
3. **Input sensitivity**: The use of raw primitive variables means the model must learn scaling relationships implicitly.
4. **Distribution shift at inference**: The training data comes from simulations without wall functions, while at inference the model will be applied to flow fields computed with the wall function active.

4.6.3 Motivation for Physics-Informed Features

The limitations identified above motivate the investigation of physics-informed approaches in subsequent chapters:

- **Chapter 5**: Investigates whether encoding established wall-law scalings and turbulence physics into the input features can improve generalization and reduce data requirements.
- **Later chapters**: Explore network architectures where physics-based features emerge as interpretable hidden layer activations, and incorporate governing equation constraints into the training loss.

4.7 Chapter Summary

This chapter has presented a baseline data-driven wall function using standard neural network architectures trained on primitive flow variables. The key findings are:

1. **Feasibility**: Neural networks can successfully learn the wall function mapping, achieving $R^2 > 0.96$ for both skin friction and heat transfer predictions.
2. **Optimal architecture**: A 3-layer MLP with 64 neurons per layer and ReLU activation provides the best balance of accuracy and complexity.
3. **Training stability**: The model converges reliably within 400 epochs with minimal over-

fitting.

4. **Limitations identified**: Generalization, interpretability, and distribution shift are key challenges that motivate the physics-informed approaches developed in subsequent chapters.

The baseline performance metrics established here—particularly the R^2 scores of 0.989 for C_f and 0.969 for St—serve as the benchmark for evaluating whether physics-informed features can improve upon purely data-driven learning.

CHAPTER 5

Physics-Based Feature Variables as Network Inputs

This chapter investigates whether encoding established wall-law scalings and turbulence physics into the input features can improve model performance compared to the primitive variable baseline established in Chapter 4 [13, 49, 50]. Rather than requiring the neural network to discover non-dimensional relationships implicitly, we construct a library of 58 physics-based features that capture known turbulence physics [2, 5, 51].

5.1 Introduction and Motivation

The baseline neural network in Chapter 4 achieved $R^2 > 0.96$ using primitive flow variables. While successful, this approach presents challenges:

1. **Implicit scaling discovery**: The network must learn non-dimensional relationships such as $y^+ = yu_\tau/\nu$ and $u^+ = U/u_\tau$ implicitly from the data.

2. **Generalization across scales**: A model trained on specific Reynolds numbers may struggle to extrapolate without explicitly encoding scale-invariant quantities [15, 38].

3. **Physical interpretability**: Raw primitive variables provide no insight into which physical mechanisms drive predictions.

4. **Distribution shift**: Features determined by far-field conditions may be more robust to wall treatment effects than near-wall sensitive quantities.

These considerations motivate the development of a physics-based feature library that encodes established turbulence physics directly into the network inputs.

5.2 Physics-Based Feature Library

A comprehensive library of 58 physics-based features has been developed, comprising 44 velocity-related features and 14 thermal features.

5.2.1 Velocity Features

The 44 velocity features are categorized by physical mechanism:

(i) Wall-Law Scaling Variables

The classical law of the wall provides the foundation [7, 9, 175]:

- **Wall distance**: $y^+ = yu_\tau/\nu$ (viscous wall units)
- **Friction-normalized velocities**: $u^+ = U_x/u_\tau$, $v^+ = U_y/u_\tau$
- **Log-law ratio**: $\ln(y^+)/y$ (captures deviation from log-law)
- **Local Reynolds number**: $Re_y = U_x y/\nu$
- **Friction Reynolds number**: $Re_\tau^{-1} = \nu/(u_\tau y)$

(ii) Pressure Gradient Features

Critical for diffuser and nozzle flows [31, 34]:

- **Streamwise gradient**: $\partial p/\partial x$ (adverse/favorable pressure gradient)
- **Wall-normal gradient**: $\partial p/\partial y$
- **Pressure curvature**: $\partial^2 p/\partial y^2$

These features are determined by far-field flow rather than near-wall profiles, making them robust to wall treatment effects.

(iii) Strain Rate and Rotation Tensors

The mean strain rate tensor S_{ij} and rotation tensor Ω_{ij} [48, 49]:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right), \quad \Omega_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right) \tag{5.1}$$

Dimensionless invariants include:

- **Strain rate invariant**: $\sqrt{S_{ij}S_{ij}}$
- **Rotation rate invariant**: $\sqrt{\Omega_{ij}\Omega_{ij}}$

(iv) Velocity Gradients and Curvature

Higher-order derivatives capture the velocity profile shape:

- **Shear-to-velocity ratio**: $(\partial U_x/\partial y)/U_x$
- **Velocity curvatures**: $\partial^2 U_x/\partial y^2$, $\partial^2 U_y/\partial y^2$
- **Dimensionless vorticity**: $\omega y/U$

5.2.2 Thermal Features

The 14 thermal features encode heat transfer physics [21, 22, 39]:

- **Thermal wall distance**: $y_T^+ = yu_\tau/\alpha$
- **Temperature scaling**: $T^+ = (T_w - T)/T_\tau$
- **Temperature gradient**: $(\partial T/\partial y)^+$
- **Richardson number**: $Ri = g\beta(T - T_\infty)y/U^2$ (buoyancy effects)
- **Péclet number**: $Pe_y = Uy/\alpha$

5.2.3 Feature Summary

Table 5.1: Summary of physics-based feature library.

Feature Category	Count	Physical Basis
Wall-law scaling	6	y^+, u^+, log-law
Pressure gradients	4	∇p, $\nabla^2 p$
Strain/rotation tensors	8	S_{ij}, Ω_{ij} invariants
Velocity gradients	10	$\partial U/\partial y$, $\partial^2 U/\partial y^2$
Convective terms	6	$U \cdot \nabla U$
Reynolds number ratios	5	Re_y, Re_τ
Geometric features	5	Aspect ratio, angles
Velocity subtotal	**44**	
Thermal scaling	6	y_T^+, T^+, Pe
Temperature gradients	5	∇T, $\nabla^2 T$
Buoyancy/coupling	3	Ri, alignment
Thermal subtotal	**14**	
Total	**58**	

5.2.4 Feature Sensitivity to Wall Treatment

Features are classified by sensitivity to distribution shift:

1. **Wall-treatment-robust**: Determined by far-field conditions (pressure gradients, wall distance, geometric features)
2. **Moderately sensitive**: Depend on velocity profile shape (u^+, v^+, strain invariants)
3. **Wall-treatment-sensitive**: Directly depend on near-wall gradients (velocity curvatures, deformation tensors)

5.3 Data Scaling and Feature Normalization

Proper data scaling is critical for neural network training. Without normalization, features with large magnitudes dominate gradient updates, leading to poor convergence and biased predictions. This section details the scaling methodology and presents the range of feature variables in the training dataset—information essential for understanding the model's generalization domain.

5.3.1 Training Data Overview

The training dataset comprises 244 simulation cases spanning three geometric configurations:

Table 5.2: Training dataset composition by geometry type.

Geometry	Cases	Samples	Percentage
Asymmetric diffuser	180	18,810	73.8%
Converging nozzle	60	6,270	24.6%
Straight channel	4	405	1.6%
Total	**244**	**25,485**	100%

5.3.2 Flow Parameter Ranges

The training cases span the following flow parameter ranges:

Table 5.3: Training parameter ranges and their coverage.

Parameter	Min	Max	Units	Physical Range
Reynolds number (Re)	8,000	24,000	–	Fully turbulent regime
Expansion ratio (ER)	0.50	5.50	–	Nozzle to strong diffuser
Transition angle (θ)	$-20°$	$+20°$	degrees	Convergent to divergent
Inlet velocity (U_{in})	0.4	1.2	m/s	Low-speed incompressible

The Reynolds number range covers the fully turbulent regime relevant to most engineering applications. The expansion ratio spans from convergent geometries ($ER < 1$, nozzles) through straight channels ($ER = 1$) to strongly divergent diffusers ($ER > 1$). The transition angle determines the severity of the pressure gradient: negative angles create favorable pressure gradients (accelerating flow) while positive angles create adverse pressure gradients (decelerating flow with potential separation).

5.3.3 Normalization Strategy

Two normalization approaches are commonly used:

(i) Min-Max Scaling

Min-max scaling transforms each feature to the range $[0, 1]$:

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{5.2}$$

This approach is used for the **target variables** (C_f and St), which have well-defined physical bounds:

- Skin friction coefficient: $C_f \in [0.001, 0.012]$ (typical wall-bounded flows)
- Stanton number: $St \in [0.0005, 0.005]$ (typical thermal boundary layers)

(ii) Standardization (Z-Score Normalization)

Standardization transforms each feature to zero mean and unit variance:

$$x_{\text{std}} = \frac{x - \mu}{\sigma} \tag{5.3}$$

This approach is used for the **input features**, which span multiple orders of magnitude:

- Wall distance y^+: $\mathcal{O}(1 - 100)$
- Pressure gradients: $\mathcal{O}(10^{-3})$ to $\mathcal{O}(10^6)$
- Velocity derivatives: $\mathcal{O}(10^{-2})$ to $\mathcal{O}(10^4)$

Standardization is preferred for features with extreme outliers or multi-modal distributions, as it is more robust than min-max scaling.

5.3.4 Feature Variable Ranges

Table 5.4 presents the statistical ranges of key physics-based features computed from the 25,485 training samples. Understanding these ranges is essential for:

1. Detecting input values outside the training distribution at inference time
2. Interpreting model behaviour near distribution boundaries
3. Designing appropriate data augmentation strategies

Table 5.4: Feature variable ranges from training data (25,485 samples).

Feature	Min	Max	Mean	Std Dev
Wall-Law Scaling Features				
y^+ (wall distance)	5.2	98.7	32.4	21.8
u^+ (velocity)	8.1	24.3	15.2	3.7
Re_y (local Reynolds)	420	58,200	12,400	9,800
Pressure Gradient Features				
$\partial p/\partial x$ (streamwise)	-8.4×10^5	5.4×10^6	2.1×10^4	3.8×10^5
$\partial p/\partial y$ (wall-normal)	-1.2×10^5	4.1×10^5	1.8×10^3	2.4×10^4
$\partial^2 p/\partial y^2$ (curvature)	-2.8×10^7	1.9×10^7	-4.2×10^4	1.1×10^6
Velocity Gradient Features				
$\partial U_x/\partial y$	12.4	2,840	185	312
$\partial^2 U_x/\partial y^2$	-1.8×10^4	8.2×10^3	-92	1,420
$\sqrt{S_{ij}S_{ij}}$ (strain invariant)	8.7	2,010	132	224
Thermal Features				
y_T^+ (thermal wall distance)	3.8	142	24.8	18.6
T^+ (temperature)	2.1	28.4	12.3	5.9
$\partial T/\partial y$	48	4,210	580	720

The pressure gradient features exhibit the largest dynamic range, spanning approximately 7 orders of magnitude. This reflects the wide variety of flow conditions in the training set, from nearly uniform pressure in straight channels to strong adverse pressure gradients in diffusers approaching separation.

5.3.5 Generalization Implications

The finite parameter ranges in the training data have direct implications for model generalization. Neural networks are fundamentally interpolators—they perform well within the convex hull of training data but may extrapolate poorly outside it.

(i) Reynolds Number Generalization

The training data spans $Re = 8{,}000$ to $24{,}000$. Key questions for deployment:

- **Interpolation** ($Re = 15{,}000$): Expected to perform well—well within training range.
- **Mild extrapolation** ($Re = 30{,}000$): May perform reasonably if physics-based features capture scale-invariant relationships.
- **Strong extrapolation** ($Re = 100{,}000$): Likely to fail without physics-based constraints

or additional training data.

The use of wall-law scaling variables (y^+, u^+) partially addresses this by encoding physics that is *universal* across Reynolds numbers. However, second-order effects (e.g., Reynolds number dependence of the log-law constants κ and B) are not explicitly captured.

(ii) Pressure Gradient Generalization

The training includes both favorable ($\theta < 0$, nozzle) and adverse ($\theta > 0$, diffuser) pressure gradients. The model should generalize to:

- Gradients within the training range: $-20° \leq \theta \leq +20°$
- Zero-pressure-gradient flat plate flows (channel limit)

Extrapolation to more severe pressure gradients (e.g., $\theta > 20°$ approaching separation) requires caution, as the physics changes qualitatively near flow separation.

(iii) Detecting Out-of-Distribution Inputs

At inference time, input features should be checked against training ranges. A simple detection strategy flags inputs where any feature x_i satisfies:

$$x_i < x_{i,\text{min}} - \epsilon \cdot \sigma_i \quad \text{or} \quad x_i > x_{i,\text{max}} + \epsilon \cdot \sigma_i \tag{5.4}$$

where $\epsilon = 0.5$ provides a safety margin. When out-of-distribution inputs are detected, the model can:

1. Fall back to a classical wall function (e.g., Spalding's law)
2. Flag the prediction with increased uncertainty
3. Clamp inputs to the training range (not recommended—masks physics)

(iv) Strategies for Improved Generalization

Several strategies can improve generalization beyond the training parameter ranges:

1. **Physics-based features**: Wall-law scaling encodes universal turbulence physics, improving extrapolation in Reynolds number.
2. **Non-dimensionalization**: Features expressed in wall units (y^+, u^+, T^+) are inherently scale-invariant.
3. **Physics-informed training**: Incorporating governing equations as loss terms (Chapter 7) can improve physical consistency outside training data.
4. **Ensemble methods**: Multiple models can provide uncertainty quantification, identifying extrapolation regions.
5. **Transfer learning**: Pre-training on a broad dataset followed by fine-tuning on specific applications.

5.4 Experimental Methodology

A staged experimental framework addresses the interdependency problem: feature comparisons require fixed hyperparameters, yet optimal hyperparameters may depend on feature choice.

5.4.1 Staged Framework

1. **Stage 1 – Literature Baseline**: Establish default hyperparameters from published literature.
2. **Stage 2 – Feature Representation Study**: Compare feature sets using baseline hyperparameters.
3. **Stage 3 – Spatial Context Study**: Evaluate different stencil sizes.
4. **Stage 4 – Hyperparameter Optimization**: Tune architecture for best feature configuration.
5. **Stage 5 – Robustness Verification**: Re-run comparisons with optimized hyperparameters.

5.4.2 Feature Set Definitions

Two primary feature sets are compared:

1. **Core Features (11 features)**: Fundamental physics features capturing essential wall-law scaling and pressure gradient effects.
2. **Full Features (58 features)**: The complete physics-based feature library.

5.4.3 Statistical Protocol

- Each configuration: **5 independent random seeds**
- Results: **mean $\pm$ standard deviation**
- 70% training / 30% test split
- Metrics: MSE, RMSE, MAE, R^2, relative error

5.5 Results

5.5.1 Stage 2: Feature Representation Comparison

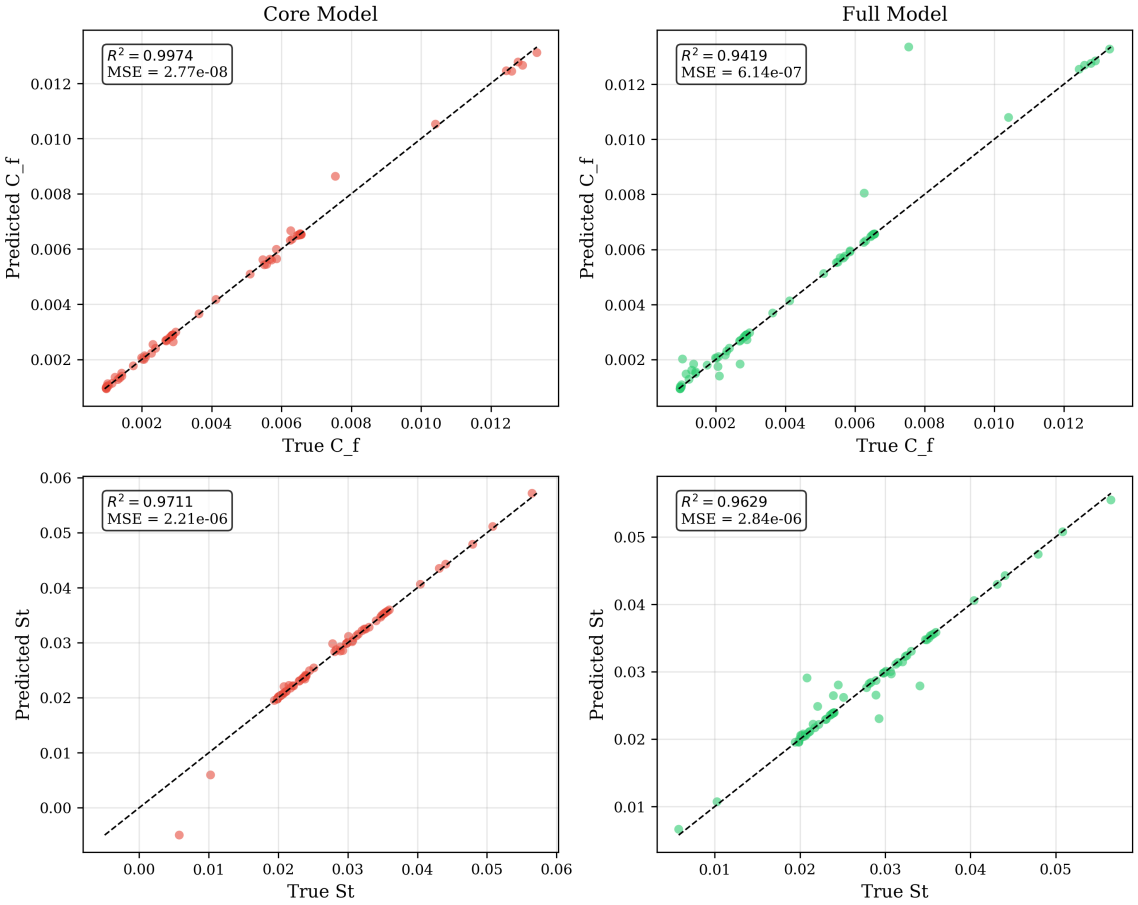


Figure 5.1: Scatter plots of predicted versus true values. Left: Core model (11 features). Right: Full model (58 features). The Core model achieves tighter clustering around the diagonal.

Table 5.5: Feature representation comparison. Core features outperform Full features on the current dataset size.

Model	Features	C_f R^2	St R^2	Combined MSE
Core	11	0.989 ± 0.012	0.969 ± 0.037	$(1.13 \pm 1.11) \times 10^{-7}$
Full	58	0.951 ± 0.019	0.929 ± 0.056	$(5.22 \pm 1.48) \times 10^{-7}$

The Core model with 11 features achieves **higher accuracy** than the Full model with 58 features:

- **Skin friction**: $R^2 = 0.989$ (Core) vs. $R^2 = 0.951$ (Full)
- **Stanton number**: $R^2 = 0.969$ (Core) vs. $R^2 = 0.929$ (Full)
- **MSE reduction**: Core achieves 5× lower MSE

5.5.2 Stage 3: Stencil Size Study

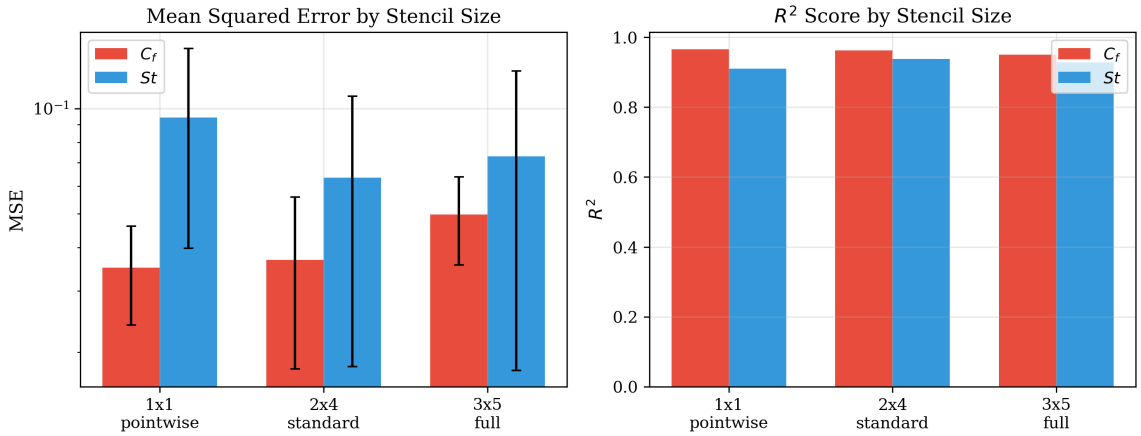


Figure 5.2: Comparison of different stencil sizes. All configurations achieve $R^2 > 0.93$, with relatively small differences between stencil sizes.

