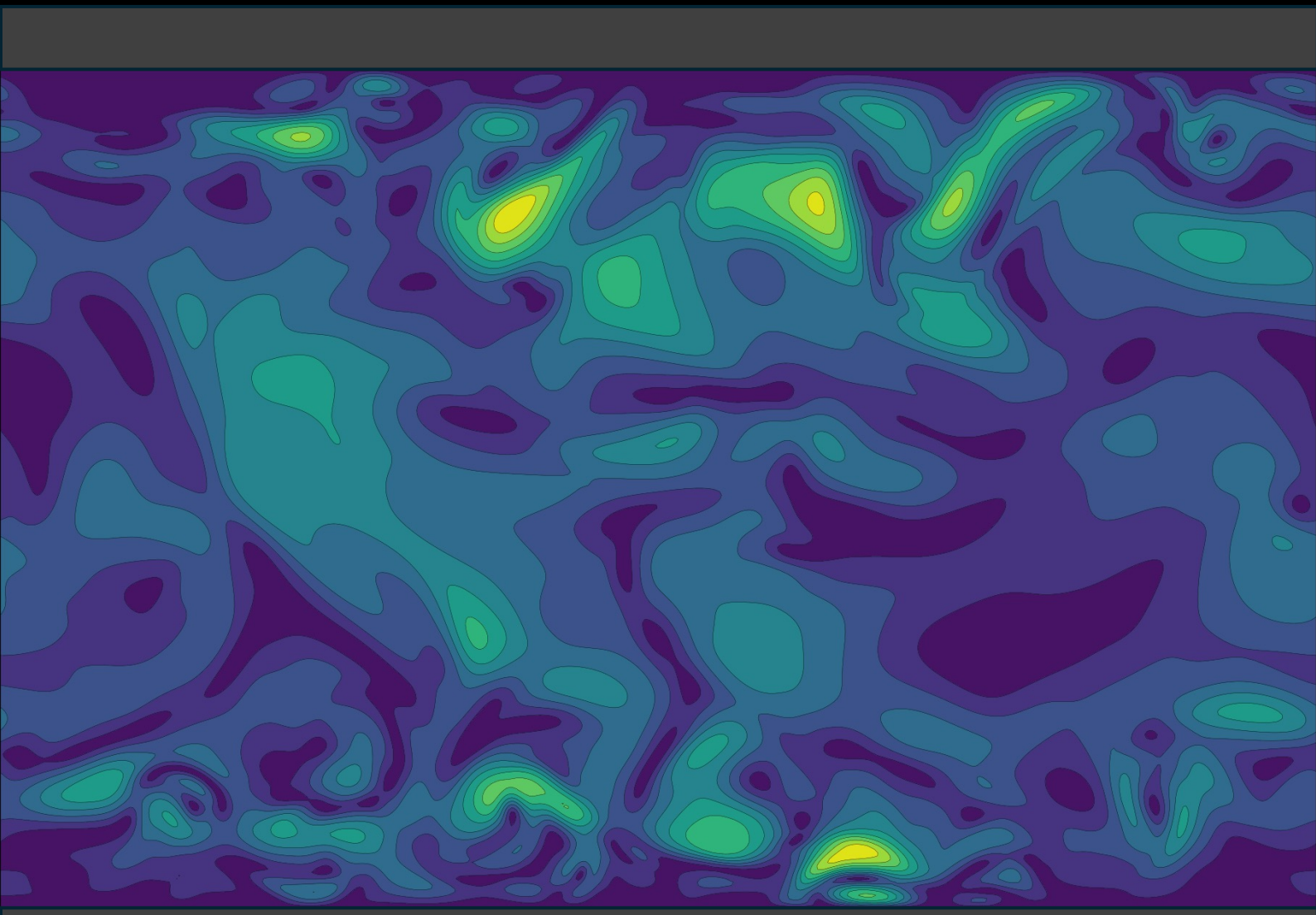
Adjoint-Based Error Estimation for Unsteady Problems

Deep Learning Techniques for Surrogate Modelling

João Colaço Romana



TU Delft

Delft University of Technology

Adjoint-Based Error Estimation for Unsteady Problems

Deep Learning Techniques for Surrogate Modelling

by

João Colaço Romana

to obtain the degree of Master of Science

at Delft University of Technology,

to be defended publicly on Thursday, 13th February 2025 at 2:00 PM.

Student Number:	5625866	
Project Duration:	February, 2024 - February, 2025	
Thesis Committee:	Dr. Steven J. Hulshoff,	TUDelft, Supervisor
	Ir. Thomas P. Hunter,	TUDelft, Supervisor
	Dr. Nguyen A. K. Doan	TUDelft, Chair
	Dr. Bo Y. Chen	TUDelft, Examiner
	Dr. Richard P. Dwight	TUDelft, Additional

TU Delft

Preface

This thesis marks the culmination of my Master's degree at the Faculty of Aerospace Engineering at Delft University of Technology. Throughout my studies in the Aerodynamics track, I developed a strong interest in machine learning applied to computational fluid dynamics. The increasing potential of these techniques and the rapid advancements in computational methods have opened up new opportunities to enhance the efficiency of aerodynamic simulations. This research aimed to contribute to the expanding knowledge in this exciting field by exploring how machine learning could be leveraged to reduce computational costs in unsteady adjoint-based error estimation.

I want to express my sincere gratitude to my supervisors, Dr. Steven J. Hulshoff and Ir. Thomas P. Hunter, for their exceptional guidance, patience and support throughout this research. Their expertise in the theoretical and practical aspects of the field has been invaluable in shaping the direction of this thesis. Their feedback during our productive weekly meetings was crucial to the success of this work from the initial proposal of research topics to the final discussion and review of the results. I am also grateful to my colleagues and fellow students at the university - a special thanks to Francisco and Marco, whose friendship and support have profoundly shaped my journey.

Furthermore, I want to thank my family for their constant encouragement and unwavering belief in my abilities. Their understanding during the challenging moments of my studies has been a source of strength and motivation to persevere - a heartfelt thank you to my parents for their unconditional love and support. I also want to thank the friends I made during my time at Delft, whose companionship and support helped me feel at home in a new country - a special thanks to Tiago, André, Mariana, João Tomás, José, Ricardo, João Gonçalo and João Tiago for their invaluable friendship. Finally, I am deeply grateful to my friends from my home country for their continued support and friendship - a special thanks to Miguel and Alexandre for always being there for me.

Abstract

Among error estimation methods, adjoint-based approaches are considered the most accurate but have the highest computational cost. For unsteady non-linear problems such as the Navier-Stokes equations, substantial storage requirements arise, as the full primal solution must be stored to solve the adjoint problem. As a result, careful management of storage resources is essential. The purpose of this research was twofold: to develop a surrogate model for the primal solution and to compute an accurate adjoint-based error estimate with the developed surrogate primal.

This study compared three methodologies to create a surrogate model of the primal solution while reducing the storage requirements of unsteady adjoint-based error estimation: a Convolutional AutoEncoder (CAE), an Echo State Network (ESN) and a combination of the first two, referred to as CAE-ESN. A benchmark Proper Orthogonal Decomposition (POD) served as a baseline for comparison with the deep learning techniques. Three numerical test cases were analyzed, where the finite element method was used for spatial discretization and implemented with the `FEniCS` computational framework. The first test case involved a manufactured solution to verify the implemented solver and methodologies. The remaining test cases used a turbulent channel flow dataset to force the unsteady viscous Burgers' equations in 1D (wall-normal component) and 2D (spanwise and wall-normal components).

For the manufactured solution, the ESN and POD outperformed the remaining approaches for the lowest and highest spatial resolutions, respectively. The success of the ESN was linked to its training being a linear regression problem. As established in previous studies, the smooth nature of the solution rendered the POD optimal. For the 1D case, the CAE was optimal, particularly for lower spatial resolutions. This method offered equivalent compression ratios to the POD while being more efficient in terms of computational cost and accuracy. In contrast, the ESN-based methods failed to accurately capture the error estimate, as they were not able to accurately compute the primal residual. However, these methods offered a higher compression than other approaches, along with a decrease in accuracy. Moreover, the error indicators produced by the ESN-based methods continued to effectively pinpoint the elements required for mesh adaptation. In the 2D case, only the POD and CAE were investigated. The ESN was excluded due to the high-dimensional nature of the test case. The CAE-ESN was not applied because of the limited time interval, which provided insufficient data for training. The CAE again proved optimal due to its efficiency and higher compression capabilities than the POD. While both methods provided accurate error estimates and indicator fields, the CAE outperformed the POD due to its superior compression. The CAE was also able to compute the adjoint solution and primal residual more accurately than the POD for most spatial resolutions. This research highlighted the potential for the CAE to outperform more conventional methods, such as POD.

Table of Contents

Preface i

Abstract iii

List of Figures xi

List of Tables xiii

List of Symbols xv

List of Abbreviations xvii

1 Introduction 1
- 1.1 Adaptive Mesh Refinement . . . 2
- 1.2 Error Estimation . . . 2
  - 1.2.1 Metric-Based Error Estimation . . . 3
  - 1.2.2 Residual-Based Error Estimation . . . 3
  - 1.2.3 Adjoint-Based Error Estimation . . . 4
- 1.3 Research Questions and Objectives . . . 4

2 Adjoint-Based Error Estimation: Description and Limitations 7
- 2.1 Primal Problem . . . 7
- 2.2 Adjoint Problem . . . 8
- 2.3 Primal and Adjoint Solutions in Unsteady Problems . . . 9
- 2.4 Adjoint-Based Error Estimation and Localization . . . 10
- 2.5 Limitations of Adjoint-Based Error Estimation . . . 11

3 Surrogate Modelling Techniques: Description and Applications 15
- 3.1 Modal Decomposition Techniques . . . 15
- 3.2 Deep Learning Techniques . . . 16
  - 3.2.1 Applications in Computational Fluid Dynamics . . . 17
- 3.3 Applications and Research in Adjoint-Based Error Estimation . . . 18

4 Framework and Methodology 19
- 4.1 Computational Framework . . . 19
- 4.2 Benchmark: Proper Orthogonal Decomposition . . . 20
- 4.3 Convolutional Autoencoder . . . 21
- 4.4 Echo State Network . . . 23
- 4.5 Convolutional Autoencoder - Echo State Network . . . 26

5 Framework Verification: Manufactured Solution 29
- 5.1 Primal Problem Formulation and Solution . . . 29
- 5.2 Primal Solution Surrogate Models . . . 31
  - 5.2.1 Proper Orthogonal Decomposition . . . 32
  - 5.2.2 Convolutional Autoencoder . . . 32
  - 5.2.3 Echo State Network . . . 34
  - 5.2.4 Convolutional Autoencoder - Echo State Network . . . 36
  - 5.2.5 Performance Assessment . . . 38
- 5.3 Adjoint Problem Formulation and Solution . . . 39
- 5.4 Adjoint-Based Error Estimation . . . 41

6 Results: Unsteady 1D Viscous Burgers' Equation 45
- 6.1 Primal Problem Formulation and Solution . . . 45
- 6.2 Primal Solution Surrogate Models . . . 49

6.2.1 Proper Orthogonal Decomposition . . . 49
6.2.2 Convolutional Autoencoder . . . 50
6.2.3 Echo State Network . . . 51
6.2.4 Convolutional Autoencoder - Echo State Network . . . 53
6.2.5 Performance Assessment . . . 55
6.3 Adjoint Problem Formulation and Solution . . . 56
6.4 Adjoint-Based Error Estimation . . . 58

7 Results: Unsteady 2D Viscous Burgers' Equation 63
7.1 Primal Problem Formulation and Solution . . . 63
7.2 Primal Solution Surrogate Models . . . 67
7.2.1 Proper Orthogonal Decomposition . . . 68
7.2.2 Convolutional Autoencoder . . . 70
7.2.3 Performance Assessment . . . 71
7.3 Adjoint Problem Formulation and Solution . . . 74
7.4 Adjoint-Based Error Estimation . . . 77

8 Conclusions and Future Work 81
8.1 Conclusions . . . 81
8.2 Future Work and Recommendations . . . 82

References 85

A Convolutional Autoencoder Architectures and Training 91
A.1 Manufactured Solution . . . 91
A.2 Unsteady 1D Viscous Burgers' Equation . . . 94
A.3 Unsteady 2D Viscous Burgers' Equation . . . 96

B Echo State Network Optimization 99
B.1 Manufactured Solution . . . 99
B.2 Unsteady 1D Viscous Burgers' Equation . . . 100

C Convolutional Autoencoder - Echo State Network Optimization 103
C.1 Manufactured Solution . . . 103
C.2 Unsteady 1D Viscous Burgers' Equation . . . 104

List of Figures

1.1 Adaptive Mesh Refinement Flowchart . . . 3

2.1 Primal and Adjoint Solution Procedure for Unsteady Problems (Dashed Arrows for Non-Linear Problems) . . . 10

3.1 Artificial Neural Network Architecture . . . 17

4.1 Computational Framework Flowchart . . . 20
4.2 Autoencoder Architecture . . . 22
4.3 Echo State Network Architecture . . . 24
4.4 Echo State Network Validation Strategies [40] . . . 25
4.5 Convolutional Autoencoder - Echo State Network Architecture . . . 27

5.1 Primal Solution and Relative Error of Manufactured Solution for $N_x = 32$ Spatial Elements 30
5.2 Quantity of Interest, Output Error and Root Mean Square Error of Manufactured Solution for Different Spatial Resolutions . . . 31
5.3 Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_x = 32$ Spatial Elements and Number of Modes Retained of Manufactured Solution for Different Spatial Resolutions . . . 32
5.4 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 33
5.5 Training Time and Mean Square Error for Convolutional Autoencoder of Manufactured Solution for Different Spatial Resolutions . . . 34
5.6 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks of Manufactured Solution for $N_x = 32$ Spatial Elements and $N_r = 1024$ Reservoir Neurons . . . 35
5.7 Compression Ratio, Training Time and Mean Square Error for Echo State Networks of Manufactured Solution for $N_x = 32$ Spatial Elements and Different Number of Reservoir Neurons . . . 35
5.8 Training Time and Mean Square Error for Echo State Networks of Manufactured Solution for Different Spatial Resolutions . . . 36
5.9 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 32$ Spatial Elements and $N_r = 1024$ Reservoir Neurons . . . 36
5.10 Compression Ratio, Training Time and Mean Square Error for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 32$ Spatial Elements and Different Number of Reservoir Neurons . . . 37
5.11 Computational Time and Mean Square Error for Echo State Networks Applied to Latent Space of Manufactured Solution for Different Spatial Resolutions . . . 37
5.12 Mean Relative Error of Surrogate Primal Solutions of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 38
5.13 Evaluation Metrics of Surrogate Primal Solutions of Manufactured Solution for Different Spatial Resolutions . . . 38
5.14 Adjoint Solution of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 40
5.15 Mean Relative Error of Surrogate Adjoint Solutions of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 40
5.16 Mean Square Error of Surrogate Adjoint Solutions of Manufactured Solution for Different Spatial Resolutions . . . 41
5.17 Primal Injected Residual of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 41

5.18 Mean Relative Error of Surrogate Primal Injected Residuals of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 42
5.19 Mean Square Error of Surrogate Primal Injected Residuals of Manufactured Solution for Different Spatial Resolutions . . . 42
5.20 Quantity of Interest and Error Estimates of Surrogate Solutions of Manufactured Solution for Different Spatial Resolutions . . . 43
5.21 Error Indicator Field and Time-Averaged Error Indicators of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 43
5.22 Mean Relative Error of Surrogate Error Indicator Fields of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 44
5.23 Time-Averaged Error Indicators of Surrogate Solutions of Manufactured Solution for $N_x = 32$ Spatial Elements (Finite Element Method Results in Grey) . . . 44

6.1 Mean Primal Velocity and Reynolds Stress of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements . . . 46
6.2 Mean Wall-Based Energy Spectrum of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 47
6.3 Quantity of Interest, Output Error and Root Mean Square Error of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 48
6.4 Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_y = 64$ Spatial Elements and Number of Modes Retained of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 49
6.5 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements . . . 50
6.6 Training Time and Mean Square Error for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 51
6.7 Averaged Prediction Horizon ($k = 2 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and $N_r = 512$ Reservoir Neurons . . . 52
6.8 Compression Ratio, Training Time and Prediction Horizon for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and Different Number of Reservoir Neurons . . . 52
6.9 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and $N_r = 512$ Reservoir Neurons . . . 53
6.10 Training Time and Prediction Horizon for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 53
6.11 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and $N_r = 256$ Reservoir Neurons 54
6.12 Computational Time and Prediction Horizon for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 54
6.13 Mean Surrogate Primal Velocities and Reynolds Stresses of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements (Finite Element Method Results in Grey) . 55
6.14 Mean Wall-Based Energy Spectrum for Surrogate Primal Solutions of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions (Finite Element Method Results Transparent) . . . 55
6.15 Evaluation Metrics of Surrogate Primal Solutions of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 56
6.16 Mean Adjoint Velocity and Reynolds Stress of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements . . . 57
6.17 Surrogate Mean Adjoint Velocities and Reynolds Stresses of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements (Finite Element Method Results in Grey) . 57
6.18 Mean Square Error of Surrogate Adjoint Velocities of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 58

6.19 Primal Injected Residual Temporal Mean and Standard Deviation of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 59
6.20 Primal Injected Surrogate Residual Temporal Mean and Standard Deviation of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions (Finite Element Method Results in Grey) . . . 59
6.21 Mean Square Error of Primal Injected Surrogate Residuals of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 59
6.22 Quantity of Interest and Error Estimates of Surrogate Solutions of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 60
6.23 Time-Averaged Error Indicators of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions . . . 60
6.24 Time-Averaged Error Indicators of Surrogate Solutions of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements (Finite Element Method Results in Grey) . . . 61

7.1 Primal Velocity Magnitude and Relative Error Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 64
7.2 Vorticity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 65
7.3 Mean Primal Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 65
7.4 Mean Wall-Based 1D Energy Spectra of the Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 66
7.5 Quantity of Interest, Output Error and Root Mean Square Error of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions . . . 67
7.6 Full Snapshot Matrix Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_z \times N_y = 128 \times 64$ Spatial Elements and Number of Modes Retained of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions . . . 68
7.7 2D Snapshot Matrix Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_z \times N_y = 128 \times 64$ Spatial Elements and Number of Modes Retained of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions . . . 69
7.8 Compression Ratio, Computational Time and Mean Square Error for Proper Orthogonal Decomposition Methods of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions . . . 69
7.9 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equations for $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 70
7.10 Training Time and Mean Square Error for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions . . . 71
7.11 Relative Error of Surrogate Primal Velocity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 71
7.12 Relative Error of Surrogate Vorticity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 72
7.13 Surrogate Mean Primal Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 72
7.14 Relative Error of Mean Wall-Based 1D Energy Spectrum for Surrogate Primal Solutions of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements (Spanwise Component on Top and Wall-Normal on Bottom) . . . 73
7.15 Evaluation Metrics of Surrogate Primal Solutions of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions . . . 73
7.16 Adjoint Velocity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 75
7.17 Mean Adjoint Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 75
7.18 Relative Error of Surrogate Adjoint Velocity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 76
7.19 Surrogate Mean Adjoint Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements . . . 76

7.20 Mean Square Error of Surrogate Adjoint Velocities of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions 77
7.21 Primal Injected Residual Temporal Mean and Standard Deviation of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements 77
7.22 Relative Error of Surrogate Primal Injected Residual Temporal Mean and Standard Deviation of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements (Spanwise Component on Top and Wall-Normal on Bottom) 78
7.23 Mean Square Error of Surrogate Primal Injected Residuals of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions 78
7.24 Quantity of Interest and Error Estimates of Surrogate Solutions of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions 79
7.25 Time-Averaged Error Indicators of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements 79
7.26 Time-Averaged Error Indicators of Surrogate Solutions of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 128$ Spatial Elements 80

A.1 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 8$ Spatial Elements 92
A.2 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 16$ Spatial Elements 93
A.3 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 64$ Spatial Elements 93
A.4 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 128$ Spatial Elements 93
A.5 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 16$ Spatial Elements 95
A.6 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 32$ Spatial Elements 95
A.7 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 128$ Spatial Elements 96
A.8 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 256$ Spatial Elements 96
A.9 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equation for $N_z \times N_y = 32 \times 16$ Spatial Elements 96
A.10 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equation for $N_z \times N_y = 64 \times 32$ Spatial Elements 97
A.11 Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equation for $N_z \times N_y = 256 \times 128$ Spatial Elements 97

B.1 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 8$ Spatial Elements and $N_r = 1024$ Reservoir Neurons 99
B.2 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 16$ Spatial Elements and $N_r = 1024$ Reservoir Neurons 99
B.3 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 64$ Spatial Elements and $N_r = 1024$ Reservoir Neurons 100
B.4 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 128$ Spatial Elements and $N_r = 1024$ Reservoir Neurons 100
B.5 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 16$ Spatial Elements and $N_r = 512$ Reservoir Neurons 100

B.6 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 32$ Spatial Elements and $N_r = 512$ Reservoir Neurons 101
B.7 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 128$ Spatial Elements and $N_r = 512$ Reservoir Neurons 101
B.8 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 256$ Spatial Elements and $N_r = 512$ Reservoir Neurons 101

C.1 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 8$ Spatial Elements and $N_r = 1024$ Reservoir Neurons 103
C.2 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 16$ Spatial Elements and $N_r = 1024$ Reservoir Neurons 103
C.3 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 64$ Spatial Elements and $N_r = 1024$ Reservoir Neurons 104
C.4 Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 8$ Spatial Elements and $N_r = 128$ Reservoir Neurons 104
C.5 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 16$ Spatial Elements and $N_r = 512$ Reservoir Neurons 104
C.6 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 32$ Spatial Elements and $N_r = 512$ Reservoir Neurons 105
C.7 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 128$ Spatial Elements and $N_r = 512$ Reservoir Neurons 105
C.8 Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 256$ Spatial Elements and $N_r = 512$ Reservoir Neurons 105

List of Tables

5.1 Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 32$ Spatial Elements . . . 33

6.1 Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 64$ Spatial Elements . . . 50

7.1 Convolutional Autoencoder Architecture of Unsteady 2D Viscous Burger's Equation . . 70

A.1 Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 8$ Spatial Elements . . . 91
A.2 Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 16$ Spatial Elements . . . 91
A.3 Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 64$ Spatial Elements . . . 92
A.4 Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 128$ Spatial Elements . . . 92
A.5 Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 16$ Spatial Elements . . . 94
A.6 Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 32$ Spatial Elements . . . 94
A.7 Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 128$ Spatial Elements . . . 94
A.8 Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 256$ Spatial Elements . . . 95

List of Symbols

a	Degrees of Freedom
b	Bias Term
E	Spectral Energy Density
f	Source Term or Activation Function
g	Quantity of Interest Function
h	Fine Space
H	Coarse Space
i	Imaginary Unit
I	Time Domain or Injection Operator or Identity Matrix
J	Quantity of Interest
k	Prediction Horizon Threshold or Wavenumber
L	Differential Operator or Left Singular Matrix or Space Domain Length
M	Optimal Rank or Latent Space Dimension
N	Number of Elements
q	Weighting Function
r	Reservoir State
R	Residual or Right Singular Matrix or Reservoir States Concatenation
Re	Reynolds Number
t	Time Coordinate
T	Time Domain Length
u, v, w	Velocity Components
U	Snapshot Matrix
V	Function Space
W	Weights
x	Space Coordinate or Streamwise Direction
y	Layer State or Wall-Normal Direction
Y	Output States Concatenation
z	Latent Space State or Spanwise Direction
α	Leaking Rate
β	Tikhonov Regularization Parameter
γ	Learning Rate
δ	Infinitesimal Variation or Channel Half-Height
Δ	Numerical Step
ε	Error Indicator
ν	Kinematic Viscosity
ρ	Spectral Radius or Density
σ	Standard Deviation
Σ	Diagonal Singular Value Matrix
ϕ	Basis Functions
ψ	Adjoint Solution
ω	Vorticity
Ω	Space Domain

List of Abbreviations

AMR	Adaptive Mesh Refinement
ANN	Artificial Neural Network
BC	Boundary Condition
BO	Bayesian Optimization
CAE	Convolutional AutoEncoder
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
CR	Compression Ratio
DMD	Dynamic Mode Decomposition
DNS	Direct Numerical Simulation
ESN	Echo State Network
ESP	Echo State Property
FEM	Finite Element Method
FFT	Fast Fourier Transform
FVM	Finite Volume Method
GP	Gaussian Process
HLE	Highest Lyapunov Exponent
HPC	High-Performance Computing
IC	Initial Condition
LES	Large Eddy Simulation
LT	Lyapunov Time
ML	Machine Learning
MMS	Method of Manufactured Solution
MSE	Mean Square Error
NRMSE	Normalized Root Mean Square Error
PDE	Partial Differential Equation
PH	Prediction Horizon
POD	Proper Orthogonal Decomposition
QoI	Quantity of Interest
RANS	Reynolds-Averaged Navier Stokes
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
ROM	Reduced-Order Model
SVD	Singular Value Decomposition
TCF	Turbulent Channel Flow
TKE	Turbulent Kinetic Energy
1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional

1

Introduction

Over the past fifteen years, the use of Computational Fluid Dynamics (CFD) has been limited as the use of Reynolds-Averaged Navier-Stokes (RANS) simulations has been prevalent. This limitation is largely attributed to the incapacity of current-generation methodologies to accurately predict the dynamics of unsteady turbulent flows [1]. Substantial computational advancements are necessary to transition towards Large Eddy Simulation (LES) or Direct Numerical Simulation (DNS), which require significant increases in computational and memory resources. The computational cost associated with LES of wall-bounded flows is understood to scale approximately with $Re^{2.4}$ when wall models are not used [2]. The Reynolds number is defined as the ratio of inertial forces to viscous forces:

$$Re = \frac{uL}{\nu} \tag{1.1}$$

where u represents a characteristic velocity, L is a characteristic length scale and ν is the fluid's kinematic viscosity. Conversely, the computational cost of DNS scales with Re^3 since all turbulent scales have to be resolved. The scaling with the Reynolds number poses a significant challenge to accurately compute unsteady high-Reynolds flows with these methodologies.

The efficacy of CFD in aerospace system design and analysis relies heavily on the potency and accessibility of contemporary High-Performance Computing (HPC) systems [3]. In recent years, this domain has witnessed a shift towards increasing levels of parallelism, as low-cored supercomputers are being replaced by highly-parallelized clusters constituted by commodity hardware [1]. Although computer capability has experienced exponential growth in alignment with Moore's law in the past, it is reaching physical constraints, leading to escalating costs and diminishing returns on investment [4]. Alongside this trend, a growing gap has emerged between the computational capabilities of processors and the bandwidth available for data exchange to and from main memory [5]. Furthermore, memory capacity enhancement has progressed at a much slower rate over the past decades [1]. Therefore, the limitations of current HPC systems require innovative techniques to address the demanding computational demands of LES and DNS, particularly for unsteady turbulent flows. Given the stagnant performance of current processors, improvements should be focused on the optimal management of computational resources.

Proposed methodologies to make unsteady aerodynamic simulations more accessible include Adaptive Mesh Refinement (AMR). This method reduces computational costs for a fixed accuracy by the dynamic generation of a computational mesh with an appropriate refinement level [6]. Error estimation and mesh adaptation are key steps of AMR: the first involves estimating the solution's error whereas the latter involves the refinement or coarsening of the computational mesh based on the computed error estimate. Within the context of error estimation, adjoint-based error estimation is considered the most accurate technique [7] as it focuses on evaluating the numerical error concerning a chosen Quantity of Interest (QoI), such as lift or drag. This framework leverages both solutions of the primal and adjoint problems. However, one drawback of this approach is the significant computational resources needed

to solve the adjoint problem as a refined adjoint solution is required. Moreover, for unsteady non-linear problems like the Navier-Stokes equations, substantial storage requirements arise from the necessity to store the entire primal solution to solve the adjoint problem. The local linearization of the adjoint problem requires the primal state at every time step to obtain the adjoint solution. The high-dimensional datasets produced by LES and DNS require a trade-off between data storage, computational cost and solution accuracy for an unsteady adjoint-based error estimation framework to be implemented [6]. The methods proposed within this research aimed to address the storage challenges posed by adjoint-based error estimation when applied to unsteady non-linear problems with the use of Machine Learning (ML). These were compared to conventional modal decomposition techniques.

The introductory chapter of this report offers an overview of AMR, emphasizing error estimation and providing the necessary background to understand the research questions and objectives. In chapter 2, the pertinent literature surrounding adjoint-based error estimation is delineated, highlighting its description and limitations. A higher focus is given to its application to unsteady non-linear problems such as the Navier-Stokes equations. In chapter 3, a description and applications of surrogate modelling methods are presented, focusing on techniques relevant to data compression and computational efficiency. In chapter 4, the computational framework implemented is described along with the methodologies proposed to answer the research questions. In chapter 5, the first numerical test case in the unsteady one-dimensional (1D) viscous Burgers' equation is presented, where the implemented solver and methodologies were verified with a manufactured solution. In chapter 6, a more complex case in the framework of adjoint-based error estimation is explored: a solution driven by DNS data to produce the wall-normal velocity from Turbulent Channel Flow (TCF) with the unsteady 1D viscous Burgers' equation. The investigation is then conducted to a higher-dimensional test case in chapter 7, where the same DNS dataset was used to force the unsteady two-dimensional (2D) viscous Burgers' to produce the spanwise and wall-normal velocity components from TCF. Finally, chapter 8 encapsulates the conclusions drawn from the obtained results and recommendations for future work to further evaluate the methodologies proposed and the framework implemented.

1.1. Adaptive Mesh Refinement

Given an initial mesh with an appropriate base level of refinement, AMR offers a way to minimize the computational cost of numerical solutions for a fixed accuracy. By dynamically generating meshes with appropriate refinement levels, the accuracy of calculations is increased automatically, thus not requiring manual intervention [8].

The procedural framework of AMR is presented in Figure 1.1. It is assumed that an element-based method is employed in the spatial discretization, such as the Finite Element Method (FEM) or the Finite Volume Method (FVM). The process is initiated by constructing an initial mesh while defining a predetermined error tolerance, which dictates the desired level of accuracy for the simulation. Following this initial step, an iterative process unfolds, starting with the computation of the primal solution with a suitable numerical method, along with error estimation to assess solution accuracy. If the error tolerance criterion is not satisfied, a step for error localization is activated to evaluate the contribution of local elements in the computational grid to the overall error estimation. Based on this evaluation, mesh adaptation is executed, involving mesh refinement or coarsening in specific grid areas, followed by a new iteration. This iterative cycle continues until the error tolerance criterion is met, culminating in a final adapted mesh with the desired solution accuracy and a minimized computational cost.

Error estimation and mesh adaptation are critical stages in this procedure, and various methods have been investigated in prior literature [9–11]. However, this research focused exclusively on error estimation, thereby excluding mesh adaptation from its scope.

1.2. Error Estimation

Error estimation methods in numerical simulations can be classified into *a priori* and *a posteriori*. The former estimates the error without relying on the numerical primal solution, drawing upon user experience, mesh topology and mesh positioning. An example of a viable *a priori* approach was presented by Léonard [12], where a multi-grid algorithm was used to mitigate modelling errors introduced by LES. However, grids generated solely based on *a priori* estimates are sub-optimal in terms of computational

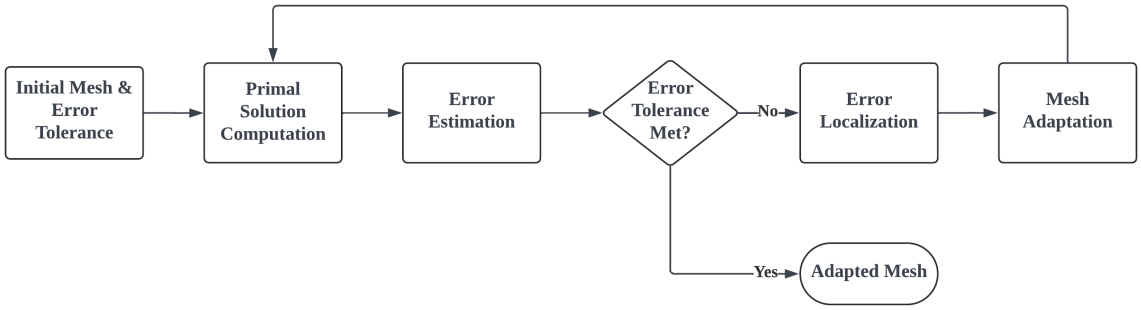


Figure 1.1: Adaptive Mesh Refinement Flowchart

cost, as they treat all local flow features with equal importance. Conversely, *a posteriori* error estimation approaches are based on the computed numerical solution. Within this framework, various methods can be delineated based on the chosen indicator to drive the mesh adaptation procedure. The most prevalent *a posteriori* approaches in the literature are metric-based, residual-based and adjoint-based error estimation.

1.2.1. Metric-Based Error Estimation

Metric-based methods for error estimation utilize a selected metric, operating on the premise that regions containing diverse flow features harbour a higher proportion of numerical errors. Two primary approaches are discernible within this framework: feature-based and criterion-based error estimation. Feature-based error estimation relies on solution gradients, solution curvature or identified solution features to guide the mesh adaptation process. Advanced feature-based estimators may incorporate more intricate flow variables, such as the percentage of the local resolved Turbulent Kinetic Energy (TKE) [13]. In contrast, criterion-based error estimation establishes a criterion that may lack direct physical significance but aims to encapsulate more intricate phenomena within the numerical simulation, thereby providing a higher degree of interpretability in the metric formulation. Examples of criterion-based error estimation were provided by Benard et al. [14] in an LES context, where two different criteria were used to either ensure a correct discretization of the mean solution field or to ensure sufficient resolution of the turbulent scales of motion.

These techniques boast lower computational overhead compared to residual-based and adjoint-based error estimation methods as they only require the computation of one feature-based or criterion-based metric. Nevertheless, they can occasionally result in over-refinement of certain features while inadequately addressing others, which leads to worse results than those provided by adjoint-based error estimation [15]. Furthermore, their efficacy is contingent upon the specific case and may prove unsuitable for more complex scenarios [16]. Another drawback stems from the substantial reliance on the selected metric. Typically, numerous indicators are available, which necessitate a prior understanding of the problem to select the most appropriate error indicator.

1.2.2. Residual-Based Error Estimation

Truncation error, defined as the difference between the exact solution and its discrete approximation, is a measure of the contribution of a local element discretization to the discretization error. Thus, residual-based indicators pinpoint regions of the flowfield where the discrete solution fails to represent the continuous governing Partial Differential Equation (PDE) within a specified tolerance. The residual is defined as a function that is zero when the PDE is satisfied. Residual-based error estimation methods exhibit a higher computational cost than metric-based techniques as they require the computation of the residual of the primal PDE. Richardson extrapolation methods serve as an example of residual-based error estimation, offering accurate estimations when grids are sufficiently fine [17]. For coarser meshes, adaptation processes can be guided by the leading truncation error terms of a Taylor series expansion of the solution's variables [11].

In his investigations, Roy [11] explored four distinct strategies to steer mesh adaptation: solution gra-

dients, solution curvature, Richardson extrapolation and leading truncation errors. Ultimately, mesh adaptation based on leading truncation errors yielded the smallest discretization errors compared to the remaining methods. Nonetheless, error estimates derived solely from residual or discretization errors may fail to account for propagation effects inherent in convection-dominated problems [10]. In such scenarios, residual-based error estimates might not detect the significant influence of critical regions upstream of the considered location.

1.2.3. Adjoint-Based Error Estimation

Adjoint-based error estimation, referred to as output or goal-oriented error estimation, evaluates the local error contribution of each element with respect to a designated QoI. Consequently, it facilitates tailored mesh adaptation based on the specific objective of the simulation. In this paradigm, the adjoint solution computation addresses propagation effects when quantifying the sensitivity of local residual errors to the selected QoI. Adjoint-based error estimation is generally regarded as the most accurate error estimation method however, it entails the highest computational cost [7]. The computational overhead is due to the additional computation of the adjoint solution when compared to residual-based error estimation.

Adjoint-based error estimation has demonstrated successful application in the context of steady flows for a variety of test cases. Venditti and Darmofal [18–20] employed adjoint-based error estimation in a variety of inviscid and viscous aerodynamic scenarios in 1D and 2D, using a FVM formulation. Similarly, Park [21] applied this method to the three-dimensional (3D) transonic flow over the ONERA M6 wing, while also employing an FVM discretization. Furthermore, adjoint-based error estimation has been effectively coupled with a continuous Galerkin FEM in steady-state test cases. For instance, Becker et al. [22] investigated low-Reynolds flow over a 2D cylinder, while Bhatia and Beran [23] explored transonic flow over a 2D airfoil. In the latter case, it was observed that residual-based error estimators tended to identify all flow features around the airfoil, while adjoint-based indicators focused solely on regions crucial for enhancing the quality of the QoI, rendering them more efficient. Also, adjoint-based error estimation has been applied within a discontinuous Galerkin formulation to study the steady flow over an airfoil at different Mach numbers, employing a hybridized scheme [24]. Furthermore, an unsteady procedure of the mesh adaptation process has been applied in the context of RANS simulations for a 2D multi-element airfoil flow [25]. This procedure estimated the error and adapted the mesh at every time step, aiming to eliminate the need to obtain converged steady-state solutions before performing error estimation.

In the context of unsteady problems, adjoint-based error estimation represents an active area of research. Nevertheless, its efficacy has been demonstrated across various test cases. For instance, Kast [26] employed this method in Navier-Stokes simulations on deformable 2D domains, employing an arbitrary Lagrangian-Eulerian mapping technique. Moreover, adjoint-based error estimation has been applied to 3D unsteady scenarios using a continuous Galerkin FEM discretization, as demonstrated in studies concerning high-Reynolds wing-body flow [27] and high-lift model [28]. Adjoint-based error estimation has also been integrated into a discontinuous Galerkin framework for transient problems, as evidenced by studies focusing on the flow over a cylinder in 2D and 3D [6, 29]. Lastly, adjoint-based error estimation has been combined with Artificial Neural Networks (ANNs) for determining the optimal anisotropy in a computational 2D mesh for a multi-element airfoil flow [30].

Several challenges exist regarding adjoint-based error estimation. In unsteady non-linear problems such as the Navier-Stokes equations, the local linearization of the adjoint PDE requires the primal solution at each time step to compute the adjoint solution. LES and DNS produce high-dimensional data which incur significant storage requirements to solve the adjoint problem in unsteady turbulent flows [6]. One solution is to store the primal solution on disk but it leads to increased storage requirements and computational time [31]. Therefore, careful management of computational storage, computational cost and primal solution accuracy is required for unsteady adjoint-based error estimation to be implemented in these methodologies.

1.3. Research Questions and Objectives

The research problem was articulated within the domain of error estimation, therefore excluding mesh adaptation from this research. The focus was on addressing the substantial storage demands inher-

ent in unsteady adjoint-based error estimation for aerodynamic problems. One potential approach to alleviate these requirements involves compressing the primal solution after its computation and subsequently reconstructing it as necessary for solving the adjoint problem. One important concept in compression methods is the Compression Ratio (CR), defined as the ratio of the original data size to the compressed size:

$$\text{CR} = \frac{\text{Original Data Size}}{\text{Compressed Data Size}} \tag{1.2}$$

Compression techniques can be classified into two main categories: lossless and lossy techniques. Lossless methods ensure that no information is lost during the data reduction process, albeit they may require significant memory resources. For instance, a switched prediction lossless coding scheme demonstrated a modest increase in compression time to achieve better compression capabilities [32]. In contrast, lossy compression techniques involve the loss of some information during the data reduction process. However, these techniques have the potential to provide higher compression capabilities while losing the ability to retrieve the entirety of the original data [33].

Within lossy compression techniques, surrogate models, more specifically Reduced-Order Models (ROMs), have been widely used to create lower-order and computationally inexpensive representations of higher-order systems [34]. Conventional modal decomposition techniques accomplish this representation with a linear decomposition of data, such as Proper Orthogonal Decomposition (POD) which uses orthogonal modes. On the other hand, deep learning techniques, a subset of ML, can provide a non-linear decomposition of datasets with the use of non-linear activation functions [35]. For non-linear problems, such as turbulent flows, modal decomposition techniques inefficiently represent the flowfield due to their inherent linear nature, thus requiring a large number of modes for an accurate representation [36]. On the other hand, deep learning methods have shown promising results in reducing the number of modes to accurately describe turbulent flows [37–39]. Recent advancements in deep learning applied to CFD have showcased the capability of creating a surrogate model of the primal solution, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). More specifically, a Convolutional AutoEncoder (CAE) has shown promise in accurately representing turbulent flows [36], while an Echo State Network (ESN) offers the advantage of significantly reduced training times compared to other RNN architectures [40]. Moreover, a hybrid approach, known as the CAE-ESN, holds the potential to enhance compression capabilities further [39].

By leveraging recent advancements in deep learning techniques within the framework of adjoint-based error estimation, it is feasible to develop methodologies for mitigating the demanding storage requirements associated with unsteady aerodynamic problems. The CAE, ESN and CAE-ESN were investigated in this research to create a surrogate model of the primal solution in the context of unsteady adjoint-based error estimation. POD served as a baseline comparison with the deep learning techniques. Therefore, the research question was formulated as follows:

Research Question

How can deep learning techniques be employed to efficiently compress the primal solution while ensuring an accurate adjoint-based error estimate for unsteady simulations?

Additionally, secondary research questions were formulated:

1. What is the compression rate achieved in the primal solution compression?
2. What is the computational cost involved in the primal solution compression?
3. How accurate is the surrogate primal solution representation?
4. How efficient is the surrogate model of the primal solution?
5. How accurate is the surrogate error estimate computed?

The first four items aimed to evaluate the performance of the deep learning technique in creating a surrogate model for the primal solution. The first three metrics focused on individual aspects of performance, while the fourth provided a global evaluation. The final item was designed to assess the

surrogate model's effectiveness in adjoint-based error estimation. These research questions were formulated for the deep learning techniques proposed whereas the POD served as a baseline method for comparison.

Furthermore, the following research objective was formulated:

Research Objective

Implement and evaluate the use of convolutional and recurrent artificial neural networks in inexpensive, compact and accurate adjoint-based error estimation for unsteady simulations.

The proposed research objective hinged on the capability to address the identified need by solving the adjoint problem with a surrogate model of the primal solution, facilitated by convolutional and recurrent ANNs. This approach held the promise of significantly reducing the storage requirements associated with adjoint-based error estimation in unsteady simulations while upholding the accuracy of the error estimate.

Based on the primary research objective, the following secondary research objectives were formulated to serve as guidelines for the research:

1. Evaluate the applications and limitations of state-of-the-art adjoint-based error estimation in CFD.
2. Evaluate the applications and limitations of state-of-the-art convolutional and recurrent ANNs in CFD.
3. Evaluate the applications and limitations of state-of-the-art convolutional and recurrent ANNs in adjoint-based error estimation.
4. Evaluate the use of convolutional and recurrent ANNs to create a surrogate model of the primal solution.
5. Evaluate the error introduced in adjoint-based error estimation by predicting the primal solution with convolutional and recurrent ANNs.

The first item is assessed in this introductory chapter and chapter 2, whereas the second and third items are evaluated in chapter 3. The last two items are assessed in the remaining chapters across the methodologies proposed and the test cases considered, which involved implementing, validating and evaluating the proposed framework.

2

Adjoint-Based Error Estimation: Description and Limitations

In this chapter, a description and limitations of the utilization of adjoint-based error estimation are explored along with the presentation of literature examples. Particular focus is placed on unsteady non-linear problems. The aim is to elucidate the concepts necessary to understand the research problem.

First, the primal problem is presented along with the definition of the user-defined Quantity of interest (QoI). Additionally, the continuous technique is derived to obtain the adjoint or dual problem for a chosen QoI from the primal problem. The discussion is further extended to non-linear problems where a local linearization of the adjoint problem is necessary. Furthermore, the primal and adjoint solutions are presented in the unsteady context of linear and non-linear problems. Error estimation and localization are then described within the framework of adjoint-based error estimation when employing an element-based numerical method for spatial discretization. The chapter concludes by examining the different limitations of adjoint-based error estimation, supplemented by examples from the literature.

2.1. Primal Problem

As a starting point, a primal continuous Partial Differential Equation (PDE) is considered with given Boundary Conditions (BCs) in a space domain Ω:

$$\begin{cases} Lu = f, & x \in \Omega \\ \text{Primal BCs}, & x \in \partial\Omega \end{cases} \tag{2.1}$$

where x is the spatial coordinate, L is a differential operator, u is the continuous primal solution and f is a source term. Additionally, a continuous residual $R(u)$ can be defined from the primal PDE:

$$R(u) = Lu - f \tag{2.2}$$

It is described as a function that is zero when the strong form of the primal PDE is satisfied. Furthermore, let J be a QoI of the following form for any desired choice of the function $g(x)$:

$$J = \int_{\Omega} g(x)u(x)\, d\Omega \tag{2.3}$$

Instead of being focused on the accurate computation of the entire primal solution, the focus is the accurate computation of the QoI. For many combinations of differential operators and BCs, solving the primal PDE analytically is impractical, necessitating the use of numerical methods to approximate the solution. Consequently, these problems must be discretized using an appropriate spatial discretization

scheme. Examples of spatial discretization schemes are the Finite Element Method (FEM) or the Finite Volume Method (FVM). If the primal PDE is unsteady, an Initial Condition (IC) has to be imposed and an appropriate temporal scheme must also be applied.

2.2. Adjoint Problem

Discrete and continuous techniques can be used to derive the adjoint solution from the primal problem for a specific scalar output J. The discrete method is developed directly from a set of discrete residuals defined from the discretized primal problem. Conversely, the continuous technique is derived from the continuous primal equation and discretized afterwards. In adjoint-based error estimation and also in design optimization, both the discrete approach [18–20] and the continuous method [7, 41, 42] have been successfully applied. It has been demonstrated that the discrete technique leads to more accurate error estimation for finer grids, while the continuous technique yields better results when the primal and adjoint solutions are well resolved [16]. Nonetheless, the discrete method involves the computation of a Jacobian matrix, thus requiring automatic code differentiation or analytical differentiation [26]. Also, within this framework. adjoint consistency imposes additional difficulties [43]. As a result, the continuous technique is adopted in this research. A more detailed derivation of the continuous and discrete techniques is given in [26].

The adjoint differential operator L^* can be defined from the adjoint identity as the relation between the primal differential operator L and the continuous adjoint solution ψ:

$$\int_\Omega (Lu)\,\psi \; d\Omega = \int_\Omega u\,(L^*\psi) \; d\Omega, \qquad \forall\, u, \psi \in V \tag{2.4}$$

where V is a function space where the L^2 inner product is defined. The adjoint solution represents the sensitivity of the chosen QoI concerning an infinitesimal perturbation in the residual, thus the following equation has to be satisfied:

$$J'(\delta u) = \int_\Omega \psi R'(\delta u) \; d\Omega, \qquad \forall \text{ permissible } \delta u \tag{2.5}$$

Equation 2.5 is the generalized form of the continuous adjoint equation, where δu is an infinitesimal variation of the primal solution and a prime indicates the sensitivity of a quantity with respect to the solution, also denoted as Fréchet linearization. The infinitesimal variation of the primal solution must satisfy any BC imposed on the primal problem. For instance, if a Dirichlet BC is imposed on the primal problem, then the value of the primal solution is fixed, implying no variation ($\delta u = 0$) is allowed in that specific boundary.

