Learning to Compress: Deep Learning for Storage-Efficient Unsteady Adjoint-Based Error Estimation

A Framework for Compression of the Residuals

Amirreza Sadat Madani

TU Delft

Learning to Compress: Deep Learning for Storage-Efficient Unsteady Adjoint-Based Error Estimation

A Framework for Compression of the Residuals

by

Amirreza Sadat Madani

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday, 14th November 2025 at 2:00 PM.

Student Number:	5785472	
Project Duration:	January 2025 - October 2025	
Thesis Committee:	Dr. Steven J. Hulshoff,	TU Delft, Supervisor
	Ir. Thomas P. Hunter,	TU Delft, Daily Supervisor
	Dr. Nguyen Anh Khoa Doan,	TU Delft, Committee Chair
	Dr. Richard P. Dwight,	TU Delft
Faculty:	Faculty of Aerospace Engineering	

Cover: Distorted contour of the wall-normal velocity field from 1D DNS-forced viscous unsteady Burgers' simulation of a turbulent channel flow

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

I vividly remember my first encounter with the concept of machine learning. It was during the second semester of my bachelor's studies in the course *Introduction to Programming*. The professor introduced us to machine learning through a simple, almost funny story:

> "One day, a person asks a machine, *what is 2 + 2?* The machine replies, *3*. The person corrects it and says, *No, it's 4.* Then asks again, *what is 2 + 2?* This time, the machine answers *4*. Encouraged, the person asks, *what is 2 + 3?* And the machine proudly says, *4*!!!"

That story made me smile, but it also sparked something in me. From that moment, I became curious about how machines can learn from mistakes and improve over time. Since then, I have been learning about machine learning through online courses and by integrating it into my projects, especially since there are still very few opportunities to study it outside of computer science programs.

Years later, I am grateful to have completed my Master's thesis on a topic that lies at the intersection of my greatest interests: **Computational Fluid Dynamics** (**CFD**) and **Machine Learning**. Over the course of this work, I expanded my knowledge and had the privilege of spending an inspiring period at the *AI for Fluid Mechanics* research group in the *Aerodynamics Department* of the *Faculty of Aerospace Engineering* at *TU Delft*.

I would like to sincerely thank my supervisors, **Dr. Steven J. Hulshoff** and **Ir. Thomas P. Hunter**, for their outstanding mentorship, understanding, and unwavering support during the course of this research. I am especially thankful for their understanding, as I had to learn many fundamental topics to carry out this work, and for their encouragement during the many challenges I faced along the way. I want to thank **Steven** in particular for his motivating words during our meetings, sometimes I would find myself counting the days until the next one, just to have a five-minute chat after our long thesis discussions that always left me inspired. I am also very grateful to **Thomas** for his continuous support whenever I encountered difficulties, and for his thoughtful feedback and deep discussions that pushed me to think critically about each concept.

I am deeply grateful to my friends from my Master's, **Affan, Dominik, Adrian, Sahir, Gleb, Jan**, and many others I met along the way. Their company, conversations, and countless shared moments of laughter and coffee breaks made this journey both inspiring and joyful. Outside the faculty, I want to thank my friends **Ferrari, Pouria, Elyas, Toranj, Roham,** and **Parsa** for always being there and supporting me through every challenge of living abroad.

Lastly, I owe my deepest gratitude to my parents and my little brother, **Panam**, for their endless love and support. I cannot put my feelings into words because I know that without them, I could not have reached where I am today. And to my beloved partner, **Mahtab**, thank you for being my source of calm and motivation through all the rainy days.

Amirreza Sadat Madani
Delft, November 2025

Summary

Adjoint-based error estimation is among the most accurate methodologies for quantifying numerical errors, forming the foundation of Adaptive Mesh Refinement (AMR) strategies in turbulent flow simulations, particularly within Large Eddy Simulations (LES). However, in unsteady problems, the need to access fully time-resolved primal fields during the backward-in-time adjoint computation, along with the corresponding injected residuals, leads to prohibitive storage requirements. This thesis addresses this bottleneck by developing and assessing data-driven compression frameworks for both injected-residual and primal fields, aiming to substantially reduce storage demands while preserving the accuracy and consistency of adjoint-based error estimation.

Two machine learning–based surrogate modeling techniques, the Convolutional AutoEncoder (CAE) and the Echo State Network (ESN), were investigated to assess their ability to capture the spatial and temporal dynamics of the compressed fields, respectively. Their performance was compared against Proper Orthogonal Decomposition (POD), which served as the benchmark method. The computational framework was implemented in OpenFOAM for both primal and adjoint solvers, while the compression and reconstruction models were trained and evaluated in Python using `PyTorch` and `ReservoirPy` packages. Two test cases were considered: a smooth Manufactured Solution (MMS) of the 1D viscous Burgers' equation, used to verify the framework and ensure consistency under controlled conditions, and a DNS-forced 1D viscous Burgers' problem derived from Turbulent Channel Flow (TCF) data, used to evaluate the method's robustness in representing complex, unsteady turbulent flow dynamics.

The MMS results confirmed that the framework can accurately reconstruct both the primal and injected-residual fields without altering the adjoint-based output error estimation. When the compressed residuals were used in the adjoint computation, the recovered error indicators were in close agreement with the fully resolved reference, verifying that the compression and reconstruction stages preserve consistency. Minor differences were observed only when surrogate primals were employed for adjoint solution computation, yet these deviations remained negligible, confirming the robustness of the overall formulation before applying it to more complex unsteady cases.

For the DNS-forced 1D viscous Burgers problem, representing the wall-normal velocity fluctuations of a TCF, the framework was evaluated under more realistic, nonlinear, and temporally evolving conditions. In this case, the ESN demonstrated the highest capability in reconstructing temporally varying residuals and maintaining consistent adjoint sensitivities, benefiting from its recurrent dynamics. Across all refinement levels, the time-averaged adjoint-based error indicators remained closely aligned with the reference, even at compression ratios exceeding two orders of magnitude, confirming the framework's ability to drastically reduce storage without compromising estimation accuracy. The CAE effectively captured spatial features but showed mild instability and localized artifacts when surrogate primal reconstructions were employed for adjoint computation. While the CAE provided better reconstruction of mean values, the ESN more accurately reproduced the second-moment statistics, leading to improved representation of unsteady dynamics. The POD, used as a benchmark, yielded stable reconstructions but was limited in representing nonlinear temporal behavior and performed notably worse than the ESN when compared at equivalent compression ratios.

Overall, the results demonstrate that the proposed compression framework can reliably reduce storage demands in unsteady adjoint-based error estimation while preserving high fidelity in the output error indicators. The findings emphasize that while POD remains the most stable baseline, CAE provides strong nonlinear spatial encoding and ESN achieves the most accurate temporal reconstruction. The framework establishes a practical and scalable foundation for applying adjoint-based AMR in LES and other turbulent flow analyses where unsteady data storage remains a critical limitation.

Contents

List of Figures

List of Tables

List of Symbols

Latin Symbols

b	Bias Term
$\mathcal{B}$	Batch
c	Channel Size
E	Energy Spectrum
$\mathcal{E}$	Modal Energy
f	Context 1: Primal Source-term Function Context 2: Activation Function
g	Context 1: Quantity of Interest Function Context 2: Adjoint Source-Term Function
H	Coarse Space
h	Fine Space
I	Context 1: Interpolation Operator Context 2: Time Domain
J	Quantity of Interest
k	Kernel Size
L	Context 1: Left Singular Matrix Context 2: Sequence Length Context 3: Space-domain Length
$\mathcal{L}$	Primal Differential Operator
$\mathcal{L}^*$	Adjoint Differential Operator
$\mathcal{M}$	Computational Mesh
$\mathcal{N}$	Gaussian Noise
N	Number of Elements
$\mathcal{O}$	Order of Magnitude
p	Context 1: Padding Context 2: Pressure
r	Reservoir State
R	Context 1: Residual Context 2: Right Singular Matrix
s	Stride
t	Time Coordinate
T	Time-domain Length
u	Context 1: Primal Solution in x-coordinate Context 2: Network Input
U	Context 1: Primal Solution Context 2: Snapshot Matrix
v	Primal Solution in y-coordinate
W	Weight Matrix
x	Streamwise Coordinate
y	Context 1: Wall-normal Coordinate Context 2: Network Output
z	Spanwise Coordinate
$\mathcal{Z}$	Latent Space

Greek Symbols

α	Context 1: Positive Scalar of Advection Speed Context 2: Leaking Rate
β	Tikhonov Regularization Parameter
ξ	Wave Number Magnitude
Δ	Context 1: Numerical Step Context 2: Filter Width
δ	Context 1: Infinitesimal Variation Context 2: Channel Half-height
ϵ	Context 1: Error Tolerance Context 2: Error Indicator
ψ	Adjoint Solution
Ω	Spatial Domain
ρ	Context 1: Spectral Radius Context 2: Density
Σ	Diagonal Matrix of Singular Values
σ	Standard Deviation
λ	Eigenvalue
γ	Learning Rate
ν	Kinematic Viscosity
μ	Mean Value

List of Abbreviations

Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
AMR	Adaptive Mesh Refinement
ANN	Artificial Neural Network
ARD	Automatic Relevance Determination
BC	Boundary Conditions
BN	Batch Normalization
BO	Bayesian Optimization
BP	Back Propagation
CAE	Convolutional Autoencoder
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
CPR	Correction Procedure via Reconstruction
CPU	Central Processing Unit
CR	Compression Ratio
DES	Detached Eddy Simulation
DG	Discontinuous Galerkin
DMD	Dynamic Mode Decomposition
DNS	Direct Numerical Simulation
DWR	Dual-Weighted Residual
EI	Expected Improvement
ESN	Echo State Network
ESP	Echo State Property
FEM	Finite Element Method
FFNN	Feed-Forward Neural Networks
FIML	Field-Inversion Machine Learning
FVM	Finite Volume Method
$\mathcal{GP}$	Gaussian Process
GPU	Graphics Processing Unit
GS	Grid Search
HDG	Hybridizable Discontinuous Galerkin
HIT	Homogeneous Isotropic Turbulence
HLE	Highest Lyapunov Exponent
IC	Initial Condition
iSVD	incremental SVD
LES	Large Eddy Simulation
LReLU	Leaky Rectified Linear Unit
LT	Lyapunov Time
ML	Machine Learning
MSE	Mean Squared Error
NS	Navier-Stokes
PCA	Principal Component Analysis
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
POD	Proper Orthogonal Decomposition
QoI	Quantity of Interest
RANS	Reynolds-Averaged Navier-Stokes
RC	Reservoir Computing
Re	Reynolds numbers
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
ROM	Reduced-Order Model
ROR	Reduced-Order Representation
RQ	Research Question
SA	Spalart-Allmaras (Turbulence Model)
SGS	Subgrid-Scale
SRNN	Super-Resolution Neural Network
STE	Source Term Estimation
SVD	Singular Value Decomposition
TCF	Turbulent Channel Flow
1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional

1

Introduction

Computational Fluid Dynamics (CFD) has become a foundational tool in aerospace engineering, enabling the simulation and analysis of fluid flows around complex geometries [1]. The ability of CFD to visualize flow fields, predict surface load distributions, and calculate aerodynamic forces and moments is essential for the preliminary design of aerospace configurations [2]

The increasing adoption of CFD in the aerospace industry is closely linked to advances in computational power and numerical methods, which have made it possible to model complex physical processes such as turbulence and transition with greater accuracy [3]. Aerospace was among the first industries to make extensive use of simulation technologies, relying on CFD to design without the need for costly experimental facilities [4]. This shift has significantly reduced the number of wind tunnel tests required, leading to shorter design cycles and lower development costs [2].

CFD is routinely used to predict aerodynamic forces, structural loads, and thermal loads, as well as to analyze processes such as combustion, noise, and vibrations under various flow conditions [5]. The integration of advanced turbulence models and high-order numerical algorithms has further improved the accuracy and efficiency of CFD simulations in aerospace applications [6].

Despite these advances, CFD simulations remain computationally expensive and can still be inaccurate for highly nonlinear, unsteady, or turbulent flows. The resolution requirements needed to capture all relevant spatial and temporal scales make high-fidelity simulations, such as Direct Numerical Simulation (DNS) or Large Eddy Simulation (LES), computationally demanding, particularly for realistic three-dimensional (3D) geometries [7]. Moreover, the accuracy of CFD predictions often depends on the quality of turbulence modeling, grid resolution, and numerical discretization, which remain sources of uncertainty in practical applications [7].

1.1. Computational Fluid Dynamics for Turbulence

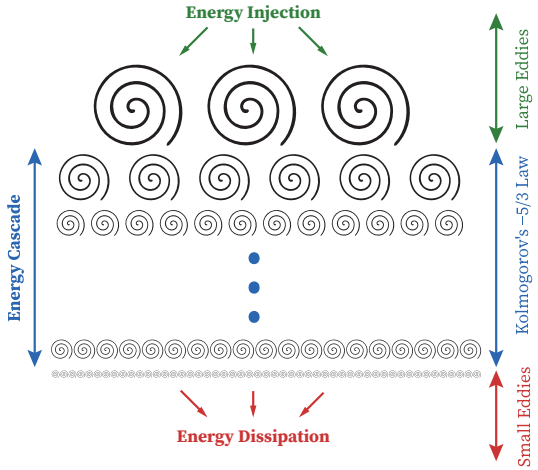
Turbulence is a complex and chaotic state of fluid motion characterized by vorticity, irregular fluctuations, and energy dissipation over multiple scales [8]. Unlike laminar flows, where momentum transfer occurs in an orderly and predictable manner governed primarily by viscous effects, turbulent flows exhibit strong nonlinear interactions that disrupt this organization, leading to a continuous redistribution of energy across different length scales. This phenomenon can be modeled by the Navier–Stokes (NS) equations [9], in which the nonlinear convective term is responsible for the energy transfer from large to small scales [10].

The concept of the energy cascade, introduced by Kolmogorov [11], explains how kinetic energy injected at large scales is transferred down to progressively smaller scales until it is dissipated by viscosity, as schematically visualized in Figure 1.1a. In a turbulent flow, energy is typically introduced at the largest eddy scales, corresponding to the external forcing or boundary conditions, and is then transferred through an inertial subrange where viscosity is negligible. This inertial subrange follows a self-similar distribution, characterized by Kolmogorov's $-5/3$ law for the energy spectrum $E(\xi)$ [11].

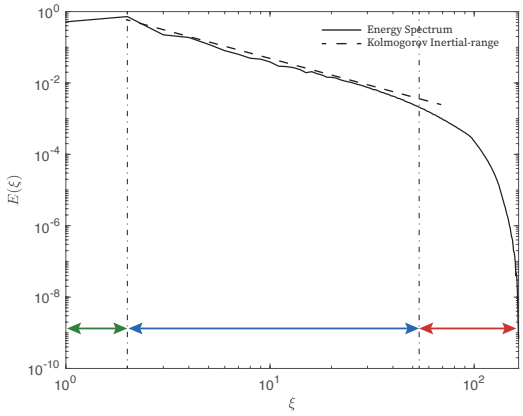
Eventually, at sufficiently small scales, the energy reaches the Kolmogorov microscale, where viscous dissipation dominates, converting turbulent kinetic energy into heat [12].

The energy spectrum plot in Figure 1.1b illustrates the distribution of turbulent kinetic energy across different wave numbers ξ. The spectrum obtained from a DNS of Homogeneous Isotropic Turbulence (HIT) reveals three distinct regions:

1. **Energy Injection Range:** At low wave numbers, corresponding to length scales on the order of the integral scale, energy is supplied to the flow through external forcing mechanisms such as shear or buoyancy-driven turbulence.
2. **Inertial Subrange:** Energy transfer is governed by nonlinear interactions without significant viscous effects, leading to Kolmogorov's $-5/3$ scaling law $E(\xi) \propto \xi^{-5/3}$.
3. **Dissipation Range:** At high wave numbers, small eddies experience strong viscous forces, leading to the conversion of kinetic energy into thermal energy and a rapid drop in the energy spectrum.



(a) Schematic of the turbulent energy cascade, illustrating energy injection at large eddy scales, transfer through intermediate scales, and dissipation at small eddy scales.



(b) Energy spectrum $E(\xi)$ from a DNS of HIT plotted against wave number magnitude ξ. The spectrum shows the energy injection range, Kolmogorov's $-5/3$ inertial subrange, and the dissipation range at high wave numbers. Computed by the author as part of the *CFD3: Large Eddy Simulations* course individual assignment at TU Delft

Figure 1.1: Turbulent energy cascade and energy spectrum

Turbulent flows' broad spectrum of spatial and temporal scales makes their DNS highly computationally demanding, especially for high Reynolds number flows. Solving the NS equations while resolving all turbulence scales is infeasible in most practical applications. Instead, turbulence modeling approaches vary in fidelity, ranging from DNS to Reynolds-Averaged Navier-Stokes (RANS) simulations. This hierarchy of turbulence models is depicted in Figure 1.2, where DNS represents the most accurate but computationally intensive method, while RANS provides a more empirical and cost-effective alternative.

Figure 1.2: Turbulence modeling approaches spectrum

DNS involves solving the NS equations on a high-resolution mesh, ensuring that all relevant spatial and temporal scales of turbulence are fully resolved [13]. This approach provides highly accurate representations of turbulent flow physics. However, the computational cost of DNS becomes prohibitive for 3D problems, particularly at high Reynolds numbers (Re), due to the exponential growth in resource re-

quirements with increasing flow complexity. To simulate all turbulent fluctuations, including the most intense ones, computational demands scale as Re^4 rather than the traditionally assumed Re^3 based on Kolmogorov's theory [14]. As a result, DNS is often limited to academic studies and benchmark cases rather than practical engineering applications.

The RANS method is extensively utilized in aerospace engineering [15], automotive engineering [16], and environmental engineering [17] thanks to its computational efficiency and effectiveness in predicting time-averaged flow characteristics [18]. The RANS framework employs Reynolds decomposition, wherein the flow variables are split into mean and fluctuating components. The resulting Reynolds stress terms are not directly solved but are instead approximated using turbulence models that rely on ensemble-averaged quantities [19]. Despite its advantages, RANS methods exhibit significant limitations in accurately predicting flows that are highly unsteady or involve complex phenomena such as mixing layers, large pressure gradients [20], or flow separation and reattachment [21]. In such cases, the assumptions underlying RANS models often lead to inaccuracies, making them less reliable compared to more advanced simulation techniques.

LES offers an alternative to traditional turbulence modeling by directly resolving the large, energy-containing eddies of the flow while modeling the effects of smaller, less influential scales through subgrid-scale (SGS) models [7]. This method has demonstrated strong performance in cases where the computational mesh is sufficiently detailed to capture the dominant turbulent structures [22]. However, achieving both accuracy and computational efficiency with LES in practical applications requires meeting specific conditions, such as fine mesh resolution, appropriate time integration schemes, or effective SGS modeling strategies [7]. When these challenges are properly addressed, LES has the potential to outperform and even replace RANS approaches for many flow configurations; however, its widespread application still demands strategies for mesh adaptation to efficiently capture local flow features [7].

1.2. Adaptive Mesh Refinement

LES of high-Reynolds-number, wall-bounded flows demand exceptionally fine grid resolutions, particularly within the inner regions of boundary layers [7]. This requirement often approaches the computational intensity of DNS, making LES computationally prohibitive for practical applications [22]. To mitigate computational costs without compromising accuracy, it is essential to develop high-quality computational meshes that must be sufficiently refined to capture pertinent flow structures across various physical regions while remaining computationally feasible.

NASA's 2030 CFD Vision report [23] specifically highlights mesh generation and adaptivity as major bottlenecks in the present-day CFD workflow. Even for relatively standard simulations, concerns persist regarding the robustness of CFD methods in accurately predicting Quantities of Interest (QoI). A notable example is the American Institute of Aeronautics and Astronautics (AIAA) Drag Prediction Workshop (DPW) [24, 25], where force and moment outputs for a representative set of wing-body geometries and flow conditions were compared across CFD codes used in industry, government laboratories, and academia. Despite advancements in computational power, results from multiple DPW submissions have consistently exhibited significant variations in computed outputs. For instance, in the third DPW, drag coefficient discrepancies of 0.0025, equivalent to a relative error of $\approx 10\%$, were observed for a DLR-F6 wing-body configuration, corresponding to a payload difference of over 100 passengers on a large transport aircraft [26].

A key factor contributing to these discrepancies is the quality of the computational mesh, which heavily depends on user expertise [26]. Even experienced CFD engineers often struggle to generate meshes that achieve an optimal balance between predictive accuracy and computational efficiency, given the multiscale nature and complexity of turbulent flows [27]. As a result, multiple trial-and-error iterations are typically required to determine appropriate refinement regions, significantly impeding the practical implementation of LES [28].

This issue becomes even more pronounced in complex and highly variable flow conditions, such as those involving separations, wakes, or shock–boundary layer interactions, where uniform or pre-generated meshes may fail to provide adequate local resolution. Such limitations can lead to either insufficient accuracy in critical regions or excessive refinement in less important areas, resulting in unnecessary

computational costs. Further emphasizing this challenge, tests conducted in the third DPW revealed that even for simplified wing-only geometries, results obtained using the same CFD code and identical turbulence models but generated on independently constructed meshes failed to converge consistently, even with uniform mesh refinement [25]. These findings underscore the critical influence of mesh design on CFD accuracy and highlight the need for adaptive mesh refinement strategies to achieve more reliable aerodynamic predictions.

Thus, a suitable grid-adaptation technique is required, which is the process of refining an existing computational grid based on the solution obtained, aiming to generate a more optimized mesh for the given problem. Specifically, it can be viewed as a multi-objective optimization challenge, where the goal is to determine the spatial distribution $\Delta(x)$ that minimizes error while using the least number of grid points [27].

In numerous scenarios, $\Delta(x)$ primarily influences numerical errors. For instance, in solving the RANS equations, while significant errors may arise from turbulence modeling, these errors remain largely unaffected by $\Delta(x)$, provided the grid is sufficiently fine. In such contexts, numerical analysis establishes a relationship between $\Delta(x)$ and the local error sources, offering systematic methods to estimate these errors, such as through truncation error or residual assessments [19].

However, the scenario changes in nonlinear multiscale problems have been coarse-grained situations where models represent small-scale dynamics. Here, $\Delta(x)$ serves as the coarse-graining length scale, e.g., the filter width, directly influencing errors stemming from imperfect modeling of unresolved scales [7]. The numerical grid spacing can be set equal to or smaller than $\Delta(x)$; in the former case, $\Delta(x)$ also governs numerical errors. Nevertheless, estimating modeling errors in these situations often relies on heuristics and physical insights, rendering traditional truncation error estimation less effective.

To address the aforementioned, Adaptive Mesh Refinement (AMR) techniques have been developed. AMR dynamically adjusts the mesh based on solution features, enhancing resolution in regions with complex flow structures while coarsening it elsewhere to conserve computational resources [22]. This approach reduces reliance on user experience and minimizes trial-and-error in mesh generation. The general proposed algorithm for AMR is presented in Algorithm 1. The main key steps in AMR are [22]:

- **Error Estimation**: Identifies regions where mesh refinement or coarsening is necessary based on numerical accuracy and solution features.
- **Mesh Adaptation Strategy**: Defines the approach for modifying the mesh, determining when and how refinement or coarsening should be applied. The candidate techniques can be broadly classified into three categories [29]. r-adaptation, which adjusts node positions without changing mesh connectivity; p-adaptation, which increases the polynomial order of the solution approximation locally; and h-adaptation, which refines or coarsens the mesh by adding or removing elements.

The scope of this research is limited to the error estimation techniques, while the subsequent mesh adaptation procedure is not explicitly addressed.

1.3. Error Estimation

While the numerical truncation error can be rigorously derived and analyzed, the error arising from the modeling of unresolved scales cannot be quantified with the same level of certainty. Since this error stems from unresolved physical processes, its estimation inherently relies on physics-based assumptions. Consequently, absolute confidence in the estimated modeling error within LES cannot be assured, regardless of how reasonable the estimate may appear [27].

Error estimation techniques can be classified into two distinct categories: *a priori* and *a posteriori*. The former estimates the error bounds before performing the numerical simulation, relying on theoretical assumptions about the regularity of the underlying (unknown) exact solution and on the characteristics of the numerical scheme and computational grid. These estimates depend on user experience and analytical properties of the governing equations, but do not provide localized information about the error distribution within the computational domain. In contrast, *a posteriori* approaches evaluate the error based on the obtained numerical solution. Such estimations yield localized error indicators,

Algorithm 1: Adaptive Mesh Refinement (AMR) Algorithm

Input: Initial mesh $\mathcal{M}_0$, error tolerance ϵ
Output: Adapted mesh $\mathcal{M}_{\text{final}}$ satisfying ϵ
`/* Initialization */`
Set initial mesh $\mathcal{M} \leftarrow \mathcal{M}_0$;
`/* Adaptive Refinement Loop */`
repeat
  `/* Primal Solution Computation */`
  Solve the governing equations on the current mesh $\mathcal{M}$ to obtain solution $\mathbf{u}$;
  `/* Error Estimation */`
  Compute the numerical error $\eta(\mathbf{u}, \mathcal{M})$;
  `/* Check Convergence */`
  if $\eta(\mathbf{u}, \mathcal{M}) < \epsilon$ **then**
    Return: Adapted mesh $\mathcal{M}_{\text{final}}$;
  else
    `/* Error Localization */`
    Identify regions where $\eta(\mathbf{u}, \mathcal{M}) > \epsilon$;
    `/* Mesh Adaptation */`
    Refine or coarsen mesh elements in identified regions;
  end
until *Error tolerance condition* $\eta(\mathbf{u}, \mathcal{M}) < \epsilon$ *is met*;

which are invaluable for AMR strategies, as they directly relate to the computed solution rather than theoretical bounds.

Feature-based Error Estimation: It is often called physics-based approaches due to their specific application-related physical knowledge or reasoning [27]. Feature-based error estimation methods rely on a predefined flow feature, assuming that regions with complex flow features are more likely to contain a higher concentration of numerical errors. These features could be solution gradients, solution curvature, or identified solution features [30]. Although the feature-based methods benefit from lower computational overhead compared to other methods, they often lead to excessive refinement of certain features while leaving others insufficiently refined, resulting in worse solutions compared to adjoint-based ones [22]. This is also proven by Dwight with a test case of inviscid transonic flow over an airfoil using an unstructured finite-volume discretization [31].

Various feature-based refinement strategies have been applied in mesh adaptation across different applications. A widely used method involves refining regions near flames or detonation waves in turbulent flows, typically implemented within the AMR framework [32]. Another prevalent approach prioritizes regions with high small-scale kinetic energy, which is particularly beneficial in turbulence-resolving simulations [33, 34, 35]. In steady turbulence simulations, refinement is often directed toward high-vorticity regions, ensuring improved resolution in areas of intense rotational flow dynamics [36]. Moreover, in various other applications, expert-defined criteria are utilized to determine refinement zones, adapting the mesh based on specific flow features and simulation objectives [27].

Criterion-based Error Estimation: Criterion-based error estimation defines a standard that, while not necessarily having direct physical relevance like the feature-based approach, is designed to capture complex phenomena within the numerical simulation. In this context, two mesh adaptation criteria are introduced by Benard et al. [37]: (1) ensuring an accurate resolution of the mean flow field and (2) resolving a sufficient portion of turbulent scale motions. Although the methodology demonstrated significant improvements in accuracy and efficiency compared to traditional uniform refinement, it has several limitations. It does not modify boundary cells, restricting near-wall refinements, and its effectiveness may depend on the initial mesh. While computationally cheaper than other candidates, it still incurs additional costs due to the iterative process.

Residual-based Error Estimation: This approach, commonly referred to as truncation-error-based error estimation, is grounded in the definition of truncation error across various discretization techniques. Truncation error, which quantifies the difference between an exact Partial Differential Equation (PDE) and its discrete numerical approximation, represents the contribution of local element discretization parameters (e.g., cell size, skewness) to the overall discretization error. As a result, it serves as a valuable indicator for guiding mesh adaptation. The fundamental principle behind truncation error-based adaptation is to distribute truncation error uniformly across the computational domain, thereby minimizing total discretization error [30].

While simple discretization schemes allow for direct computation of truncation error, more complex numerical methods require estimation techniques [30]. In the Finite Element Method (FEM), for example, error estimators operate by substituting the finite element solution, constructed from basis functions and coefficients, into the original PDE, yielding residuals [38]. Residual-based indicators highlight regions where the discrete solution deviates beyond an acceptable tolerance from the continuous governing PDE. The residual, defined as a function that equals zero when the PDE is exactly satisfied, forms the foundation for error estimation. However, not only are residual-based methods computationally more expensive than other candidates, such as feature-based and criterion-based approaches, as they require the explicit computation of the PDE residual [39], but they also do not effectively account for propagation effects, which are intrinsic to convection-dominated problems [40].

Adjoint-based Error Estimation Adjoint-based error estimation, also known in the literature as *output-* or *goal-oriented error estimation*, evaluates the contribution of local discretization errors in each element with respect to a specific Quantity of Interest (QoI) [30]. This method enables targeted mesh adaptation, ensuring refinement efforts are focused on regions that most influence the desired output. A key aspect of this approach is the computation of the adjoint solution, which quantifies the sensitivity of the QoI to local perturbations in the primal solution. In this framework, these sensitivities are used to weight the local residuals, thereby identifying regions where discretization errors have the largest impact on the QoI. Although adjoint-based error estimation is widely recognized as the most accurate among existing error assessment techniques, it also incurs the highest computational cost [41]. This overhead primarily arises from the additional requirement of solving the adjoint system alongside the primal equations.

The adjoint problem, often referred to as the *dual problem*, is solved in conjunction with the *primal problem*, which represents the original flow field [42]. The adjoint solution quantifies how sensitive a defined QoI is to perturbations in the residuals of the governing equations. Mathematically, the primal governing equations can be expressed in residual form as

$$R(U) = 0, \tag{1.1}$$

where R is the residual operator and U is the primal solution. Based on this, the adjoint-based error estimation defines a local *error indicator field* $\epsilon(x)$ as

$$\epsilon(x) = -\psi(x)R(U(x)), \tag{1.2}$$

where $\psi(x)$ is the adjoint variable representing the sensitivity of the QoI to residual perturbations at spatial location x. This local quantity reflects how much the numerical solution violates the governing equations, weighted by how important that violation is to the QoI. Hence, known as the *Adjoint-Weighted Residual* (DWR) method [43]. The global error in the QoI, denoted by δJ, can then be approximated by integrating the local contributions across the domain:

$$\delta J \approx \int_{\Omega} \epsilon(x)\,\mathrm{d}x. \tag{1.3}$$

In unsteady nonlinear problems, such as the Navier–Stokes equations, computing the adjoint solution requires access to the complete time history of the primal solution, since the adjoint equation at each time step depends on the corresponding primal state. Consequently, the entire primal trajectory must first be computed and stored. The same time history is also required to evaluate the injected residuals used in error estimation. Hence, both the residual and adjoint solution fields are necessary to compute the error indicators and overall error estimate. In addition, LES generates such high-dimensional

datasets that the storage required for adjoint-based error estimation becomes prohibitively large and infeasible for practical use [43].

1.4. Thesis Scope & Structure

Adjoint-based error estimation is widely regarded as the most accurate method for AMR, particularly in LES applications. However, its use in unsteady simulations poses significant challenges due to the high storage requirements associated with accessing primal flow fields during backward-in-time adjoint computations and residual evaluations. This research specifically aims to address this bottleneck by exploring compression strategies based on Artificial Intelligence (AI) for reducing storage demands while maintaining error estimation accuracy. To maintain a controlled yet physically relevant framework, the study is limited to the one-dimensional (1D) unsteady Burgers' problem forced by DNS data from a Turbulent Channel Flow (TCF), serving as a simplified model for testing the proposed methodology.

Following the conceptual introduction in this chapter, Chapter 2 presents the research proposal. It reviews the state of the art, outlines previous efforts to mitigate storage costs, identifies the remaining research gap, and articulates the main research objective and questions. Chapter 3 provides a theoretical foundation for adjoint-based error estimation, including its formulation, capabilities, and inherent limitations. To address storage constraints, Chapter 4 introduces several surrogate modeling techniques, from classical modal decomposition to advanced deep learning architectures, and discusses their applicability to unsteady simulations. The methodology is outlined in Chapter 5, which describes the computational setup, the benchmark method used for comparison, the proposed surrogate models, and the performance evaluation framework. Chapter 6 verifies the framework using a Method of Manufactured Solutions (MMS) forcing term applied to the unsteady viscous 1D Burgers' equation. Chapter 7 further demonstrates the approach using a more realistic scenario, in which the 1D viscous Burgers' equation is solved using a forcing term extracted from a DNS dataset of turbulent channel flow, particularly targeting the wall-normal component of the momentum equation. Finally, Chapter 8 summarizes the findings of the research, and Chapter 9 outlines potential directions for future investigation.

2

Research Proposal

As discussed briefly in Chapter 1, adjoint-based error estimation remains the accurate reliable approach among the available candidates for use within AMR frameworks for LES applications. Nevertheless, several challenges limit its practical implementation. To compute the error indicator field, two primary quantities are required: the injected residuals and the adjoint solution. In unsteady nonlinear problems, such as the NS equations, the computation of the adjoint solution necessitates access to the time history of the primal solution, as the adjoint PDE at each time step depends on the corresponding primal state. Therefore, access to both fields is essential for evaluating local error indicators and global error estimates.

For time-dependent problems, the computational cost and storage requirements increase substantially because the adjoint solution must be propagated backward in time while accessing the corresponding primal states required for both the adjoint computation and residual evaluation. One straightforward strategy is to store the complete time series of primal solutions to provide the necessary flow information at each adjoint time step. However, storing all primal states in memory is prohibitively expensive for large-scale turbulence simulations [43]. To address this challenge, some researchers opt to store the entire time history on disk [44], though this introduces significant I/O overhead due to limited data access speed compared to direct memory reads. Others mitigate this by storing snapshot solutions at discrete time intervals and reconstructing intermediate states via temporal interpolation [26].

An alternative, more memory-efficient strategy is the *checkpointing technique* [45, 46], in which primal solutions are stored only at selected time steps (so-called *checkpoints*) and recomputed locally as needed during the backward adjoint integration. Although checkpointing substantially reduces memory requirements, it increases computational cost due to the repeated forward recomputations between checkpoints. Consequently, the practical application of unsteady adjoint-based error estimation requires a delicate balance between storage usage, computational cost, and the accuracy of the primal and adjoint solutions.

In addition, LES generate extremely high-dimensional datasets due to the fine spatial and temporal resolutions required to resolve the large, energy-containing eddies of turbulent flows. The number of degrees of freedom scales approximately with $(L/\Delta x)^3(T/\Delta t)$, where L and T represent the characteristic length and time scales of the flow, and Δx and Δt denote the grid spacing and time step, respectively. For example, a moderate-resolution LES with 512^3 spatial points and 10^4 time steps already exceeds 10^{11} stored variables per field, making direct storage of all primal states infeasible. These scaling considerations amplify the storage and computational demands when adjoint-based error estimation is applied to unsteady turbulent flows [43]. Such challenges motivate the development of alternative strategies that reduce storage requirements while maintaining the fidelity of the adjoint-based error estimation framework.

2.1. Research Gap Identification

Several approaches have been proposed in the literature to address these challenges, each tackling either the primal and residual storage problem, the cost of adjoint computations, or both. Hunter and Hulshoff [41] propose a Super-Resolution Neural Network (SRNN) that reconstructs fine-grid adjoint fields from coarser adjoints, thereby reducing the need for refined adjoint solves. On the residual side, their method allows subsampling of the stored primal history (2× and 4×), demonstrating storage reductions approaching one-eighth of the original size in 3D cases, within the limits of acceptable accuracy. Challenges remain in the form of over-prediction near adjoint extrema, the enforcement of boundary conditions, and the scalability of the approach to higher-dimensional flows while effective in maintaining error estimate accuracy and local error indicators, demonstrating the potential of neural networks for adjoint and primal reconstruction.

Following this potential demonstration, work by Roth et al. [47] sought to remove the dependency on coarse-grid inputs entirely by training adjoint solvers directly from governing equations. They shift the focus entirely to the adjoint side by replacing the classical FEM-based adjoint solve with neural networks trained via residual minimization in the strong form. This avoids the computational overhead of refined adjoint solves and demonstrates effectivity indices close to unity across linear and nonlinear stationary problems. However, since no explicit residual compression is included, storage requirements remain unchanged. Their main difficulties lie in unstable training behaviour, sensitivity to collocation point selection, and limited applicability to multiphysics systems, motivating future research in variational formulations and energy-based methods. Moreover, because the network is trained for a specific objective functional, its generalization to other QoIs remains limited, requiring retraining or reformulation for each new output.

In response to these limitations, other researchers have sought to approximate adjoint information indirectly rather than replacing the adjoint solver itself. Fidkowski [48] instead circumvents the unsteady adjoint altogether by introducing a Field-Inversion Machine Learning (FIML) framework. Only time-averaged residuals are stored from the primal run, while a corrected steady RANS system provides a surrogate adjoint to weight these residuals. This significantly reduces both storage and computational cost and has been demonstrated successfully for airfoil flows at a range of Reynolds numbers. However, the effectiveness of this averaging relies on its representativeness of the underlying unsteady dynamics, implying a hidden dependence on the chosen QoI. Moreover, since the method assumes spatial discretization errors dominate, its corrected estimator can under-predict errors on coarse meshes, raising questions of robustness for broader applications. This approach shows that surrogate steady adjoints can approximate unsteady behavior cost-effectively, but it also highlights the limitations of temporal averaging when dealing with fully unsteady turbulent flows.

Building upon this motivation to lower storage costs, subsequent research focus directly on data compression and reconstruction as an alternative path. Directly targeting the residual storage bottleneck, Sitaram [49] investigates compression and reconstruction of the primal fields using Proper Orthogonal Decomposition (POD) and autoencoders to be used in the residuals evaluation. Both methods achieve significant storage savings, with autoencoders providing competitive reconstruction fidelity. The main challenge, however, lies in ensuring reconstructed fields retain the accuracy needed for reliable residuals evaluation.

To test these ideas under more realistic unsteady conditions, Romana [39] expanded this comparison to include both spatial and temporal learning models. Romana extends this line of work by comparing POD with Convolutional Autoencoder (CAE), Echo State Network (ESN), and hybrid CAE–ESN surrogates. While CAEs provided robust performance in turbulent channel flows, ESNs struggled in nonlinear cases despite good results for smooth problems. These findings suggest that while spatially oriented architectures such as CAEs handle structured turbulent data effectively, recurrent models like ESNs are more suitable for capturing temporal dependencies, motivating hybrid strategies for unsteady problems.

Li [43] approaches the same storage challenge by proposing reduced-order representations via POD and online incremental Singular Value Decomposition (SVD), constructed on the fly to avoid storing the full primal trajectory. When coupled with adaptive refinement, this reduces memory requirements by approximately three orders of magnitude while still producing reliable error indicators. Never-

theless, the reduced-order framework risks neglecting small-scale features essential for adjoint stability, and checkpointing remains necessary in some cases, adding extra computational cost. This work demonstrates the scalability of reduced-order frameworks for large unsteady simulations, though it also underlines the importance of preserving small-scale dynamics that influence adjoint consistency.

Finally, beyond data-driven and reduced-order strategies, attention has also been given to improving the numerical consistency of adjoint formulations themselves. Shi et al. [50] emphasize the importance of dual consistency within the Correction Procedure via Reconstruction (CPR) framework. Dual consistency ensures that the discrete adjoint operator faithfully represents the continuous adjoint equation in the limit of mesh refinement, allowing the discrete adjoint to yield accurate functional sensitivities. By enforcing this property, they obtained smooth adjoint fields and super-convergent functionals, enabling highly efficient mesh adaptation. However, achieving dual consistency in practice is challenging, as it depends strongly on discretization aspects such as solution points, correction functions, and boundary flux definitions. In summary, prior works have addressed the residual and adjoint challenges through multiple strategies: surrogate compression models (POD, CAE, ESN) to reduce storage, AI-guided solvers and super-resolution networks to approximate fine adjoints at lower cost, surrogate adjoint frameworks (FIML) to bypass unstable unsteady adjoints to improve adjoint accuracy and reduce computational and storage burden. Each of these approaches demonstrates promising results but also faces specific challenges, ranging from reconstruction fidelity and training stability to dual consistency enforcement and adjoint instability in turbulence, that currently limit their scalability to LES.

A comparative overview of these efforts is provided in Table 2.1, highlighting how different studies have approached the residual storage and adjoint computation problems within adjoint-based error estimation frameworks. Considering recent developments, various attempts have been made to reduce the computational cost of computing the adjoint solution and to mitigate its inherent instability when solved backward in time [47, 48, 50]. These efforts include neural network–based surrogate solvers [47, 41] and discretization-level innovations such as dual-consistent formulations [50]. In parallel, significant progress has been achieved in compressing the primal solution and reconstructing it as needed for adjoint computations and residual evaluaiton [49, 39, 43]. While these approaches have proven effective in reducing storage requirements for the primal trajectories, challenges remain in the evaluation of residuals when using reconstructed primal fields, as reported for example by Romana [39]. Moreover, comparatively little attention has been given to directly addressing the storage of the residuals themselves, which are equally required for adjoint-based error estimation.

Table 2.1: Overview of prior works addressing residual storage and adjoint computation challenges in adjoint-based error estimation.

Research	Residuals	Adjoint Solution
Romana (2025) [39]	Compressed primal using CAE, ESN, CAE-ESN, and POD baseline, reconstructed when needed.	Solved adjoint using reconstructed primal.
Hunter & Hulshoff (2024) [41]	Subsampled primal and reconstructed fine residual fields using SRNN.	Reconstructed fine adjoints from coarse adjoints; avoided refined solves.
Sitaram (2023) [49]	Compressed primal with POD and CAE, reconstructed when needed.	Solved adjoint using reconstructed primal.
Roth et al. (2022) [47]	–	NN trained by residual minimization to approximate adjoint solutions.
Fidkowski (2022) [48]	Stored only time-averaged residuals, not full trajectories.	Replaced unstable unsteady adjoint with steady adjoint using FIML.
Li (2022) [43]	POD and iSVD built online; avoided full storage.	Computed adjoints on reduced-order states.
Shi et al. (2015) [50]	Dual-consistent CPR for accurate residuals.	Enforced dual consistency to stabilize adjoint fields.

Despite these advances, no existing study provides a unified or scalable framework capable of efficiently compressing and reconstructing both the primal and injected residual data while preserving adjoint accuracy and the fidelity of error indicators in unsteady flows. The literature primarily targets

either the cost reduction of adjoint computation [47, 48] or the compression of primal trajectories [49, 43, 39], but the residual compression problem remains largely unexplored. This constitutes the central research gap that the present work aims to address: developing and systematically assessing surrogate-based compression strategies for both primal and injected residual fields within adjoint-based error estimation, and quantifying their impact on reconstruction accuracy, computational efficiency, and reliability of the resulting error indicators for unsteady cases.

2.2. Research Formulation

This research builds on the established idea of compressing the primal solution to support adjoint computations [39], while extending the concept further by also compressing and reconstructing the evaluated residuals from the original primal solution. In this way, a compression framework is proposed that simultaneously reduces the storage requirements of both the primal trajectories and the injected residuals. Such an approach directly addresses the two dominant storage bottlenecks in adjoint-based error estimation and aims to improve its scalability for application to large-scale unsteady turbulent flows.

Data compression strategies can be broadly divided into two classes: *lossless* and *lossy*. Lossless techniques guarantee exact recovery of the data but provide limited compression, whereas lossy approaches discard part of the information in exchange for substantially higher compression rates [51]. Given the prohibitive storage demands of adjoint-based methods in unsteady simulations, lossy compression is preferred in this context. Moreover, exact recovery of the full flow field is unnecessary, since adjoint-based error estimation requires high accuracy only in regions where the solution strongly influences the QoI, making localized information loss acceptable if global error indicators remain reliable.

Within lossy approaches, Reduced-Order Models (ROMs) have been widely employed to represent high-dimensional dynamical systems at a fraction of the original cost [52]. However, these methods are generally based on linear modal decompositions, which struggle to capture the strongly nonlinear features of turbulent flows. In practice, this limitation requires a large number of modes to achieve acceptable accuracy, thereby undermining the intended storage reduction [53].

On the other hand, deep learning methods offer nonlinear decomposition capabilities through the use of nonlinear activation functions [52]. Recent advances in this area have shown that deep neural networks can efficiently capture the essential dynamics of turbulent flows with far fewer modes, achieving significant compression without losing the physical integrity of the data [54, 55]. Their successful application in CFD highlights the potential of deep learning–based surrogates as powerful alternatives to classical ROMs for data reduction.

By exploiting these advances, this research seeks to develop a compression methodology that integrates state-of-the-art deep learning techniques with adjoint-based error estimation. The ultimate goal is to alleviate the storage requirements associated with both primal and injected residual fields in unsteady aerodynamic problems, while preserving the accuracy and reliability of the resulting error estimates. Hence, the research objective is defined as follows:

The research objective is to investigate the potential benefits of directly compressing and reconstructing the injected residuals for adjoint-based error estimation for unsteady problems.

Before addressing the main research question, it is essential to establish a foundational understanding of the existing research landscape. This involves reviewing prior developments in adjoint-based error estimation and data compression to identify their applications, limitations, and current bottlenecks, particularly in the context of unsteady and turbulent CFD problems. Furthermore, existing data compression and reconstruction techniques, ranging from classical modal approaches to AI-driven surrogates, must be examined to assess their respective capabilities and limitations in representing complex flow data. Based on these insights, a suitable benchmark methodology can then be defined to serve as a reference framework for evaluating and comparing the performance of the proposed compression strategies within adjoint-based error estimation.

Building upon this foundation, the research is guided by the main research question, followed by sub-questions that define the specific directions of investigation and evaluation.

Main Research Question

What is the impact of AI-based compression and reconstruction of the injected residuals on the accuracy of unsteady adjoint-based error estimates?

RQ1. What are the fundamental differences between the primal solution and the injected residual in terms of their data characteristics in the context of compression and reconstruction?

RQ2. How do spatial and temporal deep learning architectures compare in their ability to learn turbulent flow dependencies relevant to the accurate reconstruction of injected residuals?

RQ3. How should compression and reconstruction performance be evaluated against the benchmark approach in terms of accuracy, compression efficiency, and computational cost?

RQ4. Between compressed injected residuals and adjoint fields computed from compressed primals, which contributes more significantly to the overall error in output error estimation?

RQ5. Can the developed framework for injected residual compression and reconstruction be effectively extended to the compression of primal fields, and under what conditions does this generalization remain valid?

3

Adjoint-based Error Estimation: Definition & Limitations

This chapter introduces the theoretical foundations of adjoint-based error estimation, which forms the cornerstone of goal-oriented adaptive strategies in CFD. The adjoint framework enables the quantification of numerical errors in relation to a user-defined output QoI. Adjoint solutions have been used in various contexts, ranging from design optimization to output error estimation [42]. The chapter begins by defining the adjoint problem and distinguishing between its discrete and continuous formulations, outlining their derivation and practical implications. Subsequently, the principal advantages and limitations of the adjoint methodology are discussed, emphasizing challenges such as computational cost, storage requirements, nonlinearity, and instability in unsteady or chaotic systems.

3.1. Adjoint Problem Definition

Adjoint-based formulations have been implemented through two principal paradigms: the discrete and the continuous approaches. While both have proven effective in applications such as error estimation and design optimization [42], their practical implications differ considerably. The discrete formulation, derived directly from the discretized governing equations, tends to deliver higher accuracy on fine meshes [56]. However, its implementation demands explicit access to the Jacobian of the numerical scheme, often requiring complex analytical derivations or automatic differentiation frameworks [42]. Additionally, maintaining adjoint consistency within this discrete setting can be challenging [57]. In contrast, the continuous formulation defines the adjoint problem at the PDE level before discretization, offering greater conceptual clarity and reduced implementation complexity when both primal and adjoint solutions are sufficiently resolved [56]. For these reasons, the continuous adjoint approach is employed in this study as the most practical and robust choice for the problem considered. This section briefly presents the theoretical foundation of the continuous form of the adjoint formulation. For detailed explanations, see [42]. Moreover, the discrete form of the adjoint is presented in Appendix A.1

Considering that the discrete adjoint is an approximation of the more fundamental concept of the continuous adjoint, this section presents the definition of the continuous adjoint. Considering the general form of a differential equation with $\mathcal{L} \equiv \alpha\frac{\partial}{\partial x}$, where α is a positive scalar representing the advection speed, and the source term is $f = f(x)$. Moreover, a homogeneous Dirichlet boundary condition is imposed on the left side of the domain, defined as $\Omega \equiv [a, b]$. This differential equation is formulated as

$$\mathcal{L}u = f: \qquad \alpha\frac{\partial u}{\partial x} = f(x), \qquad x \in \Omega, \tag{3.1}$$

$$\text{Primal BC}: \qquad u = 0, \qquad x = a. \tag{3.2}$$

Rather than solving for the entire solution u, the focus is on computing a single QoI as the output. This

QoI can be expressed as either a scalar or a function:

$$J = \int_\Omega g(x)u(x)\,dx \equiv (g,u), \tag{3.3}$$

where for any desired QoI J, an appropriate $g(x)$ should be selected.[1] The continuous adjoint ψ satisfies the following differential equation:

$$\mathcal{L}^*\psi = g: \qquad -\alpha\frac{\partial\psi}{\partial x} = g(x), \qquad x\in\Omega, \tag{3.4}$$

$$\text{Adjoint BC}: \qquad \psi = 0, \qquad x = b. \tag{3.5}$$

Comparing Eq. (3.4) with Eq. (3.1), it is evident that the adjoint equation introduces a negative sign and imposes the boundary condition on the right side of the domain. This reversal illustrates the inverse flow of information in the dual problem compared to the primal one. To derive the dual form in the continuous setting, starting from Eq. (3.3):

$$\underline{\text{Advection Case}} \qquad\qquad \underline{\text{General Form}} \tag{3.6}$$

$$J = \int_\Omega g(x)u\,dx \qquad = (g,u) \tag{3.7}$$

$$\xrightarrow{(3.4)} J = \int_\Omega -\alpha\frac{\partial\psi}{\partial x}u\,dx \qquad = (\mathcal{L}^*,u) \tag{3.8}$$

$$\xrightarrow{\text{Integration by parts}} J = \int_\Omega \psi\alpha\frac{\partial u}{\partial x}\,dx - \alpha\psi u\Big|_a^b \tag{3.9}$$

$$\xrightarrow[\text{BCs}]{(3.2)\ \&\ (3.5)} J = \int_\Omega \psi\alpha\frac{\partial u}{\partial x}\,dx \qquad = (\psi,\mathcal{L}u) \tag{3.10}$$

$$\xrightarrow{(3.1)} J = \int_\Omega \psi f(x)\,dx \qquad = (\psi,f). \tag{3.11}$$

The adjoint ψ and the source term f are sufficient to calculate J. Moreover, the adjoint represents the sensitivity of the output J to perturbations in the continuous source terms f.

Having derived the *Dual Form* for the continuous case, Eq. (3.11), of the advection equation Eq. (3.1), it is time to formulate a general form. The general form is presented as Eq. (3.12), which means having a differential operator $\mathcal{L}$ acting on an unknown solution u to form a differential term $\mathcal{L}u$ integrated against a so-called adjoint function ψ, there is another differential operator $\mathcal{L}^*$ which is acting on the aforementioned function ψ and is integrated against function u.

$$\int_\Omega (\mathcal{L}u)\,\psi\,dx = \int_\Omega u\,(\mathcal{L}^*\psi)\,dx \quad \forall \text{ suitable } u,\psi \tag{3.12}$$

To generalize a bit further using the L^2 inner product notation, this can be written as:

$$(\mathcal{L}u,v) = (u,\mathcal{L}^*v) \quad \forall u,v\in\bar{\Omega}, \tag{3.13}$$

where $\bar{\Omega}$ is a suitable function space, and v corresponds to ψ in Eq. (3.12). In the literature, this relation is referred to as the *Adjoint Identity*.

So far, homogeneous boundary conditions are assumed and an output J defined as the interior integral (g,u). However, this is not always the case. For example, some outputs, like the lift coefficient, are calculated on the boundaries. Thus, by introducing the residual in continuous form as:

$$R(u) \equiv \mathcal{L}u - f, \tag{3.14}$$

[1] For instance, if the output is the value of $u(x)$ at an arbitrary point x_i, the delta Dirac function can be used: $g(x) = \delta(x - x_i)$. In this case,

$$J = \int_\Omega \delta(x-x_i)u(x)\,dx = u(x_i).$$

and noting that the adjoint represents the sensitivity of a QoI J to perturbations in the residuals $R(u)$, the following relation holds:

$$J'(\delta u) = \int_\Omega \psi\, R'(\delta u)\, d\Omega \quad \forall \text{ (permissible) } \delta u, \tag{3.15}$$

which defines ψ as the *Continuous Adjoint Equation*, regardless of boundary conditions, output type, or problem nonlinearity [42]. Here, δu must be permissible[2]. This equation states how a change in the residuals $R'(\delta u)$ impacts the output $J'(\delta u)$, leveraging the concept of *First Variation*[3] which is denoted as the prime in the aforementioned equations based on the variational calculus.

3.2. Adjoint-based Error Estimation

One of the main applications of adjoints is *Adjoint-based error estimation* [42]. This section presents the concept in its continuous form, while its discrete formulation is discussed later in Appendix A.2.

Using the general form of the primal differential equation,

$$\mathcal{L}u = f, \qquad x \in \Omega, \qquad \text{Primal BC}, \qquad x \in \delta\Omega, \tag{3.16}$$

the numerical solution on a coarse mesh H yields an approximate solution u_H to the exact solution u, which is assumed to be sufficiently smooth to allow the evaluation of the residual $R(u_H)$.

Following defining a desired QoI as

$$J(u_H) = (u_H, g), \tag{3.17}$$

the error δJ in this output is defined by comparing the approximate and exact solutions:

$$\delta J = J(u) - J(u_H) \xrightarrow{(3.3)} \delta J = (u, g) - J(u_H, g) \tag{3.18}$$

$$\xrightarrow{\text{Linearity}} \delta J = (u - u_H, g) \xrightarrow{(3.4)} \delta J = (u - u_H, \mathcal{L}^*\psi) \tag{3.19}$$

$$\xrightarrow{(3.13)} \delta J = (L(u - u_H), \psi) \xrightarrow{\text{Linearity}} \delta J = (Lu, \psi) - (Lu_H, \psi) \tag{3.20}$$

$$\xrightarrow{(3.1)} \delta J = (f, \psi) - (Lu_H, \psi) \xrightarrow{\text{Linearity}} \delta J = (f - Lu_H, \psi) = -(Lu_H - f, \psi) \tag{3.21}$$

$$\xrightarrow{(3.14)} \delta J = -(R(u_H), \psi) \xrightarrow{\text{Integral Form}} \delta J = -\int_\Omega \psi R(u_H)\, d\Omega. \tag{3.22}$$

Eq. (3.22) expresses the total output error as the *Adjoint-Weighted Residual*, which quantifies how local residuals influence the QoI through the adjoint weighting. In practice, the exact adjoint ψ is not available, and a numerical approximation ψ_h must be used:

$$\delta J_{\text{est}} = -\int_\Omega \psi_h R(u_H)\, d\Omega. \tag{3.23}$$

However, since the exact solution u is generally unknown, a solution computed on a finer spatial mesh h is used. The approximate output error can thus be estimated as

$$\delta J_{\text{est}} = J_h(u_h) - J_H(u_H). \tag{3.24}$$

Let $u_h^H = I_h^H u_H$ be the projection of the coarse solution into the fine space, where I_h^H is an interpolation operator from the coarse to the fine mesh. A Taylor expansion of the fine-space functional and residual about the injected solution u_h^H gives, after neglecting higher-order terms,

$$\delta J_{\text{est}} \approx -\psi_h R_h(u_h^H), \tag{3.25}$$

[2]Permissible refers to satisfying any boundary conditions required by the primal problem.

[3]Taking the first variation of J involves examining how a change in the state, δu (defined across the entire domain Ω impacts the value of J. Instead of representing a rate of change, $J'(\delta u)$ reflects a direct change in J, which can be concisely denoted as δJ. The variation of the residual, $R'(\delta u)$, follows a similar principle.

where ψ_h is the fine-space adjoint and $R_h(u_h^H)$ is the fine-space residual evaluated with the injected coarse solution. This expression forms the basis of practical output error estimation techniques.

For nonlinear problems, however, this linearized relation is not exact. The derivation assumes that the governing operator $\mathcal{L}$ can be locally linearized around u_H, which introduces higher-order terms of $\mathcal{O}(\delta u^2)$ in the Taylor expansion. The full expression therefore becomes

$$\delta J_{\text{est}} \approx -\psi_h^T R_h(u_h^H) + \mathcal{O}(\delta u^2). \tag{3.26}$$

Even if the adjoint is computed on a highly refined mesh, the $\mathcal{O}(\delta u^2)$ term represents a residual linearization error arising from the nonlinear coupling between the adjoint and the injected coarse primal solution. Consequently, in nonlinear regimes, such as unsteady turbulent flows, the accuracy of the linearized adjoint formulation depends strongly on the smoothness and resolution of the primal solution. Insufficiently resolved primals can amplify these higher-order effects, leading to reduced accuracy or instability in the estimated output error.

3.3. Unsteady Adjoints

From a mathematical standpoint, extending adjoint methods to unsteady problems is relatively straightforward. Time can be considered analogous to an additional spatial dimension, allowing the adjoint to be formulated similarly to the steady case. However, key differences arise in the formulation of the adjoint itself due to the inherent directionality of information flow in time, which contrasts with spatial dimensions [42].

Analogous to the steady-state case, if an element-based spatial discretization method is employed, the error estimate can be localized at the element level. Moreover, in the case of a time-averaged QoI, the resulting error estimate can also be expressed in a time-averaged form:

$$\delta \bar{J}_{\text{est}} \approx -\frac{1}{T} \sum_{t=1}^{N_t} \sum_{e=1}^{N_e^h} \psi_{h,e,t} R_{h,e,t}(U_h^H), \tag{3.27}$$

where N_t defines the total number of time steps. Accordingly, a local time-averaged error indicator can be derived for each element as:

$$\bar{\epsilon}_e = \frac{1}{T} \sum_{t=1}^{N_t} \left| \psi_{h,e,t} R_{h,e,t}(U_h^H) \right|. \tag{3.28}$$

Mesh adaptation strategies can be developed by analyzing the spatial distribution of the computed error indicators. The central aim of these strategies is to reduce the estimated error while maintaining computational efficiency. In the case of uniform h-refinement, the local indicators obtained on the refined mesh can be aggregated to evaluate a representative indicator for each element of the original coarse grid. This aggregation process allows the construction of a global adaptation criterion that preserves accuracy at a lower computational cost [42]. Moreover, it should be noted that, since the injected residuals in this work are defined on the fine space, they are denoted as $R(U_h^H)$ rather than $R_h(U_h^H)$.