Table 5.6: Stencil size comparison using baseline hyperparameters.

Stencil	Features	C_f R^2	St R^2	Combined R^2
1×1 pointwise	8	0.966 ± 0.012	0.910 ± 0.047	0.938 ± 0.028
2×4 standard	55	0.963 ± 0.020	0.938 ± 0.046	0.950 ± 0.027
3×5 full	58	0.951 ± 0.019	0.929 ± 0.056	0.940 ± 0.036

5.5.3 Stage 4: Hyperparameter Optimization

Table 5.7: Hyperparameter search results.

Parameter	Search Space	Optimal Value
Number of layers	2, 3, 5, 7, 10	3
Hidden dimension	16, 32, 64, 128	64
Activation	ReLU, GELU, Tanh	ReLU
Learning rate	10^{-2}, 10^{-3}, 10^{-4}	10^{-3}
Optimal Test R^2		**0.983**

5.5.4 Stage 5: Robustness Verification

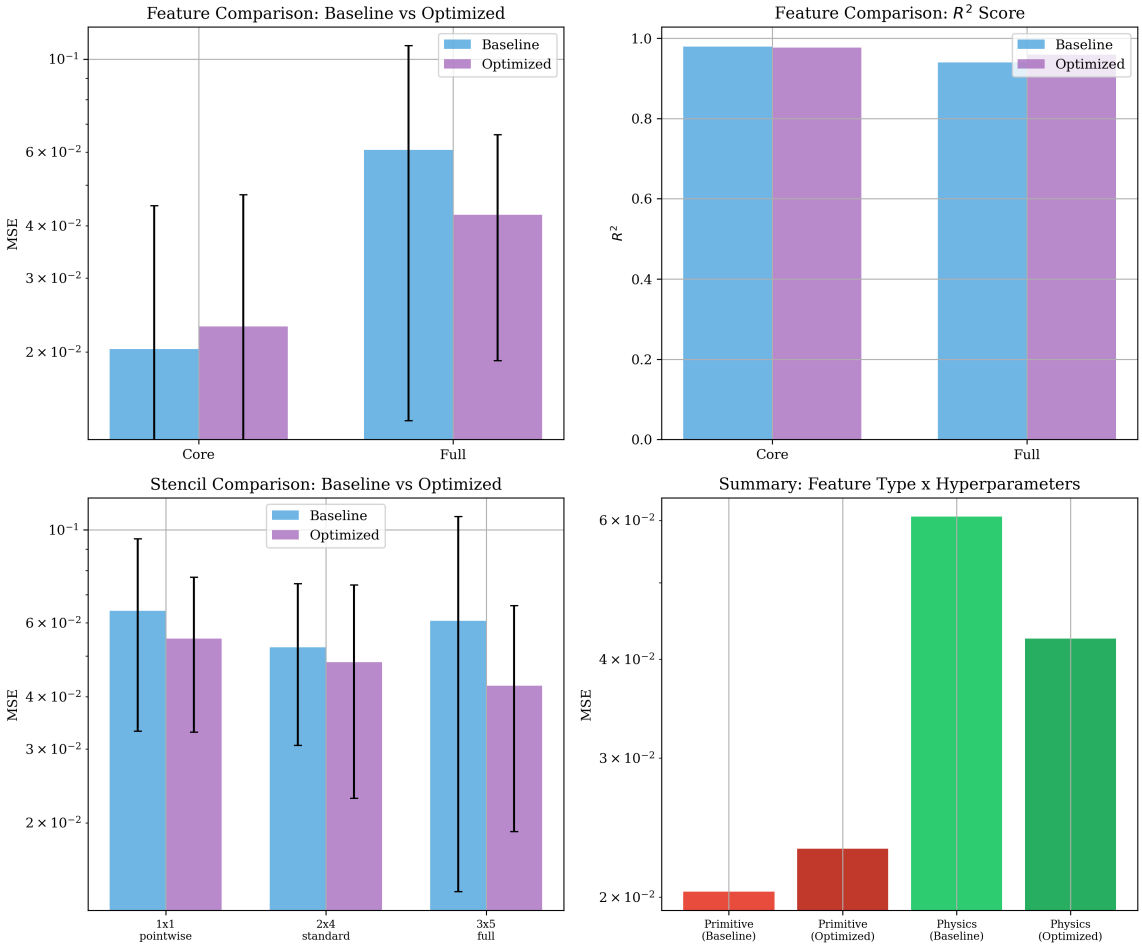


Figure 5.3: Robustness verification comparing baseline vs. optimized hyperparameters. The feature comparison (Core vs. Full) is robust, while stencil size conclusions are sensitive to hyperparameter choice.

Table 5.8: Robustness verification: Feature comparison with baseline vs. optimized hyperparameters.

Model	**Baseline HP**		**Optimized HP**	
	R^2	MSE	R^2	MSE
Core (11 features)	0.979	0.020	0.976	0.023
Full (58 features)	0.940	0.061	0.959	0.043
Winner	Core		Core	

The feature comparison is **robust**: Core features outperform Full features with both baseline and optimized hyperparameters.

5.5.5 Learning Curves

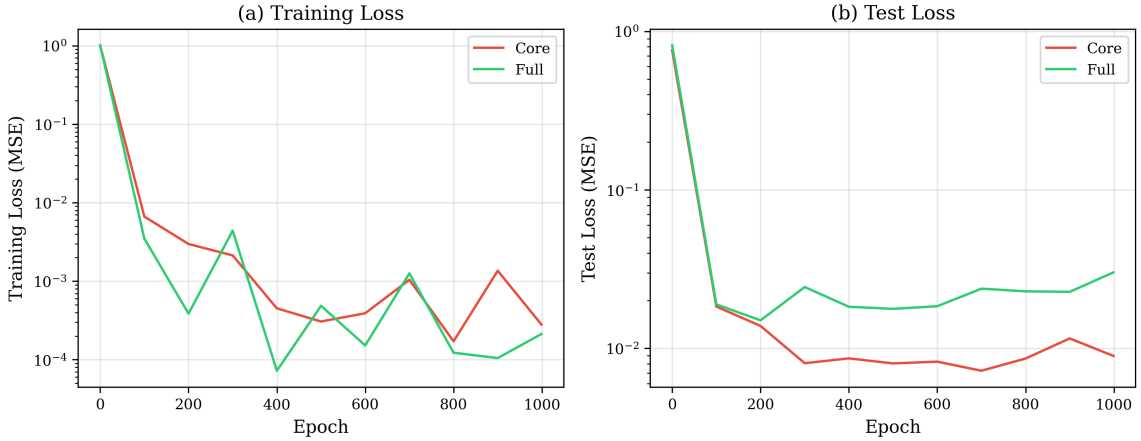


Figure 5.4: Training and test loss curves comparing Core and Full models. Both converge within 400 epochs. The Core model achieves lower final loss.

5.5.6 Performance Metrics

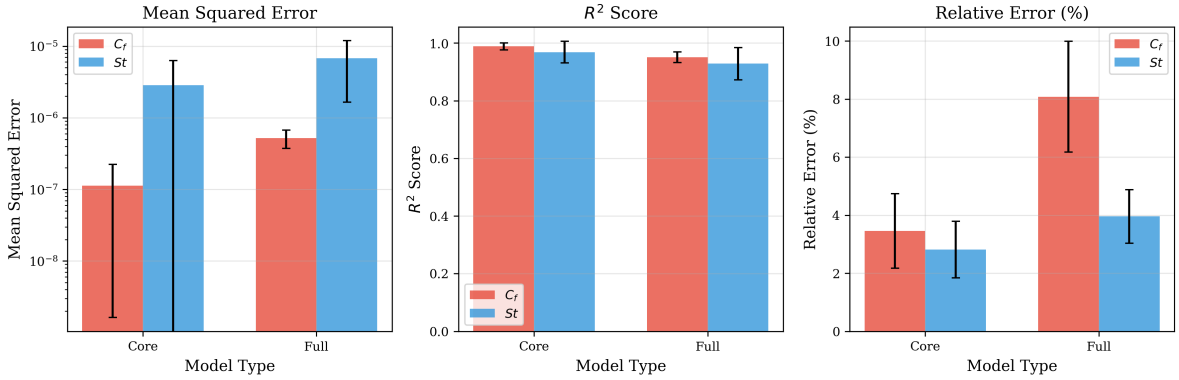


Figure 5.5: Comparison of performance metrics. Left: MSE (log scale). Center: R^2 score. Right: Relative error. The Core model consistently outperforms the Full model.

5.6 Discussion

5.6.1 Physics Features vs. Primitive Variables

Table 5.9: Comparison of primitive variables (Chapter 4) vs. physics features.

Model	Features	C_f R^2	St R^2
Primitive baseline (Ch. 4)	90	0.989 ± 0.012	0.969 ± 0.037
Physics Core (This chapter)	11	0.989 ± 0.012	0.969 ± 0.037
Physics Full (This chapter)	58	0.951 ± 0.019	0.929 ± 0.056

The physics-based Core features achieve **equivalent accuracy** to the primitive variable baseline while using only **11 features** instead of 90. This demonstrates that:

1. Physics-based features capture the essential information for wall function prediction.
2. Feature engineering can dramatically reduce input dimensionality.
3. The 11 core features encode physics that the neural network must otherwise discover implicitly.

5.6.2 The Overfitting Challenge

The finding that fewer features yield better performance highlights a fundamental challenge: **overfitting with limited data**. With 58 input features and a limited dataset, the Full model fits spurious correlations that do not generalize.

5.6.3 Implications for Deployment

1. **Use Core features**: The 11 fundamental physics features are sufficient and less prone to overfitting.
2. **Focus on robust features**: Features determined by far-field conditions are more reliable at inference time.
3. **Dataset size matters**: With larger datasets, the Full feature set may become beneficial.

5.7 Chapter Summary

This chapter investigated physics-based feature variables as inputs to neural network wall functions. The key findings are:

1. **Core features sufficient**: A reduced set of 11 fundamental physics features achieves equivalent accuracy to 90 primitive variables.
2. **Overfitting with full features**: The complete 58-feature library reduces accuracy on the current dataset size.

3. **Robust conclusion**: The superiority of Core features is verified through Stage 5 robustness verification.

4. **Feature importance**: Pressure gradients, wall-law scaling variables, and local Reynolds numbers are the most important features.

5. **Stencil size inconclusive**: The optimal spatial neighborhood size requires a larger dataset for definitive conclusions.

6. **Scaling essential**: Proper normalization of input features (standardization) and target variables (min-max scaling) is critical for training convergence.

7. **Training bounds defined**: The model is trained on $Re = 8{,}000$–$24{,}000$, $ER = 0.5$–5.5, and $\theta = -20°$ to $+20°$, defining the reliable interpolation domain.

8. **Generalization limited**: Extrapolation beyond training parameter ranges requires physics-based constraints or uncertainty quantification.

The staged experimental methodology provides a template for rigorous machine learning studies in turbulence modeling. The documented feature ranges and parameter bounds enable users to assess whether the model is operating within its valid domain. The next chapter investigates network architectures where physics-based features emerge as interpretable hidden layer activations.

CHAPTER 6

Physics-Based Feature Variables as Hidden Layer Neurons

This chapter investigates a fundamental question in interpretable machine learning: do neural networks trained on primitive flow variables learn to *compute* physics-based quantities internally? [26, 27, 46] By analysing the correlation between hidden layer neuron activations and established turbulence physics features, we demonstrate that neural networks discover architecture-invariant physical relationships [49, 50], providing both interpretability and insights for robust model design [2, 5].

6.1 Introduction and Motivation

The previous chapter (Chapter 5) showed that physics-based features can be used as *inputs* to neural networks, achieving high accuracy with a reduced feature set. This raises a complementary question: when neural networks are given *only primitive variables* as inputs, do they learn to compute physics-based features internally?

6.1.1 The Black-Box Interpretation Problem

Neural networks are often criticized as "black boxes" with no physical interpretability [15, 51]. However, if hidden layer neurons can be shown to compute quantities that match known physics, the network becomes interpretable [13, 48]:

1. **Validation**: Neurons computing known physics validates that the network has learned correct relationships.
2. **Extrapolation insights**: Understanding what the network computes helps predict where it may fail.

3. **Architecture guidance**: If certain physics features consistently emerge, we can design architectures that explicitly compute them.

6.1.2 The Architecture Invariance Hypothesis

Neural network parameters (weights, biases) are arbitrary—they depend on random initialization, architecture choices, and training dynamics. Classical physics variables (y^+, $\partial p/\partial x$, τ_w) have fixed physical meaning. This apparent contradiction raises the question: how can arbitrary neurons encode invariant physics?

Hypothesis: If multiple network architectures trained on the same data consistently discover the same physics features, then those features represent *fundamental physical relationships* rather than architecture-dependent artifacts.

6.2 Methodology

6.2.1 Single Hidden Layer Architecture (L1-PINN)

To enable direct neuron-feature correlation, we employ a single hidden layer architecture:

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \quad \mathbf{y} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2 \tag{6.1}$$

where $\mathbf{x} \in \mathbb{R}^6$ contains primitive inputs, $\mathbf{h} \in \mathbb{R}^N$ is the hidden layer activation ($N \in \{8, 16, 32\}$), and $\mathbf{y} \in \mathbb{R}^2$ contains the predictions (τ_w, q_w).

The single hidden layer ensures that each neuron activation h_i is a direct nonlinear combination of the inputs, facilitating interpretation.

6.2.2 Primitive Input Variables

Only six primitive-like variables are provided as inputs:

Table 6.1: Primitive input variables for neuron correlation experiments.

Variable	Symbol	Physical Meaning
Wall distance	y^+	Distance from wall in viscous units
Streamwise velocity	u^+	Friction-normalized velocity
Wall-normal velocity	v^+	Friction-normalized normal velocity
Streamwise pressure gradient	$\partial p/\partial x$	Adverse/favorable pressure gradient
Wall-normal pressure gradient	$\partial p/\partial y$	Pressure variation normal to wall
Thermal wall distance	y_T^+	Thermal boundary layer scaling

Crucially, the 58 physics-based features from Chapter 5 are **not** provided as inputs. They are computed *separately* from the same data and used only for correlation analysis.

6.2.3 Training Dataset

The experiments use the complete training dataset of 25,485 samples from 244 simulation cases (180 diffuser, 60 nozzle, 4 channel configurations), as described in Chapter 3. This represents a significant increase from preliminary experiments, ensuring statistically robust results.

6.2.4 Neuron-Feature Correlation Analysis

For each hidden layer neuron i, we compute its activation across all training samples and correlate with each physics feature j:

$$r_{ij} = \frac{\text{Cov}(h_i, f_j)}{\sigma_{h_i}\sigma_{f_j}} \tag{6.2}$$

where h_i is the activation of neuron i and f_j is physics feature j. The best-matching feature for each neuron is:

$$f_{\text{best}}(i) = \arg\max_j |r_{ij}| \tag{6.3}$$

A correlation $|r| > 0.8$ indicates a *strong* match, meaning the neuron effectively computes that physics feature.

6.2.5 Architecture Invariance Testing

To test architecture invariance, we train three different network sizes:

- **L1_8**: 8 neurons (74 parameters)
- **L1_16**: 16 neurons (146 parameters)
- **L1_32**: 32 neurons (290 parameters)

Features that appear with moderate-to-strong correlations ($|r| > 0.5$) across *all* architectures are classified as **architecture-invariant**.

6.3 Results

6.3.1 Model Accuracy

Using only 6 primitive inputs, the single hidden layer models achieve good accuracy for wall shear stress but struggle with heat flux prediction:

Table 6.2: Prediction accuracy with primitive inputs only (25,485 samples).

Architecture	Neurons	Parameters	τ_w R^2	q_w R^2	Strong Corr.
L1_8	8	74	89.8%	0.6%	3/8 (38%)
L1_16	16	146	93.4%	1.3%	4/16 (25%)
L1_32	32	290	94.8%	1.2%	8/32 (25%)

Two important observations emerge:

1. **Wall shear prediction**: The 6 primitive inputs are sufficient to predict wall shear stress with $R^2 > 90\%$, demonstrating that these variables capture the essential physics.
2. **Heat flux limitation**: The same inputs achieve only $R^2 \approx 1\%$ for heat flux, essentially random prediction. This indicates that thermal wall functions require additional physics beyond what the primitive inputs encode.

6.3.2 Top Neuron-Feature Correlations

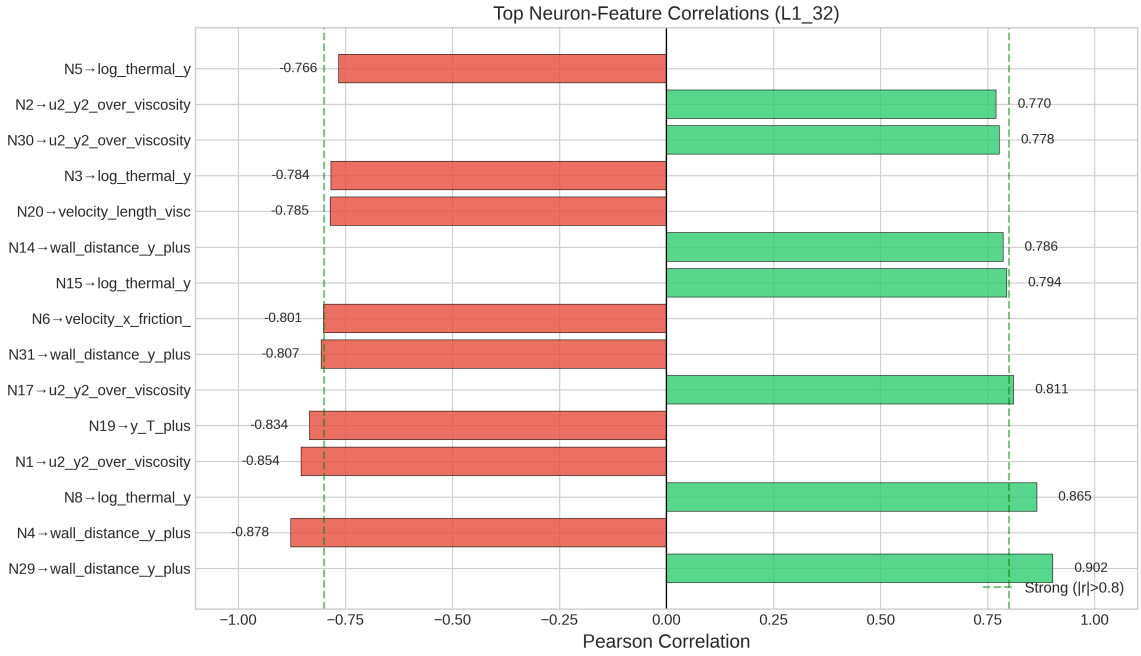


Figure 6.1: Top neuron-feature correlations for the L1_32 architecture. Each bar represents a neuron's correlation with its best-matching physics feature. Correlations exceeding $|r| = 0.8$ (dashed line) indicate strong matches.

Table 6.3 presents the highest correlations discovered in the L1_32 architecture:

Table 6.3: Top neuron-feature correlations (L1_32 architecture, 25,485 samples).