In the context of non-linear problems, the adjoint equation arises from a local linearization about a specific primal state u_0 which is treated as fixed. Consequently, the continuous adjoint equation is expressed as follows:

$$J'[u_0](\delta u) = \int_\Omega \psi R'[u_0](\delta u) \; d\Omega, \qquad \forall \text{ permissible } \delta u \tag{2.6}$$

The residual sensitivity R' can be defined based on the definition of the residual given in Equation 2.2 and the QoI sensitivity J' based on the chosen QoI used in Equation 2.3. By employing integration by parts on the right-hand side of the continuous adjoint equation and juxtaposing it with the QoI sensitivity, one can derive the adjoint operator. The adjoint BCs are determined by associating the primal BCs to the terms obtained through integration by parts. The resulting adjoint PDE is given by:

$$\begin{cases} L^*\psi = g, & x \in \Omega \\ \text{Adjoint BCs}, & x \in \partial\Omega \end{cases} \tag{2.7}$$

Similar to the primal problem, this PDE is then discretized with an appropriate numerical method to obtain the adjoint solution. It is important to note that the adjoint PDE is always linear, regardless of

whether the primal problem is linear or non-linear. Also, the adjoint problem is defined based on the selected QoI as the adjoint forcing term is given by the user-defined function $g(x)$.

2.3. Primal and Adjoint Solutions in Unsteady Problems

Research on adjoint-based error estimation for unsteady problems has been relatively constrained, primarily owing to implementation hurdles and computational costs [6, 44]. In unsteady problems, as the primal solution marches forward in time, the adjoint equation marches backwards in time, functioning as a sensitivity problem. As a demonstration of the backwards time marching, a linear advection primal PDE in a one-dimensional (1D) space-time domain $\Omega : [x_L, x_R] \times I : [0, T]$ with a Dirichlet BC imposed on the left boundary and a homogeneous IC is considered:

$$\begin{cases} \dfrac{\partial u}{\partial t} + \dfrac{\partial u}{\partial x} = f, & (x,t) \in \Omega \times I \\ u(x_L, t) = 0, & t \in I \\ u(x, 0) = 0, & x \in \Omega \end{cases} \tag{2.8}$$

where t is the temporal coordinate. In many problems, a time-averaged QoI $\bar{J}$ is considered as steady-state QoIs can fail to accurately capture statistically-steady transient solutions [7, 45]:

$$\bar{J} = \frac{1}{T} \int_I \int_\Omega g(x,t) u(x,t) \, d\Omega dI \tag{2.9}$$

Selecting $g(x,t) = 1$ results in a QoI representing the mean solution over space and time. The continuous technique, given in Equation 2.5 for linear problems, is employed to derive the adjoint PDE. First, the continuous residual can be defined from the primal problem:

$$R(u) = \frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - f \tag{2.10}$$

The residual sensitivity is defined given an infinitesimal variation of the primal solution δu:

$$R'(\delta u) = \frac{\partial (\delta u)}{\partial t} + \frac{\partial (\delta u)}{\partial x} \tag{2.11}$$

The adjoint operator L^* can then be derived using integration by parts:

$$\begin{aligned} \int_I \int_\Omega \delta u \, d\Omega dI &= \int_I \int_\Omega \psi \left(\frac{\partial (\delta u)}{\partial t} + \frac{\partial (\delta u)}{\partial x} \right) \, d\Omega dI \\ &= \int_I \int_\Omega \left(-\frac{\partial \psi}{\partial t} - \frac{\partial \psi}{\partial x} \right) (\delta u) \, d\Omega dI + \left[\int_\Omega \psi (\delta u) \, d\Omega \right]_{\partial I} + \left[\int_I \psi (\delta u) \, dI \right]_{\partial \Omega} \\ &= \int_I \int_\Omega \left(-\frac{\partial \psi}{\partial t} - \frac{\partial \psi}{\partial x} \right) (\delta u) \, d\Omega dI + \int_\Omega \psi (\delta u) \, d\Omega \Big|_{t=T} + \int_I \psi (\delta u) \, dI \Big|_{x=x_R} \end{aligned} \tag{2.12}$$

The resulting adjoint PDE is given by the following equation, where the BCs have been determined through the additional terms from integration by parts:

$$\begin{cases} -\dfrac{\partial \psi}{\partial t} - \dfrac{\partial \psi}{\partial x} = 1, & (x,t) \in \Omega \times I \\ \psi(x_R, t) = 0, & t \in I \\ \psi(x, T) = 0, & x \in \Omega \end{cases} \tag{2.13}$$

The negative sign of the partial time derivative in the adjoint PDE and the adjoint IC set at $t = T$ indicates that the adjoint solution marches backwards in time. Also, it is noteworthy that the Dirichlet

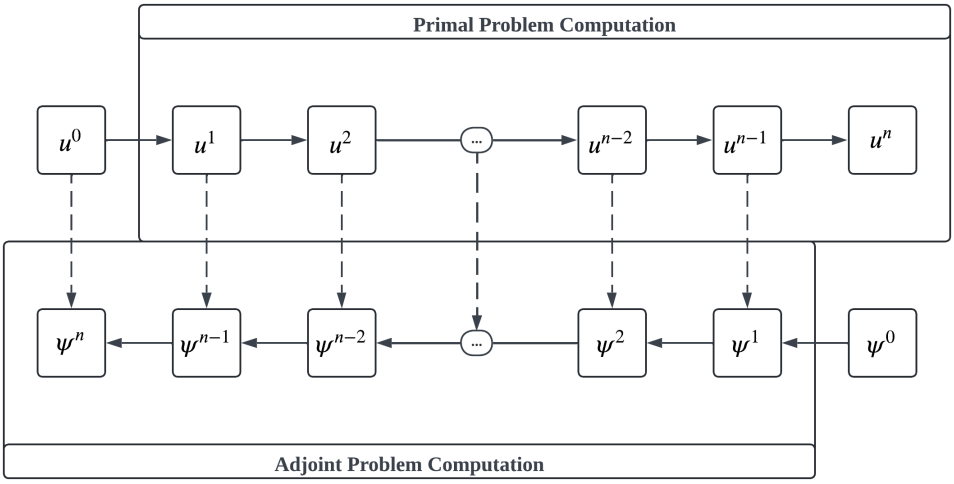


Figure 2.1: Primal and Adjoint Solution Procedure for Unsteady Problems (Dashed Arrows for Non-Linear Problems)

BC in the adjoint PDE is set at the opposing boundary of the primal PDE. Furthermore, the adjoint forcing term is dictated by the QoI selection as $g(x,t) = 1$.

The process for acquiring both the primal and adjoint solutions in unsteady problems is depicted in Figure 2.1, where the dashed arrows are utilized solely for non-linear problems. The latter results from the local linearization of the adjoint problem. The procedure commences with the numerical computation of the primal solution at each time step, marching in time using an appropriate temporal scheme. In linear problems, the adjoint solution is independent of the primal solution at each time step, allowing for backwards time marching with a suitable temporal scheme to obtain the adjoint solution. However, for non-linear problems, the adjoint local linearization necessitates the primal state at each adjoint time step, which, if stored on disk, leads to increased storage requirements and computational time [31].

2.4. Adjoint-Based Error Estimation and Localization

Following the numerical computation of the adjoint solution, the next step is the estimation of numerical errors. Adjoint-based error estimation can also be defined in a continuous or discrete context. Since the adjoint solution obtained through the continuous technique is ultimately discrete, the discrete approach to compute the error estimate is preferred [7, 18, 26]. A more detailed derivation of the discrete technique is given in [26].

The discrete context framework assumes the primal PDE is discretized in a spatial mesh H, providing an approximated discrete solution $\boldsymbol{u}_H$. Ideally, the true output error δJ is computed:

$$\delta J = J(u) - J_H(\boldsymbol{u}_H) \tag{2.14}$$

However, the exact solution u may be unknown. Thus, as a surrogate for the exact solution, a solution computed on a finer spatial mesh h is used. The fine-space mesh can be obtained through different refinement methods, where $h-$ and $p-$refinement are the most popular choices [11]. $h-$refinement involves adding points within the computational domain whereas $p-$refinement is associated with increasing the order of accuracy of the discretization method. As an approximation of the true output error, the error estimate δJ_{est} can be defined from a fine-space solution $\boldsymbol{u}_h$:

$$\delta J_{est} = J_h(\boldsymbol{u}_h) - J_H(\boldsymbol{u}_H) \tag{2.15}$$

Let $\boldsymbol{u}_h^H = \boldsymbol{I}_h^H \boldsymbol{u}_H$ be an injection of the coarse-space solution into the fine space h, given a lossless injection operator $\boldsymbol{I}_h^H$ from the coarse space to the fine space. Taylor expansions of the fine-space

QoI $J_h(\boldsymbol{u}_h)$ and the fine-space residual $\boldsymbol{R}_h(\boldsymbol{u}_h)$ about the injected coarse-space solution $\boldsymbol{u}_h^H$ can be formulated. An approximation for the error estimate can be defined by neglecting higher-order terms:

$$\delta J_{est} \approx -\boldsymbol{\psi}_h^T \boldsymbol{R}_h(\boldsymbol{u}_h^H) \tag{2.16}$$

Equation 2.16 is normally referred to as the adjoint-weighted residual expression where ψ_h is the fine-space adjoint solution and $\boldsymbol{R}_h(\boldsymbol{u}_h^H)$ is the fine-space residual evaluated with the injected primal solution. For non-linear problems, there is a linearization error $O(\delta \boldsymbol{u}^2)$ associated with the error estimate due to higher-order terms from the Taylor expansions:

$$\delta J_{est} \approx -\boldsymbol{\psi}_h^T \boldsymbol{R}_h(\boldsymbol{u}_h^H) + O(\delta \boldsymbol{u}^2) \tag{2.17}$$

As a result, if the adjoint solution was computed on an infinitely refined mesh, associated with the error estimate there would still exist an $O(\delta \boldsymbol{u}^2)$ error. This is due to the adjoint problem resulting from a linearization of the injected coarse-space primal state. Therefore, for non-linear problems, the error estimate can be more sensitive to underresolved solutions of the primal problem.

If an element-based method is employed in the spatial discretization, the error estimate can be localized to individual element contributions in the mesh. Thus, the error estimate can be rewritten:

$$\delta J_{est} \approx -\sum_{i=1}^{N_h} \psi_{h,i} R_{h,i}(\boldsymbol{u}_h^H) \tag{2.18}$$

where N_h is the number of elements in the fine space. For unsteady problems, this can be reformulated as a time-averaged error estimate if a time-averaged QoI is considered:

$$\delta \bar{J}_{est} \approx -\frac{1}{T} \sum_{j=1}^{N_t} \sum_{i=1}^{N_h} \psi_{h,i,j} R_{h,i,j}(\boldsymbol{u}_h^H) \tag{2.19}$$

Furthermore, a local error indicator can be defined as the absolute value of the local error estimate:

$$\varepsilon_{h,i} = \left| \psi_{h,i} R_{h,i}(\boldsymbol{u}_h^H) \right| \tag{2.20}$$

If the contribution of numerical errors stemming from the temporal discretization scheme is neglected, the following expression defines a time-averaged local error indicator:

$$\bar{\varepsilon}_{h,i} = \frac{1}{T} \sum_{j=1}^{N_t} \left| \psi_{h,i,j} R_{h,i,j}(\boldsymbol{u}_h^H) \right| \tag{2.21}$$

Based on the error indicator distribution, it is possible to formulate adaptation methods to modify the mesh. The objective is to minimize the error estimate with reduced computational cost. When uniform h-refinement is used, the fine-space error indicators can be aggregated across all fine-space elements contained within a particular coarse-space element to derive the final coarse-space indicator [26]. Conversely, if p-refinement is employed to construct the fine space, the error indicator directly quantifies the amount of error contributed by a specific coarse-space element to the QoI.

2.5. Limitations of Adjoint-Based Error Estimation

While adjoint-based error estimation offers remarkable accuracy by computing the sensitivity of the primal solution to a predefined QoI, its application is subject to three primary limitations. Firstly, adjoint-based error estimation necessitates the computation of the adjoint solution on a fine space, which can incur significant computational expense [7]. Secondly, managing storage requirements for unsteady non-linear problems poses an additional challenge. As the primal solution can be stored on disk [31]

or reconstructed by checkpoint techniques [46], an intricate balance between data storage, computational cost and solution accuracy is necessary. Lastly, the adjoint solution stability represents a critical consideration, particularly in unsteady turbulent flows [47].

Fine-Space Adjoint Solution Computational Cost

Various techniques have been employed in the literature to address the computational cost of the fine-space adjoint solution. One approach involves reconstructing higher-order adjoint solutions based on coarse-space adjoint solutions using an embedded finer mesh [21]. Additionally, temporal reconstruction of the adjoint solution has been applied to avoid solving the adjoint problem on a temporally enriched space [44]. However, in the cited cases it was observed that the adjoint solution remained more computationally expensive than the primal solution. As an alternative, hybridized discontinuous Galerkin methods have been utilized to compute both primal and adjoint solutions, resulting in faster solutions with iterative solvers and reduced storage requirements [24, 26]. The key advantage of hybridization is that the resulting set of algebraic equations has globally coupled degrees of freedom only on the skeleton of the computational mesh, leading to the solution of a much smaller system.

Moreover, Machine Learning (ML) has been applied to mitigate the computational cost associated with the fine-space adjoint computation. A super-resolution Artificial Neural Network (ANN) has been implemented to reconstruct fine-space adjoint solutions from coarse ones [7]. This approach has the advantage of generalizing to a new QoI for which it was untrained. Additionally, a Convolutional AutoEncoder (CAE) with feed-forward layers has been used to directly predict the error estimate and error indicator field from the primal solution on a coarse mesh [48]. ANNs have also been employed to establish mappings between the primal and adjoint solutions in aerodynamic design optimization [49].

Unsteady Non-Linear Problems Storage Requirements

Unsteady non-linear problems require the primal solution at each time step to obtain the adjoint solution, resulting in substantial storage requirements. This is true for large-scale turbulence problems like Large-Eddy Simulation (LES) [6]. To address storage demands, several techniques have been proposed in the literature. A simple solution is to store the primal solution on an external hard disk which can be then used when computing the adjoint solution [50]. However, the cost of data communication between the external disk and the solver is significant. Also, the slow development of memory capacity over the past decades [1] is an important limitation to the use of an external hard disk to store the primal solution.

Compression and reconstruction methods have also been applied to the primal solution to reduce storage requirements. One example is the checkpoint technique [46], which saves the primal solution at a relatively small number of states in time. When solving the adjoint solution which marches backwards in time, the primal solution is solved once again between the saved states. This procedure alleviates memory requirements but results in additional computational costs. A Reduced-Order Model (ROM) of the primal solution also shows great potential for reducing storage requirements. Modal decomposition techniques have been successfully applied in the compression and reconstruction of the primal solution, such as an enhanced *online* Proper Orthogonal Decomposition (POD) [6]. While *offline* variants require access to the entire flowfield solution, *online* POD techniques are computed on the fly, making it suitable for realistic large-scale problems. Also, ML has been applied to construct ROMs of the primal solution, such as a CAE with feed-forward layers applied to a lid-driven cavity flow [51]. Nonetheless, the implementation of more complex ML techniques in adjoint-based error estimation for unsteady problems is an area of research that is yet to be explored.

For statistically-steady turbulent problems, approximation methods have been proposed to derive a single-solve adjoint system while still performing LES [52]. One of these methods relied on the time-averaged residual and time-averaged QoI, whereas the other involved the conversion of the unsteady flow solution to a steady flow solution. It was shown that the first approximation yielded a better computational grid for better capture of flow features as well as a better prediction of the QoI. While the approximation methods were able to effectively reduce the computational cost and storage demands, the loss in accuracy of the error estimate is significant by considering steady-state quantities.

Unstable Unsteady Adjoint Solution

Turbulence is inherently chaotic, meaning that small variations in design parameters can lead to substantial changes in the flowfield. This poses a challenge in obtaining a stable adjoint solution for un-

steady problems. Coherent turbulent structures are convicted in the flowfield and get smaller until they are dissipated. As the adjoint problem marches backwards in time, the adjoint solution has to be constructed from lost information, which can result in divergence. To circumvent this issue, the least-squares shadowing method has been employed in chaotic flows to mitigate the divergence of the adjoint solution [47]. This method aims to obtain adjoint solutions that do not demonstrate unbounded exponential growth. While this technique has been demonstrated to be accurate, its implementation comes with a high computational cost, especially for large-scale simulations. To address this limitation, Shimizu and Fidkowski [53] combined the method with a ROM to approximate QoIs more accurately and economically compared to previous approaches. On the other hand, Fidkowski [54] introduced a field-inversion ML framework that only required unsteady primal solutions without the need for full storage. The ML model produced an adjoint solution for the averaged solution, which helped to address the instability issue.

In conclusion, addressing the storage requirements for unsteady non-linear problems in adjoint-based error estimation remains an active research area, with various methods applied in the literature. Surrogate modelling techniques, ranging from modal decomposition to deep learning, hold the potential to effectively compress the primal solution. Due to their inherent non-linear nature, deep learning methods can accurately represent turbulent flows [37–39]. In this study, the utilization of deep learning to create a surrogate model of the primal solution was investigated to alleviate storage constraints in unsteady adjoint-based error estimation.

3

Surrogate Modelling Techniques: Description and Applications

In this chapter, the use of surrogate modelling within the areas of Computational Fluid Dynamics (CFD) and adjoint-based error estimation is explored. Particular emphasis is given to modal decomposition techniques and Machine Learning (ML). The elucidation of these concepts aims to provide reasoning for the methodologies proposed.

First, a theoretical background of modal decomposition techniques is provided, highlighting its applications in the compression and reconstruction of CFD data. Furthermore, the fundamental principles of ML are presented, where the underlying concepts of Artificial Neural Networks (ANNs) and their different types are elucidated. A more thorough examination of these fundamental concepts is provided in [35, 55]. Subsequently, the applications of ML in CFD surrogate modelling are expounded upon and supplemented with examples sourced from the available literature. Lastly, specific applications of modal decomposition methods and ML in adjoint-based error estimation are provided together with research directions yet to be explored.

3.1. Modal Decomposition Techniques

Modal decomposition techniques have emerged as effective tools for data compression and reconstruction in CFD. Among these techniques, Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD) stand out as the most popular methods for Reduced-Order Models (ROMs). Both of these extract low-dimensional modes from flowfield snapshots [56]. While POD identifies an optimal set of orthonormal modes to represent data based on their energy content [57], DMD captures modes associated with growth rates and frequencies [58]. The POD modes have multiple temporal frequencies whereas DMD uses temporally-varying modes.

The DMD approach extracts dynamic information from the flowfield, where the resulting modes are not necessarily orthogonal. These dynamic modes can be used to represent the physical mechanisms present in the flowfield, as they are associated with frequency information and spatial structures. This method has been effectively applied to cavity flow and the wake behind a flexible membrane [58]. On the other hand, POD involves the eigendecomposition of the snapshot matrix, providing a set of orthonormal modes and their corresponding temporal coefficients to express the approximated flowfield. DMD and POD have been applied to investigate a jet in channel cross-flow [59]. While POD was able to reconstruct the flow with a low number of modes, DMD necessitated more modes to capture an equivalent amount of energy. This difference stems from the orthonormal property of POD modes, which renders them optimal in the L^2-norm [60]. Thus, POD can capture the largest possible amount of kinetic energy for any given number of modes [61].

Various variants of POD have been developed for the compression and reconstruction of CFD data. The original *offline* variant requires access to the entire set of flow solution snapshots. It has been

applied to lid-driven cavity flows obtained with Direct Numerical Solution (DNS) [61] as well as turbulent flowfields obtained with particle image velocimetry [62]. Common *offline* POD methodologies include eigenvalue decomposition, the method of snapshots and Singular Value Decomposition (SVD). Given a snapshot matrix $\boldsymbol{U} \in \mathbb{R}^{N_x \times N_t}$ with the temporal mean removed, the first method involves the eigenvalue decomposition of the correlation matrix $\boldsymbol{U}\boldsymbol{U}^T \in \mathbb{R}^{N_x \times N_x}$. This procedure, introduced by Lumley [57], is preferred when the number of snapshots is much larger than the number of degrees of freedom. On the contrary, if the number of degrees of freedom is much larger than the number of snapshots, the method of snapshots [63] can be employed to solve the eigenvalue decomposition of the correlation matrix $\boldsymbol{U}^T\boldsymbol{U} \in \mathbb{R}^{N_t \times N_t}$. Lastly, SVD [62] can be applied directly to the snapshot matrix to compute the POD modes as it generalizes the decomposition to non-square matrices. The SVD technique is known to be more robust in the presence of round-off errors [64]. On the other hand, *online* variants consider each snapshot solution individually and are constructed on the fly, making them suitable for large-scale problems. A randomized single-pass SVD was applied to various canonical turbulent flows [65]. Also, an incremental SVD was applied to a three-dimensional (3D) cylinder flow [6].

In this research, POD was selected as a baseline method over DMD due to its ability to capture the largest possible amount of kinetic energy, making it more efficient for data compression [61]. The use of an *online* variant, such as incremental SVD [6], was deemed unnecessary, as it involves many parameters which went beyond the scope of this study. Within the *offline* techniques, the SVD-based variant was chosen since it is more robust in the presence of round-off errors [64].

3.2. Deep Learning Techniques

Recent advancements in ML have been propelled by the evolution of novel learning algorithms and theoretical frameworks, coupled with the exponential proliferation of data availability and cost-effective computational resources [66]. While early efforts primarily focused on computer-human interactions such as image processing and speech parsing, there has been a growing interest in exploring the application of ML to scientific research. Among ML, deep learning models have attracted significant interest due to their capability to capture complex interactions and achieve remarkable performance across various domains, including information technology, healthcare and engineering [33].

Within the subset of deep learning methodologies, ANNs employ data-manipulation techniques to model non-linear functions. An example of an ANN architecture is presented in Figure 3.1. In contrast to other deep learning methods that rely on algorithms for pattern recognition, ANNs distinguish themselves through the utilization of a set of neurons, organized in layers and interconnected according to principles loosely inspired by physiological neural networks in biological brains [67]. The layers allow the network to process input data through successive transformations to obtain the output data. An ANN includes an input layer, one or more hidden layers and an output layer. The input layer receives the initial data, where each neuron represents a feature of the input data. The hidden layers process the input data through weighted connections to produce an output. This output serves as the input to the next layer. Like the hidden layers, the output layer processes the inputs received from the preceding layers to produce an output. Additionally, the architecture includes the arrangement, connection and dimension of the layers. Typically, the transformations are expressed as a composition of non-linear and affine functions, which govern the passage from one layer to the next:

$$\hat{\boldsymbol{y}} = f\left(\boldsymbol{W}^T\boldsymbol{y} + \boldsymbol{b}\right) \tag{3.1}$$

where $\boldsymbol{y} \in \mathbb{R}^{N_1}$ and $\hat{\boldsymbol{y}} \in \mathbb{R}^{N_2}$ are the values of the neurons in the preceding and subsequent layers respectively, f is an activation function, $\boldsymbol{W} \in \mathbb{R}^{N_1 \times N_2}$ are the weights and $\boldsymbol{b} \in \mathbb{R}^{N_2}$ is the bias term. The activation function is the one responsible for introducing non-linearities into the network. The network's architecture governs the adaptation of weights and biases to enable the network to approximate the training data. This adaptation process constitutes a regression task, which is concerned with predicting a numerical value, as opposed to classification, that refers to which category the value belongs to [35].

The training phase of an ANN is a crucial process wherein the network learns from the training data. Throughout this iterative procedure, the network iteratively refines its weights and biases by evaluating the predicted output against the desired output. Conceptually, training an ANN amounts to solving an

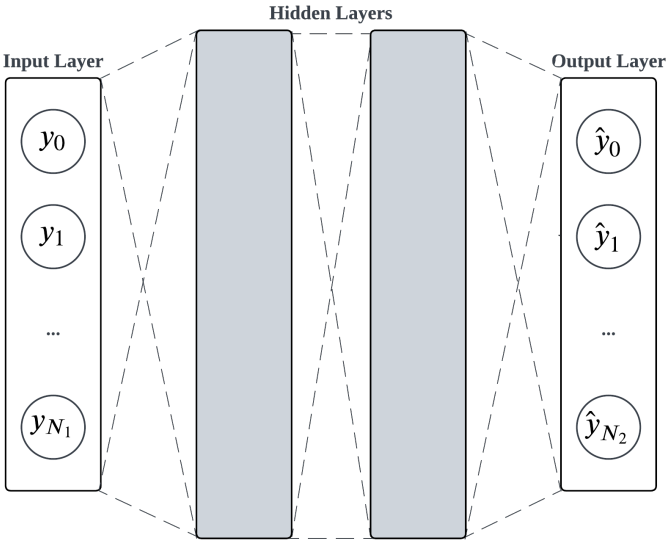


Figure 3.1: Artificial Neural Network Architecture

optimization problem, where the objective is to minimize a predefined loss function, also known as a cost function. Forward propagation is the process by which input data is passed through the ANN to generate an output prediction whereas backward propagation computes the gradient of the loss function with respect to the network's weights and biases to update these parameters in the direction that minimizes the loss function.

Regarding their architecture, ANNs can adopt three primary configurations: feed-forward, convolutional or recurrent. In feed-forward neural networks, data flows in a unidirectional manner from input nodes through hidden layers to output nodes with the use of weighted connections given in Equation 3.1. These are commonly used for tasks such as computer vision and speech recognition [68]. Convolutional Neural Networks (CNNs) are designed to effectively capture spatial patterns in data through the use of convolutional layers using filter kernels. These networks can efficiently extract specific features from multidimensional input data, making them successful in many practical applications [35]. Lastly, Recurrent Neural Networks (RNNs) employ an internal time-evolving state which allows them to exhibit dynamic temporal behaviour. By saving the output layer and feeding it back to the input layer, the time-advanced state can be predicted. This architecture is particularly well-suited for sequential data processing tasks [55].

3.2.1. Applications in Computational Fluid Dynamics

Surrogate models based on deep learning methods have gained widespread popularity in CFD for their robust capability to account for non-linearity in complex datasets [36]. Within deep learning methods, CNNs have emerged as powerful tools for super-resolution reconstruction of flowfields from low-resolution data [7, 69]. Liu et al. [70] conducted a comparative study between a static CNN and an innovative multi-temporal paths CNN in Turbulent Channel Flow (TCF). Both methods demonstrated high accuracy in reconstructing flowfields, but the latter, which incorporated spatial and temporal information simultaneously, provided more detailed information about under-resolved flow features. Furthermore, an *online* data compression technique using CNNs was developed for application to large-scale problems [65]. It significantly outperformed an *online* POD method with similar compression capabilities. Hierarchical Convolutional AutoEncoders (CAEs) have also been applied to the compression and reconstruction of a cylinder wake [38]. The hierarchical property organizes the non-linear modes by their contributions to the reconstructed field while achieving efficient order reduction, which enables more efficient data compression than conventional CAEs.

Moreover, modal decomposition techniques have been integrated with ML methods to achieve a non-linear mode decomposition. Murata et al. [36] fused POD with a CAE to visualize decomposed flow-

fields. The findings suggested considerable potential for this non-linear approach, necessitating fewer dimensions than the original POD. Additionally, a fusion of POD with a recurrent long short-term memory network has been explored for modelling transient fluid simulations [71]. Numerical experiments demonstrated that these POD coefficients were more effectively modelled compared to the inclusion of traditional POD modes. Furthermore, a combination of POD with a recurrent Echo State Network (ESN) has been proposed to mitigate the substantial number of states required by the ESN [72]. In gesture recognition, it has been demonstrated that ESNs achieve relatively good performance when compared to long short-term memory networks but have significantly lower training times [73].

Combinations of CNNs and RNNs have been developed to perform simultaneously spatial and temporal prediction. For example, a CAE combined with a long short-term memory network has been used [74, 75] for more accurate reproduction of the large and inertial scale statistics, making it attractive for many engineering applications. Furthermore, Racca et al. [39] successfully combined a CAE with an ESN in the prediction of turbulent dynamics, which accurately predicts the flow in terms of temporal dynamics and statistical properties at different Reynolds numbers.

Physics-informed deep learning techniques have recently been applied in data compression and reconstruction methods by incorporating physical constraints directly into the ANN's loss function. Mohan et al. [76] introduced a CAE while enforcing the concept of incompressible fluid flow in 3D fully developed turbulence. This approach significantly enhanced local mass conservation without compromising performance according to other metrics. Furthermore, the implementation of enforced incompressibility and preserved enstrophy has been explored in data compression [37], resulting in an improvement of the compression model. Additionally, Momenifar et al. [33] investigated the incorporation of physical constraints such as incompressibility and global statistical characteristics of velocity gradients into a CNN. They employed vector quantization to capture spatially correlated features of the flow that also existed across different scales.

3.3. Applications and Research in Adjoint-Based Error Estimation

Although modal decomposition techniques and deep learning methods have been widely utilized in the compression and reconstruction of CFD data, their application in adjoint-based error estimation has been limited. This section aims to elucidate this research gap through the presentation of literature examples relevant to the research thesis, followed by a potential avenue for future research and the methodologies proposed.

To address the computational overhead linked with the refined adjoint solution computation, super-resolution ANNs have been deployed to reconstruct fine adjoint solutions from coarse ones [7]. This research involved the use of a CNN, which significantly decreases the number of trainable parameters compared to feed-forward ANNs. Additionally, a CAE incorporating feed-forward layers has been employed to directly forecast the error estimate and error indicator field from the primal solution on a coarse mesh [48]. On the other hand, a limited number of deep learning applications for adjoint-based error estimation aim to mitigate the storage demands inherent in unsteady non-linear problems. For instance, an *online* POD with an incremental SVD has been successfully applied to create a ROM of the primal solution in the context of LES [6]. Also, a surrogate model of the primal solution was constructed with a CAE containing feed-forward layers and compared against POD, in the context of a lid-driven cavity flow [51]. However, POD managed to reconstruct the solution with a minimal number of modes owing to the smoothness of the primal solution, thereby not fully exploiting the capabilities of the CAE.

This research aimed to further exploit the capabilities of deep learning techniques to reduce storage requirements associated with unsteady adjoint-based error estimation. Building upon literature examples, three different methodologies were proposed to create a surrogate model of the primal solution: a CAE, an ESN and a CAE-ESN. An *offline* SVD-based POD is used as a baseline for comparison with the deep learning techniques. For the CAE, an architecture containing CNNs and feed-forward ANNs was proposed [48]. However, for higher-dimensional test cases, feed-forward layers were disregarded due to the substantial number of trainable parameters presented [7]. The ESN was proposed as it presents significantly lower training times than other RNNs [73]. Lastly, the CAE-ESN was proposed as a hybrid method as it offers the potential to enhance compression efficiency while minimizing additional computational costs [39].

4

Framework and Methodology

In this chapter, the computational framework and the theoretical foundation of the methodologies proposed are presented. These methodologies are introduced to create a surrogate model of the primal solution to be implemented in the framework of unsteady adjoint-based error estimation.

Initially, the implemented Finite Element Method (FEM) solver is described along with the adjoint-based error estimation framework used in this research. An *offline* variant of Proper Orthogonal Decomposition (POD) based on Singular Value Decomposition (SVD) is then outlined to serve as a baseline method. Additionally, the basic methodology behind Convolutional Neural Networks (CNNs) is explained, followed by the introduction of a Convolutional AutoEncoder (CAE). Furthermore, the theory of Echo State Networks (ESNs) is expounded, highlighting the optimization procedure of its hyperparameters. Moreover, a combination of a CAE and an ESN is also presented, referred to as a CAE-ESN. The Compression Ratio (CR) was defined for the different methods based on their structure, thus serving as a performance metric.

For the one-dimensional (1D) test cases, the CAE training was conducted on a single NVIDIA GeForce GTX 1070 GPU, while the ESN optimization and training were carried out on a 4-cored Intel Xeon E5-1620 CPU. For the two-dimensional (2D) test case, the CAE training used a single NVIDIA Tesla P40 GPU with the ESN optimization and training performed on an 18-cored Intel Xeon E5-2696 V3 CPU.

4.1. Computational Framework

This research investigated three test cases: a manufactured solution of the unsteady 1D viscous Burgers' equation; a Direct Numerical Solution (DNS) data-driven solution for the unsteady 1D viscous Burgers' equation to produce the wall-normal velocity in turbulent channel flow (TCF); and a DNS data-driven solution for the unsteady 2D viscous Burgers' equations to produce the spanwise and wall-normal velocity components in TCF.

The FEM was utilized as the numerical approach for spatial discretization of the primal and adjoint problems. Within the FEM framework, the solution is approximated as a linear combination of predefined local basis functions $\boldsymbol{\phi}(x)$:

$$u(x,t) \approx \sum_{i=1}^{N} \phi_i(x) a_i(t) \tag{4.1}$$

where u is the solution, N represents the number of basis functions and $\boldsymbol{a}(t)$ denotes the basis functions amplitudes, commonly referred to as degrees of freedom. These degrees of freedom serve as the discrete unknowns, which were determined using the method of weighted residuals. This approach required reformulating the governing equations into a weak or variational formulation. In this formulation, the weighted integrals of the residual of the strong form of the Partial Differential Equation (PDE) must vanish for all admissible weighting functions $q \in V$. The function space V is identical for the primal and

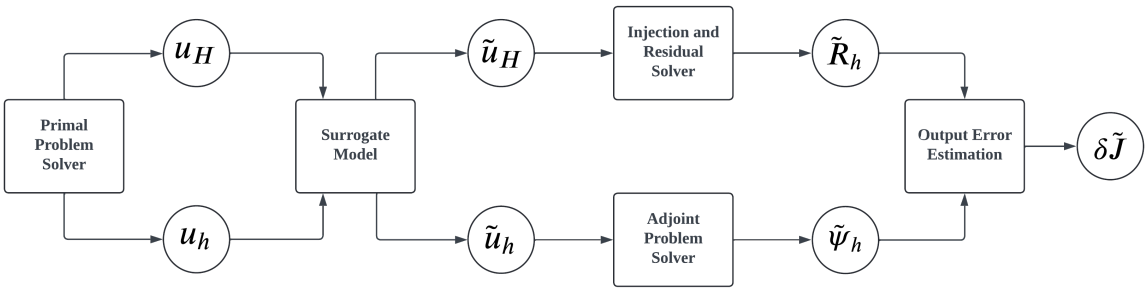


Figure 4.1: Computational Framework Flowchart

weighting functions. The variational problem was solved using the legacy `FEniCS`[1] package, employing the Bubnov-Galerkin method [77]. This method assumes the weighting functions are identical to the basis functions. Within the non-linear problem defined in `FEniCS`, the non-linear iterator `SNES` was employed, with `MUMPS` chosen as the solver and `BT` as the preconditioner [7].

Additionally, the second-order Crank-Nicholson method was selected for temporal discretization, represented by the following expression:

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^{n}}{\Delta t} + \frac{1}{2}\boldsymbol{\mathcal{R}}(\boldsymbol{u}^{n}) + \frac{1}{2}\boldsymbol{\mathcal{R}}(\boldsymbol{u}^{n+1}) = 0 \tag{4.2}$$

where $\boldsymbol{u}^n \in \mathbb{R}^N$ and $\boldsymbol{u}^{n+1} \in \mathbb{R}^N$ are the primal discrete solutions at times t_n and t_{n+1} respectively, Δt is the time step interval and $\boldsymbol{\mathcal{R}} \in \mathbb{R}^N$ is the primal discrete residual obtained by the spatial discretization of the variational formulation.

The computational framework flowchart for this research is presented in Figure 4.1, where the surrogate modelling techniques were integrated into the adjoint-based error estimation structure. Initially, the implemented primal solver was utilized to generate discrete primal solutions $\boldsymbol{u}$ in the coarse space H and fine space h. Uniform $h-$refinement with a factor of two was used to obtain the fine space from the coarse space. Surrogate primal solutions $\tilde{\boldsymbol{u}}$ in both spaces were then constructed using one of the methodologies proposed. The refined adjoint solution was computed with the fine-space surrogate primal solution $\tilde{\boldsymbol{u}}_h$. On the other hand, an injection of the coarse-space solution $\tilde{\boldsymbol{u}}_H$ into the fine space was conducted to evaluate the fine-space strong residual. With these quantities computed, a surrogate error estimate was computed along with local error indicator fields in the fine space. The error estimate quantifies the numerical errors of the coarse-space primal solution concerning a case-specific Quantity of Interest (QoI). Since $h-$refinement was used to obtain the fine space, the error indicators were summed across all fine-space elements contained within a specific coarse-space element to obtain a final coarse-space indicator.

Regarding the surrogate modelling stage, three methods were proposed in this research to create a surrogate primal solution: a CAE, an ESN and a CAE-ESN. An *offline* POD method was employed as a baseline for comparison against the deep learning techniques.

4.2. Benchmark: Proper Orthogonal Decomposition

An *offline* POD was used as a standard reference method to create a Reduced-Order Model (ROM) of the primal solution. An *online* variant, such as an incremental SVD [6], was deemed unnecessary due to the dependence on many parameters which go beyond the scope of this research. The advantage of compression with POD lies in the ordered arrangement of computed modes according to the contained amount of kinetic energy. The SVD technique was employed due to its higher robustness when compared to other *offline* POD techniques [64].

Assuming a statically-stationary flow, a snapshot matrix $\boldsymbol{U} \in \mathbb{R}^{N_x \times N_t}$ is assembled by the horizontal

[1] `https://fenicsproject.org`

concatenation of all solution vectors $\boldsymbol{u} \in \mathbb{R}^{N_x}$ with the temporal mean $\overline{\boldsymbol{u}} \in \mathbb{R}^{N_x}$ removed. SVD is directly applied to factorize the snapshot matrix $\boldsymbol{U} = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}$, where $\boldsymbol{L}$ and $\boldsymbol{R}$ are the left and right singular vector matrix respectively and $\boldsymbol{\Sigma}$ is the diagonal singular value matrix [56]. The columns of $\boldsymbol{L}$ contain the spatial POD modes and the temporal coefficients are determined by the product of $\boldsymbol{\Sigma}$ and $\boldsymbol{R}$. Reconstruction using this method is based on a specified percentage of kinetic energy to be contained within the predicted solution. This energy is determined through the cumulative energy contribution of the singular values of the matrix $\boldsymbol{U}$, which are the diagonal entries of the matrix $\boldsymbol{\Sigma}$. A specific threshold of 99% cumulative energy was selected as it is effective in accurately representing flowfields [6, 36, 59]. From this criterion, the optimal rank M was defined from which the obtained matrices from SVD were truncated. As a result, a ROM of the primal solution was computed from the following relationship:

$$\tilde{\boldsymbol{u}}(x,t) = \overline{\boldsymbol{u}}(x) + \boldsymbol{L}_M \boldsymbol{\Sigma}_M \boldsymbol{R}_M \tag{4.3}$$

where $\boldsymbol{L}_M$, $\boldsymbol{\Sigma}_M$ and $\boldsymbol{R}_M$ are the truncated SVD matrices based on the user-selected rank M. Additionally, the CR was defined based on the mean flowfield and SVD truncated matrices size:

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Size}(\overline{\boldsymbol{u}}) + \text{Size}(\boldsymbol{L}_M) + \text{Size}(\boldsymbol{\Sigma}_M) + \text{Size}(\boldsymbol{R}_M)} \tag{4.4}$$

The matrices $\boldsymbol{L}_M$ and $\boldsymbol{R}_M$ had the highest impact on the CR as they are dense matrices. Contrarily, the matrix $\boldsymbol{\Sigma}_M$ is a diagonal matrix.

4.3. Convolutional Autoencoder

In CNNs, a kernel filter with a certain size slides across the input layer with a specified stride [36]. The following equation describes the transition from one layer to the next:

$$\hat{\boldsymbol{y}} = f\left(\boldsymbol{y} \otimes \boldsymbol{W} + \boldsymbol{b}\right) \tag{4.5}$$

where $\boldsymbol{y} \in \mathbb{R}^{N_1}$ and $\hat{\boldsymbol{y}} \in \mathbb{R}^{N_2}$ are the values of the neurons in the preceding and subsequent layers respectively, f is an activation function, $\otimes$ is a convolution operator, $\boldsymbol{W}$ are the learnable weights of the kernel filter and $\boldsymbol{b} \in \mathbb{R}^{N_2}$ is the learnable bias term. The Rectified Linear Unit (ReLU) was chosen as the activation function and it is defined as:

$$f(y) = \max(0, y) \tag{4.6}$$

The ReLU was selected as it is non-saturated for positive values. Typically, additional artificial values are inserted around the input layer in a process known as padding. In this research, solely zero-padding was used, which involves the addition of elements with a value of zero in the boundaries. Depending on the defined values for the kernel filter size, stride and padding, the size of the output layer is given by the following relationship when using the convolution operator:

$$N_2 = \frac{N_1 + 2 \times \text{Padding} - \text{Kernel Size}}{\text{Stride}} + 1 \tag{4.7}$$

If the convolution operator was used, the CNN parameters were selected in order to reduce the size of the output. On the other hand, the size of the output was increased by using the transpose convolution operator [78]. In this case, the size of the output layer is given by the following expression:

$$N_2 = (N_1 - 1) \times \text{Stride} - 2 \times \text{Padding} + \text{Kernel Size} \tag{4.8}$$

Within unsupervised learning algorithms, autoencoders are a set of neural networks trained to compress their inputs and then reconstruct them as the outputs. Unsupervised learning involves training the algorithm on an unlabeled dataset, where data instances are provided without explicit output targets

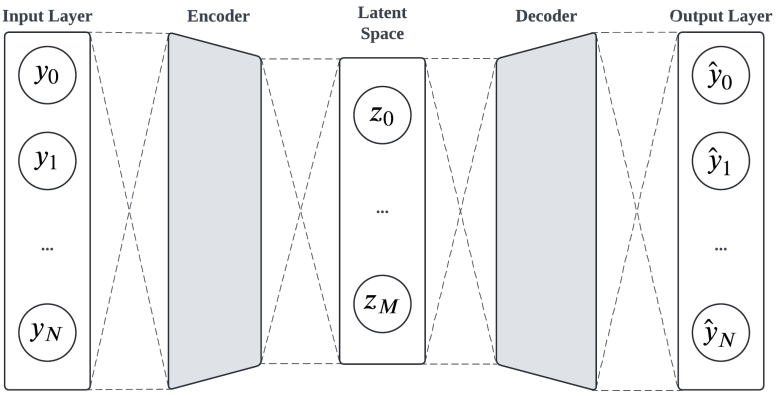


Figure 4.2: Autoencoder Architecture

[35]. An example of an autoencoder architecture is presented in Figure 4.2. The encoder stores the representational information needed to convert the data from its original high-dimensional state $\boldsymbol{y} \in \mathbb{R}^N$ to a low-dimensional latent space $\boldsymbol{z} \in \mathbb{R}^M$ with $M << N$. The decoder learns how to rebuild the data from the latent space to the original reconstructed state $\hat{\boldsymbol{y}} \in \mathbb{R}^N$. It is a robust data-driven method that computes a low-dimensional representation of the data [38]. Instead of using pooling and up-sampling layers in the CAE architecture [36], convolution layers were used in the encoder architecture and transpose convolution layers were employed in the decoder architecture. As a result, the CAE can capture spatial phenomena of different sizes due to the use of different kernel filter sizes [39].

For the 1D test cases, CAEs were composed of CNNs and feed-forward layers to and from the latent space. Besides controlling the latent space dimension [36, 51], the inclusion of feed-forward layers enabled the CAE to capture more features from the system's dynamics, which might be beyond the capability of CNNs alone. For the higher-dimensional test case, only CNNs were used due to the high number of trainable parameters presented by feed-forward layers [7]. The CAEs developed in this research were implemented using the `PyTorch`[2] `(2.5.0)` package with `CUDA (11.5)`.

The Adam algorithm, also known as adaptive moments, is opted as the optimization algorithm [7, 51] with an adaptive learning rate as it is more robust regarding the choice of hyperparameters [35]. The learning rate γ is one of the most important hyperparameters since it has the highest influence on the model's capacity and its optimization process [55]. Generally, a high value of γ allows the model to learn faster, at the cost of arriving at a sub-optimal final set of parameters. On the other hand, a lower learning rate may allow the model to learn a more optimal set of parameters but may take significantly longer to train. Therefore, the selection of the value γ can be cumbersome. To circumvent this issue, a decaying learning rate was chosen which decreases with a linear factor until reaching a fixed minimum:

$$\gamma(\text{Epoch}) = \gamma_0 \left(1 + \frac{\text{Epoch}}{\text{Total Number of Epochs}}(\gamma_r - 1)\right) \tag{4.9}$$

where γ_0 is the initial learning rate and γ_r is the learning rate factor. These factors were user-defined based on trial and error. In the optimization procedure, a batch refers to a subset of the training dataset used to train the model in one optimization iteration. A smaller batch size implies a slower optimization process but can lead to better generalization performance to unseen data. An epoch refers to one complete pass through the entire training dataset. The selection of the batch size and total number of epochs were also user-defined and dependent on the test case investigated. The Mean Square Error (MSE) was selected as the loss function during the optimization process:

$$\text{MSE}(\hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{4.10}$$

[2] `https://pytorch.org`

For the optimization, the training data was normalized to have a zero mean and a unit standard deviation. Furthermore, Gaussian noise $\mathcal{N}(0, \sigma)$ was added to generate more training data which was statistically representative of the considered system. Within the entire training dataset, 80% of the data was used for training, whereas 20% was used for validation. Across all test cases, the dataset selected for validation corresponded to a portion of the original primal solution.

Taking into account the CAE architecture, the CR was defined when using a CAE to produce a surrogate model of the primal solution as:

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Size}(\boldsymbol{z}) \times \text{Time Resolution} + \text{Decoder Parameters} + \text{Normalization Parameters}} \tag{4.11}$$

Solely the decoder's trainable parameters were taken into account, since the latent space was stored at every time step instead of the original high-dimensional space. The mean and standard deviation of the primal solution also had to be stored due to the normalization process.