3.4. Applications & Limitations of Adjoint Methods

Adjoint-based error estimation and mesh adaptation have become well-established techniques in computational fluid dynamics, particularly for steady-state problems. Over the past two decades, numerous studies have advanced the methodology across discretization types, flow regimes, and applications, progressively extending its scope from inviscid, steady flows to unsteady turbulent regimes and, more recently, integrating reduced-order and machine learning strategies.

Venditti and Darmofal [58, 59], building on earlier verification studies in quasi-1D settings [60, 61], developed a discrete adjoint-based framework for a posteriori error estimation and anisotropic mesh adaptation for the steady, inviscid Euler equations. Using a high-order Finite Volume Method (FVM), they targeted functional outputs and demonstrated that anisotropic refinement achieves superior accuracy and computational efficiency compared to isotropic or gradient-based strategies. Validations on subsonic, transonic, and supersonic flows, including single- and multi-element airfoils, highlighted the method's self-terminating behavior and sensitivity to flow features influencing the functional of

interest. Extending this work, Park [62] applied the same adjoint-based adaptation principles to 3D unstructured meshes, enabling simulations of 3D wing–body and high-lift configurations.

Nemec et al. [63, 64] further extended the approach to complex geometries using embedded-boundary Cartesian meshes for the steady compressible Euler equations. Their framework was validated on challenging 3D cases, including a supersonic vortex, the ONERA M6 wing, a sonic boom model, and launch-vehicle configurations, showing robustness even on coarse initial meshes and reduced computational costs through refinement localized to regions influencing the output.

Hartmann and Houston [57] applied adjoint-based adaptation to the compressible Euler equations using high-order discontinuous Galerkin FEMs. Comparing adjoint-weighted (Type I) and unweighted (Type II) error indicators, they found that Type I indicators yield more effective and accurate refinement patterns, validated on nozzle and airfoil cases.

The extension of adjoint methods to unsteady turbulent flows introduces new challenges such as chaotic sensitivity, time-averaging, and significant storage costs. Fidkowski [48] addressed these issues in RANS and Detached Eddy Simulation (DES) (with the Spalart–Allmaras (SA) closure), exploring three adaptation strategies: output-based and residual-based p-adaptation, and anisotropic mesh optimization (MOESS). Applications to high-Reynolds-number airfoils demonstrated effective capture of unsteady features relevant to functionals like lift.

Several works have targeted LES, where high-dimensional, unsteady datasets exacerbate storage demands. Li [22] introduced a Reduced-Order Representation (ROR) using POD combined with an enhanced iSVD algorithm to reconstruct the primal solution without storing the full dataset. This was built on earlier efforts by Li et al. [28], who proposed an enhanced online algorithm for real-time ROR construction, validated on a 1D unsteady Burgers problem with multi-frequency forcing.

For unsteady simulations, various authors have tackled the temporal component explicitly. Fidkowski [65] proposed a continuous-in-time adjoint framework for estimating spatial and temporal errors in scalar advection–diffusion and compressible NS problems. Mani et al. [66, 67] developed discrete adjoint formulations that separate total output error into temporal discretization and algebraic convergence components, using adjoint-weighted residuals to adapt both time-step sizes and solver tolerances. Belme, Dervieux, and Alauzet [68] proposed a time-accurate anisotropic space–time adaptation method using a discrete adjoint to construct an optimal metric field, validated on blast-wave and acoustic-wave problems. Ding et al. [69] introduced cost-reduction strategies, including sub-iterations and coarser adjoint spaces, while Doetsch and Fidkowski [70] combined coarse-mesh adjoints with entropy-based indicators to reduce the expense of unsteady adaptation.

Adjoint frameworks have also been applied beyond traditional aerodynamic contexts. Jia and Kikumoto [71] integrated unsteady LES-based adjoints with Bayesian inference for Source Term Estimation (STE) in urban environments, employing wavelet-based compression to address data storage challenges. Lin et al. [72] extended STE to time-varying sources, highlighting the importance of temporal resolution in reducing uncertainty. Granzow et al. [73] applied adjoint-based anisotropic mesh adaptation to incompressible nonlinear elasticity, while Becker et al. [74] provided the theoretical foundation for the dual weighted residual method, interpreting the adjoint as an optimal control problem.

Recent trends incorporate machine learning into adjoint-based frameworks. Chen and Fidkowski [75] replaced the costly adjoint solve with an encoder decoder Convolutional Neural Network (CNN) trained on low-fidelity solutions to predict both global output errors and localized error indicators. Roth et al. [47] developed a neural network solver for the adjoint PDE in strong form, trained via residual minimization, achieving competitive accuracy with reduced computational cost. Chakraborty et al. [76] integrated the dual weighted residual method into Physics-Informed Neural Networks (PINNs) for multi-goal error estimation on complex geometries, combining mesh-free learning with goal-oriented adaptivity.

An overview of representative works employing adjoint-based error estimation across various flow regimes and numerical frameworks are presented in Table 3.1. Collectively, these works chart the evolution of adjoint-based error estimation from steady, inviscid problems to fully unsteady, turbulent, and multiphysics settings. The progression shows clear lines of influence, such as Park's extension of Venditti's work to 3D, or Li's POD-based ROR building on earlier LES adjoint efforts, and highlights

the growing role of reduced-order modeling and machine learning in addressing computational bottlenecks. This trajectory underscores the method's adaptability, from traditional mesh refinement to advanced hybrid and data-driven strategies.

As presented earlier, adjoint-based error estimation offers remarkable accuracy although its application is subject to specific limitations. A major limitation is the high computational cost, as adjoint-based methods often require solving the adjoint problem on a refined mesh or with higher-order discretizations, especially for unsteady problems [41, 39]. For time-dependent simulations, the need to store the entire trajectory of the primal solution for backward-in-time adjoint solves leads to extremely large storage requirements, which can become intractable for large-scale or high-dimensional problems [41, 43]. Moreover, when multiple QoIs are of interest, a separate adjoint problem must be solved for each, significantly increasing the computational burden and making the method less practical for engineering applications involving many QoIs [77].

Adjoint-based error estimators for nonlinear problems typically rely on linearizations, which can introduce errors. These linearization errors may cause the estimator to under-predict the true error, especially for nonlinear QoIs, and can result in suboptimal mesh adaptation or premature termination of adaptive algorithms [78].

A critical and often underappreciated limitation is the potential instability of the adjoint solver, especially in chaotic or long-time unsteady simulations. For example, in LES and other chaotic systems, the adjoint equations can become unstable over long time integrations, leading to unreliable error estimates and difficulties in grid adaptation [79, 39]. This instability can severely limit the effectiveness of adjoint-based error estimation in practical, real-world simulations where long time horizons or chaotic dynamics are present [79].

In light of these limitations, particularly the prohibitive storage demands associated with unsteady adjoint computations, recent research has increasingly focused on developing strategies to alleviate data storage and retrieval costs. Techniques ranged from ROM [43] to AI-driven surrogate modeling [39, 41] in order to address the challenge of retaining the full temporal history of the primal solution without compromising accuracy. In the context of this research, the problem of storage requirement in adjoint-based error estimation is specifically tackled, aligning with this active and rapidly evolving area of study within the current literature.

Table 3.1: Overview of representative studies employing adjoint-based error estimation across various flow regimes and numerical frameworks.

Research	Test Case	Flow	Viscosity	Problem	Numerical Method	Turbulence Modeling	QoI
Lin et al. [72]	3D pollutant dispersion	Incompressible	Viscous	Unsteady	FVM	RANS Realizable k–ϵ	Location and intensity of pollutant
Chakraborty et al. [76]	Generic 2D PDEs	N/A	N/A	Steady	FEM	–	Scalar functionals
Roth et al. [47]	2D Poisson problem	Incompressible	N/A	Steady	FEM	–	Domain-integrated scalar value
Fidowski [48]	2D High-Re airfoil	Compressible	Viscous	Unsteady	FEM	DES, RANS + SA	Functional outputs (e.g., lift)
Jia et al. [71]	3D pollutant dispersion	Incompressible	Viscous	Unsteady	FVM	LES	Location and strength of pollutant
Li et al. [28]	1D Burgers equation	Incompressible	Viscous	Unsteady	FVM	LES	Aerodynamic forces
Chen et al. [75]	2D advection-diffusion	Incompressible	Viscous	Steady	FEM	–	Boundary flux of scalar field
Doetsch et al. [70]	2D airfoil	Compressible	Viscous	Unsteady	FEM	–	Scalar outputs
Granzow et al. [73]	2D Cook's membrane; 3D biological cell	Incompressible	N/A	Steady	FEM	N/A	Displacements, stresses, and integrated displacement over subdomains
Fidkowski [65]	2D advection–diffusion	Both	Viscous	Unsteady	FEM	–	Scalar field integrals
Ding et al. [69]	2D Euler; scalar advection	Both	Inviscid	Steady	FEM	–	Scalar functionals
Belme et al. [68]	2D Euler equations	Compressible	Inviscid	Unsteady	FVM	–	Functional outputs (e.g., pressure signatures, wave amplitudes)
Mani et al. [66]	2D Euler equations	Compressible	Inviscid	Unsteady	FVM	–	Integrated & instantaneous functionals
Park [80]	2D bump channel & airfoils; 3D wing	Compressible	Inviscid	Steady	FEM	–	Lift, Drag
Nemec et al. [64]	2D Diamond airfoil; 3D wing	Compressible	Inviscid	Steady	FVM	–	Lift, Drag, Moment, Near-field pressure integrals
Mani et al. [67]	2D Euler equations	Compressible	Inviscid	Unsteady	FVM	–	Scalar outputs
Nemec et al. [63]	Supersonic vortex model, 3D wing	Compressible	Inviscid	Steady	FVM	–	Lift, Drag, Axial/Normal force, Pitching moment
Venditti et al. [59]	2D flow over airfoils	Compressible	Inviscid	Steady	FVM	–	Lift, Drag
Venditti et al. [58]	2D flow over airfoild, Gaussian bump	Compressible	Inviscid	Steady	FVM	–	Lift, Drag, Moment
Hartman et al. [57]	2D Transonic nozzle, airfoils	Compressible	Inviscid	Steady	FEM	–	Lift, Drag, Pointwise pressure
Becker et al. [74]	Generic 1D and 2D PDE	–	–	Steady	FEM	–	Functional outputs

4

Surrogate Modeling Techniques: Description & Applications

This chapter introduces the surrogate modeling techniques employed in this work for compressing and reconstructing CFD data within the proposed framework. The objective is to identify and assess representative methods capable of reducing the dimensionality of unsteady flow fields and injected residuals while preserving their essential characteristics relevant to adjoint-based error estimation.

The chapter begins by outlining the theoretical background of modal decomposition techniques and their use in representing flow dynamics through a reduced number of spatial modes. Their mathematical formulation and applications in data compression and reconstruction are briefly discussed, followed by the selection of a benchmark method to serve as a reference throughout this study.

Subsequently, the fundamental principles of Machine Learning (ML) are introduced, focusing on the underlying concepts of Artificial Neural Networks (ANNs) and their variants. The discussion emphasizes how these methods have been applied within CFD, particularly for data-driven modeling, flow-field reconstruction, and adjoint-based error estimation. Finally, the selected deep learning frameworks used in this research are presented in detail, including their configuration, working principles, and key hyperparameters that govern their performance.

4.1. Modal Decomposition Techniques

Modal decomposition techniques provide effective low-dimensional representations of complex flow fields and are widely used for extracting coherent structures, data compression, and reduced-order modeling in fluid dynamics [81]. Among these, Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD) are the most extensively applied, with several recent extensions and hybrid approaches enhancing their capabilities.

POD, first introduced to fluid dynamics by Lumley [82], constructs modes from the eigenvectors of the autocovariance matrix of snapshot data. These modes are ordered by their associated energy, providing an orthogonal basis that is optimal in capturing variance in the data [83, 84]. DMD, on the other hand, focuses on extracting modes characterized by distinct frequencies and growth/decay rates [85], thus offering a spectral perspective complementary to POD's energy-based ranking. While POD emphasizes energy optimality, DMD highlights dynamical interpretability, although the resulting modes are not guaranteed to be orthogonal [83].

Several computational strategies exist for performing POD. When the number of snapshots N_t exceeds the degrees of freedom N_x, the correlation matrix UU^T is used. Conversely, the *Method of Snapshots* [86] is preferred for $N_x \gg N_t$, requiring the decomposition of U^TU. Alternatively, Singular Value Decomposition (SVD) can be directly applied to the snapshot matrix [87, 88], improving numerical robustness. More recently, online and incremental variants have been developed to handle high-dimensional or streaming datasets efficiently, such as randomized SVD approaches for canonical turbulent flows [89]

and incremental SVD implementations applied to 3D cylinder flows [43].

In this study, POD was chosen as the benchmark approach instead of DMD, as it maximizes the captured kinetic energy and is therefore more suitable for efficient data compression [83]. Online approaches such as incremental SVD [43] were not considered, since their reliance on numerous parameters was beyond the scope of the present work. Among the available offline techniques, the SVD-based formulation was adopted because of its improved robustness against round-off errors [88].

4.2. Deep Learning Techniques

Recent developments in deep learning have opened new possibilities for accelerating CFD through surrogate models that approximate or replace expensive numerical solvers [90]. These models aim to retain physical fidelity while enabling faster predictions, uncertainty quantification, and design exploration. Broadly, current approaches can be categorized into physics-constrained, data-driven, and hybrid formulations [91].

Deep learning, a subset of ML, uses multi-layer ANNs to approximate nonlinear mappings through activation functions between inputs and outputs. Each ANN is composed of an input layer, multiple hidden layers, and an output layer, with neurons connected by trainable weights and biases. Through forward and backward propagation, the network iteratively updates the trainable parameters to minimize a loss function that measures the deviation between predictions and reference data [91, 90].

Depending on their topology, ANNs can take various forms, including feed-forward, convolutional, or recurrent architectures. Feed-forward networks map input data directly to output quantities and are suitable for regression or classification tasks [92]. In contrast CNNs extract spatially correlated features using convolutional filters, making them particularly effective for multidimensional data such as velocity or pressure fields [91]. Recurrent Neural Networks (RNNs) introduce an internal time-evolving state that captures temporal dependencies, allowing them to predict the evolution of dynamical systems [90]. These architectural families serve as foundations for more specialized structures such as the Convolutional AutoEncoder (CAE) and the Echo State Network (ESN), both of which have found increasing application in fluid mechanics for spatial and temporal modeling, respectively.

In fluid dynamics, deep-learning-based surrogates have emerged as powerful alternatives to conventional numerical or reduced-order models. Physics-constrained neural networks represent one of the earliest developments in this direction. Sun et al. [92] demonstrated that a fully connected network trained solely by minimizing the residuals of the incompressible NS equations can reproduce steady laminar flows without any labeled data, achieving large computational speedups while maintaining physical accuracy. Similarly, Zhu et al. [93] combined deterministic and probabilistic formulations using a convolutional encoder–decoder and a conditional flow-based generative model, capable of uncertainty propagation in stochastic Darcy flows. Both studies showed that embedding physical constraints into the loss function improves generalization and reduces data requirements.

A parallel research direction involves learning adjoint-based error indicators for mesh adaptation. Chen and Fidkowski [75, 94] trained encoder–decoder CNNs on discontinuous Galerkin simulations to approximate local and global output errors. Their networks closely matched adjoint-based references and successfully guided adaptive refinement for both advection–diffusion and transonic aerodynamic flows, offering adjoint-free error estimation at negligible cost.

Another growing field focuses on data-driven compression and reduced-order modeling. Fukami et al. [95] introduced a hierarchical convolutional autoencoder capable of learning nonlinear mode decompositions analogous to POD, yet with enhanced capability to capture strongly nonlinear structures. Olmo et al. [54] extended this approach by incorporating physical regularizations, such as enforcing incompressibility and preserving enstrophy, into the training loss, improving reconstruction fidelity and interpretability of turbulent flow fields. Such results confirm that CAEs provide a compact, data-efficient representation of spatially complex dynamics while preserving essential flow physics.

In parallel, temporal modeling has increasingly relied on recurrent or reservoir-based networks to predict the evolution of reduced-order representations. Racca et al. [55] demonstrated that a purely data-driven ESN can accurately forecast the temporal evolution of latent flow dynamics extracted from DNS data, maintaining statistical consistency over long horizons with only a fraction of the computational

cost. Similarly, Jordanou et al. [96] combined POD with ESNs and applied dimensionality reduction to the reservoir itself, achieving up to 80% computational savings without loss of dynamic accuracy. These works highlight the ability of reservoir computing to capture nonlinear temporal dependencies efficiently, even for chaotic flow regimes.

Overall, the literature indicates two major directions for deep-learning-based surrogates in fluid mechanics: convolutional architectures for spatial compression and reconstruction, and recurrent or reservoir-based architectures for temporal evolution. CAEs have proven particularly effective for nonlinear modal decomposition and flow-field reconstruction, offering a natural comparison against linear methods such as POD. Meanwhile, ESNs provide a lightweight yet powerful framework for modeling temporal dynamics in high-dimensional systems, bridging the gap between reduced-order modeling and sequential learning. Given their complementary nature, CAE for spatial encoding and ESN for temporal prediction, both architectures are well-suited as independent candidates for evaluating deep-learning-based surrogates against the established POD framework. The following sections briefly introduce the concepts of CAE and ESN.

4.2.1. Convolutional Autoencoder

Convolutional Autoencoders (CAEs) are a specialized class of autoencoders that leverage CNN layers to perform unsupervised feature learning, dimensionality reduction, and data reconstruction, particularly for spatially structured data [97, 98].

In CNNs, information is propagated through successive layers using convolution operations. A kernel filter of fixed size is shifted across the input with a specified stride, producing the feature maps of the next layer [53]. This operation can be written as

$$\hat{y} = f\big(y \otimes W + b\big), \tag{4.1}$$

where $y \in \mathbb{R}^{N_1}$ is the input vector, $\hat{y} \in \mathbb{R}^{N_2}$ is the output vector, W represents the learnable kernel weights, and $b \in \mathbb{R}^{N_2}$ is the bias term. The symbol $\otimes$ denotes the convolution operator. The nonlinear activation function f is applied element-wise to introduce nonlinearity and enhance the representational capacity of the network.

As illustrated in Figure 4.1, a CAE consists of two main components: an encoder and a decoder, both built using convolutional layers. The encoder compresses the input data into a lower-dimensional latent representation by applying a series of convolutional operations, which extract features while reducing dimensionality. The decoder then reconstructs the original input from this compressed representation, typically using transposed convolutions or upsampling layers to restore the spatial structure [99, 100].

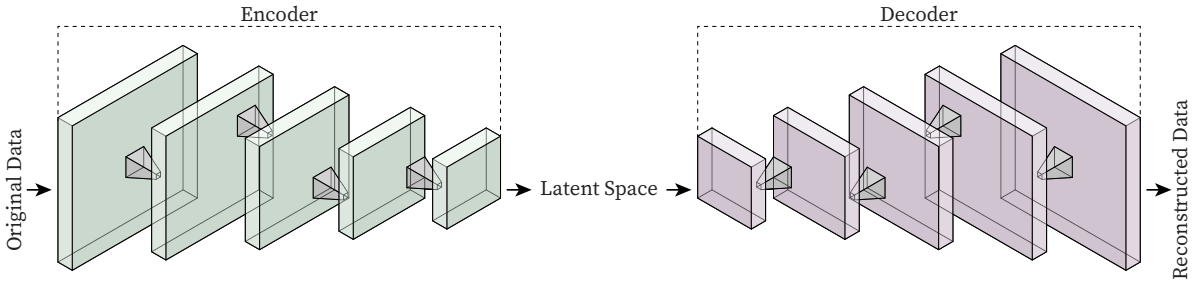


Figure 4.1: Schematic illustration of a Convolutional Autoencoder (CAE).

CAEs are trained in an unsupervised manner, optimizing the reconstruction loss to ensure the latent space captures essential features [99]. The performance and efficiency of a CAE depend on several key hyperparameters:

- **Number of Convolutional Layers:** Controls model depth; more layers can capture complex features but may risk overfitting or vanishing gradients [101].
- **kernel Size** Affects the receptive field; larger kernels capture broader context but may blur fine details [101].

- **Latent Space Dimensionality:** Balances compression and information retention; too small may lose details, too large may overfit [102].
- **Stride:** Influence spatial downsampling; higher stride reduces size but may lose information [102].
- **Activation Functions:** Impact nonlinearity and learning dynamics [99].

On top of the aforementioned hyperparametes, learning rate, batch size, and number of epochs are standard training hyperparameters affecting convergence and stability.

4.2.2. Echo State Network

Error Back Propagation (BP) [103] has been widely adopted as the standard technique for training, particularly Feed-Forward Neural Networks (FFNNs) [104]. While BP techniques have been adapted for RNNs [105, 106], their effectiveness remains limited. A key challenge in applying BP to RNNs lies in the presence of bifurcations, which can hinder convergence; Even in cases where convergence occurs, the process is typically slow, computationally demanding, and susceptible to suboptimal local solutions [107]. Recent advancements in BP for RNNs, specifically second-order gradient descent methods known as Hessian-free optimization [108] address some of the aforementioned limitations. Notably, these methods demonstrate improved performance in tasks that demand long-term memory [108], which are typically challenging for BP-based RNN training [109].

As an alternative, Reservoir Computing (RC) has emerged as a promising approach to tackle the persistent challenges in RNN training [110]. A notable application within this framework is the Echo State Network (ESN) [111]. Since their introduction, ESNs have proven to be a practical method for training RNNs, characterized by their conceptual simplicity and low computational cost [112]. However, the simplicity of ESNs can be somewhat misleading. Effective deployment requires a degree of experience, as several potential errors can arise. A critical aspect is the initial construction of the raw reservoir network, which depends on a set of hyper-parameters. These parameters must be carefully selected, as improper configuration can adversely affect performance [104].

In general, ESNs are beneficial for supervised learning in temporal machine learning tasks, where a given training input signal $\mathbf{u}(n) \in \mathbb{R}^{N_u}$ is provided along with a corresponding target output signal $\mathbf{y}^{\text{target}}(n) \in \mathbb{R}^{N_y}$. Here, $n = \{1, \ldots, T\}$ represents discrete time, and T denotes the total number of data points in the training dataset. Indeed, the objective is to train a model that produces an output $\mathbf{y}(n) \in \mathbb{R}^{N_y}$, which ideally matches the target signal $\mathbf{y}^{\text{target}}(n) \in \mathbb{R}^{N_y}$. This is achieved by minimizing a chosen error function $E(\mathbf{y}^{\text{target}}, \mathbf{y})$. Additionally, for tasks requiring generalization, the model should perform well on unseen data.

ESNs utilize a type of RNN that incorporates leaky-integrated discrete-time continuous-value units. Leaky integration is a mechanism in which the current state of a neuron is updated as a weighted combination of its previous state and a newly computed activation. The weighting is controlled by a leaking rate parameter α. The typical update equations are defined as:

$$\tilde{\mathbf{r}}(n) = \tanh\left(\mathbf{W}^{\text{in}}[1; \mathbf{u}(n)] + \mathbf{W}\mathbf{r}(n-1)\right), \tag{4.2}$$

$$\mathbf{r}(n) = (1-\alpha)\mathbf{r}(n-1) + \alpha\tilde{\mathbf{r}}(n), \tag{4.3}$$

where $\mathbf{r}(n) \in \mathbb{R}^{N_r}$ represents the reservoir neuron activations, and $\tilde{\mathbf{r}}(n) \in \mathbb{R}^{N_r}$ denotes their update at time step n. The function $\tanh(\cdot)$ is applied element-wise, while $[\cdot;\cdot]$ represents vertical concatenation of vectors or matrices. Although $\tanh$ is the most commonly used activation function, other sigmoid-based activation functions can also be employed. The weight matrices $\mathbf{W}^{\text{in}} \in \mathbb{R}^{N_r \times (1+N_u)}$ and $\mathbf{W} \in \mathbb{R}^{N_r \times N_r}$ correspond to the input and recurrent connections, respectively. The parameter $\alpha \in (0, 1]$ defines the leaking rate, which determines how much past information is retained. This helps regulate temporal dynamics, improve memory capacity, and stabilize training. When $\alpha = 1$, the model operates without leaky integration, reducing to $\tilde{\mathbf{r}}(n) \equiv \mathbf{r}(n)$ [104].

The reservoir is generated by a tuple of $(\mathbf{W}^{\text{in}}, \mathbf{W}, \alpha)$, of which the weights are randomly generated using a couple of hyper-parameters.

- **Reservoir Size N_r:** A larger reservoir state space $\mathbf{r}(n)$ facilitates the discovery of a suitable linear combination to approximate the target output $\mathbf{y}^{\text{target}}(n)$. However, a trade-off must be considered

between computational cost and model performance, as increasing the reservoir size generally enhances predictive accuracy but also raises computational demands. When generalization is required, appropriate regularization techniques should be applied to mitigate the risk of overfitting.

- **Reservoir Sparsity:** Although it is recommended to maintain sparse reservoir connections in $\mathbf{W}^{\text{in}}$, studies have shown that this has a minimal impact on model performance. Thus, prioritizing its optimization is not essential [104].
- **Distribution of Nonzero Elements:** A uniform or Gaussian distribution is recommended [104].
- **Spectral Radius ($\rho(\mathbf{W})$):** It is defined as the largest absolute eigenvalue of the reservoir weight matrix $\mathbf{W}$, is a key factor in ensuring the Echo State Property (ESP)[1]. Typically, $\rho(\mathbf{W}) < 1$ is recommended to maintain stability, preventing multiple fixed points, periodic behavior, or chaotic attractors. However, in practice, optimal $\rho(\mathbf{W})$ can sometimes exceed 1, as strong input signals and activation nonlinearity help regulate feedback dynamics, making strict adherence to $\rho(\mathbf{W}) < 1$ not always necessary [104].
- **Input Weight ($\mathbf{W}^{\text{in}}$) Scaling:** It is recommended to scale $\mathbf{W}^{\text{in}}$ uniformly to have fewer hyperparameters [104].
- **Leaking Rate (α):** It controls the speed of reservoir state updates, acting as a time constant in the discretized system. It influences the network's ability to capture short-term memory and can be tuned to match the dynamics of input $\mathbf{u}(n)$ and target output $\mathbf{y}^{\text{target}}(n)$. While a higher α allows faster updates, a lower α slows down state changes, improving memory retention.

The linear readout layer is defined as:

$$\mathbf{y}(n) = \mathbf{W}^{\text{out}}[1; \mathbf{u}(n); \mathbf{r}(n)], \tag{4.4}$$

where $\mathbf{W}^{\text{out}} \in \mathbb{R}^{N_y \times (1+N_u+N_r)}$ ia the output weight matrix and $\mathbf{y}(n) \in \mathbb{R}^{N_y}$ is the network output. Since ESNs are typically feed-forward and linear, (4.4) can be rewritten to represent all data points ($n = \{1, \ldots, T\}$) as:

$$\mathbf{Y} = \mathbf{W}^{\text{out}}[1; \mathbf{U}; \mathbf{R}], \tag{4.5}$$

$$\xrightarrow{\text{Notational Brevity}} \mathbf{Y} = \mathbf{W}^{\text{out}}\mathbf{X}, \tag{4.6}$$

where $\mathbf{Y} \in \mathbb{R}^{N_y \times T}$ are all outputs $\mathbf{y}(n)$, and $\mathbf{X} \in \mathbb{R}^{(1+N_u+N_r) \times T}$ are all $[1; \mathbf{u}(n); \mathbf{r}(n)]$ where $n = \{1, \ldots, T\}$.

Training the network or in other words finding the optimized output weights $\mathbf{W}^{\text{out}}$ that can minimize the squared error between the target and network output (i.e. $E(\mathbf{Y}^{\text{target}}, \mathbf{Y})$) requires solving typically an overdetermined system of linear equations [104]. One way to solve this system is to use ridge regression (regression with Tikhonov regularization [113]) as

$$(\mathbf{X}\mathbf{X}^T + \beta\mathbf{I})\mathbf{W}^{\text{out}} = \mathbf{Y}^{\text{target}}\mathbf{X}^T, \tag{4.7}$$

where $\mathbf{I}$ is an identity matrix and β is the Tikhonov regularization parameter which is used to prevent overfitting and feedback instabilities. The general algorithm of training and inference in ESN is presented in Algorithm 2.

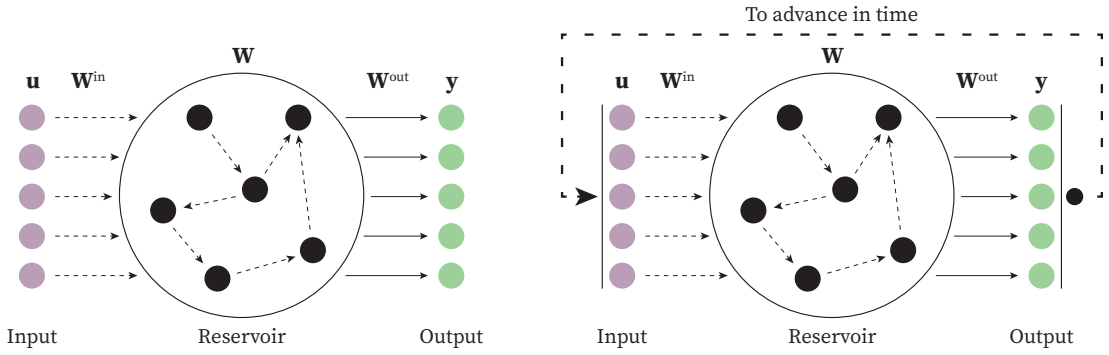
The ESN can operate in two distinct configurations: *open-loop* and *closed-loop*, as illustrated in Figure 4.2. In the *open-loop* configuration, the reservoir receives input directly from the external signal $\mathbf{u}$, and the network output $\mathbf{y}$ does not influence subsequent time steps. This mode is primarily used during the training phase, allowing the reservoir states to evolve under teacher-forced inputs while maintaining the stability of the internal dynamics.

In contrast, in the *closed-loop* configuration, the network operates autonomously by feeding back its own output $\mathbf{y}$ to the input layer, thereby advancing the system in time without external inputs. This setup is typically employed during validation or inference, enabling the ESN to reconstruct or predict temporal sequences based solely on its internal state.

[1]The reservoir state is uniquely determined by the fading history of inputs [111]

Algorithm 2: Training and Inference in an Echo State Network (ESN)

Input: Training input $\mathbf{u}(n)$, target output $\mathbf{y}^{\text{target}}(n)$
Output: Trained output weights $\mathbf{W}^{\text{out}}$ for inference
`/* Reservoir Initialization */`
1 Generate a large random reservoir RNN with weight matrices $\mathbf{W}^{\text{in}}$, $\mathbf{W}$ and leaking rate α;
`/* Training Phase */`
2 Run the reservoir using the training input $\mathbf{u}(n)$ and collect the corresponding activation states $\mathbf{x}(n)$;
3 Compute the linear readout weights $\mathbf{W}^{\text{out}}$ by solving a linear regression problem, minimizing the defined cost function governed by $\mathbf{y}(n)$ and $\mathbf{y}^{\text{target}}(n)$;
`/* Inference Phase */`
4 For new input data $\mathbf{u}(n)$, compute the output $\mathbf{y}(n)$ using the trained output weights $\mathbf{W}^{\text{out}}$;



(a) Open-loop (b) Closed-loop

Figure 4.2: Schematic representation of the two operational configurations of the Echo State Network (ESN). (a) In the *open-loop* configuration, the reservoir receives external inputs $\mathbf{u}$ through the fixed, pseudo-randomly generated matrices $\mathbf{W}^{in}$ and $\mathbf{W}$ (dashed arrows), while the trainable matrix $\mathbf{W}^{out}$ (solid arrows) maps the reservoir states to the output $\mathbf{y}$. (b) In the *closed-loop* configuration, the predicted output $\mathbf{y}$ is fed back to the input layer to advance the network in time, enabling autonomous temporal evolution without external driving signals.

In both configurations, the dashed arrows in Figure 4.2 represent the pseudo-randomly initialized and fixed matrices ($\mathbf{W}^{in}$ and $\mathbf{W}$), which define the input-to-reservoir and reservoir-to-reservoir connections, respectively. The solid arrows indicate the trainable matrix $\mathbf{W}^{out}$, which maps the reservoir states to the network output.

5

Methodology & Framework

This chapter presents the methodology adopted in this research. It begins with an overview of the baseline framework and the proposed approach developed to compress and reconstruct the injected residuals, aiming to alleviate the storage requirements associated with unsteady adjoint-based error estimation. Subsequently, the computational setup of both the primal and adjoint solvers is described in detail. The chapter then introduces the benchmark surrogate modeling technique, POD, along with the proposed deep learning–based methods, namely CAE and ESN. For each method, the rationale behind the architectural design and hyperparameter optimization strategy is discussed. Finally, the *Performance Assessment Framework* is introduced, in which different evaluation scenarios are designed to compare the efficiency and accuracy of the proposed surrogate approaches as well as identify the main sources of error in the error estimate.

5.1. Baseline & Proposed Frameworks

Figure 5.1 illustrates the baseline and proposed frameworks that form the methodological foundation of this research. The baseline framework (top) represents the conventional adjoint-based error estimation workflow. In this process, the primal solver provides the solution on the coarse space U_H, which is projected to the fine space and injected into the governing PDE to compute the injected residual $R(U_h^H)$. The adjoint solver then computes the adjoint field Ψ_h using the fine solution U_h. These quantities are subsequently coupled in the output error estimation block to obtain the error in the QoI estimation δJ_{FVM} and the error indicator field ϵ_{FVM} to be used in AMR.

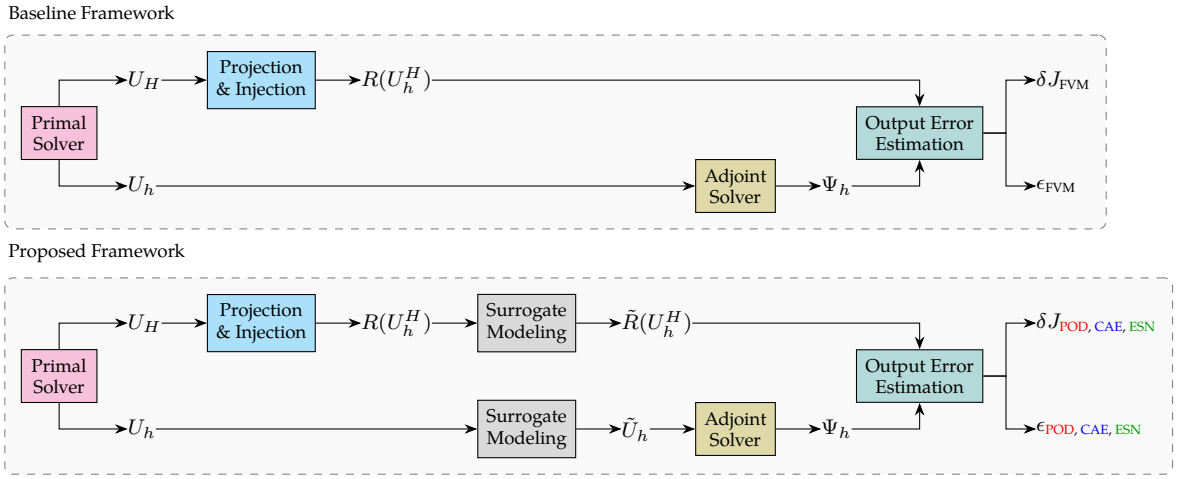


Figure 5.1: Comparison between the baseline and proposed frameworks.

The proposed framework (bottom) extends this process by introducing surrogate modeling techniques

to compress and reconstruct both the injected residuals and the fine primal solutions. The surrogate models, POD, CAE, and EESN, replace the direct storage of high-dimensional fields with compact representations to be reconstructed, denoted as $\tilde{R}(U_h^H)$ and $\tilde{U}_h$. These reconstructed fields are then used in the adjoint and error estimation stages to recover error in QoI estimation $\delta J_{\text{POD, CAE, ESN}}$ and the corresponding error indicator field $\epsilon_{\text{POD, CAE, ESN}}$.

5.2. Computational Setup

The Finite Volume Method (FVM) was employed for the spatial discretization of both primal and adjoint problems. All computations were performed with the open-source package `OpenFOAM` (v2412). Temporal and spatial discretization schemes, as well as solver settings, were carefully selected to balance stability, accuracy, and computational cost.

The setup of discretization schemes and solvers relied on established best practices in OpenFOAM, where user experience plays a role in ensuring numerical stability and convergence. Nevertheless, as all configurations were uniformly applied across the investigated cases, their influence is limited to absolute performance rather than the comparative outcomes discussed in this work.

5.2.1. Primal Solver

For the primal simulations, a second-order Crank–Nicolson scheme with an off-centering factor of 0.5 was adopted for the temporal discretization, expressed as

$$\frac{u^{n+1} - u^n}{\Delta t} + \frac{1}{2}\,\mathcal{R}(u^n) + \frac{1}{2}\,\mathcal{R}\big(u^{n+1}\big) = 0, \tag{5.1}$$

where $u^n \in \mathbb{R}^N$ and $u^{n+1} \in \mathbb{R}^N$ denote the discrete primal solutions at time levels t_n and t_{n+1}, respectively. Here, Δt represents the time step size, while $\mathcal{R} \in \mathbb{R}^N$ corresponds to the discrete residual arising from the spatial discretization of the governing equation. This choice provides a compromise between numerical stability and accuracy in unsteady problems, reducing numerical diffusion compared to fully implicit schemes while achieving better numerical stability and avoiding nonphysical temporal oscillations associated with fully centered schemes.

Spatial derivatives were discretized using cell-limited linear gradient schemes, which mitigate nonphysical overshoots and undershoots in regions of steep gradients while retaining second-order accuracy in smooth regions. Convective fluxes were approximated with a linearUpwind scheme based on local gradients, offering a balance between robustness and low numerical dissipation. Diffusive terms were treated with a corrected Laplacian scheme. The linear systems arising from the discretized equations were solved with the stabilized bi-conjugate gradient method (`PBiCGStab`) combined with a diagonal incomplete LU (`DILU`) preconditioner. This iterative solver is efficient for large, sparse systems that are mildly nonsymmetric, which is the case for the Burgers' equation discretization [114].

5.2.2. Adjoint Solver

The adjoint problem is formulated as a time-dependent system that must be integrated backward in time, consistent with the discrete temporal discretization of the primal. Accordingly, the adjoint variable was advanced using a first-order backward Euler scheme to ensure robustness and consistency with the discrete primal formulation during the reverse-time integration.

Gradient approximations were again performed with cell-limited linear schemes. Convective adjoint fluxes were discretized with a bounded upwind scheme to guarantee monotonicity and suppress nonphysical oscillations. Diffusion terms for the adjoint were handled with an uncorrected Laplacian scheme. The adjoint equations were solved with the same `PBiCGStab` solver and `DILU` preconditioner.

5.3. Surrogate Modeling Framework

This section presents the methodologies adopted for the surrogate modeling techniques used in this work. It begins with the benchmark method of POD, outlining its formulation and implementation procedure. Subsequently, the methodology of the CAE is introduced, detailing the architectural choices and design decisions made to define its structure and features. Finally, the methodology of the ESN is

described, including its optimization process, validation strategy, and training procedure. It is important to note that the Compression Ratio (CR) is defined independently for each method according to its respective representation framework.

5.3.1. Proper Orthogonal Decomposition

An offline POD was employed as the benchmark approach to construct a ROM of the injected residuals and the primal solution. Online variants, such as iSVD [43], were not considered, as their complexity as a simple benchmark approach also their dependence on numerous additional parameters lies beyond the scope of this work. As discussed in Section 4.1, the main strength of POD-based compression is that the resulting modes are naturally ordered according to their associated kinetic energy. Among available offline POD methods, the SVD was adopted due to its superior robustness compared to alternative formulations [88].

Considering the selected method as the benchmark approach to compress the datasets, for a statistically stationary flow, the input data were organized into a snapshot matrix $U \in \mathbb{R}^{N_x \times N_t}$, constructed from the solution vectors $u \in \mathbb{R}^{N_x}$ after subtracting their temporal mean $\overline{u}$. An SVD was then applied,

$$U = L\,\Sigma\,R, \tag{5.2}$$

where L and R contain the left and right singular vectors, respectively, and Σ is the diagonal matrix of singular values. In this formulation, the columns of L represent the spatial POD modes, while the temporal evolution is given by the product ΣR.

To construct a reduced-order model, the decomposition was truncated by retaining only the k most energetic modes. The truncation was determined based on the cumulative energy content of the singular values in Σ, with a defined threshold for the injected residual and fine primal solution. The reconstructed datasets were thus expressed as

$$\tilde{u}(x,t) = \overline{u}(x) + L_k\,\Sigma_k\,R_k, \tag{5.3}$$

where L_k, Σ_k, and R_k denote the truncated SVD matrices corresponding to the chosen rank k.

Finally, the Compression Ratio (CR) was defined in terms of the ratio between the original data size and the storage required for the truncated representation,

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Size}(\overline{u}) + \text{Size}(L_k) + \text{Size}(\Sigma_k) + \text{Size}(R_k)}. \tag{5.4}$$

In this expression, the dominant contributions to storage arise from L_k and R_k, whereas Σ_k remains a compact diagonal matrix.

5.3.2. Convolutional Autoencoder

The choice of activation function has a significant impact on the training stability and efficiency of deep neural networks [115]. Traditional activation functions such as hyperbolic tangent *tanh* and the logistic sigmoid are prone to the vanishing gradient problem, where saturation outside the interval $[-5, 5]$ leads to very small weight updates, slowing down learning in deep architectures [109]. Although the standard Rectified Linear Unit (ReLU) [116] alleviates this issue, it suffers from the so-called *dying ReLU* problem, which is the neurons receiving negative inputs produce zero gradients and therefore stop updating, effectively dropping out of the training process [117].

Furthermore, studies have shown that maintaining mean unit activations close to zero is beneficial for reducing bias shift and accelerating convergence [118]. Since the standard ReLU does not allow negative activations, it fails to meet this criterion.

To overcome these limitations, the Leaky Rectified Linear Unit (LReLU) [117] was adopted in this work. Unlike the standard ReLU, the LReLU introduces a small negative slope for inputs below zero, which prevents neurons from becoming permanently inactive. At the same time, it retains the computational efficiency and simplicity of ReLU while helping to keep the mean activation closer to zero. This combination makes LReLU a robust and effective activation function for the present application. This

activation function is defined as

$$f(x) = \begin{cases} x, & \text{if } x \geq 0, \\ \alpha x, & \text{if } x < 0, \end{cases} \tag{5.5}$$

where α is a small positive constant.

To construct autoencoder architectures for all test cases with varying refinement levels, an automated procedure was developed, as outlined in Algorithm 3. The algorithm begins with a series of convolutional layers that progressively downsample the input sequence while increasing the channel depth, followed by a fully connected bottleneck that maps the compressed representation to the prescribed latent dimension. The decoder mirrors this process by first applying a fully connected layer and then reconstructing the sequence through successive upsampling and convolutional blocks.

Algorithm 3: Construction of the Autoencoder1D Network

Input: Sequence length L, latent dimension $d_{\mathcal{Z}}$, input channels c_{in}, base channels c_{base}, maximum channels c_{max}, kernel size k, negative slope α, batch normalization flag

Output: Autoencoder network with convolutional encoder, latent bottleneck, and mirrored decoder

`/* Sanity Checks */`

Verify L is a power of two, k is even, and $d_{\mathcal{Z}} > 0$;

`/* Encoder: Convolutional Blocks */`

Initialize encoder channel list $[c_{\text{in}}]$ and length list $[L]$;

while $L > 1$ **do**

  Add Conv1d($c_{\text{in}} \to c_{\text{out}}, k, \text{stride} = 2, \text{padding} = k/2 - 1$);

  if *batch normalization enabled* **then**

    Add BatchNorm1d(c_{out});

  end

  Add LeakyReLU(α) activation;

  Update $L \leftarrow (L + 2 \cdot \text{pad} - k)/2 + 1$;

  Update $c_{\text{in}} \leftarrow c_{\text{out}}$, $c_{\text{out}} \leftarrow \min(2c_{\text{out}}, c_{\text{max}})$;

  Append updated c_{in}, L to tracker lists;

end

Flatten output to vector of size c_{in};

`/* Encoder: Fully-Connected Bottleneck */`

Add Linear layer mapping flattened size $\to d$;

`/* Decoder: Fully-Connected Mirror */`

Add Linear layer mapping $d \to$ flattened size;

`/* Decoder: Convolutional Blocks */`

Reverse encoder channel list;

foreach *pair* (c_{in}, c_{out}) *in reversed list* **do**

  Upsample sequence length $\times 2$ (nearest neighbor);

  Add Conv1d($c_{\text{in}} \to c_{\text{out}}, k = 3, \text{padding} = 1$);

  if *not last layer* **then**

    Add BatchNorm1d(c_{out}) and LeakyReLU(α);

  end

end

return *Reconstructed sequence*

An important aspect of the generated architectures is the treatment of boundaries. In convolutional layers, the spatial size of the feature maps can be preserved by introducing additional artificial values around the input, a process known as *padding* [115]. In this work, zero-padding was employed, whereby elements with a value of zero are added along the boundaries to maintain consistent feature-map dimensions across successive layers.

Training deep neural networks is often hindered by the problem of *internal covariate shift*, which arises when the distribution of input features to deeper layers changes as the parameters of earlier layers

are updated. This causes instability in the training process, particularly in the upper layers, and typically requires the use of very small learning rates to ensure convergence [115]. A common strategy to mitigate this issue is the use of *Batch Normalization* (BN) [118]. Given a d-dimensional input $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$, the activations are normalized across each mini-batch according to

$$\hat{x}^{(k)} = \frac{x^{(k)} - \overline{x}^{(k)}}{\sqrt{s^2[x^{(k)}] + \varepsilon}}, \tag{5.6}$$

where $\overline{x}^{(k)} = \frac{1}{m}\sum_{i=1}^{m} x_i^{(k)}$ is the mini-batch mean and $s^2[x^{(k)}] = \frac{1}{m}\sum_{i=1}^{m}\left(x_i^{(k)} - \overline{x}^{(k)}\right)^2$ is the mini-batch variance computed from m samples. The small constant $\varepsilon > 0$ is introduced to avoid division by zero. Here, $x_i^{(k)}$ denotes the activation of neuron k for the i-th training sample. By standardizing the inputs in this way, BN reduces internal covariate shift, stabilizes training, and allows the use of higher learning rates, thereby accelerating convergence [115]. Moreover, as suggested by Ioffe et al. [118], the BN layers were placed before the activation functions.

The choice of latent space dimensionality is a critical design step in constructing autoencoders for fluid dynamics applications [53]. To determine an appropriate size, Principal Component Analysis (PCA) is commonly employed as a diagnostic tool [119, 120, 121]. By performing an eigenvalue decomposition of the covariance matrix of the training data, the contribution of each principal component to the total variance can be quantified. Specifically, if λ_i denotes the i-th eigenvalue of the covariance matrix, representing the variance explained by the corresponding mode, the cumulative energy retained by the first k modes is given by

$$E(k) = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{N}\lambda_i}, \tag{5.7}$$

where N is the total number of modes. The smallest k satisfying $E(k) \geq \varepsilon_k$ (with ε_k a user-defined threshold, here is 99%) is then selected as the latent space dimension [122].

The decoder architecture employs nearest-neighbor upsampling followed by convolution layers rather than transpose convolution. This design choice is motivated by the literature [53]. Although transpose convolution layers have been widely used in the context of fluid flow surrogate modeling [55], they are prone to producing checkerboard artifacts and spatial incoherence in the reconstructed outputs, which can degrade accuracy in fluid dynamics applications [123]. By contrast, nearest-neighbor upsampling combined with convolution provides a more stable reconstruction strategy that mitigates such artifacts.

Regarding the oprimizaiton procedure, the Adaptive Moment Estimation (Adam) optimizer was selected as the optimization algorithm [41]. Adam is widely adopted due to its adaptive learning rate, which provides robustness with respect to the choice of hyperparameters [124]. Among these, the learning rate γ plays the most influential role, as it directly governs both the model's training dynamics and the overall convergence process [125].

In general, a relatively large learning rate enables rapid progress in the early stages of training but often leads to convergence towards a sub-optimal solution. Conversely, a very small learning rate allows the optimizer to approach more optimal parameters but requires significantly more training iterations. Thus, choosing an appropriate value of γ is non-trivial. To alleviate this difficulty, a linearly decaying learning rate schedule was employed [39], which gradually decreases the learning rate until reaching a prescribed minimum:

$$\gamma(\text{Epoch}) = \gamma_0\left(1 + \frac{\text{Epoch}}{N_{\text{Epochs}}}(\gamma_r - 1)\right), \tag{5.8}$$

where γ_0 denotes the initial learning rate and γ_r is the user-defined reduction factor. These parameters were tuned empirically for each test case. This schedule allows the optimizer to explore the parameter space more broadly in the early stages of training while progressively refining the solution as convergence is approached. An *epoch* is defined as a complete pass through the training dataset, with the total number of epochs also set empirically depending on the test case.

In the training procedure, a *batch*, $\mathcal{B}$, refers to a randomly selected subset of the dataset used in a single optimization step. Smaller batch sizes typically result in slower convergence but are known to improve generalization to unseen data. The batch size, N_b, also set emperically for each test case. The averaged

Mean Squared Error (MSE) over batches was adopted as the loss function to guide the optimization, expressed as

$$\text{MSE}(\hat{y}) = \frac{1}{N_{\mathcal{B}}} \sum_{j=1}^{N_{\mathcal{B}}} \frac{1}{N_b} \sum_{i=1}^{N_b} (\hat{y}_i^{(j)} - y_i^{(j)})^2, \tag{5.9}$$

where $\hat{y}_i$ and y_i denote the predicted and reference values, respectively, and $N_{\mathcal{B}}$ is the number of batches.

For the optimization process, the training dataset was normalized to zero mean and unit variance. To enhance robustness and avoid overfitting, additional training samples were generated by injecting Gaussian noise $\mathcal{N}(0, \sigma)$, ensuring statistical consistency with the underlying system. From the complete dataset, 80% was allocated for training while the remaining 20% was randomly reserved for validation.

Considering the CAE architecture, the CR was defined as follows when using the CAE to generate a surrogate model:

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Size}(\mathcal{Z}) \times N_t + \text{Decoder Parameters} + \text{Normalization Parameters}}. \tag{5.10}$$

In this formulation, only the trainable parameters of the decoder were counted, since the latent representation $\mathcal{Z}$ was stored at every time step instead of the original high-dimensional solution. Additionally, the mean and standard deviation of the primal solution were stored to account for the normalization procedure.

5.3.3. Echo State Network

This section presents the methodology of implementing the ESN to compress and reconstruct the injected residuals and fine primal solution. The methodology consists of the ESN setup configuration, the design of validation procedure, hyperparameter optimization, and definition of CR.

As explained in Section 4.2.2, within the ESN framework, the reservoir must satisfy the *Echo-State Property* (ESP), which ensures that the reservoir state is uniquely determined by the fading memory of the input sequence [111]. In other words, the influence of past inputs on the reservoir diminishes over time. This condition is guaranteed here since the activation function is chosen as the hyperbolic tangent,

$$f(y) = \tanh(y). \tag{5.11}$$

Additionally, to fulfill the ESP, the spectral radius of the reservoir weight matrix W_r set to be strictly smaller than one [104].

Since ESNs can operate in both open-loop and closed-loop configurations (see Section 4.2.2), the training was carried out in the open-loop mode, while validation was performed in the closed-loop mode. The available validation strategies are illustrated in Figure 5.2. With the exception of the single-shot validation approach, all strategies are based on successive validation folds applied to the training dataset. In these schemes, *bar 1* corresponds to the first validation fold, while *bar 2* represents the second fold for non-chaotic cases. For chaotic cases, however, the second fold (*bar 2c*) is shifted forward by one Lyapunov Time (LT).

The LT represents a characteristic timescale of chaotic dynamical systems and is defined as the inverse of the Highest Lyapunov Exponent (HLE). The HLE quantifies the exponential divergence of initially close trajectories in phase space [126]. Following the work of Nikitin [127], the values of the HLE for Turbulent Channel Flows (TCFs) at different Reynolds numbers were used to compute the corresponding LTs, which were subsequently applied in the validation of the chaotic test cases considered.

For validation, the recycle strategy was adopted, as it has been shown to outperform other approaches in the context of chaotic flow prediction [126]. In this method, the network is first trained a single time for each hyperparameter set using the full dataset in the open-loop configuration. Afterwards, the same dataset is partitioned into segments and used for validation in the closed-loop configuration, thereby recycling the training data for validation.

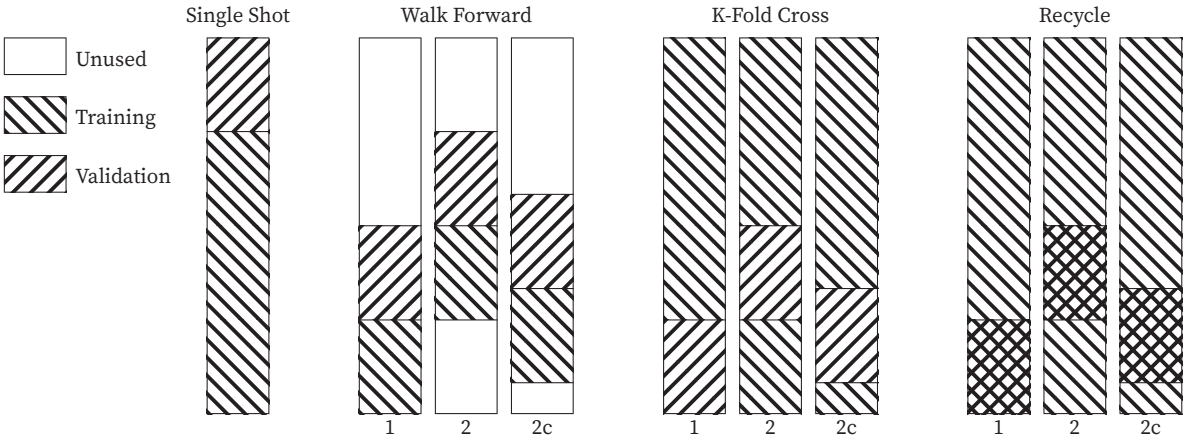


Figure 5.2: Candidate validation strategies for Echo State Network (ESN).

The loss function was defined as the MSE averaged over all validation splits. This ensures that the performance of a given hyperparameter set is not biased toward a specific segment of the dataset, but instead reflects the network's ability to generalize across different portions of the data.

Since the ESN framework requires careful hyperparameter tuning and a robust optimization strategy, a dedicated pipeline was designed, as illustrated in Figure 5.3. The procedure is divided into three main phases.

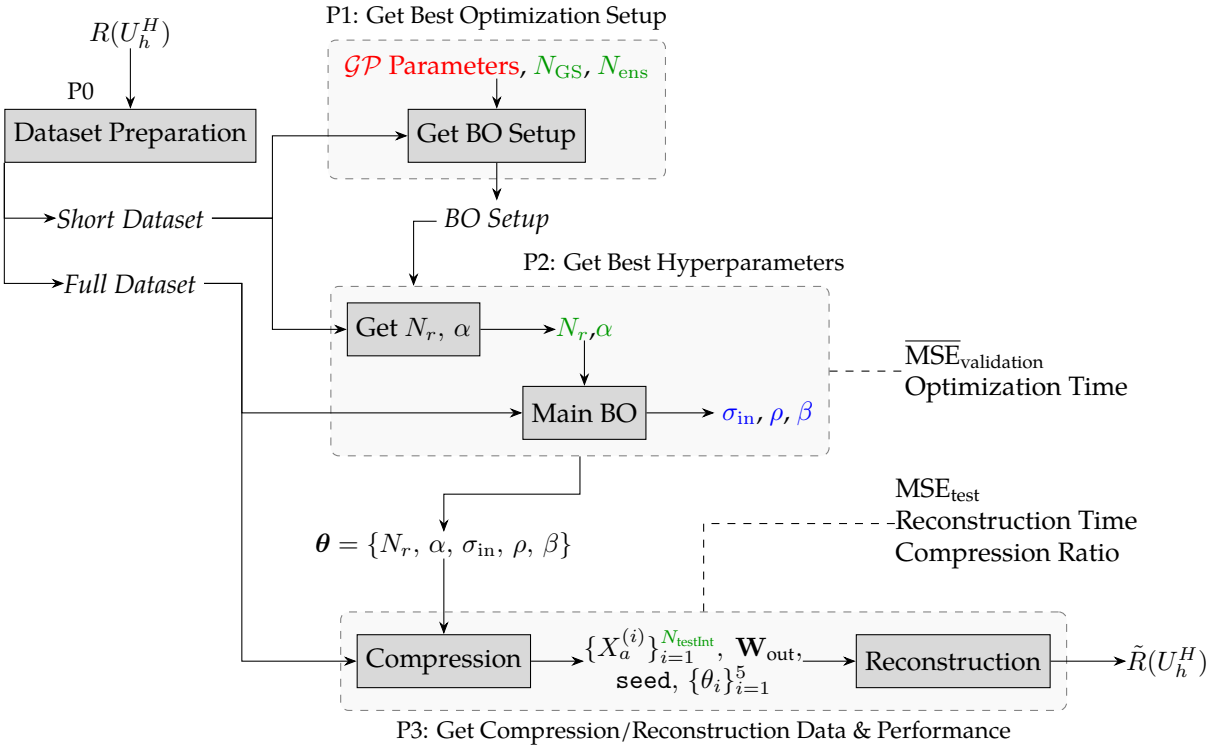


Figure 5.3: Schematic of the ESN hyperparameter optimization and compression framework. Parameters highlighted in red are fixed based on values suggested in the literature, those in green are determined empirically, and those in blue are obtained through optimization.

P0: Dataset Preparation. The process begins with dataset preparation, where two datasets are constructed. A short dataset is generated for framework tuning purposes, while the full dataset is reserved for the actual training, optimization, and compression tasks.

P1: Optimization Setup. A Bayesian Optimization (BO) procedure was employed to identify the op-

timal values of the spectral radius (ρ) and input scaling (σ_{in}) hyperparameters of the ESN. Due to the high computational cost associated with repeated ESN training and validation, BO was chosen as a sample-efficient strategy. The objective function, defined as the average logarithmic mean squared error (log-MSE) across multiple validation intervals, was treated as a black-box.