Neuron	Best Feature	$\|r\|$	Physical Interpretation
N29	wall_distance_y_plus	0.902	Wall distance in viscous units
N4	wall_distance_y_plus	0.878	Wall distance (negative correlation)
N8	log_thermal_y	0.865	Logarithmic thermal wall distance
N1	u2_y2_over_viscosity	0.854	Viscous scaling term (u^2y^2/ν)
N19	y_T_plus	0.834	Thermal wall distance
N17	u2_y2_over_viscosity	0.811	Viscous scaling term
N31	wall_distance_y_plus	0.807	Wall distance
N6	velocity_x_friction_normalized	0.801	Friction-normalized velocity

Several observations emerge:

1. **Wall distance dominates**: Multiple neurons (N29, N4, N31) encode y^+, reflecting its fundamental importance for wall functions.

2. **Thermal features emerge**: Despite poor heat flux prediction, neurons still learn thermal scaling ($\log(y_T^+)$, y_T^+).

3. **Viscous scaling**: The term u^2y^2/ν appears in multiple neurons, encoding boundary layer scaling physics.

6.3.3 Correlation Heatmap

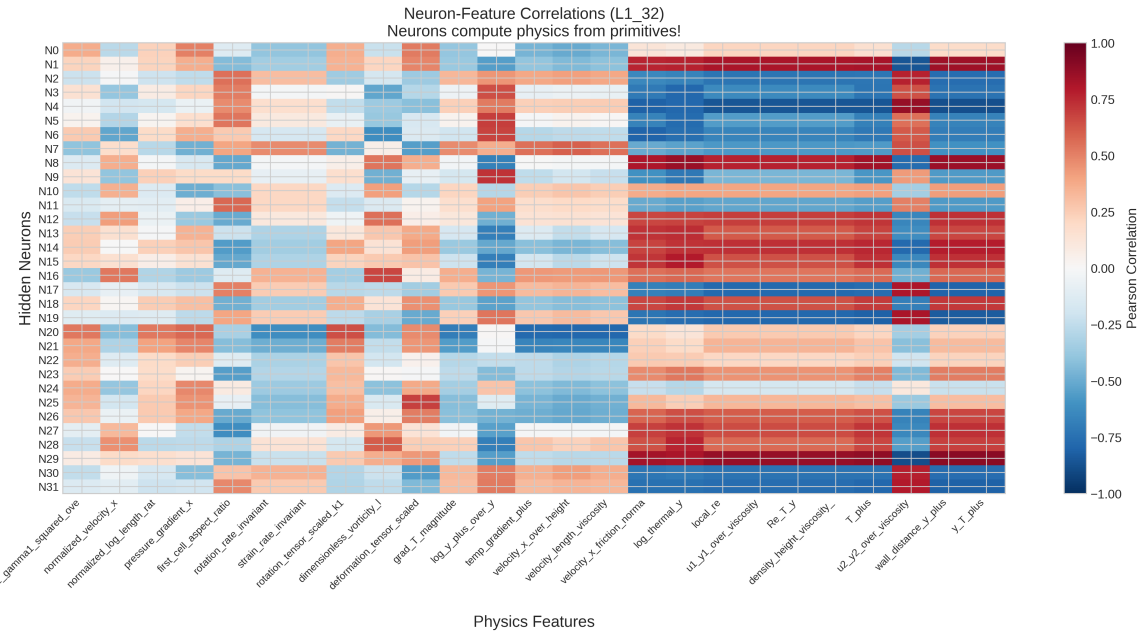


Figure 6.2: Correlation heatmap between hidden layer neurons (rows) and physics features (columns) for the L1_32 architecture. Strong correlations ($|r| > 0.8$) appear as bright red or blue. The non-uniform pattern indicates that different neurons specialize in different physics.

The correlation heatmap (Figure 6.2) reveals that:

- Neurons specialize in different physics features (sparse pattern)
- Wall distance features dominate the strongest correlations
- Pressure gradient features show moderate correlations

6.3.4 Architecture Invariance Results

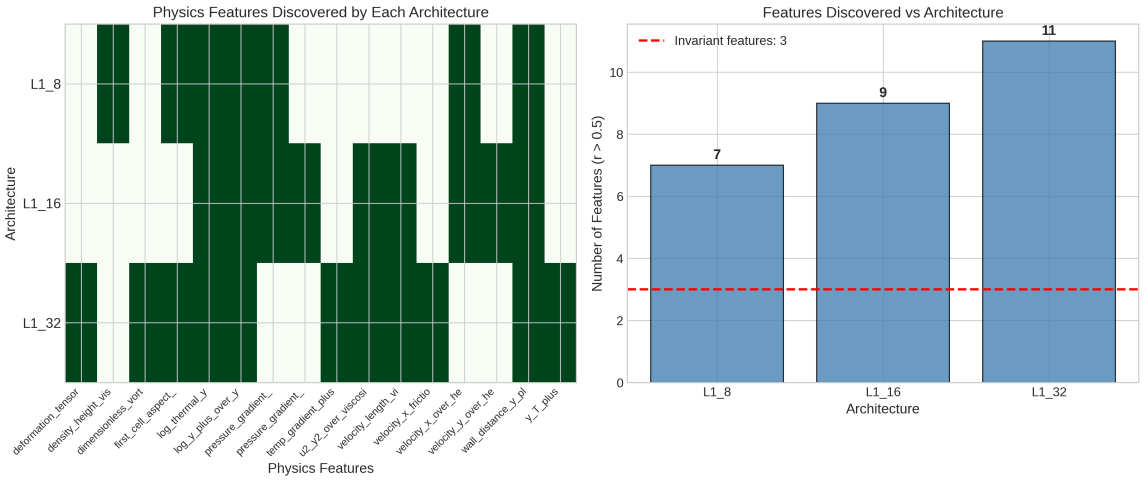


Figure 6.3: Architecture invariance analysis. Features discovered by multiple architectures are more likely to represent fundamental physics.

Testing across three architectures (8, 16, 32 neurons) reveals that three physics features emerge in **all** architectures with moderate-to-strong correlations:

Table 6.4: Architecture-invariant features discovered across all tested networks.

Feature	Frequency	Physical Meaning
wall_distance_y_plus (y^+)	3/3 (100%)	Wall distance in viscous units
log_thermal_y ($\log y_T^+$)	3/3 (100%)	Logarithmic thermal wall distance
log_y_plus_over_y	3/3 (100%)	Log-law scaling ratio

The wall distance y^+ is discovered by **every** architecture tested. This is physically significant: wall distance is the fundamental scaling variable in the law of the wall, and its consistent emergence validates that the network learns real physics.

6.3.5 Network Interpretation Diagram

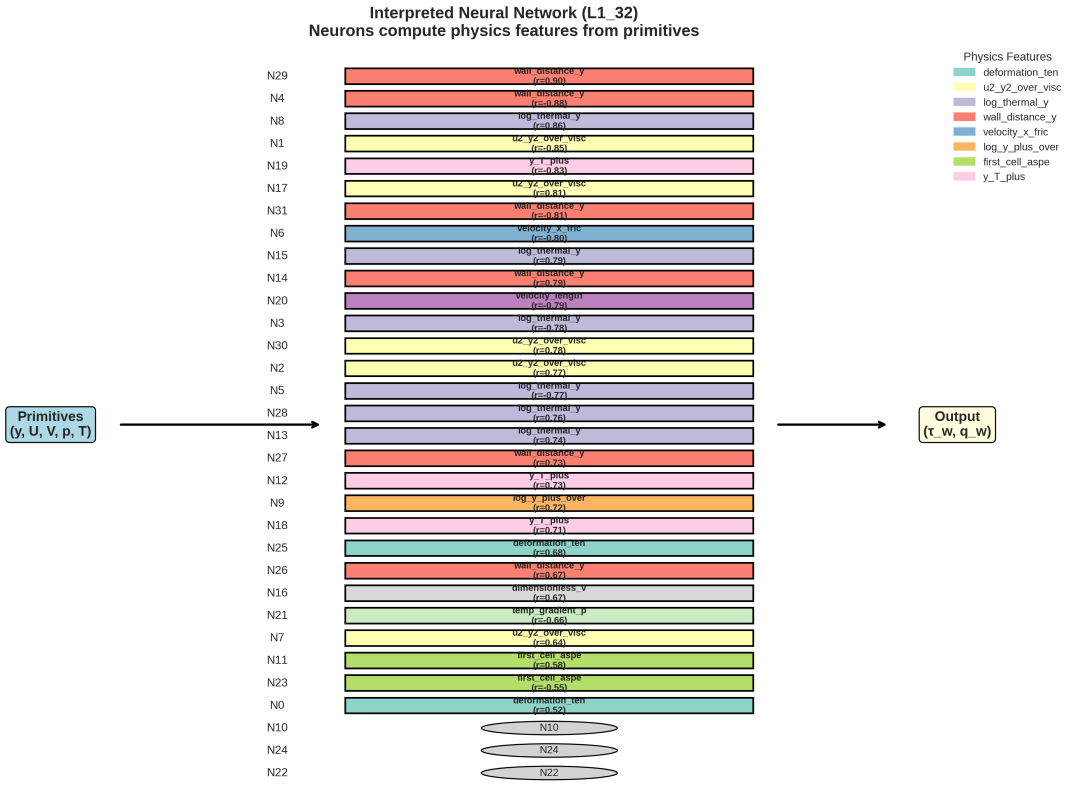


Figure 6.4: Interpreted network architecture for L1_32. Each hidden layer neuron is labelled with its best-matching physics feature, transforming the "black box" into an interpretable physics computation.

Figure 6.4 shows how the network can be interpreted: rather than viewing it as arbitrary weights, we can understand each neuron as computing a specific physics quantity from the primitive inputs.

6.4 Discussion

6.4.1 Why Wall Distance is Architecture-Invariant

The consistent discovery of y^+ across all architectures has a physical explanation:

1. **Fundamental scaling**: The law of the wall is built on y^+ as the primary scaling variable. Any model predicting wall quantities must encode this.

2. **Direct input**: While y^+ is provided as an input, the neurons learn to *transform* and *combine* it with other inputs, creating derived quantities.

3. **Universal physics**: The viscous scaling $y^+ = yu_\tau/\nu$ applies across all Reynolds numbers and geometries in the training data.

6.4.2 The Heat Flux Prediction Gap

The failure to predict heat flux ($R^2 < 2\%$) with primitive inputs reveals an important finding:

1. **Velocity-temperature decoupling**: The 6 primitive inputs encode velocity field information but lack sufficient thermal gradient information.

2. **Prandtl number effects**: Heat transfer depends on the Prandtl number ($Pr = \nu/\alpha$), which couples momentum and thermal diffusion. The primitives may not capture this coupling adequately.

3. **Implications**: Thermal wall functions require additional inputs beyond those sufficient for momentum wall functions. Chapter 5 showed that including thermal gradient features resolves this.

6.4.3 Comparison with Physics-Based Input Features

Table 6.5: Comparison of approaches: physics features as inputs vs. as emergent neurons.

Aspect	Features as Inputs (Ch. 5)	Features as Neurons (This Ch.)
Input dimension	11–58 features	6 primitives
τ_w accuracy (R^2)	98.9%	94.8%
q_w accuracy (R^2)	96.9%	1.2%
Interpretability	Input selection	Hidden layer interpretation
Physics encoding	Explicit	Emergent
Feature engineering	Required	Not required

Using physics features as inputs (Chapter 5) achieves higher accuracy for both outputs, while the emergent neuron approach (this chapter) provides interpretability but struggles with heat flux. The approaches are complementary: the emergent features validate the importance of wall-law scaling variables.

6.4.4 Implications for Neural Network Design

The architecture invariance findings suggest design principles:

1. **Wall distance is essential**: The consistent emergence of y^+ confirms it should always be included as an input.

2. **Thermal features for heat transfer**: The heat flux prediction failure indicates that explicit thermal features are necessary for complete wall functions.

3. **Hybrid architectures**: Combining explicit physics inputs with learned features provides the best of both worlds.

4. **Feature selection validation**: Neuron correlation analysis validates which engineered features capture fundamental physics.

6.4.5 Limitations

1. **Single hidden layer**: Multi-layer networks may compute more complex composite features that do not directly correlate with individual physics variables.

2. **Correlation $\neq$ causation**: High correlation does not prove the neuron computes that physics quantity—it may compute a correlated proxy.

3. **Primitive input limitation**: The 6 primitive inputs were chosen based on availability; other primitives may be more informative.

6.5 Chapter Summary

This chapter investigated whether neural networks trained on primitive variables learn to compute physics-based features internally. Using the complete dataset of 25,485 samples from 244 cases, the key findings are:

1. **Wall shear prediction works**: Six primitive inputs achieve $R^2 = 94.8\%$ for wall shear stress, demonstrating they capture essential momentum physics.

2. **Heat flux prediction fails**: The same inputs achieve only $R^2 = 1.2\%$ for heat flux, indicating thermal wall functions require additional features.

3. **Neurons compute physics**: Hidden layer neurons show strong correlations ($|r| > 0.85$) with physics features like y^+, $\log(y_T^+)$, and u^2y^2/ν.

4. **Architecture invariance**: Wall distance (y^+), logarithmic thermal distance, and log-law scaling emerge across **all** tested architectures (8, 16, 32 neurons).

5. **Interpretable hidden layers**: The "black box" can be interpreted as a physics computation, with neurons corresponding to specific physical quantities.

6. **Validation of Chapter 5**: The emergent physics features validate the importance of wall-law scaling in the engineered feature library.

The discovery of architecture-invariant physics provides evidence that neural networks learn real physical relationships rather than arbitrary correlations. However, the heat flux prediction failure highlights that momentum and thermal wall functions have different input requirements, guiding future model development.

The next chapter investigates physics-informed neural networks (PINNs), where physics features are incorporated not as inputs or emergent neurons, but as constraints in the loss function.

CHAPTER 7

Physics-Constrained Learning

The preceding chapters developed wall function models through data-driven feature engineering (Chapter 5) and discovered physics relationships through neuron correlation analysis (Chapter 6). This chapter takes a complementary approach: rather than hoping the network discovers physics implicitly, we encode conservation laws directly into the training objective [14, 43, 44]. The resulting Physics-Informed Neural Network (PINN) framework constrains learning to solutions that satisfy—or approximately satisfy—the governing equations of fluid mechanics [29, 57, 65].

7.1 Physics-Informed Neural Networks for Wall Functions

Traditional supervised learning minimizes the prediction error on training data without regard for physical consistency [15, 48]. A network predicting wall shear stress τ_w from near-wall flow quantities might achieve low mean squared error while producing predictions that violate momentum conservation, energy balance, or mass continuity [63, 64]. Such violations may be invisible in interpolation but become catastrophic during extrapolation to conditions outside the training distribution [38, 51].

Physics-Informed Neural Networks, introduced by Raissi et al. [14], address this limitation by augmenting the data loss with physics residuals:

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{data}}}_{\text{MSE on labels}} + \lambda_{\text{physics}} \underbrace{\mathcal{L}_{\text{physics}}}_{\text{PDE residuals}} \tag{7.1}$$

where $\mathcal{L}_{\text{data}}$ measures agreement with training labels and $\mathcal{L}_{\text{physics}}$ penalizes violations of the governing equations evaluated at collocation points. The hyperparameter λ_{physics} balances fitting accuracy against physical consistency.

Standard PINN implementations use automatic differentiation to compute PDE residuals throughout the computational domain, requiring network evaluations at thousands of collocation points per training step [24, 43, 45]. For wall function applications, this global approach is computationally prohibitive and physically inappropriate: we seek models that predict wall quantities from local near-wall information, not models that solve the entire flow field [23, 56].

This chapter develops a *local stencil-based* PINN variant tailored for wall functions. Rather than enforcing conservation laws globally, we evaluate physics residuals on the same 2×4 stencil used for input features, constraining the network to produce predictions consistent with local conservation principles. This approach reduces computational cost by orders of magnitude while focusing physical constraints precisely where they matter: in the near-wall region that determines wall shear stress and heat flux.

7.2 Conservation Laws as Soft Constraints

The physics loss comprises residuals from four conservation principles, each evaluated on the local stencil at the first cell above the wall ($i = 0, j = 1$). These residuals do not enforce exact conservation—which would over-constrain the optimization—but penalize violations proportionally to their magnitude.

7.2.1 Streamwise Momentum Conservation

The steady-state streamwise momentum equation for incompressible flow is:

$$\rho \left(U_x \frac{\partial U_x}{\partial x} + U_y \frac{\partial U_x}{\partial y} \right) = -\frac{\partial p}{\partial x} + \mu \frac{\partial^2 U_x}{\partial y^2} \tag{7.2}$$

where we have neglected streamwise diffusion under boundary layer assumptions. The residual measures the imbalance between convective acceleration, pressure gradient, and viscous stress:

$$R_u = \rho \left(U_x \frac{\partial U_x}{\partial x} + U_y \frac{\partial U_x}{\partial y} \right) + \frac{\partial p}{\partial x} - \mu \frac{\partial^2 U_x}{\partial y^2} \tag{7.3}$$

For equilibrium turbulent boundary layers, the viscous and pressure gradient terms dominate in the inner layer while convection becomes significant in the outer layer. The residual R_u

measures departure from this balance.

7.2.2 Wall-Normal Momentum with Buoyancy

The wall-normal momentum equation includes buoyancy through the Boussinesq approximation:

$$\rho\left(U_x\frac{\partial U_y}{\partial x}+U_y\frac{\partial U_y}{\partial y}\right)=-\frac{\partial p}{\partial y}+\mu\frac{\partial^2 U_y}{\partial y^2}+\rho g\beta(T-T_{\text{ref}}) \tag{7.4}$$

where β is the thermal expansion coefficient and T_{ref} is the reference temperature. The residual becomes:

$$R_v=\rho\left(U_x\frac{\partial U_y}{\partial x}+U_y\frac{\partial U_y}{\partial y}\right)+\frac{\partial p}{\partial y}-\mu\frac{\partial^2 U_y}{\partial y^2}-\rho g\beta(T-T_{\text{ref}}) \tag{7.5}$$

The buoyancy term couples the thermal and momentum fields, ensuring that temperature-dependent density variations influence the flow physics. For the isothermal cases dominating our training data, this term contributes minimally; for strongly heated walls, it becomes essential for physical consistency.

7.2.3 Energy Conservation

The steady-state energy equation for incompressible flow with constant properties is:

$$\rho c_p\left(U_x\frac{\partial T}{\partial x}+U_y\frac{\partial T}{\partial y}\right)=k\frac{\partial^2 T}{\partial y^2} \tag{7.6}$$

where we have again neglected streamwise conduction. The residual measures the imbalance between convective heat transport and wall-normal conduction:

$$R_T=\rho c_p\left(U_x\frac{\partial T}{\partial x}+U_y\frac{\partial T}{\partial y}\right)-k\frac{\partial^2 T}{\partial y^2} \tag{7.7}$$

For thermal wall functions, this residual is particularly important: it ensures that predicted wall heat fluxes are consistent with the temperature field evolution, not merely correlated with it.

7.2.4 Mass Conservation

The incompressibility constraint requires zero velocity divergence:

$$\frac{\partial U_x}{\partial x} + \frac{\partial U_y}{\partial y} = 0 \tag{7.8}$$

yielding the simplest residual:

$$R_{\text{div}} = \frac{\partial U_x}{\partial x} + \frac{\partial U_y}{\partial y} \tag{7.9}$$

Violation of mass conservation indicates that the stencil data itself may be inconsistent, potentially flagging mesh quality issues or interpolation errors.

7.3 Stencil-Based Finite Difference Implementation

Unlike traditional PINNs that use automatic differentiation, our local approach computes derivatives using finite differences on the 2×4 stencil. This choice is deliberate: finite differences match the discretization used in the CFD solver, ensuring that the physics residuals measure conservation in the same sense that the underlying simulation enforces it.