4.4. Echo State Network

One of the main issues of Recurrent Neural Networks (RNNs) is the inherent difficulty in backward propagation through time using gradient descent algorithms, which have a relatively high computational cost [79]. Contrarily, in ESNs, the inputs are expanded to a higher-dimensional hidden layer, referred to as the reservoir. The reservoir acts as the system's memory and as a non-linear expansion of the inputs [39]. The connections between the input layer and the reservoir are pseudo-randomly generated, while the training is restricted to a linear regression problem of the connections between the reservoir and the output layer. Consequently, ESNs eliminate the need for backward propagation. This training procedure results in significantly reduced computational time compared to other RNNs. Nevertheless, the choice of hyperparameters has a significant impact on ESN performance [40]. In terms of architecture, the evolution of an ESN is governed by the following discrete-time dynamics equations:

$$\begin{cases} \boldsymbol{r}^{n+1} &= (1-\alpha)\boldsymbol{r}^n + \alpha f\left(\boldsymbol{W}_r^T \boldsymbol{r}^n + \boldsymbol{W}_{in}^T \left[\boldsymbol{y}^n, b_1\right]\right) \\ \hat{\boldsymbol{y}}^{n+1} &= \boldsymbol{W}_{out}^T \left[\boldsymbol{r}^{n+1}, b_2\right] \end{cases} \tag{4.12}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ and $\hat{\boldsymbol{y}} \in \mathbb{R}^N$ are the values of the neurons in the input and output layers respectively, $\boldsymbol{r} \in \mathbb{R}^{N_r}$ is the reservoir state with $N_r >> N$, α is the leaking rate, $\boldsymbol{W}_{in} \in \mathbb{R}^{(N+1)\times N_r}$ is the input weight matrix, $\boldsymbol{W}_r \in \mathbb{R}^{N_r \times N_r}$ is the reservoir state matrix and $\boldsymbol{W}_{out} \in \mathbb{R}^{(N_r+1)\times N}$ is the output matrix. Normally, a larger reservoir implies a better performance, as long as measures are taken against overfitting [79]. Within the ESN architecture, the reservoir should possess the Echo-State Property (ESP), signifying that the reservoir state should be uniquely defined by the fading history of the input state [80]. The ESP implies that the influence of past input states on the reservoir states vanishes with time. This property is guaranteed if the activation function is the hyperbolic tangent defined as:

$$f(y) = \tanh(y) \tag{4.13}$$

Also, the eigenvalues of the matrix $\boldsymbol{W}_r$ have to be smaller than unity for the ESP to be satisfied [72]. Nevertheless, this latter condition limits the richness of the reservoir, thus discouraging its use.

The leaking rate α can be regarded as the speed of the input and targeted output dynamics [79]. Since the ESN was used to construct a surrogate model of an already time-discretized primal solution, the need to introduce a higher temporal correlation was disregarded. Therefore, the ESN was simplified as a fully-leaked network with $\alpha = 1$. Taking these considerations into account, the discrete-time dynamics equations of the ESN were simplified:

$$\begin{cases} \boldsymbol{r}^{n+1} &= \tanh\left(\boldsymbol{W}_r^T \boldsymbol{r}^n + \boldsymbol{W}_{in}^T \left[\boldsymbol{y}^n, b_1\right]\right) \\ \hat{\boldsymbol{y}}^{n+1} &= \boldsymbol{W}_{out}^T \left[\boldsymbol{r}^{n+1}, b_2\right] \end{cases} \tag{4.14}$$

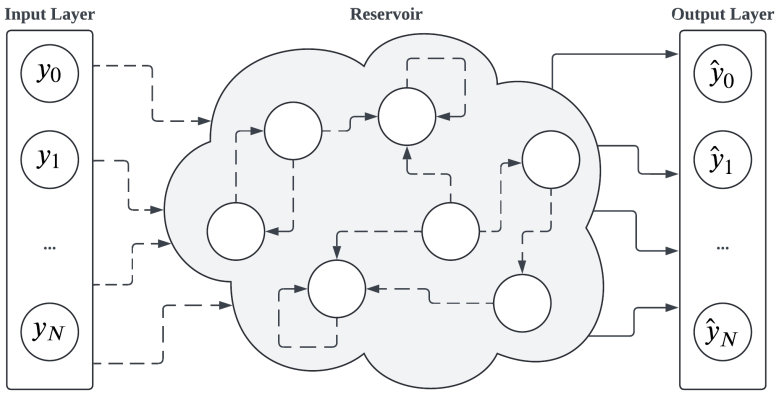


Figure 4.3: Echo State Network Architecture

The input state is mapped with a bias term b_1 into the reservoir state by $\boldsymbol{W}_{in}$, followed by an update of the reservoir state at the next time step with $\boldsymbol{W}_r$. The output state is computed with the updated reservoir state coupled with a bias term b_2 and $\boldsymbol{W}_{out}$. The input bias was selected to be the average absolute value of the normalized inputs whereas the output bias was determined through training. These terms are added to break the ESN inherent symmetry [81].

The ESN can be run in two configurations: closed-loop, where the output layer is fed back to the input layer to advance in time, or open-loop, where there is no output feedback. The simplified ESN open-loop architecture is depicted in Figure 4.3, where the dashed and solid arrows represent the pseudo-randomized and trainable parameters respectively. Initially, the pseudo-randomized matrices $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_r$ are generated. The matrix $\boldsymbol{W}_{in}$ has one element different from zero per row sampled from a uniform distribution $\mathcal{U}\left[-\sigma_{in}, \sigma_{in}\right]$. This implies that each neuron in the input layer is pseudo-randomly connected to one neuron in the reservoir. A uniform distribution is preferred due to its continuity and bounded values [79]. The hyperparameter σ_{in} is defined as the input scaling and determines the reservoir responses' non-linearity. On the other hand, $\boldsymbol{W}_r$ is an Erdős-Renyi matrix with average connectivity of three [39], which is obtained by sampling from $\mathcal{U}\left[-1, 1\right]$. This indicates that each neuron is connected on average to three other neurons within the reservoir. The spectral radius ρ is the maximum absolute eigenvalue of $\boldsymbol{W}_r$ and is one of the most crucial hyperparameters [79]. It determines how fast the influence of an input vanishes in the reservoir as time progresses. In most situations, a spectral radius smaller than unity guarantees the ESP. However, the optimal value can be greater than unity for non-zero input states [79]. Ultimately, it should be selected to maximize the ESN performance.

Regarding its learning paradigm, an ESN presents a supervised learning algorithm, where each instance is paired with a corresponding target. While the training of traditional RNNs is performed with gradient descent, the training of ESNs is restricted to the matrix $\boldsymbol{W}_{out}$ [72]. Training the network on $N_t + 1$ snapshots consists of solving a linear regression problem, known as the Ridge regression:

$$\left(\boldsymbol{R}\boldsymbol{R}^T + \beta \boldsymbol{I}\right) \boldsymbol{W}_{out} = \boldsymbol{R}\boldsymbol{Y}_{target}^T \tag{4.15}$$

where $\boldsymbol{R} \in \mathbb{R}^{(N_r+1)\times N_t}$ is the vertical concatenation of reservoir states, β is the Tikhonov regularization parameter which regulates over-fitting [79], $\boldsymbol{I} \in \mathbb{R}^{(N_r+1)\times(N_r+1)}$ is the identity matrix and $\boldsymbol{Y}_{target} \in \mathbb{R}^{N\times N_t}$ is the horizontal concatenation of targeted output states. The training is performed in open-loop configuration, where first a washout state is conducted. The washout is an initial transient period, where the output layer is not computed. The washout allows for the reservoir to be updated with the current state of the system [39]. The training data was normalized by its range. In chaotic scenarios, Gaussian noise $\mathcal{N}(0, \sigma)$ with a standard deviation $\sigma = 10^{-3}$ was added to the training data to improve the forecasting of chaotic dynamics [39, 79].

Following the training process, validation is carried out in closed-loop configuration. After an washout

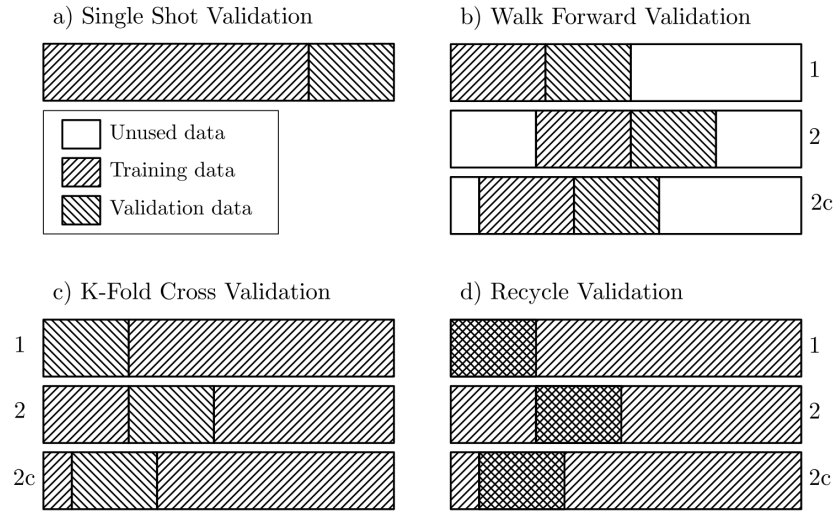


Figure 4.4: Echo State Network Validation Strategies [40]

period, σ_{in}, ρ and β are selected in validation. Since training is open-loop and validation is closed-loop, different strategies can be employed. The validation strategies are represented in Figure 4.4. In all strategies except the single shot validation strategy, bar 1 shows the first validation fold over the training set. Bars 2 and $2c$ represent the second validation fold in non-chaotic and chaotic cases respectively. In chaotic cases, the fold is shifted by 1 Lyapunov Time (LT). The LT is a fundamental timescale in chaotic dynamical systems, defined as the inverse of the Highest Lyapunov Exponent (HLE). This exponent quantifies the exponential rate at which infinitesimally close trajectories diverge [40]. Nikitin [82] determined the values of the HLE for TCFs at different Reynolds numbers, from which the required LTs were computed for the chaotic test cases investigated.

The recycle validation technique was employed as it has performed better than other validation strategies, particularly in predicting chaotic flows [39]. The network was first trained only once per set of hyperparameters using the entire dataset in open-loop configuration. Subsequently, the network was validated on splits of the dataset in closed-loop configuration, which was already used for training. For the non-chaotic case, the MSE was used as the loss function, defined in Equation 4.10, and the averaged MSE over the different validation splits determined the optimal hyperparameters. For the chaotic case, the Prediction Horizon (PH) served as the loss function. The PH is defined as the time interval during which the Normalized Root Mean Square Error (NRMSE) of the ESN prediction remains below a user-defined threshold k:

$$\mathsf{PH} = \arg\max_{t} \left(t | \mathsf{NRMSE}(\hat{\boldsymbol{y}}) < k\right) \tag{4.16}$$

For each time step, the NRMSE is defined as follows:

$$\mathsf{NRMSE}(\hat{\boldsymbol{y}}) = \sqrt{\frac{\mathsf{MSE}(\hat{\boldsymbol{y}})}{\frac{1}{N}\sum_{i=1}^{N}\sigma^2(y_i)}} \tag{4.17}$$

where σ^2 is the mean variance of the observed data. The normalization of the error enhances robustness, particularly in chaotic scenarios. The averaged PH, computed over different starting points selected from the training dataset, was used to determine the optimal hyperparameters.

A Bayesian Optimization (BO) procedure was performed to identify the optimal hyperparameters σ_{in} and ρ due to the high computational cost associated with the objective function [40]. In this optimization procedure, the objective function is modelled as a black box using a Gaussian Process (GP) regression,

thus not requiring gradient information. A Mátern kernel was selected as the GP model [39]. Following the GP evaluation in an initial grid search of 5×5 points, the objective function was evaluated at 5 more additional points which maximized the expected improvement acquisition function [83]. Finally, the hyperparameter β was selected by evaluating the objective function in a grid search space.

Following the hyperparameters selection, the solution prediction was performed in closed-loop configuration. As the ESN approximation of the primal solution would be used to solve the adjoint problem, the ESN temporal prediction was performed backwards in time. A checkpoint technique was introduced in the primal solution prediction to prevent the accumulation of errors as time progresses. In the non-chaotic scenario, the number of checkpoints was equal to the number of validation splits used during the optimization procedure. Conversely, in the chaotic case, the checkpoint length was set to match the optimized mean PH.

Taking into account the discrete-time dynamics equations of the simplified ESN given in Equation 4.14, the CR was defined when using the network to create a surrogate model of the primal solution:

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Checkpoint Parameters} + \text{Size}(\boldsymbol{W}_{out}) + \text{Hyperparameters}} \tag{4.18}$$

With the introduction of the checkpoint technique, it became necessary to store both the input and reservoir states at each checkpoint interval. Additionally, since the surrogate primal solution was used to compute the adjoint-based error estimate, the residual was also stored at every checkpoint interval. The matrix $\boldsymbol{W}_{out}$ had the most significant impact on the CR, as it is a dense matrix. In contrast, the matrices $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{r}$ can be fully determined by the hyperparameters σ_{in} and ρ, provided a specific random seed is selected.

4.5. Convolutional Autoencoder - Echo State Network

The use of RNNs becomes prohibitive when accurately predicting high-dimensional systems [84]. Even though ESNs require significantly less computational time than other types of RNNs, their training remains excessively time-consuming for higher-dimensional systems. As an alternative, a CAE is employed to compress the original high-dimensional space into a low-dimensional latent space. The ESN is then directly applied to this latent space to advance it in time. Subsequently, the decoder is applied to each latent space sample to reconstruct the original high-dimensional space. This combination, designated here as CAE-ESN, circumvents the challenges associated with high-dimensional data with reduced computational cost [39].

An example of a CAE-ESN architecture is presented in Figure 4.5. The CAE developed was repurposed in combination with a new ESN applied to the latent space. As in the CAE method itself, the high-dimensional data was normalized to have zero mean and unit standard deviation. However, a new ESN was required since it was not applied directly to the original high-dimensional space. The same BO procedure highlighted for the ESN alone was applied for the ESN applied to the latent space. A checkpoint technique was also introduced for the ESN when applied to the latent space to improve forecasting accuracy. Furthermore, the time prediction was performed backwards as the surrogate primal solution produced by the CAE-ESN was used to solve the adjoint problem.

Considering the various components of the CAE-ESN system, the CR was defined when this method was used to produce a surrogate model of the primal solution:

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Normalization and Checkpoint Parameters} + \text{Decoder Parameters} + \text{ESN Parameters}} \tag{4.19}$$

As with the ESN, the residual in the high-dimensional space was stored at every checkpoint interval due to its use in adjoint-based error estimation. The number of ESN parameters depended on the new ESN trained for the latent space. In contrast, the number of decoder parameters remained consistent with those produced for the method using only the CAE.

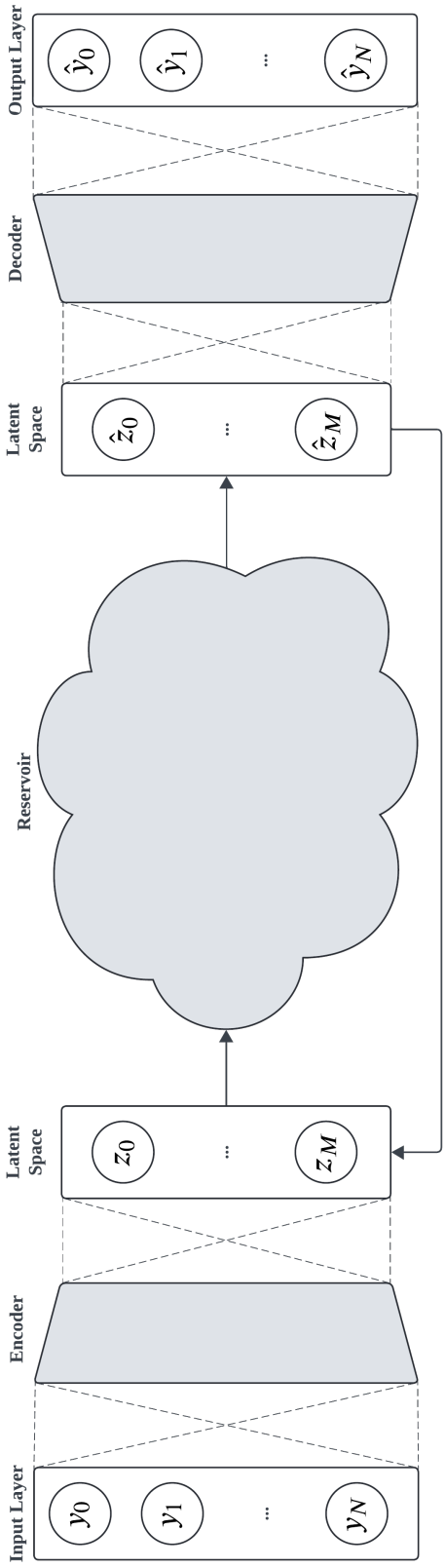


Figure 4.5: Convolutional Autoencoder - Echo State Network Architecture

5

Framework Verification: Manufactured Solution

In this chapter, a manufactured solution for the unsteady one-dimensional (1D) viscous Burgers' equation was used as a verification tool for the solver, methodologies and adjoint-based error estimation framework.

Firstly, the primal governing equations are introduced along with the solution obtained using the Finite Element Method (FEM), implemented with the `FEniCS`[1] package. Furthermore, surrogate models of the primal solution were computed with the methodologies proposed, where their performance was assessed for different metrics. Subsequently, the adjoint Partial Differential Equation (PDE) was derived with the continuous technique outlined in section 2.1, followed by the presentation of the numerical adjoint solution. The adjoint-based error estimates were then calculated based on the numerical solutions obtained with the implemented solver and with the methodologies proposed. Finally, local error indicator fields were generated which could be used to develop mesh adaptation methods.

5.1. Primal Problem Formulation and Solution

The unsteady 1D viscous Burgers' equation is a non-linear second-order PDE. This equation is frequently employed as a simplified model for turbulence, boundary layer behaviour, shock wave formation and mass transport [85]. The Method of Manufactured Solution (MMS) was employed as the verification technique, which is capable of generating exact reference solutions for PDEs given its domain [86]. Within this test case, the primal problem was considered in a space-time domain $\Omega : [0, L] \times I : [0, T]$ with $L = 1.0$ and $T = 20$. The following manufactured solution was considered [7, 45]:

$$u_{\text{MMS}}(x, t) = \sin^2(\pi t) \sin(\pi x) \tag{5.1}$$

Thus, homogeneous Dirichlet Boundary Conditions (BCs) and a homogeneous Initial Condition (IC) were imposed, resulting in the following primal PDE:

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} - \dfrac{1}{Re}\dfrac{\partial^2 u}{\partial x^2} = f_{\text{MMS}}, & (x, t) \in \Omega \times I \\ u(x, t) = 0, & (x, t) \in \partial\Omega \times I \\ u(x, 0) = 0, & x \in \Omega \end{cases} \tag{5.2}$$

where x and t are the space and time coordinates, u is the primal solution, $Re = 100$ is the Reynolds number and f_{MMS} is the manufactured forcing term. The term f_{MMS} was obtained from evaluating the PDE with the considered manufactured solution.

[1] `https://fenicsproject.org`

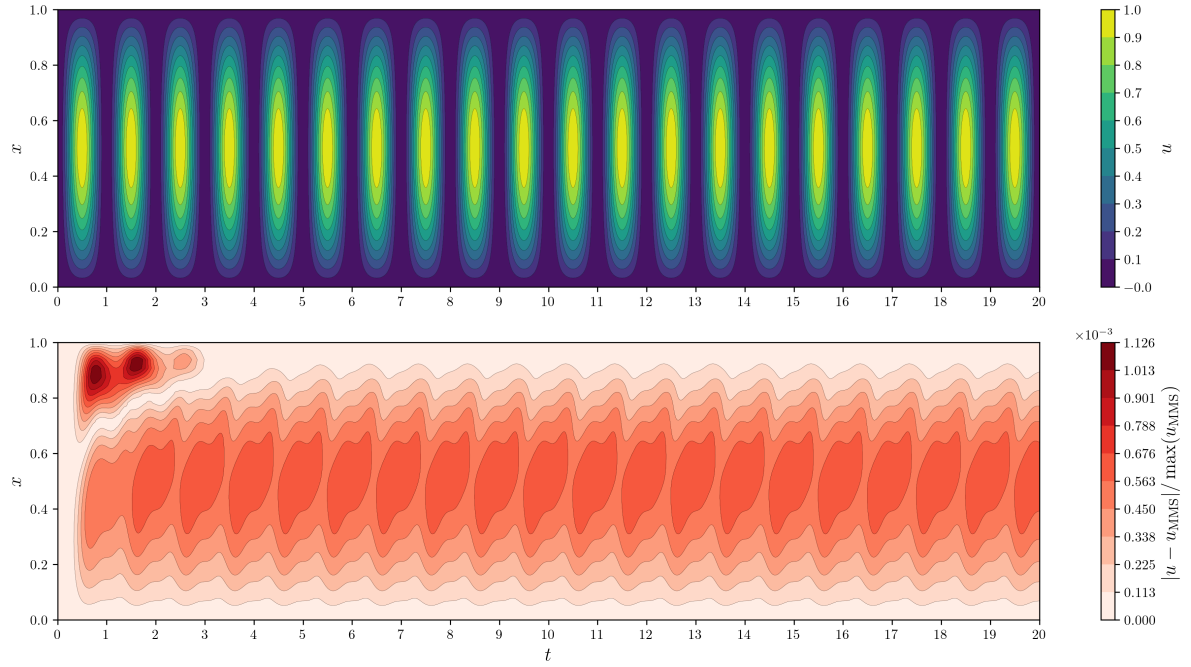


Figure 5.1: Primal Solution and Relative Error of Manufactured Solution for $N_x = 32$ Spatial Elements

As the FEM was employed as the numerical method for spatial discretization, it was imperative to derive the primal weak formulation. This formulation is established by calculating the inner product of the strong primal residual with weighting functions $q \in V_u$, where the function space $V_u(\Omega)$ remains the same for the primal and weighting functions. Integration by parts was applied to the resulting equation. The Dirichlet BCs were then applied strongly, leading to the following weak formulation:

$$\int_{\Omega} \left(\frac{\partial u}{\partial t} q + u \frac{\partial u}{\partial x} q + \frac{1}{Re} \frac{\partial u}{\partial x} \frac{\partial q}{\partial x} - f_{\mathsf{MMS}} q \right) d\Omega = 0 \tag{5.3}$$

Continuous piece-wise linear elements, also known as continuous Galerkin linear elements, were used as the finite elements for spatial discretization. In these elements, the first derivative is discontinuous while the second derivative is zero. Integration by parts eliminated the presence of second-order derivatives in the weak formulation, thus the use of continuous piece-wise linear finite elements was suitable. Regarding the Crank-Nicholson method, a time step interval $\Delta t = 1 \times 10^{-3}$ was chosen which was sufficient to ensure stability without the need for additional stabilization techniques. This time step interval minimized errors stemming from the temporal discretization scheme [45]. Thus, it was assumed that the output error was only due to the spatial discretization.

The spatial resolutions $N_x = \{8, 16, 32, 64, 128\}$ were investigated with particular emphasis on $N_x = 32$ spatial elements. For this spatial resolution, the primal discrete solution $\boldsymbol{u}$ computed and its relative error to the manufactured solution is shown in Figure 5.1. The relative error was defined as the absolute error normalized by the maximum value of the true solution. The primal solution presented a smooth and periodic behaviour centred around the spatial midpoint, effectively reflecting the imposed BCs and IC. Concerning its error, a non-periodic region was observed during the initial time steps, with a higher error observed near the spatial endpoint. This characteristic was expected from the imposed IC, which was not equal to the long-time periodic solution. For later time steps, the error demonstrated a periodic behaviour centred around the spatial midpoint of approximately $O(10^{-3})$. As a result, the domain $I : [T/2, T]$, where the primal solution error was statistically-steady, was considered as the temporal domain for the remaining analysis. For simplicity, this domain is now referred to as $I : [t_0, t_0 + T]$ with $t_0 = 10$ and $T = 10$.

A time-averaged Quantity of Interest (QoI) $\bar{J}$ was chosen for analysis and was defined as follows:

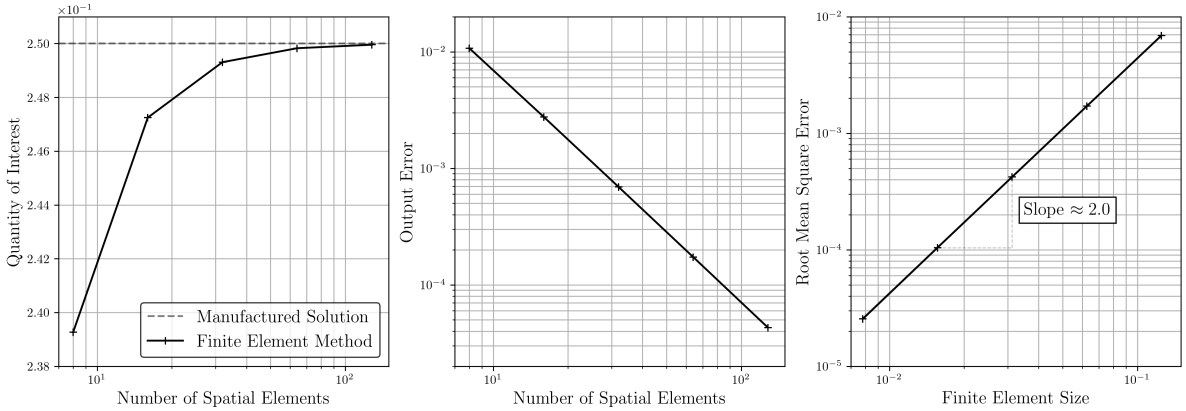


Figure 5.2: Quantity of Interest, Output Error and Root Mean Square Error of Manufactured Solution for Different Spatial Resolutions

$$\bar{J} = \frac{1}{T}\int_I \int_\Omega \sin(\pi x) u(x,t)\, d\Omega dI \tag{5.4}$$

The QoI selected had the advantage of weighting the solution near the centre of the spatial domain. This approach effectively diminishes the influence of errors stemming from elements near the boundaries in the computation of the adjoint solution and in the error estimation process [7]. The manufactured value of the QoI $\bar{J}_{\text{MMS}} = 2.5 \times 10^{-1}$ was computed while using the manufactured solution selected.

A grid convergence study was performed to verify the implemented solver. This study was based on the computed QoI and the Root Mean Square Error (RMSE) of the computed primal solution $\boldsymbol{u}$. The following expression was used to calculate the RMSE:

$$\text{RMSE}(\boldsymbol{u}) = \sqrt{\frac{1}{N_t N_x}\sum_{j=1}^{N_t}\sum_{i=1}^{N_x}\left[u(x_i,t_j) - u_{\text{MMS}}(x_i,t_j)\right]^2} \tag{5.5}$$

where N_t is the number of time steps. The computed time-averaged QoIs as well as the output error $\delta\bar{J} = \bar{J}_{\text{MMS}} - \bar{J}$ for different spatial resolutions and the RMSE for different element sizes are shown in Figure 5.2. The manufactured value of the QoI is shown for comparison. The QoI computed presented a convergent behaviour towards the manufactured value as the spatial grid was refined. As a result, the output error monotonically decreased with an increasing number of spatial elements. Thus, it was valid to assume that the output error was only due to spatial discretization. Furthermore, the RMSE exhibited a monotonically decreasing behaviour for all spatial resolutions. The slope exhibited by the RMSE aligned with the expected convergence rate of $O(\Delta x^2)$ for linear elements in the L_2-norm. Based on these observations, the implemented solver was verified.

5.2. Primal Solution Surrogate Models

This section explores the methodologies proposed to produce a surrogate model of the primal solution obtained with the implemented solver. A surrogate primal solution was developed to minimize the storage demands associated with unsteady adjoint-based error estimation.

First, results from the *offline* Proper Orthogonal Decomposition (POD) method are shown, where an analysis was conducted of the user-defined rank influence on the retained kinetic energy. Subsequently, results from the Convolutional AutoEncoder (CAE) are shown. Its training time and accuracy were compared across the spatial resolutions considered. Furthermore, a study was conducted on the influence of the Echo State Network (ESN) reservoir size to construct a surrogate primal solution. After determining the optimal reservoir size, the optimization procedure for tuning the ESN was evaluated in terms of computational time and performance across the spatial resolutions considered. For the CAE-ESN approach, a study was also conducted on the impact of the reservoir size when the ESN was applied to the

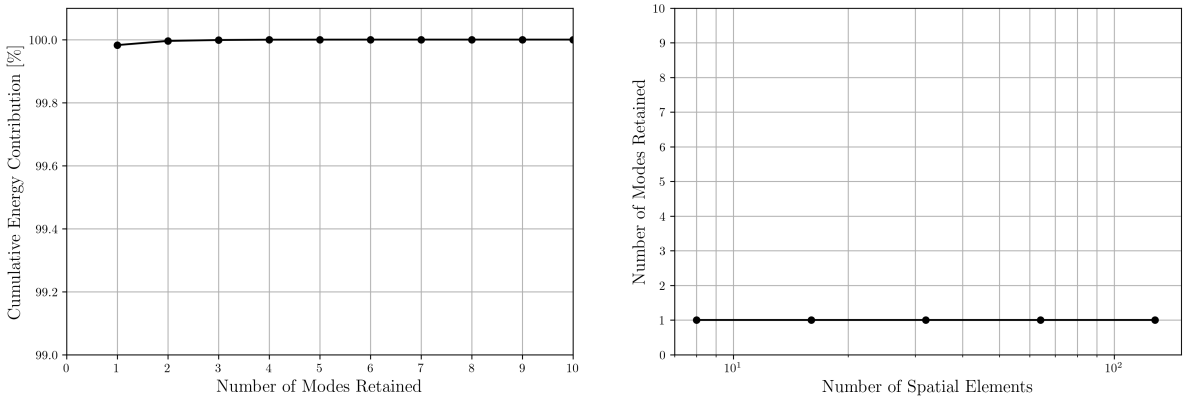


Figure 5.3: Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_x = 32$ Spatial Elements and Number of Modes Retained of Manufactured Solution for Different Spatial Resolutions

latent space. The total computational time, encompassing the CAE's training and the new ESN's training, and the performance of the combined model were analyzed for the spatial resolutions considered. Lastly, the performance of all methodologies was quantitatively assessed and compared. Performance criteria included the Compression Ratio (CR), computational time and the Mean Square Error (MSE) relative to the FEM primal solution, defined as follows:

$$\text{MSE}(\tilde{\boldsymbol{u}}) = \frac{1}{N_t N_x} \sum_{j=1}^{N_t} \sum_{i=1}^{N_x} \left[\tilde{u}(x_i, t_j) - u(x_i, t_j)\right]^2 \tag{5.6}$$

where $\tilde{\boldsymbol{u}}$ is the surrogate primal solution obtained with each of the methods proposed. Achieving an accurate surrogate solution was crucial, but it must be balanced with an effective CR and moderate computational time.

5.2.1. Proper Orthogonal Decomposition

The benchmark *offline* POD was first used to construct a Reduced-Order Model (ROM) of the primal solution as a baseline method. The temporal mean $\bar{\boldsymbol{u}} \in \mathbb{R}^{N_x}$ was removed from the solution to obtain the fluctuation vectors $\boldsymbol{u}'(t) \in \mathbb{R}^{N_x}$. The snapshot matrix $\boldsymbol{U} \in \mathbb{R}^{N_x \times N_t}$ was assembled by the horizontal concatenation of the fluctuation vectors. Singular Value Decomposition (SVD) was then applied to factorize the matrix $\boldsymbol{U}$. As a reconstruction criterion, the number of POD modes retained by the reconstructed primal solution should conserve 99% of the kinetic energy [6, 36, 59].

For a spatial resolution of $N_x = 32$ elements, the cumulative energy contribution of the first ten POD modes and the optimal rank for each spatial resolution are represented in Figure 5.3. One single POD mode retained the established amount of overall kinetic energy for $N_x = 32$ spatial elements and the remaining spatial resolutions considered. This characteristic was due to the smoothness and single spatial wavenumber of the primal solution.

5.2.2. Convolutional Autoencoder

The first method proposed to construct a surrogate model of the primal solution was the CAE. Regardless of the spatial resolution, the latent space dimension remained constant with a size of eight neurons. The latent space dimension was selected for this method to present compression capabilities similar to the other deep learning methods.

A different CAE architecture was designed for each spatial resolution considered due to the different sizes of the input layer. For every resolution, the encoder comprised Convolutional Neural Network (CNN) layers until the spatial signal was compressed to a single neuron with a defined number of channels. Feed-forward layers were then used to compress the flattened signal to the defined latent space size. The decoder architecture mirrored the encoder one but with transpose convolution operators [78]. The CAE architecture used for a spatial resolution of $N_x = 32$ elements is given in Table 5.1.

Table 5.1: Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 32$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 11,472			Number of Trainable Parameters: 11,465		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	(Size, Stride, Padding) $= (4, 1, 1)$ ReLU Activation Function	$(8, 31)$	**Linear**	(In, Out) $= (8, 16)$ ReLU Activation Function	(16)
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(16, 16)$	**Linear**	(In, Out) $= (16, 32)$ ReLU Activation Function	(32)
Convolution	(Size, Stride, Padding) $= (2, 2, 1)$ ReLU Activation Function	$(32, 9)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 0, 0)$ ReLU Activation Function	$(32, 3)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 5)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 5)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 3)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 9)$
Convolution	(Size, Stride, Padding) $= (3, 0, 0)$ ReLU Activation Function	$(32, 1)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(16, 16)$
Linear	(In, Out) $= (32, 16)$ ReLU Activation Function	(16)	**Transpose Convolution**	(Size, Stride, Padding) $= (2, 2, 1)$ ReLU Activation Function	$(8, 31)$
Linear	(In, Out) $= (16, 8)$	(8)	**Transpose Convolution**	(Size, Stride, Padding) $= (4, 1, 1)$	$(1, 32)$

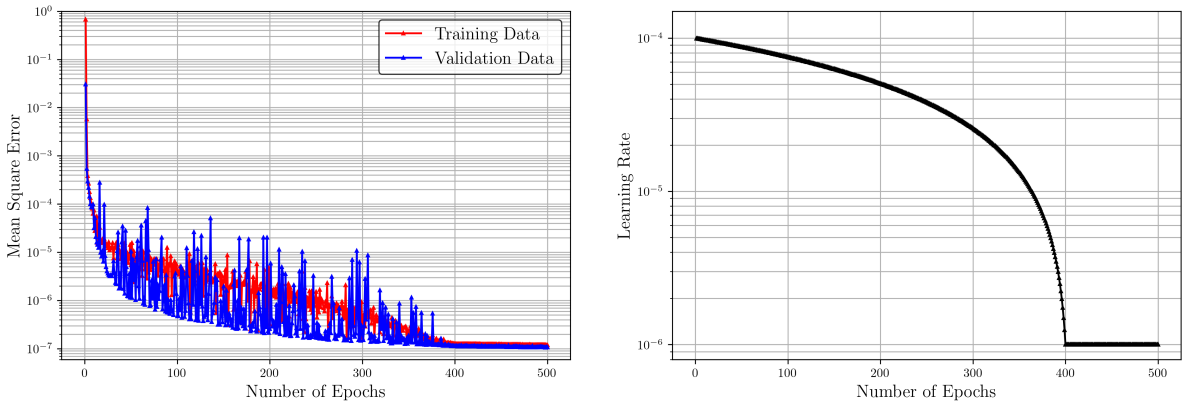


Figure 5.4: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 32$ Spatial Elements

Additional training data was generated by introducing Gaussian noise $\mathcal{N}(0, \sigma)$ with zero mean and a standard deviation of $\sigma = 10^{-4}$. As a result, a dataset of 40,000 samples was used for training and validation. This relatively low standard deviation was selected to preserve the periodicity of the solution in space and time. For the optimization procedure, a batch size of 50 samples was selected and the training was conducted over 500 epochs. The adaptive learning rate decreased over the first 400 epochs following the expression provided in Equation 4.9. An initial value $\gamma_0 = 1 \times 10^{-4}$ and a decay factor $\gamma_r = 1 \times 10^{-2}$ were selected based on trial and error. After the adaptive phase, the learning rate was maintained at a constant value for the remaining epochs.

The MSE loss for the training and validation sets along with the evolution of the adaptive learning rate are shown in Figure 5.4 for a spatial resolution of $N_x = 32$ elements. The training and validation MSE reached approximately $O(10^{-7})$ at the end of the optimization procedure, highlighting the CAE's performance. The observed overshoots suggested instances where the local learning rate might have been excessively high. However, using an adaptive learning rate facilitated the exploration of a broader area around potential local minima. When the learning rate was fixed, the MSE stabilized, indicating convergence to a minimum for this learning rate. Despite this, it was observed that the MSE convergence for the training set was not smooth at the end of the adaptive learning rate stage. A higher reduction of the loss function could be achieved by extending the total number of epochs.

The optimization procedure was repeated for the remaining spatial resolutions since a different CAE architecture was implemented. The training time and MSE of the normalized CAE solution for the spatial

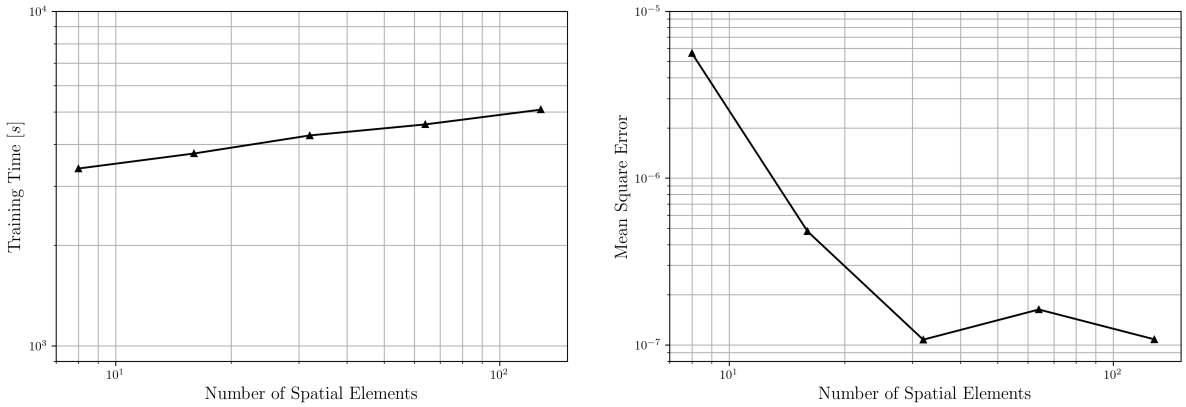


Figure 5.5: Training Time and Mean Square Error for Convolutional Autoencoder of Manufactured Solution for Different Spatial Resolutions

resolutions considered are presented in Figure 5.5. The training time increased marginally with higher spatial resolutions, exhibiting a low growth rate. Meanwhile, the MSE of the predicted primal solution decreased for lower spatial resolutions while reaching a plateau for finer spatial grids. The plateau observed was due to the insufficient total number of epochs selected, which prevented convergence to a more optimal local minimum for higher spatial resolutions.

After training, the CAE was used to generate a surrogate primal solution. As a final step, the Dirichlet BCs were reapplied to ensure consistency with the physical constraints of the problem. The remaining spatial resolutions' architectures and training performance are detailed in section A.1.

5.2.3. Echo State Network

The second method proposed was the ESN to construct a surrogate model of the primal solution. Its architecture consisted of a single hidden layer, known as the reservoir. The reservoir was pseudo-randomly generated, and the ESN was then trained with a linear regression problem to determine the connections between the reservoir and the output layer. To account for its inherent pseudo-randomness, an ensemble of 10 ESN samples was considered, each with a different random seed.

The training data comprised 10,000 time steps, corresponding to the entire primal solution, whereas the validation data consisted of 50 non-overlapping intervals of 200 samples. The washout period consisted of 1,000 time steps. The Bayesian Optimization (BO) procedure aimed to minimize the averaged MSE over the validation splits. The hyperparameter space $\rho \times \sigma_{in} : [0.7, 1.1] \times [10^{-4}, 1.0]$ for the spectral radius and input scaling was explored, where the latter was evaluated in logarithmic scale. The optimal Tikhonov parameter β was selected from the grid space $[10^{-3}, 10^{-6}, 10^{-9}]$.

The number of neurons in the reservoir, denoted as N_r, was also one of the hyperparameters that must be selected. A larger size of the reservoir implies a better performance but it also leads to a higher computational cost of the training procedure and network's deployment [79]. The reservoir size was selected to be equal for all spatial resolutions considered. As the reservoir acts as a non-linear expansion of the inputs, its size must meet a lower bound determined by the highest spatial resolution considered. Thus, the number of neurons $N_r = \{256, 512, 1024, 2048, 4096\}$ were explored for a spatial resolution of $N_x = 32$ elements. The training was repeated for all samples considered. For $N_r = 1024$ neurons, the mean and standard deviation of the Gaussian Process (GP) reconstruction are shown in Figure 5.6 along with the optimal ρ and σ_{in}. The averaged MSE in the validation splits was approximately $O(10^{-11})$ for the samples considered. On the other hand, the GP reconstruction considerably depended on the chosen random seed for the generation of the pseudo-randomized matrices. This was due to the significant standard deviation observed in the search space. As a result, it was necessary to repeat the training for the different ESNs. Regarding the hyperparameters, the optimal ρ was in the range $[0.7, 1.0]$ whereas the optimal σ_{in} lied between $[10^{-2}, 10^{-1}]$ for the majority of the samples.

For the reservoir sizes considered, the CR, training time and averaged MSE over the validation splits for the 10 ESNs ensemble considered are shown in Figure 5.7 for $N_x = 32$ spatial elements. The CR

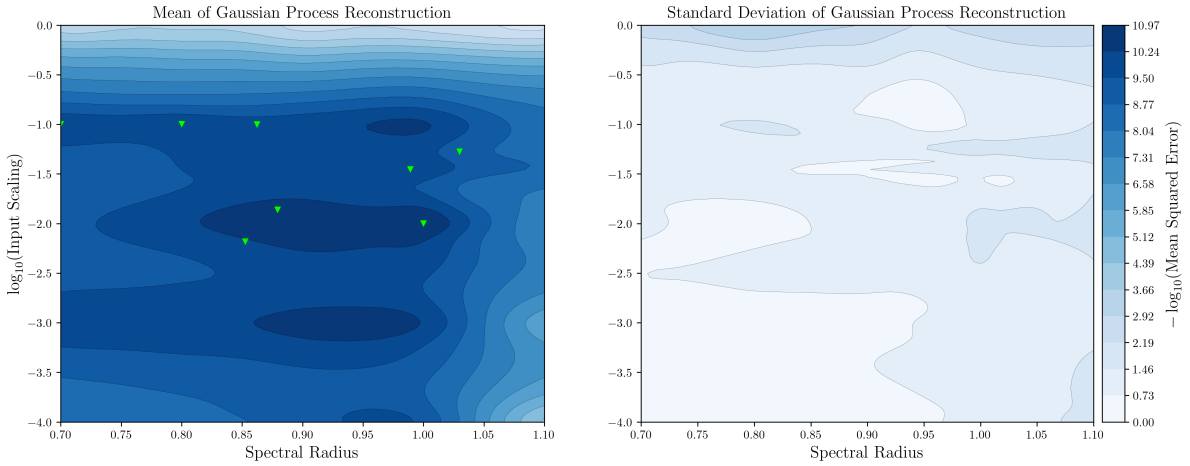


Figure 5.6: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks of Manufactured Solution for $N_x = 32$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

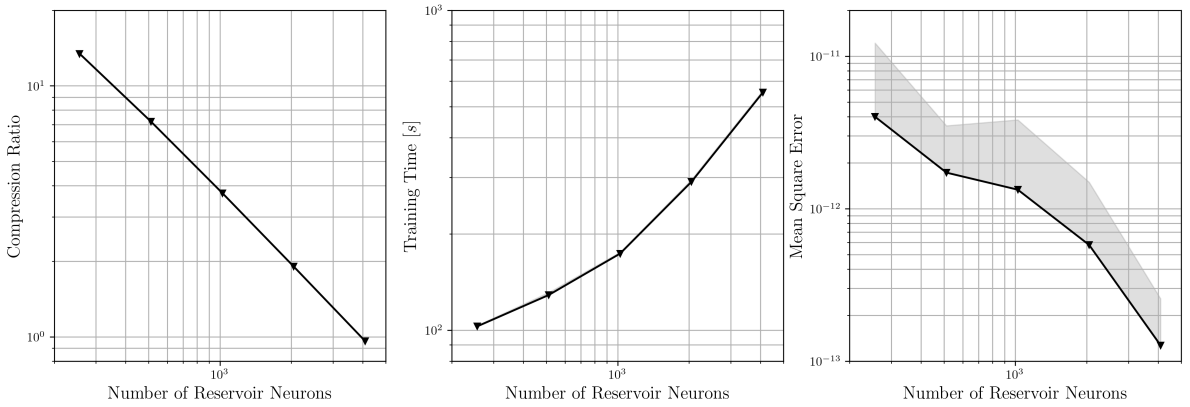


Figure 5.7: Compression Ratio, Training Time and Mean Square Error for Echo State Networks of Manufactured Solution for $N_x = 32$ Spatial Elements and Different Number of Reservoir Neurons

decreased exponentially as the reservoir size increased. Thus, larger reservoirs were less effective at compressing the solution. For the largest reservoir considered, the CR was inferior to unity, meaning that storing the surrogate model was more demanding than the original primal solution. Also, the training time increased substantially with larger reservoir sizes due to the increased count of trainable parameters. The MSE decreased as the reservoir size increased, resulting in improved performance of the ESN. Based on the observed results, a reservoir of $N_r = 1024$ neurons was selected as it provided effective compression capabilities and moderate training time.

After determining the optimal reservoir size, the BO procedure was repeated for the spatial resolutions considered. The training time and averaged MSE over the validation splits for the 10 ESNs ensemble are depicted in Figure 5.8 for each spatial resolution. The training time remained approximately constant across spatial resolutions, as the reservoir size was constant and predominantly determined the matrices' dimensions. The averaged MSE was approximately constant for the spatial resolutions considered. This was also attributed to the constant reservoir size used.

The ESNs were then deployed to predict the primal solution for the spatial resolutions considered. The primal solution was predicted with the use of a checkpoint technique, where the number of checkpoints was identical to the number of validation intervals in the BO procedure. The Dirichlet BCs were then reapplied to ensure consistency with the physical constraints of the primal solution. The BO results for the remaining spatial resolutions are detailed in section B.1.