A Gaussian Process ($\mathcal{GP}$) regression surrogate with a Matérn $5/2$ kernel and Automatic Relevance Determination (ARD) was used to approximate this function without requiring gradient information [55]. The optimization process began with an initial Grid Search (GS) across $N_{GS} = N_s \times N_s$ points, uniformly covering the search domain. The grid size N_s was not fixed a priori; instead, it was determined empirically for each test case by running the optimization on the short dataset with different grid sizes and 42 independent realizations. The choice of 42 realizations served two purposes: first, it matched the available hardware (14 physical CPU cores, enabling 42 runs to be executed efficiently in parallel); and second, it ensured that the effect of the number of realizations on the selection of N_{GS} was minimized, thereby improving the robustness of the grid size determination.

After the best N_{GS} was chosen, a secondary sweep using this grid was performed to determine the appropriate ensemble size N_{ens}, ensuring a trade-off between robustness and computational cost. This adaptive strategy ensured that the final ensemble-based optimization captured the general behavior of the network while avoiding overfitting to random fluctuations.

Following this initialization, an additional 10 BO evaluations were performed using the Expected Improvement (EI) acquisition function, which balances exploration and exploitation [55, 126]. For each candidate $(\rho, \sigma_{\text{in}})$ pair, the Tikhonov regularization parameter (β) of the Ridge regression readout was optimized through a nested grid search over a predefined list. The best-performing configuration of hyperparameters for each realization was selected based on the lowest validation log-MSE, and the overall optimal ESN configuration was identified as the one with the lowest ensemble-averaged loss.

P2: Hyperparameter Selection. After the BO setup was finalized, the number of neurons in the reservoir (N_r) and subsequently the leaking rate (α) were determined empirically on the short dataset, as suggested in the literature [104]. With these parameters fixed, the main BO procedure was executed to obtain the final optimal hyperparameter set for each dataset:

$$\theta = \{N_r, \alpha, \sigma_{\text{in}}, \rho, \beta\},$$

P3: Compression and Reconstruction. Finally, the optimized ESN was used for data compression. During this stage, the following quantities were stored: the trained readout matrix $\mathbf{W}_{\text{out}}$, the optimal hyperparameters, the associated random seed for reproducibility, and the checkpoint states. A checkpoint technique was introduced to prevent the accumulation of errors as time progresses. The number of checkpoints, N_{testInt}, was selected empirically to balance reconstruction accuracy against compression ratio. In the reconstruction stage, these stored quantities were used in a closed-loop configurationto to reproduce the compressed time series of $\tilde{R}(U_h^H)$ and $\tilde{U}_h$.

Consequently, the compression ratio (CR) for generating a surrogate model was defined as

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Checkpoint States} + \text{Size}(W_{\text{out}}) + \text{Hyperparameters}}. \tag{5.12}$$

The use of checkpointing requires storing both the input and reservoir states at each checkpoint interval. Among the different storage components, the output weight matrix W_{out} contributed most significantly to the CR due to its dense structure. By contrast, the input and reservoir weight matrices, W_{in} and W_r, did not need explicit storage, as they can be uniquely reconstructed from the hyperparameters σ_{in} and ρ when a fixed random seed is specified.

5.4. Performance Assessment Framework

To enable a consistent comparison between the proposed compression techniques, a robust benchmarking strategy and a unified set of performance metrics were established. For this purpose, the POD was adopted as the benchmark method. In the POD framework, the only user-defined parameter is the threshold ε_k, which specifies the fraction of kinetic energy to be retained in the reduced-order representation. In this study, $\varepsilon_k \leq 99\%$ was selected for the primal solution, whereas for the injected

residuals, the threshold value was determined empirically by testing multiple configurations for each test case.

Once the POD setup was finalized, the reconstructed fields were compared against the reference data using the MSE, defined as:

$$\text{MSE}_{\text{test}}(\tilde{u}) = \frac{1}{N_t N_x} \sum_{i=1}^{N_t} \sum_{j=1}^{N_x} \left(u(i,\, j) - \tilde{u}(i,\, j)\right)^2, \tag{5.13}$$

where $u(i,\, j)$ denotes the reference data (either the primal solution U or the injected residuals $R(U_h^H)$), $\tilde{u}(i,\, j)$ represents the reconstructed field, and N_t and N_x are the temporal and spatial dimensions, respectively.

In addition to MSE_{test}, three complementary metrics were employed:

- **Compression Ratio (CR):** ratio of the size of the original dataset to the total size of the stored parameters required for reconstruction.
- **Preparation Time:** this represents the total time required to generate the surrogate model; it corresponds to the training time for the CAE, the optimization time for the ESN, and the time required to compute the full set of eigenvalues for the POD.
- **Reconstruction Time:** the computational time required to rebuild the data from the compressed representation.

Two different computational devices were used during the training and optimization processes. The first, referred to as *Device 1*, was an NVIDIA Tesla P40 Graphics Processing Unit (GPU), a high-performance accelerator optimized for deep learning tasks. With thousands of CUDA cores and a large memory bandwidth, it provides efficient parallelization of matrix operations, making it ideal for training CAE. The second, *Device 2*, was an Apple MacBook Pro equipped with an M4 Pro chip comprising 14 physical CPU cores. Unlike GPUs, CPUs are optimized for sequential and control-intensive operations. Therefore, the ESN training and Bayesian hyperparameter optimization were performed on the CPU, as ESNs are relatively lightweight and do not require the high degree of parallelization demanded by CAEs.

The POD results serve as the benchmark for evaluating the performance of the CAE and ESN methods. The objective in both cases was to achieve a similar order of magnitude in the test MSE as the POD at a given refinement level and then to compare the corresponding compression ratio and reconstruction time. Additionally, to provide a consistent comparison across methods, CAE and ESN results were also evaluated at a constant compression ratio, by adjusting the POD threshold to match the achieved CR of the surrogate models.

To further isolate and analyze the different sources of error introduced by the compression and reconstruction of various components within the adjoint-based error estimation pipeline, three distinct assessment scenarios were designed, as illustrated in Figure 5.4.

- *Scenario 1:* evaluates the error arising solely from the compression and reconstruction of the injected residuals $\tilde{R}(U_h^H)$.
- *Scenario 2:* assesses the impact of compressing and reconstructing the fine primal solution $\tilde{U}_h$ used in the adjoint computation.
- *Scenario 3:* represents the combined configuration, where both the injected residuals and the fine primal solution are compressed and reconstructed simultaneously to estimate their collective effect on the final error indicators.

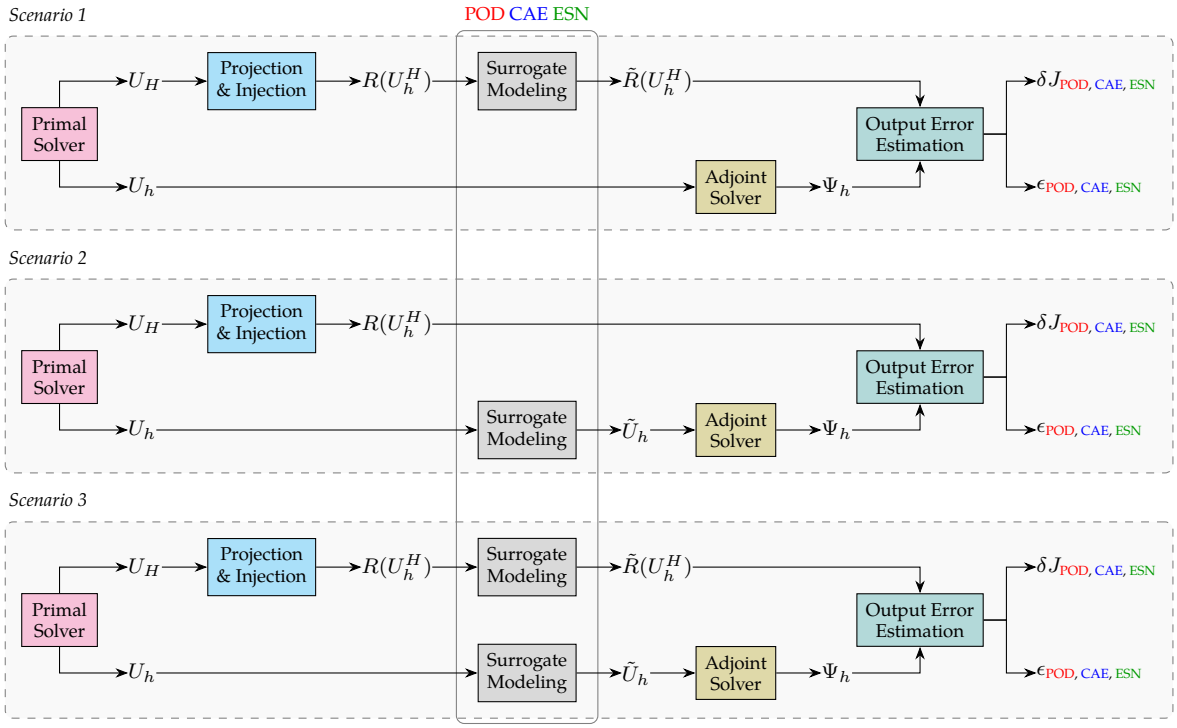


Figure 5.4: Schematic representation of the three performance assessment scenarios.

6

Framework Verification: Manufactured Solution

This chapter focuses on the verification of the proposed framework using a Manufactured Solution (MMS). The MMS serves as a controlled benchmark in which the primal solution is known analytically, allowing for the exact evaluation of its residual and the associated error estimation. Such a setting provides a rigorous basis for verifying the accuracy of the numerical solver, the fidelity of surrogate models in representing the primal and residual fields, and the consistency of the adjoint-based error estimation procedure.

The chapter is organized into several sections: the primal solution is first examined, followed by the compression and reconstruction of injected residuals and the fine primal solution, and the computation of the corresponding adjoint solution. Finally, adjoint-based error estimation and localization are assessed across different surrogate-integration scenarios, and a brief summary of the key findings is provided at the end of the chapter.

It should be noted that the primary objective of this chapter is to verify the proposed framework rather than to draw comparative conclusions between the surrogate modeling methods. Due to the inherent smoothness and simplicity of the MMS, all models are expected to perform similarly. Therefore, only the key verification results are presented herein, while the complementary and extended results are provided in Appendix C.

6.1. Problem Formulation

The 1D unsteady viscous Burgers' equation, a second-order nonlinear PDE, is employed in this study within the MMS framework to verify the implemented methodology. The MMS approach has become a widely adopted technique for code verification [28, 41], as it enables the construction of an exact analytical solution to a PDE by introducing a tailored source term. This allows for rigorous testing of numerical solvers and frameworks without requiring the equation to represent a physically realistic scenario, thus isolating and evaluating the numerical accuracy of the method under investigation [128].

This PDE serves as a fundamental model across various physical disciplines, as it is frequently employed to study simplified representations of turbulence, boundary layer development, shock wave dynamics, and mass transport phenomena. The equation retains the essential nonlinear convective term characteristic of the NS equations while incorporating diffusion and temporal evolution in a mathematically tractable form. This balance between nonlinearity and simplicity makes Burgers' equation an attractive test case for theoretical and numerical investigations. Furthermore, the availability of exact analytical solutions [129] enhances its utility as a benchmark for the verification of computational solvers [130].

As part of the verification process within this framework, the following manufactured solution is pre-

scribed [41, 28]:

$$u_{\text{MMS}} = \sin^2(\pi t)\, \sin(\pi x). \tag{6.1}$$

Additionally, the primal PDE is defined over a space-time domain $\Omega = [0,\, L] \times I = [0,\, T]$, where $L = 1.0$ and $T = 20$ [s]. The interval $[0,\, 10]$ is allocated for flow development, while the period $[10,\, 20]$ corresponds to the attainment of a statistical steady state [28]. Homogeneous Dirichlet Boundary Conditions (BCs) and a homogeneous Initial Condition (IC) are prescribed to fully define the problem, leading to the following formulation of the primal PDE:

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} - \dfrac{1}{Re}\dfrac{\partial^2 u}{\partial x^2} = f_{\text{MMS}}, & (x,t) \in \Omega \times I, \\ u(x,t) = 0, & (x,t) \in \partial\Omega \times I, \\ u(x,0) = 0, & x \in \Omega. \end{cases} \tag{6.2}$$

Here, x and t denote the spatial and temporal coordinates, respectively; u represents the primal solution; and f_{MMS} is the manufactured source term. The Reynolds number is set to $\text{Re} = 100$, where $\text{Re} = \frac{UL}{\nu}$, with U and L being the reference velocity and length scales. Since both are set to unity in the present formulation, the Reynolds number is fully controlled through the kinematic viscosity ν, as defined in the OpenFOAM setup. The expression for f_{MMS} is obtained by substituting the manufactured solution of Eq. (6.1) into the governing PDE of Eq. (6.2).

Considering the application of the error estimation framework within an AMR context using an h-adaptation strategy with a refinement factor of two, the framework was assessed over a range of coarse-grid spatial resolutions $N_x^{(H)} \in \{16,\, 32,\, 64,\, 128,\, 256\}$ with particular attention given to the case $N_x^{(H)} = 32$ for detailed visualization.

In this phase of the research, a time-averaged QoI, denoted as $\bar{J}$, was adopted and defined as

$$\bar{J} = \frac{1}{T}\int_I \int_\Omega \sin(\pi x)\, u(x,t)\, d\Omega\, dI, \tag{6.3}$$

which emphasizes the solution in the central region of the spatial domain, thereby reducing the sensitivity of the adjoint and subsequent error estimation to inaccuracies introduced near the boundaries. The manufactured solution yields an exact value of $\bar{J}_{\text{MMS}} = 2.5 \times 10^{-1}$. The same QoI have been previously investigated in the context of FEM discretizations [41, 28].

6.2. Primal Solution

A grid convergence study was performed to verify the implemented solver and to determine an appropriate time step. To this end, the Root Mean Square Error (RMSE) between the discrete solution on each grid and the corresponding manufactured solution evaluated on the same grid was computed as

$$\text{RMSE}(u) = \sqrt{\frac{1}{N_t N_x}\sum_{j=1}^{N_t}\sum_{i=1}^{N_x}\left[u(x_i,t_j) - u_{\text{MMS}}(x_i,t_j)\right]^2}, \tag{6.4}$$

where the respective N_t and N_x are the number of time steps and spatial elements. In addition, the QoI $\bar{J}$ was evaluated for each grid, and the output error was quantified as $\Delta\bar{J}_H = \bar{J}_{\text{MMS}} - \bar{J}_H$. These values were calculated for all spatial resolutions and for time steps $\Delta t \in \{10^{-3},\, 10^{-4},\, 10^{-5}\}$, with the results presented in Figure 6.1.

The left panel of Figure 6.1 shows that $\bar{J}$ converges rapidly to the exact value as the grid is refined. Consequently, the output error $\Delta\bar{J}_H$ shows a nearly monotonically decreasing trend with grid refinement, which follows the expected second-order convergence behavior with respect to grid resolution. Moreover, the difference between the QoI calculated from the coarse and fine space solutions, $\bar{J}_h - \bar{J}_H$, follows the same trend. The right panel confirms this trend, where the RMSE demonstrates an overall $\mathcal{O}(h^2)$ convergence rate, in agreement with the second-order spatial discretization employed. Overall, the implemented solver was verified based on the aforementioned observations.

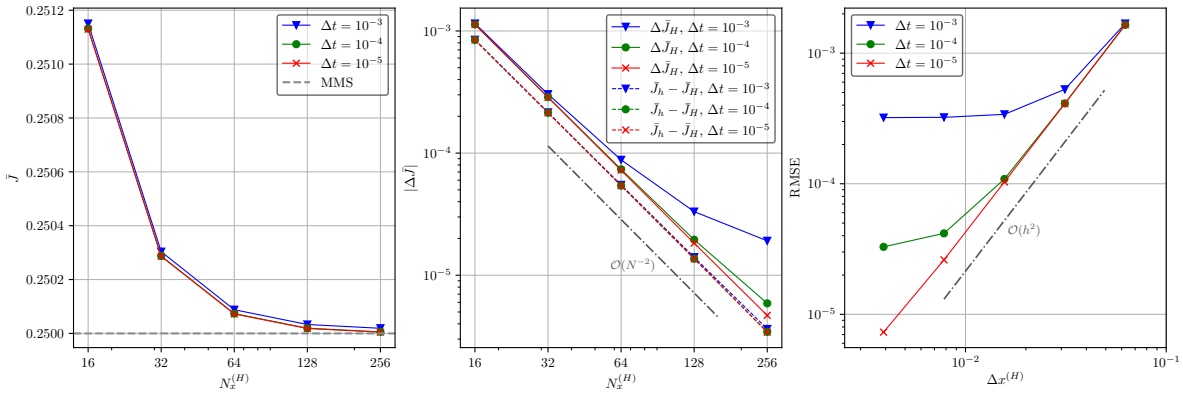


Figure 6.1: Grid convergence results for the manufactured solution. (Left) Convergence of the QoI $\bar{J}$ towards the exact value. (Center) Output error $\Delta \bar{J}_H$ for different time steps, demonstrating second-order accuracy with grid refinement. (Right) RMSE of the solution field with respect to the exact solution, confirming $O(h^2)$ convergence. Results are shown for $\Delta t \in \{10^{-3}, 10^{-4}, 10^{-5}\}$.

Regarding the choice of time step, $\Delta t = 10^{-5}$ provides the smallest errors across all metrics. However, since the purpose of this chapter is to verify the methodology rather than to achieve the absolute minimum error, $\Delta t = 10^{-4}$ was selected as a suitable compromise between accuracy and computational efficiency. The coarser step of $\Delta t = 10^{-3}$ clearly introduces temporal errors that dominate on finer meshes, whereas $\Delta t = 10^{-4}$ ensures sufficient accuracy while maintaining a practical computational cost.

Figure 6.2 presents the primal discrete solution u_H for $N_x^{(H)} = 32$ grid, alongside its relative error with respect to the manufactured solution. The relative error is defined as the absolute deviation normalized by the maximum value of the manufactured solution.

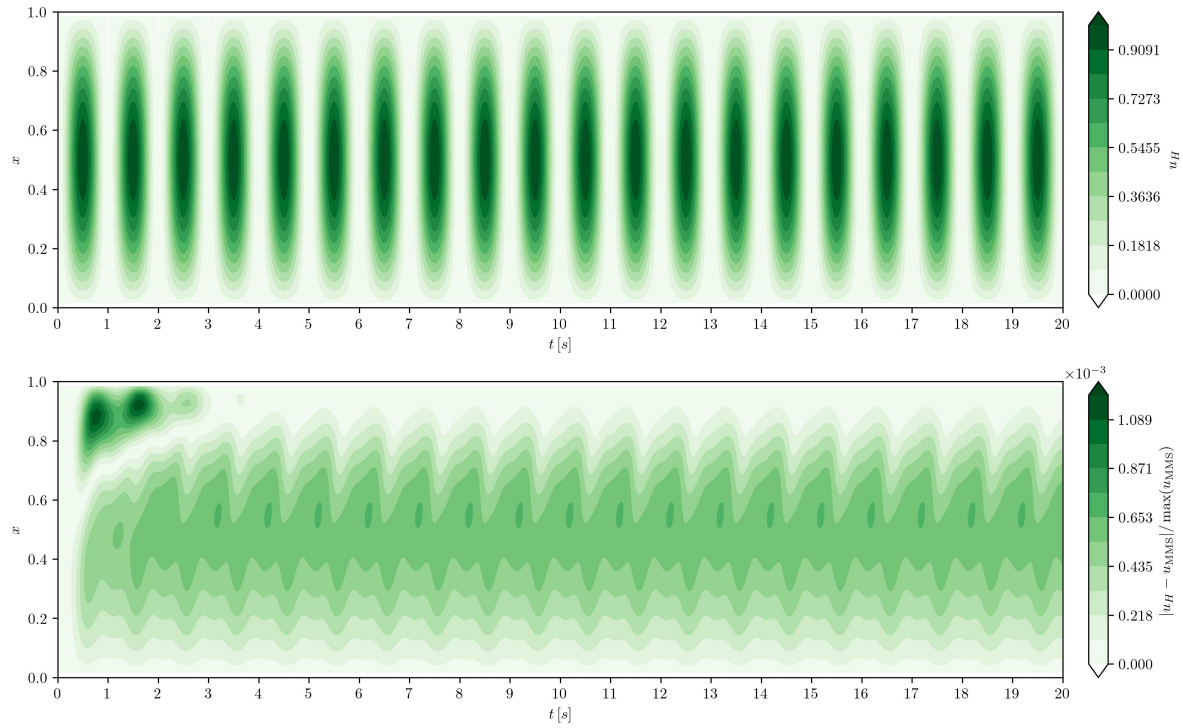


Figure 6.2: Primal discrete solution u_H (top) and its relative error with respect to the manufactured solution (bottom) for the spatial resolution of $N_x^{(H)} = 32$.

The discrete solution displays a smooth and periodic profile, with oscillations centered around the midpoint of the spatial domain. This behavior confirms that the numerical scheme correctly reproduces

the periodic dynamics of the chosen manufactured solution. The associated error field reveals two distinct regimes. During the initial transient, the error is non-periodic and exhibits localized peaks near the spatial boundary of $x = 1$. This is attributed to the fact that the prescribed initial condition does not coincide with the long-time periodic solution. As the simulation progresses, the error evolves into a periodic pattern, also centered near the spatial midpoint, with a relative error of $\approx \mathcal{O}(10^{-3})$.

From these observations, the time interval $[10,\ 20]$ was identified as statistically steady with respect to the error distribution, which is also stated by Li et al. [28]. This interval was, therefore, adopted as the temporal domain for subsequent analyses.

6.3. Injected Residual Compression

As discussed in Chapter 3, one of the required datasets for evaluating the output error estimate is the injected residuals. These residuals were obtained by first projecting the coarse solution u_H onto its corresponding refined computational grid, resulting in the fine-grid representation u_h^H. In this phase of the research, a cubic spline interpolation scheme was employed for this purpose. To ensure consistency, the corresponding forcing term was projected onto the fine grid using the same scheme.

The projected solution u_h^H was then substituted into the primal PDE of Eq. (6.2) to compute the injected residual, denoted as $R(u_h^H)$. Mathematically, this is expressed as

$$R(u_h^H) = \frac{\partial u_h^H}{\partial t} + u_h^H \frac{\partial u_h^H}{\partial x} - \frac{1}{Re}\frac{\partial^2 u_h^H}{\partial x^2} - f_{\text{MMS}}|_h^H \,. \tag{6.5}$$

Figure 6.3 illustrates the injected residuals $R(u_h^H)$ for the case with $N_x^{(H)} = 32$. The residual field exhibits a clear periodic structure in time, reflecting the periodic nature of the manufactured solution. Spatially, alternating positive and negative regions appear symmetrically around the midpoint of the domain, consistent with the sinusoidal forcing introduced in the MMS definition.

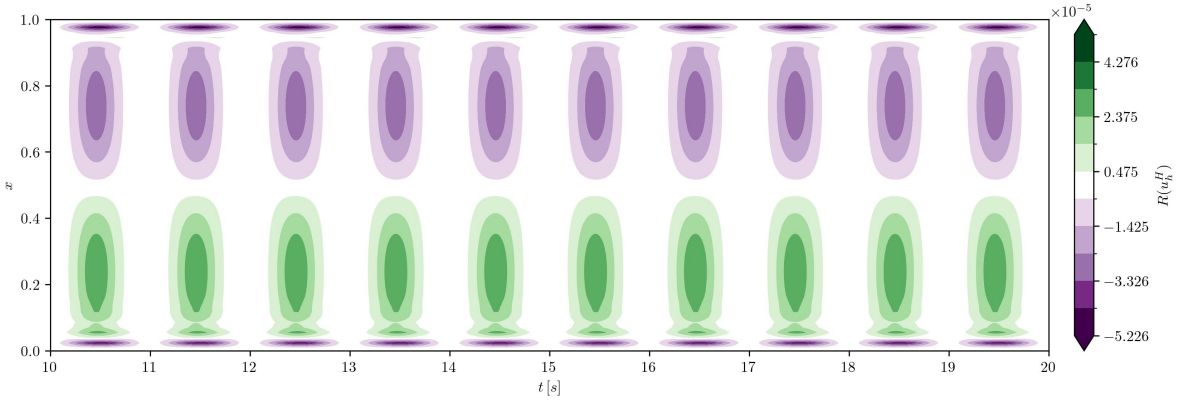


Figure 6.3: Injected residuals $R(u_h^H)$ for $N_x^{(H)} = 32$.

The magnitude of the residuals remains small, $\mathcal{O}(10^{-5})$, confirming that the projection and residual injection procedure preserves accuracy when transferring the coarse solution to the refined grid. The localized concentration of residuals near the domain boundaries indicates that the coarse grid does not fully capture the finer-scale dynamics of the PDE in these regions, where projection errors are amplified by the imposed boundary conditions. Nevertheless, these effects remain bounded and do not contaminate the overall residual distribution.

Figure 6.4 shows the temporal mean and standard deviation of the injected residuals $R(u_h^H)$ across different spatial resolutions. A clear reduction in the magnitude of the mean residual is observed as the grid is refined, confirming that the projection and injection procedures converge consistently with mesh resolution. For the coarsest mesh ($N_x^{(H)} = 16$), residual fluctuations are significant, with both the mean and variance reaching $\mathcal{O}(10^{-4})$. As the resolution increases, the magnitude of both the mean and standard deviation decreases consistently by several orders.

The narrowing of the shaded regions with refinement indicates that temporal oscillations in the injected residuals become progressively smaller, demonstrating improved consistency between the projected coarse solution and the refined PDE operator. This behavior ensures that the injected residuals remain well-controlled and suitable for use in adjoint-based error estimation.

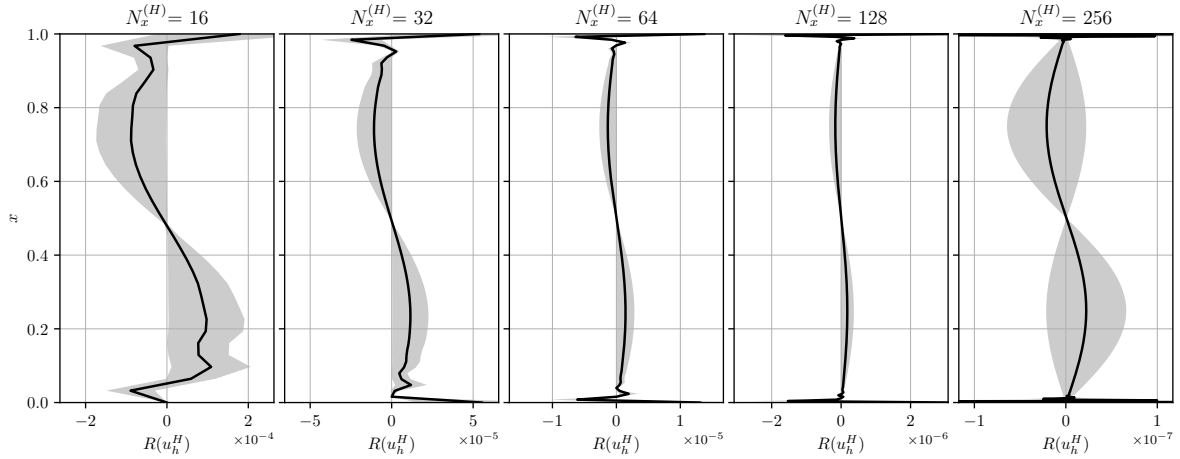


Figure 6.4: Temporal mean (solid line) and standard deviation (shaded region) of the 1D Burgers' MMS injected residuals $R(u_h^H)$ for all different spatial resolutions $N_x^{(H)}$.

6.3.1. Proper Orthogonal Decomposition

The results of the offline POD benchmark of the reconstructed injected residuals for different spatial resolutions are summarized in Figure 6.5. As illustrated, the retained modal energy $\mathcal{E}_k$ converges rapidly, with more than 95% of the energy captured by fewer than three modes for all refinement levels. This behavior is expected, given the smooth nature of the manufactured solution, which leads to highly coherent residual structures that can be effectively represented with a compact modal basis. Additional modes only marginally increase the retained energy, providing negligible benefit while increasing reconstruction complexity.

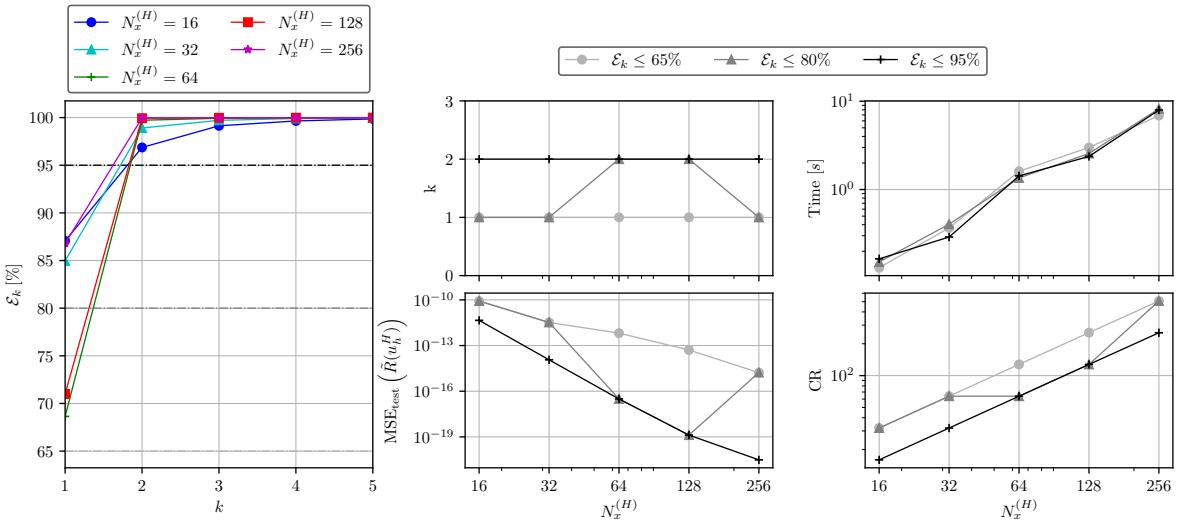


Figure 6.5: Offline POD benchmark results of reconstructed 1D MMS Burgers' injected residuals for different spatial resolutions $N_x^{(H)}$. (Left) Retained modal energy $\mathcal{E}_k$ versus retained mode k (Right) Number of retained modes k, test MSE, computation time, and compression ratio for thresholds of 65%, 80%, and 95%. (All x-axes are in log scale.)

When comparing the thresholds of 65%, 80%, and 95% for retained energy, it becomes evident that the lower threshold already delivers sufficiently accurate reconstructions. Although stricter thresholds reduce the test MSE by several orders of magnitude, the gains in accuracy are negligible when com-

pared to the additional computational time and loss of compression. For instance, the compression ratio increases sharply at higher spatial resolutions but is substantially reduced when more modes are enforced.

Based on these observations, the threshold $\mathcal{E}_k \leq 65\%$ was selected as the POD reference setting. This choice strikes a balance between accuracy and efficiency, as it avoids the excessive cost associated with 80% and 95% thresholds while still preserving the dominant features of the injected residuals. The adopted configuration, therefore, serves as a benchmark for assessing alternative surrogate modeling strategies in the remainder of this section.

6.3.2. Convolutional Autoencoder

The first proposed compression methodology was the CAE. As outlined in Section 5.3.2, the initial estimate of the suitable latent space dimension $d_{\mathcal{Z}}$ was obtained by performing a PCA on the dataset of injected residuals. This preliminary step ensures that the CAE architecture is designed to capture the dominant variance of the data while avoiding unnecessary large latent dimensions. Table 6.1 reports the selected values of $d_{\mathcal{Z}}$ for different spatial resolutions.

Table 6.1: Latent space dimension $d_{\mathcal{Z}}$ determined for the CAE architecture based on PCA analysis of injected residuals from the 1D MMS Burgers' problem.

$N_x^{(H)}$	16	32	64	128	256
$d_{\mathcal{Z}}$	2	2	2	2	2

Across all refinement levels, the analysis consistently indicates that $d_{\mathcal{Z}} = 2$ is sufficient to represent the dataset's features. This outcome reflects the inherent simplicity and low-rank structure of the residual statistics, where most of the variability can be effectively captured by only two latent variables. It should be noted that this simplicity is a direct consequence of the manufactured solution, which produces residuals with smooth and highly structured dynamics. As a result, the CAE achieves a stable and compact representation of the residuals, independent of the underlying grid resolution.

After defining the suitable latent dimension $d_{\mathcal{Z}}$, a specific CAE architecture was generated for each spatial resolution using Algorithm 3. In the test case of the 1D MMS injected residuals of Burgers' equation, the parameters were set to $c_{\text{in}} = 1$, $c_{\text{base}} = 4$, $c_{\text{max}} = 4$, and the kernel size $k = 4$. As an illustrative example, Figure 6.6 shows the generated CAE architecture for the injected residuals of $N_x^{(H)} = 16$. The corresponding encoder and decoder details are reported in Tables 6.2 and 6.3, where k, s, and p represent the kernel size, stride, and padding, respectively.

The encoder is composed of five successive convolutional layers with BN and LReLU activations, progressively reducing the spatial dimension while increasing feature abstraction. The output is then flattened and mapped to the latent space of size $d_{\mathcal{Z}} = 2$ through a fully connected layer. The decoder mirrors this structure in reverse, using upsampling layers followed by convolutional blocks with BN and LReLU to gradually reconstruct the residual field at the original resolution. This symmetric design ensures both stability in training and consistency between compression and reconstruction. The remaining architectures for higher resolutions were generated analogously and are presented in Appendix B.1.

Regarding the optimization procedure, a batch size of 200 samples was selected. This value provides a compromise between stable gradient updates and computational efficiency. The training and validation datasets were generated from a subsampled residual dataset with a sampling rate of 10, resulting in a time series of length 10,000. This subsampling ensures that the dataset retains the essential temporal dynamics while keeping the computational cost tractable.

Training was performed over 400 epochs. An adaptive learning rate strategy was employed during the first 350 epochs, following the formulation in Eq. (5.8), with an initial learning rate of $\gamma_0 = 5 \times 10^{-3}$ and a decay factor of $\gamma_r = 2 \times 10^{-2}$. After the adaptive phase, the learning rate was fixed at the final value for the remaining 50 epochs to stabilize the optimization and prevent oscillations near the minimum. It should be noted that the hyperparameter settings reported above were determined empirically through trial and error.

Table 6.2: Layer-by-layer details of the encoder architecture applied to 1D MMS Burgers' injected residuals for the case $N_x^{(H)} = 16$

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 32)$	$(4, 16)$	$(4, 2, 1)$	16
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E2	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E3	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E4	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E5	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E6	Flatten	$(4, 1)$	(4)		0
E7	Linear	(4)	(2)		10

Table 6.3: Layer-by-layer details of the decoder architecture applied to 1D MMS Burgers' injected residuals for the case $N_x^{(H)} = 16$

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(2)	(4)		12
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(1, 32)$	$(3, 1, 1)$	12

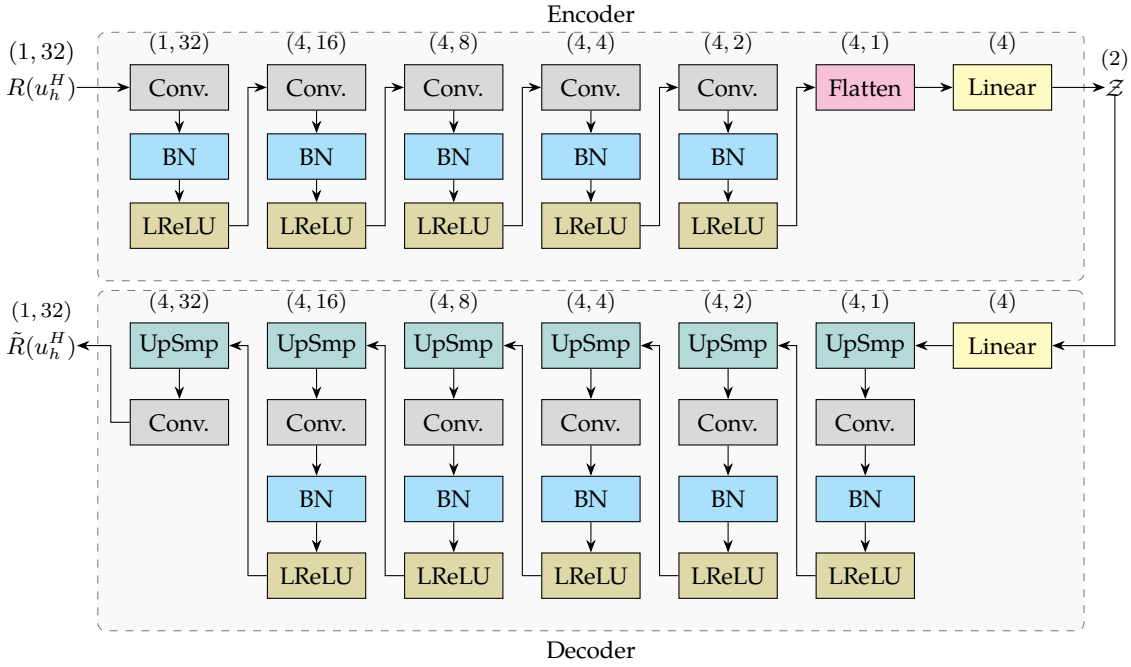


Figure 6.6: Illustration of the CAE architecture generated for the injected residuals of the 1D MMS Burgers' on $N_x^{(H)} = 16$.

Figure 6.7 shows the evolution of the batch-averaged MSE during training and validation across different spatial resolutions $N_x^{(H)}$, along with the learning rate schedule. In all cases, the MSE decreases rapidly during the first epochs, confirming that the CAE is able to capture the dominant features of the injected residuals. As training progresses, both training and validation losses continue to decrease, eventually stabilizing around values of $\mathcal{O}(10^{-2})$. The close agreement between the training and validation curves indicates no evidence of overfitting, even at higher resolutions.

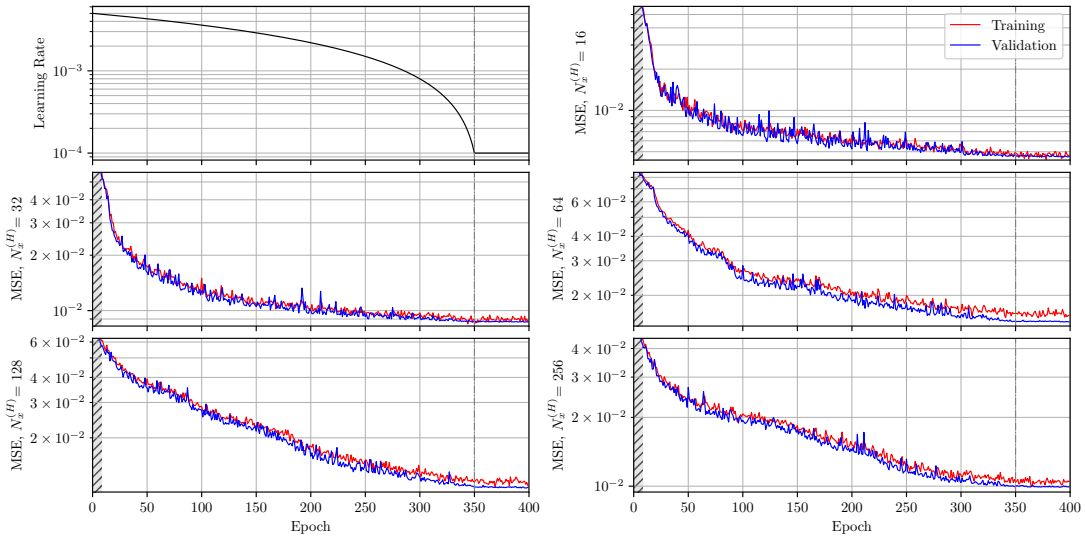


Figure 6.7: Training and validation batch-averaged MSE during CAE training of 1D MMS Burgers' injected residuals for different spatial resolutions $N_x^{(H)}$. The top-left panel shows the adaptive learning rate schedule applied during the first 350 epochs, followed by a constant value. (The initial epochs are truncated in all MSE plots for visualization purposes.)

The observed oscillations, particularly in the early phase of training, are linked to the adaptive learning rate schedule, which deliberately explores larger steps in parameter space before decaying to a stable regime. Once the learning rate is fixed after 350 epochs, the loss stabilizes and converges smoothly, showing that the network has reached a consistent representation of the residual statistics. Although the convergence is not perfectly monotonic, the overall trend demonstrates the effectiveness of the CAE

in compressing the residual fields while maintaining stability across resolutions.

It is worth noting that further reduction of the MSE could likely be achieved by extending the number of training epochs or applying a more systematic hyperparameter tuning procedure. Nevertheless, the present results are sufficient to establish the CAE as a reliable compression model in the context of the MMS test case.

6.3.3. Echo State Network

The second proposed compression method was the ESN, whose architecture consisted of a single pseudorandomly constructed reservoir. The training was performed by solving a linear regression problem to determine the connections between the reservoir states and the output layer. The dataset used for training comprised the full temporal history of 100,000 steps, while the validation set was constructed from 40 non-overlapping intervals of 2,500 samples each.

The hyperparameter search space for BO was defined as $\rho \times \sigma_{\text{in}} : [0.1,\, 1] \times [10^2,\, 10^6]$ for the respective spectral radius and input scaling in logarithmic scale, and the optimal Tikhonov regularization parameter was selected from a grid space of $\beta \in \{10^{-3},\, 10^{-6},\, 10^{-9}\}$.

The choice of these ranges was motivated by the statistical properties of the injected residuals. In particular, Figure 6.3 shows that the magnitude of the residuals decreases rapidly with spatial refinement, spanning several orders of magnitude across $N_x^{(H)}$. To account for this variability, a wide range of σ_{in} was necessary to ensure that the ESN reservoir received inputs at a comparable dynamic range, regardless of the resolution. This scaling prevents under-activation in cases where the residual magnitudes are small.

The selected range for ρ ensured coverage of both the contractive regime ($\rho < 1$), which enforces stability, and the near-critical regime ($\rho \approx 1$), which enhances memory capacity. Finally, the choice of β values allowed the BO to balance numerical stability and the risk of overfitting during the linear regression stage. As described in Section 5.3.3, the number of neurons N_r and the leaking rate α were instead determined empirically through the detailed analysis presented later in this section.

The procedure performed to set up the BO is described in Appendix C.1. Following the finalization of the BO setup, the effect of the reservoir size N_r and the leaking rate α was investigated for the case $N_x^{(H)} = 32$, as discussed in Section 5.3.3. The goal of this analysis was to identify a configuration that balances predictive accuracy, compression efficiency, and computational cost.

Figure 6.8 summarizes the results for varying reservoir sizes with a constant leaking rate of $\alpha = 0.6$. As expected, the validation MSE decreases sharply as N_r increases, reflecting the enhanced representational capacity of larger reservoirs. However, this reduction follows the law of diminishing returns; beyond $N_r \approx 64$, the accuracy gains are marginal compared to the additional computational burden. At the same time, the CR decreases with N_r, since enlarging the reservoir directly reduces the relative compression achieved by the ESN. In parallel, the optimization time grows moderately up to $N_r \approx 128$, but rises steeply for larger reservoirs due to the expansion of the search space. Taken together, these observations highlight $N_r = 64$ as an effective compromise, providing near-optimal accuracy while maintaining a favorable balance between compression efficiency and computational cost.

It is important to emphasize that the choice of such a relatively small reservoir size, compared to the dimensionality of the input residual vectors, is theoretically justified. This is possible because the effective dimensionality of the reservoir's active subspace can be significantly smaller than the number of input variables [131]. As a result, a reduced reservoir can still capture the essential dynamics without sacrificing predictive performance [132].

The influence of the leaking rate α is shown in Figure 6.9. The results reveal that the ESN performs poorly in the absence of leakage ($\alpha = 0$), where reservoir dynamics become overly stiff and fail to capture temporal dependencies. Introducing leakage leads to a substantial improvement, with the best performance observed around $\alpha = 0.6$. Beyond this value, the validation error gradually increases, indicating that excessively large leaking rates diminish the reservoir's ability to integrate information across time. It is also worth noting that α does not significantly impact either the optimization time or the compression ratio; for this reason, these metrics are not reported here. Based on this analysis, $\alpha = 0.6$ was chosen as the default setting.

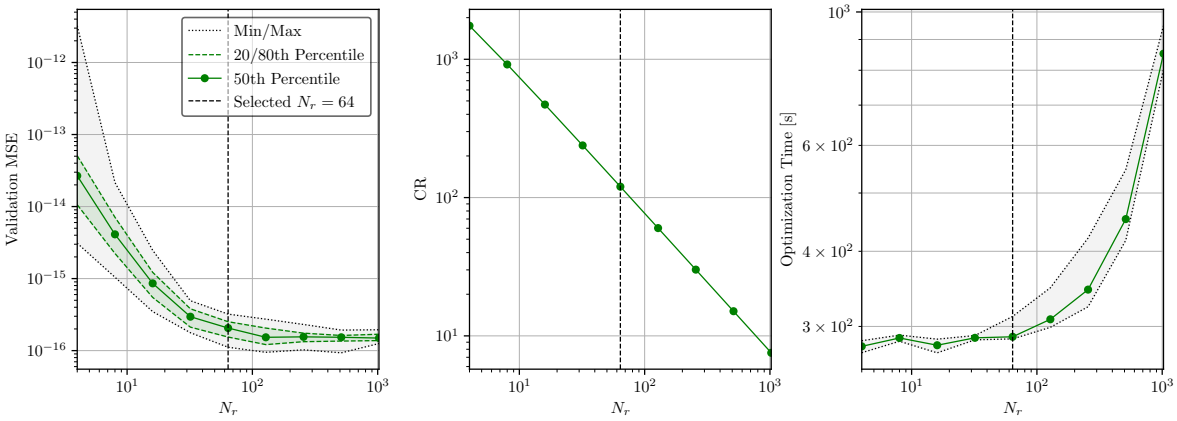


Figure 6.8: Effect of the number of reservoir neurons N_r on ESN performance. (Left) Validation MSE, (Center) Compression ratio, and (Right) Optimization time.

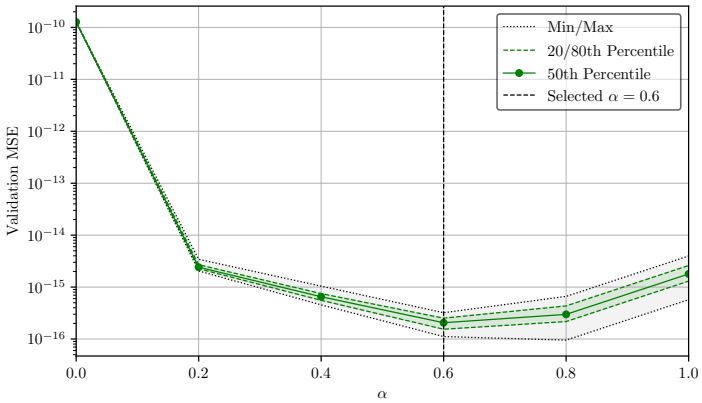


Figure 6.9: Effect of the leaking rate α on the ESN performance of validation MSE. Since α has no significant influence on optimization time or compression ratio, these metrics are omitted.

6.3.4. Performance Assessment

This section presents the performance assessment of the proposed surrogate models, namely the benchmark POD, CAE, and ESN, applied to the reconstruction of the injected residuals across all refinements.

Figure 6.10 presents a comparison of POD, CAE, and ESN across reconstruction accuracy and compression efficiency for the 1D Burgers' MMS injected residuals. Across all refinements, the reconstruction error decreases as the resolution increases, confirming that all models benefit from richer spatial representations. Among the three, the CAE consistently achieves the lowest errors, highlighting the effectiveness of nonlinear encodings in capturing localized features of the residuals. ESN provides competitive accuracy, particularly at higher resolutions, though its variability at coarse grids reflects the sensitivity of reservoir initialization. POD, while effective, shows limitations when the truncation threshold is low, leading to comparatively larger errors in coarser cases.

These trends come at different costs in terms of compression and computational requirements. The ESN achieves the highest compression ratios due to its compact reservoir structure, though this comes at the price of instability in some cases. POD maintains a balance between compression and interpretability but sacrifices accuracy as refinement increases. By contrast, the CAE achieves superior accuracy with more stable behavior across resolutions; however, its compression ratio remains lower due to the overhead associated with its trainable parameters.

The preparation and reconstruction times of the surrogate models are summarized in Figure 6.11, where the left panel reports the preparation time, and the right panel shows the corresponding reconstruction time as a function of spatial refinement. The preparation time represents the total cost required to generate the surrogate model: it corresponds to the training time for the CAE, the optimization time

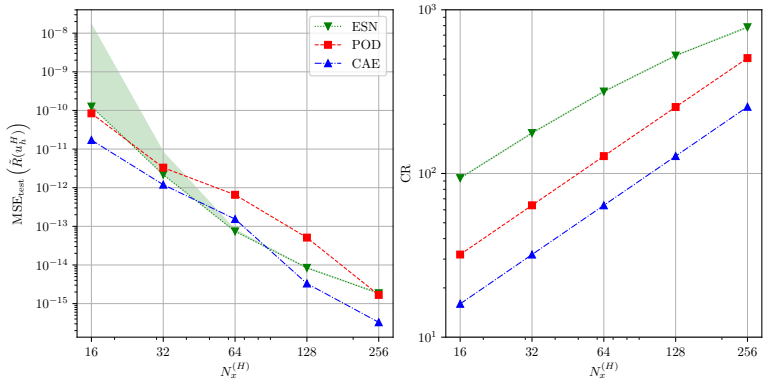


Figure 6.10: Performance comparison of POD, CAE, and ESN for the 1D Burgers' MMS injected residuals, showing test MSE and compression ratio (CR) as functions of spatial refinement $N_x^{(H)}$.

for the ESN, and the time required to compute the full set of eigenvalues for the POD, referred to as the computation time.

It should be noted that, due to the stochastic dependence of the ESN optimization and the device-specific training environment of the CAE, these time-related metrics were excluded from the performance comparison in terms of compression ratio and reconstruction fidelity, which were presented earlier in this section. The CAE training was performed on *Device 1*, while the ESN optimization and POD computation were executed on *Device 2*. Therefore, the reported values are hardware-specific and are mainly intended to illustrate the scaling trends rather than to provide absolute performance benchmarks.

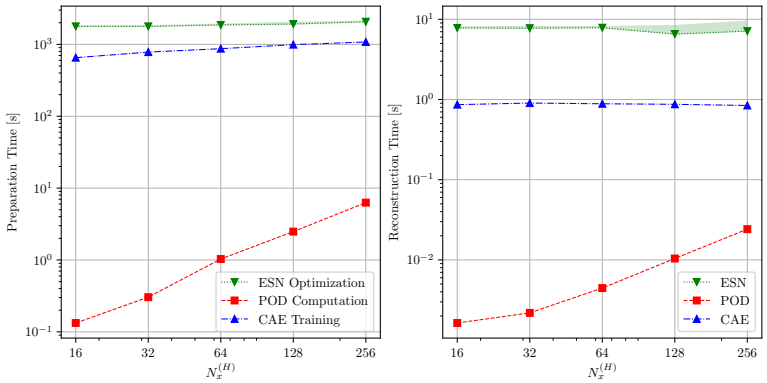


Figure 6.11: Preparation and reconstruction times of the benchmark POD, CAE, and ESN for the 1D Burgers' MMS injected residuals as functions of spatial refinement $N_x^{(H)}$. The preparation time corresponds to CAE training, ESN optimization, and POD eigenvalue computation. The CAE was trained on *Device 1*, while ESN and POD computations were performed on *Device 2*.

As expected, the CAE training time increases monotonically with spatial resolution, consistent with the growing input dimensionality and the corresponding increase in convolutional operations. For the ESN, the measured optimization times correspond to a single BO realization executed on a single CPU core. Due to the stochastic nature of reservoir initialization and BO sampling, the ESN optimization time exhibits a noticeable spread across realizations. This variability is inherent to the method and does not indicate instability; instead, it reflects the dependence of the optimization trajectory on random initialization and exploration within the BO process. Despite this, the overall scaling with $N_x^{(H)}$ remains of a constant order, confirming the computational efficiency of the ESN. In contrast, the POD exhibits the steepest slope among the three methods, indicating that at higher spatial refinements, its computational cost grows faster than that of both the CAE and ESN.

The reconstruction time, shown in the right panel, provides a consistent basis for comparison since it directly measures the cost of using the prepared surrogate models. The ESN and CAE demonstrate nearly constant reconstruction times across resolutions, confirming their scalability once trained. Conversely, the POD shows a consistently increasing reconstruction cost due to the increased size of the

matrix operations.

These results collectively emphasize that the primary goal of this chapter is the verification of the methodology rather than achieving perfect accuracy for a specific surrogate. The comparable performance of all models in reconstructing the mean residuals validates the soundness of the framework, while the difficulty in capturing standard deviations highlights the intrinsic challenge of modeling higher-order statistical features with reduced-order representations. Nevertheless, the ability of all three methods to consistently approximate the residual statistics across refinements demonstrates the robustness of the proposed approach.

6.4. Primal Solution Compression

To assess *Scenarios 2* and *3*, as introduced in Chapter 5, it is necessary to construct a surrogate representation of the fine primal solution. This surrogate serves as the basis for computing the corresponding surrogate adjoint solution, $\tilde{\psi}$. For consistency, the same compression methodologies developed for the injected residuals were adopted, with only minor modifications tailored to the primal solution. These adjustments are explicitly detailed in the respective sections.

6.4.1. Proper Orthogonal Decomposition

The results of the offline POD benchmark applied to the reconstructed fine primal solution are presented in Figure 6.12. In contrast to the injected residuals $R(u_h^H)$, the retained modal energy $\mathcal{E}_k$ converges almost instantaneously, with a single mode being sufficient to capture more than 99% of the total energy across all refinement levels. This reflects the inherent simplicity and high coherence of the manufactured solution, which was deliberately designed to contain smooth and spatially periodic features. As a result, the primal dynamics exhibit an extremely low-rank structure, enabling their full representation by only one dominant mode.

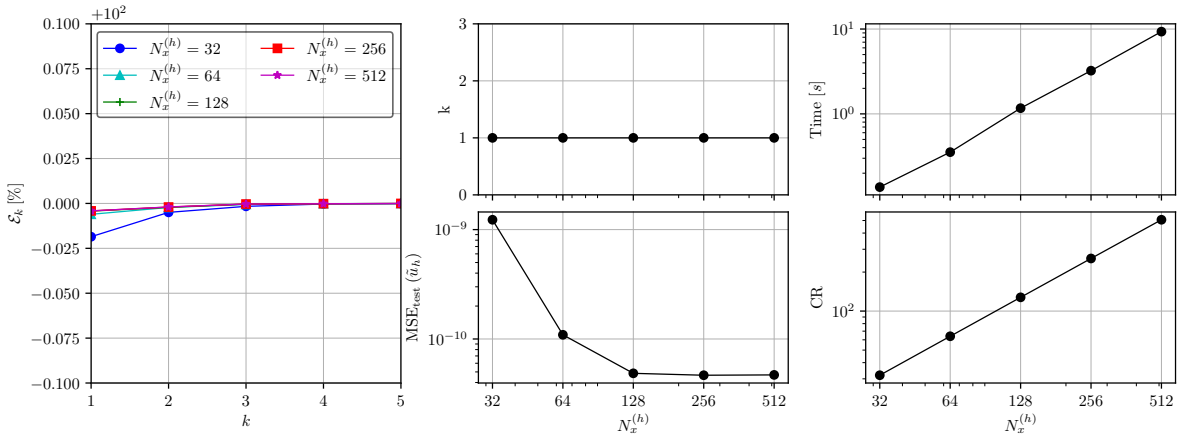


Figure 6.12: Offline POD benchmark results of reconstructed 1D MMS Burgers' fine primal solution for different spatial resolutions $N_x^{(H)}$. (Left) Retained modal energy $\mathcal{E}_k$ versus retained mode k (Right) Number of retained modes k, test MSE, computation time, and compression ratio for thresholds of 99%. (All x-axes are in log scale.)

The consequences of this rapid convergence are evident in the accompanying metrics. The number of retained modes k remains constant at unity regardless of spatial resolution, while the test MSE decreases systematically with refinement, reaching values below 10^{-10} for the largest grid sizes. At the same time, the compression ratio grows proportionally with refinement, highlighting the scalability of POD for this problem. The computation time increases mildly with resolution but remains negligible in comparison to the overall savings in dimensionality.

6.4.2. Convolutional Autoencoder

The CAE was also applied to the fine primal solution using the same training strategy and architectural framework outlined in Section 5.3.2. However, in contrast to the injected residuals, the PCA-based analysis indicated that the latent representation of the fine primal solution could be effectively reduced to a single dimension ($d_{\mathcal{Z}} = 1$) across all spatial resolutions, as reported in Table 6.4. This outcome

is a direct consequence of the manufactured solution, which generates smooth and highly coherent primal fields. The low-rank nature of the data implies that nearly all its variability is captured along one dominant direction, eliminating the need for additional latent variables.

Table 6.4: Latent space dimension $d_{\mathcal{Z}}$ determined for the CAE architecture based on PCA analysis of fine primal solution from the 1D MMS Burgers' problem.

$N_x^{(h)}$	32	64	128	256	256
$d_{\mathcal{Z}}$	1	1	1	1	1

Although the CAE architecture remained identical to that used for residuals $R(u_h^H)$ (see Figure 6.6), the reduction in $d_{\mathcal{Z}}$ directly decreased the number of trainable parameters, as summarized in Tables 6.5 and 6.6. The remaining architectures for higher resolutions were generated analogously and are presented in Appendix B.1.

The optimization settings, batch size, subsampling strategy, and adaptive learning rate schedule were kept identical to those of the residual case to ensure consistency. Nonetheless, training for the fine primal solution converged faster, reflecting the reduced complexity of the learning task.

Figure 6.13 reports the batch-averaged MSE for training and validation when learning the fine primal solution. In all resolutions $N_x^{(h)}$, the loss drops rapidly during the first ∼50–100 epochs and then decays more slowly under the adaptive learning rate schedule, stabilizing at values around 10^{-5}. The tight overlay of training and validation curves indicates no sign of overfitting. The short-lived oscillations visible in the early stage stem from the relatively large initial learning rate and batch stochasticity; they are progressively damped as the rate decays. Because the primal fields admit a 1D latent representation ($d_{\mathcal{Z}} = 1$), the model capacity is sufficient across all $N_x^{(h)}$, yielding similar training errors and convergence trends. Further reductions could be obtained by extending the constant learning rate phase or lowering the final learning rate, but this is unnecessary for the accuracy targeted in this work.