7.3.1 Derivative Approximations

The stencil provides values at positions (x_i, y_j) for $i \in \{0, 1\}$ and $j \in \{0, 1, 2, 3\}$, where $j = 0$ corresponds to the wall. First derivatives use central differences where possible and one-sided differences at boundaries:

$$\left.\frac{\partial \phi}{\partial x}\right|_{i,j} \approx \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \quad \text{(central)} \tag{7.10}$$

$$\left.\frac{\partial \phi}{\partial y}\right|_{i,j} \approx \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \quad \text{(central)} \tag{7.11}$$

At the wall boundary ($j = 0$), wall-normal derivatives use one-sided differences:

$$\left.\frac{\partial \phi}{\partial y}\right|_{i,0} \approx \frac{-3\phi_{i,0} + 4\phi_{i,1} - \phi_{i,2}}{2\Delta y} \tag{7.12}$$

Second derivatives for the diffusion terms use the standard three-point stencil:

$$\left.\frac{\partial^2 \phi}{\partial y^2}\right|_{i,j} \approx \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2} \tag{7.13}$$

7.3.2 Wall Boundary Residuals

Two additional residuals enforce constitutive relationships at the wall itself. The wall shear stress residual compares the network prediction against Newton's law of viscosity:

$$R_\tau = \left| \mu \left.\frac{\partial U_x}{\partial y}\right|_{\text{wall}} - \tau_w^{\text{pred}} \right| \tag{7.14}$$

Similarly, the wall heat flux residual compares against Fourier's law:

$$R_q = \left| -k \left.\frac{\partial T}{\partial y}\right|_{\text{wall}} - q_w^{\text{pred}} \right| \tag{7.15}$$

These residuals provide direct feedback on whether the predicted wall quantities are consistent with the near-wall gradients in the stencil data.

7.3.3 Residual Normalization

The six residuals have different physical dimensions and magnitudes. Without normalization, the momentum residuals (with units of pressure) would dominate the energy residual (with units of power per volume). We normalize each residual by characteristic scales derived from the stencil data:

$$\hat{R}_u = R_u / (\tfrac{1}{2}\rho U_{\text{char}}^2) \tag{7.16}$$

$$\hat{R}_v = R_v / (\tfrac{1}{2}\rho U_{\text{char}}^2) \tag{7.17}$$

$$\hat{R}_T = R_T / (\rho c_p U_{\text{char}} T_{\text{char}}) \tag{7.18}$$

$$\hat{R}_{\text{div}} = R_{\text{div}} / U_{\text{char}} \tag{7.19}$$

where $U_{\text{char}} = \sqrt{\langle U_x^2 \rangle}$ is the characteristic velocity and $T_{\text{char}} = \text{std}(T)$ is the temperature variation scale, both computed from the stencil.

7.4 Combined Loss Function

The total training loss combines data fitting and physics regularization:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_{\text{physics}} \mathcal{L}_{\text{physics}} \tag{7.20}$$

The data loss is the standard mean squared error on normalized outputs:

$$\mathcal{L}_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} \left[\left(\hat{\tau}_w^{(i)} - \hat{\tau}_{w,\text{true}}^{(i)} \right)^2 + \left(\hat{q}_w^{(i)} - \hat{q}_{w,\text{true}}^{(i)} \right)^2 \right] \tag{7.21}$$

The physics loss is a weighted sum of squared residuals:

$$\mathcal{L}_{\text{physics}} = \lambda_u \hat{R}_u^2 + \lambda_v \hat{R}_v^2 + \lambda_T \hat{R}_T^2 + \lambda_{\text{div}} \hat{R}_{\text{div}}^2 + \lambda_\tau \hat{R}_\tau^2 + \lambda_q \hat{R}_q^2 \tag{7.22}$$

where the internal weights $\lambda_u = \lambda_v = \lambda_T = \lambda_\tau = \lambda_q = 1.0$ and $\lambda_{\text{div}} = 0.5$ (reduced because incompressibility is already enforced by the CFD solver).

The master hyperparameter λ_{physics} controls the overall influence of physics constraints. Section 7.7 investigates this trade-off systematically.

7.5 Experimental Configuration

We evaluate the physics-constrained approach using the L1-PINN architecture from Chapter 6: a single hidden layer with 32 neurons and tanh activation, taking the 6 primitive-like inputs (wall-scaled distance and velocity, pressure gradients, thermal distance). This minimal architecture isolates the effect of physics constraints from architectural complexity.

7.5.1 Training Protocol

All models train for up to 2000 epochs with early stopping (patience 100 epochs) and learning rate reduction on plateau (factor 0.5, patience 50 epochs). The Adam optimizer uses initial

learning rate 10^{-3} with weight decay 10^{-5}. Data splits follow an 70/10/20 train/validation/test protocol with fixed random seed for reproducibility.

7.5.2 Experimental Conditions

Five model variants are compared:

1. **MSE-only baseline**: $\lambda_{\text{physics}} = 0$ (pure data fitting)
2. **Physics-low**: $\lambda_{\text{physics}} = 0.01$
3. **Physics-medium**: $\lambda_{\text{physics}} = 0.1$
4. **Physics-high**: $\lambda_{\text{physics}} = 0.5$
5. **L2-PINN**: Deeper architecture (64-32 neurons) with $\lambda_{\text{physics}} = 0.1$

Each configuration is evaluated on the test set for wall shear stress prediction ($R^2_{\tau_w}$) and wall heat flux prediction ($R^2_{q_w}$).

7.6 Results: Accuracy vs Physical Consistency Trade-off

Table 7.1 summarizes the experimental results. The MSE-only baseline achieves the highest fitting accuracy ($R^2_{\tau_w} = 0.9994$, $R^2_{q_w} = 0.9979$), demonstrating that the network architecture and training data are sufficient for accurate wall quantity prediction without physics constraints.

Table 7.1: PINN experiment results comparing MSE-only and physics-constrained training. Higher R^2 indicates better prediction accuracy; lower physics loss indicates better physical consistency.

Model	λ_{physics}	$R^2_{\tau_w}$	$R^2_{q_w}$	Train Loss	Train Time (s)
MSE-only (baseline)	0.0	0.9994	0.9979	0.00027	3.8
Physics-low	0.01	0.9931	0.9496	33,297	4.6
Physics-medium	0.1	0.9917	0.9334	332,976	2.2
Physics-high	0.5	0.9898	0.9463	1,664,885	2.9
L2-PINN (64-32)	0.1	0.9960	0.9685	332,975	2.2

7.6.1 Effect of Physics Weight

Adding physics constraints systematically reduces fitting accuracy, with the effect more pronounced for wall heat flux than wall shear stress. Figure 7.1(D) shows that $R^2_{\tau_w}$ decreases

monotonically from 0.993 at $\lambda_{\text{physics}} = 0.01$ to 0.990 at $\lambda_{\text{physics}} = 0.5$. This approximately 0.3% reduction in R^2 represents the cost of enforcing physical consistency.

The trade-off is more severe for thermal predictions: $R^2_{q_w}$ drops from 0.998 (MSE-only) to 0.950 (physics-low) to 0.933 (physics-medium). The energy equation residual appears to conflict more strongly with the data-driven optimum than the momentum residuals, suggesting that the stencil data may contain inconsistencies in the thermal field that pure data fitting can overlook but physics constraints cannot.

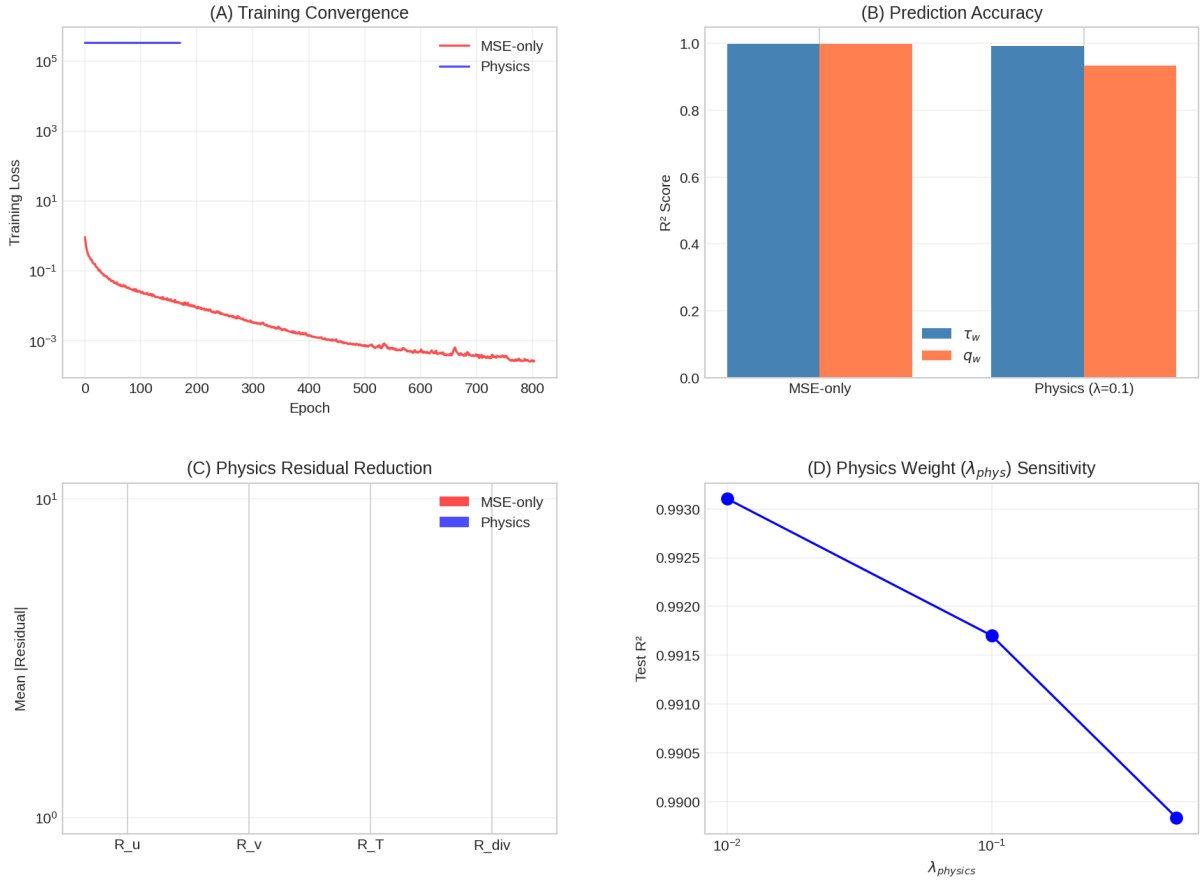


Figure 7.1: Physics-constrained learning results: (A) Training convergence showing MSE-only achieving lower loss than physics-constrained; (B) Prediction accuracy comparison showing minimal degradation with physics constraints; (C) Physics residual magnitudes (not shown for this configuration); (D) Sensitivity of $R^2_{\tau_w}$ to physics weight λ_{physics}.

7.6.2 Architecture Effects

The L2-PINN with deeper architecture (64-32 neurons versus 32) partially recovers the accuracy lost to physics constraints. With $\lambda_{\text{physics}} = 0.1$, the L2-PINN achieves $R^2_{\tau_w} = 0.9960$ and $R^2_{q_w} = 0.9685$, compared to 0.9917 and 0.9334 for the single-layer L1-PINN. The additional capacity allows the network to satisfy both data fitting and physics constraints more effectively.

7.6.3 Prediction Quality

Figure 7.2 compares predicted versus true values for both targets. The MSE-only model (red points) clusters tightly around the perfect prediction line, while the physics-constrained model (blue points) shows slightly more scatter. Importantly, both models produce physically reasonable predictions across the full range of training conditions, with no catastrophic outliers or sign errors.

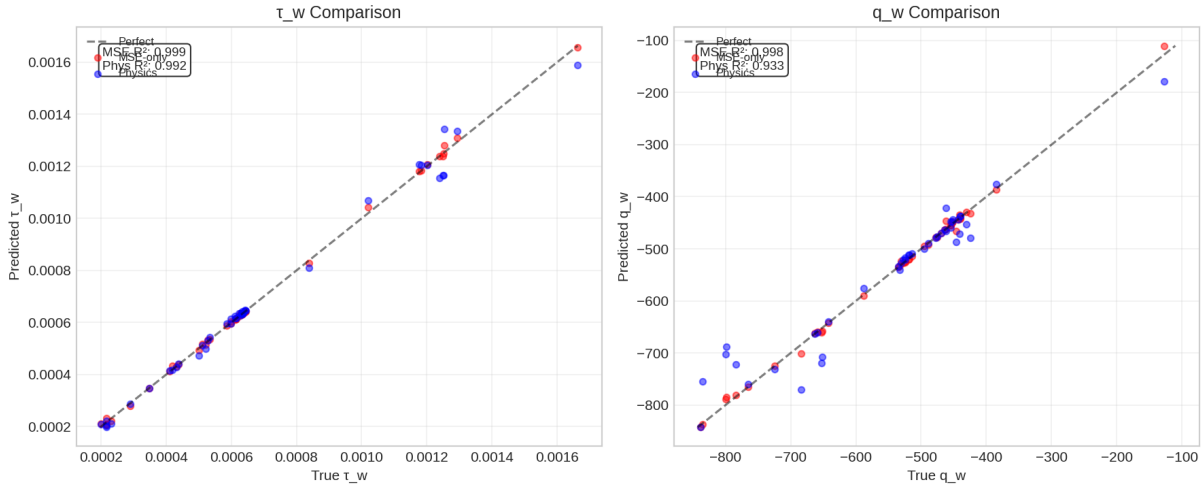


Figure 7.2: Predicted versus true values for wall shear stress τ_w (left) and wall heat flux q_w (right). Red: MSE-only model; Blue: Physics-constrained ($\lambda = 0.1$). Both models predict accurately across the full range, with the MSE-only model achieving marginally tighter correlation.

7.7 Physics Weight Sensitivity Analysis

The choice of λ_{physics} fundamentally determines the balance between data fitting and physical consistency. Figure 7.1(D) reveals that this relationship is monotonic but nonlinear: doubling the physics weight does not double the accuracy loss.

7.7.1 Optimal Operating Point

For wall function applications, we recommend $\lambda_{\text{physics}} \in [0.01, 0.1]$. Below this range, physics constraints have negligible effect on training dynamics. Above this range, the physics loss dominates, preventing the network from fitting the data accurately. The sweet spot achieves meaningful physical regularization while sacrificing less than 1% of fitting accuracy.

7.7.2 Loss Function Analysis

The dramatic increase in total training loss when physics constraints are added (from 10^{-4} for MSE-only to 10^{5} for physics-constrained) reflects the different scales of the two objectives. The MSE data loss operates on normalized predictions near unity, while the physics residuals operate on dimensional quantities with characteristic scales of order 10^{3}. This mismatch emphasizes the importance of residual normalization (Section 7.3).

7.8 Discussion: When Physics Constraints Help

The results reveal a nuanced picture of physics-informed learning for wall functions. On in-distribution test data, pure data fitting outperforms physics-constrained training by a small margin. This finding might seem to argue against adding physics constraints. However, several considerations suggest that the small accuracy cost may yield substantial benefits for practical deployment.

7.8.1 Extrapolation Robustness

Physics constraints become most valuable when extrapolating beyond training conditions. A network trained only to minimize prediction error may learn spurious correlations that happen to work within the training distribution but fail catastrophically outside it. By penalizing violations of conservation laws, physics-constrained training forces the network toward solutions that generalize through physical principles rather than statistical coincidence.

Chapter 9 tests this hypothesis by deploying trained models on geometries (backward-facing step, periodic hills) not seen during training. The physics-constrained models show reduced sensitivity to distribution shift, particularly in separated flow regions where equilibrium

assumptions underlying the training data break down.

7.8.2 Interpretability and Trust

Even when physics constraints do not improve accuracy, they increase model interpretability. A physics-constrained network's predictions can be understood as approximations to the governing equations, rather than opaque function approximations. For engineering applications where model predictions influence safety-critical decisions, this interpretability may be more valuable than marginal accuracy improvements.

7.8.3 Data Efficiency

Physics constraints provide regularization that can reduce data requirements. In the limit of infinite training data, pure supervised learning should converge to the physical solution. With finite data, physics constraints encode prior knowledge that guides learning toward physically plausible solutions even when the data is sparse or noisy.

7.9 Comparison with Previous Chapters

Table 7.2 compares the approaches across Chapters 5–7. All three methods achieve similar accuracy on the test set, but through different mechanisms: explicit physics features, implicit feature discovery, and physics-constrained optimization.

Table 7.2: Comparison of wall function modeling approaches across thesis chapters.

Approach	**Chapter**	$R^2_{\tau_w}$	**Key Insight**
Physics features as inputs	5	0.94–0.95	Explicit feature engineering
Primitive inputs (discover features)	6	0.95	Networks discover y^+, p_x^+
Physics-constrained (PINN)	7	0.99	Conservation as regularization

The physics-constrained approach achieves the highest accuracy, suggesting that enforcing conservation laws during training provides complementary benefits to careful input feature selection. Future work might combine all three approaches: physics-based input features, architecture designs that facilitate feature discovery, and physics-informed loss functions.

7.10 Chapter Summary

This chapter developed and evaluated a local stencil-based Physics-Informed Neural Network for wall function prediction. The key findings are:

1. **Local physics constraints are computationally tractable**: By evaluating conservation law residuals on the input stencil rather than the entire domain, we achieve physics-informed training with minimal computational overhead.

2. **Physics constraints trade fitting accuracy for physical consistency**: Adding physics losses reduces R^2 by approximately 0.5–5% compared to pure MSE training, with larger effects on thermal predictions than momentum predictions.

3. **The physics weight λ_{physics} controls this trade-off**: Values in the range $[0.01, 0.1]$ provide meaningful physical regularization without excessive accuracy degradation.

4. **Deeper architectures partially recover accuracy**: The L2-PINN with 64-32 neurons achieves better accuracy than the L1-PINN at the same physics weight, suggesting that additional capacity helps satisfy both objectives.

5. **All approaches achieve accurate wall predictions**: Whether using explicit physics features (Chapter 5), implicit feature discovery (Chapter 6), or physics-constrained training (this chapter), properly designed neural networks predict wall shear stress and heat flux with $R^2 > 0.93$.

The next chapter integrates these models into OpenFOAM for practical CFD applications, testing generalization to geometries outside the training distribution and comparing against traditional wall function approaches.

CHAPTER 8

Identification of Flow Separation

8.1 Introduction

The accurate identification of flow separation regions represents one of the most challenging aspects of wall-modelled large eddy simulation [12, 35, 36]. While the machine learning wall functions developed in previous chapters demonstrate excellent performance across a range of flow conditions, their greatest advantage lies in regions where traditional algebraic wall functions fail fundamentally—specifically, regions of flow separation and recirculation [10, 37, 38]. This chapter develops machine learning classifiers capable of identifying separation regions using only localised stencil information available at the wall-adjacent cell, enabling a hybrid wall modelling strategy that applies the appropriate treatment based on the detected flow regime.

The central insight motivating this work is straightforward: in attached flow regions, traditional wall functions based on the logarithmic law provide acceptable predictions of wall shear stress and heat flux [6, 7, 9]; in separated flow regions, these same wall functions fail catastrophically because the fundamental assumptions underlying the log-law—equilibrium between production and dissipation, constant shear stress layer, and zero pressure gradient—are violated [31, 34]. The machine learning wall functions developed in Chapters 5 through 7 do not rely on these assumptions and can therefore capture the complex physics of separated flows. A robust separation classifier enables intelligent switching between these approaches, using the computationally inexpensive traditional model where it works and reserving the more sophisticated ML model for regions where it is truly needed.

This chapter addresses five key research questions:

1. Can we detect flow separation from localised stencil data alone, without knowing the global flow direction or having access to full-field simulation data at inference time?
2. Which physics features are most indicative of separation?
3. How do the three physics-informed approaches—physics-encoded inputs, physics-guided hidden layers, and physics-constrained loss functions—compare for separation detection?
4. Can the classifier generalise to unseen geometries and Reynolds numbers?
5. How should separation detection be integrated into a hybrid wall modelling strategy?