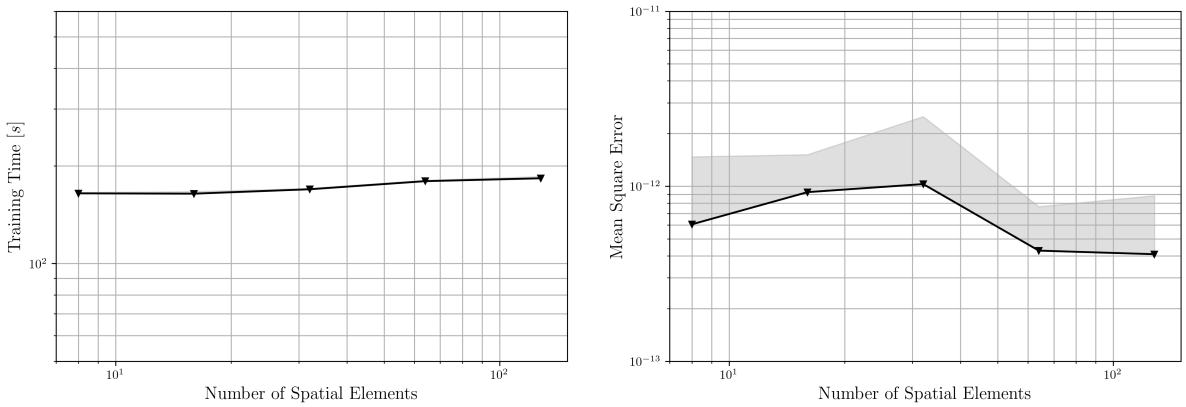


Figure 5.8: Training Time and Mean Square Error for Echo State Networks of Manufactured Solution for Different Spatial Resolutions

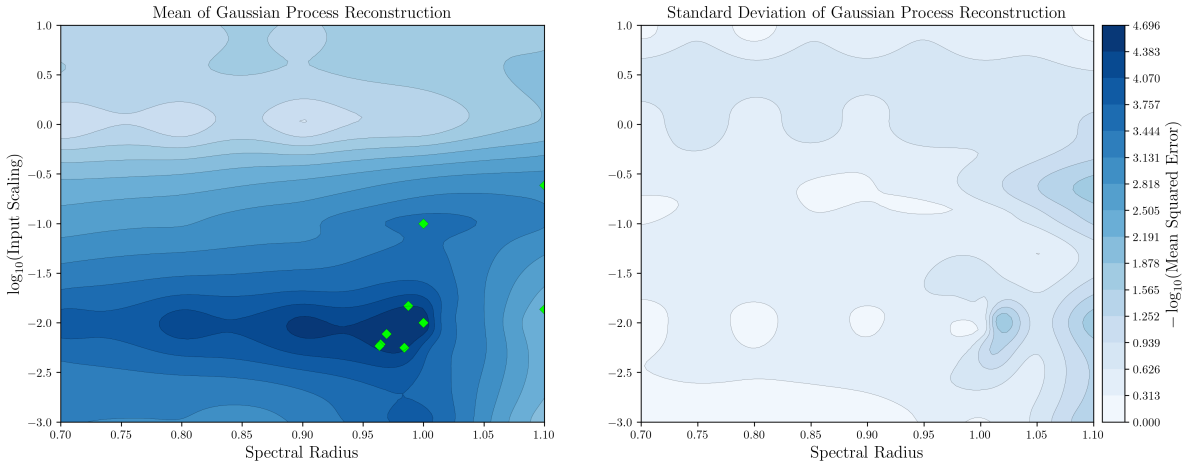


Figure 5.9: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 32$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

5.2.4. Convolutional Autoencoder - Echo State Network

The third method proposed was the CAE-ESN to construct a surrogate primal solution. Although the developed CAE was reused for this method, the ESN required new training. As with the ESN alone, an ensemble of 10 samples was considered to account for the pseudo-randomness of the reservoir.

The same parameters were employed for the BO procedure as in the method using the ESN alone. The only difference lied in the hyper-parameter space explored for the spectral radius and input scaling, which was set to the range $\rho \times \sigma_{in} : [0.7, 1.1] \times [10^{-3}, 10^{1}]$.

For a spatial resolution of $N_x = 32$ elements, a reservoir size study was also conducted in the same search space investigated for the ESN alone. For $N_r = 1024$ neurons, the mean and standard deviation of the GP reconstruction are shown in Figure 5.9 along with the optimal hyperparameters ρ and σ_{in}. The averaged MSE over the validation splits was approximately $O(10^{-5})$ for the mean GP reconstruction. The standard deviation near the optimal hyperparameters location was higher than when using the ESN alone, which indicated a higher sensitivity to the chosen random seed. The lower averaged MSE value and the higher standard deviation were attributed to the increased complexity introduced in the latent space compared to the high-dimensional space. The encoder introduces non-linearities in the latent space, which makes it more challenging for the ESN prediction. On the other hand, the optimal ρ was in the range $[0.9, 1.1]$ while the optimal σ_{in} lied between $[10^{-3}, 1.0]$.

For the reservoir sizes considered, the CR, training time and averaged MSE over the validation splits for the 10 ESNs ensemble applied to the latent space are shown in Figure 5.10 for $N_x = 32$ spatial

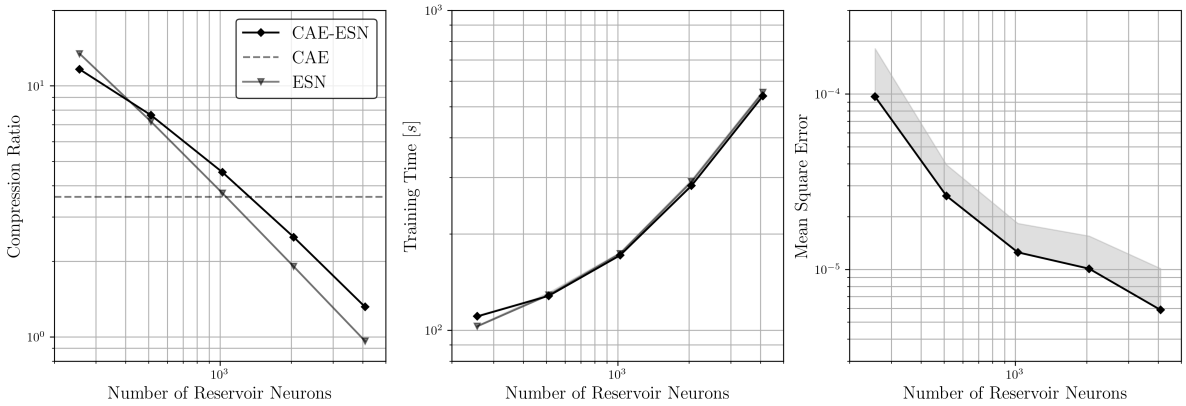


Figure 5.10: Compression Ratio, Training Time and Mean Square Error for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 32$ Spatial Elements and Different Number of Reservoir Neurons

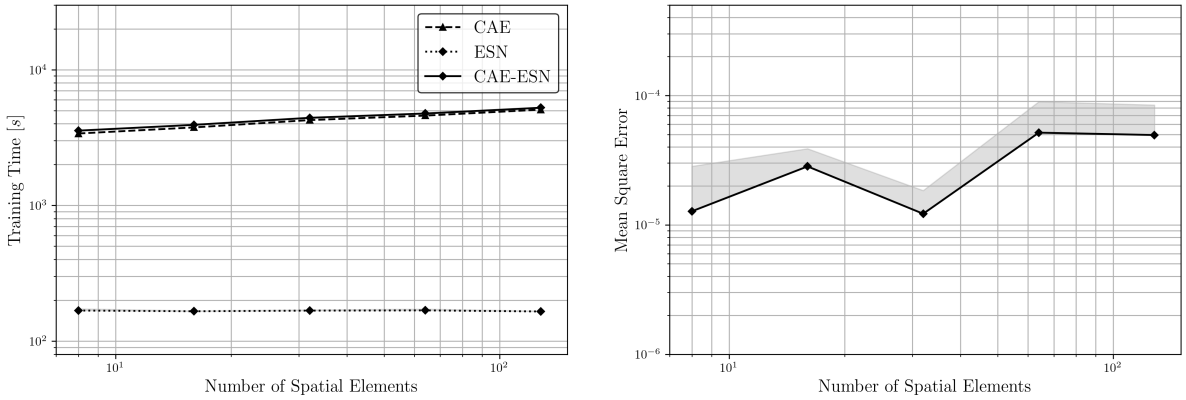


Figure 5.11: Computational Time and Mean Square Error for Echo State Networks Applied to Latent Space of Manufactured Solution for Different Spatial Resolutions

elements. The CR of using the CAE independently as well as the ESN's CR and training time are shown for comparison. Compared to using the ESN independently, a higher number of reservoir neurons did not reduce the CR as significantly in the current method. This was due to the lower count of trainable parameters resulting from the reduced size of the latent space. The ESN was applied directly to the latent space produced by the decoder to enhance the compression capabilities achieved by the CAE. This improvement was observed only when the number of neurons was in the range $N_r \leq 1024$. Additionally, the training time was approximately equal to the one presented by the ESN alone. As expected, the MSE of the latent space decreased with an increase in reservoir size, leading to enhanced ESN performance. Based on the observed results, a reservoir size of $N_r = 1024$ neurons was selected as it provided a higher CR than the CAE and a moderate training time.

Following the optimal reservoir size selection, the latent space was predicted with the ESNs for the spatial resolutions considered. The training time and averaged MSE of the reconstructed latent space are shown in Figure 5.11 for the spatial resolutions considered. The training time includes the CAE and the ESN training times. As observed with the ESN alone, the training time remained nearly constant, primarily influenced by the constant reservoir size across the spatial resolutions considered. The integration of the ESN into the CAE framework had a negligible impact on the overall training time, as the increase due to the new ESN training was minimal when compared to the training time required for the CAE. The averaged MSE over validation splits was between $O(10^{-5})$ and $O(10^{-4})$ for the spatial resolutions considered. The variations observed could be due to the different latent space complexity across spatial resolutions. The complexity of the latent space could not be controlled by the user with the methodology proposed.

The primal latent space was then predicted in the time domain with the ESNs implemented. A check-

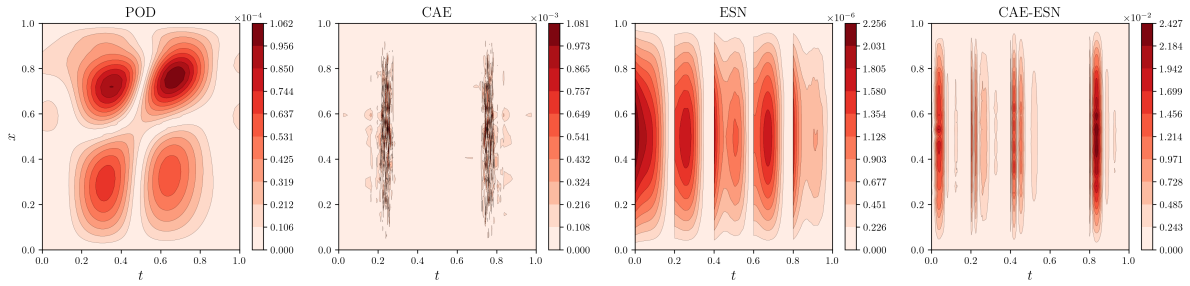


Figure 5.12: Mean Relative Error of Surrogate Primal Solutions of Manufactured Solution for $N_x = 32$ Spatial Elements

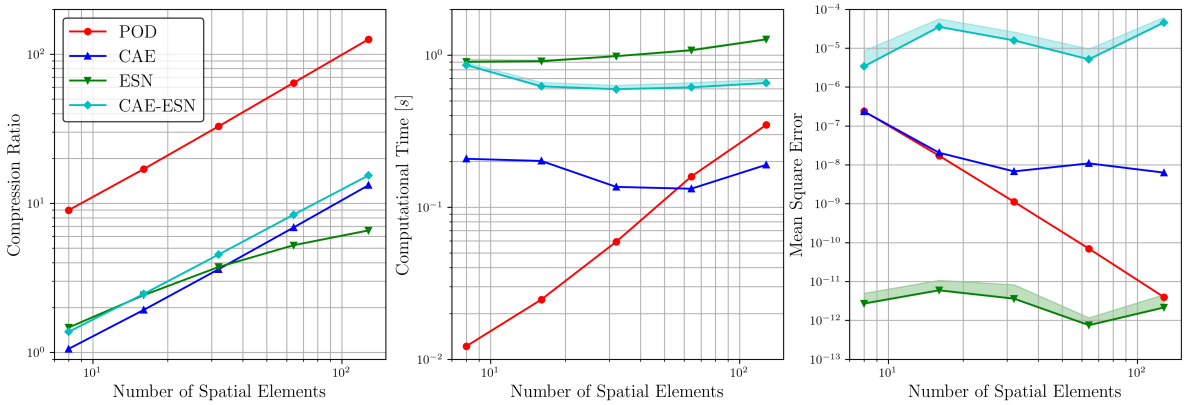


Figure 5.13: Evaluation Metrics of Surrogate Primal Solutions of Manufactured Solution for Different Spatial Resolutions

point technique was also introduced and identical to the validation dataset split. Following the latent space prediction, the decoder of the CAE was deployed to reconstruct the data back to the original high-dimensional space. Once the solution was predicted using the CAE-ESN, the Dirichlet BCs were reapplied. More details on the optimization of the ESNs applied to the latent space for the different spatial resolutions considered are presented in section C.1.

5.2.5. Performance Assessment

The performance of the different surrogate models for the primal solution is now evaluated. Given the primal solution's periodicity, the relative error was averaged over different time periods. The mean relative error of the surrogate primal solutions is illustrated in Figure 5.12 for $N_x = 32$ spatial elements, where for the ESN and CAE-ESN only one of the samples is presented. The POD primal solution exhibited a relative error of $O(10^{-4})$. This error arose from high-wave number phenomena that the POD did not fully capture when selecting the optimal rank. In contrast, the CAE displayed a mean relative error of $O(10^{-3})$. However, this error was much more localized than the POD. Furthermore, the ESN demonstrated the lowest relative error of $O(10^{-6})$. The influence of the checkpoint technique was observed, which effectively prevented the accumulation of errors as the backward prediction in time progressed. Lastly, the CAE-ESN model presented a mean relative error of $O(10^{-2})$. Similar to the CAE, the error remained localized and, like the ESN, the use of the checkpoint technique mitigated error accumulation as time progressed backwards. The CAE-ESN was constructed from the CAE implemented, thus the accuracy of the CAE-ESN was bounded by the accuracy of the CAE. This characteristic was observed in the errors of the CAE and CAE-ESN.

The evaluation parameters considered, including the CR, computational time and MSE are presented in Figure 5.13 for the spatial resolutions considered. The training time was not included in the computational time of the deep learning techniques, as these were heavily influenced by user-defined factors. Also, the training time could be potentially done once for the cases considered. In terms of CR, all methods demonstrated a monotonically increasing trend as the number of spatial elements increased. However, the ESN reached a plateau in CR at the highest spatial resolutions. The primal solution's

sinusoidal nature allowed the POD to achieve the highest CR due to its unit optimal rank. Additionally, the CR of the CAE-ESN consistently exceeded that of the standalone CAE, as it was required for efficient compression. Regarding computational time, the POD outperformed the other methods at lower spatial resolutions. For higher resolutions, the CAE achieved the lowest computational time. The ESN-based methods required additional computational cost due to dense matrix multiplications. In contrast to the ESN alone, the CAE-ESN computational time was approximately constant due to the ESN being applied to the low-dimensional latent space. When considering the MSE, only the POD exhibited a strictly monotonic decrease as the number of spatial elements increased. The CAE showed a decreasing MSE at low spatial resolutions but reached a plateau at higher resolutions, attributed to the total number of epochs imposed. In contrast, the MSE for the ESN and CAE-ESN was largely independent of the number of spatial elements due to the use of a constant reservoir size. The ESN was the most accurate method at the lowest spatial resolutions, with errors of five orders of magnitude lower. However, the POD presented an accuracy similar to the ESN for finer spatial grids.

Ultimately, in constructing a surrogate primal solution, the ESN excelled at the lowest spatial resolutions while maintaining a reasonable CR. For a higher number of spatial elements, the POD was the most suitable method, offering a higher CR and a similar MSE with a reduced computational time.

5.3. Adjoint Problem Formulation and Solution

The adjoint problem was derived using the continuous technique elaborated in section 2.1. Given the non-linearity of the Burgers' equation, a local linearization centred around a specific primal state u was necessary. Starting from the residual of the considered primal PDE, the residual sensitivity was defined as follows:

$$R'[u](\delta u) = \frac{\partial(\delta u)}{\partial t} + (\delta u)\frac{\partial u}{\partial x} + u\frac{\partial(\delta u)}{\partial x} - \frac{1}{Re}\frac{\partial^2(\delta u)}{\partial x^2} \tag{5.7}$$

where R is the primal residual and δu is an infinitesimal variation of the primal solution. Based on the generalized form of the continuous adjoint equation, defined in Equation 2.6, integration by parts was applied to derive the adjoint operator:

$$\begin{aligned} J'[u](\delta u) &= \int_I \int_\Omega \psi \left(\frac{\partial(\delta u)}{\partial t} + (\delta u)\frac{\partial u}{\partial x} + u\frac{\partial(\delta u)}{\partial x} - \frac{1}{Re}\frac{\partial^2(\delta u)}{\partial x^2} \right) d\Omega dI \\ &= \int_I \int_\Omega \left(-\frac{\partial \psi}{\partial t} - u\frac{\partial \psi}{\partial x} - \frac{1}{Re}\frac{\partial^2 \psi}{\partial x^2} \right) (\delta u)\, d\Omega dI + \left[\int_\Omega \psi(\delta u)\, d\Omega \right]_{\partial I} + \frac{1}{Re}\left[\int_I \psi \frac{\partial(\delta u)}{\partial x}\, dI \right]_{\partial \Omega} \end{aligned} \tag{5.8}$$

This process yielded the adjoint PDE along with the corresponding BCs and IC:

$$\begin{cases} -\dfrac{\partial \psi}{\partial t} - u\dfrac{\partial \psi}{\partial x} - \dfrac{1}{Re}\dfrac{\partial^2 \psi}{\partial x^2} = \sin(\pi x), & (x,t) \in \Omega \times I \\ \psi(x,t) = 0, & (x,t) \in \partial\Omega \times I \\ \psi(x,T) = 0, & x \in \Omega \end{cases} \tag{5.9}$$

As expected, the negative sign preceding the partial time derivative and the IC set at $t = T$ signified that the adjoint problem marched backwards in time. Additionally, despite the non-linearity of the primal problem, the adjoint PDE was linear. The adjoint problem was solved using the implemented solver. Similar to the primal problem, the adjoint weak formulation needed to be derived. By considering weighting functions $q \in V_\psi$, the following weak formulation was derived after integration by parts:

$$\int_\Omega \left(-\frac{\partial \psi}{\partial t} q - u\frac{\partial \psi}{\partial x} q + \frac{1}{Re}\frac{\partial \psi}{\partial x}\frac{\partial q}{\partial x} - \sin(\pi x) q \right) d\Omega = 0 \tag{5.10}$$

Continuous piece-wise linear elements were again chosen for spatial discretization of the adjoint problem. The solving environment using `FEniCS` was the same as for the primal problem.

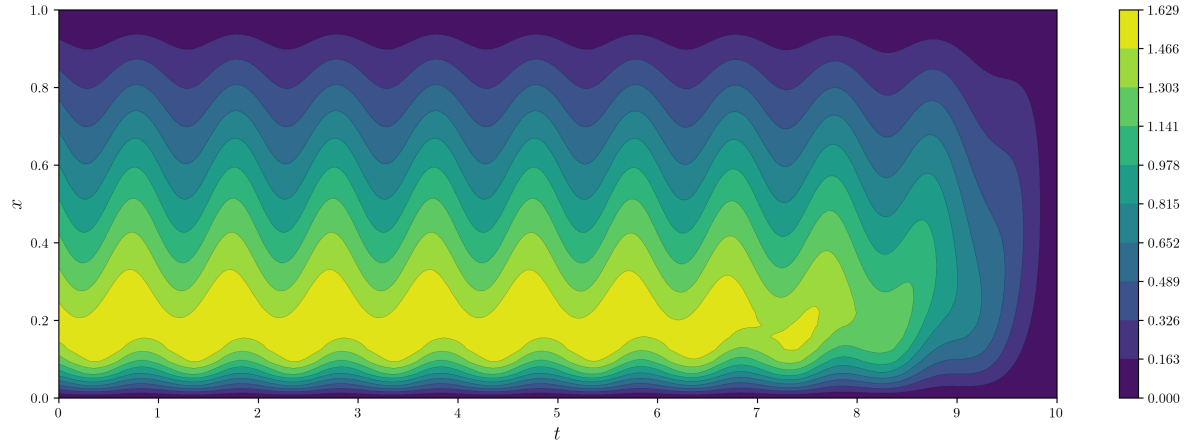


Figure 5.14: Adjoint Solution of Manufactured Solution for $N_x = 32$ Spatial Elements

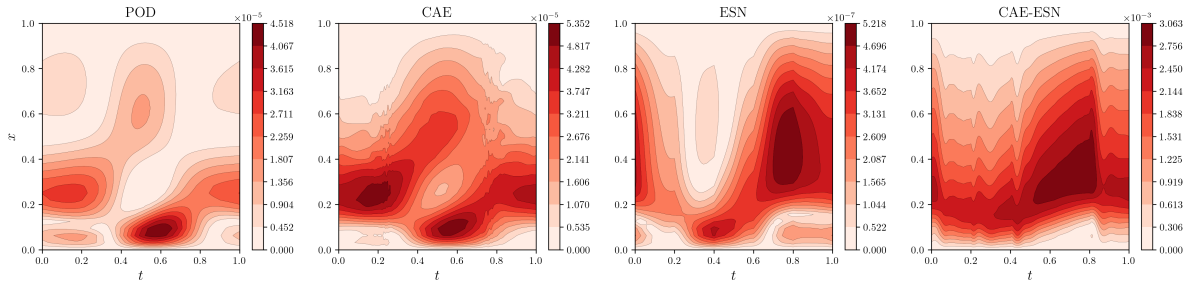


Figure 5.15: Mean Relative Error of Surrogate Adjoint Solutions of Manufactured Solution for $N_x = 32$ Spatial Elements

The adjoint discrete solution ψ computed with the FEM primal solution is shown in Figure 5.14 for $N_x = 32$ spatial elements. After an initial transient phase due to the homogeneous IC at $t = T$, the adjoint solution exhibited a periodic behaviour over the period $t \in [0, T/2]$. The maximum value of the adjoint solution was observed near the starting point of the spatial domain. The adjoint solution was statistically-steady because the primal solution and the adjoint forcing term were sinusoidal functions. Furthermore, the adjoint time period was consistent with the one presented by the primal solution.

Surrogate adjoint solutions were computed using the surrogate primal solution. To assess the error introduced by the surrogate primal solutions, the relative error of the surrogate adjoint solutions was averaged over the time periods where the adjoint solution was observed to be statistically-steady. The mean relative error of the surrogate adjoint solutions is illustrated in Figure 5.15 for $N_x = 32$ spatial elements, with only one sample presented for the ESN and CAE-ESN. The ESN demonstrated superior accuracy with a mean relative error of $O(10^{-7})$, while the CAE-ESN displayed the highest error of approximately $O(10^{-3})$. Even though the POD produced a more accurate primal solution than the CAE, the mean relative error of both methods in the adjoint solution computation was almost identical. This is attributed to the CAE's primal solution error being more localized than the POD one. Ultimately, all surrogate models were capable of delivering an accurate adjoint solution. Additionally, the pattern of the mean relative error was similar across the methods used. This was attributed to the greater sensitivity of the adjoint solution to its forcing term rather than to the primal solution.

The MSE of the surrogate adjoint solutions with respect to the FEM adjoint solution is presented in Figure 5.16 for the spatial resolutions considered. For coarser grids, the ESN outperformed the other methods, achieving a MSE of approximately $O(10^{-12})$. At higher spatial resolutions, the POD and CAE demonstrated performance similar to the ESN. The performance of these methods in solving the adjoint problem mirrored their accuracy in constructing a surrogate model of the primal solution, as shown in Figure 5.13. This correlation was expected due to the linear nature of the adjoint PDE.

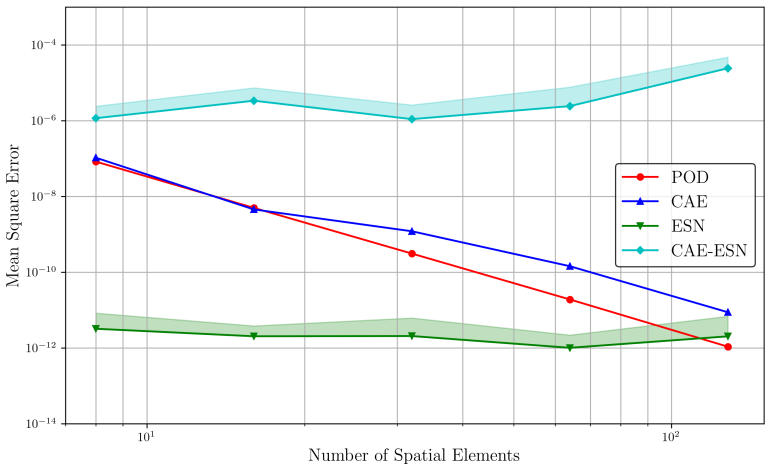


Figure 5.16: Mean Square Error of Surrogate Adjoint Solutions of Manufactured Solution for Different Spatial Resolutions

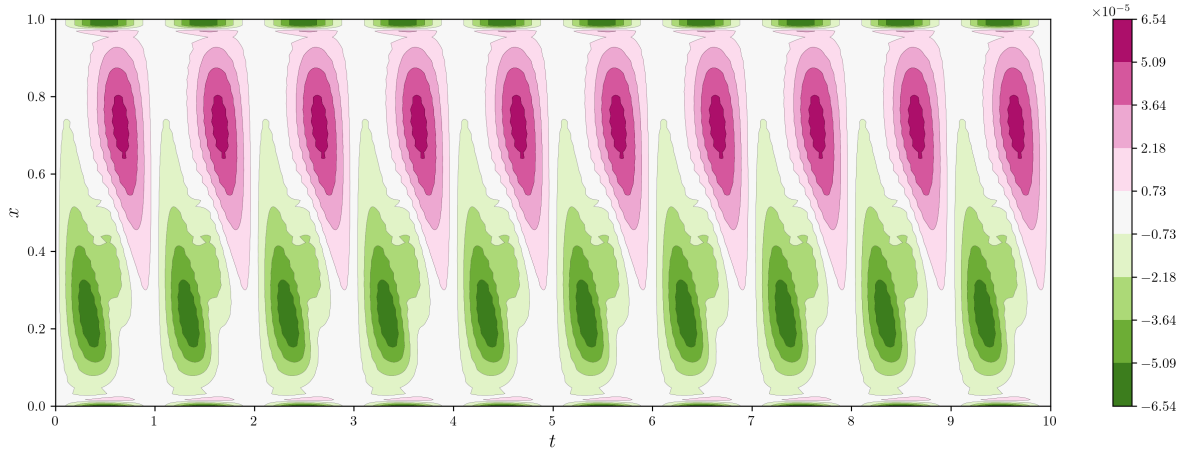


Figure 5.17: Primal Injected Residual of Manufactured Solution for $N_x = 32$ Spatial Elements

5.4. Adjoint-Based Error Estimation

Adjoint-based error estimation requires the computation of the primal and adjoint solutions within coarse and fine spaces. The injection of the coarse primal solution into the fine space is conducted with a linear interpolation of the coarse primal solution, defined within the coarse function space V_H, into the fine function space V_h. The Dirichlet BCs were reinforced in the injected primal solution to mitigate interpolation errors at the boundaries. Subsequently, the strong fine-space residual was evaluated in a discontinuous Galerkin function space due to its discontinuous nature. The same function space was used for the error estimate and error indicator computations. The computation of the strong residual necessitates the evaluation of the first and second spatial derivatives of the primal solution. These were acquired by projecting the corresponding derivatives of the primal solution into quadratic function spaces. This approach aimed to avoid zero second spatial derivatives due to the use of linear elements in the implemented solver. Also, the derivative of piece-wise continuous functions is discontinuous. This was not consistent in the evaluation of the derivatives, thus a L_2 projection was employed which conserves the energy in the element.

The fine-space discrete residual $\boldsymbol{R}_h$ computed using the injected primal discrete solution $\boldsymbol{u}_h^H$ with $N_x = 32$ coarse-space elements is illustrated in Figure 5.17. Similar to the primal solution, the strong fine-space residual displayed a periodic behaviour centred around the spatial midpoint. Additionally, a higher value of the residual was observed near the spatial endpoint, which might be attributable to the injection of the coarse-space primal solution into the fine space followed by the BCs reinforcement.

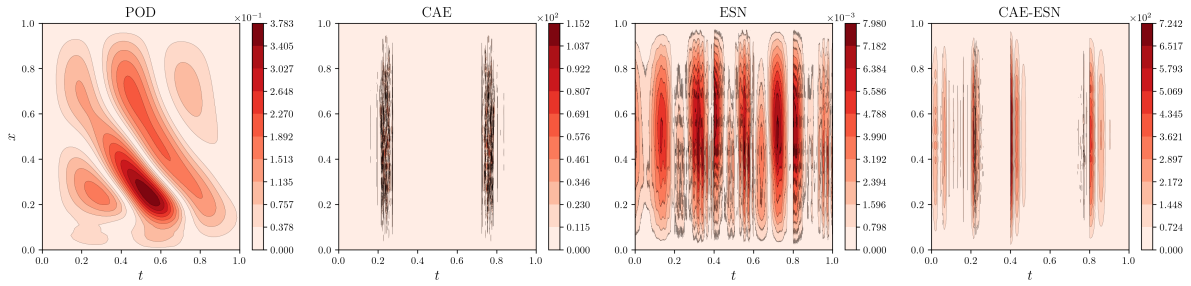


Figure 5.18: Mean Relative Error of Surrogate Primal Injected Residuals of Manufactured Solution for $N_x = 32$ Spatial Elements

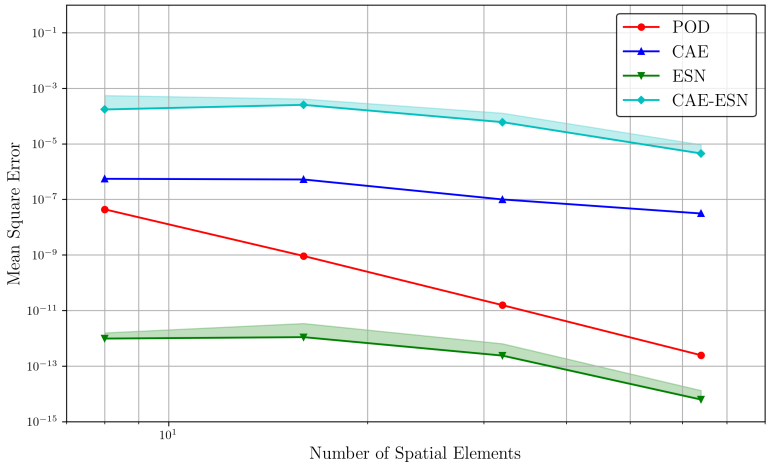


Figure 5.19: Mean Square Error of Surrogate Primal Injected Residuals of Manufactured Solution for Different Spatial Resolutions

As with the primal and adjoint solutions, the error introduced in the surrogate injected residuals was evaluated by calculating the average relative error in regions where the injected residual was periodic. The mean relative error of the surrogate injected residuals is shown in Figure 5.18 for $N_x = 32$ spatial elements, with one sample shown for the ESN and CAE-ESN. Only the relative errors presented by the POD and ESN indicated a reasonable prediction of the injected residual with an error of approximately $O(10^{-1})$ and $O(10^{-2})$ respectively. In contrast, the mean relative errors presented by the CAE and CAE-ESN were approximately $O(10^{2})$. These higher errors were highly localized but the residual was inaccurate in these regions.

The MSE of the surrogate injected residuals with respect to the FEM injected residual is presented in Figure 5.19 for the coarse-space resolutions considered. All of the methods considered displayed a monotonic decrease in MSE as the number of spatial elements increased, where the POD presented the steepest rate. However, the ESN standed out as the most accurate method to produce an accurate injected residual due to providing the most accurate surrogate primal solution.

Following the computation of the refined adjoint solution and fine-space primal injected residual, the time-averaged adjoint-based error estimates $\delta\bar{J}_{est}$ were computed for the FEM and surrogate solutions. The time-averaged QoIs $\bar{J}+\delta\bar{J}_{est}$, the adjoint-based error estimates $\delta\bar{J}_{est}$ and the absolute error in true output error $\left|\delta\bar{J}_{est} - \delta\bar{J}\right|$ are shown in Figure 5.20 for the methods proposed and spatial resolutions considered. As the number of spatial elements increased, the FEM adjoint-corrected QoI converged towards the true value, indicating improved accuracy. The FEM adjoint-corrected QoI was more accurate than the coarse-space QoI for the spatial resolutions considered. Additionally, the adjoint-based error estimate decreased with an increasing number of spatial elements. This behaviour resulted from the refinement of the coarse space, leading to a more accurate QoI estimation. Furthermore, the difference between the true error estimate $\bar{J}_h - \bar{J}_H$ and the computed error estimate approximations stemmed from the linearization error $O(\delta\boldsymbol{u}^2)$ inherent in non-linear problems [26]. These results aligned

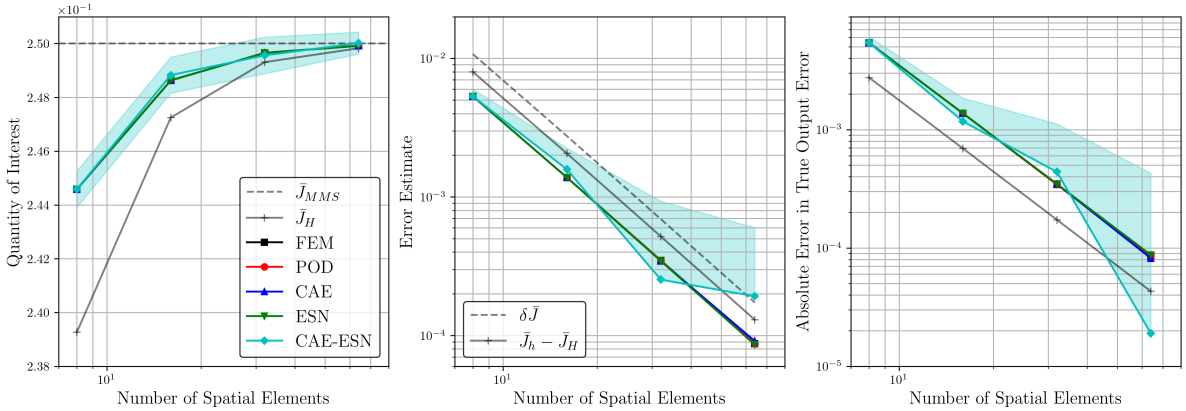


Figure 5.20: Quantity of Interest and Error Estimates of Surrogate Solutions of Manufactured Solution for Different Spatial Resolutions

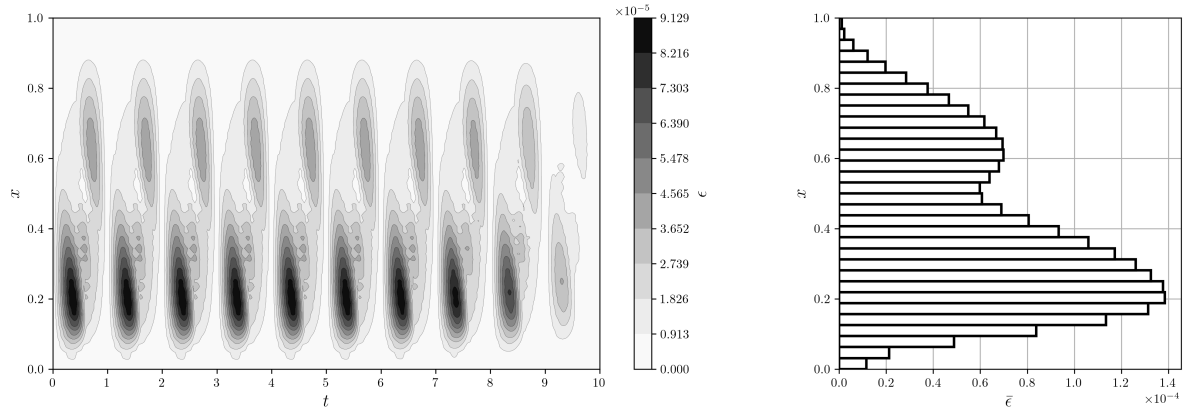


Figure 5.21: Error Indicator Field and Time-Averaged Error Indicators of Manufactured Solution for $N_x = 32$ Spatial Elements

with expectations, confirming the validity of the adjoint-based error estimation framework. Regarding the methodologies proposed, the majority of the surrogate error estimates computed were accurate, as they closely matched those computed with the FEM solution. Only the CAE-ESN error estimates deviated for finer spatial grids but with an appropriate QoI estimate for lower spatial resolutions.

Local error indicators were computed which could be used to evaluate their suitability for mesh adaptation methods. Time-averaged local error indicators were computed as the errors from the temporal discretization scheme were negligible. The coarse-space error indicator field and its time-averaged counterpart are presented in Figure 5.21 for $N_x = 32$ spatial elements. The time average was calculated only during the time periods where the adjoint solution was periodic. By definition, the adjoint solution and the injected residual directly influenced the local error indicator field. In this case, the shape of the local error indicator field closely resembled the shape of the injected residual field. Also, the maximum value of the local error was observed near the starting point of the spatial domain, similar to what was observed for the adjoint solution. By considering a time-averaged error indicator, the spatial regions that require mesh refinement or coarsening can be identified to enhance the QoI prediction. Based on the results presented, greater refinement in the first half of the spatial domain would be necessary than in the last half.

The mean relative error of the surrogate local error indicator field computed using the methods proposed is shown in Figure 5.23 for $N_x = 32$ spatial elements, with one sample shown for the ESN and CAE-ESN. Similar to the observations made for surrogate injected residuals, only the POD and ESN exhibited an accurate prediction of the error indicator field. Nonetheless, the mean relative error presented by the CAE and CAE-ESN was highly localized. This behaviour was attributed to the error present in the injected residual.

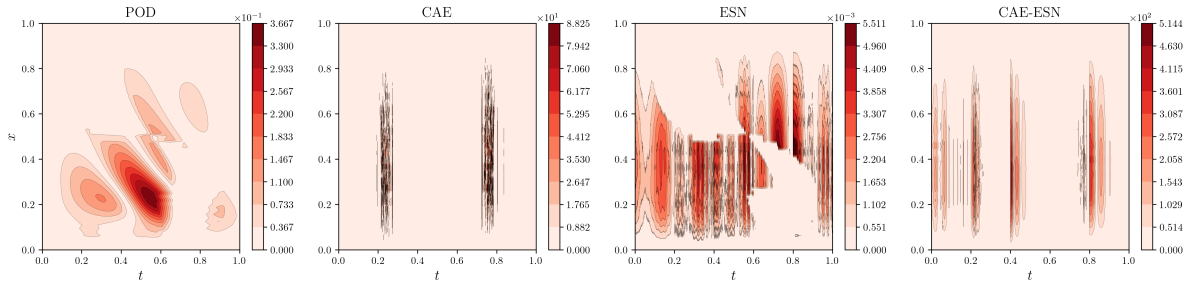


Figure 5.22: Mean Relative Error of Surrogate Error Indicator Fields of Manufactured Solution for $N_x = 32$ Spatial Elements

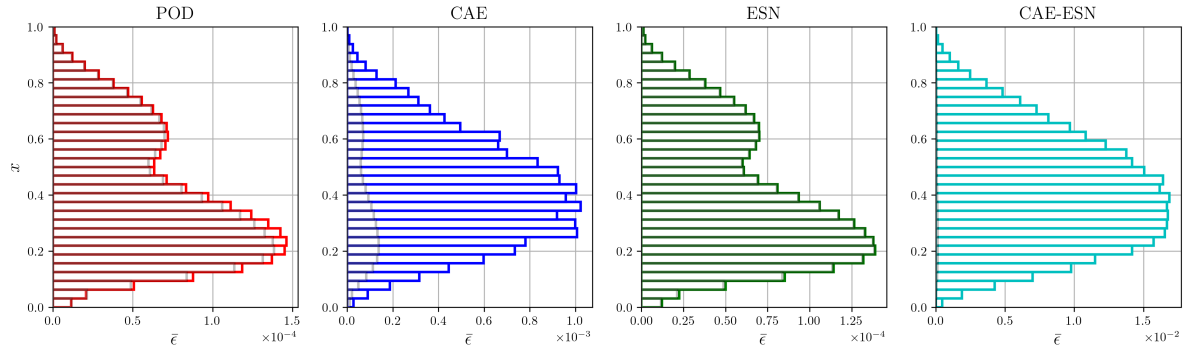


Figure 5.23: Time-Averaged Error Indicators of Surrogate Solutions of Manufactured Solution for $N_x = 32$ Spatial Elements (Finite Element Method Results in Grey)

The surrogate time-averaged error indicators are shown in Figure 5.23 for $N_x = 32$ spatial elements. For reference, the FEM error indicator is also displayed in grey. The ESN was the most accurate method for representing the shape and magnitude of the time-averaged error indicator. The POD also accurately captured its shape but with errors in magnitude. The CAE and CAE-ESN struggled to represent the magnitude accurately due to the errors present in the injected residual. However, their shapes still resembled the FEM field, making them viable options for use in mesh adaptation, as greater refinement in the first half of the spatial domain would still be necessary compared to the last half.

All the methods considered were able to accurately compute the error estimates, with larger deviations observed for the CAE-ESN at higher spatial resolutions. However, Adaptive Mesh Refinement (AMR) is relevant for coarser meshes, where minimizing computational cost is crucial. When it comes to local error indicators, only the POD and ESN accurately captured both the magnitude and shape, making these two methods the most suitable for application to the manufactured solution. Although the error in magnitude of the time-averaged error indicators was pronounced for the CAE and CAE-ESN, its impact on the overall error estimates was not as significant. These methods were able to accurately capture the adjoint solution but not the injected residual, where the local error in the latter was localized. The magnitude of the adjoint solution was higher than the magnitude of the injected residual by five orders of magnitude. As a result, the CAE and CAE-ESN were able to accurately compute the adjoint-corrected QoI as the adjoint solution had a higher influence on its computation. Also, the adjoint error estimate provides a global evaluation as it is the sum of the error indicators, thus being less sensitive to outliers. On the contrary, the error indicators are more sensitive to outliers as they indicate the local contribution of an element to the numerical error. As a result, the residual error in the CAE and CAE-ESN had a significant impact on the time-averaged error indicator fields. In general, an accurate QoI correction is expected as long as the adjoint solution is accurately computed and the injected residual error is within manageable bounds. Accurate time-averaged error indicators require also an accurate computation of the injected residual. Within the context of unsteady adjoint-based error estimation to reduce storage requirements, the ESN was the preferred method for lower spatial resolutions due to its superior accuracy and moderate CR. The POD was the best method for finer spatial grids since it demonstrated a much higher CR with better accuracy than the ESN.

6

Results: Unsteady 1D Viscous Burgers' Equation

In this chapter, the test case is the unsteady one-dimensional (1D) viscous Burgers' equation, with a solution driven by a Turbulent Channel Flow (TCF) dataset to produce the wall-normal velocity component. The methodologies proposed to construct a surrogate model of the primal solution were applied to mitigate the storage requirements associated with unsteady adjoint-based error estimation.

Firstly, the primal Partial Differential Equation (PDE) is introduced along with the primal velocity obtained using the Finite Element Method (FEM), implemented with the `FEniCS`[1] package. Furthermore, surrogate models of the primal velocity were computed using the methodologies proposed and their performance was evaluated for different evaluation metrics. Subsequently, the adjoint governing equation is presented, followed by the adjoint velocity obtained with the implemented solver. Finally, error estimates were calculated based on the numerical solutions obtained with the implemented solver and the methodologies proposed. Local error indicator fields were then generated which could be used for mesh adaptation.

6.1. Primal Problem Formulation and Solution

A Direct Numerical Simulation (DNS) dataset of a TCF at a friction Reynolds number $Re_\tau = 180$ was used to force the unsteady 1D viscous Burgers' equation, producing the wall-normal velocity component. The friction Reynolds number is the key control parameter in wall-based turbulent flows [87] and it is defined as:

$$Re_\tau = \frac{u_\tau \delta}{\nu} \tag{6.1}$$

where u_τ is the friction velocity, δ is the channel half-height and ν is the fluid's kinematic viscosity. A spectral-element DNS solver was employed for the TCF simulation from which the required data was extracted. The DNS dataset was evaluated at a specific streamwise coordinate x and spanwise coordinate z to force the 1D Burgers' equation. As a result, the 1D Burgers' equation was equivalent to the wall-normal momentum equation of the Navier-Stokes equations at the selected streamwise and spanwise location. The resulting DNS dataset was down-sampled to a constant time step of $\Delta t = 1 \times 10^{-4}$ and a uniform spatial resolution of $N_y = 256$ elements.

For the 1D Burgers' equation, the space-time domain considered was $\Omega : [-\delta, \delta] \times I : [0, T]$ with $\delta = 1.0$ and $T = 5.0$. In alignment with the wall-normal velocity conditions from the DNS simulation, homogeneous Dirichlet Boundary Conditions (BCs) were imposed along with an Initial Condition (IC) to match the DNS initial velocity. As a result, the following primal PDE was formulated:

[1] `https://fenicsproject.org`

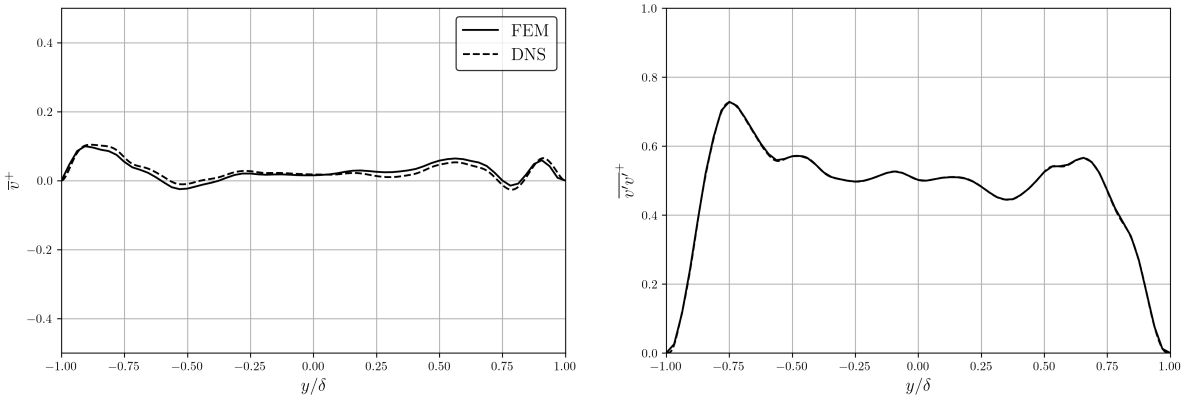


Figure 6.1: Mean Primal Velocity and Reynolds Stress of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements

$$\begin{cases} \dfrac{\partial v}{\partial t} + v\dfrac{\partial v}{\partial y} - \dfrac{1}{Re_\tau}\dfrac{\partial^2 v}{\partial y^2} = f_{\text{DNS}}, & (y,t) \in \Omega \times I \\ v(y,t) = 0, & (y,t) \in \partial\Omega \times I \\ v(y,0) = v_{\text{DNS}}(y,0), & y \in \Omega \end{cases} \tag{6.2}$$

where y and t are the wall-normal and time coordinates, v is the wall-normal velocity and f_{DNS} is the forcing term computed from the DNS dataset. The term f_{DNS} was given by the following expression:

$$f_{\text{DNS}} = \left[-u\frac{\partial v}{\partial x} - w\frac{\partial v}{\partial z} - \frac{1}{\rho}\frac{\partial p}{\partial y} + \frac{1}{Re_\tau}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2}\right)\right]_{\text{DNS}} \tag{6.3}$$

where u and w are the streamwise and spanwise velocity components, ρ is the density and p is the pressure. The derivatives necessary to evaluate the term f_{DNS} were computed using second-order finite difference schemes, except at the boundaries where first-order schemes were applied.