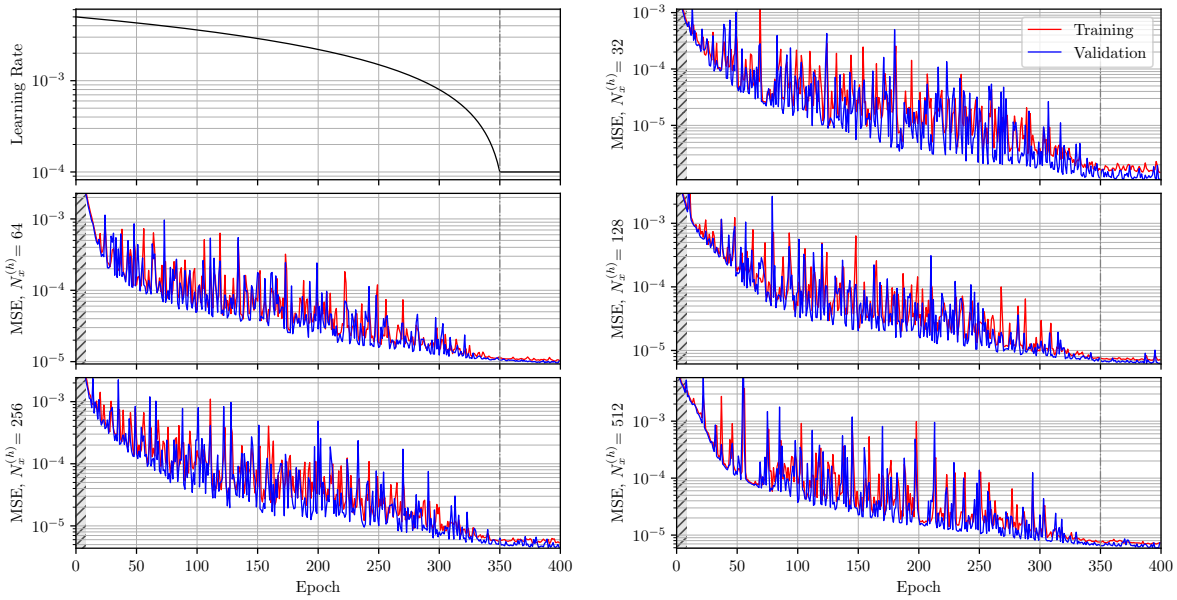


Figure 6.13: Training and validation batch-averaged MSE during CAE training of 1D MMS Burgers' fine primal solution for different spatial resolutions $N_x^{(h)}$. The top-left panel shows the adaptive learning rate schedule applied during the first 350 epochs, followed by a constant value. (The initial epochs are truncated in all MSE plots for visualization purposes.)

6.4.3. Echo State Network

The ESN was also applied to compress the fine primal solution of the 1D MMS Burgers' problem. The architecture again consisted of a single reservoir space, pseudorandomly constructed, with output weights determined via linear regression. The training dataset comprised 100,000 time steps corresponding to the full solution, while the validation set was generated from 40 non-overlapping intervals

Table 6.5: Layer-by-layer details of the encoder architecture applied to 1D MMS Burgers' fine primal solution for the case $N_x^{(h)} = 32$

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 32)$	$(4, 16)$	$(4, 2, 1)$	16
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E2	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E3	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E4	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E5	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E6	Flatten	$(4, 1)$	(4)		0
E7	Linear	(4)	(1)		5

Table 6.6: Layer-by-layer details of the decoder architecture applied to 1D MMS Burgers' fine primal solution for the case $N_x^{(h)} = 32$

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(1)	(4)		8
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(1, 32)$	$(3, 1, 1)$	12

of 2,500 samples each, following the same setup as in the residuals case.

The BO framework was employed to tune the hyperparameters. The search space was defined as $\rho \times \sigma_{\text{in}} \in [0.1, 1] \times [10^{-3}, 10^{2}]$ for the spectral radius and input scaling, respectively. The Tikhonov parameter was again selected from a discrete grid $\beta \in \{10^{-3}, 10^{-6}, 10^{-9}\}$. In contrast to the injected residuals, the search range for σ_{in} is narrower here. This reflects the statistics of the fine primal field, whose magnitude remains $\mathcal{O}(1)$ and is essentially constant across refinements in the MMS setup; consequently, the optimal input scaling does not need to span multiple orders of magnitude. The chosen range is sufficient to excite the reservoir without saturating it while preserving the balance with the internal dynamics controlled by ρ.

For consistency and because the task is not more demanding than the residual case, the reservoir size and leaking rate were kept identical to the previously selected values, i.e., $N_r = 64$ and $\alpha = 0.6$. This adheres to the earlier analysis, showing diminishing returns beyond $N_r \approx 64$ and an optimum near $\alpha \approx 0.6$ and avoids introducing additional degrees of freedom that would not materially affect performance in this context.

6.4.4. Performance Assessment

This section presents the performance assessment of the three surrogate models: benchmark POD, CAE, and ESN, applied to the reconstruction of the fine primal solution u_h across all refinement levels $N_x^{(h)}$. The quantitative performance comparison between the benchmark POD, CAE, and ESN for the fine primal solution is presented in Figure 6.14. The results reinforce the conclusions drawn from the reconstructed fields. Both POD and CAE deliver consistently low test MSE across all refinement levels, with values decreasing monotonically as resolution increases, except for the initial resolution of CAE. The CAE, in particular, achieves stable performance, demonstrating that even with a nonlinear encoder–decoder mapping, the compression task remains trivial for this simple MMS-driven solution. POD performs similarly well, although its accuracy plateaus slightly earlier, reflecting the restriction of its modal basis to linear structures.

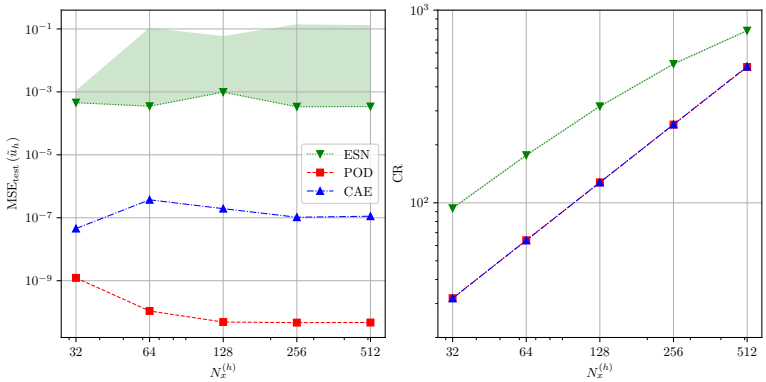


Figure 6.14: Performance comparison of POD, CAE, and ESN for the 1D Burgers' MMS fine primal solution, showing test MSE and compression ratio (CR) as functions of spatial refinement $N_x^{(h)}$.

In contrast, the ESN struggles to reproduce the fine solution reliably. Not only are its test MSE orders of magnitude larger than those of POD and CAE, but the variance in performance across optimization runs also increases with refinement, as evidenced by the widening shaded region. This indicates that the recurrent dynamics of the ESN, while useful in richer temporal problems, provide little benefit in this smooth MMS case and instead introduce instability.

The poor performance of the ESN can also be attributed to the use of the same BO setup that was originally designed for compressing the injected residuals $R(u_h^H)$. While this ensured methodological consistency, it likely limited the ESN's ability to adapt to the fundamentally different statistical properties of the fine primal solution.

From a compression perspective, all three models provide high compression ratios. Here, the ESN formally achieves the highest CR due to its fixed reservoir size, but this comes at the expense of reconstruction accuracy. POD and CAE strike a more favorable balance, offering both stable accuracy and

predictable scalability.

The preparation and reconstruction times of the surrogate models applied to the fine primal solution are summarized in Figure 6.15. The left panel shows the preparation time, while the right panel reports the reconstruction time for each model as a function of spatial refinement $N_x^{(h)}$.

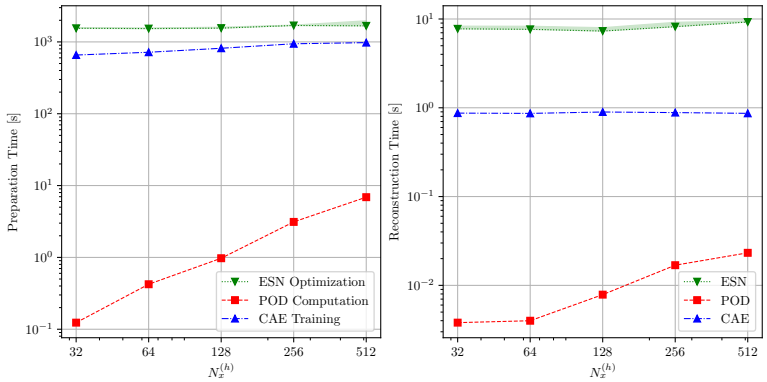


Figure 6.15: Preparation and reconstruction times of the benchmark POD, CAE, and ESN for the 1D Burgers' MMS fine primal solution as functions of spatial refinement $N_x^{(h)}$. The preparation time corresponds to CAE training, ESN optimization, and POD eigenvalue computation. The CAE was trained on *Device 1*, while ESN and POD computations were performed on *Device 2*.

As expected, the CAE training cost increases monotonically with spatial resolution, reflecting the larger input dimensionality and greater number of convolutional operations and trainable parameters. Although the trend is super-linear, the total training time remains below 10^3 s even for the finest mesh ($N_x^{(h)} = 512$), confirming the computational practicality of the approach. The ESN optimization time shows only a mild increase with spatial refinement, remaining close to $\mathcal{O}(10^3)$ s across all cases. The slight rise at higher resolutions can be attributed to the larger reservoir state space and the increasing cost of regression solves within the BO routine. Despite this, the ESN maintains stable scaling and overall computational efficiency for the problem sizes investigated. In contrast, POD exhibits the steepest increase in computational cost with refinement, primarily due to the full matrix operations involved in modal recomposition.

Regarding reconstruction, the POD computational time increases steeply, while the ESN reconstruction time shows a modest upward trend that reflects the cost of propagating reservoir states during reconstruction. The CAE lies between these two extremes, remaining nearly constant across all resolutions, making it the most efficient of the three once trained. Overall, these results confirm that, while POD becomes increasingly expensive with refinement, both CAE and ESN provide computationally scalable alternatives suitable for fine-grid applications.

Taken together, these results confirm that, while all methods achieve strong compression, only POD and CAE provide accurate and robust reconstructions of the fine primal solution. CAE further distinguishes itself by combining accuracy with consistent and efficient reconstruction times, whereas the ESN proves unreliable in the compression and reconstruction of the smooth fine primal.

6.5. Adjoint Solution

As discussed in Chapter 3, the second required dataset for evaluating the output error estimate is the adjoint solution. The adjoint problem was derived following the continuous adjoint formulation described in Section 3.1. Because of the non-linear convective term in the Burgers' equation, the derivation required a local linearization around the primal solution $u(x,t)$. To begin, consider a perturbation δu of the primal solution. The variation of the primal residual $R[u]$ with respect to this perturbation defines the linearized residual operator as

$$R'[u](\delta u) = \frac{\partial(\delta u)}{\partial t} + (\delta u)\frac{\partial u}{\partial x} + u\frac{\partial(\delta u)}{\partial x} - \frac{1}{Re}\frac{\partial^2(\delta u)}{\partial x^2}. \tag{6.6}$$

To derive the corresponding adjoint operator, this residual variation was introduced into the definition of the functional derivative. For a given functional $J[u]$, the variation induced by δu can be expressed

as

$$J'[u](\delta u) = \int_I \int_\Omega \psi \left(\frac{\partial(\delta u)}{\partial t} + (\delta u)\frac{\partial u}{\partial x} + u\frac{\partial(\delta u)}{\partial x} - \frac{1}{Re}\frac{\partial^2(\delta u)}{\partial x^2} \right) d\Omega dI, \tag{6.7}$$

where ψ is the adjoint variable. By transferring derivatives from the perturbation δu to the adjoint variable ψ through integration by parts and collecting boundary contributions, the adjoint operator is obtained as

$$J'[u](\delta u) = \int_I \int_\Omega \left(-\frac{\partial \psi}{\partial t} - u\frac{\partial \psi}{\partial x} - \frac{1}{Re}\frac{\partial^2 \psi}{\partial x^2} \right) (\delta u)\, d\Omega dI. \tag{6.8}$$

Since this expression must hold for arbitrary perturbations δu, the adjoint PDE is recovered as Eq. (6.9). The corresponding boundary and terminal conditions follow from the integration by parts and from the definition of the functional.

$$\begin{cases} -\dfrac{\partial \psi}{\partial t} - u\dfrac{\partial \psi}{\partial x} - \dfrac{1}{Re}\dfrac{\partial^2 \psi}{\partial x^2} = \sin(\pi x), & (x,t) \in \Omega \times I, \\ \psi(x,t) = 0, & (x,t) \in \partial\Omega \times I, \\ \psi(x,T) = 0, & x \in \Omega, \end{cases} \tag{6.9}$$

Here, x and t denote the spatial and temporal coordinates, respectively; ψ represents the adjoint solution; $\mathrm{Re} = 100$ is the prescribed Reynolds number (controlled by kinematic viscosity ν in OpenFOAM, identical to the primal solver); and u is the computed primal solution.

Figure 6.16 shows the computed adjoint solution ψ for $N_x^{(H)} = 32$. The adjoint exhibits a smooth, periodic behavior in time, consistent with the sinusoidal source term associated with the chosen QoI. Spatially, the solution is concentrated in the lower half of the domain, with maximum amplitudes near $x \approx 0.2$–0.4, gradually decaying toward the upper boundary where homogeneous Dirichlet conditions are imposed. The backward-in-time computation introduces wave-like structures that persist throughout the domain, with an amplitude on the order of $\mathcal{O}(1)$. This distribution indicates that the adjoint is most sensitive to perturbations in the central-lower region of the domain, reflecting the weighting introduced by $\sin(\pi x)$ in the QoI definition.

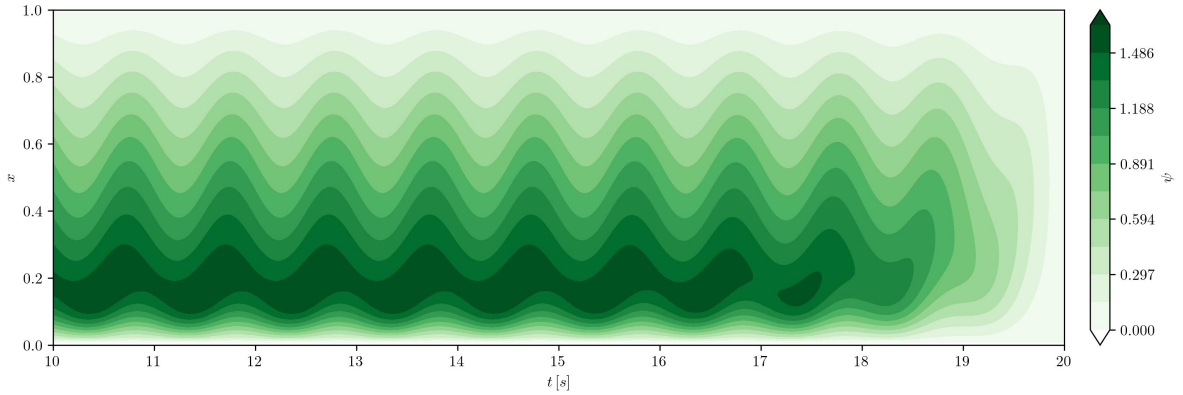


Figure 6.16: Adjoint solution ψ of 1D MMS Burgers' for spatial resolution of $N_x^{(H)} = 32$.

Using the reconstructed fine primal solution $\tilde{u}_h$, the adjoint solution $\tilde{\psi}$ was computed using the same adjoint solver developed in OpenFOAM. It is noticeable that the reconstructed fine primal solutions over 28 realizations were first averaged before computing a single adjoint solution. This strategy was adopted to mitigate stochastic variability in the surrogate reconstructions of the ESN, where fluctuations across realizations were pronounced.

The accuracy of the adjoint solutions computed using the surrogate fine primal fields was further quantified by computing the MSE with respect to the reference adjoint solution, defined as

$$\mathrm{MSE}(\tilde{\psi}) = \frac{1}{N_t N_x^{(h)}} \sum_{i=1}^{N_t} \sum_{j=1}^{N_x^{(h)}} \left(\tilde{\psi}(x_j, t_i) - \psi(x_j, t_i) \right)^2, \tag{6.10}$$

where N_t is the number of temporal samples, $N_x^{(h)}$ is the number of spatial elements in the fine space, $\tilde{\psi}$ is the adjoint solution computed from the surrogate fine primal solution, and ψ is the reference adjoint solution.

The results are presented in Figure 6.17. Both POD and CAE achieve small MSE values across all refinement levels, confirming their ability to reconstruct the adjoint field with nearly perfect accuracy. In particular, POD reaches errors close to machine precision, while CAE exhibits slightly higher but still negligible discrepancies. By contrast, the ESN reconstructions show MSE values that are orders of magnitude larger, with errors that remain essentially insensitive to refinement. This behavior reflects the limitations already observed in the primal reconstructions, wherein inaccuracies in the ESN's predicted fine primal field directly propagate into the adjoint solution.

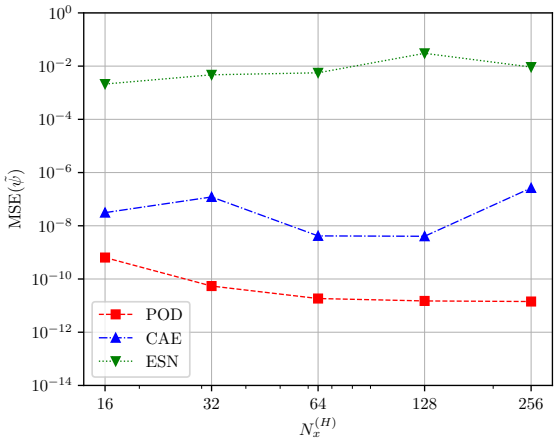


Figure 6.17: Test MSE of the computed 1D MMS Burgers' adjoint solution $\tilde{\psi}$ using the reconstructed fine space solution $\tilde{u}_h$ for all refinements $N_x^{(H)}$ using POD, CAE, and ESN.

6.6. Adjoint-Based Error Localization

The spatial distribution of the time-averaged adjoint-based error indicator $\bar{\epsilon}$ is presented in Figure 6.18 for all refinement levels. At coarse resolutions, the error indicators exhibit relatively large magnitudes and irregular distributions across the domain, reflecting the inability of the mesh to fully capture the fine-scale features of the MMS solution. As the refinement increases, the overall error levels decrease by several orders of magnitude, and the distributions become progressively smoother. This trend demonstrates the consistency of the adjoint-based error estimator, which systematically identifies reduced spatial discretization error as the resolution improves. Moreover, the localization of error near regions of steep gradients becomes increasingly evident with refinement, showing that the estimator is sensitive to the dominant features of the solution.

Having introduced the different surrogate models, the adjoint-based error indicator $\bar{\epsilon}$ averaged over time per cell is now compared across *Scenarios 1–3*. The baseline FVM without any surrogate model is plotted in gray in Figures 6.19–6.21 for reference.

In *Scenario 1*, where only the injected residual field is compressed and reconstructed, the error distribution closely follows the baseline FVM across all refinements, with only minor deviations visible at coarse resolutions. This indicates that the residual surrogates preserve the dominant error-contributing structures with sufficient fidelity.

Scenario 2, in which the fine primal solution is compressed and subsequently used for adjoint computation, introduces a slightly larger discrepancy compared to Scenario 1. While POD and CAE reconstructions remain nearly indistinguishable from the baseline, the ESN shows noticeable divergence in both the amplitude and the spatial pattern of the error indicator, particularly as the refinement increases. This highlights the sensitivity of the adjoint solution to inaccuracies in the reconstructed primal field.

Scenario 3, where both the residuals and fine primal solution are reconstructed before computing the adjoint, compounds the surrogate-induced deviations. While POD and CAE still deliver robust per-

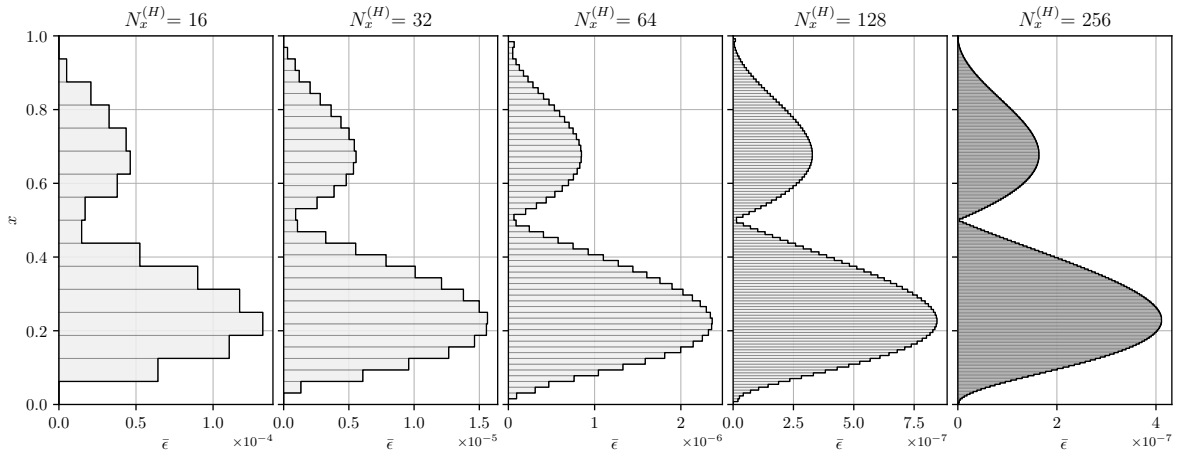


Figure 6.18: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D MMS Burgers' problem across all refinement levels $N_x^{(H)}$.

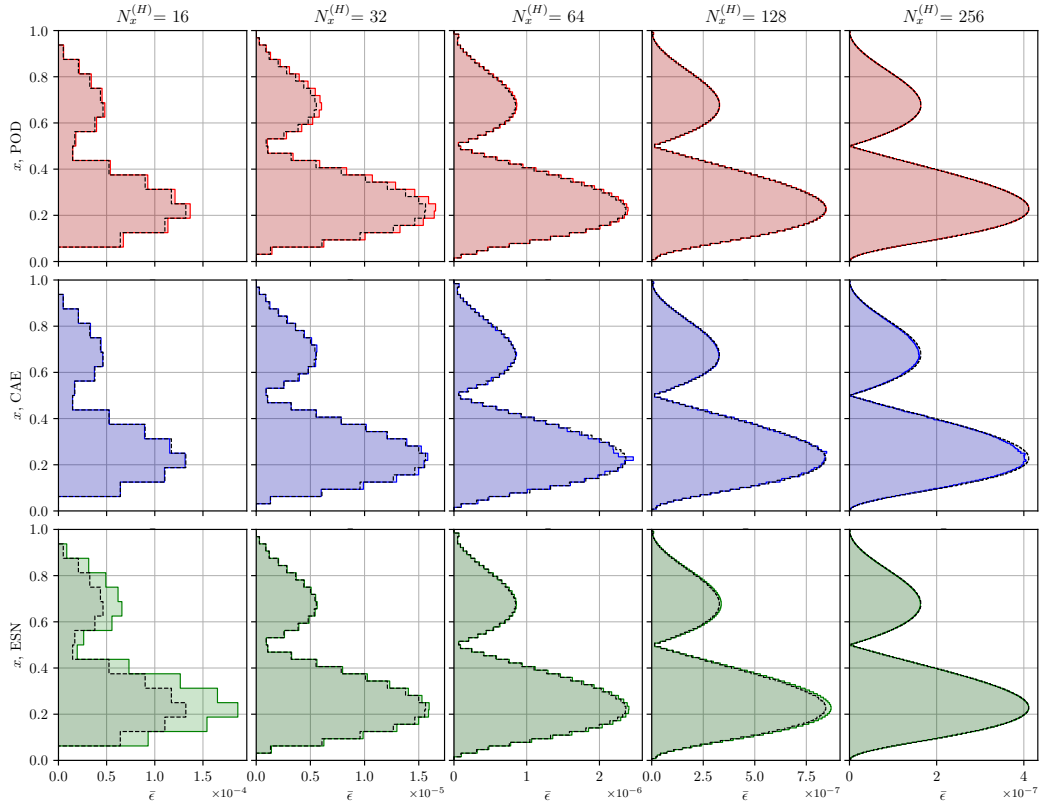


Figure 6.19: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D MMS Burgers' problem at all refinement levels $N_x^{(H)}$, computed using surrogate models POD (top), CAE (middle), and ESN (bottom) under *Scenario 1*.

formance, the ESN exhibits degradation relative to the baseline. The widening mismatch across refinements underscores that the recurrent dynamics of the ESN are poorly suited to this setting, amplifying error propagation when both surrogate stages are involved.

Overall, the comparison across scenarios suggests that the largest contributor to discrepancies in the adjoint-based error indicator arises from the use of surrogate fine primal solutions rather than the injected residual compression alone. POD and CAE prove sufficiently reliable across all scenarios, whereas ESN performance deteriorates when integrated into adjoint computations.

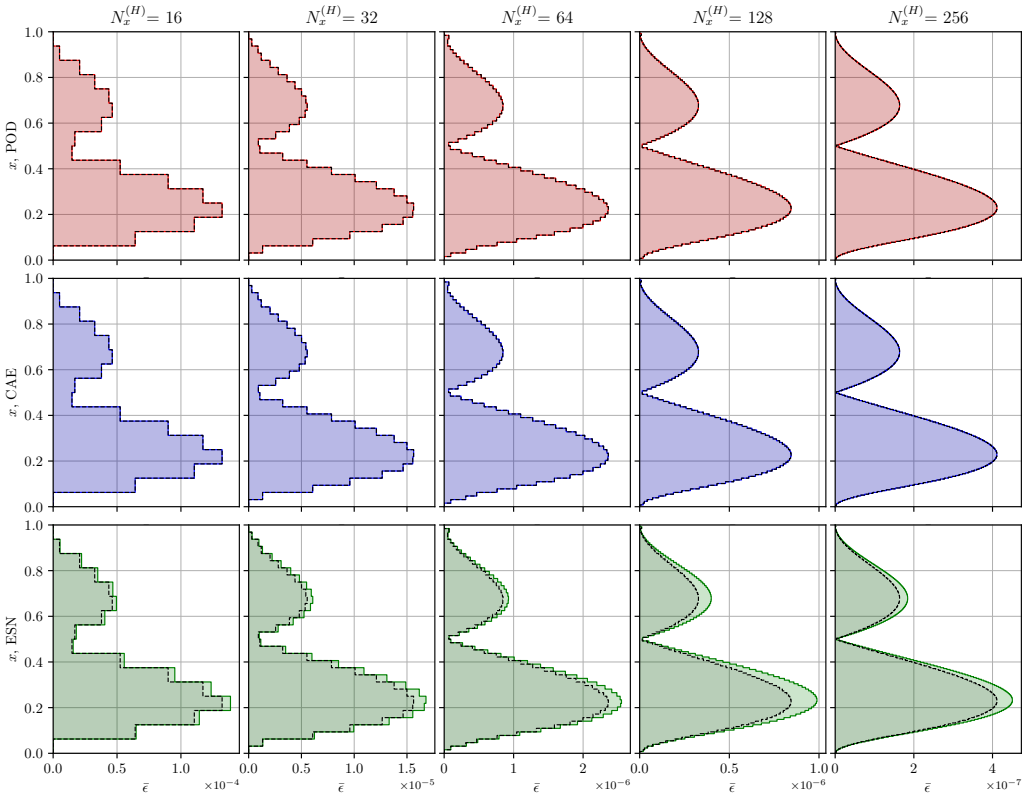


Figure 6.20: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D MMS Burgers' problem at all refinement levels $N_x^{(H)}$, computed using surrogate models POD (top), CAE (middle), and ESN (bottom) under *Scenario 2*.

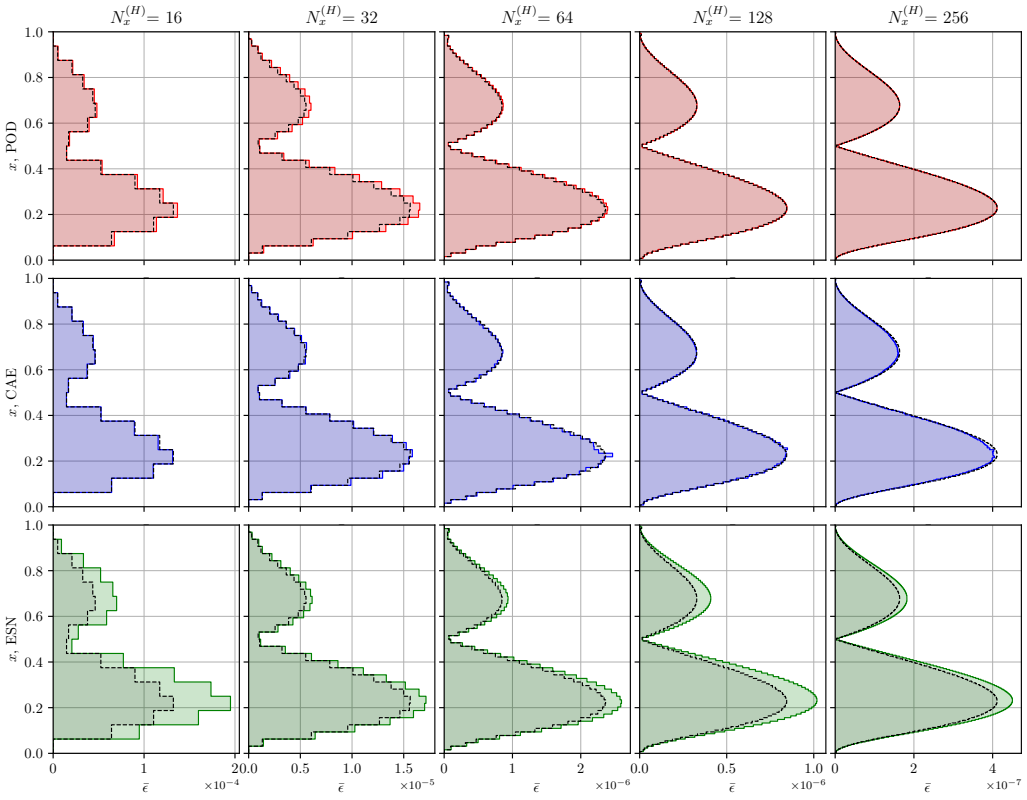


Figure 6.21: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D MMS Burgers' problem at all refinement levels $N_x^{(H)}$, computed using surrogate models POD (top), CAE (middle), and ESN (bottom) under *Scenario 3*.

6.7. Adjoint-Based Error Estimation

To evaluate the accuracy of adjoint-based error estimation across different scenarios, the error in the QoI J and its associated error estimates were examined. The time-averaged QoI is denoted by $\bar{J}$. The true error was defined as

$$\Delta\bar{J} = \bar{J}_{\text{MMS}} - \bar{J}_H, \tag{6.11}$$

where $\bar{J}_{\text{MMS}}$ denotes the exact time-averaged QoI obtained from the manufactured solution, and $\bar{J}_H$ denotes the numerical QoI computed on a coarse computational grid with $N_x^{(H)}$ cells. In the baseline framework, the adjoint-based error estimate was denoted by $\delta\bar{J}_{\text{FVM}}$, and the corrected QoI was obtained as

$$\bar{J}_{\text{FVM}} = \bar{J}_H + \delta\bar{J}_{\text{FVM}}. \tag{6.12}$$

In *Scenarios 1–3*, where surrogate models were introduced, the adjoint-based error estimates were denoted by $\delta\bar{J}_{\text{POD}}$, $\delta\bar{J}_{\text{CAE}}$, and $\delta\bar{J}_{\text{ESN}}$, with corresponding corrected QoIs $\bar{J}_{\text{POD}}$, $\bar{J}_{\text{CAE}}$, and $\bar{J}_{\text{ESN}}$. The quality of these error estimates was quantified by the error in the error estimate metric,

$$e(\delta\bar{J}) = \left|\delta\bar{J} - \Delta\bar{J}\right|, \tag{6.13}$$

which measures the deviation of the estimated error from the true error.

The results across *Scenarios 1–3* are presented in Figure 6.22. In the left panels, the time-averaged QoIs $\bar{J}$ are shown, including $\bar{J}_H$, $\bar{J}_{\text{MMS}}$, and the corrected values from each surrogate model. The middle panels illustrate the absolute adjoint-based error estimates $|\delta\bar{J}|$, while the right panels compare the error in the error estimate $e(\delta\bar{J})$.

Across all scenarios, the baseline FVM error estimates ($\delta\bar{J}_{\text{FVM}}$) were found to closely follow the true error ($\Delta\bar{J}$), confirming the consistency of the adjoint-based approach. The reconstructions obtained with POD and CAE preserved this accuracy, with error estimates nearly indistinguishable from the baseline. In contrast, larger deviations were observed for ESN, particularly at finer refinements, reflecting the instability already identified in its primal reconstructions and adjoint computation.

When scenarios were compared, it was observed that *Scenario 1* (injected residual compression only) introduced a negligible impact on the error estimate, indicating that the residual surrogate was not the dominant source of inaccuracy. In *Scenario 2* (fine primal compression only), slightly larger deviations were observed; however, POD and CAE still retained high fidelity. In *Scenario 3*, where both residuals and the primal solution were reconstructed, the accumulation of surrogate effects amplified ESN's discrepancies, while POD and CAE remained robust.

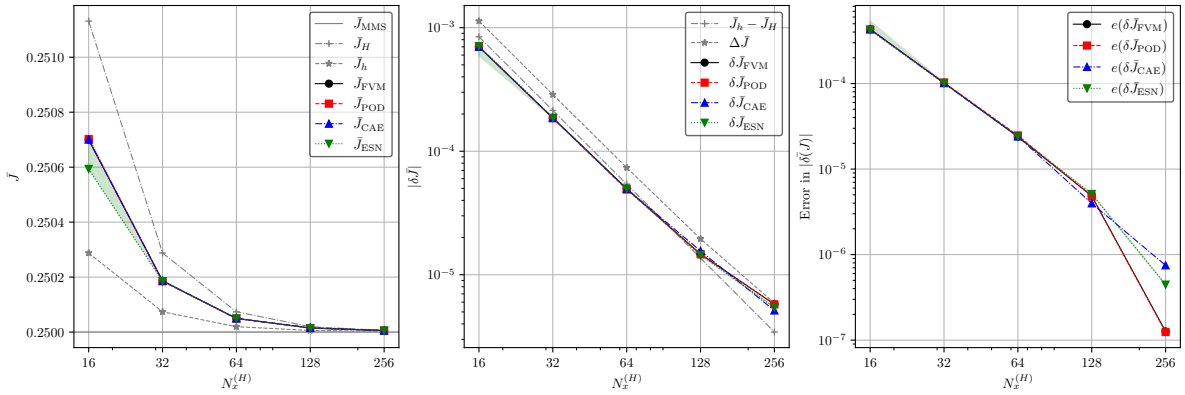
It can therefore be concluded that POD and CAE can be integrated into the adjoint-based estimation pipeline without degradation of accuracy, whereas ESN requires further tuning to achieve comparable reliability.

6.8. Summary

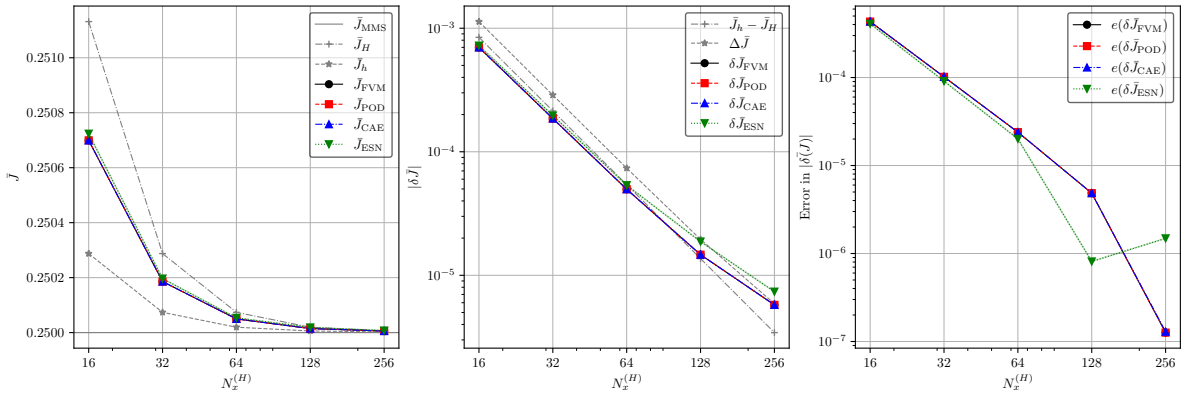
The MMS case verified the correctness and consistency of the proposed framework under analytically defined and smooth conditions. The study began with the formulation of a 1D unsteady viscous Burgers' equation with a prescribed manufactured solution and its corresponding source term f_{MMS}. A grid convergence study confirmed the second-order accuracy of the implemented solver, with both the QoI and RMSE decreasing as $\mathcal{O}(h^2)$, verifying the numerical discretization and temporal scheme.

Injected residuals were then computed by projecting the coarse solution onto a refined grid and substituting it into the governing PDE. Their spatial distribution exhibited alternating symmetric patterns consistent with the sinusoidal source, while their magnitude decreased by orders across refinements, confirming convergence and consistency of the projection and injection procedures.

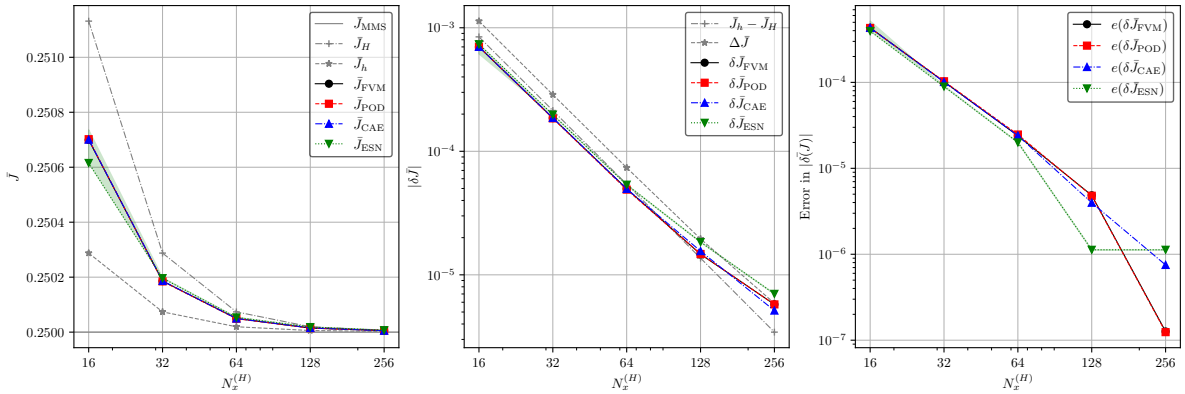
The first surrogate modeling stage addressed the compression of the injected residuals. The POD benchmark captured more than 95% of the energy with fewer than three modes across all resolutions, and the 65% threshold was adopted as a balanced benchmark. The CAE achieved stable convergence with compact latent dimensions ($d_{\mathcal{Z}} = 2$ for all cases), effectively reproducing residual structures while maintaining low reconstruction error. The ESN performance was optimized through BO of reservoir parameters, demonstrating consistent validation errors and stable compression ratios. All models achieved



(**a**) *Scenario 1*



(**b**) *Scenario 2*



(**c**) *Scenario 3*

Figure 6.22: Comparison of adjoint-based error estimation for the 1D MMS Burgers' problem across *Scenarios 1–3*. Shown are (left) the time-averaged QoI $\bar{J}$, including the exact value $\bar{J}_{\mathrm{MMS}}$, the numerical solution $\bar{J}_H$, and corrected values obtained using FVM, POD, CAE, and ESN; (middle) the adjoint-based error estimates $|\delta\bar{J}|$; and (right) the error in the error estimate $e(\delta\bar{J})$.

comparable test MSE levels, verifying that the framework accurately compresses and reconstructs residual fields without numerical artifacts.

In the second surrogate stage, the fine primal solution was compressed. The POD achieved nearly full energy recovery with a single dominant mode due to the low-rank nature of the MMS field. The CAE and ESN reproduced smooth profiles with minimal deviations; the ESN required a higher optimization time due to the stochastic nature of its reservoir search but maintained consistent accuracy across refinements.

Adjoint-based error localization and estimation were then assessed across the defined Scenarios. The adjoint field exhibited symmetric structures. *Scenario 1* (residual compression only) showed negligible

deviation from the baseline; *Scenario 2* (primal compression) produced minor differences, with POD and CAE preserving accuracy, while *Scenario 3* (combined compression) revealed amplified errors for ESN but stable performance for POD and CAE. The corrected QoIs $\bar{J}_{\text{POD}}$ and $\bar{J}_{\text{CAE}}$ closely matched the exact manufactured value, and their error estimates $\delta\bar{J}$ followed the true error trend, confirming the consistency of adjoint-based estimation under surrogate integration.

Overall, the MMS case verified each component of the proposed methodology: solver accuracy, residual convergence, surrogate compression of both residual and primal fields, and the preservation of adjoint-based error estimation and localization fidelity. These results establish the numerical soundness of the framework and its readiness for testing under nonlinear and data-driven DNS-forced conditions.

7

Results: DNS-driven Unsteady 1D Viscous Burgers' Equation

This chapter presents the application of the proposed framework to a DNS-forced unsteady 1D viscous Burgers' equation. Unlike the controlled MMS studied in Chapter 6, the DNS case introduces a physically realistic and dynamically rich dataset, characterized by nonlinear interactions and complex temporal evolution. This configuration enables a more comprehensive evaluation of the framework's robustness when applied to data with intrinsic turbulence-like variability. The analysis follows the same structure as the MMS verification: the DNS primal solution is first described, followed by the compression and reconstruction of the injected residuals and fine primal solution using the benchmark POD, CAE, and ESN approaches. Subsequently, the corresponding adjoint solutions are computed, and their integration into adjoint-based error localization and estimation is assessed across the defined surrogate scenarios. The chapter concludes with a summary of the principal findings and implications for unsteady error estimation under realistic flow dynamics.

7.1. Problem Formulation

A DNS dataset of a TCF at a friction Reynolds number $Re_\tau = 180$ was employed to provide the forcing data for the unsteady 1D viscous Burgers' equation. The simulation produced the wall-normal velocity component, which served as the input signal. In wall-bounded turbulent flows, the friction Reynolds number is a key control parameter [133] and is defined as:

$$\mathrm{Re}_\tau = \frac{u_\tau \, \delta}{\nu}, \tag{7.1}$$

where u_τ denotes the friction velocity, δ the channel half-height, and ν the kinematic viscosity of the fluid.

The TCF simulation was carried out using a spectral-element DNS solver, from which the required data were extracted. Specifically, the DNS dataset was sampled at a fixed streamwise location x and spanwise location z, and the resulting wall-normal velocity component was used to drive the 1D Burgers' equation. In this formulation, the Burgers' equation is equivalent to the wall-normal momentum equation of the NS system at the chosen locations. The extracted dataset was subsequently down-sampled to a uniform temporal resolution of $\Delta t = 1 \times 10^{-5}$, with a spatial discretization of $N_y = 4096$ elements.

For the 1D Burgers' problem, the computational domain was defined as $\Omega = [-\delta, \delta] \times I = [0, T]$ with $\delta = 1.0$ and $T = 5.0$. To ensure consistency with the DNS wall-normal velocity, homogeneous Dirichlet BCs were applied. In addition, the IC was set to match the DNS velocity profile at $t = 0$. Under these

assumptions, the following primal PDE was obtained:

$$\begin{cases} \dfrac{\partial v}{\partial t} + v\dfrac{\partial v}{\partial y} - \dfrac{1}{Re}\dfrac{\partial^2 v}{\partial y^2} = f_{\mathrm{DNS}}, & (y,t) \in \Omega \times I, \\ v(y,t) = 0, & (y,t) \in \partial\Omega \times I, \\ v(y,0) = v_{\mathrm{DNS}}(y,0), & y \in \Omega. \end{cases} \tag{7.2}$$

Here, y and t denote the spatial wall-normal and temporal coordinates, respectively; v represents the primal wall-normal velocity; and f_{DNS} is the forcing source term extracted from the DNS dataset using Eq. (1.2). The friction Reynolds number is set to $\mathrm{Re}_\tau = 180$, defined as Eq. (7.1). Since both u_τ and δ are fixed to unity in this nondimensionalized setup, the corresponding kinematic viscosity ν is obtained directly from the prescribed Re_τ and is used in the NS equations solved in OpenFOAM.

$$f_{\mathrm{DNS}} = \left[-u\frac{\partial v}{\partial x} - w\frac{\partial v}{\partial z} - \frac{1}{\rho}\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2}\right) \right]_{\mathrm{DNS}}, \tag{7.3}$$

where u and w denote the streamwise and spanwise velocity components, respectively, ρ is the fluid density, and p the pressure field. It is noticeable that, to evaluate the required derivatives, central finite difference approximations of second-order accuracy were employed. At the domain boundaries, however, first-order finite difference schemes were adopted to maintain numerical stability.

Within the AMR framework, an h-adaptation strategy with a refinement factor of 2 was considered. Accordingly, the spatial resolutions of the coarse grid were set to $N_y^{(H)} \in \{64, 128, 256, 512\}$, with a particular focus on $N_y^{(H)} = 128$ for detailed visualizations. Coarser configurations were found to produce instabilities in the adjoint solver, as the insufficiently resolved primal field led to divergence during the backward computation.

Since the DNS dataset must be represented on these coarser resolutions, it was necessary to downsample the original data. To achieve this, a top-hat filtering operation was applied, ensuring consistency with potential applications of the same framework in the context of LES. The top-hat filter smooths the DNS signal by locally averaging over a finite support, thereby removing small-scale fluctuations that cannot be represented on the coarser grids.

Mathematically, the filtered signal $\bar{v}(y)$ at a coarse grid point y is given by:

$$\bar{v}(y) = \frac{1}{\Delta}\int_{y-\Delta/2}^{y+\Delta/2} v(\eta)\, d\eta, \tag{7.4}$$

where $v(\eta)$ is the DNS velocity field, and Δ is the filter width. In the case of downsampling to a resolution $N_y^{(H)}$, the filter width is chosen as

$$\Delta = \frac{2\delta}{N_y^{(H)}},$$

corresponding to the local grid spacing on the coarse mesh. This procedure ensures that the downsampled data are both numerically stable and physically consistent, while retaining the large-scale structures necessary for meaningful error estimation within the AMR process.

A time-averaged QoI, denoted by $\bar{J}$, was defined for this analysis as

$$\bar{J} = \frac{1}{T}\int_I \int_\Omega e^{\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)} v^4(y,t)\, d\Omega\, dI, \tag{7.5}$$

where $v(y,t)$ is the wall-normal velocity component, Ω denotes the spatial domain, and I the time interval. The Gaussian weighting function was centered at $\mu = 0$ with a standard deviation of $\sigma = 0.1$ to emphasize the channel centerline.

This QoI was designed to approximate the non-normalized kurtosis near the channel centerline [134]. By normalizing $\bar{J}$ with the fourth power of the standard deviation, computed both from the DNS dataset and the Gaussian reference distribution, the statistical kurtosis can be obtained as

$$\bar{J}_{\mathrm{norm}} = \frac{\bar{J}}{\sigma^4}, \tag{7.6}$$

where σ denotes the standard deviation of the wall-normal velocity fluctuations. Kurtosis, the fourth standardized statistical moment, characterizes the heaviness of the distribution tails and the presence of outliers. For a Gaussian distribution, the kurtosis takes the value 3, which is typically used as the baseline for comparison [134].

From the DNS dataset, the following values were obtained:

$$\bar{J}_{\text{DNS}} = 0.2340, \qquad \bar{J}_{\text{DNS,norm}} = 3.596.$$

The normalized value indicates that the probability distribution of the DNS wall-normal velocity exhibits heavier tails than a Gaussian distribution near the channel centerline. However, in practical applications the exact normalization constant may not be accessible, as it requires prior knowledge of the full dataset. For this reason, the non-normalized QoI $\bar{J}_{\text{DNS}}$ is adopted as the true reference value for the subsequent error estimation framework.

7.2. Primal Solution

To assess the accuracy of the primal solutions against the DNS reference, both a functional-based and a state-based error analysis were performed. The functional error was quantified through the deviation of the time-averaged QoI,

$$\Delta\bar{J}_H = \bar{J}_{\text{DNS}} - \bar{J}_H,$$

while the state error was measured using the RMSE of the wall-normal velocity field, defined as

$$\text{RMSE}(v_H) = \sqrt{\frac{1}{N_t N_y}\sum_{j=1}^{N_t}\sum_{i=1}^{N_y}\Big(v_H(y_i,t_j) - v_{\text{DNS}}(y_i,t_j)\Big)^2}, \tag{7.7}$$

where N_t and N_y are the number of time steps and spatial elements, and v_{DNS} is the down-sampled DNS solution with a top-hat filter.

Figure 7.1 presents the results of the grid convergence study based on the selected QoI and the RMSE of the wall-normal velocity field. The results reveal a consistent convergence of both the QoI versus spatial refinements of $N_y^{(H)}$ and the RMSE versus $\Delta y^{(H)}$. The computed QoI values approach the DNS reference monotonically, with the error $\Delta\bar{J}_H$ decaying at a rate close to $O(N^{-2})$. The discrepancy at coarser resolutions highlights the inability of under-resolved grids to capture the heavy tails of the probability distribution associated with the wall-normal velocity. Similarly, the RMSE exhibits a systematic decrease as the mesh spacing is reduced, following an asymptotic slope proportional to $O(h^2)$. This confirms that the primary source of error at coarse resolutions originates from spatial discretization.

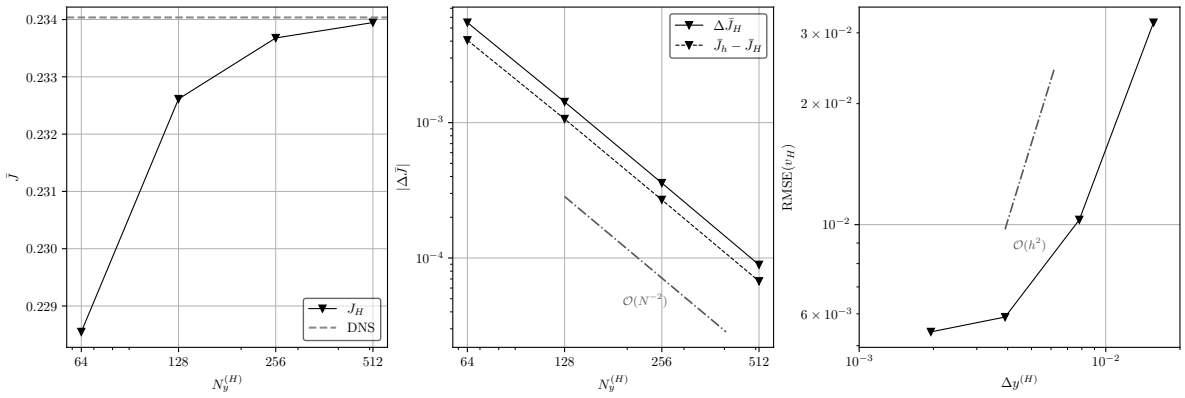


Figure 7.1: Grid convergence study of the 1D DNS-forced Burgers' equation. (Left) Convergence of the QoI $\bar{J}_H$ towards the DNS value. (Center) The absolute output error $|\Delta\bar{J}_H|$ against the spatial resolution $N_y^{(H)}$, confirming a $O(N^{-2})$ convergence rate. (Right) RMSE of the wall-normal velocity field versus DNS down-sampled solution as a function of the grid spacing $\Delta y^{(H)}$, confirming the $O(h^2)$ behavior before plateauing due to temporal discretization effects.(All x-axes are shown in log scale.)

At finer resolutions, however, the reduction in error slows down, and a plateau emerges in both the QoI and RMSE trends. This behavior indicates that once the discretization error has been sufficiently

reduced, other error sources become dominant. These include the interpolation and differentiation of the DNS dataset, as well as the finite temporal resolution used in the computations. Such secondary contributions limit the benefits of further mesh refinement and mark the onset of the asymptotic regime.

Overall, the convergence study establishes two key findings: first, the numerical discretization of the primal problem behaves consistently with theoretical predictions; and second, beyond a certain resolution, discretization errors are no longer the limiting factor.

The instantaneous behavior of the wall-normal velocity field for the resolution $N_y^{(H)} = 128$ is illustrated in Figure 7.2, where elongated streak-like structures appear across the temporal domain. These alternating positive and negative fluctuations highlight the intermittent nature of turbulence and confirm the persistence of energetic wall-normal motions throughout the simulation.

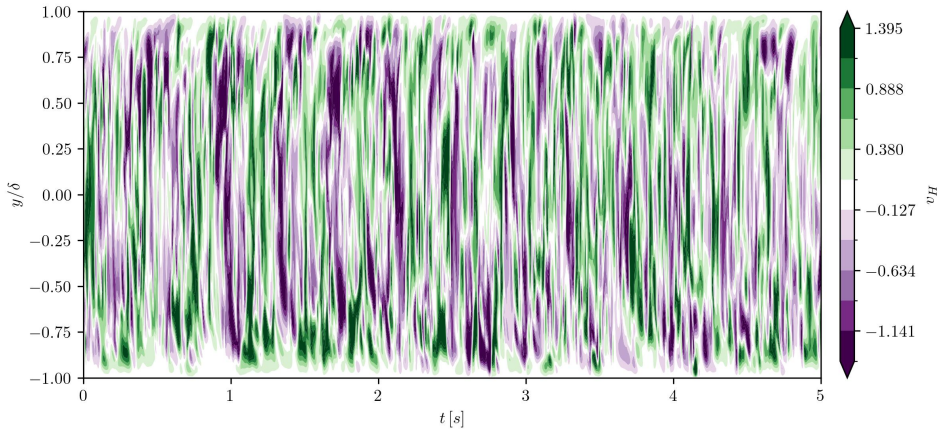


Figure 7.2: Instantaneous wall-normal primal coarse velocity of 1D DNS-forced Burgers' equation v_H for $N_y^{(H)} = 128$

To analyze these features statistically, the Reynolds decomposition was applied to the wall-normal velocity, $v = \bar{v} + v'$, where $\bar{v}$ denotes the temporal mean and v' the fluctuating component. According to the Reynolds-averaged continuity equation, and under the imposed homogeneous Dirichlet boundary conditions, the mean wall-normal velocity should vanish, $\bar{v} = 0$. This expectation was confirmed in practice, as the computed profiles yielded $\bar{v}^+ \approx 0$ across the channel domain, in agreement with the DNS dataset, where a $+$ superscript represents the viscous scales normalization [135].

The second-order statistic of interest is the Reynolds stress $\overline{v'v'}$, which quantifies the intensity of wall-normal velocity fluctuations and contributes directly to the momentum transport in the channel [135]. As shown in Figure 7.3, the Reynolds stress profile obtained from the solver closely matches the DNS reference for all tested grid resolutions. A maximum is observed at $y/\delta \approx 0.25$, as expected for the imposed Re_τ [136]. The agreement between the implemented solver and the DNS dataset demonstrates the ability of the reduced 1D formulation to reproduce the essential statistical properties of the flow.

Furthermore, the comparison across resolutions indicates that coarse grids (e.g. $N_y^{(H)} = 64$) are unable to fully capture the fine-scale structure of the fluctuations, resulting in deviations in both the mean and the Reynolds stress. As the resolution increases, however, the profiles collapse onto the DNS reference, supporting the conclusions of the grid convergence study. Extending the temporal domain would be expected to further reduce residual deviations in the mean and to produce an even more symmetric distribution of the Reynolds stress about the channel centerline.

The turbulent energy distribution of the wall-normal velocity fluctuations was further examined by computing the spectral energy density $E_{vv}(k)$ through a Fourier decomposition of the fluctuating component $v'(y)$. The discrete signal was transformed into the frequency domain using a Fast Fourier Transform (FFT), and the spectral density was defined as

$$E_{vv}(k) = \tfrac{1}{2}\,\hat{v}(k)\hat{v}^*(k), \tag{7.8}$$

where $\hat{v}(k)$ denotes the Fourier coefficients and the superscript $*$ their complex conjugate. A time average of $E_{vv}(k)$ was then computed to obtain the wall-based spectral energy spectrum.

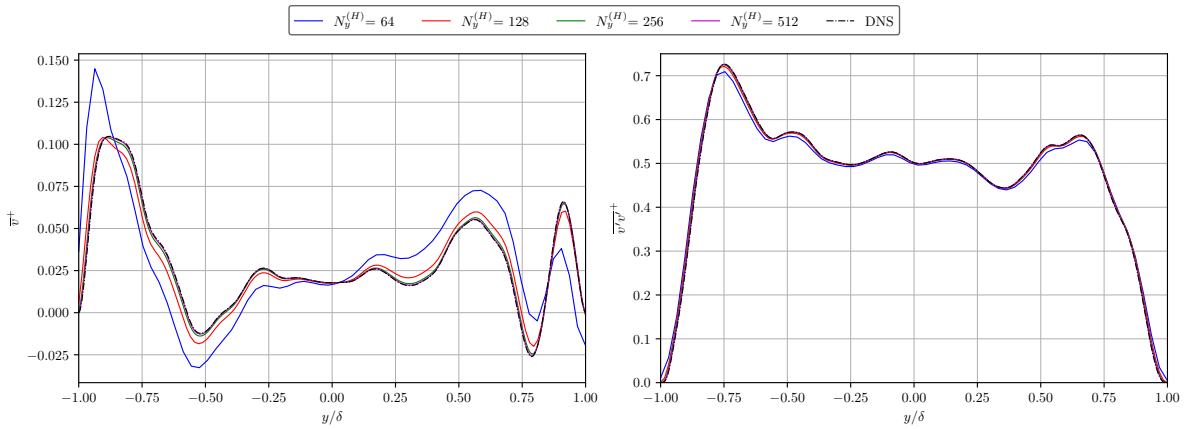


Figure 7.3: Comparison of wall-normal statistics between the primal solver and DNS. (Left) Mean wall-normal velocity $\bar{v}^+$. (Right) Reynolds stress $\overline{v'v'}^+$.

Figure 7.4 shows the resulting spectra for different spatial resolutions compared with the DNS reference. As expected, larger scales (low wavenumbers) contain most of the turbulent kinetic energy, while progressively smaller scales (high wavenumbers) are associated with reduced energy content. The well-resolved DNS solution reproduces the entire energy cascade, whereas the coarser grids fail to capture the high-wavenumber range. Increasing the resolution improves the overlap with the DNS and extends the inertial range of resolved scales.