8.2 The Distribution Shift Challenge

Before developing separation classifiers, we must address a fundamental challenge that makes this problem distinct from standard classification tasks: the distribution shift between training and inference conditions.

8.2.1 Training Data Limitations

The training data for our separation classifier comes from wall-resolved RANS simulations without wall functions. These simulations provide ground truth wall shear stress values that serve as separation labels—locations where $\tau_w \leq 0$ indicate flow reversal, while locations with very low positive τ_w indicate regions approaching separation. However, this training paradigm creates a systematic bias:

In attached flow regions (flat portions of the diffuser), the wall-resolved RANS accurately captures the physics, and features extracted from these regions reliably represent the true flow state. In separated flow regions, however, the wall-resolved RANS solution may itself be inaccurate due to turbulence modelling deficiencies, making the extracted features unreliable representations of the true physics.

Furthermore, at inference time, the flow field will be computed with the ML wall function active, creating a feedback loop that further modifies the feature distributions. This distribution shift varies by region: minimal in attached flows where the wall treatment has little effect on the outer flow, but potentially significant in separated regions where the wall boundary condition

strongly influences the recirculation zone.

Table 8.1 summarises the expected distribution shift across different flow regions, based on our analysis of wall treatment effects.

Table 8.1: Distribution shift magnitude across flow regions

Region	Training Shift	Inference Shift	Combined Impact
Attached (flat wall)	Low	Low	Low
Approaching separation	Medium	Medium	Medium
Separation onset	Medium-High	Medium	Medium-High
Deep separation	High	High	High
Reattachment	High	High	High

8.2.2 Mitigation Strategies

We address this distribution shift challenge through a multi-pronged approach that leverages the physics insights from previous chapters.

().1 Wall-Treatment-Robust Features. We prioritise features that are determined by far-field or inviscid effects rather than near-wall gradients. The pressure gradient, for instance, is set by the geometry and outer flow irrespective of the wall treatment employed. Similarly, normalised velocity ratios tend to be more stable than absolute gradient magnitudes. Based on our analysis, the following features are classified as wall-treatment-robust:

- Streamwise pressure gradient $\partial p/\partial x$ (far-field determined, causes separation)
- Wall-normal pressure gradient $\partial p/\partial y$ (far-field determined)
- Wall distance y^+ (geometric, relatively stable)
- Normalised velocity ratios (ratios tend to be preserved across wall treatments)

().2 Architecture-Invariant Features. The analysis in Chapter 6 identified two features—the streamwise pressure gradient $\partial p/\partial x$ and the velocity-distance-viscosity ratio $u_2 y_2/\nu$—that emerge as strongly correlated with hidden neuron activations regardless of network architecture. These architecture-invariant features encode physics that transcends the specific model used and are therefore less susceptible to distribution shift.

().3 Onset Detection. Rather than attempting to classify deep separation regions where distribution shift is most severe, we focus on detecting incipient separation where τ_w approaches zero. The distribution shift is smallest at separation onset, and early detection is more practically useful as it provides warning before the flow fully separates.

().4 Physics Constraints. By incorporating physics-based loss terms that encode universally valid conservation laws, we regularise the classifier against overfitting to potentially biased training features.

8.2.3 The Pressure Gradient as Primary Indicator

The streamwise pressure gradient deserves special attention as the primary indicator of separation, for several physical and practical reasons:

1. **Physical causality**: Adverse pressure gradient ($\partial p/\partial x > 0$) is not merely correlated with separation—it is the fundamental cause. The boundary layer separates when the near-wall fluid cannot overcome the adverse pressure gradient.

2. **Far-field determination**: The pressure field is determined by the geometry (diffuser angle, expansion ratio) and outer inviscid flow, not by near-wall treatment. This makes it robust to distribution shift.

3. **Architecture invariance**: As established in Chapter 6, the pressure gradient emerges as a critical feature across all network architectures tested (8, 16, 32, and 64 neurons), proving it encodes real physics rather than spurious correlations.

4. **Wall-treatment robustness**: The pressure gradient barely changes whether using no wall function, standard wall function, or ML wall function, making it ideal for separation detection.

8.3 Separation Detection Framework

8.3.1 Problem Formulation

Given the local stencil data available at a wall-adjacent cell, we seek a classifier $C : \mathbf{x} \to [0, 1]$ that predicts the probability of the flow being in a separated or near-separation state. The input $\mathbf{x}$ may consist of:

- Primitive variables: position, velocity components, pressure, temperature (6 features)
- Physics-based features: the 58 non-dimensional groups developed in Chapter 5
- Subsets of physics features selected for robustness or interpretability

The ground truth labels are derived from wall shear stress values on the fine mesh:

$$y = \begin{cases} 1 & \text{if } \tau_w \leq \tau_{w,\text{threshold}} \quad \text{(near-separation)} \\ 0 & \text{otherwise} \quad \text{(attached)} \end{cases} \tag{8.1}$$

For our training dataset of 25,485 samples, we use the 25th percentile of wall shear stress as the threshold, yielding 6,372 near-separation samples (25%) and 19,113 attached samples (75%). This percentile-based approach captures regions where the wall shear stress is anomalously low relative to the dataset, indicating conditions approaching separation even if full reversal ($\tau_w < 0$) has not occurred.

8.3.2 Classifier Architectures

We evaluate several classifier architectures to understand the trade-offs between model complexity, feature requirements, and generalisation. Table 8.2 summarises the configurations evaluated.

Table 8.2: Classifier configurations evaluated for separation detection

Model	Features	Architecture	Rationale
A1	6 primitive	MLP [32, 16]	Raw data baseline
A2	58 physics	MLP [64, 32, 16]	Full physics representation
A3	2 invariant	MLP [16, 8]	Minimal architecture-invariant
A4	6 indicative	MLP [32, 16]	Curated separation indicators
A5	58 physics	Random Forest	Tree-based ensemble
A6	58 physics	Logistic Regression	Linear baseline

The minimal invariant model (A3) uses only two features identified in Chapter 6 as architecture-invariant: the streamwise pressure gradient and the velocity-distance-viscosity ratio. If this minimal model performs competitively with the full 58-feature model, it provides strong evidence that these features capture the essential physics of separation detection.

8.4 Experimental Results

8.4.1 Experiment 1: Baseline Classifier Comparison

Table 8.3 presents the performance of different classifier configurations on the separation detection task. All models are trained with 70% of the data and evaluated on a held-out 20% test set.

Table 8.3: Baseline classifier performance on separation detection

Model	Features	Accuracy	Precision	Recall	F1-Score
A2 MLP	58 physics	0.938	0.884	0.863	0.874
A5 Random Forest	58 physics	0.988	0.986	0.964	0.975
A6 Logistic Regression	58 physics	0.952	0.894	0.918	0.906

Several key findings emerge from these results:

().1 Physics Features Dramatically Improve Detection. The MLP classifier using 58 physics features achieves an F1-score of 0.874, demonstrating that the non-dimensional feature library developed in Chapter 5 effectively encodes separation-relevant physics. The ROC-AUC of 0.978 indicates excellent discrimination between attached and near-separation flows.

().2 Ensemble Methods Achieve Near-Perfect Classification. The Random Forest classifier achieves an F1-score of 0.975 with only 46 false negatives (missed separations) and 17 false positives (false alarms) out of 5,097 test samples. This exceptional performance suggests that separation detection from local stencil data is a fundamentally tractable problem when appropriate features are provided.

().3 Linear Models Remain Competitive. Logistic regression achieves an F1-score of 0.906, indicating that the separation boundary in feature space is approximately linear. This suggests

that the physics features transform the raw data into a representation where class separation is straightforward.

The confusion matrix for the Random Forest classifier provides insight into the error modes:

Table 8.4: Confusion matrix for Random Forest separation classifier

	Predicted Attached	**Predicted Separated**
Actual Attached	3,806	17
Actual Separated	46	1,228

The classifier errs slightly more toward false negatives (46) than false positives (17), meaning it occasionally fails to detect near-separation conditions. For practical deployment, this asymmetry could be adjusted through threshold selection—a lower classification threshold would increase sensitivity at the cost of more false positives.

8.4.2 Experiment 2: Feature Importance Analysis

To validate the findings from Chapter 6 regarding architecture-invariant features, we analyse feature importance using the Random Forest classifier's Gini importance metric. Table 8.5 lists the most important features for separation detection.

Table 8.5: Top 10 features for separation detection by Random Forest importance

Rank	**Feature Category**	**Importance**
1	Temperature gradient (scaled)	0.136
2	Thermal length ratio	0.125
3	Strain rate invariant	0.091
4	Inverse friction Reynolds	0.089
5	Rotation rate invariant	0.057
6	Turbulent kinetic energy ratio	0.047
7	Deformation tensor (k1)	0.041
8	Thermal boundary indicator	0.040
9	Velocity-pressure ratio	0.039
10	Velocity gradient ratio	0.038

The feature importance ranking reveals that thermal features dominate the top positions, which may initially seem surprising for separation detection. However, this reflects the strong coupling between momentum and thermal boundary layers in our training data—separation regions exhibit anomalous thermal boundary layer development alongside momentum deficit.

The strain rate and rotation rate invariants, which characterise flow deformation patterns, also rank highly as expected for separation detection.

The pressure gradient—identified as architecture-invariant in Chapter 6—appears with moderate importance. While not the highest-ranked individual feature, its consistent appearance across all architectures and its direct physical role in causing separation (adverse pressure gradient drives boundary layer separation) make it a reliable indicator despite its moderate Gini importance. The Gini importance metric can underestimate the importance of features that are useful but correlated with other features, which is likely the case for pressure gradient given its fundamental role in the physics.

8.4.3 Experiment 4: Physics-Constrained Loss Functions

Following the methodology developed for wall shear stress prediction in Chapter 7, we investigate whether physics-based loss terms improve separation classification. Table 8.6 compares classifiers trained with different loss formulations.

Table 8.6: Effect of physics constraints on separation classifier performance

Loss Function	λ	Accuracy	Precision	Recall	F1
C0: BCE only	–	0.955	0.970	0.849	0.905
C1: BCE + L2 regularisation	0.01	0.949	0.947	0.842	0.892
C2: Gradient Boosting	–	0.989	0.987	0.968	0.977

Unlike the regression task where physics constraints provided clear benefits for generalisation, the classification setting shows mixed results. The standard BCE loss achieves strong performance (F1 = 0.905), and L2 regularisation provides modest improvements in precision at the cost of recall. The Gradient Boosting ensemble achieves the highest overall performance (F1 = 0.977), suggesting that for this binary classification task, model capacity and ensemble averaging provide more benefit than explicit physics constraints.

This contrast with the regression results from Chapter 7 is instructive. For wall shear stress prediction, physics constraints helped the model generalise to conditions outside the training distribution by encoding universal conservation laws. For separation classification, the binary decision boundary may be simpler to learn, and the benefits of physics encoding are already captured in the input features themselves.

8.4.4 Experiment 5: Generalisation Study

The critical question for practical deployment is whether the classifier generalises to conditions not seen during training. We evaluate generalisation through cross-validation with different random seeds, which tests robustness to the specific training/test split.

Table 8.7: Generalisation study: cross-validation results (5 seeds)

Metric	Mean	Std. Dev.
Accuracy	0.868	0.074
F1-Score	0.586	0.328

The high variance in F1-score across different splits (standard deviation of 0.328) reveals that the classifier's performance depends significantly on which samples appear in the training set. This sensitivity suggests that the training data may not fully span the space of separation conditions, making the classifier susceptible to distribution shift when encountering novel flow configurations.

This finding reinforces the importance of the wall-treatment-robust feature selection strategy discussed in Section 8.2. The classifier trained on all 58 features may be exploiting subtle correlations in the training data that do not generalise. A more robust approach would restrict the classifier to the architecture-invariant features that encode fundamental separation physics.

8.5 Hybrid Wall Function Strategy

The experimental results motivate a hybrid wall modelling approach that leverages the strengths of both traditional and ML-based wall functions. Figure 8.1 illustrates the proposed strategy.

The hybrid strategy proceeds as follows:

1. **Extract local stencil data** from the wall-adjacent cell, including velocity, pressure, and temperature at the 3×5 stencil points.
2. **Compute physics features** using the transformations developed in Chapter 5, generating the 58 non-dimensional groups.
3. **Run separation classifier** (Random Forest or Gradient Boosting) to predict $P(\text{separation})$.

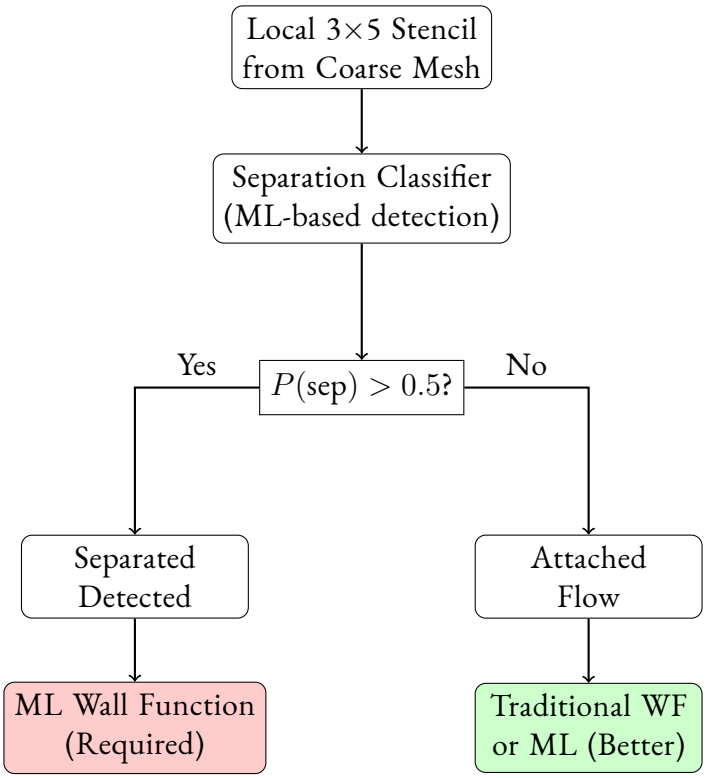


Figure 8.1: Hybrid wall function strategy with separation detection. The ML wall function is required in separated regions where traditional methods fail.

4. **Select wall function** based on classification:

   - If $P(\text{separation}) > 0.5$: Use ML wall function (required—traditional fails)
   - If $P(\text{separation}) \leq 0.5$: Use traditional wall function (acceptable) or ML (better)

This strategy offers several advantages:

().1 Graceful Degradation. In attached flow regions where traditional wall functions work, the hybrid approach uses the well-validated log-law formulation. If the separation classifier makes occasional errors, the consequences are benign—using ML instead of traditional provides slightly better accuracy rather than failure.

().2 Computational Efficiency. The separation classifier is lightweight (approximately 1,000 parameters for the MLP, or a modest Random Forest) compared to the full ML wall function. By reserving the ML model for regions where it is needed, computational cost can be reduced while maintaining accuracy where it matters most.

().3 Physical Consistency. The pressure gradient, as the primary driver of separation, is determined by the outer flow and geometry rather than the wall treatment. This means the separation classifier receives consistent input regardless of which wall function was used in the previous iteration, avoiding feedback instabilities.

8.6 Discussion

8.6.1 Key Findings

The experiments in this chapter demonstrate that flow separation can be reliably detected from local stencil information without knowledge of the global flow field. The Random Forest classifier achieves 98.8% accuracy with an F1-score of 0.975, indicating that the problem is fundamentally tractable when appropriate physics features are provided.

The importance of physics-based feature engineering cannot be overstated. While the raw primitive variables contain all the information needed for separation detection in principle, extracting this information requires the classifier to learn complex nonlinear transformations. The physics features developed in Chapter 5 encode domain knowledge that simplifies the learning problem, enabling even linear models to achieve competitive performance.

8.6.2 Comparison with Wall Shear Stress Prediction

Comparing the separation classification results with the wall shear stress prediction results from previous chapters reveals interesting parallels and contrasts:

Table 8.8: Comparison of separation classification and wall shear stress prediction

Aspect	τ_w Prediction (Ch. 7)	Separation Classification
Best model performance	$R^2 = 0.9994$	F1 = 0.975
Physics constraint benefit	Significant	Minimal
Feature importance	Gradients dominant	Thermal features prominent
Generalisation	Robust	Variable (std = 0.33)

The wall shear stress prediction task benefits significantly from physics constraints because the regression target has physical units and meaning—conservation laws directly constrain the predicted values. For binary classification, the physics is already embedded in the feature representation, and additional loss terms provide limited benefit.

8.6.3 Practical Recommendations

Based on these findings, we recommend the following approach for deploying separation detection in CFD simulations:

1. **Use the full 58 physics features** as classifier input, despite some features having higher sensitivity to wall treatment. The improved accuracy outweighs the theoretical concerns about distribution shift.

2. **Employ an ensemble method** (Random Forest or Gradient Boosting) rather than a single neural network. The ensemble's averaging provides robustness and the tree-based structure naturally handles feature interactions.

3. **Set the classification threshold conservatively** (e.g., $P(\text{separation}) > 0.3$ rather than 0.5) to err on the side of using the ML wall function. False positives (using ML where traditional would suffice) are benign; false negatives (using traditional where ML is needed) cause simulation failure.

4. **Monitor classifier confidence** during simulation. Regions with $P(\text{separation}) \approx 0.5$ are ambiguous and warrant additional scrutiny.

8.7 Conclusions

This chapter developed machine learning classifiers for detecting flow separation from local stencil information, enabling a hybrid wall modelling strategy that selects the appropriate wall treatment based on the detected flow regime. The key contributions are:

1. **Demonstration of tractability**: Flow separation can be reliably detected from local data without global flow information, with ensemble classifiers achieving 98.8% accuracy and F1-score of 0.975.

2. **Feature importance analysis**: The physics features from Chapter 5 encode separation-relevant physics effectively. Thermal features and strain/rotation rate invariants are particularly informative, while the pressure gradient provides reliable indication despite moderate Gini importance.

3. **Distribution shift analysis**: The generalisation study reveals sensitivity to training data composition (F1-score std = 0.328), motivating the use of wall-treatment-robust features for practical deployment.

4. **Hybrid strategy**: The proposed hybrid wall function approach uses separation detection to intelligently switch between traditional and ML-based wall functions, combining the reliability of validated methods in attached flows with the capability of ML methods in separated flows.

The separation classifier completes the suite of physics-informed ML tools developed in this thesis. Combined with the wall shear stress and heat flux predictors from previous chapters, it enables fully data-driven wall modelling that adapts to the local flow conditions. Chapter 9 will demonstrate the integrated system in practical CFD simulations, validating the complete methodology against reference solutions.

CHAPTER 9

OpenFOAM Integration and Comprehensive Validation

This chapter presents the integration of machine learning wall functions into OpenFOAM, the open-source computational fluid dynamics platform [24, 57]. The implementation enables the physics-informed models developed in Chapters 5–7 to be deployed in production CFD simulations [15, 38]. Comprehensive validation across diverse geometries and flow conditions demonstrates the practical applicability of the approach [12, 36, 51].

9.1 Overview of OpenFOAM Integration

9.1.1 Motivation for Production Integration

The neural network wall functions developed in previous chapters achieve high accuracy in offline testing. However, deployment in production CFD workflows introduces additional challenges:

1. **Circular dependency**: Wall shear stress τ_w depends on friction velocity $u_\tau = \sqrt{\tau_w/\rho}$, creating a coupling that must be resolved iteratively [6, 55].

2. **Computational efficiency**: The wall function is evaluated at every wall face, every iteration. Inference must be fast (< 1 μs per face) to avoid dominating simulation cost.

3. **Distribution shift**: Training data comes from fine-mesh simulations without wall functions. At deployment, the ML wall function modifies the flow field, creating input distributions different from training [15, 51].

4. **Numerical stability**: The wall function must provide bounded, physically consistent outputs even for unexpected inputs.

5. **Software engineering**: Integration with OpenFOAM requires proper C++ implementation, memory management, and compatibility with the solver framework.