Since the FEM was used for the spatial discretization of the primal problem, the primal weak formulation must be derived. Given weighting functions $q \in V_v$, where the function space $V_v(\Omega)$ was equal for the primal and weighting functions, the following weak formulation was obtained after integration by parts and strong imposition of the Dirichlet BCs:

$$\int_\Omega \left(\frac{\partial v}{\partial t} q + v\frac{\partial v}{\partial y} q + \frac{1}{Re_\tau}\frac{\partial v}{\partial y}\frac{\partial q}{\partial y} - f_{\text{DNS}} q \right) d\Omega = 0 \tag{6.4}$$

Continuous piece-wise linear elements were selected as the finite elements for spatial discretization. The non-linear variational problem was solved using the `FEniCS` environment employed for the manufactured solution. Additionally, a time step interval $\Delta t = 1 \times 10^{-4}$ was selected to match the one from the DNS dataset. This time step interval was sufficient to ensure stability without the need for additional stabilization techniques. Also, errors from the temporal discretization scheme were minimized. Thus, it was assumed that the output error was solely due to spatial discretization.

The spatial resolutions $N_y = \{16, 32, 64, 128, 256\}$ were examined with particular emphasis on $N_y = 64$ spatial elements. The Reynolds decomposition was applied to the computed wall-normal velocity $v = \overline{v} + v'$, dividing it into a statistically-averaged component $\overline{v}$ and a fluctuating component v'. The wall-normal Reynolds stress, a term of the Reynolds-averaged momentum equation, was used to evaluate the fluctuating part. The wall-based mean component and the Reynolds stress computed are shown in Figure 6.1 for $N_y = 64$ spatial elements, with DNS results included for comparison. A superscript $+$ denotes quantities normalized by viscous scales. In TCF, the mean streamwise velocity $\overline{u}$ depends solely on the coordinate y, while the mean spanwise velocity $\overline{w}$ is zero throughout the channel [88].

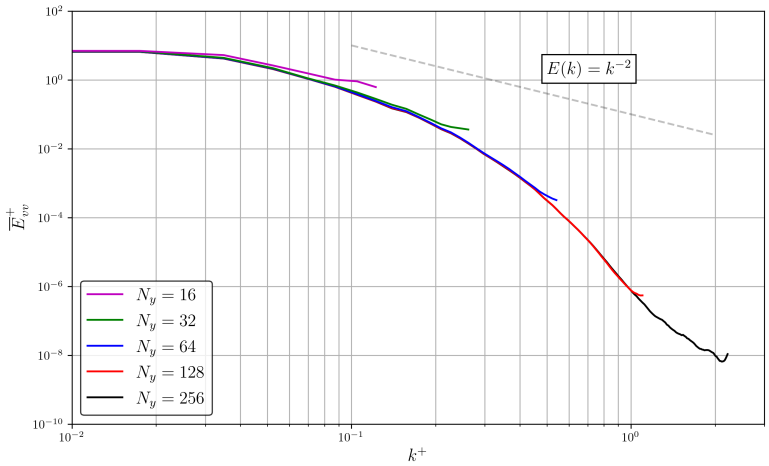


Figure 6.2: Mean Wall-Based Energy Spectrum of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

Consequently, considering the Dirichlet BCs applied to v, the Reynolds-averaged continuity equation implies $\overline{v} = 0$. This conclusion was supported by the mean results presented by the DNS dataset and the implemented solver, where $\overline{v}^+ \approx 0$ across the space domain. The Reynolds stress reached a maximum at an approximate distance of $y/\delta = 0.25$ from the walls. This distance corresponds to a wall-based distance of $y^+ = 45$, thus being within the log-law region as expected for the imposed Re_τ value [89]. The results presented by the DNS dataset and the implemented solver were similar in terms of mean velocity and Reynolds stress. Extending the temporal domain would lead to a mean velocity closer to zero and a more symmetrical distribution of the Reynolds stress about the channel centerline.

The energy cascade of the Burgers' equation relies on different mechanisms than those present in the Navier–Stokes equations [90]. Nonetheless, the former remains useful for testing Large Eddy Simulation (LES) algorithms due to its clear interpretation. The energy cascade of the Burgers' equation presents a rate k^{-2} in the inertial range [90]. The inertial range corresponds to the intermediate turbulent scales and k is the wavenumber. A Fourier representation of the fluctuating components was necessary to obtain the energy spectra at a given time step. Given the discrete characteristic of the velocity, a Fast Fourier Transform (FFT) was applied to convert the fluctuating components from physical space to the frequency domain at a given time step:

$$v'(y) = \sum_k \hat{v}(k) e^{iky} \tag{6.5}$$

where $\hat{v}$ are the complex Fourier coefficients and i is the imaginary unit such that $i^2 = -1$. The spectral energy density in the wall-normal direction E_{vv} is defined as the energy per wavenumber:

$$E_{vv}(k) = \frac{1}{2}\hat{v}(k)\hat{v}^*(k) \tag{6.6}$$

where a * denotes the complex conjugate. After the spectral energy density computation for each time step, a time average of this quantity was computed. The time-averaged wall-based energy spectrum with respect to the wall-based wavenumber is shown in Figure 6.2 for the spatial resolutions considered. The theoretical rate of the Burgers' equation is shown for comparison. The spectral energy density distribution revealed the range of resolved scales for each spatial resolution. Larger scales, associated with lower wavenumbers, exhibit higher energy content, while smaller scales, corresponding to higher wavenumbers, contain lower energy content. As anticipated, increasing the number of spatial elements allowed for the resolution of more turbulent scales. Moreover, the most energetic turbulent scales in the considered TCF were effectively resolved for the higher spatial resolutions examined. The rate of the inertial range computed was approximately higher than the theoretical value. This was due to forcing the Burgers' equation with a TCF dataset over all wavenumbers. However, it was hard to determine the wavenumbers corresponding to the inertial range due to the low value imposed for Re_τ.

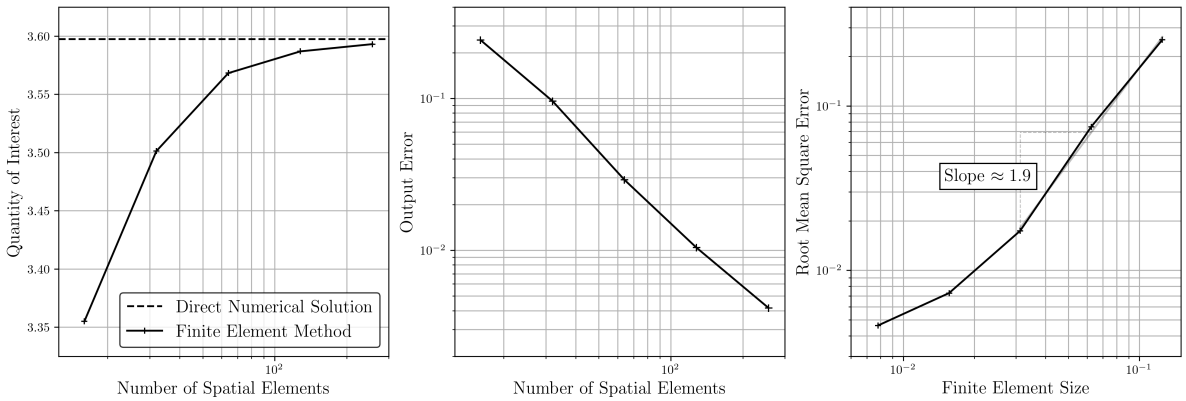


Figure 6.3: Quantity of Interest, Output Error and Root Mean Square Error of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

A time-averaged Quantity of Interest (QoI), denoted as $\bar{J}$, was selected for analysis and was defined as follows:

$$\bar{J} = \frac{1}{T}\int_I \int_\Omega e^{-50y^2} v^4(y,t)\, d\Omega dI \tag{6.7}$$

The chosen QoI serves as an indicator of the non-normalized kurtosis [91] near the channel centerline. A Gaussian distribution with zero mean and a standard deviation of $\sigma = 0.1$ was introduced to emphasize the center of the channel. The QoI was then normalized using the fourth-powered standard deviation evaluated with the DNS dataset and with the same Gaussian distribution employed in the QoI definition. Kurtosis, or the fourth standardized moment, reflects the heaviness of the tails in a probability distribution and primarily indicates the presence of outliers. For a Gaussian distribution, the kurtosis value is three, typically used as a reference for comparing other distributions [91]. The DNS-based value of the QoI $\overline{J}_{\mathsf{DNS}} = 3.597$ was computed from the DNS dataset and assumed to be the true QoI value. Therefore, the tails of the DNS wall-normal velocity probability distribution were heavier than a Gaussian distribution near the channel centerline.

A grid convergence study was conducted using the selected QoI and the Root Mean Square Error (RMSE) to evaluate the computed primal velocity $\boldsymbol{v}$ relative to the DNS dataset. The RMSE was calculated using the following expression:

$$\mathsf{RMSE}(\boldsymbol{v}) = \sqrt{\frac{1}{N_t N_y}\sum_{j=1}^{N_t}\sum_{i=1}^{N_y}\left[v(y_i,t_j) - v_{\mathsf{DNS}}(y_i,t_j)\right]^2} \tag{6.8}$$

where N_t represents the number of time steps. The computed time-averaged QoI as well as the output error $\delta\bar{J} = \bar{J}_{\mathsf{DNS}} - \bar{J}$ for the spatial resolutions considered and the RMSE of the FEM primal velocity for different element sizes are shown in Figure 6.3. For reference, the DNS value of the QoI is also shown for comparison. The computed QoI demonstrated a convergent behaviour towards the DNS value as the spatial grid was refined. For the spatial resolutions considered, the tails of the wall-normal velocity probability distribution were heavier than a Gaussian distribution. These tails became heavier as the spatial grid was refined. The output error displayed a monotonically decreasing trend with an increase in the number of spatial elements. As a result, the assumption that the output error was solely due to spatial discretization was valid. Additionally, the RMSE displayed a decreasing trend with a decreasing grid step size, reaching $O(10^{-2})$ for the higher spatial resolutions considered. The RMSE slope in the asymptotic region was consistent with the convergence rate $O(\Delta x^2)$ of linear elements for the L_2-norm, starting to plateau for higher spatial resolutions. The minimal deviation observed in the slope could be due to the DNS dataset interpolation and derivatives computation. The plateau observed could be due to the time interval step selected, for which the errors stemming from the temporal discretization were more significant than for coarser spatial grids.

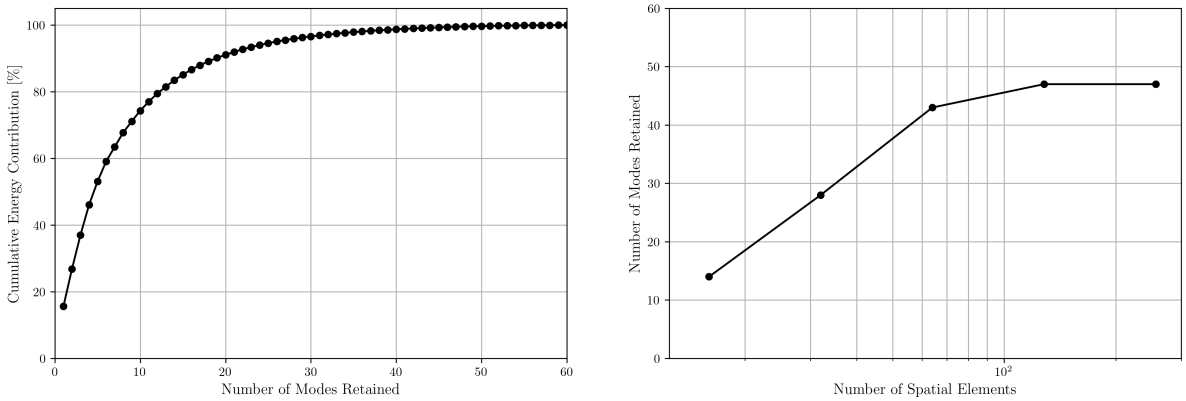


Figure 6.4: Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_y = 64$ Spatial Elements and Number of Modes Retained of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

6.2. Primal Solution Surrogate Models

This section explores the methodologies proposed to develop a surrogate model for the primal velocity to be used in the context of unsteady adjoint-based error estimation.

First, a study was conducted of the user-defined rank influence on the retained kinetic energy of the *offline* Proper Orthogonal Decomposition (POD). This was followed by the evaluation of the performance of the Convolutional AutoEncoder (CAE) during its training. The analysis compared training time and accuracy for the spatial resolutions considered. Additionally, a study was conducted to evaluate the influence of the Echo State Network (ESN) reservoir size in constructing a surrogate model of the primal velocity. After identifying the optimal reservoir size, the optimization process for tuning the ESN was evaluated in terms of computational efficiency and performance. Unlike the approach used for the manufactured solution, the reservoir study results obtained with the standalone ESN were also applied to the CAE-ESN methodology. The total computational time and performance were analyzed for the spatial resolutions considered. Finally, the methodologies' performance was quantitatively compared using the following evaluation metrics: Compression Ratio (CR), computational time and Mean Square Error (MSE) defined as follows:

$$\text{MSE}(\tilde{\boldsymbol{v}}) = \frac{1}{N_t N_y} \sum_{j=1}^{N_t} \sum_{i=1}^{N_y} \left[\tilde{v}(y_i, t_j) - v(y_i, t_j)\right]^2 \tag{6.9}$$

where $\tilde{\boldsymbol{v}}$ is the primal velocity obtained from each of the methodologies proposed. The evaluation parameters considered were fundamental to evaluate the efficiency of the surrogate models. The results of the methodologies proposed are shown in this section.

6.2.1. Proper Orthogonal Decomposition

The *offline* POD was first considered to construct a Reduced-Order Model (ROM) of the primal velocity as a baseline method. Singular Value Decomposition (SVD) was applied to factorize the snapshot matrix $\boldsymbol{U} \in \mathbb{R}^{N_y \times N_t}$. This matrix was assembled by the horizontal concatenation of the fluctuating components. To reconstruct the primal velocity, a 99% kinetic energy retention criterion was employed.

The cumulative energy contribution of the first sixty POD modes for $N_y = 64$ spatial elements and the optimal rank for each spatial resolution considered are shown in Figure 6.4. For $N_y = 64$ spatial elements, 43 POD modes were required to satisfy the established reconstruction criterion. Compared to the previous test case, a higher number of modes was necessary. This higher value of the optimal rank was attributed to the multiscale nature of turbulence, which involves a wide range of spatial and temporal scales from larger structures with high-energy content to smaller ones with lower energy. As the spatial grid was refined, the number of required POD modes increased, eventually reaching a plateau at higher spatial resolutions. Ultimately, this behaviour was attributed to the resolution of finer scales with grid refinement, after which the contribution to the overall energy is less significant.

Table 6.1: Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 64$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 24,611			Number of Trainable Parameters: 24,825		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$ ReLU Activation Function	$(8, 63)$	**Linear**	$(\text{In}, \text{Out}) = (43, 320)$ ReLU Activation Function	(320)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(16, 32)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(32, 9)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(32, 17)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(32, 17)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(32, 9)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(16, 32)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(64, 5)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(8, 63)$
Linear	$(\text{In}, \text{Out}) = (320, 43)$	(43)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$	$(1, 64)$

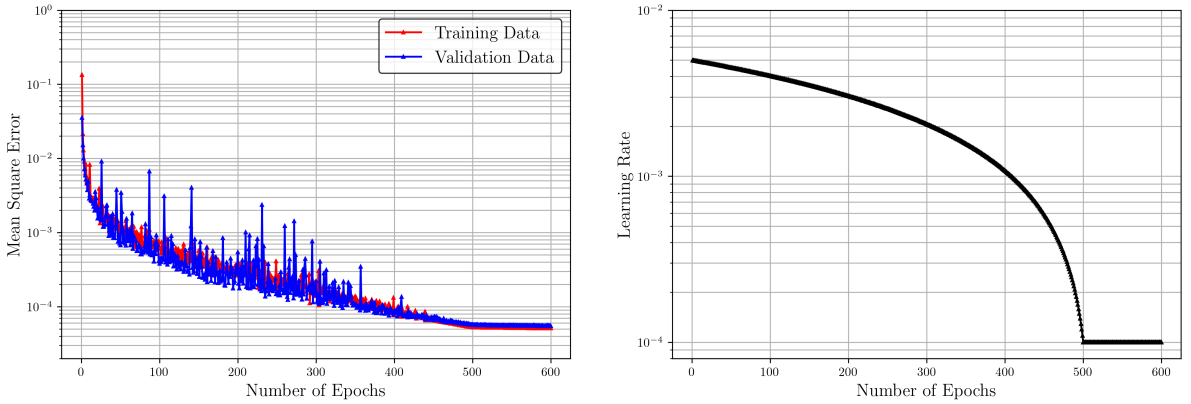


Figure 6.5: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements

6.2.2. Convolutional Autoencoder

The first deep learning method used to construct a surrogate model of the primal velocity was a CAE. The low-rank POD decomposition is optimal according to the Eckart-Young theorem [60]. Specifically, this theorem asserts that the rank-based approximation of a matrix derived from a truncated SVD minimizes the L_2-norm error. Consequently, the CAE latent space sizes were set to the optimal POD rank required to meet the 99% energy retention criterion.

A different CAE architecture was designed for each spatial resolution as the latent space dimension varied. The kernel filter sizes and the compressed spatial signal dimension after the use of the Convolutional Neural Networks (CNNs) were thus different for each spatial resolution. The decoder architecture mirrored the encoder architecture but with transpose convolution operators [78]. The CAE architecture used for a spatial resolution of $N_y = 64$ elements is given in Table 6.1.

Additional training data was generated by introducing Gaussian noise $\mathcal{N}(0, \sigma)$ with a standard deviation of $\sigma = 10^{-3}$. Compared to the previous test case, a higher value of the standard deviation was selected to enhance the robustness of the model. This augmentation resulted in a dataset containing 62,500 samples. A batch size of 75 samples was selected and the optimization procedure was performed over 600 epochs. For the first 500 epochs, the learning rate started at an initial value $\gamma_0 = 5 \times 10^{-3}$ and decreased with a decay factor $\gamma_r = 2 \times 10^{-2}$ based on trial and error. The learning rate was then held constant.

The evolution of the MSE loss for the training and validation sets along with the evolution of the adaptive learning rate is shown in Figure 6.5 for a spatial resolution of $N_y = 64$ elements. After the optimization process, the training and validation MSE stabilized at approximately $O(10^{-4})$. The use of an adaptive learning rate enabled the model to explore a wider region around potential local minima due to the

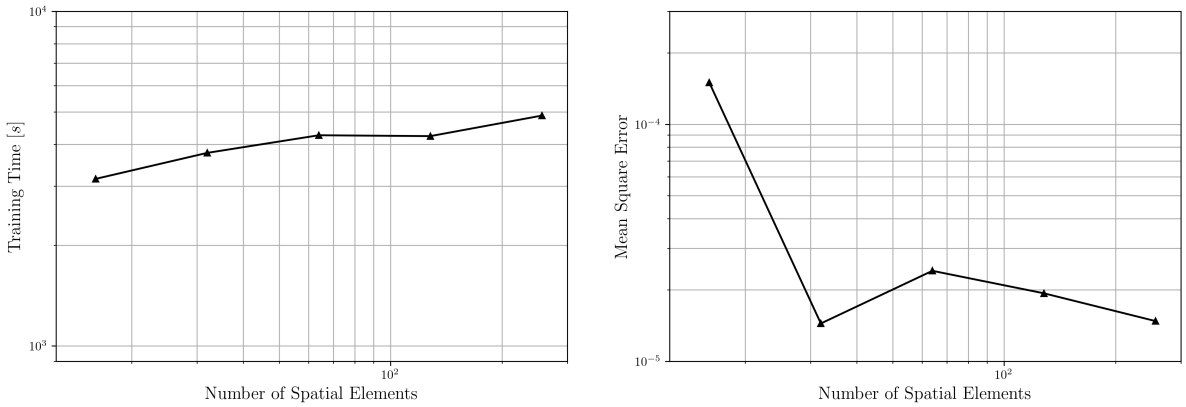


Figure 6.6: Training Time and Mean Square Error for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

observed MSE overshoots. At the end of the adaptive stage, the MSE convergence for the training set was smoother than in the previous test case but it could still be improved. Thus, a higher reduction of the loss function could be achieved by increasing the total number of epochs.

The training time and the optimal MSE are presented in Figure 6.6 for the spatial resolutions considered. Similar to the manufactured solution, the training time increased slightly with higher spatial resolutions. The MSE of the predicted primal velocity decreased for lower spatial resolutions and stabilized for finer spatial grids. As in the previous test case, the plateau was attributed to the insufficient total number of epochs selected during the adaptive learning rate stage.

Once training was complete, the CAE was employed to generate a surrogate primal velocity. The Dirichlet BCs were reapplied to ensure consistency with the physical constraints of the problem. The architectures and training performance for the remaining spatial resolutions can be found in section A.2.

6.2.3. Echo State Network

The second deep learning method investigated used an ESN to construct a surrogate primal velocity. The ESN contained a single hidden layer, known as the reservoir, which was pseudo-randomly generated and serves as a non-linear expansion of the input layer and as the memory of the system [79]. To address the inherent pseudo-randomness of the ESN, an ensemble of 10 samples was considered.

The training dataset included 50,000 samples, covering the complete primal velocity, while the validation set consisted of 19 overlapping intervals, each containing 5,000 samples. The validation fold was shifted by 2,500 time steps, which corresponded to approximately 1 Lyapunov Time (LT). For a TCF at $Re_\tau = 180$, the wall-based Highest Lyapunov Exponent (HLE) is given by $\text{HLE}^+ = 0.021$ [82]. The LT was computed as the inverse of the HLE. Furthermore, the washout period was set to 1,000 time steps. For the BO process, the hyperparameter space $\rho \times \sigma_{in} : [0.6, 1.2] \times [10^{-3}, 5.0]$ was explored for the spectral radius and the input scaling to minimize the averaged Prediction Horizon (PH) over the validation splits. The threshold $k = 1 \times 10^{-1}$ was selected in the definition of the PH. The optimal Tikhonov parameter β was chosen from the grid space $[10^{-3}, 10^{-6}, 10^{-9}]$.

The number of neurons in the reservoir, denoted as N_r, is a key hyperparameter that had to be selected and set constant for the spatial resolutions considered. Since the reservoir acts as a non-linear expansion of the inputs, a lower bound for its size was determined based on the highest spatial resolution considered. Thus, the values $N_r = \{512, 1024, 2048, 4096, 8192\}$ were investigated for a spatial resolution of $N_y = 64$. The Bayesian Optimization (BO) process was repeated for the samples considered. For $N_r = 512$ neurons, the mean and standard deviation of the Gaussian Process (GP) reconstruction are shown in Figure 6.7 along with the corresponding optimal hyperparameters ρ and σ_{in}. The averaged PH in the validation splits was measured in LTs and it was approximately $O(10^{-2})$ LTs for the samples considered. The GP reconstruction displayed significant sensitivity to the random seed selection, as evidenced by the substantial standard deviation observed in the locations of the optimal hyperparameters. As a result, repeating the BO process for different samples was needed. The optimal

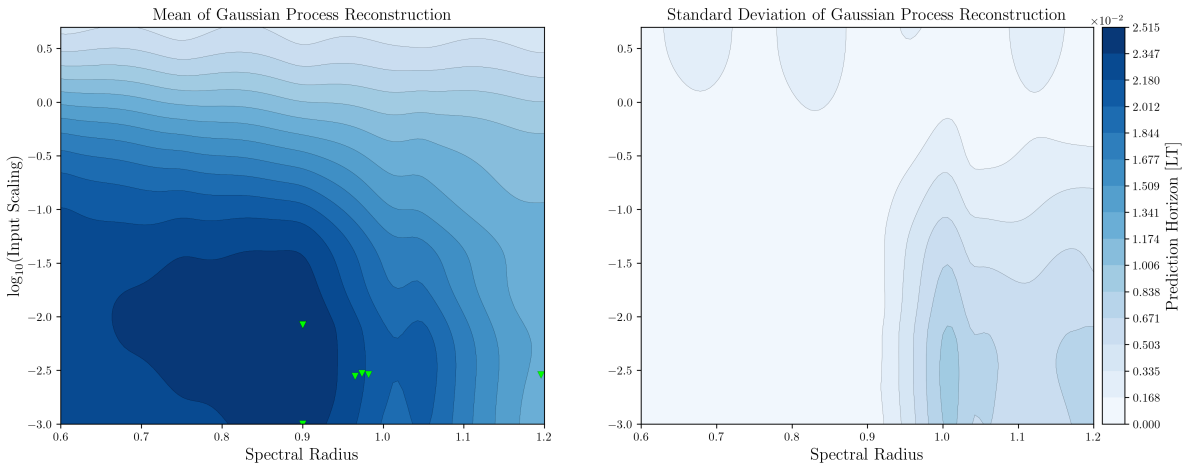


Figure 6.7: Averaged Prediction Horizon ($k = 2 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and $N_r = 512$ Reservoir Neurons

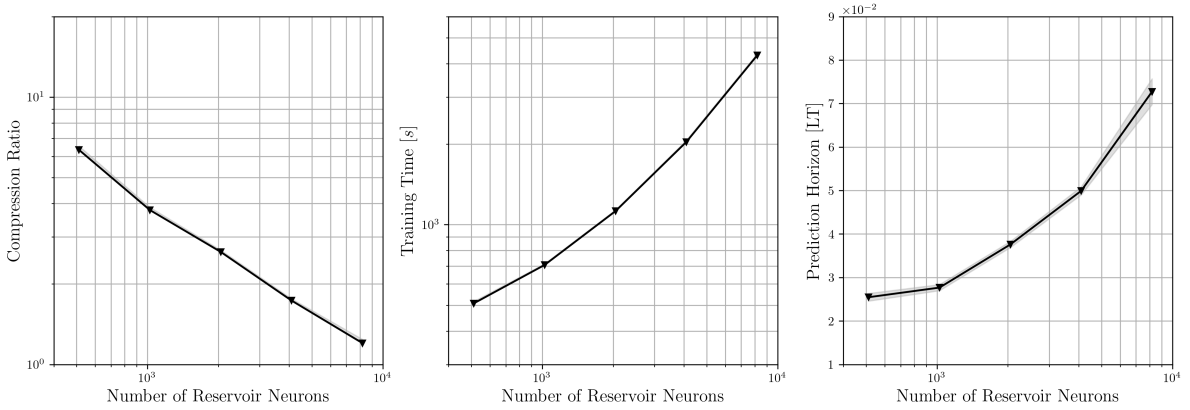


Figure 6.8: Compression Ratio, Training Time and Prediction Horizon for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and Different Number of Reservoir Neurons

ρ was within the range $[0.9, 1.0]$ while the optimal σ_{in} was in the range $[10^{-3}, 10^{-2}]$ for most samples.

The CR, training time and averaged PH measured in LTs for the 10 ESNs ensemble are shown in Figure 6.8 for a spatial resolution of $N_y = 64$ elements and the reservoir sizes considered. The CR decreased exponentially and approached unity with increasing reservoir size. This behaviour was not necessarily anticipated as the averaged PH also directly influenced the number of checkpoint parameters stored. The training time was higher than in the previous test case, primarily due to the availability of more training data. The averaged PH increased as the reservoir size increased, which indicated improved ESN performance. The accuracy of the surrogate velocity was constant across the reservoir sizes considered due to the established threshold k for the PH. Consequently, a reservoir of $N_r = 512$ neurons was selected as it provided the highest CR.

The user-defined threshold was reduced to $k = 1 \times 10^{-1}$ to increase the accuracy of the predicted velocity. However, this reduction also implied a decrease in CR. For $N_r = 512$ and the new threshold $k = 1 \times 10^{-1}$, the GP reconstruction is depicted in Figure 6.9 along with the optimal ρ and σ_{in}. As expected, a decrease of the averaged PH was observed, which remained approximately $O(10^{-2})$ LT for the different samples. This behaviour was attributed to the higher accuracy imposed in the ESN prediction. Also, most of the optimal hyperparameters converged to the same location $(\rho, \sigma_{in}) = (0.9, 10^{-2})$ of the search space explored. As a result, the need to repeat the BO process for different random seeds was significantly reduced.

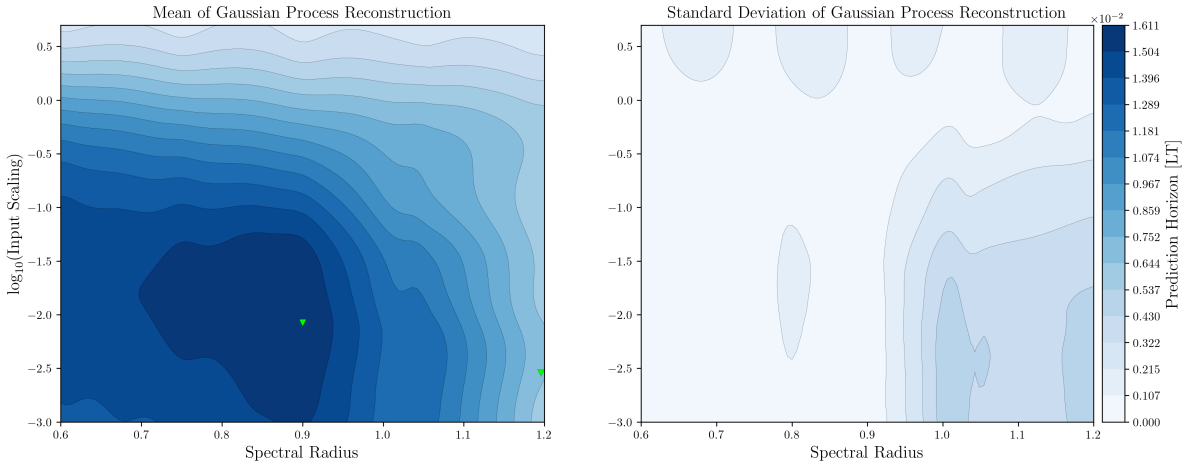


Figure 6.9: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and $N_r = 512$ Reservoir Neurons

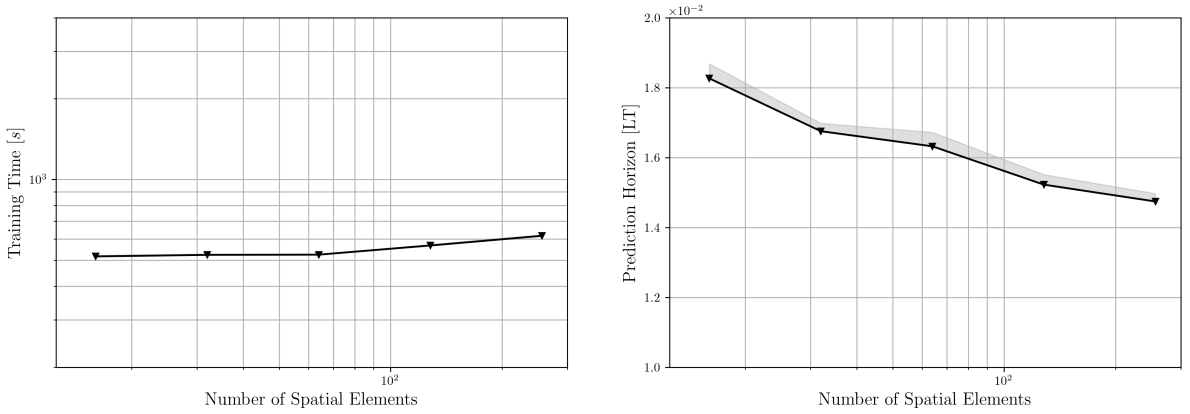


Figure 6.10: Training Time and Prediction Horizon for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

With the optimal reservoir size and threshold determined, the BO procedure was repeated for the spatial resolutions considered. The training time and averaged PH over the validation splits for the ensemble of 10 ESNs are presented in Figure 6.10 for each spatial resolution. The training time was nearly constant for the spatial resolutions considered, as the reservoir size was constant and primarily dictated the matrices' dimensions. However, the training time increased slightly for higher spatial resolutions as the number of spatial elements was of the same order as the reservoir size. The averaged PH decreased with increasing spatial resolution. This decrease was attributed to the increased complexity of the primal velocity for higher spatial resolutions as more turbulent scales were resolved.

The optimised ESN samples were then used to predict the primal velocity. A checkpoint technique was introduced, where the checkpoint length was set to match the averaged PH. As a final step, the Dirichlet BCs were reinforced. The BO results for the remaining spatial resolutions can be found in section B.2.

6.2.4. Convolutional Autoencoder - Echo State Network

The third deep learning method developed was a CAE-ESN to construct a surrogate primal velocity. The CAE was reused for this approach, but the ESN required a new training procedure. An ensemble of 10 ESN samples was used to account for the ESN inherent pseudo-randomness.

The same parameters used for the standalone ESN were applied for the BO procedure of the ESN applied to the latent space. The only exception was the search space explored for the spectral radius and input scaling which was set to $\rho \times \sigma_{in} : [0.4, 1.2] \times [10^{-3}, 5.0]$.

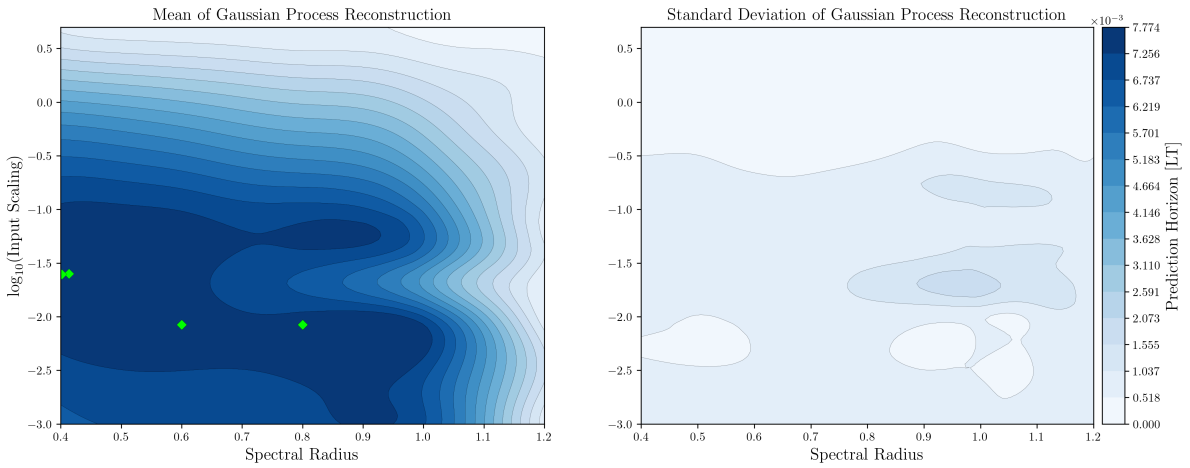


Figure 6.11: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyperparameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 64$ Spatial Elements and $N_r = 256$ Reservoir Neurons

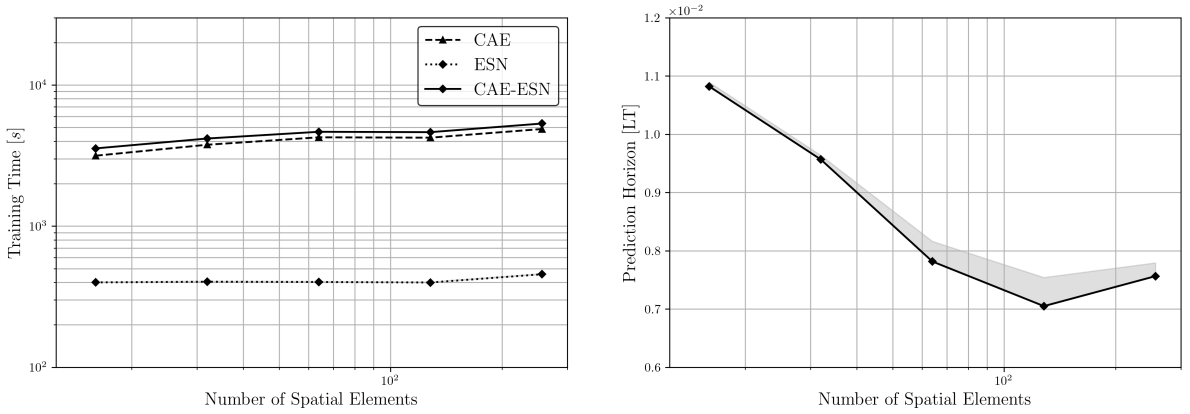


Figure 6.12: Computational Time and Prediction Horizon for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

In the reservoir study conducted for the ESN alone, the CR was maximized at the smallest viable reservoir size, as the accuracy remained constant due to the use of a constant threshold k. Thus, a reservoir size of $N_r = 256$ neurons was selected for the CAE-ESN, as it exceeded the latent space dimensions. For this reservoir size and $N_y = 64$ spatial elements, the mean and standard deviation of the GP reconstruction are presented in Figure 6.11 along with the optimal ρ and σ_{in}. The averaged PH decreased compared to the ESN alone but remained approximately $O(10^{-2})$ LTs for the samples considered. This decrease could be due to the increased latent space complexity, due to the introduction of non-linearities by the encoder, and the smaller reservoir size. Reduced sensitivity to the random seed choice was also observed as most hyperparameters converged to the same locations with decreased standard deviation. The optimal values for ρ were within the range $[0.4, 0.8]$ while the σ_{in} lied near 10^{-2}.

The latent space was then predicted for the remaining spatial resolutions considered. The training time and the averaged PH in LTs are illustrated in Figure 6.12 for the spatial resolutions considered. The training time included the CAE and the ESN training times. The ESN training time remained nearly constant as it was primarily dictated by the size of the reservoir. The integration of the ESN into the CAE framework had only a slight impact on the overall training time, as the CAE time remained the dominant factor. As observed with the ESN alone, the averaged PH decreased as the spatial grid was refined, due to the increased complexity of the high-dimensional space and consequently of the latent space. However, it reached a plateau for higher spatial resolutions.

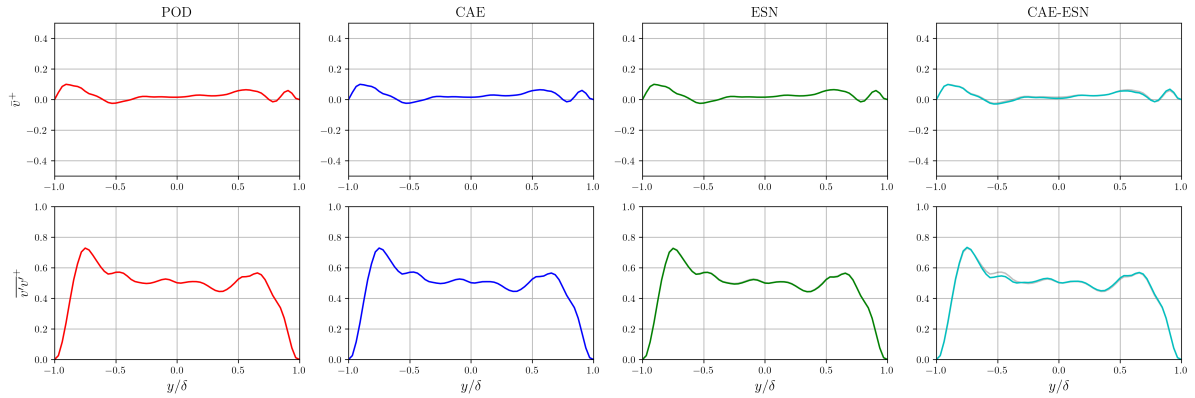


Figure 6.13: Mean Surrogate Primal Velocities and Reynolds Stresses of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements (Finite Element Method Results in Grey)

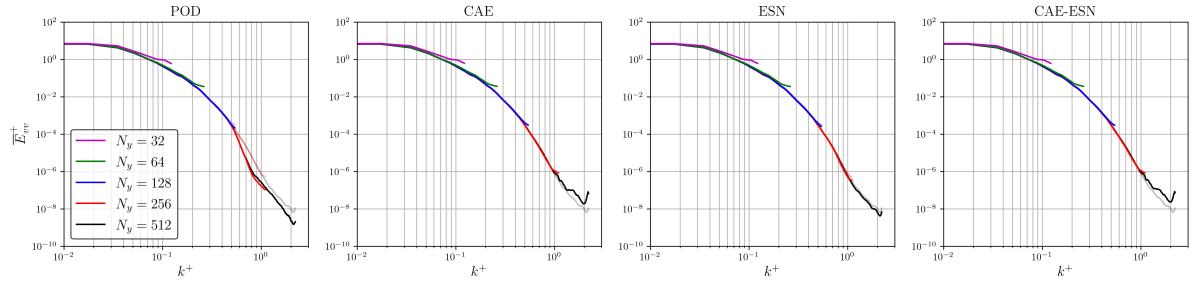


Figure 6.14: Mean Wall-Based Energy Spectrum for Surrogate Primal Solutions of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions (Finite Element Method Results Transparent)

The primal latent space was then predicted with the use of a checkpoint technique where the checkpoint length was set equal to the averaged PH. After the latent space prediction in time, the CAE's decoder was employed to reconstruct the high-dimensional data and the Dirichlet BCs were reapplied. Further details on the BO procedure of the ESN applied to the latent space for the spatial resolutions considered are found in section C.2.

6.2.5. Performance Assessment

The performance of the surrogate primal velocities is now evaluated. The Reynolds decomposition was applied to the surrogate wall-normal velocity. The mean component and Reynolds stresses produced with the surrogate velocities are shown in Figure 6.13 for $N_y = 64$ spatial elements along with the FEM results in grey for comparison. All methods captured the statistically-averaged component, as the surrogate results aligned with the FEM results. For the Reynolds stress, deviations were observed only for the CAE-ESN, where errors were observed at the quarter channel height. Despite this localized error, the overall shape of the Reynolds stress was preserved. Consequently, all surrogate models were able to accurately represent the mean and fluctuating components of the wall-normal velocity.

A FFT was applied to the surrogate fluctuating components. The time-averaged wall-based energy spectrum computed using the surrogate Fourier coefficients is shown in Figure 6.14 for the spatial resolutions considered with the FEM results displayed for comparison. The POD exhibited the lowest cut-off wave number, with its results deviating from the FEM prediction at higher wavenumbers. This deviation was observed as a sudden dip in a specific inertial wavenumber and was due to the snapshot matrix truncation, which resulted in the neglect of lower-energy scales. The CAE preserved the shape of the energy spectrum, with deviations observed at the highest wavenumbers only for the finest spatial resolution. The ESN demonstrated the closest agreement with the FEM results. The CAE-ESN inherited the limitations of the CAE at higher wavenumbers.

The evaluation parameters, including the CR, computational time and MSE, are shown in Figure 6.15 for the spatial resolutions considered. For the computational time, only the time required for the primal

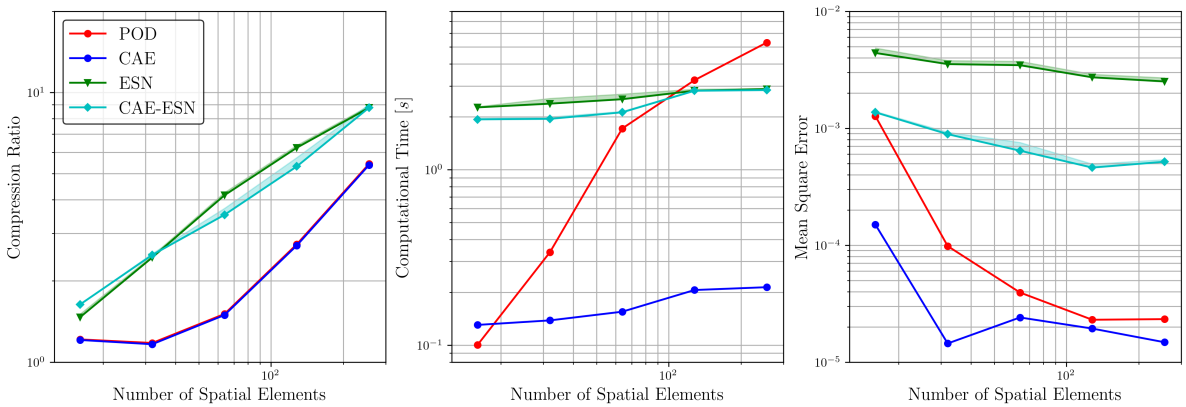


Figure 6.15: Evaluation Metrics of Surrogate Primal Solutions of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

velocity prediction was considered. In terms of CR, only the ESN and CAE-ESN exhibited a monotonically increasing trend as the number of spatial elements increased. For the lower spatial resolutions, the CR of the POD and CAE was constant due to the retention of a number of modes similar to the number of spatial elements. Regarding computational time, only the POD demonstrated a monotonically increasing trend, which was attributed to the application of SVD on larger snapshot matrices. The CAE achieved the lowest computational time for all spatial resolutions, except for the first where it was comparable to the POD. For higher spatial resolutions, the ESN-based methods outperformed the POD by exhibiting lower computational times. In terms of MSE, the CAE was the most accurate method for most spatial resolutions. The POD matched the CAE for finer spatial grids, presenting a similar MSE. Conversely, the ESN showed the highest MSE, due to significant errors in the predicted solution near the end of the checkpoints. The CAE-ESN approach also suffered from this limitation. Nevertheless, since the ESN operated on the latent space rather than the high-dimensional space, the MSE of the CAE-ESN was reduced compared to the standalone ESN. The reduced accuracy exhibited by the ESN-based methods was attributed to two primary factors: the absence of backward propagation in their methodology and the insufficient temporal information available to extract meaningful statistical data during the training process of the ESN.

Ultimately, the CAE was the best method to construct a surrogate model of the primal velocity for the spatial resolutions considered. This was attributed to its superior ability to preserve the energy spectrum and achieve the lowest MSE with a lower computational cost. However, the CAE-ESN approach remained an alternative, offering a higher CR at the cost of an increase in MSE. Despite this trade-off, the CAE-ESN approach successfully preserved the Reynolds decomposition components and the energy spectrum, making it a viable option for applications prioritizing compression efficiency.