The slope of the inertial subrange in the Burgers' equation is theoretically $E(k) \sim k^{-2}$ [137], distinct from the $k^{-5/3}$ law associated with the NS equations. At higher resolutions, the most energetic turbulent scales are adequately captured, though the inertial range remains limited due to the relatively low Reynolds number ($Re_\tau = 180$) of the forcing dataset. Nevertheless, the comparison across resolutions clearly illustrates the effect of spatial discretization; while low wavenumber energy content is captured even at coarse resolutions, only fine grids are capable of approximating the DNS behavior at intermediate and high wavenumbers.

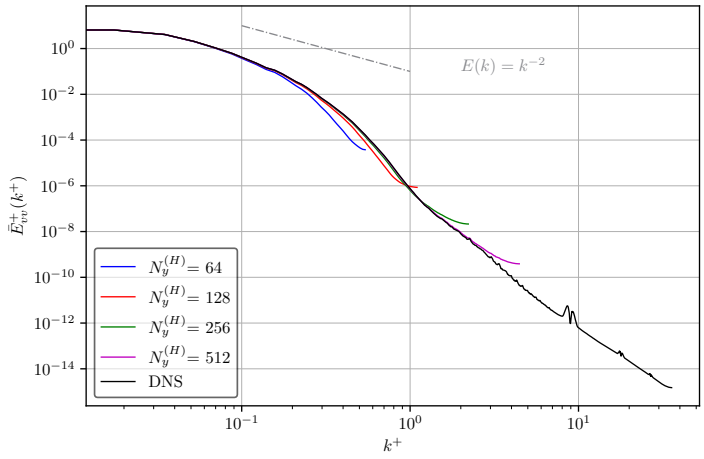


Figure 7.4: Time-averaged spectral energy density $\bar{E}_{vv}^+(k^+)$ of the wall-normal velocity fluctuations computed for 1D DNS-forced Burgers' equation for different spatial resolutions, compared with the DNS reference. The theoretical k^{-2} slope of the Burgers' equation [137] is indicated for reference.

7.3. Injected Residual Compression

As discussed, one of the essential datasets for evaluating the output error estimate is the set of injected residuals. In the 1D DNS-driven case, these residuals were obtained by first projecting the coarse-grid solution v_H onto the corresponding refined grid, yielding the fine-grid representation v_h^H. To achieve this, a linear interpolation scheme was applied. For consistency, the forcing term extracted from the

DNS dataset was projected onto the fine grid using the same procedure.

The projected field v_h^H was then substituted into the primal Burgers' equation, Eq. 7.2, to compute the injected residuals, denoted as $R(v_h^H)$. The governing expression can be written as

$$R(v_h^H) = \frac{\partial v_h^H}{\partial t} + v_h^H \frac{\partial v_h^H}{\partial y} - \frac{1}{Re}\frac{\partial^2 v_h^H}{\partial y^2} - f_{\text{DNS}}|_h^H \,, \tag{7.9}$$

where f_{DNS} represents the forcing term derived directly from the DNS dataset. This formulation ensures that the injected residuals quantify the discrepancy between the interpolated coarse-grid solution and the DNS-driven fine-grid dynamics, thereby providing the necessary input for the error estimation framework.

The injected residuals for the case $N_y^{(H)} = 128$ are shown in Figure 7.5. These residuals quantify the mismatch between the projected coarse-grid solution and the DNS-driven fine-grid dynamics, thereby providing a localized measure of discretization and projection error.

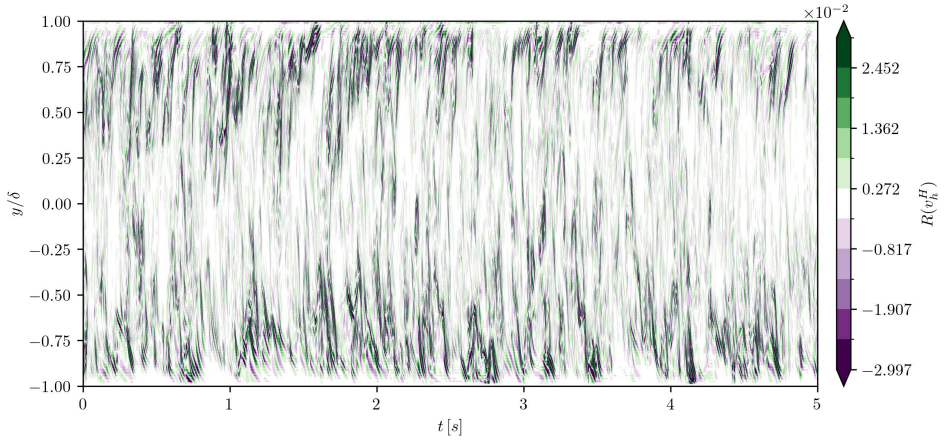


Figure 7.5: Instantaneous injected residuals $R(v_h^H)$ of 1D DNS-forced Burgers' equation for $N_y^{(H)} = 128$.

In contrast to the smoother structures observed in the wall-normal velocity field v_H (Figure 7.2), the residual field exhibits significantly higher spatial and temporal complexity. The contour plot reveals a dense pattern of alternating positive and negative values, which are strongly modulated by the turbulent fluctuations of the DNS forcing. This added complexity arises from the involvement of nonlinear convective terms, viscous contributions, and the DNS forcing term in the definition of the residual. As a result, the residual field highlights not only the turbulent dynamics but also the sensitivity of the numerical discretization to small-scale variations.

It is further observed that the magnitude of the injected residuals is relatively smaller in the middle of the channel compared to the near-wall regions. This distribution reflects the stronger velocity gradients and enhanced turbulent activity close to the walls, where discretization and projection errors accumulate more prominently. The reduced residual levels in the channel center suggest that the numerical approximation is more accurate, while the wall vicinity remains the critical zone driving the overall error behavior.

Figure 7.6 reports the temporal statistics of the injected residuals for different spatial resolutions. The black line indicates the mean residual profile, while the gray shaded region corresponds to the standard deviation over time.

Across all resolutions, the mean residuals remain close to zero, which confirms that the discretization error does not introduce a systematic bias into the solution. Instead, the dominant information is carried by the fluctuations. The standard deviation reveals the spatial structure of these fluctuations, which are consistently stronger near the walls, where velocity gradients and turbulent activity are largest, and weaker in the channel center, where the flow fluctuations are less intermittent. As the grid is refined, the magnitude of the standard deviation decreases by more than an order of magnitude, from $\mathcal{O}(10^{-1})$ at $N_y^{(H)} = 64$ to $\mathcal{O}(10^{-4})$ at $N_y^{(H)} = 512$, demonstrating the convergence of the injected residuals with

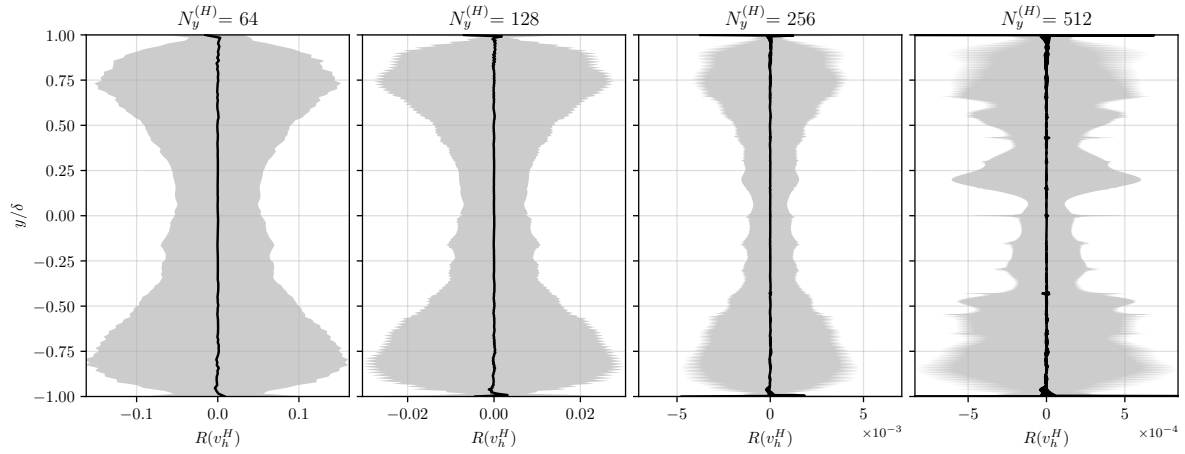


Figure 7.6: Temporal mean (black line) and standard deviation (gray shaded area) of the injected residuals $R(v_h^H)$ of 1D DNS-forced Burgers' equation across the channel height for different grid resolutions $N_y^{(H)} \in \{64, 128, 256, 512\}$.

resolution. In addition, the refinement reveals sharper peaks in the standard deviation profiles near the wall regions, indicating the emergence of higher-order spatial modes in the residual field that become resolved as smaller-scale gradients are captured on finer grids.

7.3.1. Proper Orthogonal Decomposition

The results of the residual compression using an offline POD are summarized in Figure 7.7. The left panel shows the cumulative energy content $\mathcal{E}_k$ as a function of the number of retained modes, while the remaining plots report the required number of modes k to satisfy the corresponding threshold, the POD computation time, the reconstruction MSE, and CR for different refinement levels.

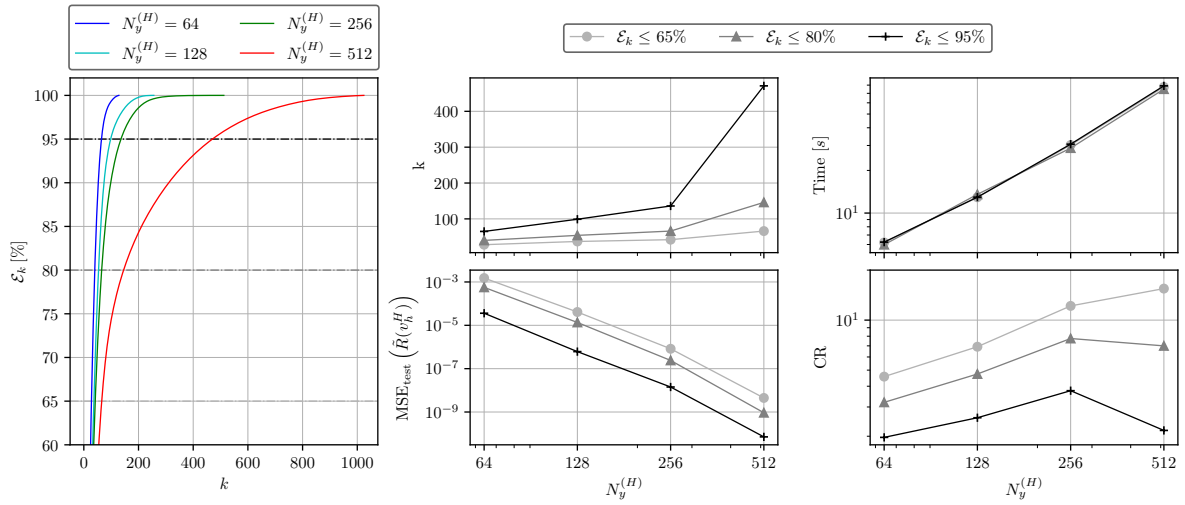


Figure 7.7: Offline POD benchmark results of reconstructed injected residuals of 1D DNS-forced Burgers' equation for different spatial resolutions. (Left) Cumulative energy content $\mathcal{E}_k$ as a function of retained modes k, with thresholds at 65%, 80%, and 95%. (Right) Corresponding required number of modes k, POD computation time, reconstruction test MSE, and compression ratio (CR).

Three energy thresholds, $\mathcal{E}_k \leq 65\%$, 80%, and 95%, were tested. As expected, increasing the threshold improves reconstruction accuracy (lower MSE) but requires significantly more retained modes and a higher computational cost. At the coarsest resolutions ($N_y^{(H)} = 64$ and 128), relatively few modes are sufficient to reach the chosen thresholds, whereas the finer grids ($N_y^{(H)} = 256$ and 512) demand a much larger number of modes to achieve the same level of accuracy. This behavior reflects the growing complexity of the residual field with increasing resolution, as already observed in Figure 7.6.

For the remainder of this work, a threshold of $\mathcal{E}_k \leq 80\%$ is selected as the benchmark for assessing

alternative surrogate modeling strategies. This choice provides a compromise between accuracy and efficiency, as it ensures that the dominant energetic structures of the residuals are retained while avoiding the excessive number of modes and computational cost associated with the 95% case. Although the 65% threshold yields higher compression ratios, its reconstruction error is too large to be reliable for error estimation purposes.

It is important to note that, compared to the MMS case discussed in Chapter 6, the DNS-driven residuals are considerably more challenging to compress. For the MMS, only one or two modes were sufficient to accurately reconstruct the injected residuals, owing to the simplicity of the manufactured forcing. Here, however, the turbulent residuals require a substantially larger modal basis to achieve even moderate accuracy, underscoring the higher spatio-temporal complexity of the DNS problem.

7.3.2. Convolutional Autoencoder

The first proposed compression method for the 1D DNS-forced injected residuals was the CAE. As in the MMS case, the latent space dimension $d_{\mathcal{Z}}$ was initially estimated using PCA, ensuring that the CAE architecture is constructed to capture the dominant variance of the data without introducing redundant latent variables. The selected values of $d_{\mathcal{Z}}$ for the different spatial resolutions are summarized in Table 7.1.

Table 7.1: Latent space dimension $d_{\mathcal{Z}}$ determined for the CAE architecture based on PCA analysis of the 1D DNS-forced Burgers' injected residuals.

$N_y^{(H)}$	64	128	256	512
$d_{\mathcal{Z}}$	57	74	90	150

In contrast to the MMS case, where only two latent variables were sufficient across all refinement levels, the DNS-driven residuals demand significantly higher latent space dimensions. Specifically, $d_{\mathcal{Z}}$ increases from 57 at $N_y^{(H)} = 64$ to 150 at $N_y^{(H)} = 512$, reflecting the growing complexity of the residual statistics with grid refinement. This strong dependence on resolution highlights the richer and more intermittent structure of DNS-forced residuals compared to the highly regular patterns obtained under manufactured forcing.

The need for larger latent dimensions can be attributed to the combined effect of nonlinear convective dynamics, turbulent fluctuations, and near-wall intermittency present in the DNS case. As a result, the CAE is required to learn a more expressive latent representation that scales with the number of spatial elements. These findings confirm that, unlike in the MMS case, the DNS-forced residuals cannot be effectively compressed into a low-rank representation and instead require a larger latent space to preserve the key statistical features across resolutions.

Following the determination of the latent space dimension $d_{\mathcal{Z}}$ through PCA, a dedicated CAE architecture was generated for each spatial resolution of the DNS-forced injected residuals using Algorithm 3. In this case, the configuration parameters were chosen as $c_{\text{in}} = 1$, $c_{\text{base}} = 4$, $c_{\text{max}} = N_y^{(H)}/2$, and kernel size $k = 4$, accounting for the inherently higher complexity of the DNS dataset relative to the MMS case, together with the additional complexity introduced by increasing grid refinement. An illustrative example of the resulting CAE architecture is presented in Figure 7.8 for the case $N_y^{(H)} = 64$, while the detailed encoder and decoder specifications are provided in Tables 7.2 and 7.3, respectively. As before, (k, s, p) denote the kernel size, stride, and padding of each convolutional layer. The remaining architectures for higher resolutions were generated analogously and are presented in Appendix B.2.

The encoder is composed of seven convolutional layers, each followed by BN and LReLU activation. These layers progressively reduce the spatial dimension from the initial input size of $(1, 128)$ to a compressed representation of $(32, 1)$, while increasing the feature depth to 32 channels. The output is then flattened and projected onto the latent space of size $d_{\mathcal{Z}} = 57$ through a fully connected layer. Compared to the MMS case, this architecture requires a deeper encoder to accommodate the larger latent dimension and the richer variability of the DNS residuals.

The decoder mirrors the encoder in a symmetric fashion. Starting from the latent vector, a linear layer maps back to $(32, 1)$, which is then sequentially upsampled and processed through convolutional blocks

with BN and LReLU activation. This gradual reconstruction restores the spatial resolution of $(1, 128)$, producing the reconstructed residual field $\tilde{R}(v_h^H)$. The increased number of upsampling–convolution stages compared to the MMS design reflects the necessity of capturing finer-scale turbulent structures inherent to the DNS case.

Table 7.2: Layer-by-layer details of the encoder architecture applied to 1D DNS-forced Burgers' injected residuals for the case $N_y^{(H)} = 64$

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	16
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E2	Conv. 1D	$(4, 64)$	$(8, 32)$	$(4, 2, 1)$	128
	BN	$(8, 32)$	$(8, 32)$		16
	LReLU	$(8, 32)$	$(8, 32)$		0
E3	Conv. 1D	$(8, 32)$	$(16, 16)$	$(4, 2, 1)$	512
	BN	$(16, 16)$	$(16, 16)$		32
	LReLU	$(16, 16)$	$(16, 16)$		0
E4	Conv. 1D	$(16, 16)$	$(32, 8)$	$(4, 2, 1)$	2048
	BN	$(32, 8)$	$(32, 8)$		64
	LReLU	$(32, 8)$	$(32, 8)$		0
E5	Conv. 1D	$(32, 8)$	$(32, 4)$	$(4, 2, 1)$	4096
	BN	$(32, 4)$	$(32, 4)$		64
	LReLU	$(32, 4)$	$(32, 4)$		0
E6	Conv. 1D	$(32, 4)$	$(32, 2)$	$(4, 2, 1)$	4096
	BN	$(32, 2)$	$(32, 2)$		64
	LReLU	$(32, 2)$	$(32, 2)$		0
E7	Conv. 1D	$(32, 2)$	$(32, 1)$	$(4, 2, 1)$	4096
	BN	$(32, 1)$	$(32, 1)$		64
	LReLU	$(32, 1)$	$(32, 1)$		0
E8	Flatten	$(32, 1)$	(32)		0
E9	Linear	(32)	(57)		1881

Regarding the optimization procedure, the DNS-forced injected residuals required a modified training setup compared to the MMS case due to their significantly larger size and higher intrinsic complexity. A batch size of 1000 samples was employed, striking a balance between stable gradient updates and computational feasibility given the dataset's scale. Unlike the MMS case, no temporal subsampling was applied ($\Delta t_{\text{sub}} = 0$), as retaining the full resolution of the DNS residuals was necessary to preserve their rich spatio-temporal dynamics, resulting in a time series of length 500,000.

Training was carried out over 500 epochs. An adaptive learning rate strategy was applied during the first 450 epochs, following the formulation in Eq. (5.8), with an initial learning rate of $\gamma_0 = 10^{-3}$ and a decay factor of $\gamma_r = 10^{-2}$. After this adaptive phase, the learning rate was fixed at its final value for the last 50 epochs to stabilize convergence and reduce oscillations near the minimum. The hyperparameter choices were determined empirically through exploratory trials, with particular attention to maintaining stable convergence at higher resolutions.

Table 7.3: Layer-by-layer details of the decoder architecture applied to 1D DNS-forced Burgers' injected residuals for the case $N_y^{(H)} = 64$

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(57)	(32)		1856
D2	Upsample	$(32, 1)$	$(32, 2)$		0
	Conv. 1D	$(32, 2)$	$(32, 2)$	$(3, 1, 1)$	3072
	BN	$(32, 2)$	$(32, 2)$		64
	LReLU	$(32, 2)$	$(32, 2)$		0
D3	Upsample	$(32, 2)$	$(32, 4)$		0
	Conv. 1D	$(32, 4)$	$(32, 4)$	$(3, 1, 1)$	3072
	BN	$(32, 4)$	$(32, 4)$		64
	LReLU	$(32, 4)$	$(32, 4)$		0
D4	Upsample	$(32, 4)$	$(32, 8)$		0
	Conv. 1D	$(32, 8)$	$(32, 8)$	$(3, 1, 1)$	3072
	BN	$(32, 8)$	$(32, 8)$		64
	LReLU	$(32, 8)$	$(32, 8)$		0
D5	Upsample	$(32, 8)$	$(32, 16)$		0
	Conv. 1D	$(32, 16)$	$(16, 16)$	$(3, 1, 1)$	1536
	BN	$(16, 16)$	$(16, 16)$		32
	LReLU	$(16, 16)$	$(16, 16)$		0
D6	Upsample	$(16, 16)$	$(16, 32)$		0
	Conv. 1D	$(16, 32)$	$(8, 32)$	$(3, 1, 1)$	384
	BN	$(8, 32)$	$(8, 32)$		16
	LReLU	$(8, 32)$	$(8, 32)$		0
D7	Upsample	$(8, 32)$	$(8, 64)$		0
	Conv. 1D	$(8, 64)$	$(4, 64)$	$(3, 1, 1)$	96
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	(128)	$(1, 128)$	$(3, 1, 1)$	12

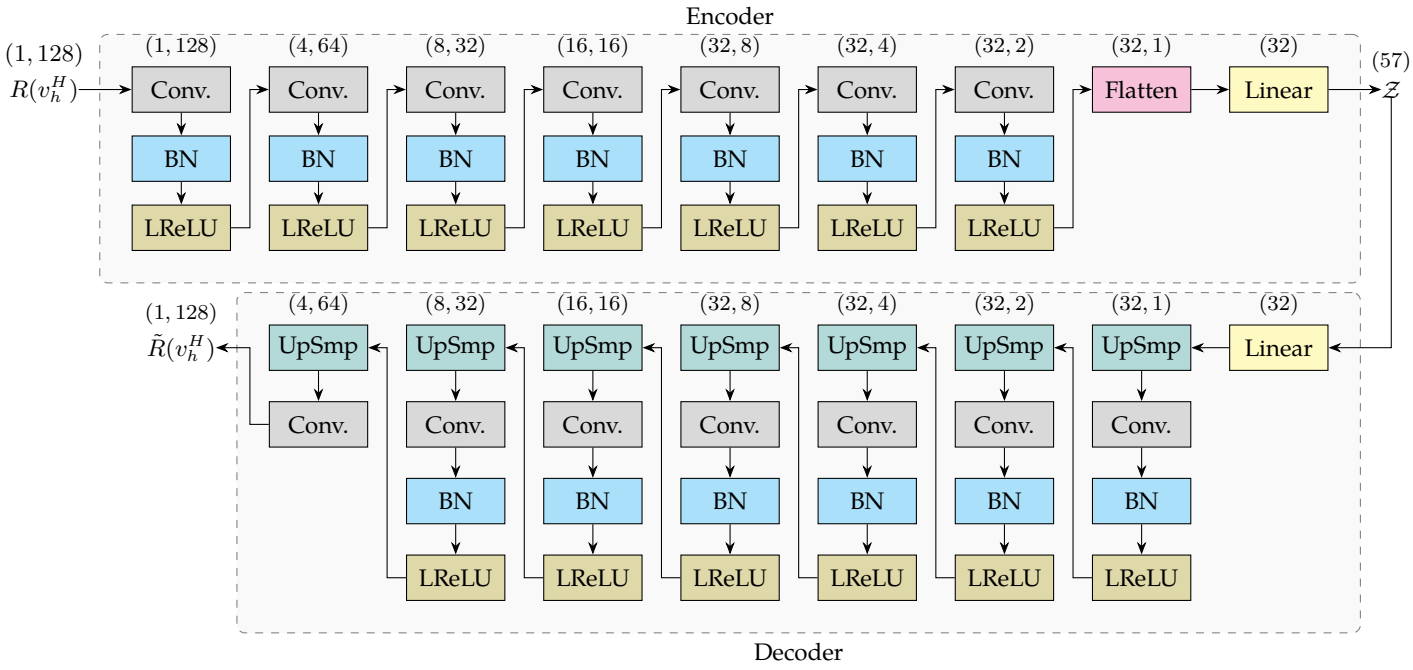


Figure 7.8: Illustration of the CAE architecture generated for the injected residuals of the 1D DNS-forced Burgers' on $N_y^{(H)} = 64$.

Figure 7.9 presents the evolution of the batch-averaged MSE for training and validation across different spatial resolutions $N_y^{(H)}$, together with the applied learning rate schedule. For all resolutions, the MSE exhibits a steep initial decrease, confirming that the CAE rapidly captures the dominant structures of the residuals. As training progresses, both both training and validation curves continue to decrease smoothly, with convergence typically reached around epochs 400–500. At the finer resolutions ($N_y^{(H)} = 256$ and 512), the MSE curves show higher variance during the early epochs, reflecting the added difficulty of compressing and reconstructing the more intricate DNS residual fields. Nevertheless, the training remains stable, and both training and validation losses eventually align, indicating no signs of overfitting.

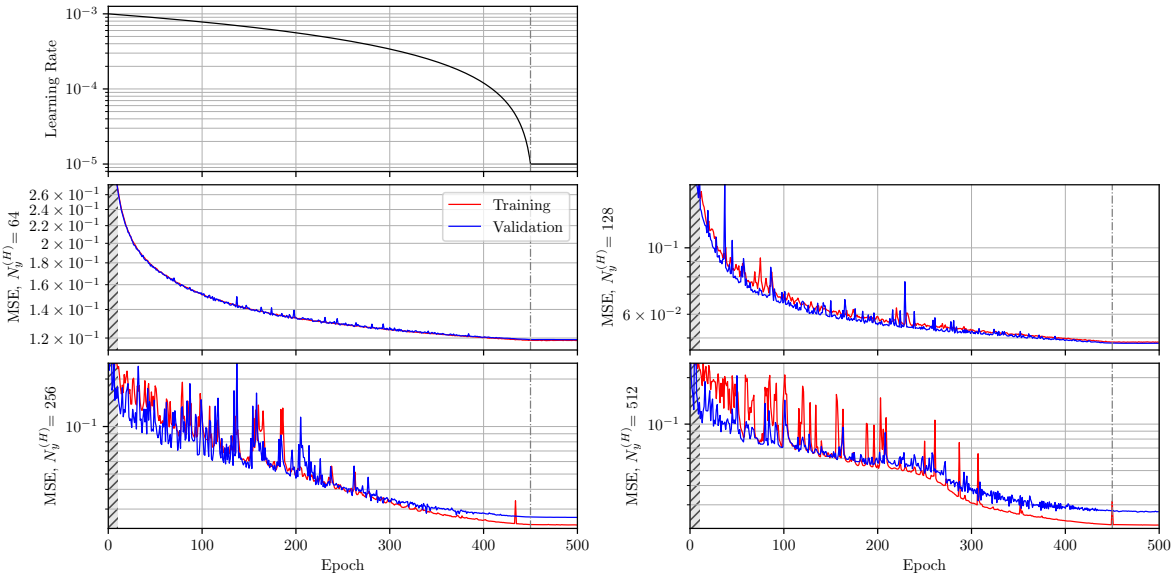


Figure 7.9: Training and validation batch-averaged MSE during CAE training of 1D DNS-forced Burgers' injected residuals for different spatial resolutions $N_y^{(H)}$. The top-left panel shows the adaptive learning rate schedule applied during the first 450 epochs, followed by a constant value. (The initial epochs are truncated in all MSE plots for visualization purposes.)

7.3.3. Echo State Network

The second proposed compression method was the ESN, constructed as a single pseudorandomly initialized reservoir. Training was carried out by solving a linear regression problem to determine the optimal mapping between the reservoir states and the output layer. For the DNS case, the dataset comprised the full temporal history of 500,000 steps. The validation set was assembled from 199 overlapping intervals of 5,000 samples each, providing a statistically representative evaluation across the full time horizon.

The hyperparameter search space explored by BO was defined as $\rho \times \sigma_{\text{in}} \in [0.1,\, 1] \times [10^{-5},\, 10^{-1}]$, with ρ denoting the spectral radius and σ_{in} the input scaling. This range represents a notable shift from the MMS configuration; the significantly smaller magnitudes of the DNS-forced residuals necessitated much lower values of σ_{in} in order to prevent overdriving the reservoir dynamics. By restricting σ_{in} to the interval $[10^{-5},\, 10^{-1}]$, the input signals were consistently rescaled to a regime where the nonlinear reservoir units could operate effectively across all refinement levels.

The spectral radius range $\rho \in [0.1,\, 1]$ again provided coverage of both the contractive regime ($\rho < 1$) for stability and the near-critical regime ($\rho \approx 1$) for enhanced memory capacity, thus ensuring that the reservoir could accommodate both short- and long-term correlations present in the DNS residuals. For the linear regression stage, the Tikhonov regularization parameter was selected from the reduced candidate set $\beta \in \{10^{-6},\, 10^{-9}\}$, which was sufficient to stabilize the solution while mitigating overfitting risks.

As in the MMS case, the number of neurons N_r and the leaking rate α were not optimized within the BO loop but were instead determined empirically in subsequent analyses. This separation ensured that the BO search remained focused on the key sensitivity drivers, namely ρ, σ_{in}, and β, while allowing for

additional flexibility in tuning reservoir capacity and memory retention.

Bayesian Optimization Setup

As introduced in Section 5.3.3, the first step in setting up the BO procedure for the ESN consisted of determining the appropriate number of initial grid search points N_{GS}. For the DNS-forced case with $N_y^{(H)} = 256$, $N_r = 128$, and $\alpha = 0.6$, the impact of N_{GS} was systematically assessed. Figure 7.10 (left) presents the distribution of validation MSE across multiple runs. Increasing N_{GS} from 4 to 25 led to a consistent reduction in the median error, accompanied by a contraction of the inter-percentile spread. This indicates that larger N_{GS} values improve both the accuracy and robustness of the BO procedure by providing a more reliable initialization of the surrogate model. Beyond $N_{GS} \approx 25$, however, the validation error plateaus, and additional grid evaluations yield negligible improvements.

The effect of N_{GS} on the optimization cost is shown in Figure 7.10 (right). The optimization time scales approximately as $O(N_{GS}^{0.74})$. This scaling emphasizes the computational trade-off, while a small N_{GS} results in higher variability and potentially suboptimal minima; excessively large values increase cost without tangible accuracy gains. Based on this balance, $N_{GS} = 25$ was again identified as a practical compromise, ensuring reliable convergence while keeping the computational overhead manageable.

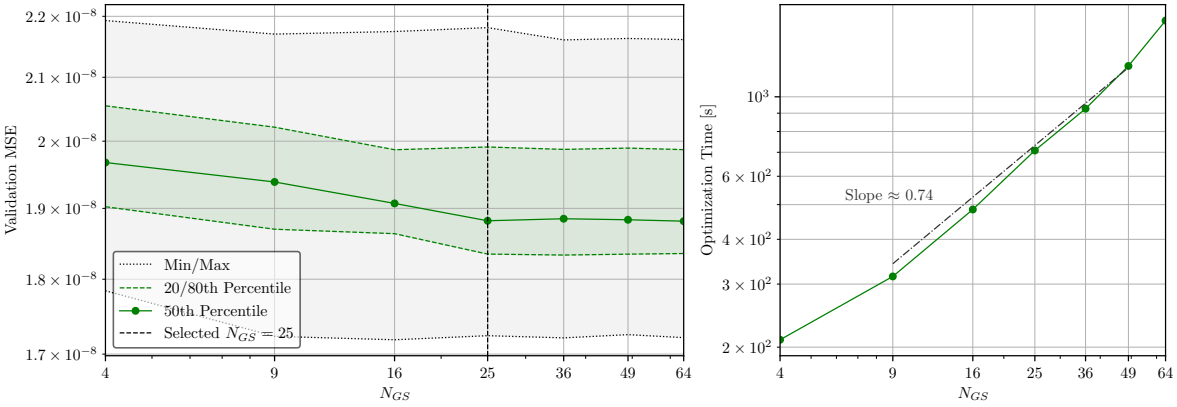


Figure 7.10: Effect of the number of initial grid search points N_{GS} on BO performance, analyzed for 1D DNS-forced injected residuals of $N_y^{(H)} = 256$ case with $N_r = 128$ and $\alpha = 0.6$. (Left) Validation MSE distribution across runs, (Right) Optimization time scaling with N_{GS}.

It is worth noting that the optimal N_{GS} for the DNS case aligns with the MMS findings, despite the significantly larger dataset and higher variability of the residuals. This suggests that the BO initialization strategy generalizes across both simplified and more complex settings, with $N_{GS} = 25$ providing a robust baseline configuration.

To mitigate the stochastic variability inherent in ESN training due to its pseudo-random reservoir initialization, an ensemble of N_{ens} ESN realizations was employed, each generated with a distinct random seed. The influence of N_{ens} on BO performance for the DNS-forced case is illustrated in Figure 7.11.

The median validation MSE stabilizes around values of $\mathcal{O}(10^{-8})$ across the tested ensemble sizes, confirming that the ESN is able to consistently capture the statistical structure of the injected residuals. For small ensembles ($N_{ens} < 10$), however, the spread between the 20th and 80th percentiles is relatively wide, reflecting the sensitivity of the BO outcome to random reservoir realizations. As N_{ens} increases, this spread contracts, indicating that larger ensembles enhance robustness by averaging out reservoir-induced fluctuations.

Beyond $N_{ens} \approx 25$, the improvements in robustness become marginal, while the computational cost continues to increase with N_{ens}. Based on this trade-off, a value of $N_{ens} = 28$ was selected as the default configuration for subsequent experiments. This choice ensures the statistical reliability of the BO results without incurring unnecessary computational overhead.

It is worth noting that, compared to the MMS case, the DNS residuals exhibit lower absolute MSE levels but greater sensitivity to ensemble averaging, reflecting the increased complexity and variability of the DNS dataset.

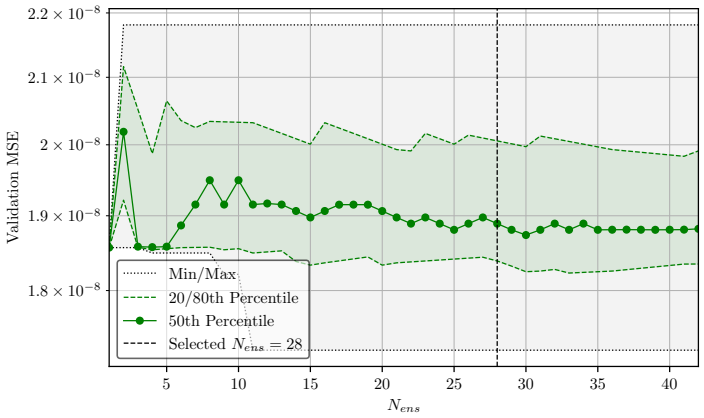


Figure 7.11: Effect of the number of ensembles N_{ens} on BO performance, analyzed for 1D DNS-forced injected residuals of $N_y^{(H)} = 256$ case with $N_r = 128$, $\alpha = 0.6$, and $N_{GS} = 25$.

Selection of Number of Neurons & Leaking Rate

After establishing the BO setup, the impact of the reservoir size N_r on ESN performance was analyzed for the 1D DNS-forced injected residuals, with the leaking rate fixed at $\alpha = 0.8$. The results are summarized in Figure 7.12, which reports the variation of validation MSE, CR, and optimization time across different reservoir sizes.

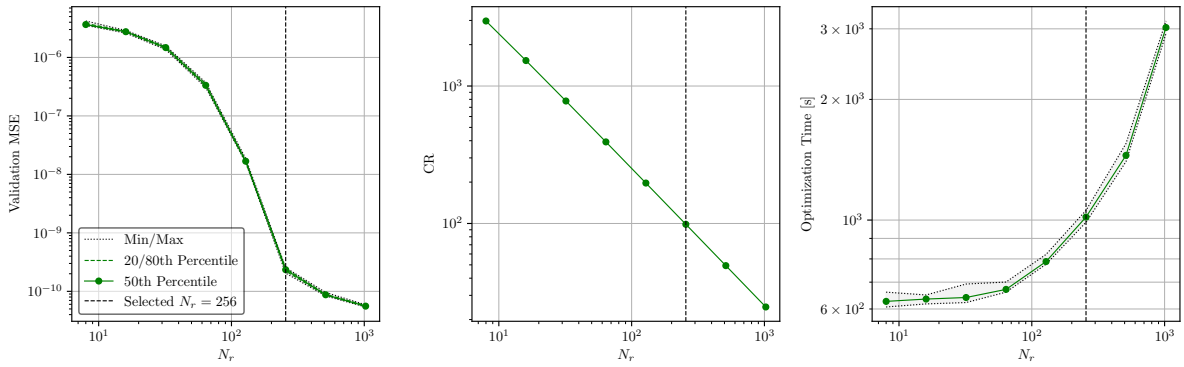


Figure 7.12: Effect of the number of reservoir neurons N_r on ESN performance for 1D DNS-forced Burgers' injected residuals. (Left) Validation MSE, (Center) Compression ratio, and (Right) Optimization time.

As expected, increasing N_r leads to a systematic reduction in validation error, reflecting the enhanced representational capacity of larger reservoirs. The decrease in MSE spans several orders of magnitude, from $\mathcal{O}(10^{-6})$ at small reservoirs to $\mathcal{O}(10^{-10})$ at large reservoirs, underscoring the ability of the ESN to capture the rich multi-scale dynamics of the DNS residuals. However, this improvement exhibits diminishing returns beyond $N_r \approx 256$, where the marginal gains in accuracy are small compared to the substantial rise in computational costs.

At the same time, the CR decreases monotonically with N_r, as enlarging the reservoir directly reduces the level of compression achieved. This highlights the fundamental trade-off between accuracy and compression. While small reservoirs yield extremely high CR values (above 10^3), they fail to capture the fine-scale variability of the DNS residuals, whereas larger reservoirs achieve near-optimal accuracy at the expense of storage efficiency.

The optimization time shows a similar trend, remaining moderate for $N_r < 128$ but increasing steeply for larger reservoirs. This behavior reflects the quadratic dependence of the regression stage on the number of reservoir states, which dominates the overall cost as N_r grows. Based on these considerations, $N_r = 256$ was selected as the default configuration, providing a balanced compromise between accuracy, compression efficiency, and computational feasibility.

Compared to the MMS case, the DNS dataset required larger reservoirs to achieve stable accuracy,

reflecting its higher intrinsic complexity and variability. This result emphasizes the sensitivity of ESN performance to the richness of the underlying dynamics, while the low-rank structure of the MMS residuals allowed accurate compression with $N_r \approx 64$, the DNS-forced case necessitates an order-of-magnitude increase in reservoir capacity to achieve comparable reliability.

The effect of the leaking rate α on ESN performance for the DNS-forced injected residuals is reported in Figure 7.13. In contrast to the MMS case, where leakage of $(\alpha \approx 0.6)$ already ensured near-optimal results, the DNS dataset exhibits a stronger dependence on α. Specifically, the validation error decreases sharply when moving away from the non-leaky configuration $(\alpha = 0)$, confirming that some degree of leakage is essential to avoid stiffness in reservoir dynamics and to capture temporal dependencies effectively.

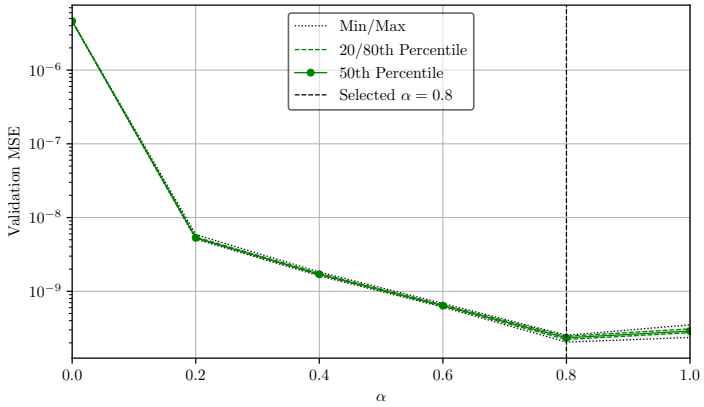


Figure 7.13: Effect of the leaking rate α on the ESN performance of validation MSE for 1D DNS-forced Burgers' injected residuals. Since α has no significant influence on optimization time or compression ratio, these metrics are omitted.

The error continues to decrease consistently up to $\alpha \approx 0.8$, at which point the validation MSE reaches its minimum. This indicates that for the more complex DNS residuals, longer effective memory, enabled by higher leaking rates, is beneficial for properly integrating the broadband temporal correlations. Beyond this point, the validation error shows only marginal increases, suggesting that excessively large α values may start to reduce the reservoir's capacity for temporal integration, but the sensitivity is weaker than in the MMS case.

As in the MMS study, the leaking rate does not significantly affect either the optimization time or the compression ratio; thus, these metrics are omitted here. Based on the trade-off between accuracy and robustness, $\alpha = 0.8$ was selected as the default value for the DNS case. This higher optimal leakage, compared to the MMS case, reflects the richer temporal structure and increased complexity of the DNS-forced residuals, which demand a reservoir with extended memory capacity.

7.3.4. Performance Assessment

This section presents the reconstruction performance of the three surrogate models, namely the benchmark POD, CAE, and ESN, applied to the 1D DNS-forced injected residuals of Burgers' equation across all refinement levels. Figure 7.14 compares the reconstructed residual distributions with the reference residuals (shown in gray, with dashed black lines denoting the temporal mean).

In line with the MMS results, all models are able to reproduce the mean values of the injected residuals with excellent accuracy across most refinement levels. The only exception is observed for the CAE at the fine resolutions $N_y^{(H)} = 256, 512$, where deviations appear in the near-wall regions. This indicates that while the dominant residual trends are consistently captured, local errors emerge when both spatial resolution and residual complexity increase.

Differences among the models become more evident in terms of the reconstructed standard deviation. At coarse and intermediate resolutions, all three methods yield similar levels of agreement with the reference spread. However, at the highest refinement $N_y^{(H)} = 512$, the CAE clearly outperforms both POD and ESN, achieving a notably better reconstruction of the temporal variability. POD suffers from its dimensionality reduction threshold, which leads to underrepresentation of higher-order fluctua-

tions, while ESN tends to smooth out residual variability due to the limited reservoir size adopted for computational tractability.

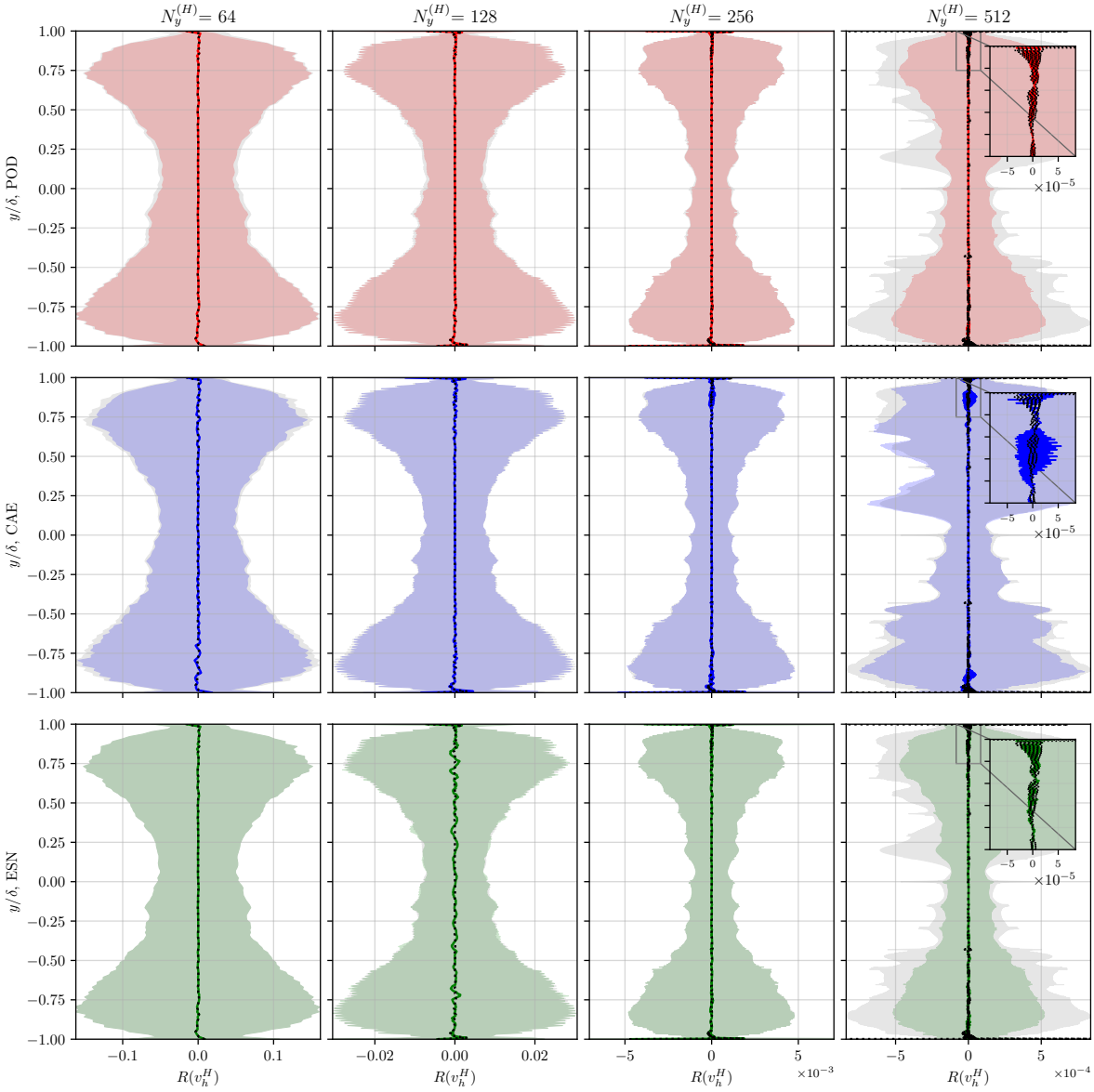


Figure 7.14: Reconstructed 1D DNS-forced Burgers' injected residuals $\tilde{R}\left(v_h^H\right)$ for all refinements $N_y^{(H)}$ using POD (top), CAE (middle), and ESN (bottom). The shaded regions are the actual residuals $R\left(v_h^H\right)$, with the dashed black lines indicating their temporal mean.

Figure 7.15 compares the performance of POD, CAE, and ESN in terms of reconstruction fidelity and compression efficiency for the 1D DNS-forced Burgers' injected residuals. In contrast to the MMS case, the relative ranking of the methods changes due to the higher variability and complexity of the residuals.

In terms of accuracy, the ESN consistently achieves the lowest reconstruction MSE values, where its recurrent dynamics allow it to resolve fine-scale temporal correlations, except for the finest resolution. POD maintains competitive accuracy, particularly at $N_y^{(H)} = 256$ and 512, where its truncation strategy is less restrictive. The CAE, by contrast, exhibits the largest errors at coarse resolutions, reflecting the difficulty of capturing stochastic fluctuations with limited latent dimensions. While its performance improves with refinement, it remains less accurate than ESN and POD, which marks a clear departure from the MMS results, where the CAE was consistently superior.

The compression ratio highlights a complementary trade-off. Owing to its compact reservoir representation, the ESN achieves compression ratios that are orders of magnitude larger than those of both POD and CAE, which remain bounded to values $\mathcal{O}(10^0\text{–}10^1)$. This demonstrates the suitability of ESN for extreme dimensionality reduction, though at the expense of higher reconstruction costs.

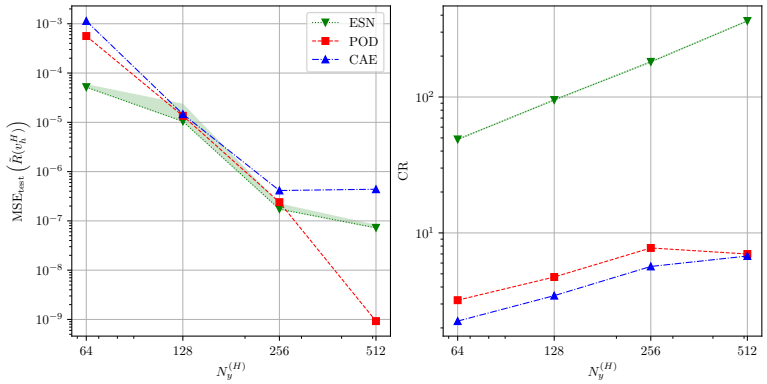


Figure 7.15: Performance comparison of POD, CAE, and ESN for the 1D DNS-forced Burgers' injected residuals, showing test MSE and compression ratio (CR) as functions of spatial refinement $N_y^{(H)}$.

To provide a unified overview of the computational requirements, Figure 7.16 presents the preparation and reconstruction times for all surrogate models applied to the 1D DNS-forced Burgers' injected residuals. These timing results are included jointly here since they were deliberately omitted from the main performance comparison, which focused solely on the compression ratio and reconstruction fidelity. This separation ensures that the core assessment remains device-independent, while the timing data serve to illustrate the relative computational scaling behavior of each model.

It should be emphasized that these results are strongly hardware-dependent and are therefore not intended as absolute performance benchmarks. The CAE was trained on *Device 1*, while both the ESN optimization and POD computations were executed on *Device 2*. For this reason, the reported times are best interpreted in terms of scaling trends rather than raw magnitudes.

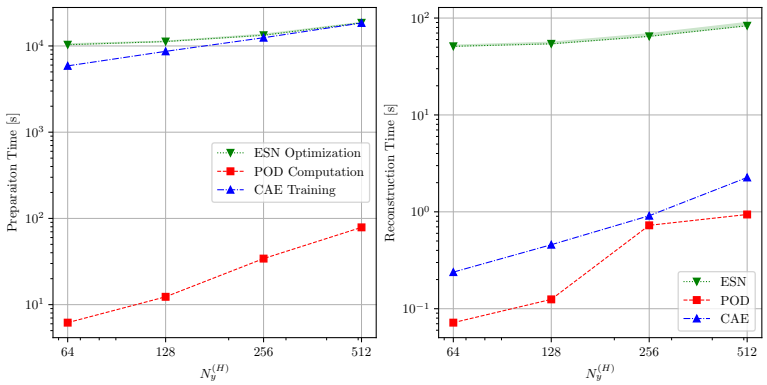


Figure 7.16: Preparation and reconstruction times for the 1D DNS-forced Burgers' injected residuals, comparing the POD, CAE, and ESN across different spatial resolutions $N_y^{(H)}$. The CAE was trained on *Device 1*, while the POD and ESN computations were performed on *Device 2*.

As expected, the computational cost for all models increases consistently with spatial refinement $N_y^{(H)}$, reflecting the growing input dimensionality and the corresponding rise in model complexity. The mild variability observed in the ESN optimization times stems from the stochastic nature of the BO process and random reservoir initialization rather than from numerical instability.

The left panel depicts the preparation time, which quantifies the total computational cost associated with building each surrogate model. Specifically, this corresponds to the training time for the CAE, the optimization time for the ESN, and the eigenvalue computation time for the POD. Conversely, the right panel reports the reconstruction time, which represents the cost of performing a single inference or for-

ward reconstruction using each trained surrogate. Here, the difference between the methods becomes more pronounced: the POD and CAE achieve near-instantaneous reconstructions, while ESN incurs additional computational overhead associated with the recurrent evaluation of reservoir dynamics over the temporal sequence.

Despite the relatively higher absolute cost of the AI-based methods, both the CAE and ESN maintain computationally feasible runtimes for DNS-scale problems, confirming their suitability for large unsteady datasets and high-resolution flow reconstructions.

To further assess the relative efficiency of the surrogate compression methods, Figure 7.17 compares the reconstructed injected residuals $R(v_h^H)$ obtained using CAE and ESN against POD reconstructions computed at identical compression ratios to those achieved by CAE and ESN. This comparison enables a fair evaluation of reconstruction quality at equal compression efficiency, isolating the effect of model architecture from the influence of data reduction level.

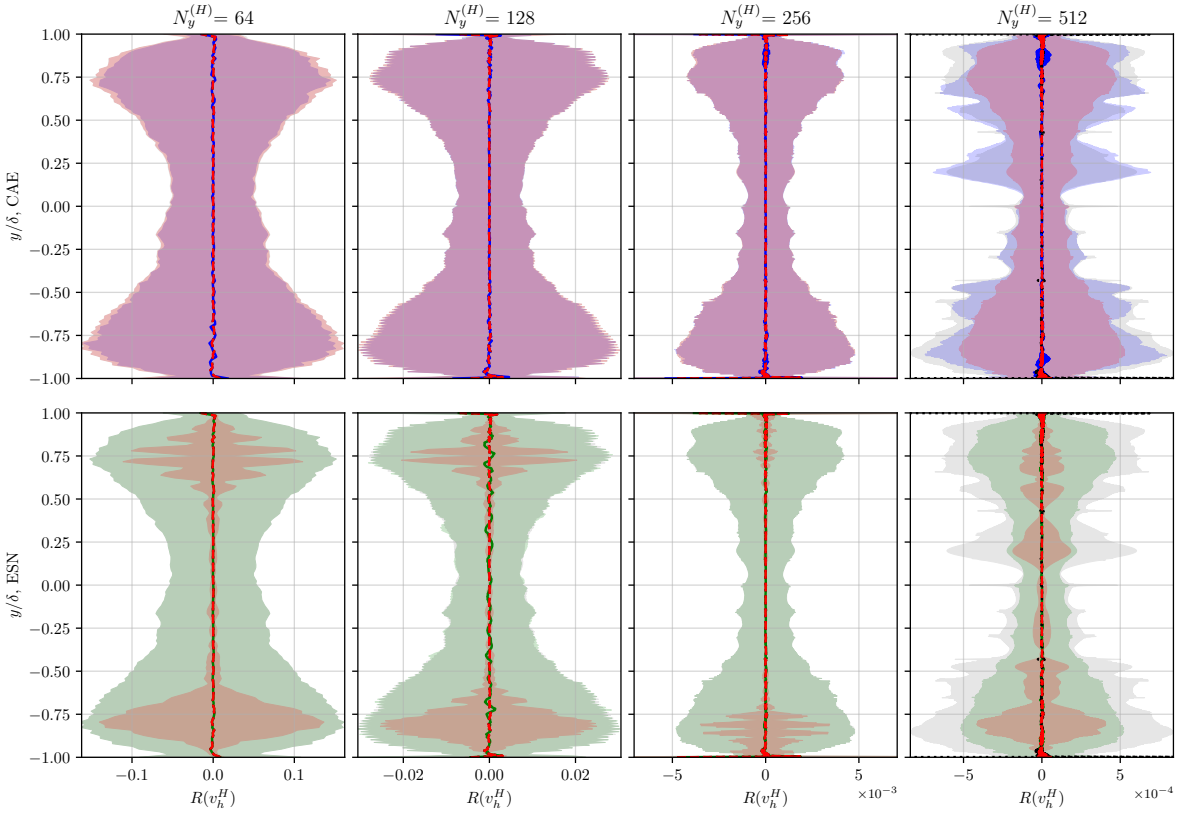


Figure 7.17: Comparison of the reconstructed 1D DNS-forced Burgers' injected residuals $R(v_h^H)$ obtained using CAE (blue), ESN (green), and POD (red), where the POD reconstruction was performed at the same CR achieved by each CAE and ESN model. The gray shaded region represents the FVM reference solution, while the colored shaded bands denote temporal variability and the solid lines indicate temporal means.

As shown, the POD reconstructions (in red) exhibit markedly different behavior depending on the reference CR. For the CAE-matched CRs, which are relatively moderate and close to the POD's original compression levels, the reduced CR leads to enhanced reconstruction fidelity, with the temporal mean and variability closely following the FVM reference (gray). However, when POD is constrained to the significantly higher compression ratios achieved by the ESN, it fails to recover meaningful spatial features, and the temporal statistics collapse toward zero, indicating severe information loss.

This behavior highlights the strong compression capability of the ESN, which maintains a stable reconstruction even at extremely low-dimensional representations, whereas linear decompositions such as POD rapidly deteriorate when subjected to the same reduction levels. The results also confirm that the CAE operates in a compression regime comparable to POD while achieving similar or slightly improved accuracy, whereas the ESN achieves far greater compression at the cost of nonlinear encoding

complexity.

Overall, the results confirm that all three surrogate models are robust in reproducing the residual means, which is the most critical quantity for error estimation in time-averaged QoIs. Yet, their relative strengths diverge when considering variability, efficiency, and scalability. The ESN achieves the best overall mean accuracy and compression efficiency, benefiting from its ability to capture temporal complexity, while the CAE demonstrates superior capability in reproducing variability at fine resolutions and consistently delivers the fastest reconstruction times. POD, though less competitive, provides a balanced alternative with reasonable accuracy and interpretability, albeit with limitations in handling the full complexity of DNS data compared to the nonlinear and reservoir-based approaches.

7.4. Primal Solution Compression

To extend the analysis of *Scenarios* 2 and 3 to the 1D DNS-forced Burgers' problem, a surrogate representation of the fine primal solution is required. This surrogate provides the foundation for computing the corresponding surrogate adjoint solution, $\tilde{\psi}$. As in the MMS case, the same compression methodologies, the benchmark POD, CAE, and ESN, were employed, with modifications introduced to account for the increased variability and complexity of the DNS dataset. The specific adaptations made for the primal solution in this context are discussed in the following subsections.

7.4.1. Proper Orthogonal Decomposition

The results of the offline POD benchmark applied to the fine primal solution of the 1D DNS-forced Burgers' equation are presented in Figure 7.18. In contrast to the MMS case, the convergence of the retained modal energy $\mathcal{E}_k$ is noticeably slower, reflecting the richer dynamics and higher variability of the DNS dataset. To reach the target threshold of 99% energy retention, between $k = 43$ and $k = 47$ modes are required, depending on the refinement level. This substantial increase in modal dimensionality highlights the more complex nature of the DNS primal field compared to the nearly rank-one structure observed for MMS.

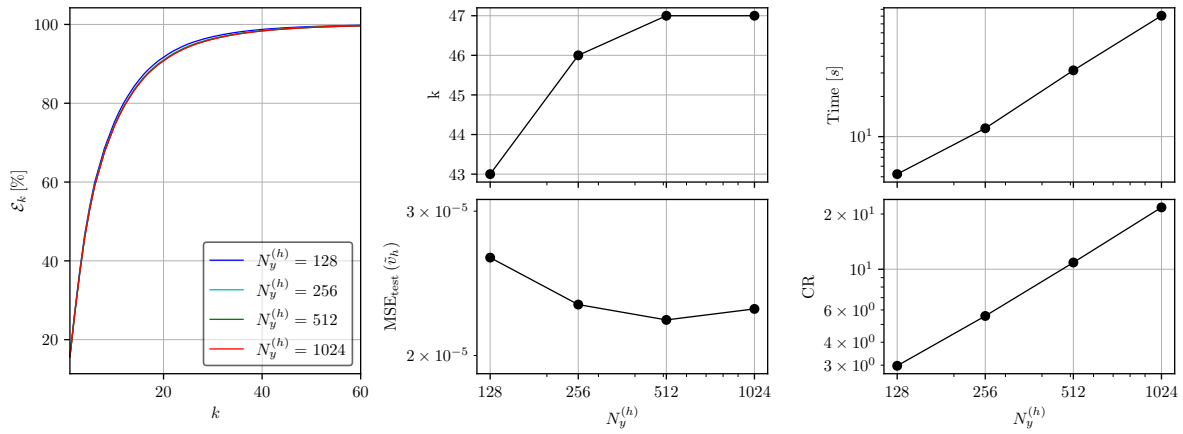


Figure 7.18: Offline POD benchmark results of reconstructed fine primal solution of 1D DNS-forced Burgers' equation for different spatial resolutions. (Left) Cumulative energy content $\mathcal{E}_k$ as a function of retained modes k, with threshold at 99%. (Right) Corresponding required number of modes k, POD computation time, reconstruction test MSE, and compression ratio (CR).