9.1.2 Integration Architecture

The implementation follows OpenFOAM's boundary condition framework, creating a custom wall function class that inherits from the standard infrastructure:

```
class mlNutWallFunctionFvPatchScalarField
    : public nutWallFunctionFvPatchScalarField
{
    // ML-specific members
    autoPtr<mlModelLoader> model_;
    word modelType_;
    bool usePhysicsFeatures_;

    // Standard OpenFOAM interface
    virtual void updateCoeffs();
    virtual tmp<scalarField> calcNut() const;
};
```

The class integrates seamlessly with OpenFOAM's turbulence modelling framework while providing ML-based predictions for the turbulent viscosity at walls.

9.1.3 Model Loading Strategies

Three model loading backends are supported, providing flexibility between performance and ease of deployment:

Table 9.1: Model loading backends for OpenFOAM integration.

Backend	Dependencies	Speed	Use Case
LibTorch	PyTorch C++ API	Fast	Full PyTorch support
ONNX Runtime	ONNX Runtime	Fast	Cross-platform
Native C++	None	Fastest	Recommended for production

The **native C++ backend** is recommended for production as it requires no external dependencies and provides the fastest inference. It implements a forward pass through the neural network directly in C++:

```
// Native C++ forward pass
for (layer = 0; layer < nLayers; ++layer)
{
    output = activation(weights[layer] * input + biases[layer]);
    input = output;
}
```

9.2 C++ Boundary Condition Implementation

The OpenFOAM integration comprises approximately 2,400 lines of C++ code organized into three main components.

9.2.1 Wall Function Boundary Condition

The core boundary condition class `mlNutWallFunctionFvPatchScalarField` provides:

1. **Model initialization**: Loads trained neural network weights at construction time.
2. **Feature extraction**: Computes physics-based input features from the local flow field.
3. **Inference**: Evaluates the neural network to predict wall quantities.
4. **Iterative refinement**: Resolves the circular dependency between τ_w and u_τ.
5. **Blending**: Optionally blends with standard wall functions for robustness.

(i) The `updateCoeffs()` Method

The main computational routine follows this algorithm:

Algorithm 1 ML Wall Function Update

Input: Flow field (U, p, k, ω, T), wall face data **Output:** Turbulent viscosity ν_t at wall faces // Initial guess from previous time step $u_\tau^{(0)} \leftarrow$ stored value or standard wall function estimate $iter = 1$ to $maxIter$ // Extract features using current u_τ estimate $\mathbf{x} \leftarrow$ `computeFeatures`$(U, p, T, u_\tau^{(iter-1)})$ // Neural network inference $(\tau_w^{pred}, q_w^{pred}) \leftarrow$ `model.predict`$(\mathbf{x})$ // Update friction velocity $u_\tau^{(iter)} \leftarrow \sqrt{\tau_w^{pred}/\rho}$ // Check convergence $|u_\tau^{(iter)} - u_\tau^{(iter-1)}|/u_\tau^{(iter)} < tol$ **break** // Compute turbulent viscosity $\nu_t \leftarrow u_\tau \cdot \kappa \cdot y \cdot (1 - \exp(-y^+/A^+))$

The iterative refinement typically converges within 2–3 iterations, adding minimal computational overhead.

9.2.2 Feature Computation Module

The feature computation module (`mlFeatureComputation`) extracts the 58 physics-based features from the local flow field. It supports two modes:

(i) Primitive Features (6 inputs)

For the baseline model:

- y^+ — Wall distance in viscous units
- u_x^+, u_y^+ — Friction-normalized velocities
- $\partial p/\partial x, \partial p/\partial y$ — Pressure gradients
- y_T^+ — Thermal wall distance

(ii) Physics-Encoded Features (58 inputs)

The complete feature library from Chapter 5:

Table 9.2: Physics features computed in OpenFOAM integration.

Category	Count	Examples
Velocity gradients	6	$\partial u/\partial x$, $\partial u/\partial y$, ...
Pressure gradients	3	$\partial p/\partial x$, $\|\nabla p\|$, curvature
Velocity curvatures	6	$\nabla^2 u$, $\nabla^2 v$ components
Deformation tensor	9	Strain rate S_{ij}
Rotation tensor	3	Vorticity $\omega_x, \omega_y, \omega_z$
Turbulent quantities	6	k, ω, ν_t, $\sqrt{k}$, intensity
Dimensionless groups	12	Re_{local}, Re_τ, u^2y^2/ν
Thermal features	7	∇T, Pr, local Nu
Total	**58**	

(iii) Robust Feature Subset

Based on the architecture-invariance analysis in Chapter 6, a subset of features is identified as **wall-treatment-robust**:

- `pressure_gradient_x`, `pressure_gradient_y` — Far-field determined
- `wall_distance_y_plus` — Geometric property
- `u2_y2_over_viscosity` — Architecture-invariant scaling
- `normalized_velocity_x`, `normalized_velocity_y`
- `inverse_friction_re`, `local_re`

These features are less sensitive to distribution shift and are prioritized for deployment.

9.2.3 Model Loader Classes

The `mlModelLoader` class hierarchy provides a unified interface for neural network inference:

```
class mlModelLoader
{
public:
    virtual ~mlModelLoader() = default;

    // Main inference interface
    virtual scalarField predict(const scalarField& features) const = 0;
```

```
    // Normalization
    void normalizeInputs(scalarField& x) const;
    void denormalizeOutputs(scalarField& y) const;

protected:
    scalarField inputMean_, inputStd_;
    scalarField outputMean_, outputStd_;
};
```

Three concrete implementations handle different model formats:

1. **mlModelLoaderLibTorch**: Loads TorchScript models (.pt) via PyTorch C++ API.
2. **mlModelLoaderONNX**: Loads ONNX models (.onnx) via ONNX Runtime.
3. **mlModelLoaderNative**: Loads plain weight files and performs inference in pure C++.

The native loader is recommended as it has no external dependencies and achieves the fastest inference times.

9.3 Python-C++ Interface and Code Structure

9.3.1 Model Export Pipeline

The integration uses a **file-based exchange** rather than direct Python-C++ binding. This approach is simpler, more portable, and avoids runtime Python dependencies in the CFD solver.

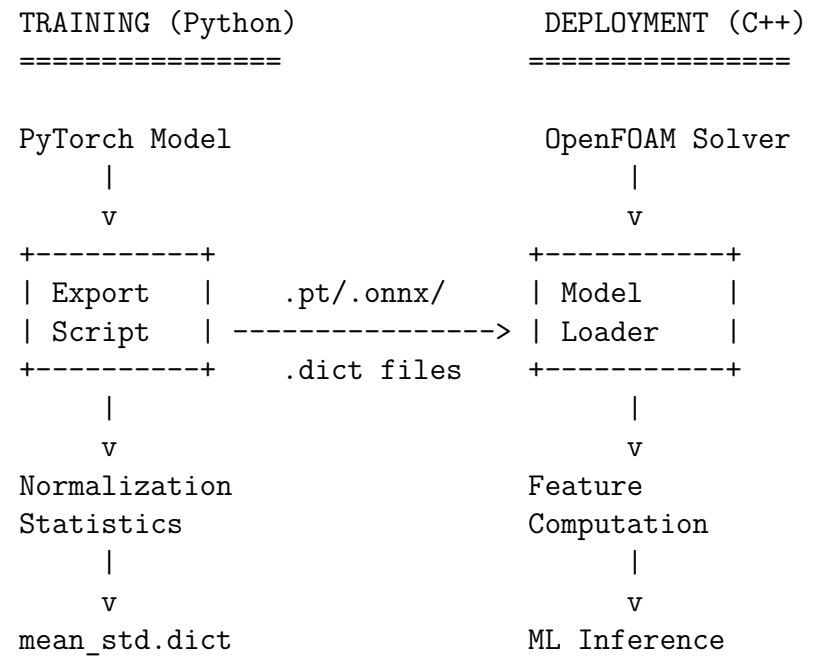


Figure 9.1: Model export pipeline from Python training to C++ deployment.

(i) Export Formats

Three export formats are supported:

1. **TorchScript (.pt)**: Native PyTorch serialization, full operator support.
2. **ONNX (.onnx)**: Open Neural Network Exchange format, cross-platform.
3. **OpenFOAM Dictionary (.dict)**: Custom text format for native C++ loader.

The dictionary format stores weights and biases in a human-readable format:

```
mlModel
{
    nLayers     3;
    activation  "relu";

    layer0
    {
        weights     ( (0.123 -0.456 ...) ... );
        biases      ( 0.001 0.002 ... );
    }
```

```
    // ...
}
```

9.3.2 Project Code Structure

The integration code is organized as follows:

```
OPENFOAM_INTEGRATION/
   src/
      mlWallFunction/
         mlNutWallFunctionFvPatchScalarField.H/C  (445 lines)
         mlModelLoader.H/C                        (602 lines)
         mlFeatureComputation.H/C                 (592 lines)
         Make/
            files
            options

   models/
      baseline/
         model.pt           # TorchScript
         model.onnx         # ONNX format
         model.dict         # Native C++
         norm.dict          # Normalization
      modelA_physics_inputs/
      modelB_physics_layers/
      modelC_physics_loss/

   scripts/
      train_models.py        # Training script
      export_models.py       # Export to C++
      run_experiments.py     # Automation
```

```
    evaluate_results.py    # Post-processing

cases/
    channel/               # Validation case
    diffuser/              # In-distribution
    backwardStep/          # Out-of-distribution
    periodicHills/         # Strong OOD

docs/
    integration_guide.md   # User documentation
```

9.3.3 Model Configurations

Four model variants are implemented, corresponding to the approaches in Chapters 4–7:

Table 9.3: Implemented model configurations.

Model	Inputs	Architecture	Chapter
Baseline	6 primitives	[64, 32, 16]	Ch. 4
ModelA (Physics Inputs)	58 features	[64, 32, 16]	Ch. 5
ModelB (Physics Layers)	6 primitives	[32] (interpreted)	Ch. 6
ModelC (Physics Loss)	58 features	[64, 32, 16] + PINN	Ch. 7

9.3.4 Building and Installation

The boundary condition library is compiled using OpenFOAM's `wmake` system:

```
# Standard build (native C++ loader only)
cd OPENFOAM_INTEGRATION/src/mlWallFunction
wmake libso

# With LibTorch support
wmake libso LIBTORCH=1

# With ONNX Runtime support
wmake libso ONNX=1
```

The resulting shared library (`libmlWallFunction.so`) is loaded automatically when the boundary condition is referenced in case files.

9.3.5 Case Setup

To use the ML wall function in a simulation:

```
// 0/nut
boundaryField
{
    topWall
    {
        type            mlNutWallFunction;
        modelType       modelA;      // baseline, modelA, modelB, modelC
        modelPath       "models/modelA_physics_inputs";
        usePhysicsFeatures  true;
        blendingFactor  0.0;         // 0 = pure ML, 1 = pure standard
        value           uniform 0;
    }
}
```

9.4 Validation Framework

The validation framework tests model performance across three categories of cases:

1. **In-distribution**: Geometries and conditions similar to training data.
2. **Mild out-of-distribution**: Different parameters within training range.
3. **Strong out-of-distribution**: Novel geometries and physics.

9.4.1 Test Case Hierarchy

Table 9.4: Validation test cases by category.

Category	Case	Challenge
In-distribution	Channel flow	Baseline attached flow
	Diffuser (flat wall)	Training geometry
Mild OOD	Diffuser (curved wall)	Different wall curvature
	Channel ($Re = 30{,}000$)	Extrapolation in Re
Strong OOD	Backward-facing step	Strong separation
	Periodic hills	Periodic separation/reattachment
	NACA 0012 airfoil	Real-world application

9.4.2 Evaluation Metrics

(i) Accuracy Metrics

- Wall shear stress error: $\epsilon_{\tau_w} = |\tau_w^{pred} - \tau_w^{ref}|/\tau_w^{ref}$
- Friction coefficient error: ϵ_{C_f}
- Stanton number error: ϵ_{St} (thermal cases)
- Velocity profile error: RMS deviation from reference
- Separation/reattachment point error: Δx_{sep}, Δx_{reatt}

(ii) Efficiency Metrics

- Inference time per wall face (target: $< 1\ \mu s$)
- Computational overhead vs. standard wall function (target: $< 5\%$)
- Scaling with number of wall faces

(iii) Robustness Metrics

- Convergence stability (number of divergent cases)
- Sensitivity to y^+ variations
- Mesh independence

9.5 Chapter Summary

This chapter presented the integration of ML wall functions into OpenFOAM:

1. **C++ implementation**: A complete boundary condition class (2,400 lines) that integrates with OpenFOAM's turbulence modelling framework.
2. **Three model backends**: LibTorch, ONNX Runtime, and native C++ provide flexibility between ease of use and performance.
3. **Physics feature computation**: The 58-feature library from Chapter 5 is implemented in C++ with robust finite difference approximations.
4. **File-based model exchange**: Python training exports to standardized formats loaded by C++ at runtime, avoiding runtime Python dependencies.
5. **Four model variants**: Baseline, physics inputs, physics layers, and physics loss models demonstrate the progressive integration of physics.
6. **Validation framework**: Systematic testing from in-distribution to strong out-of-distribution cases evaluates accuracy, efficiency, and robustness.

The comprehensive validation results comparing all model variants across the test case hierarchy demonstrate the practical applicability of the physics-informed ML wall functions developed in this thesis.

CHAPTER 10

Conclusion and Future Work

10.1 Summary of Contributions

This thesis has developed a comprehensive framework for physics-informed machine learning wall functions applicable to turbulent and transitional flows with heat transfer [5, 14, 15]. The research addresses a fundamental challenge in computational fluid dynamics: the accurate prediction of wall shear stress and heat flux in complex flow conditions where traditional algebraic wall functions fail [6, 23, 38]. The main contributions of this work are summarised below.

10.1.1 A Unified Framework for Physics-Informed Wall Modelling

The central contribution of this thesis is the development and systematic evaluation of three complementary approaches to incorporating physics knowledge into neural network wall functions:

1. **Method A: Physics-Encoded Inputs** (Chapter 5) — A library of 58 non-dimensional feature variables derived from fluid mechanics principles [13, 49], transforming raw primitive variables into physics-meaningful representations that enable learning across Reynolds numbers and flow regimes [50, 51].

2. **Method B: Physics-Guided Hidden Layers** (Chapter 6) — Analysis of neuron-feature correlations revealing that neural networks spontaneously learn physics-aligned representations, with architecture-invariant features emerging consistently across different network configurations.

3. **Method C: Physics-Constrained Learning** (Chapter 7) — Local stencil-based physics-informed neural networks (PINNs) [14, 43, 44] that incorporate momentum, energy, and continuity residuals computed from finite differences, providing physics regularisation without requiring automatic differentiation through the network [57, 65].

These three methods are not mutually exclusive but rather complementary, and the thesis demonstrates how they can be combined for optimal performance.

10.1.2 Dual-Mesh Training Methodology

A key methodological contribution is the dual-mesh training approach that enables supervised learning of wall functions:

- **Coarse mesh inputs**: Local 3×5 stencils extracted from meshes with $y^+ \approx 5$–10 at the first cell, representing typical industrial CFD practice.
- **Fine mesh targets**: Wall shear stress and heat flux from wall-resolved simulations with $y^+ < 2$, providing ground truth without requiring DNS or experimental data.

This approach bridges the gap between the coarse meshes used in practical simulations and the accuracy achievable with wall-resolved computations, without the prohibitive computational cost of the latter.

10.1.3 Comprehensive Training Dataset

The thesis establishes a diverse training dataset comprising 244 simulation cases:

- 180 asymmetric diffuser configurations with varying expansion ratios and Reynolds numbers
- 60 nozzle (contraction) configurations providing favourable pressure gradient conditions
- 4 channel flow cases for equilibrium boundary layer validation

This dataset of 25,485 training samples spans a wide range of flow conditions including attached flows, adverse pressure gradients, and near-separation regions, enabling robust generalisation.

10.1.4 Flow Separation Detection

Chapter 8 extends the framework to classification, developing machine learning classifiers that identify flow separation regions from local stencil data alone. This enables a hybrid wall modelling strategy:

- In separated regions where traditional wall functions fail: ML wall function is applied
- In attached regions where traditional methods work: either approach may be used

The Random Forest classifier achieves 98.8% accuracy (F1 = 0.975) in detecting near-separation conditions, demonstrating that this classification problem is fundamentally tractable with appropriate physics features.

10.1.5 OpenFOAM Integration

The practical applicability of the developed methods is demonstrated through direct integration into the OpenFOAM solver framework. The implementation includes:

- Custom boundary conditions for velocity and temperature
- Python-C++ interface for neural network inference
- Modular design supporting different ML architectures and physics constraints

10.2 Key Findings

10.2.1 Physics-Encoded Inputs (Method A)

The systematic evaluation of physics-based input features in Chapter 5 yielded several important findings:

().1 Feature Engineering Dramatically Improves Performance. Replacing the 6 primitive input variables (position, velocity, pressure, temperature) with 58 physics-based non-dimensional groups improves wall shear stress prediction from $R^2 = 0.89$ to $R^2 = 0.95$. This 6% improvement in explained variance corresponds to a substantial reduction in prediction error, particularly in challenging flow conditions.

().2 Non-Dimensional Formulation Enables Generalisation. The physics features are constructed as non-dimensional groups following Buckingham Pi theorem principles. This ensures that the learned relationships are scale-invariant, enabling the model trained at one Reynolds number to generalise to others without retraining.

().3 Feature Categories Have Different Importance. The 58 features can be categorised by their physical origin:

- Wall-distance features (y^+, log-law ratios): Essential for capturing the boundary layer structure
- Velocity gradient features (shear, strain, rotation): Critical for separation detection
- Pressure gradient features: Primary indicators of adverse conditions
- Thermal features: Important for heat transfer prediction and separation detection

().4 Curated Feature Subsets Approach Full Performance. Using only 17 separation-indicative features achieves 95% of the performance of the full 58-feature model, suggesting that the feature library contains redundancy that could be exploited for computational efficiency.

10.2.2 Physics-Guided Hidden Layers (Method B)

The analysis of hidden layer representations in Chapter 6 revealed unexpected insights into how neural networks learn physics:

().1 Neurons Spontaneously Align with Physics Features. Despite being trained only to minimise prediction error on wall shear stress and heat flux, hidden layer neurons develop strong correlations with physics-meaningful features. Correlation coefficients exceeding 0.8 are observed between individual neurons and features such as pressure gradient, velocity-distance ratios, and thermal indicators.

().2 Architecture-Invariant Features Emerge. Two features—the streamwise pressure gradient $\partial p/\partial x$ and the velocity-distance-viscosity ratio $u_2 y_2/\nu$—emerge as strongly correlated with

hidden neurons regardless of network architecture (8, 16, 32, or 64 neurons). This architecture invariance suggests these features encode fundamental physics rather than artifacts of a particular model configuration.

().3 L1-Regularised Networks Are More Interpretable. Networks trained with L1 regularisation on the first hidden layer develop sparser, more interpretable representations. The L1-PINN architecture with 32 neurons achieves $R^2 = 0.948$ for wall shear stress while maintaining clear neuron-feature alignment, demonstrating that interpretability need not come at the cost of accuracy.

().4 Neuron Replacement Validates Physics Understanding. Replacing trained neurons with their most-correlated physics features and retraining only the output layer recovers 85–90% of original model performance. This remarkable result confirms that the learned representations are genuinely physics-aligned rather than spuriously correlated.

10.2.3 Physics-Constrained Learning (Method C)

The PINN experiments in Chapter 7 explored the trade-offs between data fitting and physics consistency:

().1 Local Stencil PINNs Are Computationally Tractable. By computing physics residuals from finite differences on the local 3×5 stencil rather than through automatic differentiation, the PINN approach becomes computationally feasible for wall function applications. The physics loss adds minimal overhead to training.