6.3. Adjoint Problem Formulation and Solution

The adjoint problem of the unsteady 1D viscous Burgers' equation was derived in the previous test case for identical primal BCs and IC. In this test case, the only difference was the selected QoI which resulted in a different adjoint forcing term. As a result, the adjoint PDE is given by the following expression:

$$\begin{cases} -\dfrac{\partial \psi}{\partial t} - v\dfrac{\partial \psi}{\partial y} - \dfrac{1}{Re_\tau}\dfrac{\partial^2 \psi}{\partial y^2} = 4e^{-50y^2}v^3, & (y,t) \in \Omega \times I \\ \psi(y,t) = 0, & (y,t) \in \partial\Omega \times I \\ \psi(y,T) = 0, & y \in \Omega \end{cases} \tag{6.10}$$

The adjoint weak formulation must be derived. By considering weighting functions $q \in V_\psi$, the following weak formulation was obtained after integration by parts:

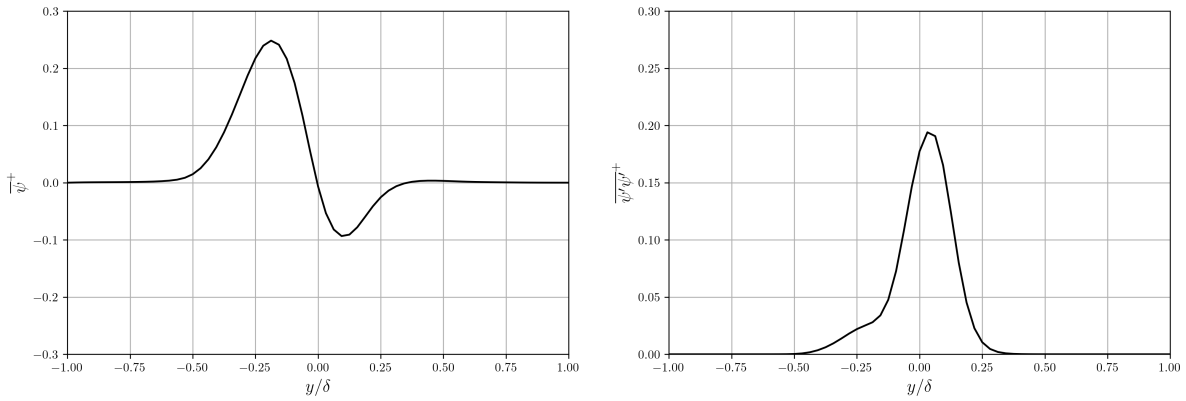


Figure 6.16: Mean Adjoint Velocity and Reynolds Stress of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements

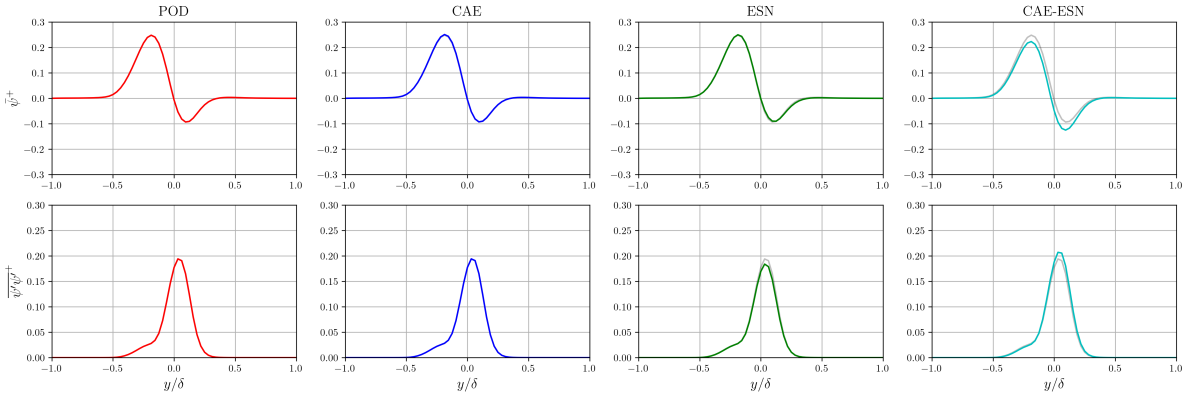


Figure 6.17: Surrogate Mean Adjoint Velocities and Reynolds Stresses of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements (Finite Element Method Results in Grey)

$$\int_\Omega \left(-\frac{\partial \psi}{\partial t} q - v \frac{\partial \psi}{\partial y} q + \frac{1}{Re_\tau} \frac{\partial \psi}{\partial y} \frac{\partial q}{\partial y} - 4e^{-50y^2} v^3 q \right) \, d\Omega = 0 \tag{6.11}$$

Continuous piece-wise linear elements were again selected for spatial discretization of the adjoint problem. The solving environment using `FEniCS` was employed as for the primal problem.

The Reynolds decomposition was used to decompose the adjoint velocity $\psi = \overline{\psi} + \psi'$ into a statistically-averaged component $\overline{\psi}$ and a fluctuating component ψ'. The latter component was evaluated in Reynolds stress form. The wall-based component $\overline{\psi}^+$ and adjoint Reynolds stress are shown in Figure 6.16 for $N_y = 64$ spatial elements. Both the mean adjoint velocity and Reynolds stress were concentrated around the channel's centerline. This behaviour was attributed to the Gaussian distribution present in the adjoint forcing term, which was also centered around the channel's centerline. Unlike the primal velocity, the adjoint statistically-averaged component was not approximately zero and the adjoint Reynolds stress magnitude was significantly lower. The former arises from the adjoint forcing term characteristics. The latter was attributed to the higher smoothness and stability of the adjoint velocity, which arose from the linearity of the adjoint problem. Due to the characteristics of the primal velocity, a larger temporal domain would lead to a more symmetrical distribution of the mean adjoint velocity and Reynolds stress about the channel centerline.

Adjoint velocities were computed with the surrogate primal velocities. The Reynolds decomposition was then applied to the surrogate adjoint velocities. The surrogate mean adjoint velocity and Reynolds stress are presented in Figure 6.17 for $N_y = 64$ spatial elements, with only one sample shown for the ESN and CAE-ESN. The FEM results are shown in grey. Only the CAE-ESN failed to capture the magnitude of the component $\overline{\psi}$, though it accurately represented its shape. In the case of the adjoint

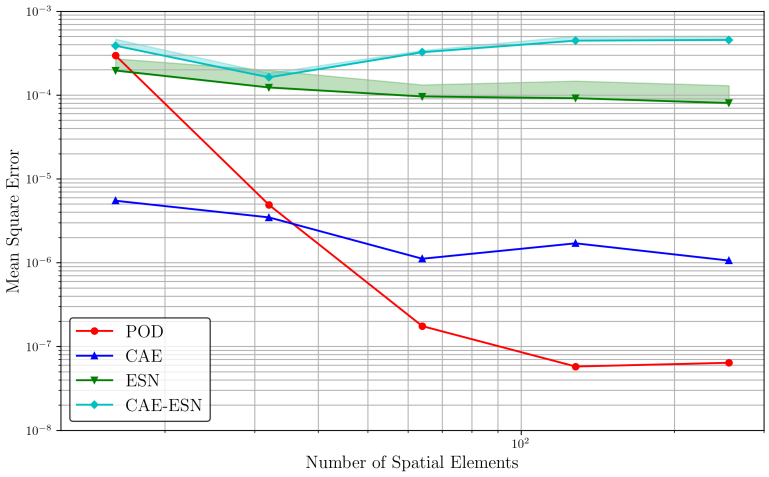


Figure 6.18: Mean Square Error of Surrogate Adjoint Velocities of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

Reynolds stress, the ESN and CAE-ESN failed to capture its magnitude, but they accurately captured the shape. These discrepancies were due to the adjoint problem's sensitivity to errors in the predicted primal velocity. Contrarily, the POD and CAE accurately capture the component $\overline{\psi}$ and Reynolds stress.

The MSE of the surrogate adjoint velocities relative to the FEM adjoint velocity is presented in Figure 6.18 for the spatial resolutions considered. All methods demonstrated a decreasing trend in error with an increasing number of spatial elements, except for the CAE-ESN. The POD showed the steepest error rate while reaching a plateau at the highest spatial resolutions. The CAE and ESN steep rates were minimal. At lower spatial resolutions, the CAE demonstrated superior performance, achieving a MSE of $O(10^{-5})$. At higher spatial resolutions, the POD was the best method with a MSE of $O(10^{-7})$. While the CAE-ESN achieved a lower MSE than the standalone ESN for the surrogate primal velocity, the ESN proved to be more accurate in reconstructing the adjoint velocity. Unlike the behaviour observed with the manufactured solution, the performance of these methods for the adjoint velocity did not align with their accuracy in building a surrogate primal velocity. This discrepancy arose from the adjoint forcing term imposed, which was dependent on the cubed primal velocity.

6.4. Adjoint-Based Error Estimation

Adjoint-based error estimation requires the primal and adjoint solutions in coarse and fine spaces. The coarse-space primal solution defined in the coarse function space V_H was linearly interpolated into the fine function space V_h. Subsequently, the strong fine-space residual was evaluated using the injected primal solution. As in the previous test case, the Dirichlet BCs were enforced on the injected primal solution and the necessary derivatives to evaluate the residual were projected into quadratic function spaces.

The temporal mean and standard deviation of the fine-space residual calculated using the injected primal solution are shown in Figure 6.19 for the coarse-space resolutions considered. Increasing the number of spatial elements resulted in a reduction in the injected residual magnitude. It also caused a decrease in the standard deviation along the channel's centerline while exhibiting higher values near the walls. This behaviour arose from the need for higher refinement near the walls to accurately capture large velocity gradients.

The temporal mean and standard deviation of the surrogate injected residuals are presented in Figure 6.20 for $N_y = 64$ spatial elements, with one sample shown for the ESN and CAE-ESN. The FEM results are shown in grey. Both the POD and CAE effectively captured the temporal mean of the injected residual. Regarding the standard deviation, the POD resembled the FEM result, with identical values near the wall. While the CAE reproduced the shape of the standard deviation of the injected residual, it struggled with its magnitude. On the contrary, the ESN-based methods failed to capture the injected residual altogether. Once again, this was attributed to errors in the predicted primal velocity.

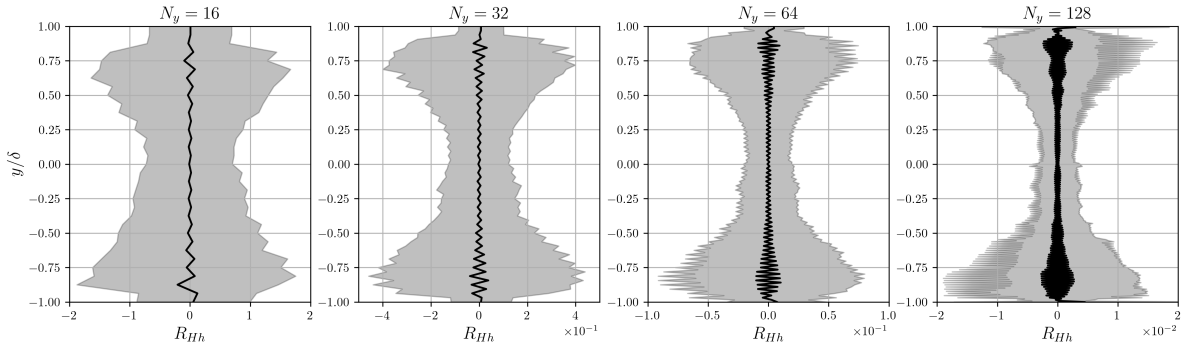


Figure 6.19: Primal Injected Residual Temporal Mean and Standard Deviation of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

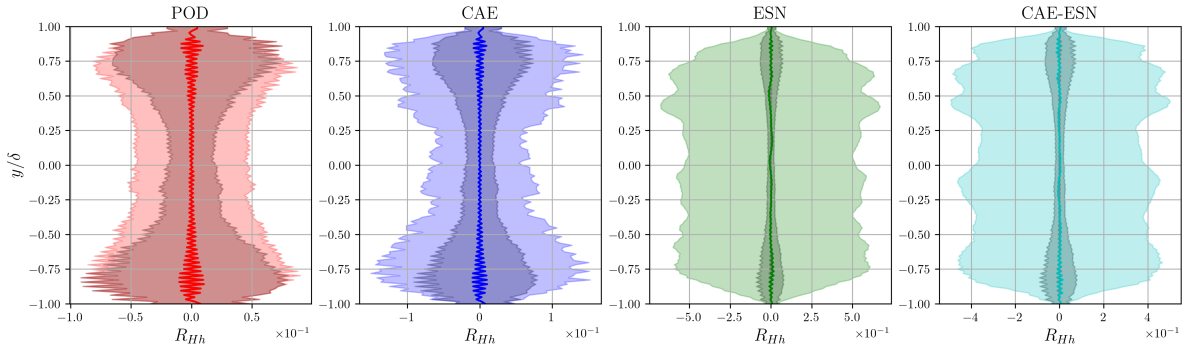


Figure 6.20: Primal Injected Surrogate Residual Temporal Mean and Standard Deviation of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions (Finite Element Method Results in Grey)

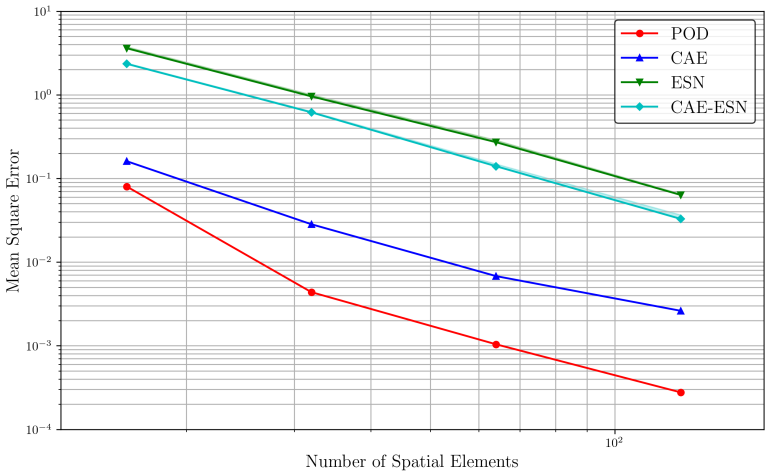


Figure 6.21: Mean Square Error of Primal Injected Surrogate Residuals of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

The MSE of the surrogate injected residuals is shown in Figure 6.21 for the spatial resolutions considered. All the methods demonstrated a monotonically decrease in MSE as the number of spatial elements increased. The POD was the best method across the spatial resolutions considered. The injected residual produced by the CAE had a similar accuracy to POD for the lowest spatial resolution. In contrast, the ESN-based methods were unable to generate an appropriate injected residual for the spatial resolutions considered.

Following the computation of the refined adjoint velocity and fine-space primal injected residual, the

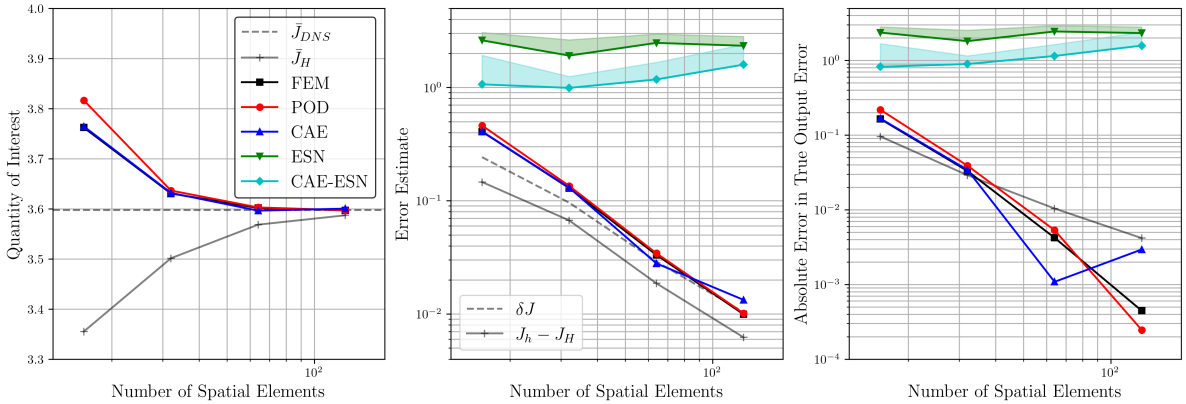


Figure 6.22: Quantity of Interest and Error Estimates of Surrogate Solutions of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

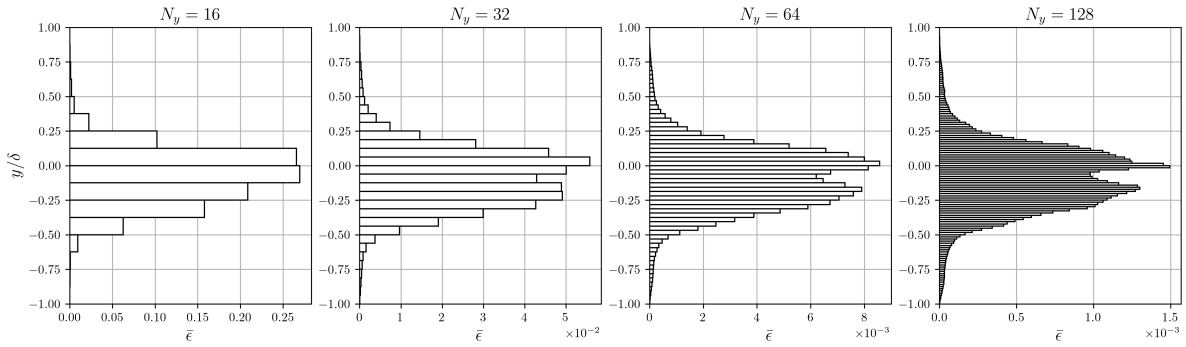


Figure 6.23: Time-Averaged Error Indicators of Unsteady 1D Viscous Burgers' Equation for Different Spatial Resolutions

time-averaged adjoint-based error estimates $\delta\bar{J}_{est}$ were calculated for the FEM and surrogate velocities. The time-averaged adjoint-corrected QoIs $\bar{J} + \delta\bar{J}_{est}$, the adjoint-based error estimates $\delta\bar{J}_{est}$ and the absolute error in true output error $\left|\delta\bar{J}_{est} - \delta\bar{J}\right|$ are presented in Figure 6.22 for the methods proposed and spatial resolutions considered. The FEM adjoint-corrected QoI was overestimated for the spatial resolutions considered, approaching the true QoI value as the spatial grid was refined. This was attributed to the linearization error $O(\delta\boldsymbol{u}^2)$ present in non-linear problems [26]. As the selected QoI involved the fourth-powered primal velocity, it had a higher sensibility to outliers, especially for coarser spatial grids. Only the POD and CAE provided accurate adjoint-corrected QoI predictions. At lower spatial resolutions, the CAE emerged as the best method, with its QoI aligning closely with the FEM reference. For higher spatial resolutions, the POD and CAE had a similar performance in predicting the adjoint-corrected QoI. A similar trend was observed in the error estimates, where the POD demonstrated a slight advantage over the CAE at higher spatial resolutions. Consequently, only the POD and CAE were suitable for adjoint-based error estimation, as they provided reliable approximations. The ESN and CAE-ESN were unsuitable due to their inability to produce accurate injected residuals.

Local error indicators were computed which could be used to to evaluate their suitability for mesh adaptation. Time-averaged local error indicators were considered as the errors introduced by the temporal discretization scheme were negligible. The time-averaged error indicator field is shown in Figure 6.23 for the spatial resolutions considered. The time-averaged error indicator fields were concentrated along the channel's centerline, reflecting the characteristics of the adjoint solution. At lower spatial resolutions, the error indicator field was primarily shaped by the Gaussian distribution present in the adjoint forcing term. As the spatial grid was refined, a gap emerged at $y/\delta \approx 0.5$. This phenomenon arose from the properties of the adjoint solution, as this region corresponded to where the mean adjoint velocity was zero.

The surrogate time-averaged error indicators are presented in Figure 6.24 for $N_y = 64$ spatial elements.

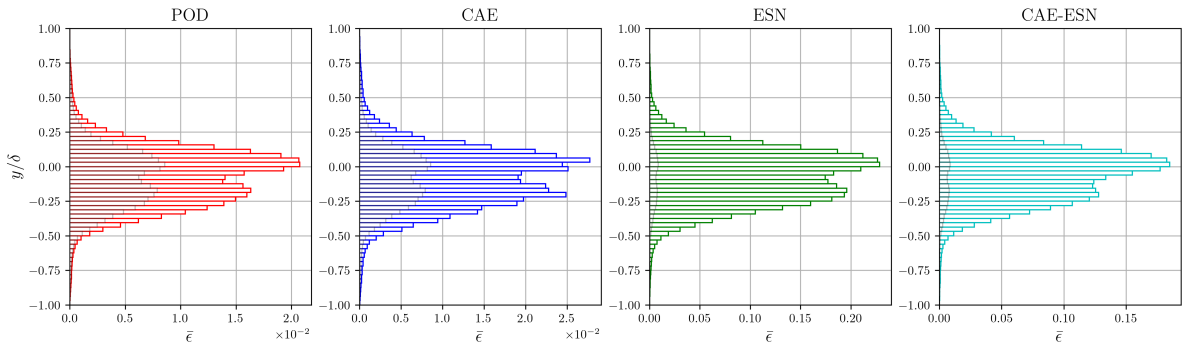


Figure 6.24: Time-Averaged Error Indicators of Surrogate Solutions of Unsteady 1D Viscous Burgers' Equation with $N_y = 64$ Spatial Elements (Finite Element Method Results in Grey)

For comparison, the FEM error indicator is also shown in grey. Only the POD and CAE achieved a magnitude comparable to the FEM results. Consequently, the POD and CAE were the most suitable approaches. In the ESN-based methods, despite failing to capture the magnitude, the shape resembled the FEM result. While these methods accurately captured the adjoint solution, they failed to predict the injected residual. As it was concluded for the manufactured solution, the shape was preserved since the ESN and CAE-ESN accurately capture the adjoint velocity.

Only the POD and CAE were capable of accurately computing the adjoint-based error estimates. Adaptive Mesh Refinement (AMR) is particularly relevant for coarser spatial meshes, where accurately representing the physics of the problem is difficult but much of the computational cost can be saved. In this context, the CAE emerged as the preferred method due to its efficiency. For local error indicators, while only the POD and CAE accurately captured the magnitude, all methods successfully reproduced the overall shape. The ESN-based methods provided an acceptable shape of the error indicator distribution due to their accurate prediction of the adjoint velocity. However, for these methods to be effectively applied in adjoint-based error estimation, a different strategy would be required to improve the injected residual prediction. The prediction of the ESN-based methods of the adjoint-based error estimate and error indicators was dominated by the error in the predicted injected residual. Ultimately, the CAE was the best method for lower spatial resolutions in the context of adjoint-based error estimation, especially for reducing storage requirements. For higher spatial resolutions, the POD was the most suitable method, as it offered greater accuracy with equal compression capabilities to the CAE.

7

Results: Unsteady 2D Viscous Burgers' Equation

In this chapter, the test case is the unsteady two-dimensional (2D) viscous Burgers' equations, with a solution forced by the same Turbulent Channel Flow (TCF) dataset to produce the spanwise and wall-normal velocity components.

Firstly, the primal Partial Differential Equation (PDE) is introduced along with the primal velocity components obtained using the Finite Element Method (FEM), implemented with the `FEniCS`[1] package. Surrogate primal solutions were then computed with the methodologies proposed and their performance was evaluated for different evaluation parameters. Next, the adjoint governing equations were derived, followed by the presentation of the adjoint velocity components. Finally, error estimates were calculated from the solutions generated by the implemented solver and the methodologies proposed. Local error indicator fields were then calculated which could be used for mesh adaptation.

7.1. Primal Problem Formulation and Solution

A Direct Numerical Simulation (DNS) dataset of a TCF at a friction Reynolds number $Re_\tau = 180$ was used to force the unsteady 2D viscous Burgers' equations, producing the spanwise and wall-normal velocity components. The same DNS dataset as in the previous test case was used. The DNS dataset was evaluated at a specific streamwise coordinate x to force the 2D Burgers' equations. Consequently, the 2D Burgers' equations were equivalent to the spanwise and wall-normal momentum equations at the specified location. The DNS dataset was down-sampled to a constant time step of $\Delta t = 1 \times 10^{-4}$ and a uniform spatial resolution of $N_z \times N_y = 512 \times 256$ quadrilateral elements.

For the 2D Burgers' equation, a spatial rectangular domain $\Omega : [0, L_z] \times [-\delta, \delta]$ was considered in the spanwise and wall-normal directions with $L_z = \pi$ and $\delta = 1.0$. The temporal domain $I : [0, T]$ was considered with $T = 1.0$ and it was relatively limited due to the higher dimensionality of the test case. Periodic and Dirichlet Boundary Conditions (BCs) were applied in the spanwise and wall-normal directions respectively to ensure consistency with the DNS velocity conditions. The Initial Condition (IC) was set to the DNS initial velocity field. Thus, the following primal PDE was formulated:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \dfrac{1}{Re_\tau} \nabla^2 \boldsymbol{u} = \boldsymbol{f}_{\text{DNS}}, & (\boldsymbol{x}, t) \in \Omega \times I \\ \boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{0}, & (\boldsymbol{x}, t) \in \partial\Omega_{\text{Wall}} \times I \\ \boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_{\text{DNS}}(\boldsymbol{x}, 0), & \boldsymbol{x} \in \Omega \end{cases} \tag{7.1}$$

where $\boldsymbol{x} = (z, y)$ and t are the space and time coordinates, $\boldsymbol{u} = (w, v)$ is the primal velocity with spanwise and wall-normal components and $\boldsymbol{f}_{\text{DNS}} = (f_z, f_y)$ is the DNS forcing term computed from the

[1] `https://fenicsproject.org`

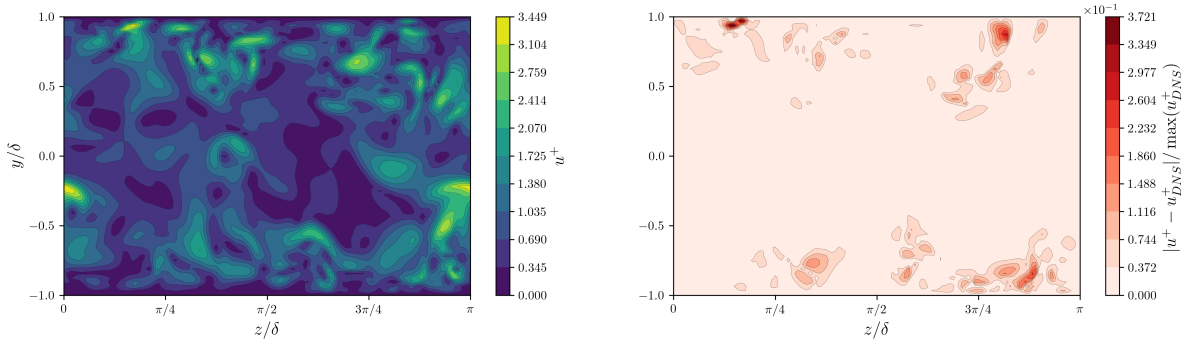


Figure 7.1: Primal Velocity Magnitude and Relative Error Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

DNS dataset. The term $\boldsymbol{f}_{\text{DNS}} = (f_z, f_y)$ was defined as follows:

$$\boldsymbol{f}_{\text{DNS}} = \left[-u\frac{\partial \boldsymbol{u}}{\partial x} - \frac{1}{\rho}\nabla p + \frac{1}{Re_\tau}\frac{\partial^2 \boldsymbol{u}}{\partial x^2} \right]_{\text{DNS}} \tag{7.2}$$

where u is the streamwise velocity component, ρ is the density and p is the pressure. The derivatives required were calculated using second-order finite difference schemes, except at the boundaries where first-order schemes were employed. For simplicity, the variable u is now the primal velocity magnitude.

The derivation of the primal weak formulation was needed as the FEM was employed for spatial discretization of the primal problem. Given weighting functions $\boldsymbol{q} \in V_{\boldsymbol{u}}$, where the vector function space $V_{\boldsymbol{u}}(\Omega)$ was equal for the primal and weighting functions, the resulting weak formulation was obtained after integration by parts:

$$\int_\Omega \left(\frac{\partial \boldsymbol{u}}{\partial t} \odot \boldsymbol{q} + (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \odot \boldsymbol{q} + \frac{1}{Re_\tau} (\nabla \boldsymbol{u} \odot \nabla \boldsymbol{q}) - \boldsymbol{f}_{\text{DNS}} \odot \boldsymbol{q} \right) d\Omega = 0 \tag{7.3}$$

where $\odot$ is the element-wise product and the Dirichlet BCs at the wall were applied strongly. Continuous piece-wise linear triangular elements were chosen as the finite elements for spatial discretization. In these elements, the first derivatives are discontinuous while the second derivatives are zero. Integration by parts eliminated the Laplacian operator in the weak formulation, thus the use of the elements selected was suitable. The non-linear variational problem was solved using the `FEniCS` environment employed previously but adapted for a 2D configuration. A time step interval $\Delta t = 1 \times 10^{-4}$ was selected to match the one from the DNS dataset and it ensured stability without the need for additional stabilization techniques. Also, the selected value for Δt minimized errors from the temporal discretization scheme. Therefore, it was assumed that the output error was only due to spatial discretization.

The spatial resolutions $N_z = \{32, 64, 128, 256, 512\}$ with $N_y = N_z/2$ were examined with particular emphasis on $N_z \times N_y = 128 \times 64$ spatial elements. The triangular grids generated were uniform, with the diagonals of the triangles oriented in the right direction. However, the spatial resolutions mentioned were defined in terms of quadrilateral elements. A snapshot of the wall-based velocity magnitude u^+ computed along with its relative error with respect to the DNS velocity is shown in Figure 7.1 for $N_z \times N_y = 128 \times 64$ spatial elements and a time step $t = T$. The relative error was defined as the absolute error normalized by the maximum value of the true solution. As expected, the primal velocity magnitude exhibited chaotic behaviour due to the turbulent nature of the flow. Also, the imposed BCs were effectively observed. The relative error was approximately $O(10^{-1})$ and it was concentrated near the walls. This distribution was anticipated, as higher spatial refinement was required near the walls to accurately capture steep velocity gradients.

Mathematically, the vorticity is defined as the curl of the velocity field:

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{u} \tag{7.4}$$

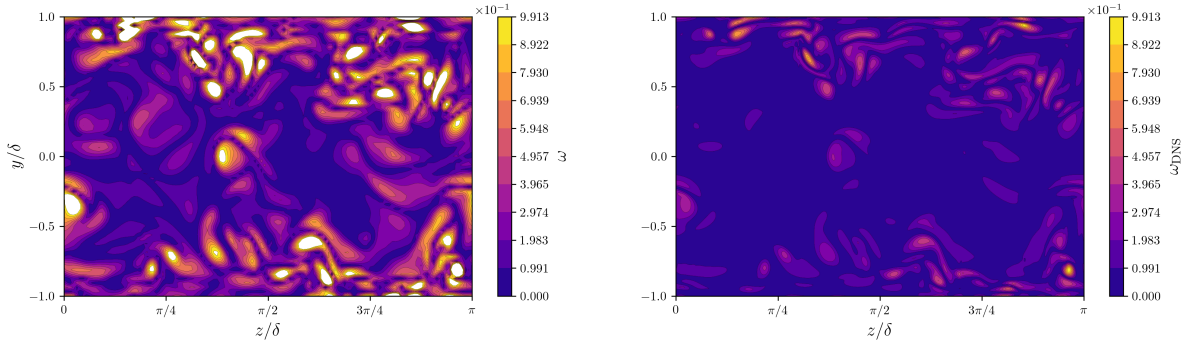


Figure 7.2: Vorticity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

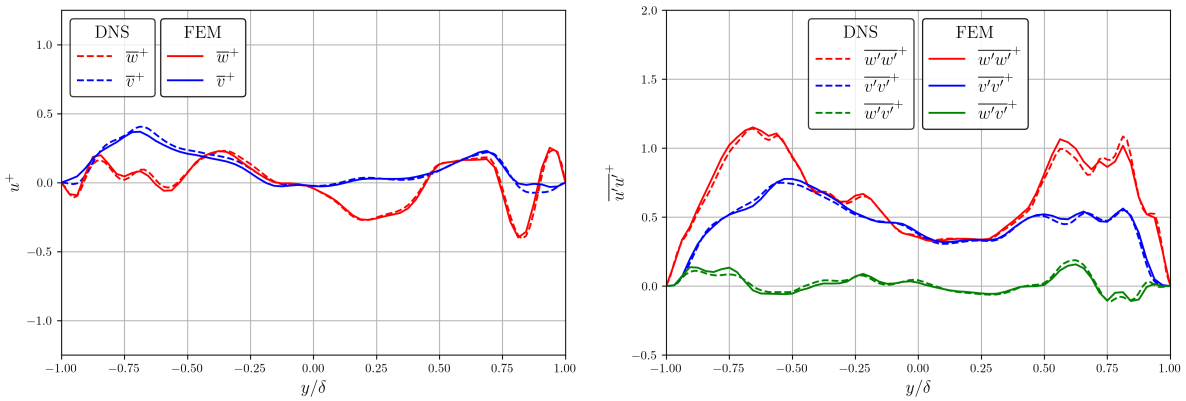


Figure 7.3: Mean Primal Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

The vorticity field serves as a tool to identify vortex structures within the flowfield. More advanced techniques, such as the Q-criterion [92], were not applied due to the 2D nature of the test case. Consequently, only the streamwise vorticity component was considered. The required velocity derivatives were computed using second-order central difference schemes. A snapshot of the vorticity magnitude ω for $N_z \times N_y = 128 \times 64$ spatial elements and a time step $t = T$ is presented in Figure 7.2, with the DNS results included for comparison. The magnitude of the vorticity computed was higher than the one presented by the DNS dataset. This discrepancy arose from the coarser spatial grid used in the solver. The underresolution of small-scale velocity fluctuations resulted in larger truncation errors in the numerical computation of the derivatives. However, the shape of the FEM vorticity field still resembled the DNS results. Vortex structures were more prominent near the walls, which was also attributed to larger velocity gradients.

The Reynolds decomposition was applied to the velocity profiles $u_j = \overline{u}_j + u'_j$, separating them into statistically-averaged components $\overline{u}_j$ and fluctuating components u'_j at a spanwise coordinate $z = \pi/2$. The Reynolds stress form was used to assess the fluctuating components. The mean velocity components and the Reynolds stresses computed are presented in Figure 7.3 for $N_z \times N_y = 128 \times 64$ spatial elements. A superscript $+$ indicates quantities normalized by viscous scales and DNS results are included for comparison. As derived in the previous test case, $\overline{w}$ and $\overline{v}$ are zero in a TCF. However, due to the time interval considered, the plotted averages did not correspond to long-time statistics. The Reynolds stresses in the spanwise and wall-normal directions exhibited a peak at a distance of $0.25 < y/\delta < 0.50$ from the walls, corresponding to a wall-based distance of $45 < y^+ < 90$. This location was within the log-law region and was expected for the imposed Re_τ value [89]. Also, it was expected for the spanwise component of the Reynolds stress to be more significant than the wall-normal one [89]. The results from the implemented solver and DNS dataset were similar in terms of mean velocities and Reynolds stresses. Extending the temporal domain would lead to zero mean velocities and a more

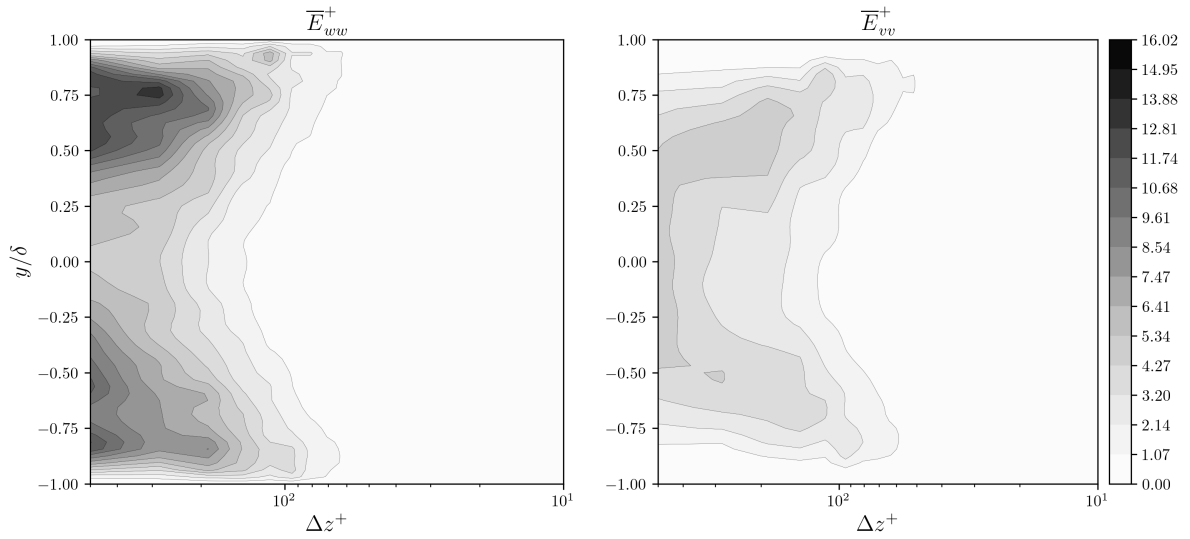


Figure 7.4: Mean Wall-Based 1D Energy Spectra of the Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

symmetric distribution of the Reynolds stresses about the channel centerline.

The energy spectra provide insight into the range of turbulent scales resolved and the energy content associated with each velocity component [89]. A Fourier representation of the fluctuating components was required to compute the energy spectra. Given the periodicity in the $z-$direction, a 1D Fast Fourier Transform (FFT) was applied at each time step and wall-normal coordinate:

$$u'_j(z) = \sum_k \hat{u}_j(k) e^{ikz} \tag{7.5}$$

where k are the wavenumbers in the $z-$direction, $\hat{u}_j$ are the complex Fourier coefficients for each j velocity component and i is the imaginary unit such that $i^2 = -1$. The spectral energy density in the spanwise and wall-normal components, denoted as E_{ww} and E_{vv}, was computed using the resulting coefficients $\hat{u}_j$. For a spatial resolution of $N_z \times N_y = 128 \times 64$ elements, the time-averaged wall-based 1D energy spectra with respect to the wall-based spanwise wavelength are shown in Figure 7.4. The wavelength is defined as the inverse of the wavenumber with a 2π factor. The spanwise component of the energy spectrum was more energetic than the wall-normal component, with higher values observed near the wall. These observations aligned with the observations made for the Reynolds stresses. The higher energy density observed near the wall was associated with near-wall coherent structures, such as ejections, sweeps and bursts of streaks [93]. Low-speed streaks tend to raise where ejection occurs, accompanied by a sweep of high-speed streaks towards the wall. After their ejection, there is a burst of the streaks into finer scales. Additionally, the range of resolved turbulent scales was wider near the walls, which was consistent with the presence of small-scale turbulent structures.

A time-averaged Quantity of Interest (QoI), denoted by $\bar{J}$, was selected and it was defined as follows:

$$\bar{J} = \frac{1}{T} \int_I \int_\Omega \left(e^{-50(y-\delta/2)^2} + e^{-50(y+\delta/2)^2} \right) \boldsymbol{u} \cdot \boldsymbol{u} \, d\Omega dI \tag{7.6}$$

The selected QoI was associated with the Turbulent Kinetic Energy (TKE) [13] near the log-law regions, calculated using only the spanwise and wall-normal velocity components. Gaussian distributions with zero mean and a standard deviation of $\sigma = 0.1$ were applied at a quarter channel distance from the walls to highlight the log-law regions. The DNS-derived value of the QoI $\overline{J}_{DNS} = 2.343$ was computed from the DNS dataset and considered the true QoI value.

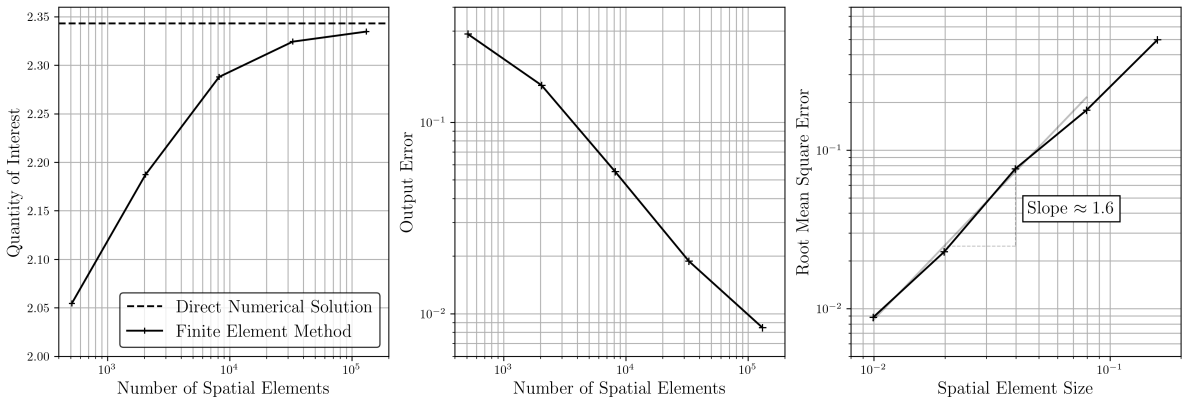


Figure 7.5: Quantity of Interest, Output Error and Root Mean Square Error of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

A grid convergence study was performed to assess the selected QoI and the accuracy of the computed primal velocity with respect to the DNS velocity. The accuracy was based on the Root Mean Square Error (RMSE), which was computed using the following expression:

$$\mathsf{RMSE}(\boldsymbol{u}) = \sqrt{\frac{1}{2N_t N_y N_z} \sum_{l=1}^{2} \sum_{k=1}^{N_t} \sum_{j=1}^{N_y} \sum_{i=1}^{N_z} \left(\boldsymbol{u}_l(\boldsymbol{x}_{ij}, t_k) - \boldsymbol{u}_{l,\mathsf{DNS}}(\boldsymbol{x}_{ij}, t_k)\right)^2} \tag{7.7}$$

where N_t is the number of time steps. The computed time-averaged QoI and the output error $\delta\bar{J} = \bar{J}_{\mathsf{DNS}} - \bar{J}$ for the spatial resolutions considered along with the RMSE of the FEM primal velocity for different element sizes are shown in Figure 7.5. The DNS value of the QoI is included for comparison. The computed QoI exhibited a convergent behaviour towards the DNS value as the spatial grid was refined. The output error showed a monotonically decreasing trend with an increase in the number of spatial elements. As a result, the assumption that the output error was solely due to spatial discretization was valid. The RMSE showed a decreasing trend with smaller grid step sizes, achieving $O(10^{-2})$ for the higher spatial resolutions considered. The RMSE slope in the asymptotic region was reasonably consistent with the theoretical convergence rate of $O(\Delta x^2)$ for linear triangular elements in the L_2-norm. The deviation in the observed slope was attributed to the DNS data interpolation and derivatives computation. For the remainder of this investigation, the spatial resolution of $N_z \times N_y = 512 \times 256$ elements was excluded due to the prohibitive computational cost involved in assessing the adjoint-based error estimate at this level of refinement.

7.2. Primal Solution Surrogate Models

This section explores the methodologies proposed to develop a surrogate model of the primal velocity to be used in the context of unsteady adjoint-based error estimation. The methods considered included the benchmark Proper Orthogonal Decomposition (POD) and the Convolutional Autoencoder (CAE). The Echo State Network (ESN) was not suitable due to the high dimensionality of the test case, as its training would be prohibitively time-intensive. The CAE-ESN was proposed as it circumvents the challenges associated with training ESNs for higher-dimensional systems [39]. However, the ESN-based methods yielded sub-optimal results in the previous test case. Furthermore, the limited time interval considered in this test case failed to provide sufficient temporal data for training. As a result, the CAE-ESN was also not considered. In the case the CAE-ESN was applied, it was expected that the accuracy of the surrogate primal solution predicted to be bounded by the accuracy of the CAE with higher compression capabilities. It was also expected for the predicted adjoint-based error estimate to be inaccurate as it was observed in the previous test case.

First, two *offline* POD techniques were explored to address the high-dimensionality of the test case. This was followed by an evaluation of the performance of the CAE during its training. The analysis focused on training times and accuracy for the spatial resolutions considered. Finally, the performance

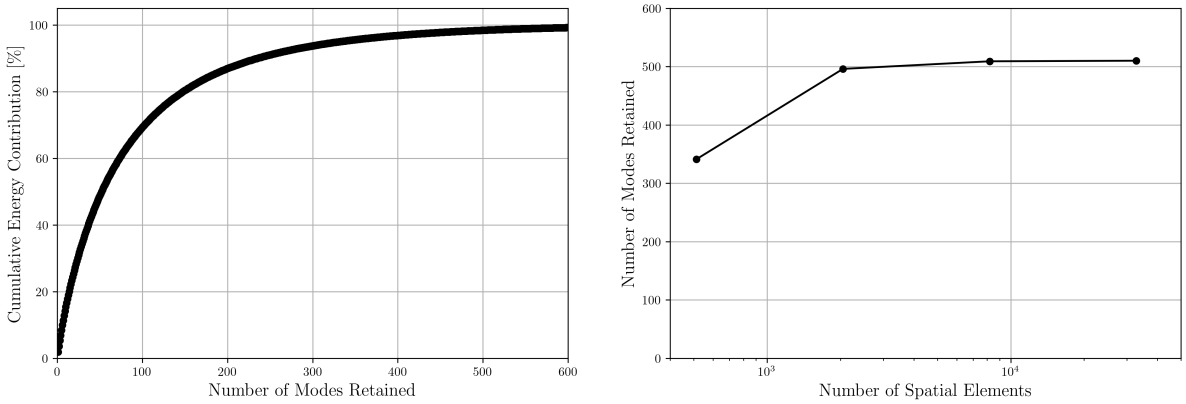


Figure 7.6: Full Snapshot Matrix Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_z \times N_y = 128 \times 64$ Spatial Elements and Number of Modes Retained of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

of the methodologies was quantitatively assessed and compared using the following evaluation metrics: Compression Ratio (CR), computational time and Mean Square Error (MSE) defined as follows:

$$\text{MSE}(\tilde{\boldsymbol{u}}) = \frac{1}{2N_t N_y N_z} \sum_{l=1}^{2} \sum_{k=1}^{N_t} \sum_{j=1}^{N_y} \sum_{i=1}^{N_z} \left(\tilde{\boldsymbol{u}}_l(\boldsymbol{x}_{ij}, t_k) - \boldsymbol{u}_l(\boldsymbol{x}_{ij}, t_k)\right)^2 \tag{7.8}$$

where $\tilde{\boldsymbol{u}}$ is the primal velocity obtained using each of the methodologies proposed. The evaluation metrics were crucial to assess the efficiency of the surrogate models to generate an approximate primal velocity. The results of the methodologies proposed are shown in this section.

7.2.1. Proper Orthogonal Decomposition

The *offline* POD method was first explored to construct a Reduced-Order Model (ROM) of the primal solution as a baseline method. Two methodologies were investigated due to the limitations of the classical approach when applied to high-dimensional datasets. For both methods, a 99% kinetic energy retention criterion was employed to reconstruct the primal velocity.

The first technique, based on the classical approach outlined in [62], started by assembling each fluctuating component $\boldsymbol{u}'_j(t) \in \mathbb{R}^{N_z \times N_y}$ into datasets $\boldsymbol{U}_j \in \mathbb{R}^{N_t \times N_z N_y}$. The two resulting matrices were horizontally concatenated to form a single snapshot matrix $\boldsymbol{U} \in \mathbb{R}^{2N_t \times N_z N_y}$. Singular Value Decomposition (SVD) was then applied to factorize the matrix $\boldsymbol{U}$.