The influence of this increased complexity is evident in the performance metrics. The reconstruction test MSE remains of the order of 10^{-5}, several orders of magnitude higher than the MMS case. This gap directly reflects the fact that a larger number of modes is needed to capture the energetic content of DNS solutions, making the reduced-order representation less compact. Similarly, the compression ratio grows with refinement but remains relatively modest, particularly at coarse resolutions, due to the higher number of modes retained.

Finally, the POD eigenvalue computation time exhibits strong scaling with spatial resolution, increasing from a few seconds at $N_y^{(h)} = 128$ to nearly one minute for $N_y^{(H)} = 1024$. While this cost remains affordable in absolute terms, it highlights the practical trade-off of POD in DNS contexts; the method provides robust and accurate reconstructions, but the dimensionality of the reduced space grows sig-

nificantly with problem complexity, limiting compression efficiency compared to the MMS scenario.

7.4.2. Convolutional Autoencoder

The CAE was also applied to the fine primal solution using the same training strategy and architectural framework outlined in Section 5.3.2. In this case, the PCA-based analysis revealed that the latent dimension $d_{\mathcal{Z}}$ required to capture the dominant variability of the DNS primal fields is significantly larger than in the MMS case. As reported in Table 7.4, the dimensionality stabilizes at $d_{\mathcal{Z}} = 17$ across all refinement levels.

This result highlights the increased complexity of the DNS dataset compared to the highly coherent MMS fields. Unlike the manufactured solution, where the dynamics were essentially rank-one, the DNS primal solution exhibits richer structures and higher-order variability that cannot be compressed into a single dominant latent direction. The fact that $d_{\mathcal{Z}}$ remains constant across refinements suggests that the intrinsic dimensionality of DNS dynamics is grid-independent but still requires a nontrivial latent space to achieve faithful compression.

Table 7.4: Latent space dimension $d_{\mathcal{Z}}$ determined for the CAE architecture based on PCA analysis of the 1D DNS-forced Burgers' fine primal solution.

$N_y^{(h)}$	128	256	512	1024
$d_{\mathcal{Z}}$	17	17	17	17

Although the CAE architecture remained identical to that used for the injected residuals $R(v_h^H)$ (see Figure 7.8), the larger latent dimension $d_{\mathcal{Z}} = 17$ resulted in a higher number of trainable parameters compared to the MMS case. The detailed encoder and decoder layer specifications for the fine primal solution are provided in Table 7.5. The remaining architectures for higher resolutions were generated analogously and are presented in Appendix B.2.

To ensure consistency and comparability, the overall optimization settings were kept identical to those used in the residual case. However, two adjustments were made; a subsampling rate of 5 was applied to the fine primal dataset, and the adaptive learning rate schedule was initialized with $\gamma_0 = 5 \times 10^{-3}$ and a decay factor of $\gamma_r = 2 \times 10^{-2}$.

Figure 7.19 presents the training and validation batch-averaged MSE during CAE learning of the fine primal solution for all spatial resolutions $N_y^{(h)}$. Across all cases, the MSE decreases rapidly during the first $\sim$100 epochs, followed by a gradual decay as the adaptive learning rate schedule progresses. Convergence stabilizes around values of $\mathcal{O}(10^{-3})$ to $\mathcal{O}(10^{-4})$, depending on the resolution. The close agreement between the training and validation curves throughout the optimization process confirms that the model generalizes well, with no evidence of overfitting.

The oscillations visible in both curves are a result of the relatively large initial learning rate and the stochasticity introduced by mini-batch sampling. These oscillations progressively diminish as the learning rate decays, leading to a smooth convergence trend. Compared to the MMS case, the convergence of the DNS model is more gradual, reflecting the increased complexity and variability of the fine-scale flow structures. Nevertheless, the CAE successfully learns a compact and stable latent representation of the primal field across all spatial refinements.

Table 7.5: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D DNS-forced Burgers' fine primal solution for the case $N_y^{(h)} = 128$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	16
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E2	Conv. 1D	$(4, 64)$	$(8, 32)$	$(4, 2, 1)$	128
	BN	$(8, 32)$	$(8, 32)$		16
	LReLU	$(8, 32)$	$(8, 32)$		0
E3	Conv. 1D	$(8, 32)$	$(16, 16)$	$(4, 2, 1)$	512
	BN	$(16, 16)$	$(16, 16)$		32
	LReLU	$(16, 16)$	$(16, 16)$		0
E4	Conv. 1D	$(16, 16)$	$(32, 8)$	$(4, 2, 1)$	2048
	BN	$(32, 8)$	$(32, 8)$		64
	LReLU	$(32, 8)$	$(32, 8)$		0
E5	Conv. 1D	$(32, 8)$	$(32, 4)$	$(4, 2, 1)$	4096
	BN	$(32, 4)$	$(32, 4)$		64
	LReLU	$(32, 4)$	$(32, 4)$		0
E6	Conv. 1D	$(32, 4)$	$(32, 2)$	$(4, 2, 1)$	4096
	BN	$(32, 2)$	$(32, 2)$		64
	LReLU	$(32, 2)$	$(32, 2)$		0
E7	Conv. 1D	$(32, 2)$	$(32, 1)$	$(4, 2, 1)$	4096
	BN	$(32, 1)$	$(32, 1)$		64
	LReLU	$(32, 1)$	$(32, 1)$		0
E8	Flatten	$(32, 1)$	(32)		0
E9	Linear	(32)	(57)		561

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(57)	(32)		576
D2	Upsample	$(32, 1)$	$(32, 2)$		0
	Conv. 1D	$(32, 2)$	$(32, 2)$	$(3, 1, 1)$	3072
	BN	$(32, 2)$	$(32, 2)$		64
	LReLU	$(32, 2)$	$(32, 2)$		0
D3	Upsample	$(32, 2)$	$(32, 4)$		0
	Conv. 1D	$(32, 4)$	$(32, 4)$	$(3, 1, 1)$	3072
	BN	$(32, 4)$	$(32, 4)$		64
	LReLU	$(32, 4)$	$(32, 4)$		0
D4	Upsample	$(32, 4)$	$(32, 8)$		0
	Conv. 1D	$(32, 8)$	$(32, 8)$	$(3, 1, 1)$	3072
	BN	$(32, 8)$	$(32, 8)$		64
	LReLU	$(32, 8)$	$(32, 8)$		0
D5	Upsample	$(32, 8)$	$(32, 16)$		0
	Conv. 1D	$(32, 16)$	$(16, 16)$	$(3, 1, 1)$	1536
	BN	$(16, 16)$	$(16, 16)$		32
	LReLU	$(16, 16)$	$(16, 16)$		0
D6	Upsample	$(16, 16)$	$(16, 32)$		0
	Conv. 1D	$(16, 32)$	$(8, 32)$	$(3, 1, 1)$	384
	BN	$(8, 32)$	$(8, 32)$		16
	LReLU	$(8, 32)$	$(8, 32)$		0
D7	Upsample	$(8, 32)$	$(8, 64)$		0
	Conv. 1D	$(8, 64)$	$(4, 64)$	$(3, 1, 1)$	96
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	(128)	$(1, 128)$	$(3, 1, 1)$	12

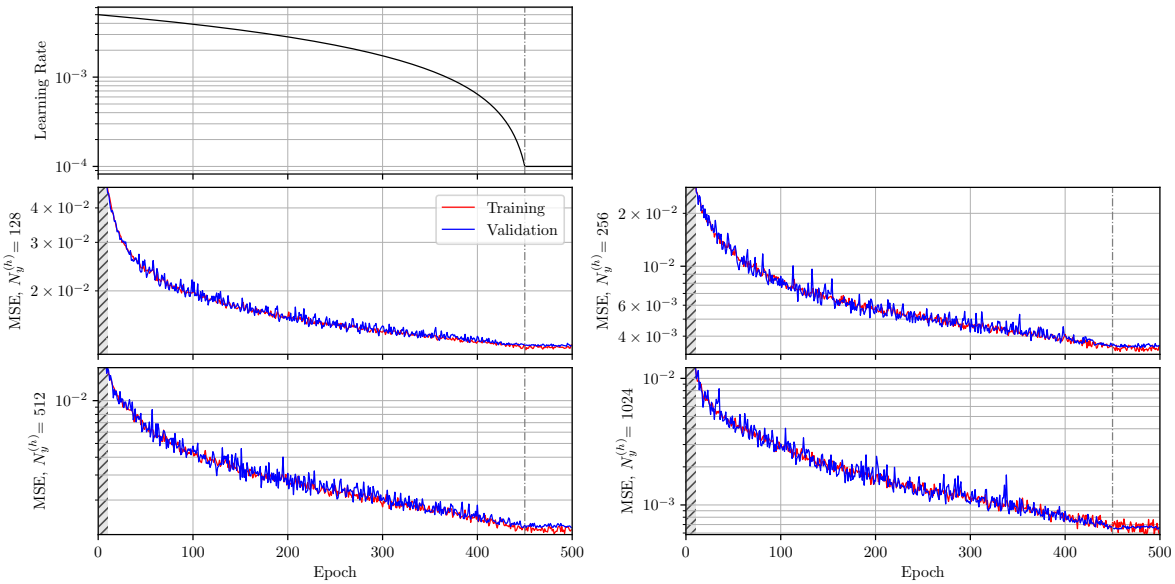


Figure 7.19: Training and validation batch-averaged MSE during CAE training of 1D DNS-forced Burgers' fine primal solution for different spatial resolutions $N_y^{(H)}$. The top-left panel shows the adaptive learning rate schedule applied during the first 450 epochs, followed by a constant value. (The initial epochs are truncated in all MSE plots for visualization purposes.)

7.4.3. Echo State Network

The ESN was employed to compress the fine primal solution of the 1D DNS-forced Burgers' problem, following the same architectural configuration used for the injected residuals. The network consisted of a single pseudorandomly constructed reservoir, with fixed internal weights and output connections determined through linear regression. The reservoir size N_r and leaking rate α were kept identical to those adopted in the residual compression experiments to ensure consistency across models.

The training dataset comprised the complete temporal history of 500,000 time steps, while the validation set was constructed from 199 overlapping intervals of 5,000 samples each. This configuration ensured statistically representative coverage of the system's temporal variability over the full simulation horizon.

BO was used to identify the optimal hyperparameters. The search space was defined as $\rho \times \sigma_{\text{in}} \in [0.1,\, 1] \times [10^{-3},\, 10^{1}]$, corresponding to the spectral radius and input scaling, respectively. The Tikhonov regularization parameter β was again selected from the discrete grid $10^{-6},\, 10^{-9}$. These bounds were chosen to accommodate the temporal variability and amplitude distribution of the DNS primal field, ensuring sufficient excitation of the reservoir while maintaining stable internal dynamics.

7.4.4. Performance Assessment

This section presents the performance assessment of the three surrogate models, benchmark POD, CAE, and ESN, applied to the reconstruction of the fine primal solution $\hat{v}_h$ across all refinement levels $N_y^{(h)}$. Figure 7.20 compares the reconstructed temporal mean (solid lines) and the standard deviation (shaded regions) against the reference fine solution v_h (dashed black lines).

All three models demonstrate excellent reconstruction quality, successfully capturing both the temporal mean and the overall distribution of fluctuations across all refinement levels. The agreement with the reference data remains strong even for the finest grid, confirming that each method can effectively encode and reconstruct the dominant spatial–temporal features of the fine DNS field. Minor deviations are observed in the coarser resolutions of CAE results, particularly near the outer regions. This small discrepancy can be attributed to the subsampling applied during training, which limits the network's exposure to certain high-frequency temporal patterns.

POD and ESN exhibit nearly indistinguishable performance in terms of mean reconstruction. Overall, all surrogate models maintain stable and accurate behavior across the tested resolutions, confirming

their robustness and consistency in reproducing the fine primal dynamics of the 1D DNS-forced Burgers' problem.

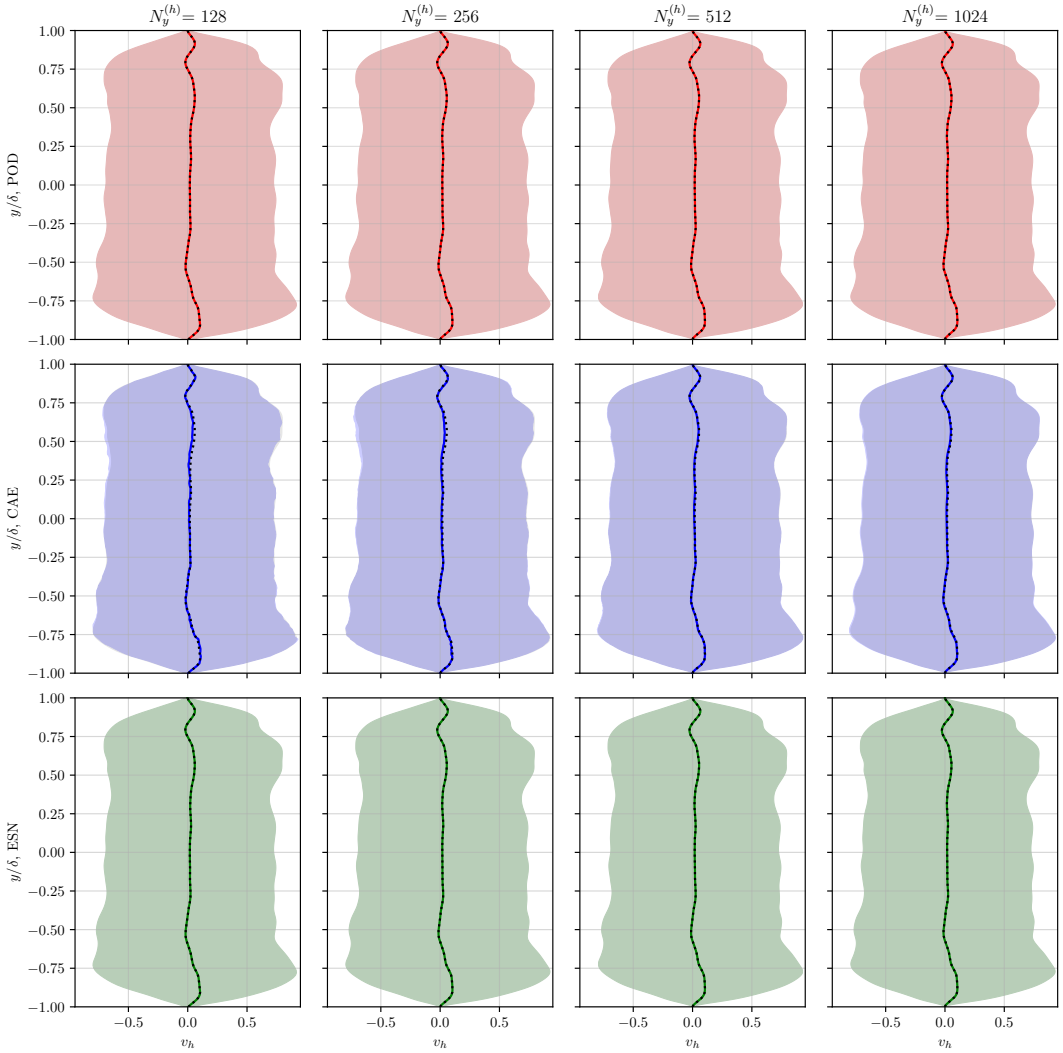


Figure 7.20: Reconstructed 1D DNS-forced Burgers' fine primal solution $\tilde{v}_h$ for all refinements $N_y^{(h)}$ using POD (top), CAE (middle), and ESN (bottom). The shaded regions are the actual solution v_h, with the dashed black lines indicating their temporal mean.

Figure 7.21 provides a quantitative comparison of POD, CAE, and ESN in terms of test MSE and compression ratio for the 1D DNS-forced Burgers' fine primal solution across all refinement levels $N_y^{(h)}$. In contrast to the residual-reconstruction case, the differences among the three methods are more pronounced here. In terms of accuracy, POD consistently yields the lowest test MSE across all resolutions, confirming its strong suitability for coherent datasets. The CAE, although slightly less accurate at coarse grids, exhibits rapid improvement with refinement, ultimately achieving near-POD accuracy at $N_y^{(h)} = 1024$. This behavior indicates that the convolutional architecture effectively learns the underlying spatial structures as the resolution increases. The ESN, on the other hand, maintains nearly constant accuracy across all resolutions, suggesting that its temporal encoding capacity is already saturated for the chosen reservoir size.

When examining compression efficiency, ESN clearly achieves the highest CR, exceeding two orders of magnitude at the finest grid. This result highlights its strength as a highly compact temporal surrogate, capable of representing long sequences with minimal storage cost. However, this gain comes at the expense of accuracy, as reflected by its higher MSE values. By contrast, the CAE provides a balanced trade-off, its CR increases consistently with refinement while maintaining high fidelity, making it an effective spatial encoder for high-dimensional data. The POD offers the lowest CR due to its explicit

mode storage, but this is offset by its superior precision and interpretability.

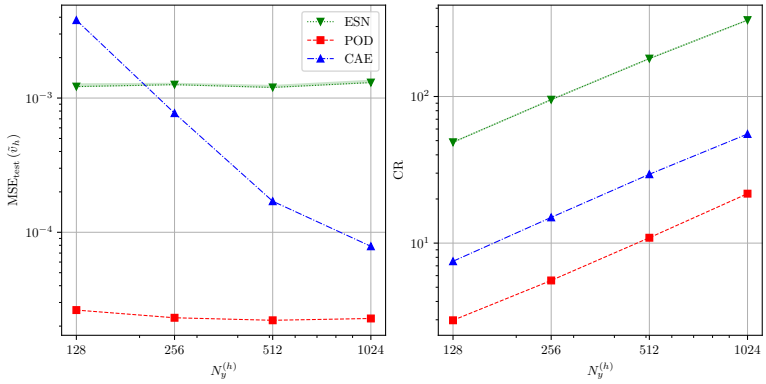


Figure 7.21: Performance comparison of POD, CAE, and ESN for the 1D DNS-forced Burgers' fine primal solution, showing test MSE and compression ratio (CR) as functions of spatial refinement $N_y^{(h)}$.

To provide a comprehensive overview of computational performance, Figure 7.22 presents the preparation and reconstruction times of the surrogate models for the DNS-forced fine primal solution. Similar to the injected residuals, these timing results are reported together here, while the corresponding performance metrics are discussed separately to maintain a hardware-independent comparison. As in the residual case, these results are intended to illustrate the relative scaling of computational cost with resolution rather than serve as absolute benchmarks, since each method was executed on a different hardware device. The CAE training was performed on *Device 1*, while both the POD and ESN computations were executed on *Device 2*.

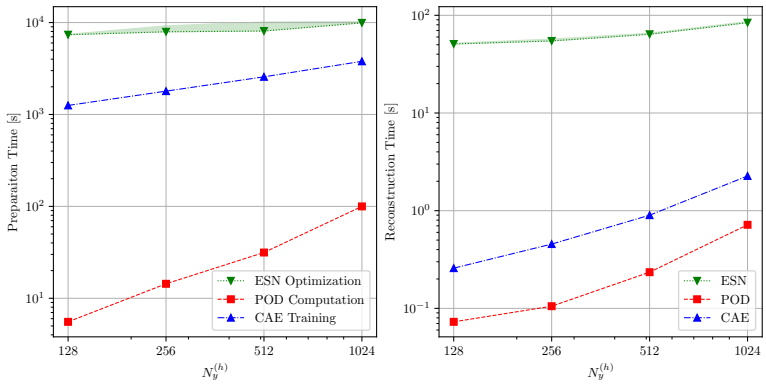


Figure 7.22: Preparation and reconstruction times for the 1D DNS-forced Burgers' fine primal solution, comparing the POD, CAE, and ESN across different spatial resolutions $N_y^{(h)}$. The POD and ESN were executed on *Device 2*, while the CAE was trained on *Device 1*.

The left panel illustrates the preparation time, which represents the total computational effort required to construct each surrogate model. For the CAE, this corresponds to the full training duration; for the ESN, it reflects the time spent on the Bayesian optimization routine; and for the POD, it accounts for the computation of the complete set of eigenpairs. Across all methods, the preparation time increases monotonically with spatial refinement $N_y^{(h)}$, as expected from the growing input dimensionality and model complexity. Among the three, POD remains the most computationally efficient, scaling nearly linearly with the number of spatial elements on a logarithmic scale. In contrast, the CAE and ESN display higher overall costs and super-linear growth trends due to their nonlinear optimization and training procedures.

The right panel shows the reconstruction time, representing the cost of performing a single forward evaluation once each surrogate has been trained. As anticipated, both POD and CAE provide nearly instantaneous reconstructions, while the ESN incurs a slightly higher cost because of the recurrent propagation of reservoir states over time. Nevertheless, the computational load of all AI-based methods

remains manageable, with reconstruction times staying below $\mathcal{O}(10^2)$ s even at the finest resolution of $N_y^{(h)} = 1024$.

The modest variability observed for the ESN arises from the stochastic initialization of its reservoir and the probabilistic nature of the BO search, rather than from any numerical instability. Despite their relatively higher preparation costs, both the CAE and ESN exhibit scalable and computationally feasible runtimes for DNS-scale problems. These findings reaffirm that the additional investment during the training and optimization stages is justified by their strong reconstruction accuracy and considerable data compression efficiency in large unsteady flow datasets.

To evaluate the reconstruction performance of the surrogate models at equivalent compression levels, Figure 7.23 compares the fine primal solutions v_h reconstructed using CAE and ESN against POD reconstructions computed at the same CRs achieved by each neural model. This comparison isolates the effect of model architecture on reconstruction quality under identical data reduction constraints.

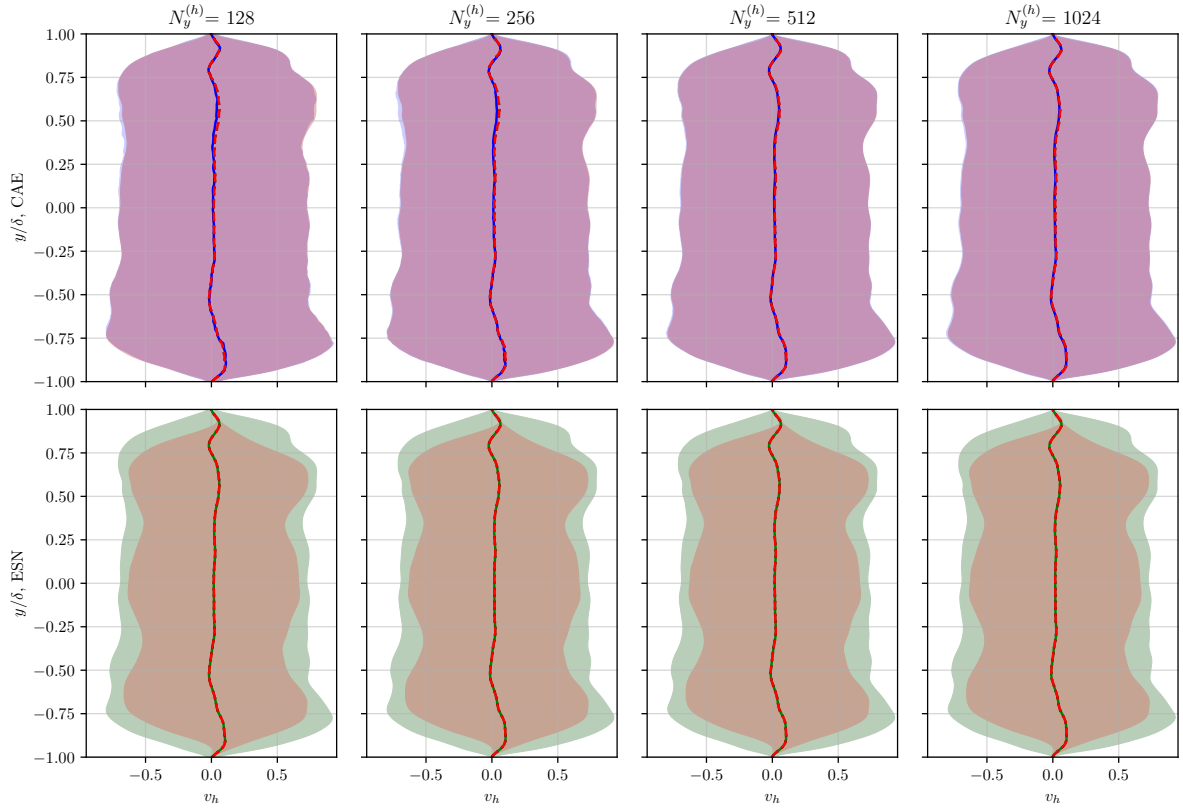


Figure 7.23: Comparison of reconstructed fine primal solutions v_h for the 1D DNS-forced Burgers' problem obtained using CAE (top) and ESN (bottom), benchmarked against POD reconstructions performed at identical compression ratios (CRs) achieved by each model. The gray shaded regions represent the FVM reference solution, while the colored shaded areas indicate temporal variability and the solid lines denote temporal means.

When the POD is compressed to match the CRs achieved by the CAE, the reconstruction accuracy decreases slightly compared to its baseline configuration, although the overall mean and variance profiles remain in close agreement with the FVM reference (gray). Both the POD and CAE reproduce the flow structures with comparable fidelity across all refinement levels, confirming that at moderate compression levels, the CAE does not compromise reconstruction quality relative to a linear modal approach.

However, when the POD is forced to match the significantly higher CRs achieved by the ESN, its reconstruction collapses, losing nearly all meaningful variability and deviating from the reference field. In contrast, the ESN maintains an accurate reconstruction of both the temporal mean and the variance envelope across all resolutions. This highlights the network's capacity to exploit temporal correlations within the data, enabling efficient compression far beyond the practical limits of purely spatial, linear decompositions such as POD.

Overall, these results confirm that while POD remains the most accurate method for reconstructing the DNS fine primal solution, CAE offers the most practical balance among accuracy, compression, and computational cost. ESN, though more computationally expensive in this setting, excels in compactness and may therefore serve as an efficient temporal surrogate in scenarios where storage constraints dominate over computational cost requirements.

7.5. Adjoint Solution

As discussed in Chapter 3, the second dataset required for evaluating the output error estimate is the adjoint solution. The adjoint problem for the unsteady 1D viscous Burgers' equation under DNS forcing was derived following the continuous adjoint formulation presented in Section 3.1. The procedure is identical to that outlined for the MMS case, involving the local linearization of the nonlinear convective term around the primal solution $v(y,t)$ and the application of integration by parts to obtain the adjoint operator.

The only difference between the two cases lies in the choice of the QoI, which directly modifies the adjoint forcing term. As a result, the governing adjoint equation for the DNS-forced Burgers' problem reads:

$$\begin{cases} -\dfrac{\partial \psi}{\partial t} - v(y,t)\dfrac{\partial \psi}{\partial y} - \dfrac{1}{Re_\tau}\dfrac{\partial^2 \psi}{\partial y^2} = 4\,e^{-50y^2}\,v^3(y,t), & (y,t) \in \Omega \times I, \\ \psi(y,t) = 0, & (y,t) \in \partial\Omega \times I, \\ \psi(y,T) = 0, & y \in \Omega. \end{cases} \tag{7.10}$$

Here, $\psi(y,t)$ denotes the adjoint variable, $v(y,t)$ the primal DNS solution, and Re_τ the friction Reynolds number, which controls the viscous diffusion term. The right-hand side forcing term corresponds to the selected QoI, which concentrates the adjoint sensitivity in the channel centerline region where the forcing is maximal. The boundary and terminal conditions are homogeneous and consistent with those used for the primal problem. This formulation provides the adjoint field $\psi(y,t)$ necessary for the subsequent evaluation of the output error estimate and surrogate-based adjoint computation discussed in the following sections.

The adjoint field $\psi(y,t)$ for the DNS-forced Burgers' problem is shown in Figure 7.24. The solution was integrated backward in time from $t = 5$ s to $t = 0$ s, starting from the terminal condition $\psi(y,T) = 0$. The adjoint structure reflects the sensitivity of the Gaussian-weighted functional $\bar{J}$ defined in Eq. (7.5), which emphasizes the wall-normal velocity component $v(y,t)$ near the channel centerline ($\mu = 0$, $\sigma = 0.1$).

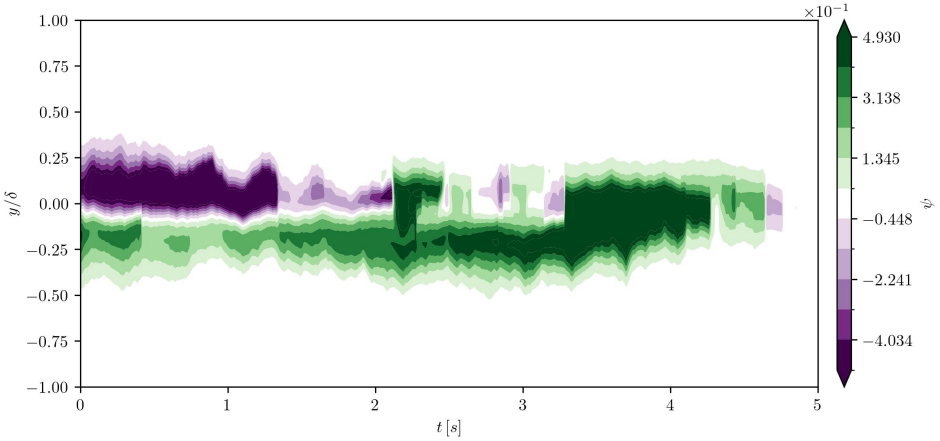


Figure 7.24: Temporal evolution of the adjoint solution $\psi(y,t)$ for the 1D DNS-forced Burgers' equation for $N_y^{(H)} = 128$.

At the onset of integration (near $t/T = 1$), distinct regions of positive and negative adjoint amplitude emerge around the centerline, indicating the strong influence of central flow perturbations on the weighted output functional. As the adjoint evolves backward in time, these high-sensitivity regions

progressively diffuse, while their influence spreads toward the channel walls due to the convective and diffusive terms. Overall, the adjoint field shows that the most significant sensitivities are concentrated near the final time and around the channel midline, confirming that perturbations in this region exert the largest cumulative effect on the Gaussian-weighted QoI $\bar{J}$.

Figure 7.25 presents the spatial distribution of the time-averaged adjoint velocity $\overline{\psi}^+$ and its corresponding adjoint Reynolds stress $\overline{\psi'\psi'}^+$ for the DNS-forced Burgers' problem across all spatial resolutions $N_y^{(H)}$. Both quantities exhibit a clear concentration around the channel centerline, consistent with the Gaussian weighting of the QoI defined in Eq. (7.5).

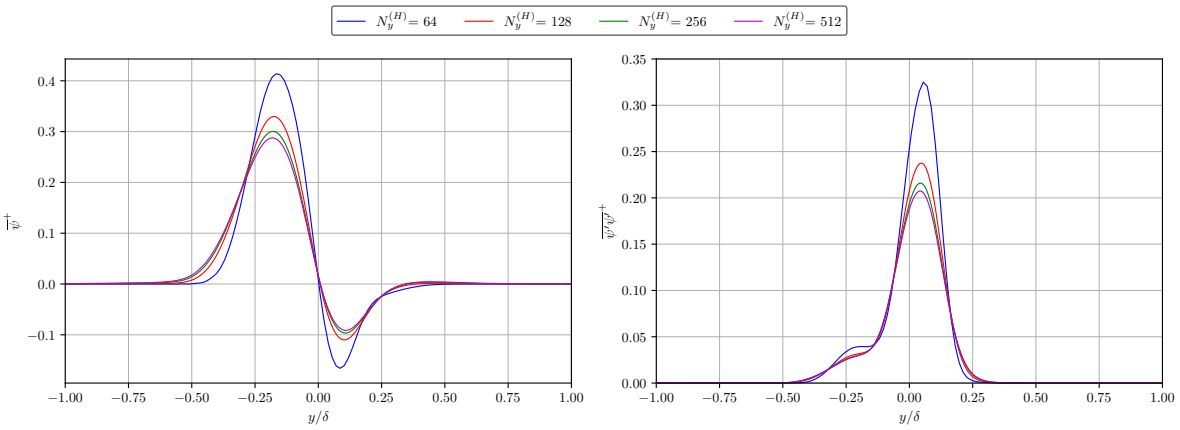


Figure 7.25: Time-averaged adjoint velocity $\overline{\psi}^+$ (left) and adjoint Reynolds stress $\overline{\psi'\psi'}^+$ (right) for the 1D DNS-forced Burgers' equation at different spatial resolutions $N_y^{(H)}$.

The mean adjoint velocity $\overline{\psi}^+$ displays a smooth, symmetric shape about the centerline, with its peak magnitude aligned with the region of highest weighting in the QoI functional. This indicates that the adjoint sensitivity is primarily localized near the channel core, where perturbations in the primal velocity $v(y,t)$ most strongly influence the output. As the grid is refined, the shape of $\overline{\psi}^+$ converges, and only minor amplitude differences persist, confirming numerical consistency of the adjoint solution across resolutions.

The adjoint Reynolds stress $\overline{\psi'\psi'}^+$ follows a similar spatial distribution but with smaller magnitudes, reflecting the relatively low temporal variability of the adjoint field. Unlike the primal flow, where turbulent-like fluctuations can dominate the variance, the adjoint field remains smooth due to the backward-in-time nature of its evolution and the absence of nonlinear amplification mechanisms. The stress peaks slightly downstream of the mean adjoint velocity maximum, suggesting localized sensitivity to transient features in the primal field.

Figure 7.26 presents the time-averaged adjoint field $\overline{\psi}$ and its standard deviation obtained by solving the adjoint PDE using the reconstructed fine primal solutions from POD (top), CAE (middle), and ESN (bottom) for all spatial refinements $N_y^{(H)}$. The shaded regions represent the instantaneous fluctuations around the mean, while the dashed black lines denote the reference adjoint solution computed from the DNS-based primal field.

The results reveal that the POD-based adjoint reconstruction closely matches the reference solution across all refinement levels. Both the mean profile and its variance distribution are accurately reproduced, reflecting the POD model's ability to retain the dominant spatial structures and gradients that govern the adjoint forcing.

In contrast, the CAE-based adjoint exhibits noticeable deviations in both the temporal mean and the variance distribution, despite the CAE achieving high reconstruction accuracy for the primal field. This discrepancy arises from the adjoint solver's intrinsic sensitivity to the spatial derivatives of the primal solution and the nonlinear nature of the adjoint forcing term. Even minor biases or smoothing effects introduced by the convolutional encoder–decoder architecture, particularly due to subsampling and interpolation operations, can alter the local gradients of $v(y,t)$. These seemingly small discrepancies are amplified through the backward-in-time integration of the adjoint equation, leading to phase shifts and

local distortions in the adjoint field. Consequently, although the CAE performs well in reconstructing the primal dynamics, it underperforms when the reconstructed field is used to drive the adjoint computation.

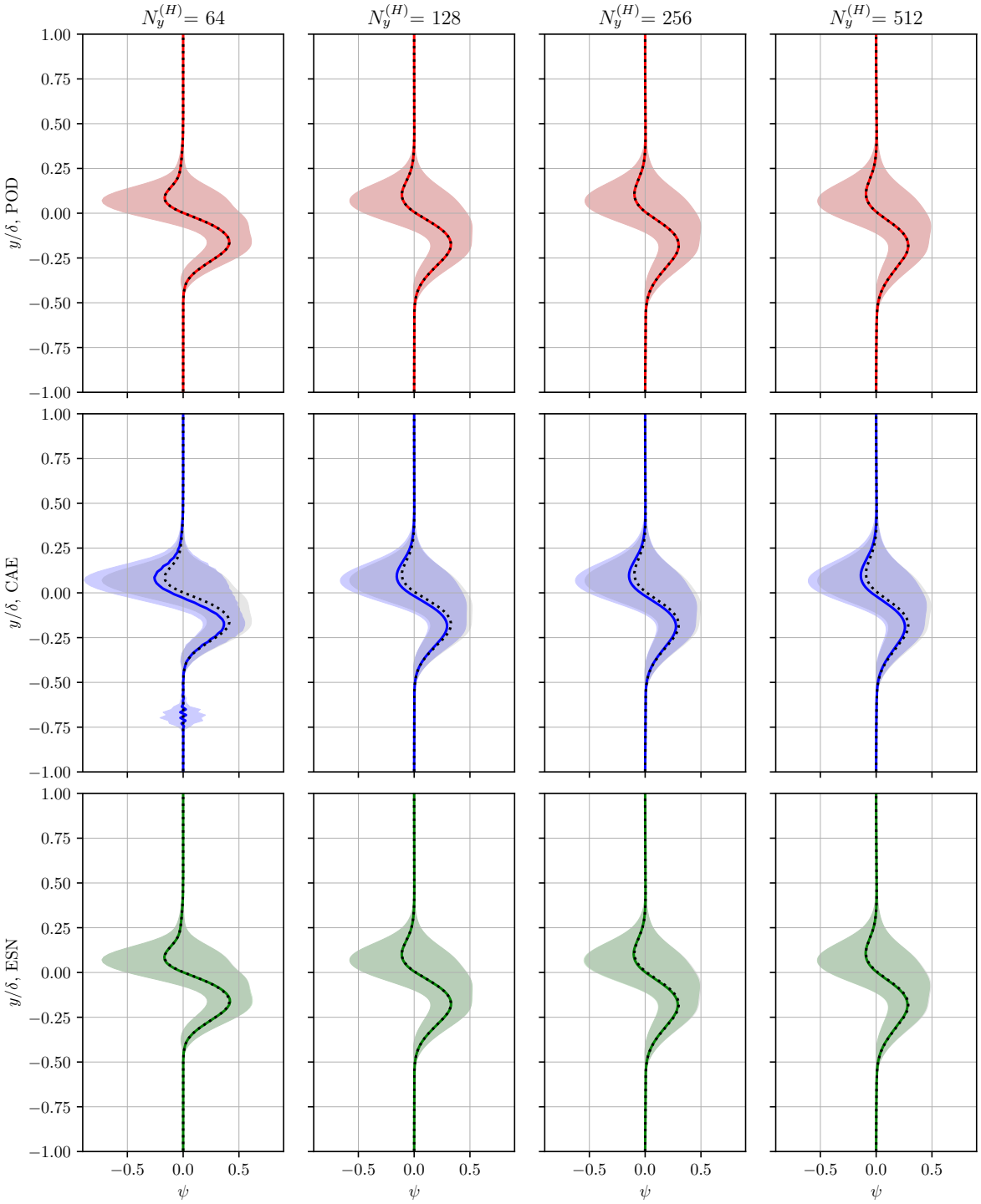


Figure 7.26: Computed 1D DNS-forced Burgers' adjoint solution $\tilde{\psi}$ using the reconstructed fine space solution $\tilde{v}_h$ for all refinements $N_y^{(H)}$ using POD (top), CAE (middle), and ESN (bottom). The shaded regions are the actual solution ψ, with the dashed black lines indicating their temporal mean.

The ESN-based adjoint solution exhibits nearly perfect agreement with the reference results in both the temporal mean and variance across all refinement levels. This consistency demonstrates the exceptional capability of the reservoir dynamics in capturing the temporal coherence and nonlinear depen-

dencies of the adjoint field. Unlike CAE, the ESN preserves the adjoint, owing to its intrinsic recurrent structure and memory effect, which effectively reconstruct long-term temporal correlations without over-smoothing. As a result, the ESN successfully reproduces even subtle fluctuations in the adjoint field while maintaining overall stability and physical consistency.

Figure 7.27 reports the test MSE of the computed adjoint solutions $\tilde{\psi}$ obtained using the surrogate-based reconstructed fine primal fields $\tilde{v}_h$. The comparison across refinement levels $N_y^{(H)}$ reveals distinct trends among the three surrogate models. The POD-based adjoint consistently achieves the lowest errors, exhibiting a rapid decrease in MSE with refinement and reaching values below 10^{-6} for the finest grid. This result reflects the strong spatial smoothness of the POD reconstruction, which effectively preserves gradient information required by the adjoint solver.

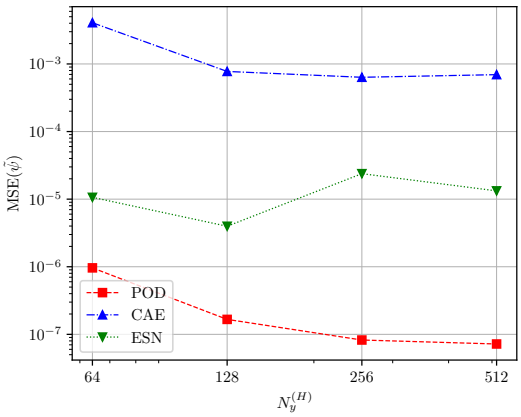


Figure 7.27: Test MSE of the computed 1D DNS-forced Burgers' adjoint solution $\tilde{\psi}$ using the reconstructed fine space solution $\tilde{v}_h$ for all refinements $N_y^{(H)}$ using POD, CAE, and ESN.

The CAE displays the highest MSE values across all refinements, with limited improvement as resolution increases. This degradation is attributed to the sensitivity of the adjoint solver to local inaccuracies in the spatial derivatives of the reconstructed field. Although the CAE achieved excellent reconstruction accuracy in the primal field, the small-scale distortions introduced during the encoding–decoding process become amplified in the backward integration of the adjoint equation, leading to larger deviations.

In contrast, the ESN-based adjoint solution maintains both stability and accuracy across all resolutions, with errors consistently within the same order of magnitude and significantly lower than those of the CAE. This robustness can be attributed to the inherent temporal coherence enforced by the reservoir dynamics, which preserves the derivative quality of the reconstructed primal solution.

To evaluate the consistency of the surrogate-based adjoint computations under equal compression efficiency, Figure 7.28 compares the reconstructed adjoint solutions $\tilde{\psi}$ obtained using CAE and ESN with those computed from POD fields compressed to the same CRs achieved by each model. This comparison highlights how surrogate-induced differences in the fine primal reconstructions propagate through the adjoint operator and affect the resulting sensitivity fields.

When the POD was forced to match the CRs achieved by the CAE, its performance remained remarkably stable. The reconstructed adjoint solutions captured both the temporal mean and the variance distribution with high fidelity relative to the FVM reference, even outperforming the CAE. In particular, the CAE exhibited localized deviations and small artifacts, especially at coarser resolutions, which were inherited from the instabilities observed in its reconstructed fine primal fields. These artifacts slightly distorted the adjoint variance profile, whereas the POD maintained a smooth and physically consistent response throughout the domain.

In contrast, when the POD was constrained to match the much higher CRs achieved by the ESN, the quality of its reconstruction deteriorated significantly. The deviation in the fine primal field propagated through the adjoint computation, yielding an adjoint solution with a distinctly altered mean and variance profile compared to the reference. The overall structure of the sensitivity field was lost, and

the distribution differed not only in value but also in spatial shape. Meanwhile, the ESN preserved the dominant adjoint features with notable accuracy despite operating at extremely high compression levels, demonstrating the reservoir's ability to retain essential temporal and spatial correlations even under aggressive data reduction.

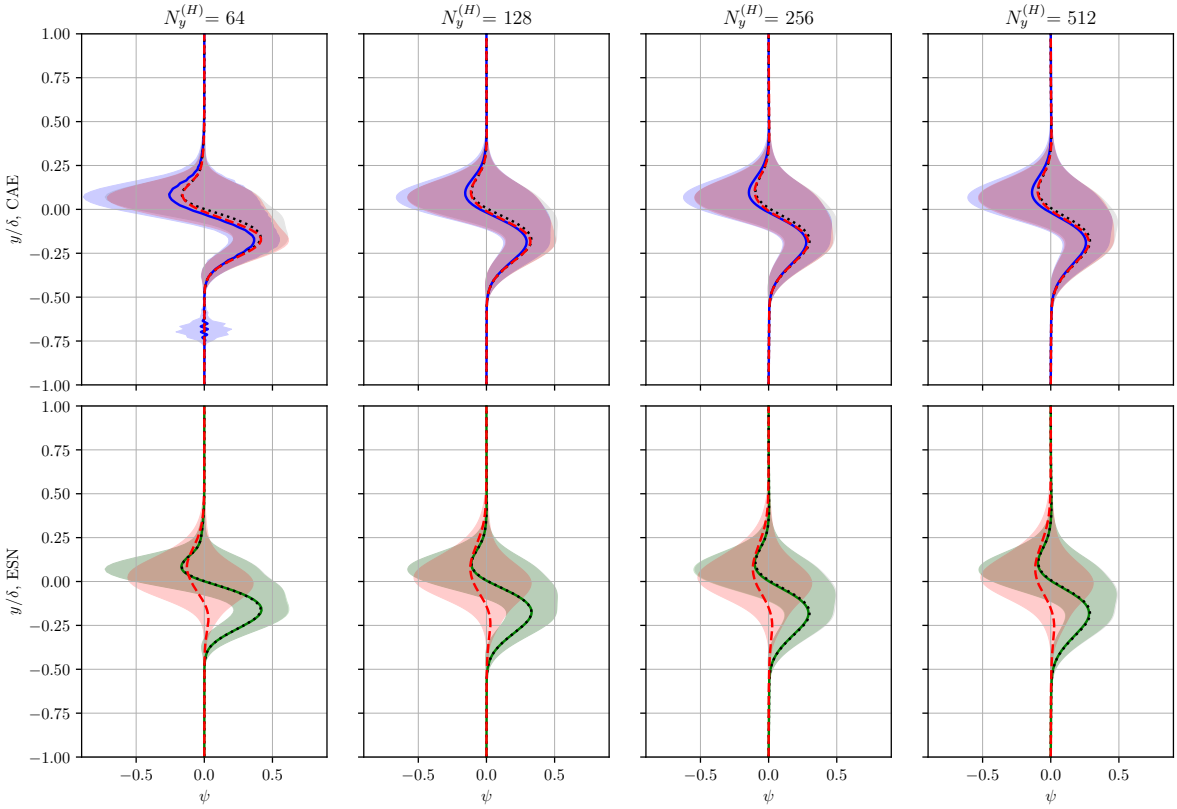


Figure 7.28: Comparison of reconstructed adjoint solutions $\tilde{\psi}$ for the 1D DNS-forced Burgers' problem obtained using CAE (top) and ESN (bottom), benchmarked against POD reconstructions performed at the same compression ratios (CRs) achieved by each model. The shaded regions represent the reference adjoint field, while the dashed red lines indicate the POD results, solid blue and green lines denote CAE and ESN, respectively, and the shaded areas show temporal variability.

Overall, these results confirm that while POD remains the most accurate in absolute error terms, the ESN offers the best trade-off between accuracy, stability, and computational efficiency. The CAE, despite its strong performance in reconstructing the primal field, demonstrates that high forward accuracy alone is insufficient to guarantee reliable adjoint computations. The adjoint solution is inherently sensitive to the quality of the spatial derivatives of the reconstructed dynamics. Consequently, ensuring derivative consistency and temporal coherence in the surrogate reconstruction is essential for achieving accurate adjoint sensitivities in error estimation contexts.

7.6. Adjoint-Based Error Localization

The spatial distribution of the time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D DNS-forced Burgers' problem is shown in Figure 7.29. At the coarsest resolution ($N_y^{(H)} = 64$), the error indicator displays broad peaks, indicating that the mesh is insufficient to resolve the localized gradients and intermittency introduced by the forcing term. As the grid is refined, the overall magnitude of $\bar{\epsilon}$ decreases by nearly two orders of magnitude, and the profiles become smoother and more concentrated around the central region ($y/\delta \approx 0$), where the adjoint fields exhibit the strongest activity. This progressive reduction highlights the consistency of the adjoint-based estimator even under dynamically rich conditions.

The time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D DNS-forced Burgers' problem is compared across *Scenarios 1–3* in Figures 7.30–7.32, with the baseline results shown in gray for reference.

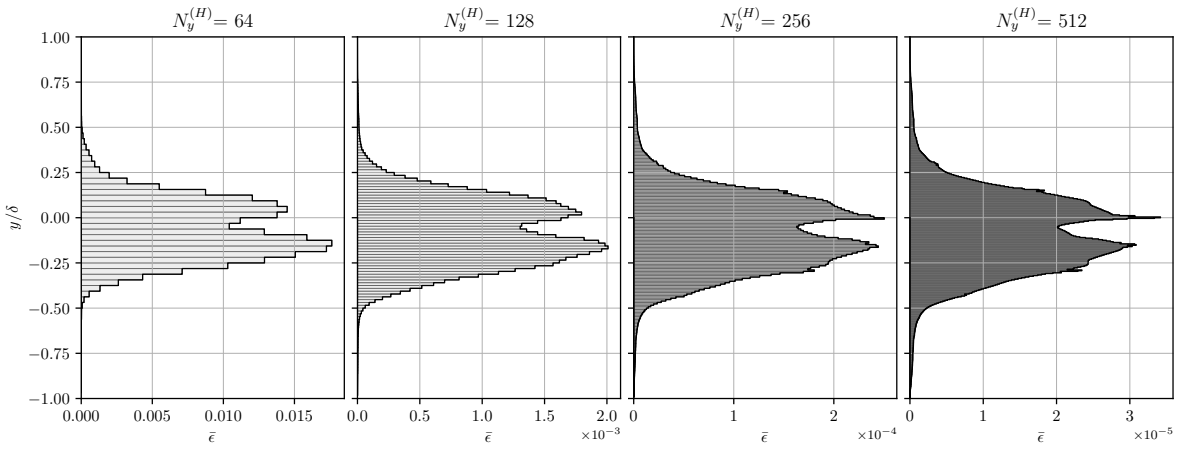


Figure 7.29: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D DNS-forced Burgers' problem across all refinement levels $N_y^{(H)}$.

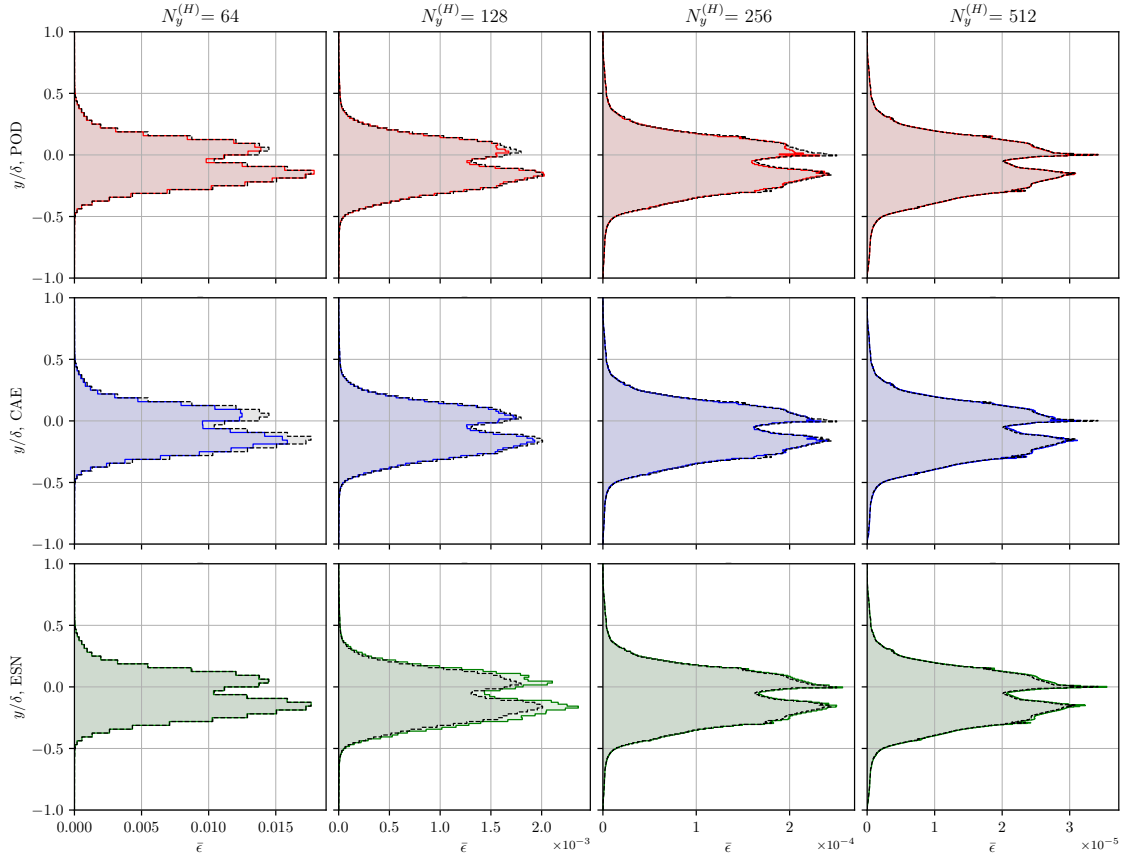


Figure 7.30: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D DNS-forced Burgers' problem at all refinement levels $N_y^{(H)}$, computed using surrogate models POD (top), CAE (middle), and ESN (bottom) under *Scenario 1*.

In *Scenario 1*, where only the injected residual field is reconstructed from surrogate models, all methods reproduce the overall spatial structure of the baseline error indicator with excellent agreement. Minor discrepancies are observed for the POD model, particularly near the centerline at coarse resolutions, but the shape and magnitude remain consistent with the reference. The CAE exhibits slightly higher deviations at the coarsest mesh, while the ESN shows a small offset at the second refinement level ($N_y^{(h)} = 128$); however, at finer resolutions, it converges to the baseline almost perfectly. These results indicate that residual reconstruction alone does not introduce significant distortion in the adjoint-based error estimation process.

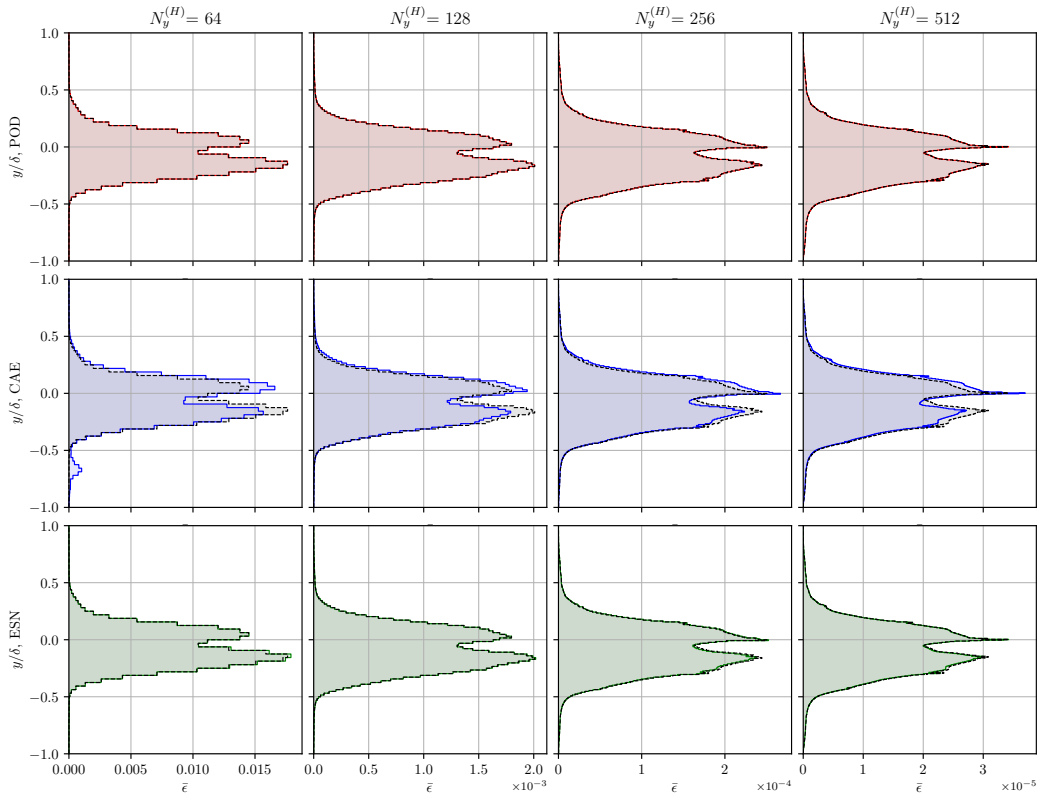


Figure 7.31: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D DNS-forced Burgers' problem at all refinement levels $N_y^{(H)}$, computed using surrogate models POD (top), CAE (middle), and ESN (bottom) under *Scenario 2*.

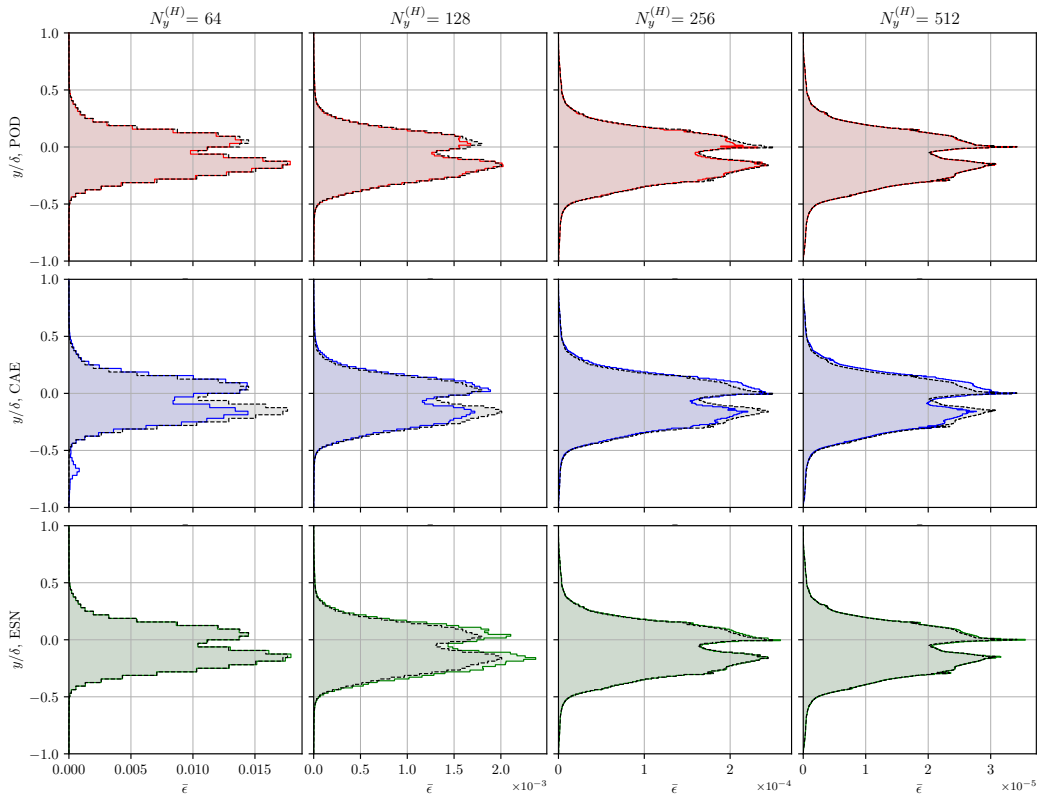


Figure 7.32: Time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D DNS-forced Burgers' problem at all refinement levels $N_y^{(H)}$, computed using surrogate models POD (top), CAE (middle), and ESN (bottom) under *Scenario 3*.