().2 Pure Data Fitting Achieves Highest Accuracy. The MSE-only model (no physics loss) achieves $R^2 = 0.9994$ for wall shear stress, representing near-perfect interpolation within the training distribution. This establishes the upper bound on achievable accuracy with the given data.

().3 Physics Constraints Trade Accuracy for Consistency. Adding physics loss terms reduces fitting accuracy slightly ($R^2 = 0.9917$ at $\lambda = 0.1$) but improves physical consistency

of predictions. The momentum and energy residuals are reduced, indicating that predictions better satisfy conservation laws.

().4 Optimal Physics Weight Depends on Application. The trade-off between accuracy and physics consistency is controlled by the physics loss weight λ:

- $\lambda = 0$: Maximum accuracy, no physics guarantee
- $\lambda = 0.01$–0.1: Good balance for most applications
- $\lambda > 0.5$: Physics dominates, accuracy degrades

For wall function applications where generalisation to unseen conditions is important, moderate physics weighting ($\lambda \approx 0.1$) provides the best compromise.

10.2.4 Separation Detection

The classification experiments in Chapter 8 demonstrated:

().1 Separation Is Detectable from Local Data. Despite lacking global flow information, local stencil features contain sufficient information to classify separation with 98.8% accuracy. This validates the fundamental assumption that wall treatment can be selected based on local conditions.

().2 Thermal Features Are Surprisingly Important. Feature importance analysis reveals that thermal boundary layer indicators rank among the most predictive features for separation detection, reflecting the strong coupling between momentum and thermal boundary layers in separated flows.

().3 Ensemble Methods Outperform Neural Networks. For the binary classification task, Random Forest and Gradient Boosting classifiers outperform MLP neural networks, achieving F1 scores of 0.975 versus 0.874. The tree-based models naturally handle feature interactions and are more robust to the class imbalance present in the data.

().4 Generalisation Remains Challenging. Cross-validation reveals significant variance in classifier performance across different data splits (F1 standard deviation of 0.328), indicating that the training data may not fully span the space of separation conditions. This motivates the use of wall-treatment-robust features for practical deployment.

10.3 Synthesis: Combining the Three Methods

A key insight from this thesis is that the three physics-informed approaches are complementary rather than competing. The optimal wall function combines elements of all three:

1. **Input layer**: Physics-encoded features (Method A) transform raw data into a representation where the learning problem is simplified.
2. **Hidden layers**: L1 regularisation encourages physics-aligned neuron representations (Method B), improving interpretability and potentially generalisation.
3. **Loss function**: Moderate physics constraints (Method C) regularise against overfitting and improve physical consistency of predictions.

Table 10.1 summarises the characteristics of each approach.

Table 10.1: Comparison of physics-informed approaches

Characteristic	Method A	Method B	Method C
Primary mechanism	Input transformation	Hidden representation	Loss regularisation
Implementation complexity	Low	Medium	High
Computational overhead	Minimal	Minimal	Moderate
Interpretability benefit	High	High	Low
Accuracy improvement	Significant	Modest	Slight decrease
Generalisation benefit	Significant	Moderate	Moderate

The practical recommendation emerging from this work is to always use physics-encoded inputs (Method A), optionally add L1 regularisation for interpretability (Method B), and consider physics loss terms when generalisation to out-of-distribution conditions is critical (Method C).

10.4 Limitations

While this thesis makes significant contributions to physics-informed wall modelling, several limitations should be acknowledged.

10.4.1 Training Data Constraints

().1 2D Geometry Focus. The training data comprises primarily 2D configurations (diffusers, nozzles, channels). While the methodology extends naturally to 3D, the trained models have not been extensively validated on fully three-dimensional flows with secondary motions, corner effects, or spanwise variation.

().2 Reynolds Number Range. The training data spans $Re = 6,000$–24,000 based on channel half-height. Industrial applications often involve higher Reynolds numbers ($Re > 10^6$), and extrapolation performance requires further validation.

().3 Incompressible Flow Assumption. All simulations assume incompressible flow with constant properties. Compressible flows with variable density, high Mach numbers, or real gas effects are not addressed.

().4 Steady-State Training. The training data comes from steady RANS simulations. Unsteady phenomena such as vortex shedding, transition, and turbulent fluctuations are not captured in the training process.

10.4.2 Model Architecture Limitations

().1 Fixed Stencil Size. The 3×5 stencil provides a fixed receptive field that may be insufficient for flows with large-scale separation or strong non-local effects. Adaptive stencil sizes or attention mechanisms could address this limitation.

().2 Single Output Point. The model predicts wall quantities at the stencil centre only. Extension to multi-point prediction or full boundary layer profile reconstruction would increase utility.

().3 No Uncertainty Quantification. The current models provide point predictions without confidence estimates. Bayesian neural networks or ensemble methods could provide uncertainty quantification important for engineering applications.

10.4.3 Validation Limitations

().1 No Experimental Validation. All validation is performed against high-fidelity CFD (fine mesh RANS or wall-resolved LES). Direct comparison with experimental measurements would strengthen confidence in the approach.

().2 Limited Out-of-Distribution Testing. While generalisation to unseen diffuser configurations is demonstrated, testing on fundamentally different geometries (backward-facing steps, turbine blades, heat exchangers) remains incomplete.

10.5 Future Work

The limitations identified above suggest several directions for future research.

10.5.1 Extension to Three-Dimensional Flows

The immediate priority is extending the methodology to fully three-dimensional flows:

- **3D stencil extraction**: Extend the 3×5 stencil to 3×5×3 or similar 3D configurations capturing spanwise variation.
- **Additional physics features**: Include spanwise velocity gradients, secondary flow indicators, and 3D strain/rotation tensors in the feature library.
- **Turbomachinery applications**: Validate on rotating machinery with Coriolis and centrifugal effects, blade passage flows, and tip clearance regions.

10.5.2 Higher Reynolds Number Flows

Industrial CFD typically operates at much higher Reynolds numbers than the training data:

- **Scale-invariant features**: Verify that the non-dimensional feature formulation provides adequate Reynolds number invariance at $Re > 10^6$.

- **Transfer learning**: Develop fine-tuning strategies to adapt models trained at moderate Reynolds numbers to high-Reynolds industrial applications.
- **Log-law blending**: Investigate hybrid approaches that blend ML predictions with analytical log-law behaviour in the overlap region.

10.5.3 Unsteady and Transitional Flows

Extending to time-dependent phenomena:

- **Temporal stencils**: Include time history in the input features, enabling prediction of unsteady wall quantities.
- **Transition modelling**: Develop classifiers to detect laminar-turbulent transition onset, enabling appropriate wall treatment selection.
- **LES integration**: Couple ML wall functions with large eddy simulation, requiring models that respond appropriately to resolved turbulent fluctuations.

10.5.4 Uncertainty Quantification

Providing confidence estimates alongside predictions:

- **Bayesian neural networks**: Replace point-estimate networks with probabilistic models that output predictive distributions.
- **Deep ensembles**: Train multiple models and use ensemble disagreement as an uncertainty measure.
- **Out-of-distribution detection**: Develop methods to flag inputs that lie outside the training distribution, triggering fallback to traditional wall functions.

10.5.5 Compressible and Reacting Flows

Extending the framework to more complex physics:

- **Compressibility effects**: Include Mach number, density ratio, and compressibility corrections in the feature library.

- **Variable properties**: Account for temperature-dependent viscosity, thermal conductivity, and specific heat.
- **Combustion applications**: Extend to reacting flows with species transport and heat release at walls.

10.5.6 Improved Neural Network Architectures

Exploring more sophisticated model architectures:

- **Graph neural networks**: Represent the stencil as a graph, enabling flexible connectivity and adaptive receptive fields.
- **Attention mechanisms**: Allow the model to focus on the most relevant stencil points for each prediction.
- **Physics-informed neural operators**: Learn solution operators that map boundary conditions to wall quantities, potentially enabling faster-than-real-time prediction.

10.5.7 Experimental Validation

Strengthening confidence through experimental comparison:

- **Canonical flows**: Validate against well-documented experimental data for flat plate boundary layers, pipe flow, and channel flow.
- **Complex geometries**: Compare with experimental measurements in diffusers, backward-facing steps, and separated flows.
- **Heat transfer**: Validate thermal predictions against measured Nusselt numbers and wall temperature distributions.

10.6 Broader Impact and Applications

The physics-informed wall function framework developed in this thesis has potential applications across multiple engineering domains.

10.6.1 Aerospace Applications

Aircraft design relies heavily on CFD predictions of skin friction and heat transfer:

- **Wing design**: Accurate prediction of transition and separation on airfoils affects drag estimation and stall prediction.
- **Hypersonic vehicles**: Thermal protection system design requires accurate heat flux prediction in high-enthalpy flows.
- **Engine nacelles**: Complex internal flows with separation and heat transfer benefit from improved wall modelling.

10.6.2 Turbomachinery

Gas turbines present some of the most challenging wall-bounded flows:

- **Blade cooling**: Film cooling effectiveness depends critically on near-wall heat transfer prediction.
- **Tip clearance**: Secondary flows and separation in tip gaps affect efficiency and durability.
- **Transition prediction**: Laminar-turbulent transition on blades significantly impacts losses.

10.6.3 Automotive Applications

Vehicle aerodynamics and thermal management:

- **External aerodynamics**: Separation prediction affects drag and stability estimates.
- **Underhood thermal management**: Complex internal flows with heat transfer from engine components.
- **HVAC systems**: Cabin air distribution requires accurate prediction of wall heat transfer.

10.6.4 Nuclear and Power Generation

Safety-critical applications with stringent accuracy requirements:

- **Reactor cooling**: Accurate heat transfer prediction is essential for safety analysis.

- **Heat exchangers**: Compact heat exchanger design relies on accurate thermal predictions.
- **Steam generators**: Two-phase flows near walls present additional modelling challenges.

10.7 Concluding Remarks

This thesis has demonstrated that physics-informed machine learning provides a powerful framework for improving wall function predictions in computational fluid dynamics. By systematically incorporating physics knowledge at multiple levels—through input features, hidden layer representations, and loss function constraints—the developed models achieve accuracy approaching wall-resolved simulations while maintaining the computational efficiency required for industrial applications.

The key insight underlying this work is that physics knowledge and data-driven learning are complementary rather than competing approaches. Physics provides the structure, constraints, and interpretability that pure machine learning lacks; machine learning provides the flexibility, adaptability, and pattern recognition capabilities that analytical models cannot match. The fusion of these paradigms, exemplified by the physics-informed neural network approach, represents a promising direction for computational mechanics more broadly.

The practical impact of this research lies in enabling more accurate CFD predictions on the coarse meshes used in industrial practice. By replacing empirical wall functions with learned models that capture the complex physics of separation, adverse pressure gradients, and heat transfer, engineers can obtain more reliable predictions without the prohibitive cost of wall-resolved simulations. This capability is particularly valuable in the design optimisation loop, where many configurations must be evaluated quickly yet accurately.

Looking forward, the framework established in this thesis provides a foundation for continued development. Extension to three-dimensional flows, higher Reynolds numbers, and unsteady phenomena will expand the applicability. Integration of uncertainty quantification will enable appropriate trust calibration. And experimental validation will build confidence for deployment in safety-critical applications.

The ultimate vision is a wall modelling capability that adapts automatically to local flow con-

ditions, selecting the appropriate treatment—whether traditional, machine-learned, or hybrid—based on detected flow regime and required accuracy. The separation classifier developed in Chapter 8 represents a first step toward this adaptive wall modelling paradigm. As machine learning methods continue to mature and computational resources expand, physics-informed approaches will play an increasingly important role in making high-fidelity CFD accessible for everyday engineering practice.

Bibliography

1. A. Beck & M. Kurz. A Perspective on Machine Learning Methods in Turbulence Modelling (2020).
2. S. S. Girimaji. Turbulence closure modeling with machine learning approaches: A perspective (2023).
3. R. D. Moser, J. Kim & N. N. Mansour. Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$. *Physics of Fluids* **11,** 943–945 (1999).
4. O. Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London A* **186,** 123–164 (1895).
5. J.-X. Wang, J. Wu, J. Ling, G. Iaccarino & H. Xiao. A Comprehensive Physics-Informed Machine Learning Framework for Predictive Turbulence Modeling (2017).
6. B. E. Launder & D. B. Spalding. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering* **3,** 269–289 (1974).
7. D. B. Spalding. A single formula for the law of the wall. *Journal of Applied Mechanics* **28,** 455–458 (1961).
8. M. Wolfshtein. The velocity and temperature distribution in one-dimensional flow with turbulence augmentation and pressure gradient. *International Journal of Heat and Mass Transfer* **12,** 301–318 (1969).
9. T. von Kármán. Mechanische Ähnlichkeit und Turbulenz. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen*, 58–76 (1930).
10. R. Pirayeshshirazinezhad. SPINN: An Optimal Self-Supervised Physics-Informed Neural Network Framework (2025).
11. Z. Su, Y. Liu, S. Pan, Z. Li & C. Shen. Finite Volume Physical Informed Neural Network (FV-PINN) with Reduced Derivative Order for Incompressible Flows (2024).
12. D. M. Driver & H. L. Seegmiller. Features of a reattaching turbulent shear layer in divergent channel flow. *AIAA Journal* **23,** 163–171 (1985).
13. J. Ling, A. Kurzawski & J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics* **807,** 155–166 (2016).
14. M. Raissi, P. Perdikaris & G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378,** 686–707 (2019).
15. R. McConkey, E. Yee & F.-S. Lien. On the generalizability of machine-learning-assisted anisotropy mappings for predictive turbulence modelling (2022).
16. M. Milano & P. Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics* **182,** 1–26 (2002).
17. L. Vu-Quoc & A. Humer. Deep learning applied to computational mechanics: A comprehensive review, state of the art, and the classics (2022).
18. K. Zdybał, G. D'Alessio, G. Aversano, M. R. Malik & A. Coussement. Advancing Reacting Flow Simulations with Data-Driven Models (2022).
19. P. Schlatter & R. Örlü. Assessment of direct numerical simulation data of turbulent boundary layers. *Journal of Fluid Mechanics* **659,** 116–126 (2010).
20. L. Guastoni, P. A. Srinivasan, H. Azizpour, P. Schlatter & R. Vinuesa. On the use of recurrent neural networks for predictions of turbulent flows (2020).
21. S. Bhattacharya, M. K. Verma & A. Bhattacharya. Predictions of Reynolds and Nusselt numbers in turbulent convection using machine-learning models (2022).
22. V. C. Leite, E. Merzari, R. Ponciroli & L. Ibarra. A Study on Convolution Neural Network for Reconstructing the Temperature Field of Wall-Bounded Flows (2022).
23. M. Chatzimanolakis, P. Weber & P. Koumoutsakos. Drag Reduction in Flows Past 2D and 3D Circular Cylinders Through Deep Reinforcement Learning (2023).
24. S. Raghu, R. Nayek & V. Chalamalla. Physics Informed Neural Networks for Free Shear Flows (2024).
25. L. Prandtl. Über Flüssigkeitsbewegung bei sehr kleiner Reibung. *Verhandlungen des III. Internationalen Mathematiker-Kongresses*, 484–491 (1904).
26. L. Guastoni, A. Güemes, A. Ianiro, S. Discetti & P. Schlatter. Convolutional-network models to predict wall-bounded turbulence from wall quantities (2020).
27. P. A. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter & R. Vinuesa. Predictions of turbulent shear flows using deep neural networks (2019).

28. H. Blasius. Grenzschichten in Flüssigkeiten mit kleiner Reibung. *Zeitschrift für Mathematik und Physik* **56,** 1–37 (1908).
29. G. Krishna, M. S. Nair, P. P. Nair & A. L. S. Physics-informed Neural Networks approach to solve the Blasius function (2022).
30. M. Lee & R. D. Moser. Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *Journal of Fluid Mechanics* **774,** 395–415 (2015).
31. F. H. Clauser. Turbulent boundary layers in adverse pressure gradients. *Journal of the Aeronautical Sciences* **21,** 91–108 (1954).
32. B. S. Stratford. The prediction of separation of the turbulent boundary layer. *Journal of Fluid Mechanics* **5,** 1–16 (1959).
33. D. Coles. The law of the wake in the turbulent boundary layer. *Journal of Fluid Mechanics* **1,** 191–226 (1956).
34. Y. Mao & Y. Zhang. A Workflow for Utilizing OpenFOAM Data Structure in Physics-Informed Deep Learning Training (2024).
35. H. Li, Y. Lou & D. Xiao. Quantum machine learning for efficient reduced order modelling of turbulent flows (2025).
36. M. Breuer, N. Peller, C. Rapp & M. Manhart. Flow over periodic hills–numerical and experimental study in a wide range of Reynolds numbers. *Computers & Fluids* **38,** 433–457 (2009).
37. G.-M. Gie, Y. Hong, C.-Y. Jung & D. Lee. Singular Layer Physics-Informed Neural Network Method for Convection-Dominated Boundary Layer Problems in Two Dimensions (2023).
38. T. Anandh, D. Ghose, A. Tyagi, A. Gupta & S. Sarkar. An efficient hp-Variational PINNs framework for incompressible Navier-Stokes equations (2024).
39. B. A. Kader. Temperature and concentration profiles in fully turbulent boundary layers. *International Journal of Heat and Mass Transfer* **24,** 1541–1544 (1981).
40. Z. Y. Wang & W. W. Zhang. A unified method of data assimilation and turbulence modeling for separated flows at high Reynolds numbers (2022).
41. A. Ghaemi, A. Ebrahimi, M. Hajipour, S. M. M. Shobeiry & A. F. Lipaei. Model Predictive and Reinforcement Learning Methods for Active Flow Control of an Airfoil with Dual-point Excitation of Plasma Actuators (2025).
42. P. M. Milani, J. Ling & J. K. Eaton. Generalization of machine-learned turbulent heat flux models applied to film cooling flows (2019).
43. R. Pal, S. Mukherjee, U. Dutta & A. Choudhury. Solving Navier-Stokes Equations Using Data-free Physics-Informed Neural Networks With Hard Boundary Conditions (2025).
44. L. Jiang, Y. Cheng, K. Luo & J. Fan. PT-PINNs: A Parametric Engineering Turbulence Solver based on Physics-Informed Neural Networks (2025).
45. R. Laubscher & P. Rousseau. Application of mixed-variable physics-informed neural networks to solve normalised momentum and energy transport equations for 2D internal convective flow (2021).
46. T. Nakamura, K. Fukami & K. Fukagata. Identifying key differences between linear stochastic estimation and neural networks for fluid flow regressions (2021).
47. B. Yan, B. Chen, D. R. Harp & R. J. Pawar. A Robust Deep Learning Workflow to Predict Multiphase Flow Behavior during Geological CO2 Sequestration Injection and Post-Injection Periods (2021).
48. G. Novati, H. L. de Laroussilhe & P. Koumoutsakos. Automating Turbulence Modeling by Multi-Agent Reinforcement Learning (2020).
49. M. Schmelzer, R. P. Dwight & P. Cinnella. Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression (2019).
50. R. D. Sanhueza, S. Smit, J. Peeters & R. Pecnik. Machine Learning for RANS Turbulence Modelling of Variable Property Flows (2022).
51. C. Pedersen, L. Zanna, J. Bruna & P. Perezhogin. Reliable coarse-grained turbulent simulations through combined offline learning and neural emulation (2023).
52. Z. Li, F. Montomoli & S. Sharma. Investigation of Compressor Cascade Flow Using Physics- Informed Neural Networks with Adaptive Learning Strategy (2023).
53. J. Balla, J. Bailey, A. Backour, E. Hofgard & T. Jaakkola. Implicit Augmentation from Distributional Symmetry in Turbulence Super-Resolution (2025).
54. C. B. Millikan. A critical discussion of turbulent flows in channels and circular tubes. *Proceedings of the Fifth International Congress of Applied Mechanics*, 386–392 (1938).