The cumulative energy contribution of the first six hundred POD modes for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements and the optimal rank for each spatial resolution are shown in Figure 7.6. For a spatial resolution of $N_z \times N_y = 128 \times 64$ elements, approximately 500 POD modes were required to meet the reconstruction criterion. The optimal rank increased as the number of spatial elements increased, reaching a plateau at moderate spatial resolutions. This behaviour was due to the limited time interval. The optimal rank of the snapshot matrix was determined by its smallest dimension. At moderate and higher spatial resolutions, the number of snapshots was inferior to the number of degrees of freedom, thus the number of retained POD modes was constrained by the number of time steps. As the number of time steps was equal for the spatial resolutions considered, an increase in the number of spatial elements did not lead to a higher number of retained POD modes.

An alternative POD method was investigated to address the limitations of the first method. Instead of applying SVD to a single high-dimensional snapshot matrix, multiple snapshot matrices were assembled. The velocity fluctuation vectors $\boldsymbol{u}'_j(t) \in \mathbb{R}^{N_z \times N_y}$ were first assembled into N_y snapshot matrices $\boldsymbol{U}_k \in \mathbb{R}^{2N_t \times N_z}$. SVD was then applied to each snapshot matrix resulting in N_y different optimal ranks. The CR was defined for this method as follows:

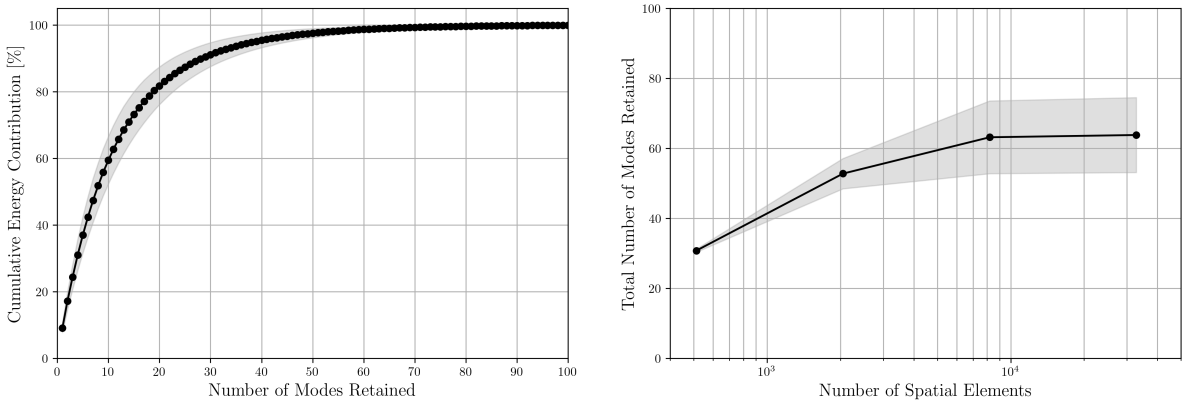


Figure 7.7: 2D Snapshot Matrix Proper Orthogonal Decomposition Modes Cumulative Energy Contribution for $N_z \times N_y = 128 \times 64$ Spatial Elements and Number of Modes Retained of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

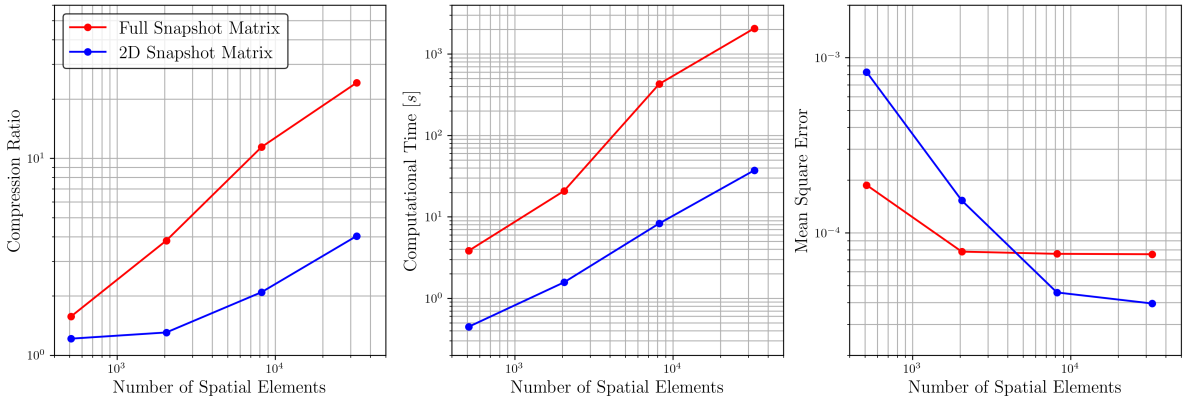


Figure 7.8: Compression Ratio, Computational Time and Mean Square Error for Proper Orthogonal Decomposition Methods of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

$$\text{CR} = \frac{\text{Space Resolution} \times \text{Time Resolution}}{\text{Size}(\overline{\boldsymbol{u}}) + \sum_{k=1}^{N_y} \left(\text{Size}(\boldsymbol{L}_{M,k}) + \text{Size}(\boldsymbol{\Sigma}_{M,k}) + \text{Size}(\boldsymbol{R}_{M,k})\right)} \tag{7.9}$$

For a spatial resolution of $N_z \times N_y = 128 \times 64$, the mean and standard deviation of cumulative energy contribution of the first hundred POD modes and the optimal rank for each spatial resolution considered is presented in Figure 7.7. The mean and standard deviation are computed for the N_y different SVD results. For a spatial resolution of $N_z \times N_y = 128 \times 64$ elements, an average of approximately 60 POD modes was required to satisfy the reconstruction criterion. The number of averaged POD modes increased as the spatial resolution increased, reaching a plateau for finer spatial grids. The plateau was attributed to the resolution of finer scales with lower energy as the spatial grid was refined. Also, the standard deviation of the number of POD modes retained increased as the spatial resolution increased. This behaviour was due to more turbulent scales being resolved as the spatial grid was refined. As regions near the wall contain more energetic scales than in the channel's centerline, the difference in energy was more relevant as the spatial resolution increased.

The CR, computational time and MSE relative to the FEM solution are presented in Figure 7.8 for the spatial resolutions considered to compare the two POD methods proposed. The first technique is referred to as the full snapshot matrix while the second method is labelled as the 2D snapshot matrix. The full snapshot matrix method presented a higher CR than the 2D method for the spatial resolutions considered. This was attributed to the retention of fewer modes in the classical approach.

Table 7.1: Convolutional Autoencoder Architecture of Unsteady 2D Viscous Burger's Equation

Encoder			**Decoder**		
Number of Trainable Parameters: 53,756			Number of Trainable Parameters: 53,746		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	(Size, Stride, Padding) $= (4, 1, 1)$ ReLU Activation Function	$(16, \boldsymbol{N_y - 1})$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 1, 1)$ ReLU Activation Function	$(64, \boldsymbol{N_y/4 + 1})$
Convolution	(Size, Stride, Padding) $= (5, 2, 2)$ ReLU Activation Function	$(32, \boldsymbol{N_y/2})$	**Transpose Convolution**	(Size, Stride, Padding) $= (4, 2, 2)$ ReLU Activation Function	$(32, \boldsymbol{N_y/2})$
Convolution	(Size, Stride, Padding) $= (4, 2, 2)$ ReLU Activation Function	$(64, \boldsymbol{N_y/4 + 1})$	**Transpose Convolution**	(Size, Stride, Padding) $= (5, 2, 2)$ ReLU Activation Function	$(16, \boldsymbol{N_y - 1})$
Convolution	(Size, Stride, Padding) $= (3, 1, 1)$	$(10, \boldsymbol{N_y/4 + 1})$	**Transpose Convolution**	(Size, Stride, Padding) $= (4, 1, 1)$	$(2, \boldsymbol{N_y})$

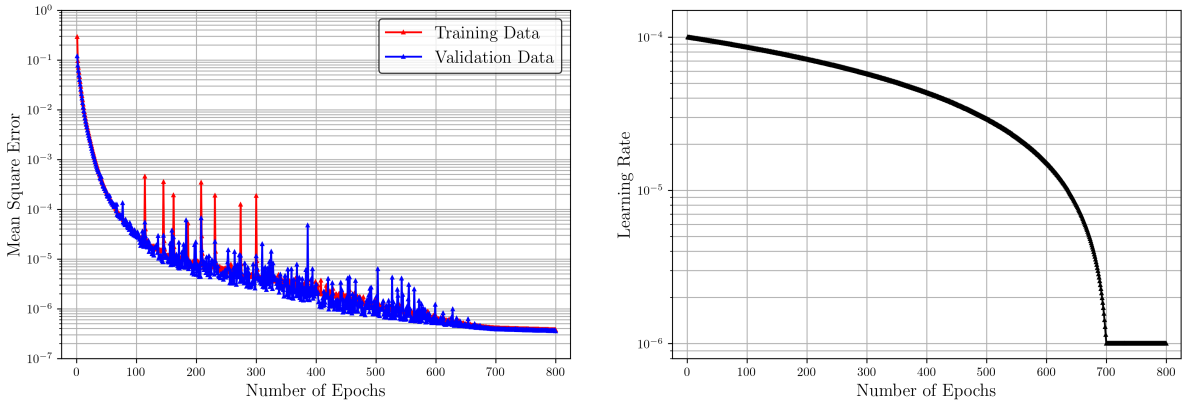


Figure 7.9: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equations for $N_z \times N_y = 128 \times 64$ Spatial Elements

The 2D method presented a lower computational time for the spatial resolutions considered. This was due to the higher efficiency of applying SVD to multiple smaller snapshot matrices than to a single larger matrix. In terms of accuracy, the full snapshot matrix method maintained a nearly constant value of the MSE for most of the spatial resolutions considered. The 2D method exhibited a convergent trend as the spatial grid was refined and outperformed the full method at higher spatial resolutions. Based on the results obtained, the 2D snapshot matrix method was preferred as it boasted a lower computational overhead and the MSE presented a convergent behaviour. The latter was due to the rank of the truncated matrices being consistently determined by the spatial dimension.

7.2.2. Convolutional Autoencoder

The second method proposed was a CAE to construct a surrogate model of the primal solution. Feed-forward layers were excluded from its architecture to avoid a large number of trainable parameters [7]. Thus, the architecture is only composed of Convolutional Neural Networks (CNNs).

A single CAE architecture was designed for the spatial resolutions considered. As a result, a different latent space dimension was imposed for each spatial resolution. The input layer consisted of two channels, each corresponding to a different velocity component. A window technique was introduced, which divided the spatial domain at $z = \pi/2$ into two subdomains $N_y \times N_y$ of equal width and height. Thus, kernel filters with equal size in the spanwise and wall-normal directions were used and the amount of original training data was doubled. The decoder architecture mirrored the encoder architecture but with transpose convolutional operators [78]. The CAE architecture used for the spatial resolutions considered is given in Table 7.1.

Additional training data was generated by adding Gaussian noise $\mathcal{N}(0, \sigma)$ with a standard deviation of $\sigma = 10^{-3}$ to improve the robustness of the model. This augmentation yielded a dataset with a total of 50,000 samples. A batch size of 75 samples was selected and the optimization procedure was performed over 800 epochs. Based on trial and error, the learning rate was started at $\gamma_0 = 1 \times 10^{-4}$ and decreased with a decay factor of $\gamma_r = 1 \times 10^{-2}$ for the first 700 epochs. Following the adaptive stage, the learning rate remained constant.

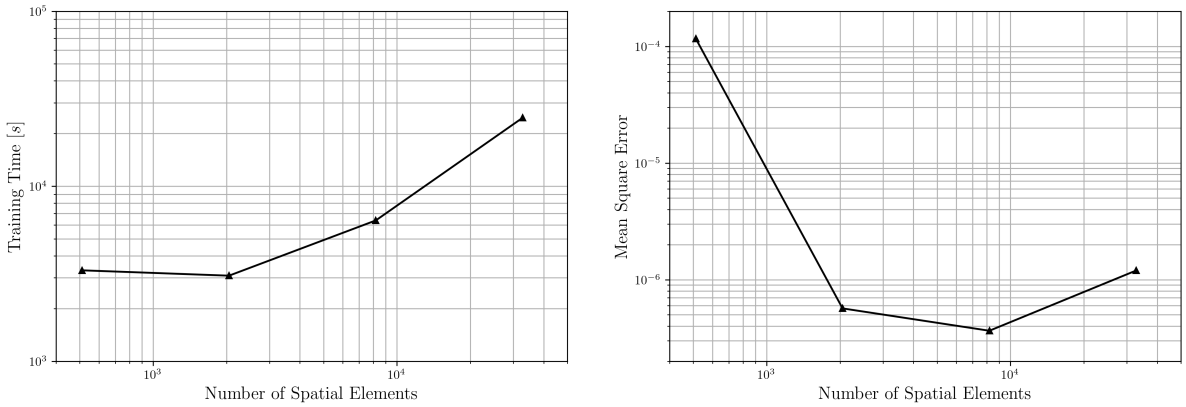


Figure 7.10: Training Time and Mean Square Error for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

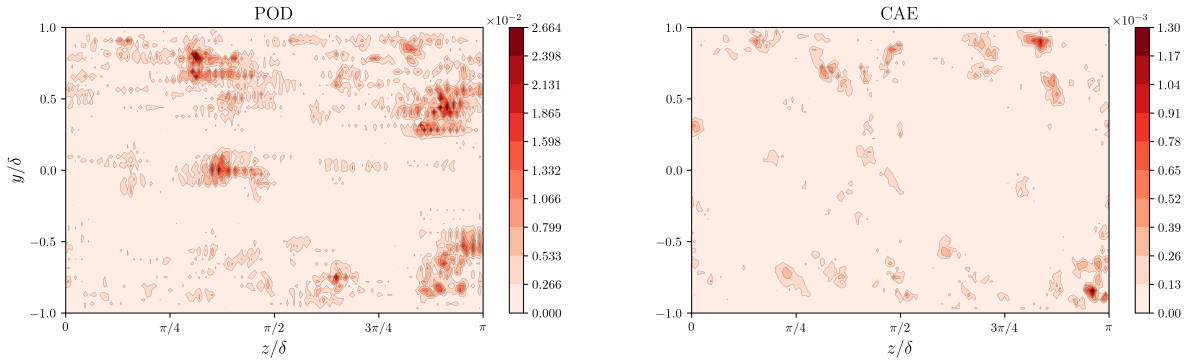


Figure 7.11: Relative Error of Surrogate Primal Velocity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

The MSE loss for the training and validation sets and the learning rate evolution is shown in Figure 7.9 for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements. After the optimization process, the training and validation MSE stabilize at approximately $O(10^{-6})$. The use of the adaptive learning rate enabled the exploration of a broader region around potential local minima. The MSE convergence for the training set at the end of the adaptive stage was much smoother than in the previous test cases. Thus, a higher number of epochs was unnecessary as the loss function was fully converged for the total number of epochs selected.

The training time and optimal MSE are shown in Figure 7.10 for the spatial resolutions considered. The training time remained constant for coarse spatial grids but increased exponentially for higher spatial resolutions. This behaviour arose from the increasing GPU memory and computational requirements needed as the spatial resolution increased. The MSE decreased for lower spatial resolutions while reaching a plateau at moderate spatial refinement levels. This trend could be attributed to the latent space selection for the spatial resolutions considered. A lower MSE value could be achieved for finer grids if a higher latent space dimension was selected.

Once training was complete, the CAE was used to construct a surrogate primal velocity. As a final step, the periodic and Dirichlet BCs were reapplied in the spanwise and wall-normal directions respectively. The training performance for the other spatial resolutions is provided in section A.3.

7.2.3. Performance Assessment

The performance of the surrogate primal velocities is now assessed. The relative error of the surrogate velocity magnitudes is shown in Figure 7.11 for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements and a time step $t = T$. The POD relative error was approximately $O(10^{-2})$ while the CAE achieved a lower relative error of $O(10^{-3})$. Additionally, the CAE error field was more localized than the POD one.

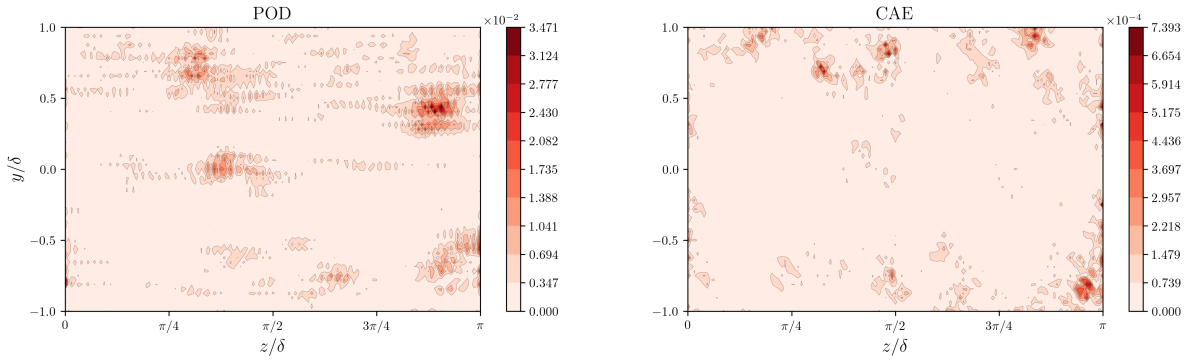


Figure 7.12: Relative Error of Surrogate Vorticity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

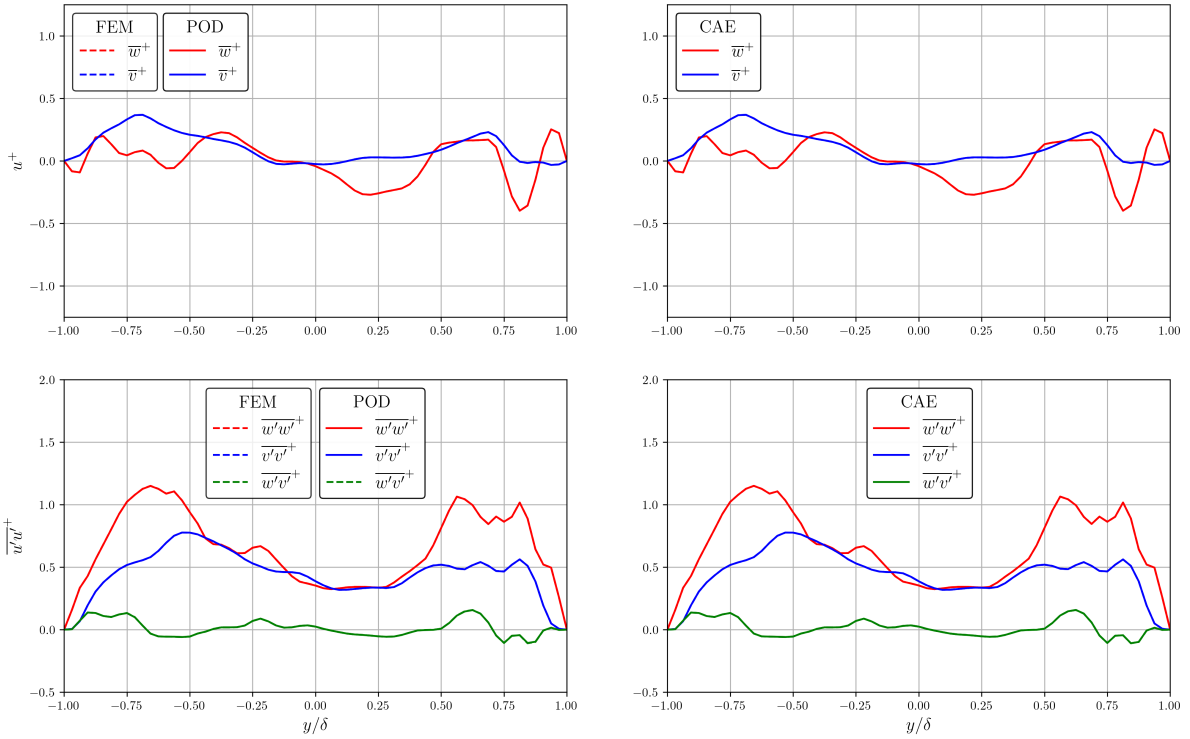


Figure 7.13: Surrogate Mean Primal Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

A similar analysis was conducted for the vorticity magnitude. The relative error of the surrogate vorticity magnitudes is evaluated in Figure 7.12 for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements and a time step $t = T$. The POD achieved a relative error of approximately $O(10^{-2})$, while the CAE attained a lower relative error of $O(10^{-3})$. However, the relative error field of both methods was similar in terms of sparseness.

The Reynolds decomposition was applied to the surrogate velocities. The resulting mean components and Reynolds stresses computed from the surrogate solutions are shown in Figure 7.13 for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements at a spanwise coordinate $z = \pi/2$. For reference, the FEM components computed are included for comparison. All methods effectively captured the mean components and Reynolds stresses. Deviations were not observed for either method.

A FFT was applied to the surrogate fluctuating components in the spanwise direction. The spectral energy density was then computed from the surrogate Fourier coefficients. For a spatial resolution of $N_z \times N_y = 128 \times 64$ elements, the relative error of the mean wall-based energy spectra is shown

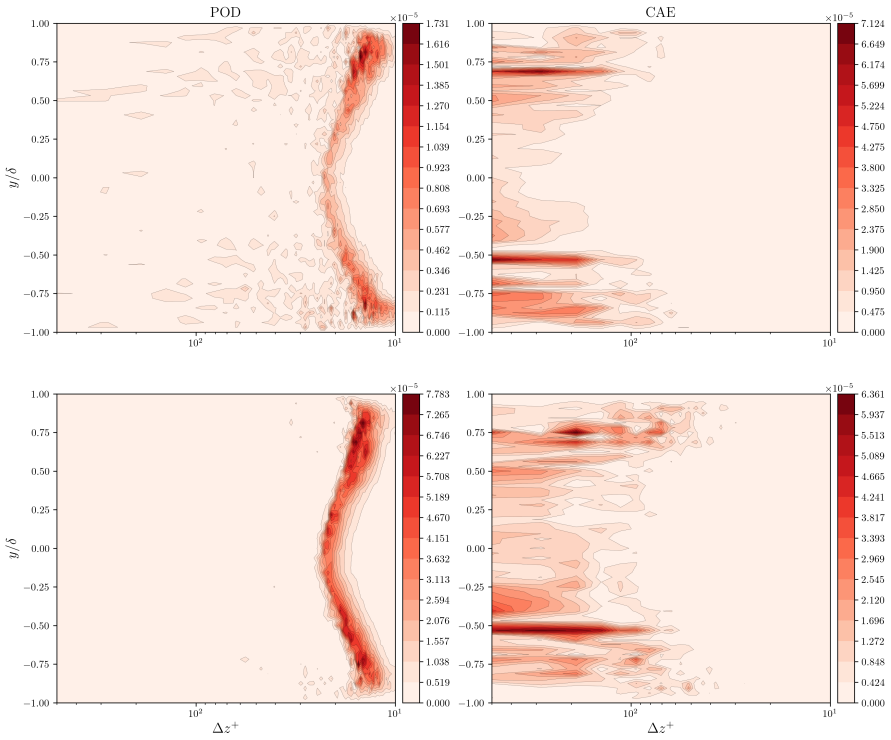


Figure 7.14: Relative Error of Mean Wall-Based 1D Energy Spectrum for Surrogate Primal Solutions of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements (Spanwise Component on Top and Wall-Normal on Bottom)

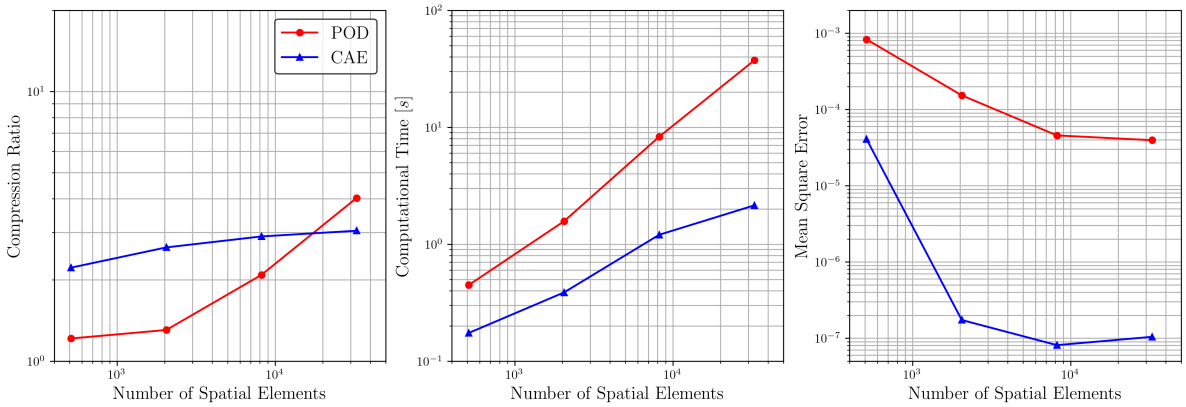


Figure 7.15: Evaluation Metrics of Surrogate Primal Solutions of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

in Figure 7.14. Both the spanwise and wall-normal components are displayed. The POD exhibited errors of approximately $O(10^{-5})$ for the lowest wavelengths, corresponding to higher wavenumbers. As in the previous test case, this was attributed to the snapshot matrix truncation. On the contrary, the CAE presented errors of the same order but for the highest wavelengths. However, the equal order of magnitude observed was due to the definition of the relative error employed: since the error was normalized by the maximum value of the true solution, regions with a higher absolute value were more pronounced than regions with an absolute value closer to zero. Thus, even though the POD provided slightly more accurate energy spectra for larger wavelengths, the CAE prediction for lower wavelengths was much more accurate.

The evaluation parameters, including CR, computational time, and MSE, are shown in Figure 7.15 for the spatial resolutions considered. The computational time only included the time required for the primal velocity prediction. The POD exhibited a constant value of the CR at lower spatial resolutions, with an

increase observed for more refined spatial grids. For coarser spatial grids, the CR remained constant due to the optimal rank being of the same order as the spatial resolution. The CAE demonstrated an increase in CR for coarser grids, followed by a plateau at higher spatial resolutions. This behaviour was due to the use of a single CAE architecture for the spatial resolutions considered, which resulted in a proportional latent space size. The POD presented a higher computational cost for the spatial resolutions considered, with a steeper growth rate compared to the CAE. This was due to the application of SVD being more computationally demanding than the application of convolution filters. In terms of accuracy, the CAE consistently outperformed the POD, achieving an MSE of approximately $O(10^{-7})$ for most spatial resolutions considered.

Ultimately, the CAE was the best method to construct a surrogate model of the primal velocity for the spatial resolutions considered. This superiority was attributed to the ability of the CAE to accurately reconstruct velocity and vorticity snapshots, preserve the energy spectrum and achieve the lowest MSE compared to the POD. Additionally, the CAE exhibited higher compression capacities with a lower computational overhead.

7.3. Adjoint Problem Formulation and Solution

The adjoint PDE was derived using the continuous technique. A local linearization around a specific primal state $\boldsymbol{u}$ was required due to the non-linearity of the Burgers' equations. Starting from the residual of the primal PDE, the residual sensitivity was defined as follows:

$$\boldsymbol{R}'[\boldsymbol{u}](\delta\boldsymbol{u}) = \frac{\partial(\delta\boldsymbol{u})}{\partial t} + (\delta\boldsymbol{u})\cdot\nabla\boldsymbol{u} + \boldsymbol{u}\cdot\nabla(\delta\boldsymbol{u}) - \frac{1}{Re_\tau}\nabla^2(\delta\boldsymbol{u}) \tag{7.10}$$

where $\boldsymbol{R}$ is the primal residual, and $\delta\boldsymbol{u}$ is an infinitesimal variation of the primal velocity. Based on the generalized form of the continuous adjoint equation, integration by parts was applied to derive the corresponding adjoint operator:

$$\begin{aligned} J'[\boldsymbol{u}](\delta\boldsymbol{u}) &= \int_I\int_\Omega \boldsymbol{\psi}\cdot\left(\frac{\partial(\delta\boldsymbol{u})}{\partial t} + (\delta\boldsymbol{u})\cdot\nabla\boldsymbol{u} + \boldsymbol{u}\cdot\nabla(\delta\boldsymbol{u}) - \frac{1}{Re_\tau}\nabla^2(\delta\boldsymbol{u})\right)\, d\Omega dI \\ &= \int_I\int_\Omega\left(-\frac{\partial\boldsymbol{\psi}}{\partial t} - \boldsymbol{u}\cdot\nabla\boldsymbol{\psi} - \frac{1}{Re_\tau}\nabla^2\boldsymbol{\psi}\right)\cdot(\delta\boldsymbol{u})\, d\Omega dI + \left[\int_\Omega \boldsymbol{\psi}\cdot(\delta\boldsymbol{u})\, d\Omega\right]_{\partial I} \\ &+ \frac{1}{Re_\tau}\left[\int_I \boldsymbol{\psi}\cdot\nabla(\delta\boldsymbol{u})\, dI\right]_{\partial\Omega} \end{aligned} \tag{7.11}$$

This process yielded the adjoint PDE along with the associated BCs and IC:

$$\begin{cases} -\dfrac{\partial\boldsymbol{\psi}}{\partial t} - \boldsymbol{u}\cdot\nabla\boldsymbol{\psi} - \dfrac{1}{Re_\tau}\nabla^2\boldsymbol{\psi} = \boldsymbol{g}, & (\boldsymbol{x},t)\in\Omega\times I \\ \boldsymbol{\psi}(\boldsymbol{x},t) = \boldsymbol{0}, & (\boldsymbol{x},t)\in\partial\Omega_{\text{Wall}}\times I \\ \boldsymbol{\psi}(\boldsymbol{x},T) = \boldsymbol{0}, & \boldsymbol{x}\in\Omega \end{cases} \tag{7.12}$$

where $\boldsymbol{g} = dJ/d\boldsymbol{u} = 2(e^{-50(y-\delta/2)^2} + e^{-50(y+\delta/2)^2})\boldsymbol{u}$ is the adjoint forcing term and the periodic BCs were preserved in the spanwise direction. As expected, the backwards marching of the adjoint velocity was concluded from the negative sign preceding the partial time derivative and the IC at $t = T$. Also, the linearity of the adjoint PDE was observed. The adjoint problem was solved using the implemented solver. As for the primal problem, the adjoint weak formulation must be derived. By considering weighting functions $\boldsymbol{q}\in V_{\boldsymbol{\psi}}$, the following variational formulation was obtained after integration by parts:

$$\int_\Omega\left(-\frac{\partial\boldsymbol{\psi}}{\partial t}\odot\boldsymbol{q} - (\boldsymbol{u}\cdot\nabla\boldsymbol{\psi})\odot\boldsymbol{q} + \frac{1}{Re_\tau}(\nabla\boldsymbol{\psi}\odot\nabla\boldsymbol{q}) - \boldsymbol{g}\odot\boldsymbol{q}\right)\, d\Omega = 0 \tag{7.13}$$

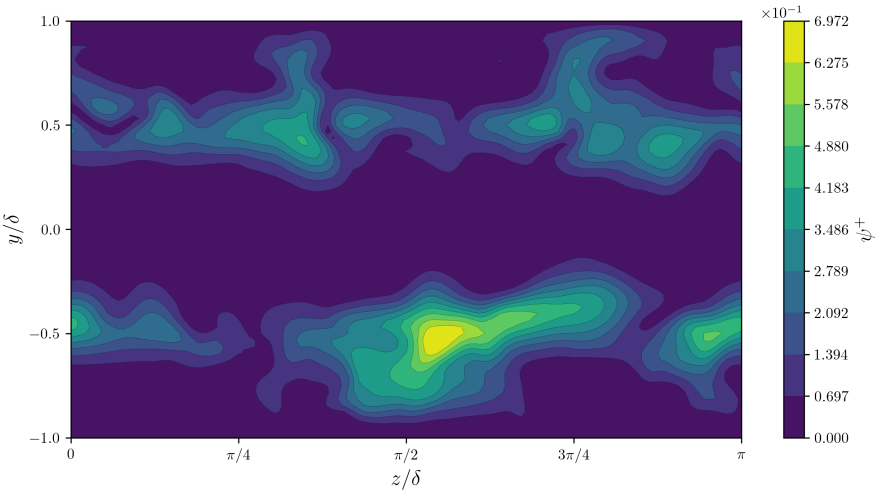


Figure 7.16: Adjoint Velocity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

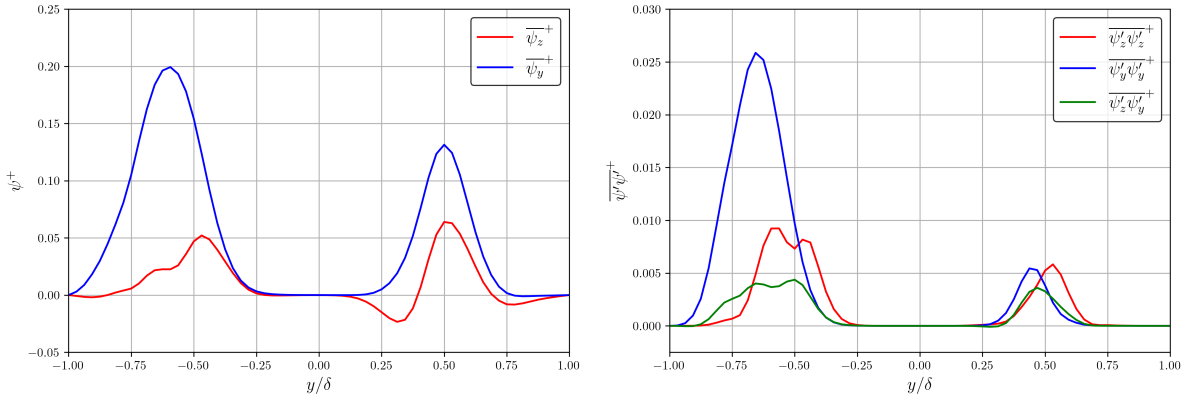


Figure 7.17: Mean Adjoint Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

Piece-wise continuous linear triangular elements were selected as the finite elements for the spatial discretization of the adjoint problem. The solving environment using `FEniCS` was the same as for the primal problem.

A snapshot of the FEM adjoint velocity magnitude ψ is shown in Figure 7.16 for $N_z \times N_y = 128 \times 64$ spatial elements and a time step $t = 0$. The adjoint velocity was concentrated in the log-law regions, due to the imposed adjoint forcing term.

The Reynolds decomposition was used to decompose each adjoint velocity component $\psi_j = \overline{\psi}_j + \psi'_j$ into a statistically-averaged component $\overline{\psi}_j$ and a fluctuating component ψ'_j. The latter components were evaluated in Reynolds stress form. The wall-based components $\overline{\psi}_j^+$ and adjoint Reynolds stresses are presented in Figure 7.3 for $N_z \times N_y = 128 \times 64$ spatial elements at a spanwise coordinate $z = \pi/2$. Both the mean components and adjoint Reynolds stresses were concentrated in the log-law regions, influenced by the adjoint forcing term. The magnitude of the adjoint Reynolds stress was lower than the one presented by the primal components. This was attributed to the linearity of the adjoint problem. Moreover, the wall-normal adjoint velocity component ψ_y exhibited a higher magnitude than the spanwise component ψ_z, both in terms of the mean components and Reynolds stresses. This was due to the presence of sweeps, ejection and burst events associated with turbulent streaks in the flowfield. Sweeps, ejections and bursts contribute to the production of TKE and are linked to the streamwise and wall-normal primal velocity fluctuations [94]. As the adjoint velocity reflects the sensitivity of the chosen

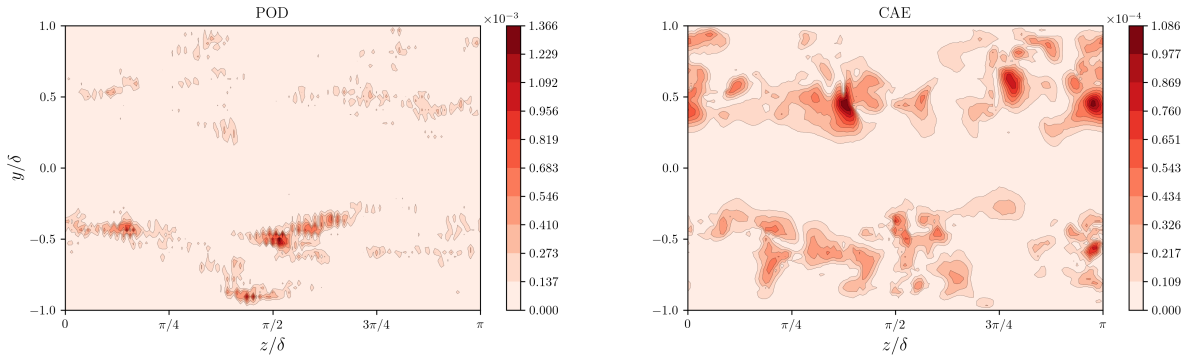


Figure 7.18: Relative Error of Surrogate Adjoint Velocity Magnitude Snapshot of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

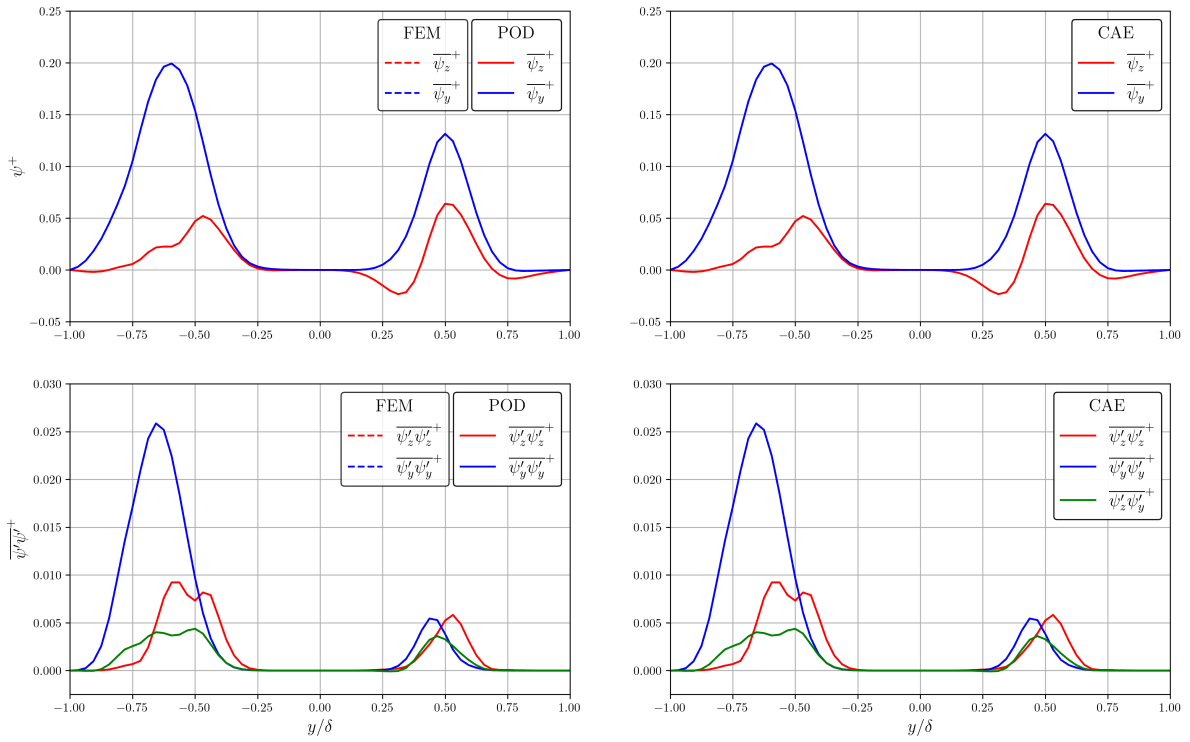


Figure 7.19: Surrogate Mean Adjoint Velocities and Reynolds Stresses of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

QoI, it was expected for the component ψ_y to be more significant than the component ψ_z. It was anticipated that a larger temporal domain would result in a more symmetric distribution of the adjoint mean velocities and Reynolds stresses about the channel centerline.

Adjoint velocities were computed with the surrogate primal velocities. The relative error in the surrogate adjoint velocity magnitudes is presented in Figure 7.11 for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements and a time step $t = 0$. The relative error of the POD was approximately $O(10^{-3})$, whereas the CAE achieved a lower relative error of $O(10^{-4})$. The POD error field was more localized compared to the CAE field. The superior accuracy of the CAE was due to a more accurate prediction of the primal velocity. Despite these observations, both methods were capable of providing an accurate adjoint velocity snapshot.

The Reynolds decomposition was applied to the surrogate adjoint velocities. The surrogate mean adjoint velocity components and Reynolds stresses are shown in Figure 7.13 for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements at a spanwise coordinate $z = \pi/2$. For reference, the components

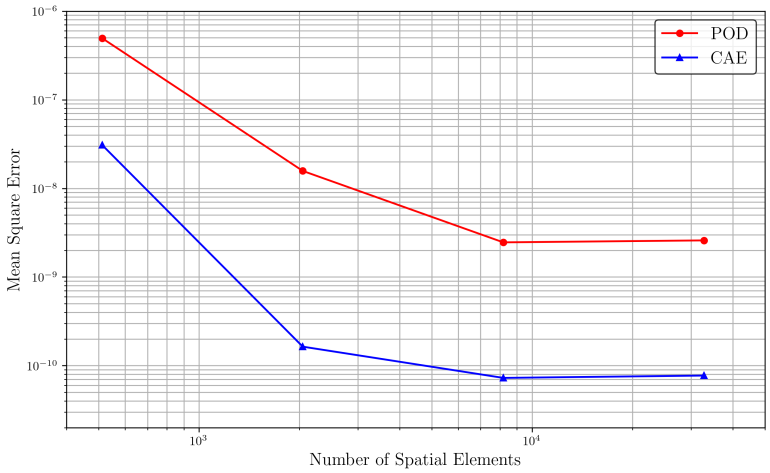


Figure 7.20: Mean Square Error of Surrogate Adjoint Velocities of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

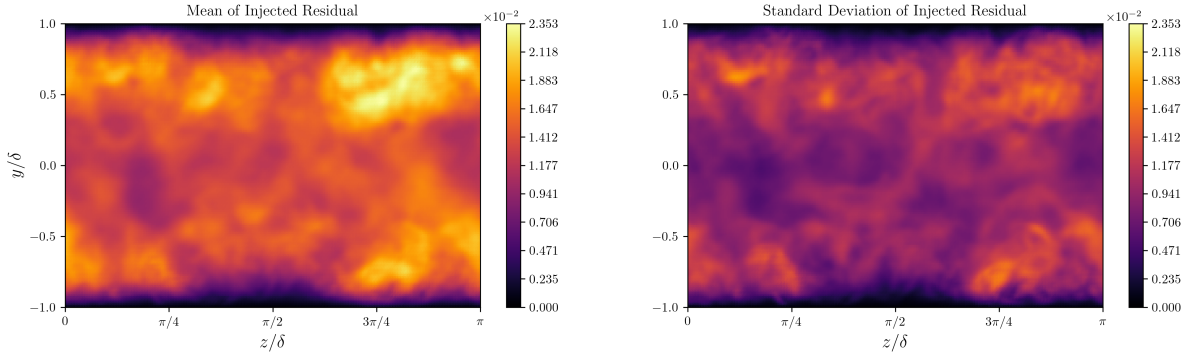


Figure 7.21: Primal Injected Residual Temporal Mean and Standard Deviation of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

obtained from the FEM velocity are included. All methods accurately captured the mean components and Reynolds stresses, with the surrogate velocities matching the FEM results.

The MSE of the surrogate adjoint velocities relative to the FEM adjoint velocity is presented in Figure 7.20 for the spatial resolutions considered. Both methods exhibited a decreasing trend in error with an increasing number of spatial elements, reaching a plateau at the highest spatial resolutions. However, the CAE consistently outperformed the POD for the spatial resolutions considered. The performance of the methods proposed for the adjoint velocity resembled their accuracy in constructing surrogate primal velocities. This was attributed to the linear nature of the adjoint PDE and the linearity of the adjoint forcing term to the primal velocity.

7.4. Adjoint-Based Error Estimation

Adjoint-based error estimation requires the primal and adjoint solutions in coarse and fine spaces. A linear interpolation of the coarse primal solution, defined in the coarse function space V_H, was performed into the fine function space V_h to obtain an injected primal solution. The fine-space strong residual was then evaluated with the injected primal solution. The strong residual and error estimates were evaluated within a Discontinuous Galerkin (DG) function space, with basis functions defined independently within each finite element. As in the previous test cases, the primal BCs were reinforced and the required derivatives to compute the residual were projected into quadratic function spaces.