In *Scenario 2*, where the reconstructed fine primal solution $\tilde{v}_h$ is used to compute the adjoint field, both POD and ESN reproduce the temporal mean and spatial distribution of the error indicator with remarkable accuracy across all refinements. The CAE, on the other hand, displays clear deviations, most prominently at coarse resolutions, arising from the instability and localized artifacts in its computed adjoint solution. These artifacts, already observed in the CAE adjoint fields, propagate into the error indicator, particularly near the centerline where the adjoint sensitivity peaks. This reinforces the notion that even minor inconsistencies in the surrogate primal field can be amplified during adjoint computation.

In *Scenario 3*, which combines both surrogate-based residual and fine primal reconstructions, the cumulative effects of reconstruction become visible. The POD maintains the closest agreement with the baseline, while the CAE and ESN exhibit larger deviations in both magnitude and profile shape. Although the ESN remains relatively stable, the interaction between surrogate residuals and reconstructed primal dynamics introduces a mild broadening of the indicator distribution, reflecting accumulated compression errors. The CAE deviations persist, consistent with its less stable adjoint reconstructions.

To further assess the influence of surrogate compression on the adjoint-based error estimation process, Figure 7.33 compares the spatial distribution of the time-averaged error indicator $\bar{\epsilon}$ obtained using CAE and ESN against that of POD reconstructions performed at the same CRs. This analysis corresponds to *Scenario 3*, in which both the injected residuals and fine primal fields were simultaneously compressed and reconstructed prior to adjoint computation.

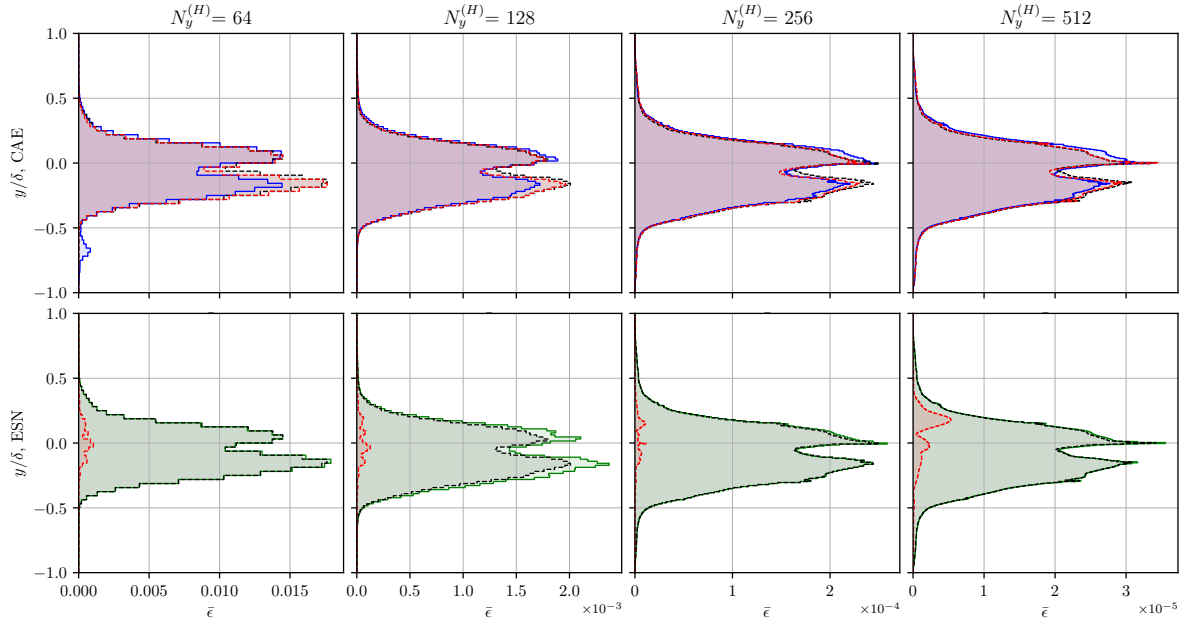


Figure 7.33: Comparison of the time-averaged adjoint-based error indicator $\bar{\epsilon}$ per cell for the 1D DNS-forced Burgers' problem under *Scenario 3*, where both the injected residuals and fine primal solutions were reconstructed. The CAE (top) and ESN (bottom) results are compared against POD reconstructions performed at identical compression ratios (CRs) achieved by each model. The black dashed lines indicate the FVM reference.

When the POD is forced to match the CRs achieved by the CAE, it delivers a slightly more accurate reproduction of the reference error indicator, both in magnitude and spatial profile. The CAE results remain close to those of the POD, with only minor deviations near the channel centerline where the sensitivity peaks. These differences are not substantial, confirming that both linear and nonlinear surrogates are capable of reproducing consistent adjoint-based error patterns at moderate compression levels.

However, when the POD is constrained to match the considerably higher CRs achieved by the ESN, the discrepancy becomes pronounced. The resulting POD-based error indicator diverges noticeably from the reference in both amplitude and shape, losing the characteristics and localized gradients that define the true error structure. The ESN, despite its strong performance in reconstructing the residual

and adjoint fields individually, fails to maintain the correct spatial distribution of $\bar{\epsilon}$ in this scenario. The high compression amplifies the deviations introduced in both reconstructed fields, resulting in a physically inconsistent error indicator.

7.7. Adjoint-Based Error Estimation

The comparison of adjoint-based error estimation for the 1D DNS-forced Burgers' problem across *Scenarios 1–3* is presented in Figure 7.34. The left panels display the time-averaged QoI $\bar{J}$, including the exact value $\bar{J}_{\text{DNS}}$, the numerical QoI $\bar{J}_H$, and the corrected QoIs obtained using each surrogate model. The middle panels show the adjoint-based error estimates $|\delta\bar{J}|$, while the right panels illustrate the deviation from the true error, defined as $e(\delta\bar{J}) = |\delta\bar{J} - \Delta\bar{J}|$.

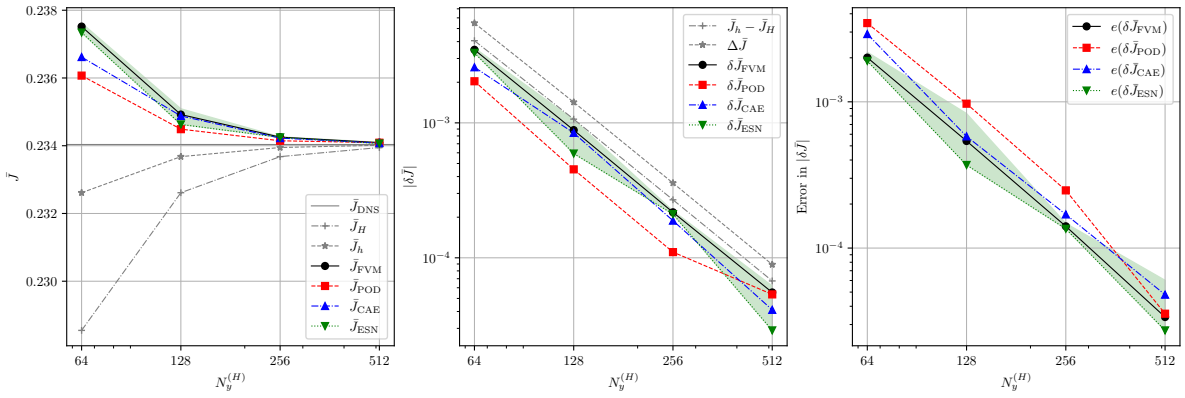
In all scenarios, the baseline FVM estimates ($\delta\bar{J}_{\text{FVM}}$) exhibit close agreement with the true error $\Delta\bar{J}$ across all spatial refinements, confirming the accuracy and consistency of the adjoint-based formulation for the DNS case. The surrogate-based results maintain this behavior with varying degrees of fidelity depending on the reconstruction configuration.

In *Scenario 1*, where only the injected residuals were compressed and reconstructed, all surrogate models, POD, CAE, and ESN, yield corrected QoIs closely matching the baseline. The ESN demonstrates particularly strong performance, with $|\delta\bar{J}_{\text{ESN}}|$ and $e(\delta\bar{J}_{\text{ESN}})$ aligning almost perfectly with the FVM reference, confirming that the recurrent architecture effectively preserves the temporal coherence of the residuals. The CAE also performs comparably to the FVM across all refinements, achieving the closest agreement. In contrast, the POD shows a small but consistent offset in the estimated error relative to the baseline, which diminishes with refinement and becomes nearly indistinguishable from the FVM at the finest grid resolution.

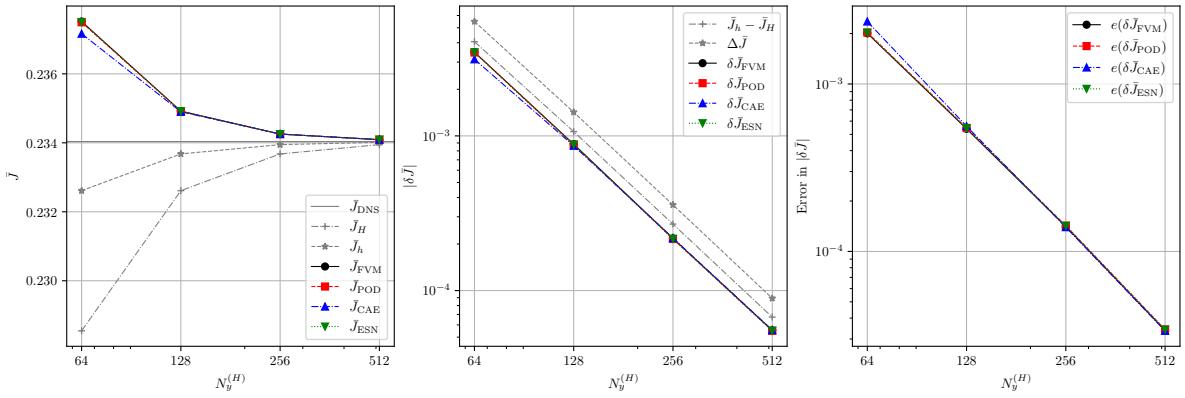
In *Scenario 2*, where the reconstructed fine primal solution $\tilde{v}_h$ was used to compute the adjoint field, the POD and ESN again produce error estimates in close alignment with the baseline, validating their ability to reconstruct temporally consistent and derivative-accurate primal dynamics. The CAE, however, shows noticeable discrepancies at lower resolutions, both in the corrected QoI and in $e(\delta\bar{J})$. These deviations are directly linked to the instabilities observed in the CAE-based adjoint solutions, where local artifacts in the reconstructed primal field propagate into the adjoint sensitivity and error estimation process.

In *Scenario 3*, where both injected residuals and the fine primal were reconstructed, the results are practically indistinguishable from **Scenario 1** across all refinements. This indicates that the error is dominated by the residual reconstruction; replacing the fine primal with a surrogate has a negligible effect on the adjoint-based correction. POD and ESN are visually coincident with the Scenario 1 and FVM curves for $\bar{J}$, $|\delta\bar{J}|$, and $e(\delta\bar{J})$. The only noticeable departure occurs for the coarsest CAE case, where a small bias appears in both $|\delta\bar{J}_{\text{CAE}}|$ and $e(\delta\bar{J}_{\text{CAE}})$. This deviation matches the artifact seen in the CAE-based primal reconstruction and its adjoint, and it vanishes with mesh refinement.

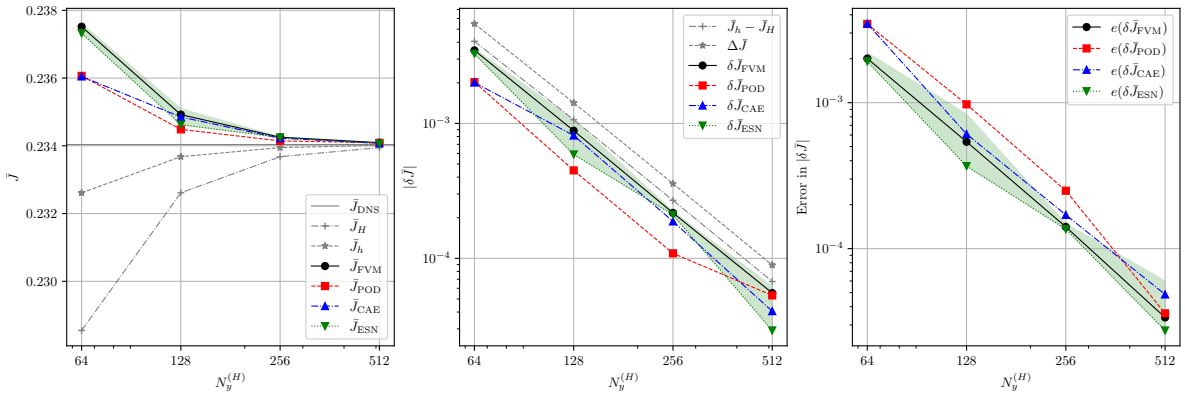
Overall, the results of the adjoint-based error indicators and error estimations confirm the robustness and reliability of the proposed surrogate-assisted framework across the examined scenarios. The adjoint-based error indicators consistently captured the main spatial features of the discretization error and demonstrated the expected convergence behavior with mesh refinement, validating the consistency of the dual-weighted formulation. Among the surrogate strategies, the POD and ESN reconstructions preserved the accuracy and spatial localization of the baseline FVM estimates, while the CAE exhibited minor deviations at coarse resolutions, primarily linked to local artifacts in the reconstructed primal and adjoint fields. The comparison of error estimations further revealed that the primary source of discrepancy stems from the residual reconstruction rather than the surrogate primal field, as Scenarios 1 and 3 yielded nearly identical outcomes. These findings emphasize that accurate residual representation is the most critical component for adjoint-based error estimation, and that properly regularized and temporally coherent surrogates, particularly the ESN, can deliver stable and precise adjoint corrections with minimal computational overhead.



(a) *Scenario 1*



(b) *Scenario 2*



(c) *Scenario 3*

Figure 7.34: Comparison of adjoint-based error estimation for the 1D DNS-forced Burgers' problem across *Scenarios 1–3*. Shown are (left) the time-averaged QoI $\bar{J}$, including the exact value $\bar{J}_{\mathrm{DNS}}$, the numerical solution $\bar{J}_H$, and corrected values obtained using FVM, POD, CAE, and ESN; (middle) the adjoint-based error estimates $|\delta\bar{J}|$; and (right) the error in the error estimate $e(\delta\bar{J})$.

7.8. Summary

The DNS-forced Burgers' case extended the framework to a nonlinear and temporally evolving configuration derived from a 1D wall-normal velocity signal extracted from a DNS of TCF at $Re_\tau = 180$. This setup introduced intermittent dynamics and strong gradient variations, serving as a representative test for data with turbulence-like complexity. A preliminary grid refinement study confirmed the second-order accuracy of the solver, with both the RMSE and QoI errors decaying as $\mathcal{O}(h^2)$ before saturating due to temporal discretization limits. The mean velocity and Reynolds stress profiles were consistent with the reference DNS, verifying the accuracy of the numerical scheme and the physical fidelity of the reconstructed forcing.

The injected residuals obtained from the DNS-forced primal fields revealed strong spatial intermittency and temporal variability, dominated by alternating positive and negative values near the domain boundaries. The mean values remained close to zero across all resolutions, while the standard deviation decreased monotonically with refinement by more than two orders of magnitude. The residual variance distributions exposed sharper and higher wall peaks with grid refinement, reflecting the emergence of higher-order spatial modes and the accurate resolution of localized gradients. These observations confirmed the consistency and convergence of the projection procedure under unsteady forcing.

The first surrogate modeling stage addressed the compression of the injected residuals. Compared to the smooth MMS case, the residuals here exhibited broadband spectra, preventing a low-rank representation. The POD required a significantly larger number of modes to capture 80–90% of the total energy, while the CAE demanded higher latent dimensions ($d_{\mathcal{Z}} = 57$ to 150) to achieve accurate reconstructions. The ESN, optimized via BO over 28 ensemble realizations, achieved validation errors on the order of $\mathcal{O}(10^{-8})$ while maintaining stability across refinements. Despite its higher optimization cost, the ESN consistently outperformed the other methods in terms of reconstruction accuracy and compression efficiency, whereas POD and CAE provided stable but less compact representations.

In the second surrogate modeling stage, the fine primal solution was compressed to evaluate the generalization of the framework to physically meaningful fields. The POD captured 99% of the energy with approximately 43–47 modes, while the CAE produced smooth and accurate reconstructions with rapidly improving accuracy at higher resolutions. The ESN demonstrated robust temporal reconstruction with consistent test errors across refinements, achieving compression ratios exceeding two orders of magnitude. These results demonstrated that, despite the increased flow complexity, all surrogate methods maintained convergence and reconstruction stability, with ESN achieving the best balance between accuracy and compression.

Adjoint-based error localization and estimation were then performed for the defined scenarios. The adjoint fields exhibited asymmetric structures with strong sensitivities concentrated near the channel centerline. In *Scenario 1* (residual-only compression), all surrogate-based adjoint fields reproduced the reference error indicators nearly identically. In *Scenario 2* (primal compression), POD and ESN maintained excellent alignment with the baseline, whereas CAE showed mild local deviations due to small-scale reconstruction artifacts. *Scenario 3* (combined compression) amplified these differences, yet POD and ESN still captured the global sensitivity distribution accurately. The time-averaged error indicator $\bar{\epsilon}$ decreased by nearly two orders of magnitude with refinement, and the corrected QoIs $\bar{J}_{\mathrm{POD}}$ and $\bar{J}_{\mathrm{CAE}}$ closely matched the reference values, with $\delta\bar{J}$ following the true error trend.

Overall, the DNS-forced Burgers' case demonstrated the robustness and scalability of the proposed compression and reconstruction framework under unsteady and nonlinear conditions. The POD provided a reliable linear baseline, the CAE achieved accurate spatial reconstructions at moderate latent dimensions, and the ESN offered the best trade-off between compression ratio, accuracy, and sensitivity preservation. These results confirmed the framework's capability to retain adjoint consistency and predictive reliability even for turbulence-inspired, dynamically rich data.

8

Conclusion

This chapter presents the conclusions drawn from the obtained results and provides the answers to the defined research questions in Chapter 2.

This thesis explored how the practicality of adjoint-based error estimation for unsteady turbulent flows can be improved through direct compression and reconstruction of injected residuals, without evaluating them via surrogate primal solutions. The research proposed, implemented, and evaluated a unified framework that integrates deep-learning-based surrogate models, namely the Convolutional AutoEncoder (CAE) and the Echo State Network (ESN), to alleviate the prohibitive storage requirements that have historically limited the practical application of unsteady adjoint methods. An offline version of Proper Orthogonal Decomposition (POD) was used as a benchmark technique for comparison. Two test cases were used to verify and assess the methodology: a smooth manufactured-solution (MMS) one-dimensional unsteady viscous Burgers' problem, and a rougher, more chaotic DNS-driven Burgers' problem forced by Turbulent Channel Flow (TCF) data.

Across all scenarios, the results demonstrated that the direct compression of injected residuals preserves the accuracy of the adjoint-based error estimation. The obtained Quantities of Interest (QoIs) and error indicators remain nearly identical to those from the fully resolved, uncompressed reference solutions, even at compression ratios exceeding two orders of magnitude. Among the considered surrogates, the ESN consistently yielded the most faithful reconstructions of temporally evolving residuals, preserving both local sensitivity structures and global output-error estimates. The CAE provided competitive spatial reconstruction quality, though artifacts at low spatial resolutions were observed. The POD exhibited limitations in capturing nonlinear features of residual dynamics. Overall, the research establishes that the proposed strategy can significantly reduce storage costs in unsteady adjoint-based error estimation without compromising accuracy, marking a practical step toward scalable adaptive simulations for large-eddy and turbulent-flow analyses.

Looking back to the defined main research question:

What is the impact of AI-based compression and reconstruction of the injected residuals on the accuracy of unsteady adjoint-based error estimates?

The results confirm that injected residuals can be directly compressed and reconstructed up to compression ratios of $\mathcal{O}(10^2)$ without significant degradation in adjoint-based error estimation or localization accuracy. Among the evaluated AI-based frameworks, the ESN proved most effective due to its capacity to capture long-term temporal correlations with a compact set of parameters. The CAE, while robust in representing nonlinear spatial features, required higher-resolution feature maps to avoid localized reconstruction noise. In contrast, the linear POD method served as a lightweight baseline but lacked the expressive capability of the AI-based models, particularly at finer resolutions.

Within the h-adaptation strategy employing a refinement factor of two, directly compressing the injected residuals offers a more representative measure of the deviation between successive refinement levels than evaluating residuals indirectly from a surrogate primal. Although this approach requires

storing approximately twice the data, it provides a clearer and more physically consistent depiction of discretization errors across mesh hierarchies, enhancing the reliability of adjoint-based error estimation in unsteady flow simulations.

RQ1: What are the fundamental differences between the primal solution and the injected residual in terms of their data characteristics in the context of compression and reconstruction?

The primal solution exhibits large-scale, spatially correlated flow structures, whereas the injected residual field is characterized by short correlation lengths and strong intermittency, with localized peaks corresponding to regions of high numerical imbalance. Unlike the primal field, whose spatial complexity tends to converge with grid refinement, the residual field becomes increasingly complex as higher-order spatial modes emerge and finer-scale gradients are resolved. This growing complexity reflects the localized nature of discretization errors, which are amplified near regions of strong nonlinearity or boundary effects, highlighting the need for compression and reconstruction strategies capable of capturing these fine-scale variations without loss of stability or accuracy.

RQ2: How do spatial and temporal deep learning architectures compare in their ability to learn turbulent flow dependencies relevant to the accurate reconstruction of injected residuals?

The CAE and ESN architectures were found to be complementary in nature. The CAE effectively captures nonlinear spatial features through its convolutional encoding layers, enabling the identification of coherent structures within the flow fields. In contrast, the ESN efficiently models temporal dependencies via its recurrent reservoir dynamics, making it particularly well suited for the unsteady and intermittent behavior of the injected residuals. Although the benchmark POD approach is computationally efficient at lower refinement levels, its cost increases rapidly with grid resolution, eventually exceeding that of the CAE and ESN. Overall, the ESN demonstrated superior capability in representing the temporal correlations and variabilities present in the injected residual data, indicating that temporal architectures are generally more appropriate for residual reconstruction than their purely spatial counterparts.

RQ3: How should compression and reconstruction performance be evaluated against the benchmark approach in terms of accuracy, compression efficiency, and computational cost?

Compression performance was assessed using three main metrics: the achieved compression ratio, the reconstruction fidelity measured by the Mean Squared Error (MSE), and the computational cost of preparation and reconstruction. Across these measures, the ESN achieved the highest compression ratio while maintaining sub-percent deviations in $\delta\bar{J}$ relative to the baseline framework; however, it required the highest preparation and reconstruction time compared with the other candidates. The CAE achieved comparable reconstruction quality for moderate compression levels, whereas the POD trailed behind due to its linear basis limitation.

RQ4: Between compressed injected residuals and adjoint fields computed from compressed primals, which contributes more significantly to the overall error in output error estimation?

Comparative scenario analyses revealed that errors arising from residual compression dominate the deviation in error estimations, whereas the error in adjoints computed from compressed primals contribute only marginally once residual fidelity is ensured. Thus, improving residual reconstruction accuracy yields the greatest benefit for maintaining estimator reliability and reducing total output-error deviation.

RQ5: Can the developed framework for injected residual compression and reconstruction be effectively extended to the compression of primal fields, and under what conditions does this generalization remain valid?

The framework was also tested for compressing primal fields; however, when applied using the CAE, accuracy degraded more rapidly with increasing compression, as the loss of small-scale flow features during reconstruction affected the stability of the adjoint solution and led to visible artifacts in the adjoint fields. In particular, reconstructed primals obtained from the CAE occasionally induced localized instabilities in the adjoint computations, highlighting its sensitivity to spatial reconstruction errors. In contrast, the ESN exhibited greater robustness, as its temporal encoding better preserved the dynamic evolution of the flow, resulting in smoother adjoint behavior and more consistent error estimations.

By demonstrating that directly compressed residuals can sustain accurate adjoint-based error estimates, this research bridges data-driven modeling and goal-oriented CFD adaptation. It provides a practical pathway to deploy adjoint frameworks in high-fidelity unsteady simulations, where storage has long been the primary barrier. The developed methodology establishes the foundation for future extensions toward multidimensional Large-Eddy Simulations (LES), real-time Adaptive Mesh Refinement (AMR), and hybrid physics-informed compression networks.

9

Future Work & Recommendations

While this thesis demonstrated that direct compression of injected residuals enables accurate adjoint-based error estimation with storage reductions exceeding two orders of magnitude, several promising research avenues remain open for further development and validation of the proposed framework. These directions span improvements to model architecture, optimization strategies, adjoint stability, and the extension of the framework to more complex flow configurations and physical models.

Extension to Multidimensional and Turbulent Flow Configurations The present work focused on one-dimensional unsteady viscous Burgers' equations to verify and isolate the fundamental behavior of residual compression and reconstruction. A natural extension is to apply the same framework to higher-dimensional Navier–Stokes (NS) problems, including two- and three-dimensional configurations with complex boundary conditions. In particular, testing the framework within canonical Large-Eddy Simulations (LES) would assess its robustness in representing anisotropic and multiscale turbulent structures. Integrating the residual compression pipeline into established CFD solvers such as `OpenFOAM` would also enable validation on realistic aerodynamic test cases and facilitate the adoption of the method within adaptive mesh refinement (AMR) workflows.

Hybrid and Physics-Constrained Surrogate Architectures Although the Convolutional Autoencoder (CAE) and Echo State Network (ESN) were studied independently, combining their complementary strengths represents a promising path forward. A hybrid CAE–ESN or POD–ESN architecture could exploit the CAE's nonlinear spatial encoding together with the ESN's ability to learn temporal dependencies, thereby enabling joint spatio-temporal compression at higher fidelity. Moreover, incorporating *physics-constrained* or *physics-informed* regularization terms within these networks could ensure consistency between reconstructed fields and governing conservation laws, reducing the risk of nonphysical artifacts during reconstruction. Physics-constrained loss formulations or energy-preserving autoencoders may further enhance the generalization of the surrogate models across different Reynolds numbers and flow regimes.

Optimization Framework Enhancement The Bayesian optimization framework adopted for the ESN provided high accuracy but was computationally demanding due to the large number of hyperparameter evaluations and repeated network realizations. Future work should explore strategies to accelerate this process, including:

- Multi-fidelity or transfer-learning–based Bayesian optimization, where information from lower-resolution or previously optimized cases informs new searches.
- Surrogate-assisted optimization using Gaussian Processes or lightweight meta-models of the ESN loss surface to predict optimal regions with fewer evaluations.
- Automated differentiation or gradient-based optimization approaches for differentiable ESN variants, potentially reducing total optimization time by an order of magnitude.

These improvements would make ESN-based compression feasible for large datasets typical of high-fidelity unsteady CFD simulations.

Adjoint Stability and Coarse-Mesh Consistency One of the persistent challenges in adjoint-based methods, particularly noted in this work, is numerical instability when solving the adjoint equations on coarse meshes or under strong temporal coarsening. Since Adaptive Mesh Refinement (AMR) efficiency largely depends on the feasibility of stable adjoint computations at lower resolutions, future efforts should target adjoint stabilization mechanisms. Possible approaches include:

- Incorporating residual-based stabilization terms or selective temporal filtering to damp high-frequency adjoint oscillations.
- Developing consistent temporal interpolation strategies for backward-in-time adjoint propagation under coarsened snapshots.
- Exploring dual-consistency enforcement and entropy-stable flux formulations that have shown promise in maintaining smooth adjoint fields in nonlinear regimes.

Improving adjoint stability would directly extend the applicability of the proposed compression approach to coarser adaptive simulations, where its benefits are most pronounced.

Adaptive and Online Compression Another potential direction lies in *adaptive compression*, where the compression ratio dynamically adjusts according to local flow activity or residual magnitude. In such a framework, higher-fidelity storage would be retained only in dynamically active regions or during transient phases, while smoother regions would be stored with more aggressive compression. Similarly, developing an *online compression* approach, where data are compressed concurrently with the primal computation, would eliminate the need for post-processing, enabling real-time integration of residual compression into unsteady CFD workflows.

Integration with Uncertainty Quantification and Surrogate Error Estimation The proposed framework can also serve as a foundation for uncertainty-aware adjoint-based analyses. By quantifying how compression-induced errors propagate through the adjoint-based error estimation process, a probabilistic bound on the error indicator could be established. This integration with Uncertainty Quantification (UQ) frameworks, using ensemble-based or Bayesian methods, could yield more robust error estimators for decision-making in adaptive simulations or optimization under uncertainty.

In summary, future research should focus on advancing three main fronts: (1) expanding the applicability of the framework to multidimensional turbulent and chaotic flows, (2) improving surrogate architectures and optimization efficiency, and (3) enhancing adjoint stability and physics consistency. Together, these developments would bring adjoint-based error estimation closer to real-time, scalable deployment in high-fidelity unsteady CFD simulations, bridging the gap between traditional numerical methods and modern machine learning–assisted computational modeling.

References

[1] Antony Jameson. "A perspective on computational algorithms for aerodynamic analysis and design". In: *Progress in Aerospace Sciences* 37.2 (2001), pp. 197–243.

[2] Antony Jameson and John Vassberg. "Computational fluid dynamics for aerodynamic design-Its current and future impact". In: *39th Aerospace Sciences Meeting and Exhibit*. 2001, p. 538.

[3] Mori Mani and Andrew J Dorgan. "A perspective on the state of aerospace computational fluid dynamics technology". In: *Annual Review of Fluid Mechanics* 55.1 (2023), pp. 431–457.

[4] Shahrouz K Aliabadi and Tayfun E Tezduyar. "Parallel fluid dynamics computations in aerospace applications". In: *International Journal for Numerical Methods in Fluids* 21.10 (1995), pp. 783–805.

[5] Fu Zhang Wang et al. "Recent advancements in fluid dynamics: drag reduction, lift generation, computational fluid dynamics, turbulence modelling, and multiphase flow". In: *Arabian Journal for Science and Engineering* 49.8 (2024), pp. 10237–10249.

[6] ZhiJian Wang. "A perspective on high-order methods in computational fluid dynamics". In: *Science China Physics, Mechanics & Astronomy* 59.1 (2016), p. 614701.

[7] Yang Zhiyin. "Large-eddy simulation: Past, present and the future". In: *Chinese Journal of Aeronautics* 28.1 (2015), pp. 11–24.

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

[9] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. "Turbulence modeling in the age of data". In: *Annual Review of Fluid Mechanics* 51.1 (2019), pp. 357–377.

[10] Peter Davidson. *Turbulence: an introduction for scientists and engineers*. Oxford university press, 2015.

[11] Andrey Nikolaevich Kolmogorov. "The local structure of turbulence in incompressible viscous fluid for very large Reynolds". In: *Numbers. In Dokl. Akad. Nauk SSSR* 30 (1941), p. 301.

[12] Hendrik Tennekes and John Leask Lumley. *A first course in turbulence*. MIT press, 1972.

[13] Parviz Moin and Krishnan Mahesh. "Direct numerical simulation: a tool in turbulence research". In: *Annual Review of Fluid Mechanics* 30.1 (1998), pp. 539–578.

[14] Victor Yakhot and Katepalli R Sreenivasan. "Anomalous scaling of structure functions and dynamic constraints on turbulence simulations". In: *Journal of Statistical Physics* 121 (2005), pp. 823–841.

[15] Pietro Catalano and Marcello Amato. "An evaluation of RANS turbulence modelling for aerodynamic applications". In: *Aerospace Science and Technology* 7.7 (2003), pp. 493–509.

[16] David Uystepruyst and Siniša Krajnović. "Numerical simulation of the transient aerodynamic phenomena induced by passing manoeuvres". In: *Journal of Wind Engineering and Industrial Aerodynamics* 114 (2013), pp. 62–71.

[17] In-Bok Lee et al. "The past, present and future of CFD for agro-environmental applications". In: *Computers and Electronics in Agriculture* 93 (2013), pp. 168–183.

[18] Freddie D Witherden and Antony Jameson. "Future directions in computational fluid dynamics". In: *23rd AIAA Computational Fluid Dynamics Conference*. 2017, p. 3791.

[19] Heng Xiao and Paola Cinnella. "Quantification of model uncertainty in RANS simulations: A review". In: *Progress in Aerospace Sciences* 108 (2019), pp. 1–31.

[20] JF Quaatz et al. "Large-eddy simulation of a pseudo-shock system in a Laval nozzle". In: *International Journal of Heat and Fluid Flow* 49 (2014), pp. 108–115.

[21] David Corson, Rajeev Jaiman, and Farzin Shakib. "Industrial application of RANS modelling: capabilities and needs". In: *International Journal of Computational Fluid Dynamics* 23.4 (2009), pp. 337–347.

[22] Xiaodong Li. "Towards Efficient Adjoint-Based Mesh Optimization for Predictive Large Eddy Simulation". Dissertation (TU Delft). PhD thesis. Delft University of Technology, 2023. DOI: 10.4233/uuid:3ec425ae-2055-42e2-b911-e00ccf035a16. URL: https://doi.org/10.4233/uuid:3ec425ae-2055-42e2-b911-e00ccf035a16.

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

[24] David W Levy et al. "Data summary from the first AIAA computational fluid dynamics drag prediction workshop". In: *Journal of Aircraft* 40.5 (2003), pp. 875–882.

[25] Joseph Morrison and Michael Hemsch. "Statistical analysis of CFD solutions from the third AIAA drag prediction workshop". In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. 2007, p. 254.

[26] Krzysztof J Fidkowski and David L Darmofal. "Review of output-based error estimation and mesh adaptation in computational fluid dynamics". In: *AIAA Journal* 49.4 (2011), pp. 673–694.

[27] Johan Larsson. "Grid-adaptation for chaotic multi-scale simulations as a verification-driven inverse problem". In: *2018 AIAA Aerospace Sciences Meeting*. 2018, p. 0371.

[28] Xiaodong Li, Steven Hulshoff, and Stefan Hickel. "Towards adjoint-based mesh refinement for Large Eddy Simulation using reduced-order primal solutions: Preliminary 1D Burgers study". In: *Computer Methods in Applied Mechanics and Engineering* 379 (2021), p. 113733. DOI: 10.1016/j.cma.2021.113733.

[29] Timothy J Baker. "Mesh adaptation strategies for problems in fluid dynamics". In: *Finite Elements in Analysis and Design* 25.3-4 (1997), pp. 243–273.

[30] Christopher Roy. "Strategies for driving mesh adaptation in CFD". In: *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. 2009, p. 1302.

[31] Richard P Dwight. "Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation". In: *Journal of Computational Physics* 227.5 (2008), pp. 2845–2863.

[32] Ralf Deiterding. "Construction and application of an AMR algorithm for distributed memory computers". In: *Adaptive Mesh Refinement-Theory and Applications: Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3–5, 2003*. Springer. 2005, pp. 361–372.

[33] Stephen B Pope. "Ten questions concerning the large-eddy simulation of turbulent flows". In: *New journal of Physics* 6.1 (2004), p. 35.

[34] Sanjeeb T Bose. *Explicitly filtered large-eddy simulation: with application to grid adaptation and wall modeling*. Stanford University, 2012.

[35] Siavash Toosi and Johan Larsson. "Anisotropic grid-adaptation in large eddy simulations". In: *Computers & Fluids* 156 (2017), pp. 146–161.

[36] William J Coirier and Kenneth G Powell. "Solution-adaptive Cartesian cell approach for viscous and inviscid flows". In: *AIAA Journal* 34.5 (1996), pp. 938–945.

[37] Pierre Benard et al. "Mesh adaptation for large-eddy simulations in complex geometries". In: *International Journal for Numerical Methods in Fluids* 81.12 (2016), pp. 719–740.

[38] I. Babuška and A. Miller. "Post-Processing Approach in the Finite Element Method – Part 3: a Posteriori Error Estimates and Adaptive Mesh Selection". In: *International Journal for Numerical Methods in Engineering* 20.12 (1984), pp. 2311–2324.

[39] João Colaço Romana. "Adjoint-Based Error Estimation for Unsteady Problems: Deep Learning Techniques for Surrogate Modelling". Public defense on 13th February 2025. MA thesis. Delft University of Technology, 2025. URL: http://repository.tudelft.nl/.

[40] Timothy J Barth and Herman Deconinck. *Error estimation and adaptive discretization methods in computational fluid dynamics*. Vol. 25. Springer Science & Business Media, 2013.

[41] Thomas P Hunter and Steven J Hulshoff. "SuperAdjoint: Super-resolution neural networks in adjoint-based error estimation". In: *Journal of Computational and Applied Mathematics* 442 (2024), p. 115722.

[42] Steven Michael Kast. "Methods for Optimal Output Prediction in Computational Fluid Dynamics." PhD thesis. University of Michigan, 2016.

[43] Xiaodong Li. "Towards efficient adjoint-based mesh optimization for predictive large eddy simulation". In: (2023).

[44] Krzysztof J Fidkowski and Yuxing Luo. "Output-based space–time mesh adaptation for the compressible Navier–Stokes equations". In: *Journal of Computational Physics* 230.14 (2011), pp. 5753–5773.

[45] Qiqi Wang, Parviz Moin, and Gianluca Iaccarino. "Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation". In: *SIAM Journal on Scientific Computing* 31.4 (2009), pp. 2549–2567.

[46] Qiqi Wang. "Forward and adjoint sensitivity computation of chaotic dynamical systems". In: *Journal of Computational Physics* 235 (2013), pp. 1–13.

[47] Julian Roth, Max Schröder, and Thomas Wick. "Neural network guided adjoint computations in dual weighted residual error estimation". In: *SN Applied Sciences* 4.2 (2022), p. 62.

[48] Krzysztof J Fidkowski. "Output-based error estimation and mesh adaptation for unsteady turbulent flow simulations". In: *Computer Methods in Applied Mechanics and Engineering* 399 (2022), p. 115322.

[49] Arnish Sitaram. "Output error estimation for unsteady flows using reconstructed solutions". PhD thesis. MSc Thesis. Delft University of Technology, 2023.

[50] Lei Shi and Zhi Jian Wang. "Adjoint-based error estimation and mesh adaptation for the correction procedure via reconstruction method". In: *Journal of Computational Physics* 295 (2015), pp. 261–284.

[51] Mohammadreza Momenifar et al. "Dimension reduced turbulent flow data from deep vector quantisers". In: *Journal of Turbulence* 23.4-5 (2022), pp. 232–264.

[52] Pranshu Pant et al. "Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations". In: *Physics of Fluids* 33.10 (2021).

[53] Takaaki Murata, Kai Fukami, and Koji Fukagata. "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics". In: *Journal of Fluid Mechanics* 882 (2020), A13.

[54] Alberto Olmo et al. "Physics-driven convolutional autoencoder approach for CFD data compressions". In: *arXiv preprint arXiv:2210.09262* (2022).

[55] Alberto Racca, Nguyen Anh Khoa Doan, and Luca Magri. "Predicting turbulent dynamics with the convolutional autoencoder echo state network". In: *Journal of Fluid Mechanics* 975 (2023), A2.

[56] Yi Li. *Automatic mesh adaptation using the continuous adjoint approach and the spectral difference method*. Stanford University, 2013.

[57] Ralf Hartmann and Paul Houston. "Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations". In: *Journal of Computational Physics* 183.2 (2002), pp. 508–532.

[58] David A Venditti and David L Darmofal. "Grid adaptation for functional outputs: application to two-dimensional inviscid flows". In: *Journal of Computational Physics* 176.1 (2002), pp. 40–69.

[59] David A Venditti and David L Darmofal. "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows". In: *Journal of Computational Physics* 187.1 (2003), pp. 22–46.

[60] David Venditti and David Darmofal. "A multilevel error estimation and grid adaptive strategy for improving the accuracy of integral outputs". In: *14th Computational Fluid Dynamics Conference*. 1999, p. 3292.

[61] David A Venditti and David L Darmofal. "Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow". In: *Journal of Computational Physics* 164.1 (2000), pp. 204–227.

[62] Michael Andrew Park. "Adjoint-based, three-dimensional error prediction and grid adaptation". In: *AIAA journal* 42.9 (2004), pp. 1854–1862.

[63] Marian Nemec and Michael Aftosmis. "Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes". In: *18th AIAA Computational Fluid Dynamics Conference*. 2007, p. 4187.

[64] Marian Nemec, Michael Aftosmis, and Mathias Wintzer. "Adjoint-based adaptive mesh refinement for complex geometries". In: *46th AIAA Aerospace Sciences Meeting and Exhibit*. 2008, p. 725.

[65] Krzysztof J Fidkowski. "Output-based space–time mesh optimization for unsteady flows using continuous-in-time adjoints". In: *Journal of Computational Physics* 341 (2017), pp. 258–277.

[66] Karthik Mani and Dimitri J Mavriplis. "Error estimation and adaptation for functional outputs in time-dependent flow problems". In: *Journal of Computational Physics* 229.2 (2010), pp. 415–440.

[67] Karthik Mani and Dimitri Mavriplis. "Discrete adjoint based time-step adaptation and error reduction in unsteady flow problems". In: *18th AIAA Computational Fluid Dynamics Conference*. 2007, p. 3944.

[68] Anca Belme, Alain Dervieux, and Frédéric Alauzet. "Time accurate anisotropic goal-oriented mesh adaptation for unsteady flows". In: *Journal of Computational Physics* 231.19 (2012), pp. 6323–6348.

[69] Kaihua Ding, Krzysztof J Fidkowski, and Philip L Roe. "Acceleration techniques for adjoint-based error estimation and mesh adaptation". In: *Eighth International Conference on Computational Fluid Dynamics ICCFD8-0249*. 2014.

[70] Kevin T Doetsch and Krzysztof Fidkowski. "Unsteady combined entropy and output-based adjoint approach for mesh refinement and error estimation". In: *AIAA Aviation 2019 Forum*. 2019, p. 2951.

[71] Hongyuan Jia and Hideki Kikumoto. "Source term estimation in complex urban environments based on Bayesian inference and unsteady adjoint equations simulated via large eddy simulation". In: *Building and Environment* 193 (2021), p. 107669.

[72] Yiping Lin, Hong Huang, and Xiaole Zhang. "Source term estimation of a time-varying source around a building based on Bayesian inference and unsteady adjoint equations". In: *Building and Environment* 267 (2025), p. 112251.

[73] Brian N Granzow, Assad A Oberai, and Mark S Shephard. "Adjoint-based error estimation and mesh adaptation for stabilized finite deformation elasticity". In: *Computer Methods in Applied Mechanics and Engineering* 337 (2018), pp. 263–280.

[74] Roland Becker and Rolf Rannacher. "An optimal control approach to a posteriori error estimation in finite element methods". In: *Acta Numerica* 10 (2001), pp. 1–102.

[75] Guodong Chen and Krzysztof Fidkowski. "Output-based error estimation and mesh adaptation using convolutional neural networks: Application to a scalar advection-diffusion problem". In: *AIAA Scitech 2020 Forum*. 2020, p. 1143.

[76] Ayan Chakraborty et al. "Multigoal-oriented dual-weighted-residual error estimation using pinns". In: *Machine Learning for Computational Science and Engineering* 1.1 (2025), pp. 1–18.

[77] William Conrad Tyson. "On numerical error estimation for the finite-volume method with an application to computational fluid dynamics". In: (2018).

[78] Vít Dolejší, Ondřej Bartoš, and Filip Roskovec. "Goal-oriented mesh adaptation method for nonlinear problems including algebraic errors". In: *Computers & Mathematics with Applications* 93 (2021), pp. 178–198.

[79] Yao Jiang and Siva Nadarajah. "Adjoint-based error estimation for grid adaptation for large eddy simulation". In: *Computers & Fluids* 236 (2022), p. 105295.

[80] Michael Andrew Park. "Anisotropic output-based adaptation with tetrahedral cut cells for compressible flows". PhD thesis. Massachusetts Institute of Technology, 2008.

[81] Kunihiko Taira et al. "Modal analysis of fluid flows: Applications and outlook". In: *AIAA journal* 58.3 (2020), pp. 998–1022.

[82] John Leask Lumley. "The structure of inhomogeneous turbulent flows". In: *Atmospheric Turbulence and Radio Wave Propagation* (1967), pp. 166–178.

[83] Zhao Wu et al. "Proper orthogonal decomposition and dynamic mode decomposition of jet in channel crossflow". In: *Nuclear Engineering and Design* 344 (2019), pp. 54–68.

[84] John L Lumey. *Stochastic tools in turbulence*. Elsevier, 2012.

[85] Peter J Schmid. "Dynamic mode decomposition of numerical and experimental data". In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28.

[86] Lawrence Sirovich. "Turbulence and the dynamics of coherent structures. II. Symmetries and transformations". In: *Quarterly of Applied mathematics* 45.3 (1987), pp. 573–582.

[87] Julien Weiss. "A tutorial on the proper orthogonal decomposition". In: *AIAA aviation 2019 forum*. 2019, p. 3333.

[88] James P Howard et al. "Data-driven modeling & scientific computation: methods for complex systems & big data". In: *Journal of Statistical Software* 67 (2015), pp. 1–3.

[89] Andrew Glaws, Ryan King, and Michael Sprague. "Deep learning for in situ data compression of large turbulent flow simulations". In: *Physical Review Fluids* 5.11 (2020), p. 114602.

[90] Clement Caron, Philippe Lauret, and Alain Bastide. "Machine Learning to speed up Computational Fluid Dynamics engineering simulations for built environments: A review". In: *Building and Environment* 267 (2025), p. 112229.

[91] Mario Lino et al. "Current and emerging deep-learning methods for the simulation of fluid dynamics". In: *Proceedings of the Royal Society A* 479.2275 (2023), p. 20230058.

[92] Luning Sun et al. "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data". In: *Computer Methods in Applied Mechanics and Engineering* 361 (2020), p. 112732.

[93] Yinhao Zhu et al. "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data". In: *Journal of Computational Physics* 394 (2019), pp. 56–81.

[94] Guodong Chen and Krzysztof J Fidkowski. "Output-based adaptive aerodynamic simulations using convolutional neural networks". In: *Computers & Fluids* 223 (2021), p. 104947.

[95] Kai Fukami, Taichi Nakamura, and Koji Fukagata. "Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data". In: *Physics of Fluids* 32.9 (2020).

[96] Jean Panaioti Jordanou et al. "Investigation of proper orthogonal decomposition for echo state networks". In: *Neurocomputing* 548 (2023), p. 126395.

[97] Mark A Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE journal* 37.2 (1991), pp. 233–243.

[98] Ian Goodfellow. *Deep learning*. Vol. 196. MIT press, 2016.

[99] Emmanuel Pintelas, Ioannis E Livieris, and Panagiotis E Pintelas. "A convolutional autoencoder topology for classification in high-dimensional noisy image datasets". In: *Sensors* 21.22 (2021), p. 7731.

[100] Shuangshuang Chen and Wei Guo. "Auto-encoders in deep learning—a review with new perspectives". In: *Mathematics* 11.8 (2023), p. 1777.

[101] Xia Wu et al. "Effects of hyperparameters on flow field reconstruction around a foil by convolutional neural networks". In: *Ocean Engineering* 247 (2022), p. 110650.

[102] Lanfei Zhang et al. "Surrogate Modeling of Hydrogen-Enriched Combustion Using Autoencoder-Based Dimensionality Reduction". In: *Processes* 13.4 (2025), p. 1093.

[103] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "(1986) DE Rumelhart, GE Hinton, and RJ Williams, Learning internal representations by error propagation, Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I, DE Rumelhart and JL McClelland (Eds.) Cambridge, MA: MIT Press, pp. 318-362". In: (1988).

[104] Mantas Lukoševičius. "A practical guide to applying echo state networks". In: *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 659–686.

[105] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[106] Ronald J Williams and David Zipser. "A learning algorithm for continually running fully recurrent neural networks". In: *Neural Computation* 1.2 (1989), pp. 270–280.

[107] Kenji Doya et al. "Bifurcations in the learning of recurrent neural networks 3". In: *learning (RTRL)* 3 (1992), p. 17.

[108] James Martens and Ilya Sutskever. "Learning recurrent neural networks with hessian-free optimization". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 1033–1040.

[109] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[110] David Verstraeten et al. "An experimental unification of reservoir computing methods". In: *Neural Networks* 20.3 (2007), pp. 391–403.

[111] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.34 (2001), p. 13.

[112] Herbert Jaeger. "Echo state network". In: *Scholarpedia* 2.9 (2007), p. 2330.

[113] Andre-i Nikolaevich Tikhonov et al. *Numerical methods for the approximate solution of ill-posed problems on compact sets*. Springer, 1995.

[114] Sandip Maji and Srinivasan Natesan. "Error analysis for discontinuous Galerkin method for time-fractional Burgers' equation". In: *Mathematical Methods in the Applied Sciences* 47.12 (2024), pp. 9703–9717.

[115] Martin Thoma. "Analysis and optimization of convolutional neural network architectures". In: *arXiv preprint arXiv:1707.09725* (2017).

[116] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[117] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1. Atlanta, GA. 2013, p. 3.

[118] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. pmlr. 2015, pp. 448–456.

[119] César Quilodrán Casas, Rossella Arcucci, and Yike Guo. "Urban air pollution forecasts generated from latent space representation". In: *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*. 2020.

[120] Cesar Quilodran-Casas and Rossella Arcucci. "A data-driven adversarial machine learning for 3D surrogates of unstructured computational fluid dynamic simulations". In: *Physica A: Statistical Mechanics and its Applications* 615 (2023), p. 128564.

[121] Víctor Francés-Belda et al. "Toward aerodynamic surrogate modeling based on β-variational autoencoders". In: *Physics of Fluids* 36.11 (2024).

[122] Sanparith Marukatat. "Tutorial on PCA and approximate PCA and approximate kernel PCA". In: *Artificial Intelligence Review* 56.6 (2023), pp. 5445–5477.

[123] Jordi Pons et al. “Upsampling artifacts in neural audio synthesis”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 3005–3009.

[124] Masanori Suganuma, Mete Ozay, and Takayuki Okatani. “Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4771–4780.

[125] Aston Zhang et al. *Dive into deep learning*. Cambridge University Press, 2023.

[126] Alberto Racca and Luca Magri. “Robust optimization and validation of echo state networks for learning chaotic dynamics”. In: *Neural Networks* 142 (2021), pp. 252–268.

[127] Nikolay Nikitin. “Characteristics of the leading Lyapunov vector in a turbulent channel flow”. In: *Journal of Fluid Mechanics* 849 (2018), pp. 942–967.

[128] Patrick J Roache. “Code verification by the method of manufactured solutions”. In: *J. Fluids Eng.* 124.1 (2002), pp. 4–10.

[129] Julian D Cole. “On a quasi-linear parabolic equation occurring in aerodynamics”. In: *Quarterly of Applied Mathematics* 9.3 (1951), pp. 225–236.

[130] Bülent Saka and İdris Dağ. “A numerical study of the Burgers’ equation”. In: *Journal of the Franklin Institute* 345.4 (2008), pp. 328–348.

[131] Brian Whiteaker and Peter Gerstoft. “Reducing echo state network size with controllability matrices”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 32.7 (2022).

[132] Bowen Wang et al. “Echo state network structure optimization algorithm based on correlation analysis”. In: *Applied Soft Computing* 152 (2024), p. 111214.

[133] Le Yin, Yongyun Hwang, and John Christos Vassilicos. “Dynamics of turbulent energy and dissipation in channel flow”. In: *Journal of Fluid Mechanics* 996 (2024), A12.

[134] Jung-Il Choi, Kyongmin Yeo, and Changhoon Lee. “Lagrangian statistics in turbulent channel flow”. In: *Physics of fluids* 16.3 (2004), pp. 779–793.

[135] Sergio Hoyas and Javier Jiménez. “Scaling of the velocity fluctuations in turbulent channels up to Reτ= 2003”. In: *Physics of fluids* 18.1 (2006).

[136] Robert D Moser, John Kim, Nagi N Mansour, et al. “Direct numerical simulation of turbulent channel flow up to Re= 590”. In: *Phys. fluids* 11.4 (1999), pp. 943–945.

[137] Steven J Hulshoff. “Implicit subgrid-scale models in space–time variational-multiscale discretizations”. In: *International journal for numerical methods in fluids* 47.10-11 (2005), pp. 1093–1099.

A

Discrete Adjoints

A.1. Discrete Adjoints Formulation

Considering a differential equation of the form:

$$\mathcal{L}u = f, \tag{A.1}$$

where $\mathcal{L}$ is an arbitrary differential operator (e.g., $\mathcal{L} \equiv \frac{\partial}{\partial x}$, $\mathcal{L} \equiv \nabla^2$, etc.), u is the unknown solution, and f is the prescribed source term. These are defined on a domain Ω with appropriate boundary conditions. For most practical problems, obtaining an analytical solution to this equation is not straightforward due to its potential complexity. Consequently, numerical approximations of u are typically preferred. Using a numerical method such as Finite Element or Finite Volume, the differential equation can be discretized to yield a set of algebraic equations:

$$\mathbf{A}\mathbf{U} = \mathbf{F}, \tag{A.2}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a matrix representing the differential operator $\mathcal{L}$; $\mathbf{U} \in \mathbb{R}^N$ is a vector approximating u; and $\mathbf{F} \in \mathbb{R}^N$ is a vector containing the source term f and boundary data. To solve this system, we require the matrix $\mathbf{A}^{-1}$ to compute the solution $\mathbf{U}$ as:

$$\mathbf{U} = \mathbf{A}^{-1}\mathbf{F}. \tag{A.3}$$

However, if only a single entry U_i of $\mathbf{U}$ is of interest, it suffices to compute only the i^{th} row of $\mathbf{A}^{-1}$. For example, as shown in Eq. (A.4), only the highlighted row of $\mathbf{A}^{-1}$ and $\mathbf{F}$ are needed to calculate U_i:

$$\underbrace{\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_i \\ \vdots \\ U_N \end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix} A_{11}^{-1} & A_{12}^{-1} & \cdots & A_{1N}^{-1} \\ A_{21}^{-1} & A_{22}^{-1} & \cdots & A_{2N}^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{i1}^{-1} & A_{i2}^{-1} & \cdots & A_{iN}^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1}^{-1} & A_{N2}^{-1} & \cdots & A_{NN}^{-1} \end{bmatrix}}_{\mathbf{A}^{-1}} \underbrace{\begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}}_{\mathbf{F}} \tag{A.4}$$

The highlighted row of $\mathbf{A}^{-1}$ provides not solely the information needed to compute U_i but also the sensitivity of U_i to perturbations in the source terms of $\mathbf{F}$. This row is referred to in the literature as the *Dual Vector*, *Output Adjoint Vector*, or simply the *Adjoint*, and is denoted as $\mathbf{\Psi}$ in discrete form. The output of interest, U_i, is typically represented as J. This leads to the following expression:

$$J = \mathbf{\Psi}^T\mathbf{F}, \tag{A.5}$$

which is known as the *Dual Form* of the output.

To define the adjoint formally, we first compute the derivative of the output J with respect to $\mathbf{U}$ [42]. Since $J = U_i$, this derivative is a Cartesian row vector with only the i^{th} entry being nonzero:

$$\frac{\partial J}{\partial \mathbf{U}} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}. \tag{A.6}$$

Thus, the adjoint can then be expressed as:

$$\mathbf{\Psi}^T = \frac{\partial J}{\partial \mathbf{U}} \mathbf{A}^{-1}. \tag{A.7}$$

In the context of CFD, the governing equations are often written in terms of residuals:

$$\mathbf{R}(\mathbf{U}) \equiv \mathbf{A}\mathbf{U} - \mathbf{F}, \tag{A.8}$$

where $\mathbf{R} \in \mathbb{R}^N$ is the residual vector. The governing equations are satisfied when $\mathbf{R} = \mathbf{0}$. Taking the derivative of $\mathbf{R}$ with respect to $\mathbf{U}$ yields:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}} = \mathbf{A}. \tag{A.9}$$

Substituting this into Eq. (A.7) and transposing both sides gives the *Adjoint Equation*:

$$\mathbf{\Psi}^T = \frac{\partial J}{\partial \mathbf{U}} \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^{-1} \rightarrow \mathbf{\Psi} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^{-T} \frac{\partial J}{\partial \mathbf{U}}^{T} \tag{A.10}$$

$$\rightarrow \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^{T} \mathbf{\Psi} = \frac{\partial J}{\partial \mathbf{U}}^{T}. \tag{A.11}$$

For nonlinear problems, the adjoint equation becomes [42]:

$$\left.\frac{\partial \mathbf{R}^T}{\partial \mathbf{U}}\right|_{\mathbf{U}} \mathbf{\Psi} = \left.\frac{\partial J^T}{\partial \mathbf{U}}\right|_{\mathbf{U}}, \tag{A.12}$$

where both the Jacobian and the output linearization are evaluated at a specific state $\mathbf{U}$.