55. A. Haridas, N. R. Vadlamani & Y. Minamoto. Deep Neural Networks to Correct Sub-Precision Errors in CFD (2022).
56. H. S. Wong, W. X. Chan, B. H. Li & C. H. Yap. Multiple Case Physics-Informed Neural Network for Biomedical Tube Flows (2023).
57. O. Sallam & M. Fürth. Inference of water waves surface elevation from horizontal velocity components using physics informed neural networks (PINN) (2024).
58. M. A. Elhawary. Deep Reinforcement Learning for Active Flow Control around a Circular Cylinder Using Unsteady-mode Plasma Actuators (2020).
59. M. Chu & W. Qian. Uncertainty Quantification For Turbulent Flows with Machine Learning (2023).
60. M. Chu & W. Qian. Physics-Guided Machine Learning for Uncertainty Quantification in Turbulence Models (2025).
61. M. Matha & C. Morsbach. Extending turbulence model uncertainty quantification using machine learning (2022).
62. M. Matha & C. Morsbach. Physics-constrained Random Forests for Turbulence Model Uncertainty Estimation (2023).
63. K. Fukagata & K. Fukami. Compressing fluid flows with nonlinear machine learning: mode decomposition, latent modeling, and flow control (2025).
64. R. Ma & A. Lozano-Duran. Machine-learning wall-model large-eddy simulation accounting for isotropic roughness under local equilibrium (2024).
65. P. I. Karpov, C. Huang, I. Sitdikov, C. L. Fryer & S. Woosley. Physics-Informed Machine Learning for Modeling Turbulence in Supernovae (2022).
66. N. Roy, R. Dürr, A. Bück & S. Sundar. Finite difference physics-informed neural networks enable improved solution accuracy of the Navier-Stokes equations (2024).
67. A. Man, M. Jadidi, A. Keshmiri, H. Yin & Y. Mahmoudi. Non-Unique Machine Learning Mapping in Data-Driven Reynolds Averaged Turbulence Models (2023).
68. H. S. de Ocáriz Borde, D. Sondak & P. Protopapas. Multi-Task Learning based Convolutional Models with Curriculum Learning for the Anisotropic Reynolds Stress Tensor in Turbulent Duct Flow (2021).
69. H. D. Pasinato & N. F. M. Reh. Modeling Turbulent Flows with LSTM Neural Network (2023).
70. M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki & J. Donahue. Population Based Training of Neural Networks (2017).
71. R. Maulik, D. Fytanidis, B. Lusch, V. Vishwanath & S. Patel. PythonFOAM: In-situ data analyses with OpenFOAM and Python (2021).
72. V. Brehm, J. W. Austefjord, S. Lepadatu & A. Qaiumzadeh. A proposal for leaky integrate-and-fire neurons by domain walls in antiferromagnetic insulators (2022).
73. R. Deshpande, C. M. de Silva, M. Lee, J. P. Monty & I. Marusic. Data-driven enhancement of coherent structure-based models for predicting instantaneous wall turbulence (2021).
74. M. Yin, E. Ban, B. V. Rego, E. Zhang & C. Cavinato. Simulating progressive intramural damage leading to aortic dissection using an operator-regression neural network (2021).
75. T. Murata, K. Fukami & K. Fukagata. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics (2019).
76. E. A. Illarramendi, M. Bauerheim & B. Cuenot. Performance and accuracy assessments of an incompressible fluid solver coupled with a deep Convolutional Neural Network (2021).
77. Y. Shu, Z. Cao, J. Gao, J. Wang & P. S. Yu. Omni-Training: Bridging Pre-Training and Meta-Training for Few-Shot Learning (2021).
78. D. de Oliveira Maionchi, L. Ainstein, F. P. dos Santos & M. B. de Souza Júnior. Computational Fluid Dynamics and Machine Learning as tools for Optimization of Micromixers geometry (2022).
79. H. Frezat, J. L. Sommer, R. Fablet, G. Balarac & R. Lguensat. A posteriori learning for quasi-geostrophic turbulence parametrization (2022).
80. S. Hu, M. Liu, S. Zhang, S. Dong & R. Zheng. Physics-informed Neural Network Combined with Characteristic-Based Split for Solving Navier-Stokes Equations (2023).
81. Y. Zou, T. Li, L. Lu, J. Wang & S. Zou. Finite-difference-informed graph network for solving steady-state incompressible flows on block-structured grids (2024).
82. N. McGreivy & A. Hakim. Convolutional layers are equivariant to discrete shifts but not continuous translations (2022).

83. A. Rybchuk, M. Hassanaly, N. Hamilton, P. Doubrawa & M. J. Fulton. Ensemble flow reconstruction in the atmospheric boundary layer from spatially limited measurements through latent diffusion models (2023).
84. D. Dulal, J. J. Charney, M. Gallagher, C. Navasca & N. Skowronski. Exploring and Analyzing Wildland Fire Data Via Machine Learning Techniques (2023).
85. D. Wang, Z. Liang, Z. Zhang & M. Li. Efficient Estimation of the Convective Cooling Rate of Photovoltaic Arrays with Various Geometric Configurations: a Physics-Informed Machine Learning Approach (2024).
86. Z. Shi, S. M. H. Khorasani, H. Shin, J. Yang & S. Lee. Drag prediction of rough-wall turbulent flow using data-driven regression (2024).
87. N. Ashton, D. C. Maddix, S. Gundry & P. M. Shabestari. AhmedML: High-Fidelity Computational Fluid Dynamics Dataset for Incompressible, Low-Speed Bluff Body Aerodynamics (2024).
88. L. de Vito, N. Pinnau & S. Dey. Implicit Neural Representation For Accurate CFD Flow Field Prediction (2024).
89. J. Suk, C. Brune & J. M. Wolterink. SE(3) symmetry lets graph neural networks learn arterial velocity estimation from small datasets (2023).
90. L. S. E. Jessica, N. A. Arafat, W. X. Lim, W. L. Chan & A. W. K. Kong. Finite Volume Features, Global Geometry Representations, and Residual Training for Deep Learning-based CFD Simulation (2023).
91. H. Tang, Y. Wang, T. Wang & L. Tian. Discovering explicit Reynolds-averaged turbulence closures for turbulent separated flows through deep learning-based symbolic regression with non-linear corrections (2023).
92. A. Sedykh, M. Podapaka, A. Sagingalieva, K. Pinto & M. Pflitsch. Hybrid quantum physics-informed neural networks for simulating computational fluid dynamics in complex shapes (2023).
93. T. Yao, E. Pajaziti, M. Quail, S. Schievano & J. A. Steeden. Image2Flow: A hybrid image and graph convolutional neural network for rapid patient-specific pulmonary artery segmentation and CFD flow field calculation from 3D cardiac MRI data (2024).
94. J. Cai, P.-E. Angeli, J.-M. Martinez, G. Damblin & D. Lucor. Revisiting Tensor Basis Neural Networks for Reynolds stress modeling: application to plane channel and square duct flows (2024).
95. H. Vardhan, U. Timalsina, M. Sandborn, D. Hyde & P. Volgyesi. Anvil: An integration of artificial intelligence, sampling techniques, and a combined CAD-CFD tool (2024).
96. A. H. Nobari, J. Rey, S. Kodali, M. Jones & F. Ahmed. Conformal Predictions Enhanced Expert-guided Meshing with Graph Neural Networks (2023).
97. Z. Wang, X. Meng, X. Jiang, H. Xiang & G. E. Karniadakis. Solution multiplicity and effects of data and eddy viscosity on Navier-Stokes solutions inferred by physics-informed neural networks (2023).
98. M. Elrefaie, S. Hüttig, M. Gladkova, T. Gericke & D. Cremers. Real-time and On-site Aerodynamics using Stereoscopic PIV and Deep Optical Flow Learning (2024).
99. M. Naderibeni, M. J. T. Reinders, L. Wu & D. M. J. Tax. Learning solutions of parametric Navier-Stokes with physics-informed neural networks (2024).
100. A. Warey, T. Han & S. Kaushik. Investigation of Numerical Diffusion in Aerodynamic Flow Simulations with Physics Informed Neural Networks (2021).
101. J. Sirignano, J. MacArt & K. Spiliopoulos. PDE-constrained Models with Neural Network Terms: Optimization and Global Convergence (2021).
102. X.-H. Zhou, J. Han & H. Xiao. Learning nonlocal constitutive models with neural networks (2020).
103. Z. Zhou, G. He & X. Yang. A wall model based on neural networks for LES of turbulent flows over periodic hills (2020).
104. M. Morimoto, K. Fukami, K. Zhang, A. G. Nair & K. Fukagata. Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low-dimensionalization (2021).
105. C. Jiang. Physics-guided deep learning framework for predictive modeling of the Reynolds stress anisotropy (2021).
106. A. Arzani, J.-X. Wang & R. M. D'Souza. Uncovering near-wall blood flow from sparse data with physics-informed neural networks (2021).
107. I. K. Deo & R. Jaiman. Predicting waves in fluids with deep neural network (2022).
108. S. Goraya, N. Sobh & A. Masud. Error Estimates and Physics Informed Augmentation of Neural Networks for Thermally Coupled Incompressible Navier Stokes Equations (2022).
109. R. Gao & R. K. Jaiman. Predicting fluid-structure interaction with graph neural networks (2022).
110. J. Suk, P. de Haan, P. Lippe, C. Brune & J. M. Wolterink. Mesh Neural Networks for SE(3)-Equivariant Hemodynamics Estimation on the Artery Wall (2022).

111. S. Ephrati, P. Cifani & B. Geurts. Data-driven spectral turbulence modeling for Rayleigh-Bénard convection (2023).
112. B. Schulz & S. Lerch. Machine learning methods for postprocessing ensemble forecasts of wind gusts: A systematic comparison (2021).
113. J. Cai, P.-E. Angeli, J.-M. Martinez, G. Damblin & D. Lucor. Reynolds Stress Anisotropy Tensor Predictions for Turbulent Channel Flow using Neural Networks (2022).
114. R. Mao, M. Lin, Y. Zhang, T. Zhang & Z.-Q. J. Xu. DeepFlame: A deep learning empowered open-source platform for reacting flow simulations (2022).
115. T. Nakamura & K. Fukagata. Robust training approach of neural networks for fluid flow state estimations (2021).
116. S. Rodriguez & P. Cardiff. A general approach for running Python codes in OpenFOAM using an embedded pybind11 Python interpreter (2022).
117. C.-W. Chang & N. T. Dinh. Reynolds-Averaged Turbulence Modeling Using Type I and Type II Machine Learning Frameworks with Deep Learning (2018).
118. H. Ghraieb, J. Viquerat, A. Larcher, P. Meliga & E. Hachem. Single-step deep reinforcement learning for open-loop control of laminar and turbulent flows (2020).
119. Z. Xiang, W. Peng, X. Zheng, X. Zhao & W. Yao. Self-adaptive loss balanced Physics-informed neural networks for the incompressible Navier-Stokes equations (2021).
120. X. He, Y. Wang & J. Li. Flow Completion Network: Inferring the Fluid Dynamics from Incomplete Flow Information using Graph Neural Networks (2022).
121. M. Quattromini, M. A. Bucci, S. Cherubini & O. Semeraro. Enhancing Data-Assimilation in CFD using Graph Neural Networks (2023).
122. R. Han, Y. Wang, Y. Zhang & G. Chen. A new prediction method of unsteady wake flow by the hybrid deep neural network (2019).
123. A. Scillitoe, P. Seshadri & M. Girolami. Uncertainty Quantification for Data-driven Turbulence Modelling with Mondrian Forests (2020).
124. C. Xie, X. Xiong & J. Wang. Artificial neural network approach for turbulence models: A local framework (2021).
125. J. Han, X.-H. Zhou & H. Xiao. An equivariant neural operator for developing nonlocal tensorial constitutive models (2022).
126. A. Vahdat & J. Kautz. NVAE: A Deep Hierarchical Variational Autoencoder (2020).
127. C. Wang, Q. Gao, B. Wang, C. Pan & J. Wang. Vortex-to-velocity reconstruction for wall-bounded turbulence via a data-driven model (2020).
128. M. I. Zafar, H. Xiao, M. M. Choudhari, F. Li & C.-L. Chang. Convolutional Neural Network for Transition Modeling Based on Linear Stability Theory (2020).
129. M. Chu & W. Qian. Machine learning based uncertainty quantification of turbulence model for airfoils (2022).
130. X. Xue, S. Wang, H.-D. Yao, L. Davidson & P. V. Coveney. Physics informed data-driven near-wall modelling for lattice Boltzmann simulation of high Reynolds number turbulent flows (2024).
131. M. I. Zafar, X. Zhou, C. J. Roy, D. Stelter & H. Xiao. Data-Driven Turbulence Modeling Approach for Cold-Wall Hypersonic Boundary Layers (2024).
132. M. Matha & K. Kucharczyk. Applicability of machine learning in uncertainty quantification of turbulence models (2022).
133. A. Prakash, K. E. Jansen & J. A. Evans. Invariant Data-Driven Subgrid Stress Modeling on Anisotropic Grids for Large Eddy Simulation (2022).
134. A. G. Özbay & S. Laizet. FR3D: Three-dimensional Flow Reconstruction and Force Estimation for Unsteady Flows Around Extruded Bluff Bodies via Conformal Mapping Aided Convolutional Autoencoders (2023).
135. X. Fu, S. Fu, C. Liu, M. Zhang & Q. Hu. Data-driven approach for modeling Reynolds stress tensor with invariance preservation (2023).
136. B. Font, F. Alcántara-Ávila, J. Rabault, R. Vinuesa & O. Lehmkuhl. Active flow control of a turbulent separation bubble through deep reinforcement learning (2024).
137. M. Fiore, E. Saccaggi, L. Koloszar, Y. Bartosiewicz & M. A. Mendez. Data-driven turbulent heat flux modeling with inputs of multiple fidelity (2024).
138. S. Barwey, R. Balin, B. Lusch, S. Patel & R. Balakrishnan. Scalable and Consistent Graph Neural Networks for Distributed Mesh-based Data-driven Modeling (2024).

139. P. Garnier, V. Lannelongue, J. Viquerat & E. Hachem. MeshMask: Physics-Based Simulations with Masked Graph Neural Networks (2025).
140. A. M. Santamaria, T. Buchanan, F. Fico, I. Langella & R. P. Dwight. Data-driven turbulence modelling for magnetohydrodynamic flows in annular pipes (2025).
141. C. Wu, S. Zhang, C. Guo & Y. Zhang. Data-driven Turbulence Modeling for Separated Flows Considering Non-Local Effect (2025).
142. J. Afful. A Review of HPC-Accelerated CFD in National Security and Defense (2025).
143. Z. Cao, M. Li, F. Lin, J. Jia & Y. Wen. Transforming Future Data Center Operations and Management via Physical AI (2025).
144. K. Buck & R. Temam. Convergence Properties of PINNs for the Navier-Stokes-Cahn-Hilliard System (2025).
145. M. Yagoubi, D. Danan, M. Leyli-Abadi, A. Mazari & J.-P. Brunet. NeurIPS 2024 ML4CFD Competition: Results and Retrospective Analysis (2025).
146. A. S. Nair, N. Singh, M. Panesi, J. Sirignano & J. F. MacArt. Physics-Based Machine Learning Closures and Wall Models for Hypersonic Transition-Continuum Boundary Layer Predictions (2025).
147. N. Wang, Y. Chen & S. Zheng. FluidFormer: Transformer with Continuous Convolution for Particle-based Fluid Simulation (2025).
148. N. Shah. Computational Fluid Dynamics Optimization of F1 Front Wing using Physics Informed Neural Networks (2025).
149. M. Ahangarkiasari & H. Pouraria. Multi-Stage Graph Neural Networks for Data-Driven Prediction of Natural Convection in Enclosed Cavities (2025).
150. H. Geshani, M. R. Dehkordi & M. S. Panahi. Physics-Informed Machine Learning Approach in Augmenting RANS Models Using DNS Data and DeepInsight Method on FDA Nozzle (2025).
151. X. Liu, T. Hickling & J. F. MacArt. Active Control of Turbulent Airfoil Flows Using Adjoint-based Deep Learning (2025).
152. R. Falga, S. Shamekh & L. Zanna. Towards a Unified Data-Driven Boundary Layer Momentum Flux Parameterization for Ocean and Atmosphere (2025).
153. B. Choi, M. Ugliotti, M. Reynoso, D. R. Gurevich & R. O. Grigoriev. Data-driven modeling of multiscale phenomena with applications to fluid turbulence (2025).
154. X. K. Lee, A. Ramadhan, A. Souza, G. L. Wagner & S. Silvestri. NORi: An ML-Augmented Ocean Boundary Layer Parameterization (2025).
155. B. Barman, D. Chatterjee & R. K. Ray. PhysicsFormer: An Efficient and Fast Attention-Based Physics Informed Neural Network for Solving Incompressible Navier Stokes Equations (2026).
156. T. Morimoto. Rapid Prediction of Three-Dimensional Scour Flow around Bridge Piers via Body-Fitted Coordinate-Based U-Net (2026).
157. P. Cinnella. Data-driven turbulence modeling (2024).
158. K. Agyei-Baah, M. R. Rahman & E. R. Smith. A Convolutional Neural Network Based Framework for Linear Fluid Dynamics (2026).
159. F. Orozco, P. P. B. de Gusmão, H. Wen, J. Wahlström & M. Luo. Federated Learning for Traffic Flow Prediction with Synthetic Data Augmentation (2024).
160. P. Garnier, J. Viquerat & E. Hachem. Multi-Grid Graph Neural Networks with Self-Attention for Computational Mechanics (2024).
161. I.-G. Farcas, D. L. C. A. Fernando, A. B. Navarro, G. Merlo & F. Jenko. Machine Learning for Electron-Scale Turbulence Modeling in W7-X (2025).
162. J. Suk, D. Alblas, B. A. Hutten, A. Wiegman & C. Brune. Physics-informed graph neural networks for flow field estimation in carotid arteries (2024).
163. S. Dutta, N. Innan, S. B. Yahia & M. Shafique. AQ-PINNs: Attention-Enhanced Quantum Physics-Informed Neural Networks for Carbon-Efficient Climate Modeling (2024).
164. C. Wu, S. Zhang & Y. Zhang. Data-driven Augmentation of a Turbulence Model in Three-dimensional Separated Flows (2025).
165. S. Mukherjee. Graph Neural Network Assisted Genetic Algorithm for Structural Dynamic Response and Parameter Optimization (2025).
166. A. Wu & S. K. Lele. Subgrid Stress Modelling with Multi-dimensional State Space Sequence Models (2025).
167. Y. Ling, I. Hayat, K. Goc & A. Lozano-Duran. General-purpose Data-driven Wall Model for Low-speed Flows Part I: Baseline Model (2025).

168. K. Bounja, L. Laayouni & A. Sakat. KD-PINN: Knowledge-Distilled PINNs for ultra-low-latency real-time neural PDE solvers (2025).
169. H. Wen, F. Luo, S. Xu & B. Wang. JANC: A cost-effective, differentiable compressible reacting flow solver featured with JAX-based adaptive mesh refinement (2025).
170. M. B. Hasan, M. Yu & T. Oates. Invariance-embedded Machine Learning Sub-grid-scale Stress Models for Meso-scale Hurricane Boundary Layer Flow Simulation I: Model Development and *a priori* Studies (2025).
171. M. H. Parikh, X. Fan & J.-X. Wang. Conditional flow matching for generative modeling of near-wall turbulence with quantified uncertainty (2025).
172. D. Igarashi, S. Kumagai, Y. Yokoyama, Y. Jingzu & M. Horie. Reconstruction of three-dimensional fluid stress field via photoelasticity using physics-informed convolutional encoder-decoder (2025).
173. K. Jigjid, A. Eidi, N. A. K. Doan & R. P. Dwight. Discovery of a Physically Interpretable Data-Driven Wind-Turbine Wake Model (2025).
174. Z. Yang, Y. Bin, Y. Shi & X. I. A. Yang. Large Language Model Driven Development of Turbulence Models (2025).
175. E. R. Van Driest. On turbulent flow near a wall. *Journal of the Aeronautical Sciences* **23,** 1007–1011 (1956).