A.2. Discretized Error Estimation

The idea presented in Section 3.2 is formulated for linear cases. To generalize this concept to all problems, regardless of nonlinearity and steadiness, the introduction of discrete error estimation is necessary. Considering the discretized set of nonlinear equations:

$$\mathbf{R}_H(\mathbf{U}_H) = 0, \tag{A.13}$$

where $\mathbf{R}_H$ represents the residuals and the subscript H denotes data on a given space (mesh). Following the same concept discussed for the continuous case, the output error is given by:

$$\delta J = J(\mathbf{U}) - J_H(\mathbf{U}_H). \tag{A.14}$$

Since the exact solution $\mathbf{U}$ is unknown, the solution on a finer space $\mathbf{U}_h$ is considered to estimate the output error as in Eq. (A.15). The finer space, denoted as $\mathcal{V}_h$, can be obtained through an order-enriched space or a uniformly h-refined mesh from the original space $\mathcal{V}_H$ [42]. Consequently, the coarse space is included in the fine space, i.e., $\mathcal{V}_H \subset \mathcal{V}_h$:

$$\delta J_{\text{est}} = J_h(\mathbf{U}_h) - J_H(\mathbf{U}_H). \tag{A.15}$$

Since computing the fine-space solution is often impractical, the error estimate δJ_{est} is preferably computed without solving for $\mathbf{U}_h$. This requires injecting the coarse solution $\mathbf{U}_H$ into the fine space using a lossless injection operator $\mathbf{I}_h^H$:

$$\mathbf{U}_h^H = \mathbf{I}_h^H \mathbf{U}_H. \tag{A.16}$$

Prior to defining δJ_{est}, the state perturbation $\delta\mathbf{U}$ is introduced as the difference between the fine solution $\mathbf{U}_h$ and the injected coarse solution into the fine mesh $\mathbf{U}_h^H$:

$$\delta\mathbf{U} = \mathbf{U}_h - \mathbf{U}_h^H. \tag{A.17}$$

Since the fine-space output is unknown, it can be Taylor-expanded about the coarse space state $\mathbf{U}_h^H$ as

$$J_h(\mathbf{U}_h) = J_h(\mathbf{U}_h^H) + \left.\frac{\partial J_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H} \delta\mathbf{U} + \mathcal{O}(\delta\mathbf{U}^2) \tag{A.18}$$

$$\xrightarrow{\text{Dropping High Orders}} J_h(\mathbf{U}_h) \approx J_h(\mathbf{U}_h^H) + \left.\frac{\partial J_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H} \delta\mathbf{U}. \tag{A.19}$$

Since the perturbation $\delta\mathbf{U}$ is unknown, a similar expansion is applied on the fine space residuals:

$$\mathbf{R}_h(\mathbf{U}_h) \approx \mathbf{R}_h(\mathbf{U}_h^H) + \left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H} \delta\mathbf{U} = 0. \tag{A.20}$$

To derive δJ_{est}, the following steps are performed:

$$\xrightarrow{\text{(A.19) \& (A.20)}} \delta\mathbf{U} = -\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H}^{-1} \mathbf{R}_h(\mathbf{U}_h^H) \tag{A.21}$$

$$\xrightarrow{\text{(A.21) in (A.19)}} J_h(\mathbf{U}_h) \approx J_h(\mathbf{U}_h^H) - \left.\frac{\partial J_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H} \left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H}^{-1} \mathbf{R}_h(\mathbf{U}_h^H) \tag{A.22}$$

$$\xrightarrow{\text{(A.11)}} J_h(\mathbf{U}_h) \approx J_h(\mathbf{U}_h^H) - \mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) \tag{A.23}$$

$$\xrightarrow{J_h(\mathbf{U}_h^H) = J_H(\mathbf{U}_H)} J_h(\mathbf{U}_h) - J_H(\mathbf{U}_H) \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) \tag{A.24}$$

$$\rightarrow \delta J_{\text{est}} \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) \tag{A.25}$$

Provided that an element-based discretization method is used, the output error estimate can be localized into individual elements as:

$$\delta J_{\text{est}} \approx \sum_{e=1}^{N_e^h} -\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H), \tag{A.26}$$

where N_e^h is the total number of elements in the fine space. Accordingly, a local error indicator can be derived for each element:

$$\varepsilon_e = \left| \mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H) \right|. \tag{A.27}$$

B

Convolutional Autoencoder Architectures

This chapter presents the generated CAE architectures used for the compression and reconstruction of the injected residuals and fine primal solutions in both the MMS and DNS cases.

B.1. Manufactured Solution

B.1.1. Injected Residuals

Table B.1: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' injected residuals for the case $N_x^{(H)} = 32$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	16
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E2	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E3	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E4	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E5	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E6	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E7	Flatten	$(4, 1)$	(4)		0
E8	Linear	(4)	(2)		10

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(2)	(4)		12
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(1, 64)$	$(3, 1, 1)$	12

Table B.2: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' injected residuals for the case $N_x^{(H)} = 64$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	16
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E2	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	64
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E3	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E4	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E5	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E6	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E7	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E8	Flatten	$(4, 1)$	(4)		0
E9	Linear	(4)	(2)		10

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(2)	(4)		12
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(4, 64)$	$(3, 1, 1)$	48
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	$(4, 128)$	$(1, 128)$	$(3, 1, 1)$	12

Table B.3: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' injected residuals for the case $N_x^{(H)} = 128$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 256)$	$(4, 128)$	$(4, 2, 1)$	16
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
E2	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	64
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E3	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	64
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E4	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E5	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E6	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E7	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E8	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E9	Flatten	$(4, 1)$	(4)		0
E10	Linear	(4)	(2)		10

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(2)	(4)		12
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(4, 64)$	$(3, 1, 1)$	48
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	$(4, 128)$	$(4, 128)$	$(3, 1, 1)$	48
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
D9	Upsample	$(4, 128)$	$(4, 256)$		0
	Conv. 1D	$(4, 256)$	$(1, 256)$	$(3, 1, 1)$	12

Table B.4: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' injected residuals for the case $N_x^{(H)} = 256$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 512)$	$(4, 256)$	$(4, 2, 1)$	16
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
E2	Conv. 1D	$(1, 256)$	$(4, 128)$	$(4, 2, 1)$	64
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
E3	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	64
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E4	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	64
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E5	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E6	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E7	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E8	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E9	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E10	Flatten	$(4, 1)$	(4)		0
E11	Linear	(4)	(2)		10

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(2)	(4)		12
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(4, 64)$	$(3, 1, 1)$	48
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	$(4, 128)$	$(4, 128)$	$(3, 1, 1)$	48
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
D9	Upsample	$(4, 128)$	$(4, 256)$		0
	Conv. 1D	$(4, 256)$	$(4, 256)$	$(3, 1, 1)$	48
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
D10	Upsample	$(4, 256)$	$(4, 512)$		0
	Conv. 1D	$(4, 512)$	$(1, 512)$	$(3, 1, 1)$	12

B.1.2. Primal Solution

Table B.5: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' fine primal solution for the case $N_x^{(h)} = 64$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	16
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E2	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E3	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E4	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E5	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E6	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E7	Flatten	$(4, 1)$	(4)		0
E8	Linear	(4)	(1)		5

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(1)	(4)		8
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(1, 64)$	$(3, 1, 1)$	12

Table B.6: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' fine primal solution for the case $N_x^{(h)} = 128$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	16
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E2	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	64
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E3	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E4	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E5	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E6	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E7	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E8	Flatten	$(4, 1)$	(4)		0
E9	Linear	(4)	(1)		5

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(1)	(4)		8
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(4, 64)$	$(3, 1, 1)$	48
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	$(4, 128)$	$(1, 128)$	$(3, 1, 1)$	12

Table B.7: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' fine primal solution for the case $N_x^{(h)} = 256$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 256)$	$(4, 128)$	$(4, 2, 1)$	16
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
E2	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	64
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E3	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	64
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E4	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E5	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E6	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E7	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E8	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E9	Flatten	$(4, 1)$	(4)		0
E10	Linear	(4)	(1)		5

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(1)	(4)		8
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(4, 64)$	$(3, 1, 1)$	48
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	$(4, 128)$	$(4, 128)$	$(3, 1, 1)$	48
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
D9	Upsample	$(4, 128)$	$(4, 256)$		0
	Conv. 1D	$(4, 256)$	$(1, 256)$	$(3, 1, 1)$	12

Table B.8: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D MMS Burgers' fine primal solution for the case $N_x^{(h)} = 512$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 512)$	$(4, 256)$	$(4, 2, 1)$	16
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
E2	Conv. 1D	$(1, 256)$	$(4, 128)$	$(4, 2, 1)$	64
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
E3	Conv. 1D	$(1, 128)$	$(4, 64)$	$(4, 2, 1)$	64
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
E4	Conv. 1D	$(1, 64)$	$(4, 32)$	$(4, 2, 1)$	64
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
E5	Conv. 1D	$(4, 32)$	$(4, 16)$	$(4, 2, 1)$	64
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
E6	Conv. 1D	$(4, 16)$	$(4, 8)$	$(4, 2, 1)$	64
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
E7	Conv. 1D	$(4, 8)$	$(4, 4)$	$(4, 2, 1)$	64
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
E8	Conv. 1D	$(4, 4)$	$(4, 2)$	$(4, 2, 1)$	64
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
E9	Conv. 1D	$(4, 2)$	$(4, 1)$	$(4, 2, 1)$	64
	BN	$(4, 1)$	$(4, 1)$		8
	LReLU	$(4, 1)$	$(4, 1)$		0
E10	Flatten	$(4, 1)$	(4)		0
E11	Linear	(4)	(1)		5

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(1)	(4)		8
D2	Upsample	$(4, 1)$	$(4, 2)$		0
	Conv. 1D	$(4, 2)$	$(4, 2)$	$(3, 1, 1)$	48
	BN	$(4, 2)$	$(4, 2)$		8
	LReLU	$(4, 2)$	$(4, 2)$		0
D3	Upsample	$(4, 2)$	$(4, 4)$		0
	Conv. 1D	$(4, 4)$	$(4, 4)$	$(3, 1, 1)$	48
	BN	$(4, 4)$	$(4, 4)$		8
	LReLU	$(4, 4)$	$(4, 4)$		0
D4	Upsample	$(4, 4)$	$(4, 8)$		0
	Conv. 1D	$(4, 8)$	$(4, 8)$	$(3, 1, 1)$	48
	BN	$(4, 8)$	$(4, 8)$		8
	LReLU	$(4, 8)$	$(4, 8)$		0
D5	Upsample	$(4, 8)$	$(4, 16)$		0
	Conv. 1D	$(4, 16)$	$(4, 16)$	$(3, 1, 1)$	48
	BN	$(4, 16)$	$(4, 16)$		8
	LReLU	$(4, 16)$	$(4, 16)$		0
D6	Upsample	$(4, 16)$	$(4, 32)$		0
	Conv. 1D	$(4, 32)$	$(4, 32)$	$(3, 1, 1)$	48
	BN	$(4, 32)$	$(4, 32)$		8
	LReLU	$(4, 32)$	$(4, 32)$		0
D7	Upsample	$(4, 32)$	$(4, 64)$		0
	Conv. 1D	$(4, 64)$	$(4, 64)$	$(3, 1, 1)$	48
	BN	$(4, 64)$	$(4, 64)$		8
	LReLU	$(4, 64)$	$(4, 64)$		0
D8	Upsample	$(4, 64)$	$(4, 128)$		0
	Conv. 1D	$(4, 128)$	$(4, 128)$	$(3, 1, 1)$	48
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
D9	Upsample	$(4, 128)$	$(4, 256)$		0
	Conv. 1D	$(4, 256)$	$(4, 256)$	$(3, 1, 1)$	48
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
D10	Upsample	$(4, 256)$	$(4, 512)$		0
	Conv. 1D	$(4, 512)$	$(1, 512)$	$(3, 1, 1)$	12

B.2. DNS-forced 1D Burgers' Equation

B.2.1. Injected Residuals

Table B.9: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D DNS-forced Burgers' injected residuals for the case $N_y^{(H)} = 128$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 256)$	$(4, 128)$	$(4, 2, 1)$	16
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
E2	Conv. 1D	$(4, 128)$	$(8, 64)$	$(4, 2, 1)$	128
	BN	$(8, 64)$	$(8, 64)$		16
	LReLU	$(8, 64)$	$(8, 64)$		0
E3	Conv. 1D	$(8, 64)$	$(16, 32)$	$(4, 2, 1)$	512
	BN	$(16, 32)$	$(16, 32)$		32
	LReLU	$(16, 32)$	$(16, 32)$		0
E4	Conv. 1D	$(16, 32)$	$(32, 16)$	$(4, 2, 1)$	2048
	BN	$(32, 16)$	$(32, 16)$		64
	LReLU	$(32, 16)$	$(32, 16)$		0
E5	Conv. 1D	$(32, 16)$	$(64, 8)$	$(4, 2, 1)$	8192
	BN	$(64, 8)$	$(64, 8)$		128
	LReLU	$(64, 8)$	$(64, 8)$		0
E6	Conv. 1D	$(64, 8)$	$(64, 4)$	$(4, 2, 1)$	16384
	BN	$(64, 4)$	$(64, 4)$		128
	LReLU	$(64, 4)$	$(64, 4)$		0
E7	Conv. 1D	$(64, 4)$	$(64, 2)$	$(4, 2, 1)$	16384
	BN	$(64, 2)$	$(64, 2)$		128
	LReLU	$(64, 2)$	$(64, 2)$		0
E8	Conv. 1D	$(64, 2)$	$(64, 1)$	$(4, 2, 1)$	16384
	BN	$(64, 1)$	$(64, 1)$		64
	LReLU	$(64, 1)$	$(64, 1)$		0
E9	Flatten	$(64, 1)$	(64)		0
E10	Linear	(64)	(74)		4810

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(74)	(64)		4800
D2	Upsample	$(64, 1)$	$(64, 2)$		0
	Conv. 1D	$(64, 2)$	$(64, 2)$	$(3, 1, 1)$	12288
	BN	$(64, 2)$	$(64, 2)$		128
	LReLU	$(64, 2)$	$(64, 2)$		0
D3	Upsample	$(64, 2)$	$(64, 4)$		0
	Conv. 1D	$(64, 4)$	$(64, 4)$	$(3, 1, 1)$	12288
	BN	$(64, 4)$	$(64, 4)$		128
	LReLU	$(64, 4)$	$(64, 4)$		0
D4	Upsample	$(64, 4)$	$(64, 8)$		0
	Conv. 1D	$(64, 8)$	$(64, 8)$	$(3, 1, 1)$	12288
	BN	$(64, 8)$	$(64, 8)$		128
	LReLU	$(64, 8)$	$(64, 8)$		0
D5	Upsample	$(64, 8)$	$(64, 16)$		0
	Conv. 1D	$(64, 16)$	$(32, 16)$	$(3, 1, 1)$	6144
	BN	$(32, 16)$	$(32, 16)$		64
	LReLU	$(32, 16)$	$(32, 16)$		0
D6	Upsample	$(32, 16)$	$(32, 32)$		0
	Conv. 1D	$(32, 32)$	$(16, 32)$	$(3, 1, 1)$	1536
	BN	$(16, 32)$	$(16, 32)$		32
	LReLU	$(16, 32)$	$(16, 32)$		0
D7	Upsample	$(16, 32)$	$(16, 64)$		0
	Conv. 1D	$(16, 64)$	$(8, 64)$	$(3, 1, 1)$	384
	BN	$(8, 64)$	$(8, 64)$		16
	LReLU	$(8, 64)$	$(8, 64)$		0
D8	Upsample	$(8, 64)$	$(8, 128)$		0
	Conv. 1D	$(8, 128)$	$(4, 128)$	$(3, 1, 1)$	96
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
D9	Upsample	$(4, 128)$	$(4, 256)$		0
	Conv. 1D	(256)	$(1, 256)$	$(3, 1, 1)$	12

Table B.10: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D DNS-forced Burgers' injected residuals for the case $N_y^{(H)} = 256$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 512)$	$(4, 256)$	$(4, 2, 1)$	16
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
E2	Conv. 1D	$(4, 256)$	$(8, 128)$	$(4, 2, 1)$	128
	BN	$(8, 128)$	$(8, 128)$		16
	LReLU	$(8, 128)$	$(8, 128)$		0
E3	Conv. 1D	$(8, 128)$	$(16, 64)$	$(4, 2, 1)$	512
	BN	$(16, 64)$	$(16, 64)$		32
	LReLU	$(16, 64)$	$(16, 64)$		0
E4	Conv. 1D	$(16, 64)$	$(32, 32)$	$(4, 2, 1)$	2048
	BN	$(32, 32)$	$(32, 32)$		64
	LReLU	$(32, 32)$	$(32, 32)$		0
E5	Conv. 1D	$(32, 32)$	$(64, 16)$	$(4, 2, 1)$	8192
	BN	$(64, 16)$	$(64, 16)$		128
	LReLU	$(64, 16)$	$(64, 16)$		0
E6	Conv. 1D	$(64, 16)$	$(128, 8)$	$(4, 2, 1)$	32768
	BN	$(128, 8)$	$(128, 8)$		256
	LReLU	$(128, 8)$	$(128, 8)$		0
E7	Conv. 1D	$(128, 8)$	$(128, 4)$	$(4, 2, 1)$	65536
	BN	$(128, 4)$	$(128, 4)$		256
	LReLU	$(128, 4)$	$(128, 4)$		0
E8	Conv. 1D	$(128, 4)$	$(128, 2)$	$(4, 2, 1)$	65536
	BN	$(128, 2)$	$(128, 2)$		256
	LReLU	$(128, 2)$	$(128, 2)$		0
E9	Conv. 1D	$(128, 2)$	$(128, 1)$	$(4, 2, 1)$	65536
	BN	$(128, 1)$	$(128, 1)$		64
	LReLU	$(128, 1)$	$(128, 1)$		0
E10	Flatten	$(128, 1)$	(128)		0
E11	Linear	(128)	(90)		11610

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(90)	(128)		11648
D2	Upsample	$(128, 1)$	$(128, 2)$		0
	Conv. 1D	$(128, 2)$	$(128, 2)$	$(3, 1, 1)$	49152
	BN	$(128, 2)$	$(128, 2)$		256
	LReLU	$(128, 2)$	$(128, 2)$		0
D3	Upsample	$(128, 2)$	$(128, 4)$		0
	Conv. 1D	$(128, 4)$	$(128, 4)$	$(3, 1, 1)$	49152
	BN	$(128, 4)$	$(128, 4)$		256
	LReLU	$(128, 4)$	$(128, 4)$		0
D4	Upsample	$(128, 4)$	$(128, 8)$		0
	Conv. 1D	$(128, 8)$	$(128, 8)$	$(3, 1, 1)$	49152
	BN	$(128, 8)$	$(128, 8)$		256
	LReLU	$(128, 8)$	$(128, 8)$		0
D5	Upsample	$(128, 8)$	$(128, 16)$		0
	Conv. 1D	$(128, 16)$	$(64, 16)$	$(3, 1, 1)$	24576
	BN	$(64, 16)$	$(64, 16)$		128
	LReLU	$(64, 16)$	$(64, 16)$		0
D6	Upsample	$(64, 16)$	$(64, 32)$		0
	Conv. 1D	$(64, 32)$	$(32, 32)$	$(3, 1, 1)$	6144
	BN	$(32, 32)$	$(32, 32)$		64
	LReLU	$(32, 32)$	$(32, 32)$		0
D7	Upsample	$(32, 32)$	$(32, 64)$		0
	Conv. 1D	$(32, 64)$	$(16, 64)$	$(3, 1, 1)$	1536
	BN	$(16, 64)$	$(16, 64)$		32
	LReLU	$(16, 64)$	$(16, 64)$		0
D8	Upsample	$(16, 64)$	$(16, 128)$		0
	Conv. 1D	$(16, 128)$	$(8, 128)$	$(3, 1, 1)$	384
	BN	$(8, 128)$	$(8, 128)$		16
	LReLU	$(8, 128)$	$(8, 128)$		0
D9	Upsample	$(8, 128)$	$(8, 256)$		0
	Conv. 1D	$(8, 256)$	$(4, 256)$	$(3, 1, 1)$	96
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
D10	Upsample	$(4, 256)$	$(4, 512)$		0
	Conv. 1D	(512)	$(1, 512)$	$(3, 1, 1)$	12

Table B.11: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D DNS-forced Burgers' injected residuals for the case $N_y^{(H)} = 512$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 1024)$	$(4, 512)$	$(4, 2, 1)$	16
	BN	$(4, 512)$	$(4, 512)$		8
	LReLU	$(4, 512)$	$(4, 512)$		0
E2	Conv. 1D	$(4, 512)$	$(8, 256)$	$(4, 2, 1)$	128
	BN	$(8, 256)$	$(8, 256)$		16
	LReLU	$(8, 256)$	$(8, 256)$		0
E3	Conv. 1D	$(8, 256)$	$(16, 128)$	$(4, 2, 1)$	512
	BN	$(16, 128)$	$(16, 128)$		32
	LReLU	$(16, 128)$	$(16, 128)$		0
E4	Conv. 1D	$(16, 128)$	$(32, 64)$	$(4, 2, 1)$	2048
	BN	$(32, 64)$	$(32, 64)$		64
	LReLU	$(32, 64)$	$(32, 64)$		0
E5	Conv. 1D	$(32, 64)$	$(64, 32)$	$(4, 2, 1)$	8192
	BN	$(64, 32)$	$(64, 32)$		128
	LReLU	$(64, 32)$	$(64, 32)$		0
E6	Conv. 1D	$(64, 32)$	$(128, 16)$	$(4, 2, 1)$	32768
	BN	$(128, 16)$	$(128, 16)$		256
	LReLU	$(128, 16)$	$(128, 16)$		0
E7	Conv. 1D	$(128, 16)$	$(256, 8)$	$(4, 2, 1)$	131072
	BN	$(256, 8)$	$(256, 8)$		512
	LReLU	$(256, 8)$	$(256, 8)$		0
E8	Conv. 1D	$(256, 8)$	$(256, 4)$	$(4, 2, 1)$	262144
	BN	$(256, 4)$	$(256, 4)$		512
	LReLU	$(256, 4)$	$(256, 4)$		0
E9	Conv. 1D	$(256, 4)$	$(256, 2)$	$(4, 2, 1)$	262144
	BN	$(256, 2)$	$(256, 2)$		512
	LReLU	$(256, 2)$	$(256, 2)$		0
E10	Conv. 1D	$(256, 2)$	$(256, 1)$	$(4, 2, 1)$	262144
	BN	$(256, 1)$	$(256, 1)$		512
	LReLU	$(256, 1)$	$(256, 1)$		0
E11	Flatten	$(256, 1)$	(256)		0
E12	Linear	(256)	(150)		38550

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(150)	(256)		38656
D2	Upsample	$(256, 1)$	$(256, 2)$		0
	Conv. 1D	$(256, 2)$	$(256, 2)$	$(3, 1, 1)$	196608
	BN	$(256, 2)$	$(256, 2)$		512
	LReLU	$(256, 2)$	$(256, 2)$		0
D3	Upsample	$(256, 2)$	$(256, 4)$		0
	Conv. 1D	$(256, 4)$	$(256, 4)$	$(3, 1, 1)$	196608
	BN	$(256, 4)$	$(256, 4)$		512
	LReLU	$(256, 4)$	$(256, 4)$		0
D4	Upsample	$(256, 4)$	$(256, 8)$		0
	Conv. 1D	$(256, 8)$	$(256, 8)$	$(3, 1, 1)$	196608
	BN	$(256, 8)$	$(256, 8)$		512
	LReLU	$(256, 8)$	$(256, 8)$		0
D5	Upsample	$(256, 8)$	$(256, 16)$		0
	Conv. 1D	$(256, 16)$	$(128, 16)$	$(3, 1, 1)$	98304
	BN	$(128, 16)$	$(128, 16)$		256
	LReLU	$(128, 16)$	$(128, 16)$		0
D6	Upsample	$(128, 16)$	$(128, 32)$		0
	Conv. 1D	$(128, 32)$	$(64, 32)$	$(3, 1, 1)$	24576
	BN	$(64, 32)$	$(64, 32)$		128
	LReLU	$(64, 32)$	$(64, 32)$		0
D7	Upsample	$(64, 32)$	$(64, 64)$		0
	Conv. 1D	$(64, 64)$	$(32, 64)$	$(3, 1, 1)$	6144
	BN	$(32, 64)$	$(32, 64)$		64
	LReLU	$(32, 64)$	$(32, 64)$		0
D8	Upsample	$(32, 64)$	$(32, 128)$		0
	Conv. 1D	$(32, 128)$	$(16, 128)$	$(3, 1, 1)$	1536
	BN	$(16, 128)$	$(16, 128)$		32
	LReLU	$(16, 128)$	$(16, 128)$		0
D9	Upsample	$(16, 128)$	$(16, 256)$		0
	Conv. 1D	$(16, 256)$	$(8, 256)$	$(3, 1, 1)$	384
	BN	$(8, 256)$	$(8, 256)$		16
	LReLU	$(8, 256)$	$(8, 256)$		0
D10	Upsample	$(8, 256)$	$(8, 512)$		0
	Conv. 1D	$(8, 512)$	$(4, 512)$	$(3, 1, 1)$	96
	BN	$(4, 512)$	$(4, 512)$		8
	LReLU	$(4, 512)$	$(4, 512)$		0
D11	Upsample	$(4, 512)$	$(4, 1024)$		0
	Conv. 1D	$(4, 1024)$	$(1, 1024)$	$(3, 1, 1)$	12

B.2.2. Primal Solution

Table B.12: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D DNS-forced Burgers' fine primal solution for the case $N_y^{(h)} = 256$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 256)$	$(4, 128)$	$(4, 2, 1)$	16
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
E2	Conv. 1D	$(4, 128)$	$(8, 64)$	$(4, 2, 1)$	128
	BN	$(8, 64)$	$(8, 64)$		16
	LReLU	$(8, 64)$	$(8, 64)$		0
E3	Conv. 1D	$(8, 64)$	$(16, 32)$	$(4, 2, 1)$	512
	BN	$(16, 32)$	$(16, 32)$		32
	LReLU	$(16, 32)$	$(16, 32)$		0
E4	Conv. 1D	$(16, 32)$	$(32, 16)$	$(4, 2, 1)$	2048
	BN	$(32, 16)$	$(32, 16)$		64
	LReLU	$(32, 16)$	$(32, 16)$		0
E5	Conv. 1D	$(32, 16)$	$(64, 8)$	$(4, 2, 1)$	8192
	BN	$(64, 8)$	$(64, 8)$		128
	LReLU	$(64, 8)$	$(64, 8)$		0
E6	Conv. 1D	$(64, 8)$	$(64, 4)$	$(4, 2, 1)$	16384
	BN	$(64, 4)$	$(64, 4)$		128
	LReLU	$(64, 4)$	$(64, 4)$		0
E7	Conv. 1D	$(64, 4)$	$(64, 2)$	$(4, 2, 1)$	16384
	BN	$(64, 2)$	$(64, 2)$		128
	LReLU	$(64, 2)$	$(64, 2)$		0
E8	Conv. 1D	$(64, 2)$	$(64, 1)$	$(4, 2, 1)$	16384
	BN	$(64, 1)$	$(64, 1)$		64
	LReLU	$(64, 1)$	$(64, 1)$		0
E9	Flatten	$(64, 1)$	(64)		0
E10	Linear	(64)	(17)		1105

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(17)	(64)		1152
D2	Upsample	$(64, 1)$	$(64, 2)$		0
	Conv. 1D	$(64, 2)$	$(64, 2)$	$(3, 1, 1)$	12288
	BN	$(64, 2)$	$(64, 2)$		128
	LReLU	$(64, 2)$	$(64, 2)$		0
D3	Upsample	$(64, 2)$	$(64, 4)$		0
	Conv. 1D	$(64, 4)$	$(64, 4)$	$(3, 1, 1)$	12288
	BN	$(64, 4)$	$(64, 4)$		128
	LReLU	$(64, 4)$	$(64, 4)$		0
D4	Upsample	$(64, 4)$	$(64, 8)$		0
	Conv. 1D	$(64, 8)$	$(64, 8)$	$(3, 1, 1)$	12288
	BN	$(64, 8)$	$(64, 8)$		128
	LReLU	$(64, 8)$	$(64, 8)$		0
D5	Upsample	$(64, 8)$	$(64, 16)$		0
	Conv. 1D	$(64, 16)$	$(32, 16)$	$(3, 1, 1)$	6144
	BN	$(32, 16)$	$(32, 16)$		64
	LReLU	$(32, 16)$	$(32, 16)$		0
D6	Upsample	$(32, 16)$	$(32, 32)$		0
	Conv. 1D	$(32, 32)$	$(16, 32)$	$(3, 1, 1)$	1536
	BN	$(16, 32)$	$(16, 32)$		32
	LReLU	$(16, 32)$	$(16, 32)$		0
D7	Upsample	$(16, 32)$	$(16, 64)$		0
	Conv. 1D	$(16, 64)$	$(8, 64)$	$(3, 1, 1)$	384
	BN	$(8, 64)$	$(8, 64)$		16
	LReLU	$(8, 64)$	$(8, 64)$		0
D8	Upsample	$(8, 64)$	$(8, 128)$		0
	Conv. 1D	$(8, 128)$	$(4, 128)$	$(3, 1, 1)$	96
	BN	$(4, 128)$	$(4, 128)$		8
	LReLU	$(4, 128)$	$(4, 128)$		0
D9	Upsample	$(4, 128)$	$(4, 256)$		0
	Conv. 1D	(256)	$(1, 256)$	$(3, 1, 1)$	12

Table B.13: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D DNS-forced Burgers' fine primal solution for the case $N_y^{(h)} = 512$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 512)$	$(4, 256)$	$(4, 2, 1)$	16
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
E2	Conv. 1D	$(4, 256)$	$(8, 128)$	$(4, 2, 1)$	128
	BN	$(8, 128)$	$(8, 128)$		16
	LReLU	$(8, 128)$	$(8, 128)$		0
E3	Conv. 1D	$(8, 128)$	$(16, 64)$	$(4, 2, 1)$	512
	BN	$(16, 64)$	$(16, 64)$		32
	LReLU	$(16, 64)$	$(16, 64)$		0
E4	Conv. 1D	$(16, 64)$	$(32, 32)$	$(4, 2, 1)$	2048
	BN	$(32, 32)$	$(32, 32)$		64
	LReLU	$(32, 32)$	$(32, 32)$		0
E5	Conv. 1D	$(32, 32)$	$(64, 16)$	$(4, 2, 1)$	8192
	BN	$(64, 16)$	$(64, 16)$		128
	LReLU	$(64, 16)$	$(64, 16)$		0
E6	Conv. 1D	$(64, 16)$	$(128, 8)$	$(4, 2, 1)$	32768
	BN	$(128, 8)$	$(128, 8)$		256
	LReLU	$(128, 8)$	$(128, 8)$		0
E7	Conv. 1D	$(128, 8)$	$(128, 4)$	$(4, 2, 1)$	65536
	BN	$(128, 4)$	$(128, 4)$		256
	LReLU	$(128, 4)$	$(128, 4)$		0
E8	Conv. 1D	$(128, 4)$	$(128, 2)$	$(4, 2, 1)$	65536
	BN	$(128, 2)$	$(128, 2)$		256
	LReLU	$(128, 2)$	$(128, 2)$		0
E9	Conv. 1D	$(128, 2)$	$(128, 1)$	$(4, 2, 1)$	65536
	BN	$(128, 1)$	$(128, 1)$		64
	LReLU	$(128, 1)$	$(128, 1)$		0
E10	Flatten	$(128, 1)$	(128)		0
E11	Linear	(128)	(17)		2193

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(17)	(128)		2304
D2	Upsample	$(128, 1)$	$(128, 2)$		0
	Conv. 1D	$(128, 2)$	$(128, 2)$	$(3, 1, 1)$	49152
	BN	$(128, 2)$	$(128, 2)$		256
	LReLU	$(128, 2)$	$(128, 2)$		0
D3	Upsample	$(128, 2)$	$(128, 4)$		0
	Conv. 1D	$(128, 4)$	$(128, 4)$	$(3, 1, 1)$	49152
	BN	$(128, 4)$	$(128, 4)$		256
	LReLU	$(128, 4)$	$(128, 4)$		0
D4	Upsample	$(128, 4)$	$(128, 8)$		0
	Conv. 1D	$(128, 8)$	$(128, 8)$	$(3, 1, 1)$	49152
	BN	$(128, 8)$	$(128, 8)$		256
	LReLU	$(128, 8)$	$(128, 8)$		0
D5	Upsample	$(128, 8)$	$(128, 16)$		0
	Conv. 1D	$(128, 16)$	$(64, 16)$	$(3, 1, 1)$	24576
	BN	$(64, 16)$	$(64, 16)$		128
	LReLU	$(64, 16)$	$(64, 16)$		0
D6	Upsample	$(64, 16)$	$(64, 32)$		0
	Conv. 1D	$(64, 32)$	$(32, 32)$	$(3, 1, 1)$	6144
	BN	$(32, 32)$	$(32, 32)$		64
	LReLU	$(32, 32)$	$(32, 32)$		0
D7	Upsample	$(32, 32)$	$(32, 64)$		0
	Conv. 1D	$(32, 64)$	$(16, 64)$	$(3, 1, 1)$	1536
	BN	$(16, 64)$	$(16, 64)$		32
	LReLU	$(16, 64)$	$(16, 64)$		0
D8	Upsample	$(16, 64)$	$(16, 128)$		0
	Conv. 1D	$(16, 128)$	$(8, 128)$	$(3, 1, 1)$	384
	BN	$(8, 128)$	$(8, 128)$		16
	LReLU	$(8, 128)$	$(8, 128)$		0
D9	Upsample	$(8, 128)$	$(8, 256)$		0
	Conv. 1D	$(8, 256)$	$(4, 256)$	$(3, 1, 1)$	96
	BN	$(4, 256)$	$(4, 256)$		8
	LReLU	$(4, 256)$	$(4, 256)$		0
D10	Upsample	$(4, 256)$	$(4, 512)$		0
	Conv. 1D	(512)	$(1, 512)$	$(3, 1, 1)$	12

Table B.14: Layer-by-layer details of the encoder (left) and decoder (right) architectures applied to the 1D DNS-forced Burgers' fine primal solution for the case $N_y^{(h)} = 1024$.

(a) Encoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
E1	Conv. 1D	$(1, 1024)$	$(4, 512)$	$(4, 2, 1)$	16
	BN	$(4, 512)$	$(4, 512)$		8
	LReLU	$(4, 512)$	$(4, 512)$		0
E2	Conv. 1D	$(4, 512)$	$(8, 256)$	$(4, 2, 1)$	128
	BN	$(8, 256)$	$(8, 256)$		16
	LReLU	$(8, 256)$	$(8, 256)$		0
E3	Conv. 1D	$(8, 256)$	$(16, 128)$	$(4, 2, 1)$	512
	BN	$(16, 128)$	$(16, 128)$		32
	LReLU	$(16, 128)$	$(16, 128)$		0
E4	Conv. 1D	$(16, 128)$	$(32, 64)$	$(4, 2, 1)$	2048
	BN	$(32, 64)$	$(32, 64)$		64
	LReLU	$(32, 64)$	$(32, 64)$		0
E5	Conv. 1D	$(32, 64)$	$(64, 32)$	$(4, 2, 1)$	8192
	BN	$(64, 32)$	$(64, 32)$		128
	LReLU	$(64, 32)$	$(64, 32)$		0
E6	Conv. 1D	$(64, 32)$	$(128, 16)$	$(4, 2, 1)$	32768
	BN	$(128, 16)$	$(128, 16)$		256
	LReLU	$(128, 16)$	$(128, 16)$		0
E7	Conv. 1D	$(128, 16)$	$(256, 8)$	$(4, 2, 1)$	131072
	BN	$(256, 8)$	$(256, 8)$		512
	LReLU	$(256, 8)$	$(256, 8)$		0
E8	Conv. 1D	$(256, 8)$	$(256, 4)$	$(4, 2, 1)$	262144
	BN	$(256, 4)$	$(256, 4)$		512
	LReLU	$(256, 4)$	$(256, 4)$		0
E9	Conv. 1D	$(256, 4)$	$(256, 2)$	$(4, 2, 1)$	262144
	BN	$(256, 2)$	$(256, 2)$		512
	LReLU	$(256, 2)$	$(256, 2)$		0
E10	Conv. 1D	$(256, 2)$	$(256, 1)$	$(4, 2, 1)$	262144
	BN	$(256, 1)$	$(256, 1)$		512
	LReLU	$(256, 1)$	$(256, 1)$		0
E11	Flatten	$(256, 1)$	(256)		0
E12	Linear	(256)	(17)		4369

(b) Decoder Architecture

Main Layer	Sublayer	Input Shape	Output Shape	$(k,\ s,\ p)$	# Parameters
D1	Linear	(17)	(256)		4608
D2	Upsample	$(256, 1)$	$(256, 2)$		0
	Conv. 1D	$(256, 2)$	$(256, 2)$	$(3, 1, 1)$	196608
	BN	$(256, 2)$	$(256, 2)$		512
	LReLU	$(256, 2)$	$(256, 2)$		0
D3	Upsample	$(256, 2)$	$(256, 4)$		0
	Conv. 1D	$(256, 4)$	$(256, 4)$	$(3, 1, 1)$	196608
	BN	$(256, 4)$	$(256, 4)$		512
	LReLU	$(256, 4)$	$(256, 4)$		0
D4	Upsample	$(256, 4)$	$(256, 8)$		0
	Conv. 1D	$(256, 8)$	$(256, 8)$	$(3, 1, 1)$	196608
	BN	$(256, 8)$	$(256, 8)$		512
	LReLU	$(256, 8)$	$(256, 8)$		0
D5	Upsample	$(256, 8)$	$(256, 16)$		0
	Conv. 1D	$(256, 16)$	$(128, 16)$	$(3, 1, 1)$	98304
	BN	$(128, 16)$	$(128, 16)$		256
	LReLU	$(128, 16)$	$(128, 16)$		0
D6	Upsample	$(128, 16)$	$(128, 32)$		0
	Conv. 1D	$(128, 32)$	$(64, 32)$	$(3, 1, 1)$	24576
	BN	$(64, 32)$	$(64, 32)$		128
	LReLU	$(64, 32)$	$(64, 32)$		0
D7	Upsample	$(64, 32)$	$(64, 64)$		0
	Conv. 1D	$(64, 64)$	$(32, 64)$	$(3, 1, 1)$	6144
	BN	$(32, 64)$	$(32, 64)$		64
	LReLU	$(32, 64)$	$(32, 64)$		0
D8	Upsample	$(32, 64)$	$(32, 128)$		0
	Conv. 1D	$(32, 128)$	$(16, 128)$	$(3, 1, 1)$	1536
	BN	$(16, 128)$	$(16, 128)$		32
	LReLU	$(16, 128)$	$(16, 128)$		0
D9	Upsample	$(16, 128)$	$(16, 256)$		0
	Conv. 1D	$(16, 256)$	$(8, 256)$	$(3, 1, 1)$	384
	BN	$(8, 256)$	$(8, 256)$		16
	LReLU	$(8, 256)$	$(8, 256)$		0
D10	Upsample	$(8, 256)$	$(8, 512)$		0
	Conv. 1D	$(8, 512)$	$(4, 512)$	$(3, 1, 1)$	96
	BN	$(4, 512)$	$(4, 512)$		8
	LReLU	$(4, 512)$	$(4, 512)$		0
D11	Upsample	$(4, 512)$	$(4, 1024)$		0
	Conv. 1D	$(4, 1024)$	$(1, 1024)$	$(3, 1, 1)$	12

C

Additional Results on 1D MMS Burgers' Problem

C.1. Bayesian Optimization Setup

As stated in Section 5.3.3, the initial step was getting the best BO setup. To do so, the performance of the BO procedure for the ESN was analyzed with respect to two key parameters for the case $N_x^{(H)} = 32$, the number of initial grid search points N_{GS}, and the number of ensembles N_{ens}.

Figure C.1 (left) shows the effect of N_{GS} on the validation MSE across multiple BO runs. Increasing N_{GS} from 4 to 25 results in a noticeable reduction of the median validation error, together with a shrinking inter-percentile range, which indicates improved robustness of the optimization. Beyond $N_{GS} \approx 25$, additional grid evaluations do not yield significant gains in accuracy, while the computational cost continues to rise steeply, as evidenced in Figure C.1 (right). The optimization time scales approximately as $O(N_{GS}^{0.8})$, leading to diminishing returns for larger values. Based on this trade-off, $N_{GS} = 25$ was selected as a suitable balance between accuracy and efficiency.

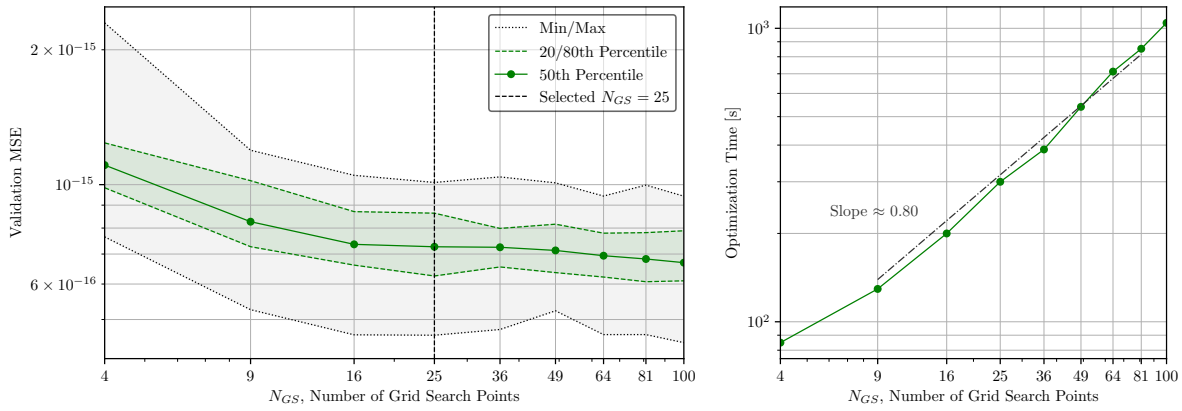


Figure C.1: Effect of the number of initial grid search points N_{GS} on BO performance, analyzed for $N_x^{(H)} = 32$ case with $N_r = 64$ and $\alpha = 0.4$. (Left) Validation MSE distribution across runs, (Right) Optimization time scaling with N_{GS}.

To account for ESN's inherent pseudo-randomness, an ensemble of N_{ens} ESN samples was considered, each with a different random seed. The effect of ensemble size N_{ens} is summarized in Figure C.2. The median validation MSE remains stable across a wide range of N_{ens} values, fluctuating around 6×10^{-6}. However, the spread between the 20th and 80th percentiles decreases as N_{ens} increases, indicating that larger ensembles reduce stochastic variability across BO runs. Nevertheless, beyond $N_{ens} \approx 25$ the improvements are marginal, while computational cost grows proportionally with N_{ens}. For this reason,

$N_{ens} = 28$ was adopted as the default configuration, ensuring statistical reliability without unnecessary cost.

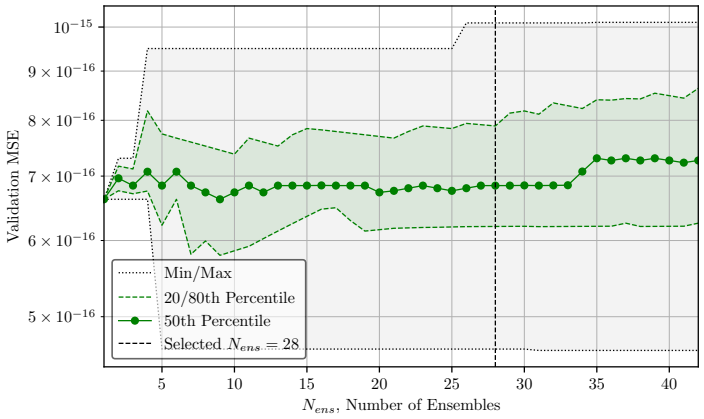


Figure C.2: Effect of the number of ensembles N_{ens} on BO performance, analyzed for $N_x^{(H)} = 32$ case with $N_r = 64$, $\alpha = 0.4$, and $N_{GS} = 25$.

C.2. Performance Assessment of the Reconstructed Injected Residuals

Figure C.3 illustrates the comparison between the reconstructed residual distributions and the actual reference values (plotted in light gray with dashed black lines indicating the temporal mean).

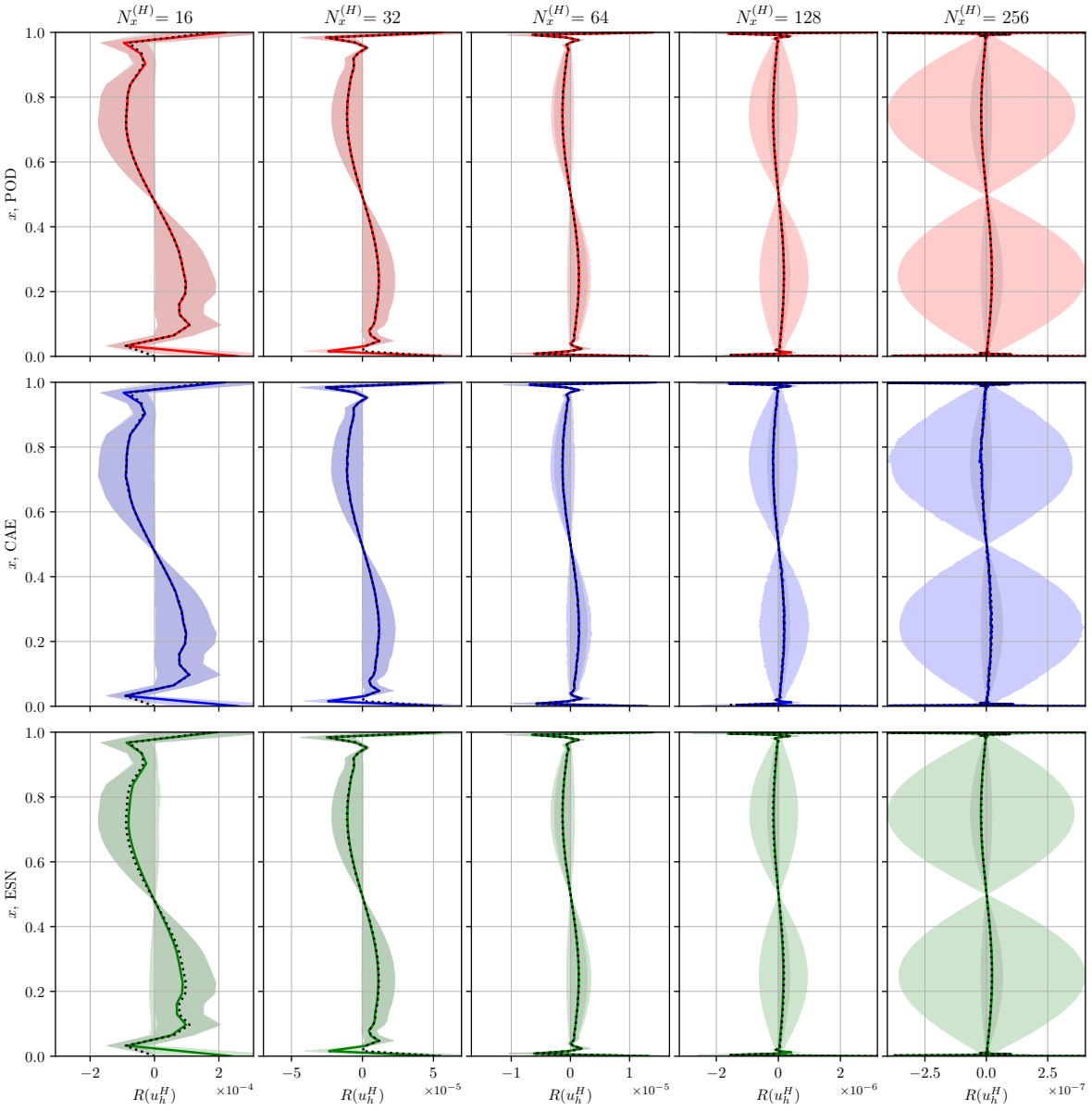


Figure C.3: Reconstructed 1D Burgers' MMS injected residuals $\tilde{R}\left(u_h^H\right)$ for all refinements $N_x^{(H)}$ using POD (top), CAE (middle), and ESN (bottom). The shaded regions are the actual residuals $R\left(u_h^H\right)$, with the dashed black lines indicating their temporal mean.

It can be observed that all three surrogate models are capable of reproducing the mean values of the injected residuals with excellent accuracy, regardless of the refinement level $N_x^{(H)}$. This indicates that the dominant trend of the residuals, which is the most critical component for error estimation purposes when having a time-averaged QoI, is consistently captured. However, discrepancies arise in the prediction of the standard deviation, which reflects the temporal variability of the residuals. While the models capture the general spread of the distributions, none of them fully recover the amplitude of fluctuations, particularly at finer resolutions where the variability is more pronounced.

The overprediction of fluctuations observed in all models can be attributed to limitations in their re-

spective configurations. For POD, this behavior stems from the low threshold of 65% energy retention used for mode selection. While this choice was intentional to avoid excessive dimensionality and to maintain computational efficiency, it inevitably restricts the number of modes retained, causing distortions in the representation of higher-order variability. In the case of CAE, the early stopping of the optimization procedure, combined with the trial-and-error tuning strategy, likely prevented the network from fully converging to a configuration capable of capturing the finer-scale residual fluctuations. Similarly, the ESN performance could have been improved by employing larger reservoir sizes, which would enhance its ability to resolve long-term temporal correlations. Nonetheless, since the primary QoI in this work is a time-averaged functional, the accurate reproduction of the mean values by all models remains the key outcome, making the imperfect representation of standard deviations less critical for the purposes of error estimation.

C.3. Performance Assessment of the Reconstructed Primal Solution

Figure C.4 compares the reconstructed temporal mean (solid lines) and standard deviation (shaded regions) with the actual fine solution (dashed black lines).

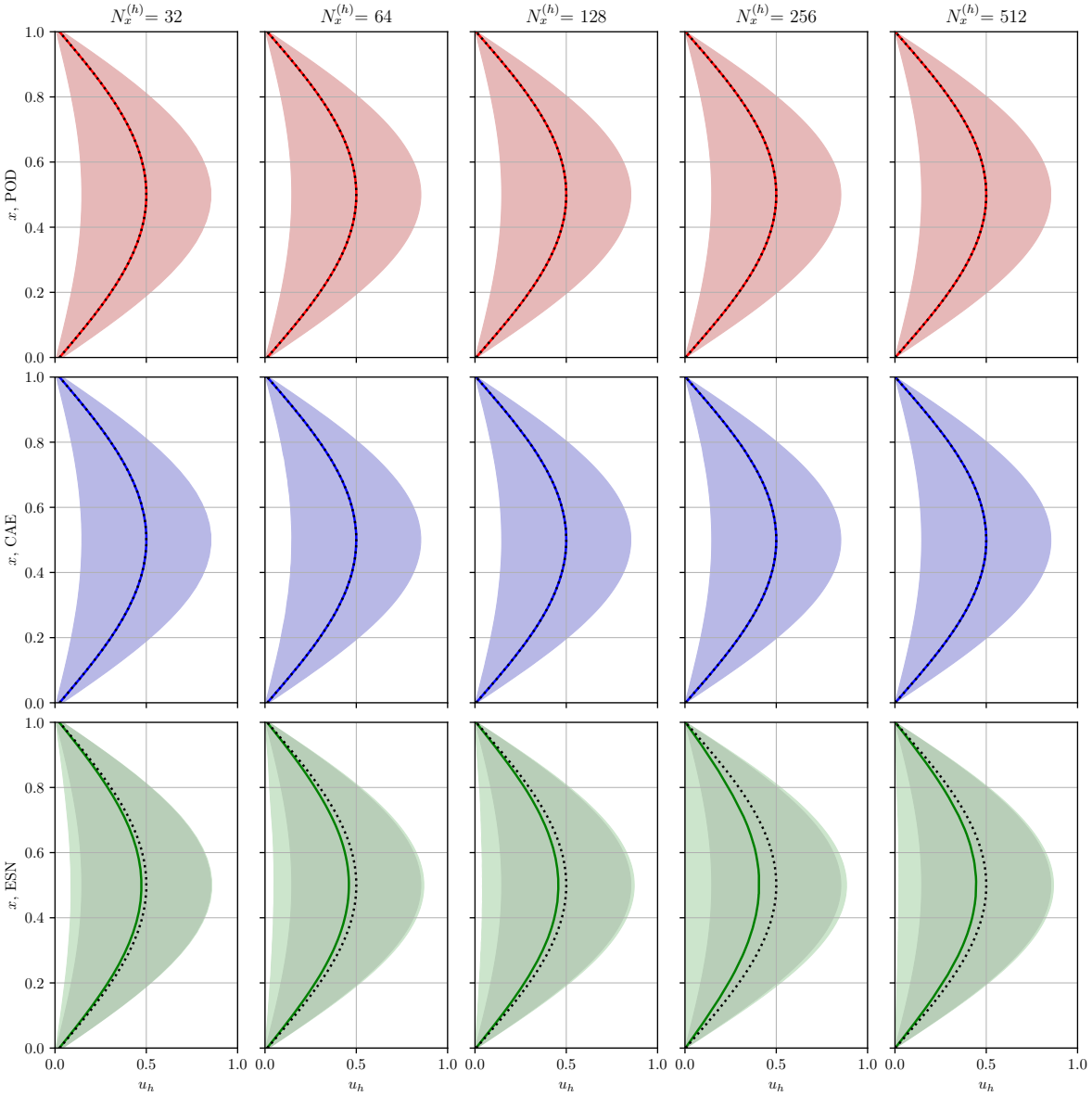


Figure C.4: Reconstructed 1D Burgers' MMS fine primal solution $\tilde{u}_h$ for all refinements $N_x^{(H)}$ using POD (top), CAE (middle), and ESN (bottom). The shaded regions are the actual solution u_h, with the dashed black lines indicating their temporal mean.

The results show that both POD and CAE are able to perfectly reconstruct not only the temporal mean but also the variance distribution across all refinement levels. This behavior directly reflects the simplicity of the manufactured primal solution, which is essentially low-rank and smooth, making it ideally suited to both the linear modal structure of POD and the nonlinear mapping of CAE with only a latent space size of $d_{\mathcal{Z}}$. In both cases, the surrogate representation achieves near-identical agreement with the reference data, independent of grid refinement.

In contrast, the ESN exhibits a gradual loss of accuracy as the refinement increases. The reconstructed temporal mean begins to deviate slightly from the reference, and the standard deviation is consistently underestimated. This trend highlights a limitation of the recurrent structure; while the ESN can capture

temporal dependencies, its effective capacity is tied to the reservoir size and input scaling. As the dimensionality of the fine solution increases with refinement, the fixed reservoir configuration becomes insufficient to resolve the full dynamics, leading to degraded performance.

C.4. Performance Assessment of the Adjoint Solution via Reconstructed Primal Solution

The results are presented in Figure C.5 and compared with the actual adjoint solution ψ in shaded regions and black dashed lines as their respective temporal standard deviation and mean. For both POD- and CAE-based surrogates, the reconstructed adjoint solution is nearly indistinguishable from the reference, both in terms of the temporal mean and standard deviation. This agreement was expected, since these models already reproduced the fine primal solution with high accuracy, thereby preserving the input fidelity required for the adjoint computation.

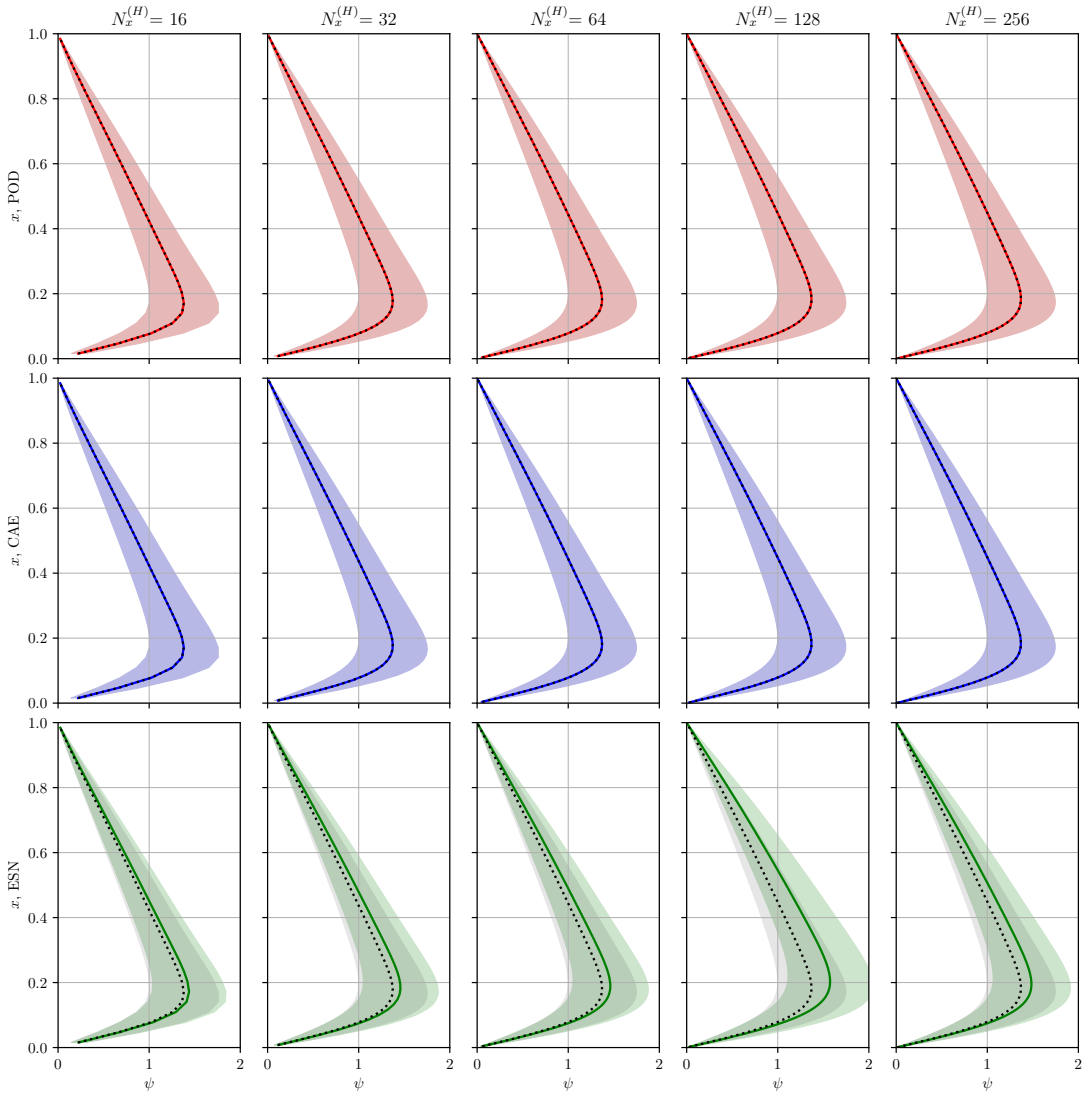


Figure C.5: Computed 1D MMS Burgers' adjoint solution $\tilde{\psi}$ using the reconstructed fine space solution $\tilde{u}_h$ for all refinements $N_x^{(H)}$ using POD (top), CAE (middle), and ESN (bottom). The shaded regions are the actual solution ψ, with the dashed black lines indicating their temporal mean.

In contrast, the ESN surrogate, which performed poorly in reconstructing the fine primal solution, also propagated this deficiency into the adjoint solution. The discrepancies manifest in both the temporal mean and the variance of $\tilde{\psi}$ across all refinement levels, reflecting the sensitivity of the adjoint system to inaccuracies in the input field. Although the deviations remain moderate, they clearly highlight that reliable adjoint prediction requires an accurate reconstruction of the fine primal solution, a condition not met by the ESN in this case.