The temporal mean and standard deviation of the fine-space residual magnitude computed with the injected primal solution are shown in Figure 7.21 for $N_z \times N_y = 128 \times 64$ coarse-space elements. The

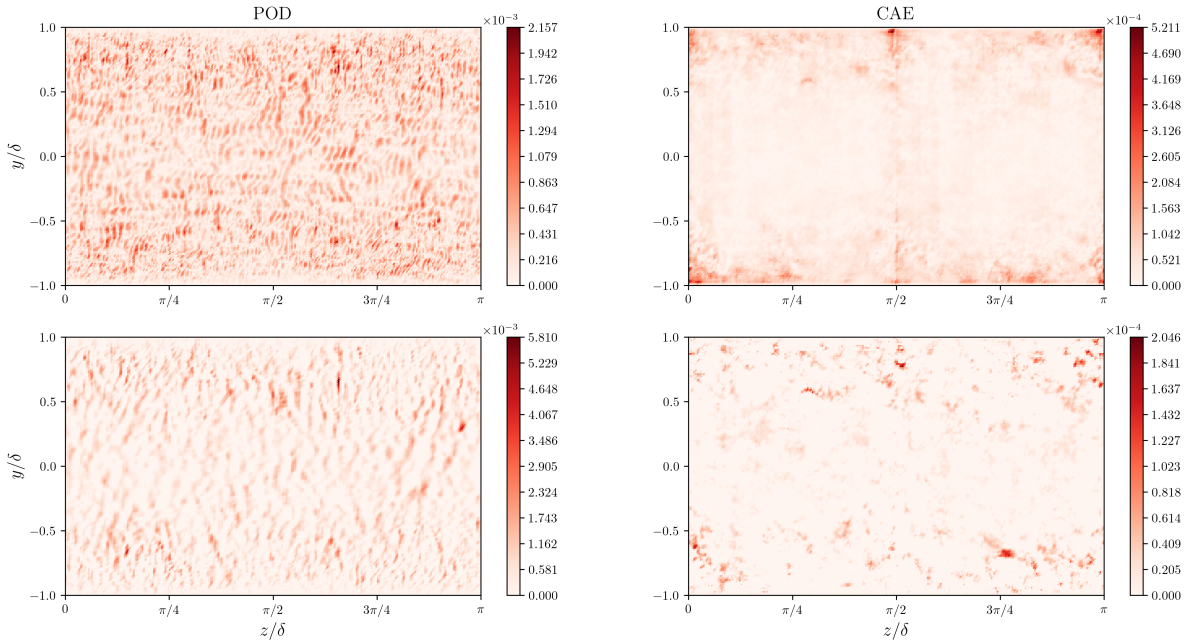


Figure 7.22: Relative Error of Surrogate Primal Injected Residual Temporal Mean and Standard Deviation of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements (Spanwise Component on Top and Wall-Normal on Bottom)

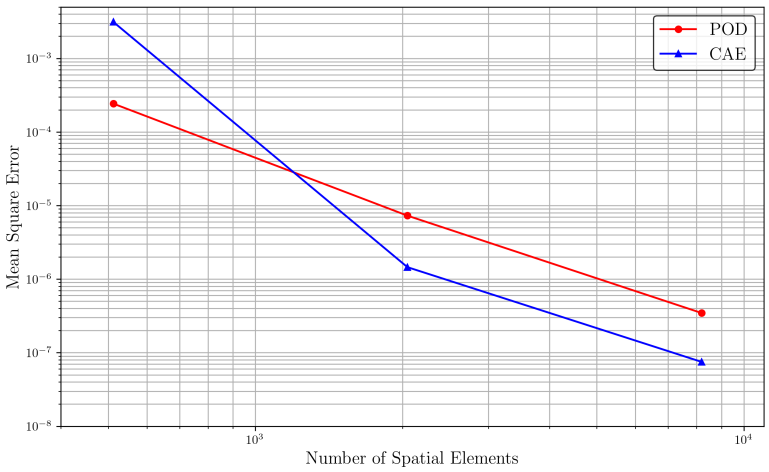


Figure 7.23: Mean Square Error of Surrogate Primal Injected Residuals of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

temporal mean of the injected residual presented a magnitude of approximately $O(10^{-2})$, with peaks observed near the walls. This behaviour was attributed to the need for higher refinement near the walls to accurately resolve large velocity gradients. The standard deviation of the residual was comparable to the temporal mean, highlighting the unsteadiness of the flowfield.

The relative error in terms of temporal mean and standard deviation of the surrogate injected residuals are presented in Figure 7.22 for $N_z \times N_y = 128 \times 64$ spatial elements. The relative error of the POD was approximately $O(10^{-3})$, whereas the CAE achieved a lower relative error of $O(10^{-4})$. Additionally, the POD error field was more sparse than the CAE field.

The MSE of the surrogate injected residuals is presented in Figure 7.23 for the spatial resolutions considered. Both methods exhibited a monotonic decrease in MSE as the number of spatial elements increased. For the lowest spatial resolutions, the POD was the most accurate method, while the CAE

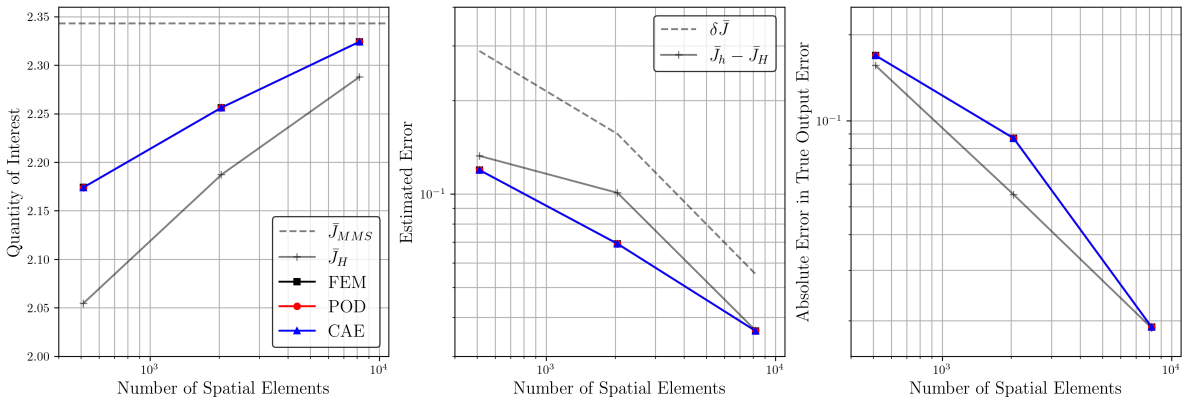


Figure 7.24: Quantity of Interest and Error Estimates of Surrogate Solutions of Unsteady 2D Viscous Burgers' Equations for Different Spatial Resolutions

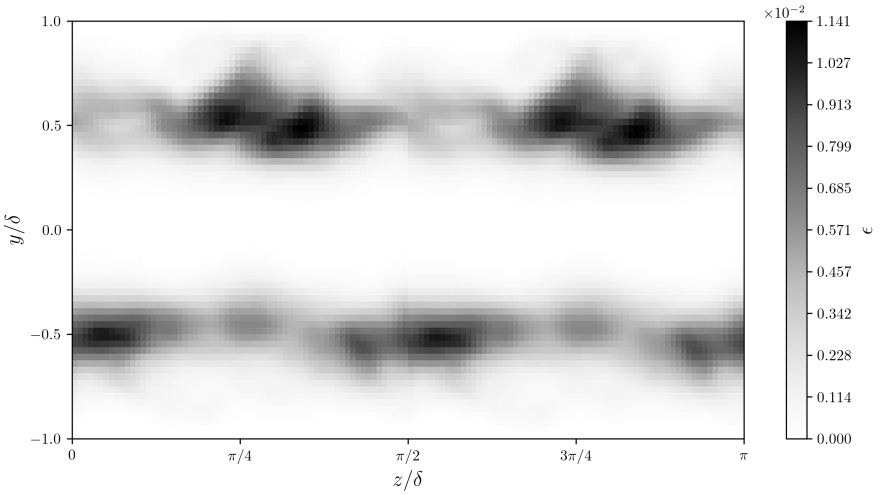


Figure 7.25: Time-Averaged Error Indicators of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 64$ Spatial Elements

proved to be more accurate for finer spatial grids.

Following the computation of the refined adjoint velocity and fine-space primal injected residual, the time-averaged adjoint-based error estimates $\delta\bar{J}_{est}$ were calculated for the FEM and surrogate velocities. The time-averaged adjoint-corrected QoIs $\bar{J} + \delta\bar{J}_{est}$, the adjoint-based error estimates $\delta\bar{J}_{est}$ and the absolute error in the true output error $\left|\delta\bar{J}_{est} - \delta\bar{J}\right|$ are presented in Figure 7.24 for the methods proposed and spatial resolutions considered. Both methods successfully predicted the adjoint-corrected QoI and the adjoint-based error estimates for the spatial resolutions considered. At lower spatial resolutions, the linearization error $O(\delta\boldsymbol{u}^2)$ inherent in non-linear problems was evident in the error between the true error estimate $\bar{J}_h - \bar{J}_H$ and the computed error estimates. This discrepancy was attributed to the primal velocity being underresolved. However, the influence of the linearization error was negligible for the finest spatial grid.

Local error indicators were computed which could be used to evaluate their suitability for mesh adaptation strategies. A time-averaged local error indicator was considered as the introduced by the temporal discretization scheme were negligible. The time-averaged error indicator field is shown in Figure 7.25 for a spatial resolution of $N_z \times N_y = 128 \times 64$ elements. The error field was concentrated in the log-law regions, reflecting the characteristics of the adjoint velocity. The influence of the injected residual was evident as well, as areas where the injected residual was significant exhibited a significant error indicator.

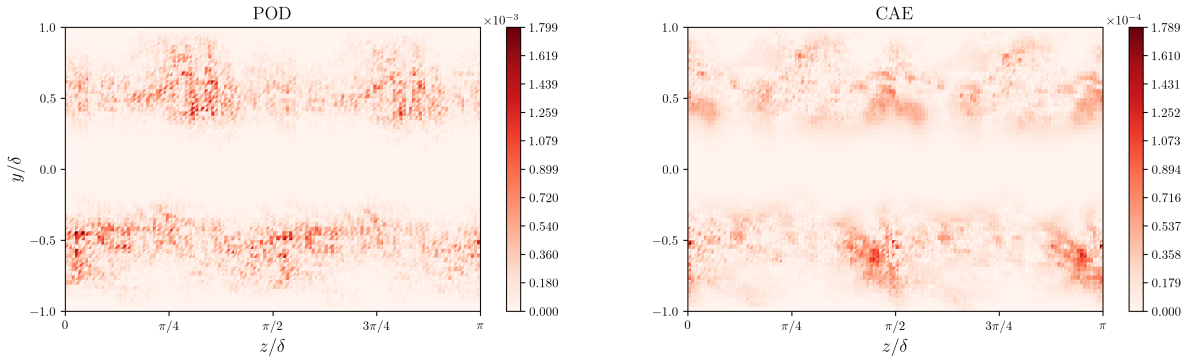


Figure 7.26: Time-Averaged Error Indicators of Surrogate Solutions of Unsteady 2D Viscous Burgers' Equations with $N_z \times N_y = 128 \times 128$ Spatial Elements

The relative error of the surrogate time-averaged error indicators with respect to the FEM result is presented in Figure 7.26 for $N_z \times N_y = 128 \times 64$ spatial elements. The POD exhibited a relative error of approximately $O(10^{-3})$ while the CAE achieved a relative error of $O(10^{-4})$. The relative error field of both methods was similar in terms of sparseness.

Both the POD and CAE were able to accurately compute the adjoint-based error estimates for the spatial resolutions considered. Regarding the local error indicator field, the CAE was the most accurate. The CAE consistently demonstrated a higher CR for the spatial resolutions considered, except for the most refined spatial grid. Adaptive Mesh Refinement (AMR) is particularly important for coarser meshes. In this context, the CAE was the best method due to its superior accuracy and higher CR for lower spatial resolutions. For higher spatial resolutions, the POD was the most suitable method, offering comparable accuracy to the CAE with a higher CR. As it was mentioned in the previous test case, the CR of the CAE could be easily adjusted by modifying the latent space size.

8

Conclusions and Future Work

In this chapter, the conclusions drawn from the obtained results are summarized and the research questions and objectives presented in the introductory chapter are addressed. The chapter closes with a section on future work and recommendations highlighting potential avenues for further development and improvement of the research presented.

Within error estimation methods, adjoint-based error estimation is considered the most accurate with respect to a user-defined Quantity of Interest (QoI) but it incurs the highest computational cost [7]. For unsteady non-linear problems such as the Navier-Stokes equations, the entire primal solution must be stored on disk to solve the adjoint problem. As a result, careful management of storage requirements is necessary. This research aimed to mitigate the substantial storage requirements associated with unsteady adjoint-based error estimation by the creation of a surrogate model of the primal solution. Three methodologies were proposed: a Convolutional AutoEncoder (CAE), an Echo State Network (ESN) and a hybrid approach of the first two, referred to as CAE-ESN. An *offline* variant of Proper Orthogonal Decomposition (POD) served as a baseline for comparison with the deep learning techniques. These methods were evaluated for their ability to construct a surrogate primal solution and for their effectiveness in accurately capturing the adjoint-based error estimate.

8.1. Conclusions

Firstly, a manufactured solution of the unsteady one-dimensional (1D) viscous Burgers' equation was employed as a verification tool for the computational framework. The POD presented the highest compression capabilities due to the retention of a unit optimal rank, as established in previous studies [6, 51]. Also, the POD provided the lowest computational overhead due to the low dimensionality of the problem. The training time was not included in the computational time of the deep learning techniques. All methodologies proposed were able to provide an accurate representation of the primal solution. Nonetheless, the ESN was the most accurate due to its training being a linear regression problem. However, the POD presented an accuracy similar to the ESN for higher spatial resolutions. Globally, the ESN was the best method for coarser spatial grids as it was the most accurate method with a reasonable compression rate. For finer spatial grids, the POD was the most suitable method, as it provided higher compression capabilities and an accuracy similar to that of the ESN. In the adjoint-based error estimation framework, the ESN and POD were the preferred methods for the lowest and highest spatial resolutions, respectively. Additionally, the CAE-based approaches were able to accurately capture the adjoint-based error estimate, with minor deviations observed for the CAE-ESN for finer spatial grids. These methods accurately predicted the adjoint solution but struggled to accurately capture the primal injected residual. The inaccuracy present in the injected residual predicted by the CAE-based methods led to errors in the magnitude of the time-averaged error indicator fields. This error was attributed to the error indicators being more sensitive to outliers than the adjoint-based error estimate. Despite this, all methods proposed provided an error indicator field which effectively identified areas in the spatial grid for refinement or coarsening.

Secondly, the unsteady 1D viscous Burgers' equation was forced by a Turbulent Channel Flow (TCF) dataset at a friction Reynolds number of $Re_\tau = 180$ to produce the wall-normal velocity component. For this problem, the ESN-based methods demonstrated the highest compression capabilities. The latent space size of the CAE was selected to be equal to the POD optimal rank, thus their compression rates were equivalent. The CAE presented the lowest computational cost while the POD displayed the steepest growth rate in computational cost as the number of spatial elements increased. The CAE was the most accurate method, while the POD had a similar accuracy for finer spatial grids. Regarding the energy spectrum, the POD exhibited the lowest cut-off wave number due to the snapshot matrix truncation. Nonetheless, the ESN was the best method to accurately predict the energy spectrum. Globally, the CAE was the best method to construct a surrogate primal solution as it was the most accurate and presented the lowest computational cost. Within the framework of adjoint-based error estimation, the CAE and POD were the preferred methods for coarser and finer spatial grids, respectively. Conversely, the ESN-based approaches struggled to provide an accurate error estimate due to their difficulty in capturing the primal injected residual. This difficulty was attributed to the lack of backward propagation in the ESN methodology and the insufficient temporal information to retrieve meaningful statistics from the flowfield. Despite these limitations, the error indicators produced by the ESN-based methods were effective in identifying the elements necessary for mesh adaptation. This behaviour was due to the characteristics of the primal injected residual of this test case.

Thirdly, the unsteady 2D viscous Burgers' equation was forced by the TCF dataset used in the previous test case to produce the spanwise and wall-normal velocity components. The high dimensionality of this test case rendered the ESN unsuitable, while the limited time interval considered also the CAE-ESN unsuitable. Furthermore, a POD method was developed that incurred lower computational costs compared to the application of SVD to one single snapshot matrix. In the developed approach, SVD was applied individually to snapshot matrices defined at each wall-normal coordinate. In terms of compression capabilities, the CAE was the best method for lower spatial resolutions while the POD had a better performance for the most refined spatial grid. Despite the development of a new POD method, the CAE still presented a lower computational overhead. The CAE also demonstrated superior accuracy in representing the primal solution. In terms of energy spectra, the CAE provided a higher cut-off wavenumber than the POD. Globally, the CAE was the most efficient method, as it was the most accurate with the lowest computational overhead and higher compression capabilities for coarser and moderate spatial grids. Regarding adjoint-based error estimation, both methods were able to accurately predict the error estimate and the error indicator fields. This was due to the less complex nature of the selected QoI compared to the previous test case. However, the CAE was still able to provide a more accurate representation of the adjoint solution and primal injected residual.

Ultimately, this investigation highlighted the potential of the CAE to outperform more traditional methods, such as the POD, in the context of unsteady adjoint-based error estimation. When the compression rates of the CAE and POD were equivalent, the CAE was able to provide a more accurate adjoint-based error estimate for coarser spatial grids. AMR is particularly important for coarser spatial meshes, where accurately representing the physics of the problem is difficult but much of the computational cost can be saved. As a result, the CAE was found to be the best method due to its efficiency. Even when the compression rate of the CAE was superior to the POD, the CAE outperformed the POD with a more accurate representation of the primal solution and the error estimate with lower computational overhead. Additionally, ESN-based methods failed to produce accurate adjoint-based error estimates. This deficiency was attributed to the absence of backward propagation in their methodology and the limited time intervals considered in the test cases, which restricted the retrieval of meaningful statistical information.

8.2. Future Work and Recommendations

This research explored different methodologies to construct a surrogate primal solution within the framework of unsteady adjoint-based error estimation. Based on the results of this research, the following recommendations are proposed to enhance and extend the presented work.

In the context of unsteady adjoint-based error estimation, the results of this research highlighted the potential of the CAE to outperform more traditional methods, such as the POD. An *offline* variant of the POD was considered. However, further investigation is necessary to evaluate the performance of the

CAE in comparison to optimized POD methodologies, such as an enhanced *online* POD [6].

The variational problems derived were solved using the legacy `FEniCS`[1] package. However, this framework was not optimal for the high-dimensional problem considered as it demanded high computational resources. Additionally, the implementation was challenging due to the limited availability of documentation. An alternative is the newer version, `FEniCSx`, which offers significant improvements in parallelization. These improvements could alleviate the computational overhead required for high-dimensional problems. However, the lack of documentation is still an issue. Another potential option is the `OpenFOAM` [2] environment. A more efficient solver could enable the consideration of larger temporal domains in the test cases. As a result, the ESN-based approaches may exhibit an improvement in their performance as more statistically-representative data would be available.

Although the CAE was the best method for the chaotic test cases investigated, there was room for improvement in its architecture and optimization process. A simple architecture was employed, which sufficed for the test cases considered. However, the use of more complex architectures could yield even better results. For instance, a mode-decomposing autoencoder [36] could improve the interpretability of the CAE's output modes by employing different decoders for the same latent space. Additionally, hierarchical autoencoders [38] extend the mode-decomposing approach by incorporating multiple encoders. This extension allows for the ordering of latent space modes based on their contribution to the reconstructed field. This not only improves the interpretability of the latent space modes but could also improve the compression capabilities of the CAE. In terms of the optimization procedure, alternative schedulers could be employed for the adaptive learning rate. In this research, a decaying linear learning rate was used which was sensitive to its parameters. These were selected based on trial and error. An alternative approach could be the use of a scheduler which automatically reduces the learning rate when a specified metric, like the loss function, plateaus. This type of scheduler is implemented in `PyTorch` and would circumvent the trial and error approach used in this research for the adaptive learning rate. Furthermore, the loss function could be reformulated as a metric with a physical meaning. One possibility is embedding physical constraints directly into the loss function [33, 37]. In the context of adjoint-based error estimation, the primal residual could be used as a loss function. While this approach could yield a more accurate primal solution and primal residual, it would increase the computational cost of the optimization process due to the need to evaluate derivatives of the input layer of the CAE.

The ESN-based approaches were found to be sub-optimal for the chaotic test cases, due to their inability to accurately capture the primal injected residual. This was attributed to the limited temporal domain considered. One potential solution could be the implementation of a new ESN to predict the primal injected residual. However, this would have an impact on the compression capabilities of the ESN. As the ESN-based methods demonstrated higher compression rates and lower computational overhead compared to other approaches, this approach could still be viable. For higher-dimensional cases, this approach might not be viable due to the inherent challenges posed by the high-dimensionality constraints of the ESN. Another alternative could be the use of a more robust Recurrent Neural Network (RNN) architecture. One example is the long short-term memory network, which uses backward propagation during the optimization procedure. Such networks could also be integrated with the CAE to develop a hybrid approach [74, 75]. This combined methodology has the potential to achieve higher compression capabilities with improved performance. However, an increase in the computational cost is anticipated which must be balanced against the increase in performance.

Lastly, another significant limitation in adjoint-based error estimation is the high computational cost due to the need for a refined adjoint solution. While this investigation focused solely on the storage requirements, the computational cost of the fine adjoint solution remains an important challenge. One potential solution to address both limitations is the incorporation of super-resolution Artificial Neural Networks (ANNs) [7] into the CAE architecture. This integration could enable the approximation of the fine-space primal solution without the need to directly run the primal solver. As a result, the computational cost associated with the fine-space adjoint solution could be reduced, while also alleviating the storage requirements for unsteady non-linear problems.

[1] `https://fenicsproject.org`
[2] https://www.openfoam.com

References

[1] Freddie D. Witherden and Antony Jameson. “Future directions of computational fluid dynamics”. In: 2017. ISBN: 9781624105067. DOI: 10.2514/6.2017-3791.

[2] Ugo Piomelli. *Wall-layer models for large-eddy simulations*. 2008. DOI: 10.1016/j.paerosci.2008.06.001.

[3] Jeffrey P. Slotnick and Dimitri J. Mavriplis. “A grand challenge for the advancement of numerical prediction of high lift aerodynamics”. In: 2021, pp. 1–21. ISBN: 9781624106095. DOI: 10.2514/6.2021-0955.

[4] Thomas N. Theis and Philip Wong. “The End of Moore’s Law: A New Beginning for Information Technology”. In: *Computing in Science and Engineering* 19 (2017), pp. 41–50.

[5] Freddie D. Witherden, Antony M. Farrington, and Peter E. Vincent. “PyFR: An open source framework for solving advection-diffusion type problems on streaming architectures using the flux reconstruction approach”. In: *Computer Physics Communications* 185 (2014), pp. 3028–3040. ISSN: 00104655. DOI: 10.1016/j.cpc.2014.07.011.

[6] Xiaodong Li. “Towards efficient adjoint-based mesh optimization for predictive large eddy simulation”. PhD thesis. Delft University of Technology, 2023.

[7] 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). ISSN: 03770427. DOI: 10.1016/j.cam.2023.115722.

[8] Jeffrey Slotnick et al. *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*. 2014. URL: http://www.sti.nasa.gov.

[9] Timothy J. Baker. “Mesh adaptation strategies for problems in fluid dynamics”. In: *Finite Elements in Analysis and Design* 25 (1997), pp. 243–273. ISSN: 0168-874X. DOI: 10.1016/S0168-874X(96)00032-7.

[10] Krzysztof J. Fidkowski and David L. Darmofal. “Review of output-based error estimation and mesh adaptation in computational fluid dynamics”. In: *AIAA Journal* 49 (2011), pp. 673–694. ISSN: 00011452. DOI: 10.2514/1.J050073.

[11] Christopher J. Roy. “Strategies for driving mesh adaptation in CFD”. In: *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition* (2009). DOI: 10.2514/6.2009-1302.

[12] Sébastien Léonard, Marc Terracol, and Pierre Sagaut. “A wavelet-based adaptive mesh refinement criterion for large-eddy simulation”. In: *Journal of Turbulence* 7 (2006), pp. 1–25. ISSN: 14685248. DOI: 10.1080/14685240601021608.

[13] Ismail B. Celik, Zeynep N. Cehreli, and Ibrahim Yavuz. “Index of resolution quality for large eddy simulations”. In: *Journal of Fluids Engineering* 127 (2005), pp. 949–958. ISSN: 00982202. DOI: 10.1115/1.1990201.

[14] Pierre Benard et al. “Mesh adaptation for large-eddy simulations in complex geometries”. In: *International Journal for Numerical Methods in Fluids* 81 (2016), pp. 719–740.

[15] Pengcheng Cui et al. “The comparison of adjoint-based grid adaptation and feature-based grid adaptation method”. In: vol. 2280. 2022. DOI: 10.1088/1742-6596/2280/1/012003.

[16] Yi Li. “Automatic mesh adaptation using the continuous adjoint approach and the spectral difference method”. PhD thesis. Stanford University, 2013.

[17] Tyrone S. Phillips and Christopher J. Roy. “Richardson extrapolation-based discretization uncertainty estimation for computational fluid dynamics”. In: *Journal of Fluids Engineering* 136 (2014). ISSN: 1528901X. DOI: 10.1115/1.4027353.

[18] 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 (2000), pp. 204–227. ISSN: 00219991. DOI: 10.1006/jcph.2000.6600.

[19] David A. Venditti and David L. Darmofal. “Grid adaptation for functional outputs: Application to two-dimensional inviscid flows”. In: *Journal of Computational Physics* 176 (2002), pp. 40–69. ISSN: 00219991. DOI: 10.1006/jcph.2001.6967.

[20] 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 (2003), pp. 22–46. ISSN: 00219991. DOI: 10.1016/S0021-9991(03)00074-3.

[21] Michael A. Park. “Adjoint-based, three-dimensional error prediction and grid adaptation”. In: *AIAA Journal* 42 (2004), pp. 1854–1862. ISSN: 00011452. DOI: 10.2514/1.10051.

[22] Roland Becker, Vincent Heuveline, and Rolf Rannacher. “An optimal control approach to adaptivity in computational fluid mechanics”. In: *International Journal for Numerical Methods in Fluids* 40 (2002), pp. 105–120. ISSN: 02712091. DOI: 10.1002/fld.269.

[23] Manav Bhatia and Philip Berany. “Adjoint-based h-adaptive calculation of generalized aerodynamic forces”. In: *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (2015). DOI: 10.2514/6.2015-0172.

[24] Michael Woopen, Georg May, and Jochen Schütz. “Adjoint-based error estimation and mesh adaptation for hybridized discontinuous Galerkin methods”. In: *International Journal for Numerical Methods in Fluids* 76 (2014), pp. 811–834. ISSN: 10970363. DOI: 10.1002/fld.3959.

[25] Todd A. Oliver. “A high-order, adaptive, discontinuous Galerkin finite element method for the Reynolds-averaged Navier-Stokes equations”. PhD thesis. Massachusetts Institute of Technology, 2004.

[26] Steven M. Kast. “Methods for optimal output prediction in computational fluid dynamics”. PhD thesis. University of Michigan, 2016.

[27] Johan Hoffman et al. “Towards a parameter-free method for high Reynolds number turbulent flow simulation based on adaptive finite element approximation”. In: *Computer Methods in Applied Mechanics and Engineering* 288 (2015), pp. 60–74. ISSN: 00457825. DOI: 10.1016/j.cma.2014.12.004.

[28] Johan Jansson et al. “Time-resolved adaptive direct FEM simulation of high-lift aircraft configurations”. In: 2018, pp. 67–92. ISBN: 9783319621364. DOI: 10.1007/978-3-319-62136-4_5.

[29] Timothy J. Barth. “Space-Time Error Representation and Estimation in Navier-Stokes Calculations”. In: *Lecture Notes in Computational Science and Engineering*. 2007, pp. 29–48.

[30] Krzysztof J. Fidkowski and Guodong Chen. “A machine-learning anisotropy detection algorithm for output-adapted meshes”. In: *AIAA Scitech 2020 Forum* (2020). DOI: 10.2514/6.2020-0341.

[31] 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. ISSN: 10902716. DOI: 10.1016/j.jcp.2017.04.005.

[32] Nathaniel Fout and Kwan-Liu Ma. “An Adaptive Prediction-Based Approach to Lossless Compression of Floating-Point Volume Data”. In: *IEEE Transactions on Visualization and Computer Graphics* 18 (2012), pp. 2295–2304.

[33] Mohammadreza Momenifar et al. “Dimension reduced turbulent flow data from deep vector quantisers”. In: *Journal of Turbulence* 23 (2022), pp. 232–264. ISSN: 14685248. DOI: 10.1080/14685248.2022.2060508.

[34] Pranshu Pant et al. “Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations”. In: *Physics of Fluids* 33 (2021). ISSN: 10897666. DOI: 10.1063/5.0062546.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[36] Takaaki Murata, Kai Fukami, and Koji Fukagata. “Nonlinear mode decomposition with convolutional neural networks for fluid dynamics”. In: *Journal of Fluid Mechanics* 882 (2020). ISSN: 14697645. DOI: 10.1017/jfm.2019.822.

[37] Alberto Olmo et al. *Physics-Driven Convolutional Autoencoder Approach for CFD Data Compressions*. 2022. arXiv: 2210.09262.

[38] 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 (2020). ISSN: 10897666. DOI: 10.1063/5.0020721.

[39] Alberto Racca, Nguyen A. K. Doan, and Luca Magri. “Predicting turbulent dynamics with the convolutional autoencoder echo state network”. In: *Journal of Fluid Mechanics* 975 (2023). ISSN: 14697645. DOI: 10.1017/jfm.2023.716.

[40] 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. ISSN: 18792782. DOI: 10.1016/j.neunet.2021.05.004.

[41] Michael B. Giles and Niles A. Pierce. “An Introduction to the Adjoint Approach to Design”. In: *Flow, Turbulence and Combustion* 65 (2000), pp. 393–415.

[42] Lei Shi and Zhi J. 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. ISSN: 10902716. DOI: 10.1016/j.jcp.2015.04.011.

[43] Ralf Hartmann. “Adjoint consistency analysis of discontinuous Galerkin discretizations”. In: *SIAM Journal on Numerical Analysis* 45 (2007), pp. 2671–2696. ISSN: 00361429. DOI: 10.1137/060665117.

[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 (2011), pp. 5753–5773. ISSN: 10902716. DOI: 10.1016/j.jcp.2011.03.059.

[45] 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). ISSN: 00457825. DOI: 10.1016/j.cma.2021.113733.

[46] Qiqi Wang, Parviz Moin, and Gianluca Iaccarino. “Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation”. In: *SIAM Journal on Scientific Computing* 31 (2009), pp. 2549–2567. ISSN: 10648275. DOI: 10.1137/080727890.

[47] Patrick J. Blonigan et al. “Least-squares shadowing sensitivity analysis of chaotic flow around a two-dimensional airfoil”. In: *AIAA Journal* 56 (2018), pp. 658–672. ISSN: 00011452. DOI: 10.2514/1.J055389.

[48] Guodong Chen and Krzysztof J. Fidkowski. “Output-based adaptive aerodynamic simulations using convolutional neural networks”. In: *Computers and Fluids* 223 (2021). ISSN: 00457930. DOI: 10.1016/j.compfluid.2021.104947.

[49] Mengfei Xu et al. “Machine learning for adjoint vector in aerodynamic shape optimization”. In: *Acta Mechanica Sinica* 37 (2021), pp. 1416–1432. ISSN: 16143116. DOI: 10.1007/s10409-021-01119-6.

[50] Michael Besier and Rolf Rannacher. “Goal-oriented space-time adaptivity in the finite element Galerkin method for the computation of nonstationary incompressible flow”. In: *International Journal for Numerical Methods in Fluids* 70 (2012), pp. 1139–1166. ISSN: 02712091. DOI: 10.1002/fld.2735.

[51] Arnish Sitaram. “Output error estimation for unsteady flows using reconstructed solutions”. MSc Thesis. Delft University of Technology, 2023.

[52] Yao Jiang and Siva Nadarajah. “Adjoint-based error estimation for grid adaptation for large eddy simulation”. In: *Computers and Fluids* 236 (2022). ISSN: 00457930. DOI: 10.1016/j.compfluid.2021.105295.

[53] Yukiko S. Shimizu and Krzysztof J. Fidkowski. “Output-based error estimation for chaotic flows using reduced-order modeling”. In: *AIAA Aerospace Sciences Meeting, 2018* (2018). DOI: 10.2514/6.2018-0826.

[54] 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). ISSN: 00457825. DOI: 10.1016/j.cma.2022.115322.

[55] Aston Zhang et al. *Dive into Deep Learning*. https://D2L.ai. Cambridge University Press, 2023.

[56] Kunihiko Taira et al. “Modal analysis of fluid flows: applications and outlook”. In: *AIAA Journal* 58 (2020), pp. 998–1022. ISSN: 00011452. DOI: 10.2514/1.J058462.

[57] John L. Lumley. *Stochastic tools in turbulence*. Academic Press, 1970.

[58] Peter J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28. ISSN: 14697645. DOI: 10.1017/S0022112010001217.

[59] 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. ISSN: 00295493. DOI: 10.1016/j.nucengdes.2019.01.015.

[60] Miguel A. Mendez, Mikhaël Balabane, and Jean M. Buchlin. “Multi-scale proper orthogonal decomposition of complex fluid flows”. In: *Journal of Fluid Mechanics* 870 (2019), pp. 988–1036. ISSN: 14697645. DOI: 10.1017/jfm.2019.212.

[61] Willem Cazemier, Roel W.C.P. Verstappen, and Arthur E.P. Veldman. “Proper orthogonal decomposition and low-dimensional models for driven cavity flows”. In: *Physics of Fluids* 10 (1998), pp. 1685–1699. ISSN: 10706631. DOI: 10.1063/1.869686.

[62] Julien Weiss. “A tutorial on the proper orthogonal decomposition”. In: 2019, pp. 1–21. ISBN: 9781624105890. DOI: 10.2514/6.2019-3333.

[63] Lawrence Sirovich. “Turbulence and the dynamics of coherent structures. II. Symmetries and transformations”. In: *Quarterly of Applied Mathematics* 45 (1987), pp. 573–582. ISSN: 0033-569X. DOI: 10.1090/qam/910463.

[64] James P. Howard. “Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data”. In: *Journal of Statistical Software* 67 (2015). DOI: 10.18637/jss.v067.b01.

[65] Andrew Glaws, Ryan King, and Michael Sprague. “Deep learning for in situ data compression of large turbulent flow simulations”. In: *Physical Review Fluids* 5 (2020). ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.5.114602.

[66] Michael I. Jordan and Tom M. Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349 (2015), pp. 255–260. ISSN: 10959203. DOI: 10.1126/science.aac4520.

[67] Oludare I. Abiodun et al. “State-of-the-art in artificial neural network applications: A survey”. In: *Heliyon* 4 (2018). DOI: 10.1016/j.heliyon.2018.

[68] Mohaiminul Islam, Guorong Chen, and Shangzhu Jin. “An Overview of Neural Network”. In: *American Journal of Neural Networks and Applications* 5 (2019), pp. 7–11. ISSN: 2469-7400. DOI: 10.11648/j.ajnna.20190501.12.

[69] Kevin T. Carlberg et al. “Recovering missing CFD data for high-order discretizations using deep neural networks and dynamics learning”. In: *Journal of Computational Physics* 395 (2019), 105–124. ISSN: 10902716. DOI: 10.1016/j.jcp.2019.05.041.

[70] Bo Liu et al. “Deep learning methods for super-resolution reconstruction of turbulent flows”. In: *Physics of Fluids* 32 (Feb. 2020). ISSN: 10897666. DOI: 10.1063/1.5140772.

[71] Chinchun Ooi et al. “Modeling transient fluid simulations with proper orthogonal decomposition and machine learning”. In: *International Journal for Numerical Methods in Fluids* 93 (2021), pp. 396–410. ISSN: 10970363. DOI: 10.1002/fld.4888.

[72] Jean P. Jordanou et al. “Investigation of proper orthogonal decomposition for echo state networks”. In: *Neurocomputing* 548 (2023). ISSN: 18728286. DOI: 10.1016/j.neucom.2023.126395.

[73] Doreen Jirak et al. “Echo State Networks and Long Short-Term Memory for Continuous Gesture Recognition: a Comparative Study”. In: *Cognitive Computation* 15 (2023), pp. 1427–1439. ISSN: 18669964. DOI: 10.1007/s12559-020-09754-0.

[74] Arvind T. Mohan et al. “Spatio-temporal deep learning models of 3D turbulence with physics informed diagnostics”. In: *Journal of Turbulence* 21 (2020), pp. 484–524. ISSN: 14685248. DOI: 10.1080/14685248.2020.1832230.

[75] Rohan Thavarajah et al. “Fast modeling and understanding fluid dynamics systems with encoder–decoder networks”. In: *Machine Learning: Science and Technology* 2 (2021), p. 025022. DOI: 10.1088/2632-2153/abd1cf.

[76] Arvind T. Mohan et al. “Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence”. In: *Physical Review Fluids* 8 (2023). ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.8.014604.

[77] Quanji Cai et al. “On the natural stabilization of convection dominated problems using high order Bubnov-Galerkin finite elements”. In: *Computers and Mathematics with Applications* 66 (2014), pp. 2545–2558. ISSN: 08981221. DOI: 10.1016/j.camwa.2013.09.009.

[78] Yuepeng Zhou et al. “DenseUNet: Improved image classification method using standard convolution and dense transposed convolution”. In: *Knowledge-Based Systems* 254 (2022). ISSN: 09507051. DOI: 10.1016/j.knosys.2022.109658.

[79] Mantas Lukoševičius. “A Practical Guide to Applying Echo State Networks”. In: vol. 7700. Springer, 2012, pp. 659–686.

[80] Herbert Jaeger. *The” echo state” approach to analysing and training recurrent neural networks-with an erratum note’*. 2001.

[81] Zhixin Lu et al. “Reservoir observers: Model-free inference of unmeasured variables in chaotic systems”. In: *Chaos* 27 (2017). ISSN: 10541500. DOI: 10.1063/1.4979665.

[82] Nikolay Nikitin. “Characteristics of the leading Lyapunov vector in a turbulent channel flow”. In: *Journal of Fluid Mechanics* 849 (2018), pp. 942–967. ISSN: 14697645. DOI: 10.1017/jfm.2018.418.

[83] Eric Brochu, Vlad Cora, and Nando De Freitas. *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. 2010. URL: http://arxiv.org/abs/1012.2599.

[84] Pantelis R. Vlachas et al. “Backpropagation Algorithms and Reservoir Computing in Recurrent Neural Networks for the Forecasting of Complex Spatiotemporal Dynamics”. In: *Neural Networks* 126 (2020), pp. 191–217. DOI: 10.1016/j.neunet.2020.02.016.

[85] Bülent Saka and Idris Dag. “A numerical study of the Burgers’ equation”. In: *Journal of the Franklin Institute* 345 (2008), pp. 328–348. ISSN: 00160032. DOI: 10.1016/j.jfranklin.2007.10.004.

[86] Patrick J. Roache. “Code verification by the method of manufactured solutions”. In: *Journal of Fluids Engineering* 124 (2002), pp. 4–10. ISSN: 00982202. DOI: 10.1115/1.1436090.

[87] Sergio Hoyas et al. “Wall turbulence at high friction Reynolds numbers”. In: *Physical Review Fluids* 7 (2022). ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.7.014602.

[88] Sergio Hoyas and Javier Jiménez. “Scaling of the velocity fluctuations in turbulent channels up to $Re_\tau = 2003$”. In: *Physics of Fluids* 18 (2006). ISSN: 10706631. DOI: 10.1063/1.2162185.

[89] Robert D. Moser, John Kim, and Nagi N. Mansour. “Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$”. In: *Physics of Fluids* 11 (1999), pp. 943–945. ISSN: 10706631. DOI: 10.1063/1.869966.

[90] Steven J. Hulshoff. “Implicit subgrid-scale models in space-time variational-multiscale discretizations”. In: *International Journal for Numerical Methods in Fluids* 47 (2005), pp. 1093–1099. ISSN: 02712091. DOI: 10.1002/fld.879.

[91] Jung-Il Choi, Kyongmin Yeo, and Changhoon Lee. “Lagrangian statistics in turbulent channel flow”. In: *Physics of Fluids* 16 (2004), pp. 779–793. ISSN: 10706631. DOI: 10.1063/1.1644576.

[92] Yu N. Zhang et al. “Comparisons and analyses of vortex identification between Omega method and Q criterion”. In: *Journal of Hydrodynamics* 31 (2 2019), pp. 224–230. ISSN: 18780342. DOI: 10.1007/s42241-019-0025-1.

[93] Juan C. Del Álamo et al. “Scaling of the energy spectra of turbulent channels”. In: *Journal of Fluid Mechanics* 500 (2004), pp. 135–144. ISSN: 00221120. DOI: 10.1017/S002211200300733X.

[94] William G. Tiederman. “Characteristics of ejections in turbulent channel flow”. In: *Journal of Fluid Mechanics* 179 (1987), pp. 1–19. ISSN: 14697645. DOI: 10.1017/S002211208700140X.

A

Convolutional Autoencoder Architectures and Training

A.1. Manufactured Solution

Table A.1: Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 8$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 5,264			Number of Trainable Parameters: 5,257		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$ ReLU Activation Function	$(8, 7)$	**Linear**	$(\text{In}, \text{Out}) = (8, 16)$ ReLU Activation Function	(16)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(16, 4)$	**Linear**	$(\text{In}, \text{Out}) = (16, 32)$ ReLU Activation Function	(32)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(32, 3)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 0, 0)$ ReLU Activation Function	$(32, 3)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 0, 0)$ ReLU Activation Function	$(32, 1)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(16, 4)$
Linear	$(\text{In}, \text{Out}) = (32, 16)$ ReLU Activation Function	(16)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(8, 7)$
Linear	$(\text{In}, \text{Out}) = (16, 8)$	(8)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$	$(1, 8)$

Table A.2: Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 16$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 8,368			Number of Trainable Parameters: 8,361		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$ ReLU Activation Function	$(8, 15)$	**Linear**	$(\text{In}, \text{Out}) = (8, 16)$ ReLU Activation Function	(16)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(16, 8)$	**Linear**	$(\text{In}, \text{Out}) = (16, 32)$ ReLU Activation Function	(32)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(32, 5)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 0, 0)$ ReLU Activation Function	$(32, 3)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(32, 3)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(32, 5)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 0, 0)$ ReLU Activation Function	$(32, 1)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(16, 8)$
Linear	$(\text{In}, \text{Out}) = (32, 16)$ ReLU Activation Function	(16)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(8, 15)$
Linear	$(\text{In}, \text{Out}) = (16, 8)$	(8)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$	$(1, 16)$

Table A.3: Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 64$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 14,576			Number of Trainable Parameters: 14,569		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	(Size, Stride, Padding) $= (4, 1, 1)$ ReLU Activation Function	$(8, 63)$	**Linear**	(In, Out) $= (8, 16)$ ReLU Activation Function	(16)
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(16, 32)$	**Linear**	(In, Out) $= (16, 32)$ ReLU Activation Function	(32)
Convolution	(Size, Stride, Padding) $= (2, 2, 1)$ ReLU Activation Function	$(32, 17)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 0, 0)$ ReLU Activation Function	$(32, 3)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 9)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 5)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 5)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 9)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 3)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 17)$
Convolution	(Size, Stride, Padding) $= (3, 0, 0)$ ReLU Activation Function	$(32, 1)$	**Transpose Convolution**	(Size, Stride, Padding) $= (2, 2, 1)$ ReLU Activation Function	$(16, 32)$
Linear	(In, Out) $= (32, 16)$ ReLU Activation Function	(16)	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(8, 63)$
Linear	(In, Out) $= (16, 8)$	(8)	**Transpose Convolution**	(Size, Stride, Padding) $= (4, 1, 1)$	$(1, 64)$

Table A.4: Convolutional Autoencoder Architecture of Manufactured Solution for $N_x = 128$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 17,680			Number of Trainable Parameters: 17,673		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	(Size, Stride, Padding) $= (4, 1, 1)$ ReLU Activation Function	$(8, 127)$	**Linear**	(In, Out) $= (8, 16)$ ReLU Activation Function	(16)
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(16, 64)$	**Linear**	(In, Out) $= (16, 32)$ ReLU Activation Function	(32)
Convolution	(Size, Stride, Padding) $= (2, 2, 1)$ ReLU Activation Function	$(32, 33)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 0, 0)$ ReLU Activation Function	$(32, 3)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 17)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 5)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 9)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 9)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 5)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 17)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 3)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 33)$
Convolution	(Size, Stride, Padding) $= (3, 0, 0)$ ReLU Activation Function	$(32, 1)$	**Transpose Convolution**	(Size, Stride, Padding) $= (2, 2, 1)$ ReLU Activation Function	$(16, 64)$
Linear	(In, Out) $= (32, 16)$ ReLU Activation Function	(16)	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(8, 127)$
Linear	(In, Out) $= (16, 8)$	(8)	**Transpose Convolution**	(Size, Stride, Padding) $= (4, 1, 1)$	$(1, 128)$

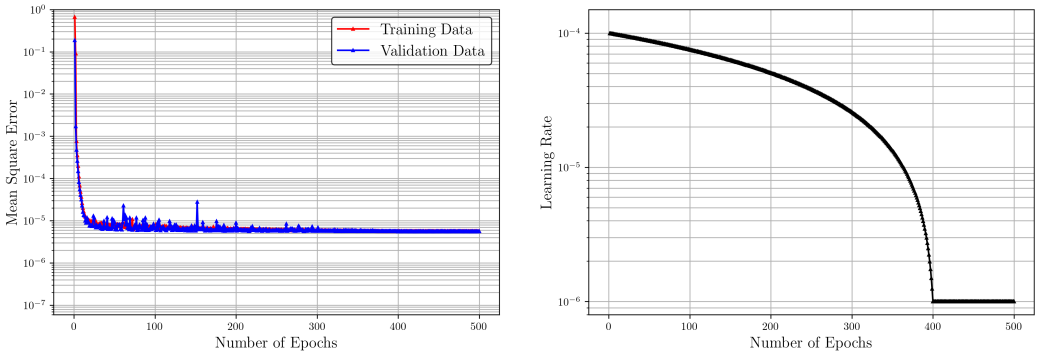


Figure A.1: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 8$ Spatial Elements

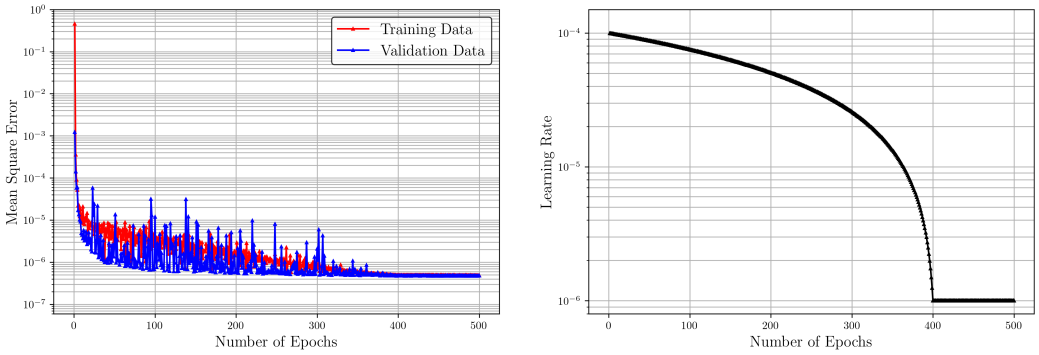


Figure A.2: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 16$ Spatial Elements

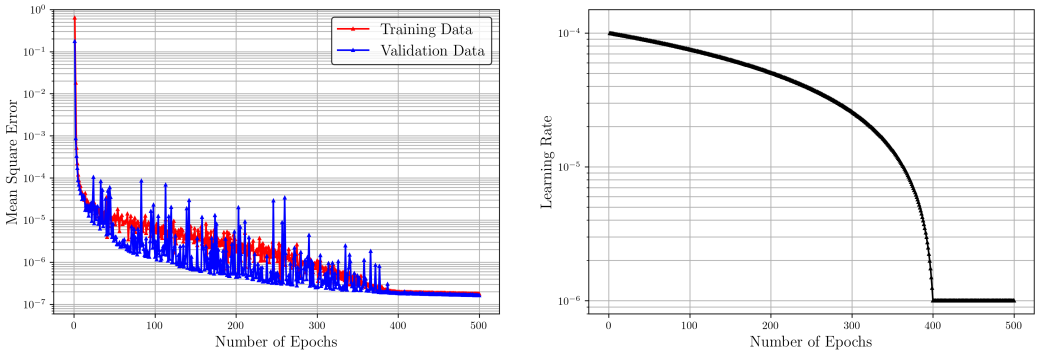


Figure A.3: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 64$ Spatial Elements

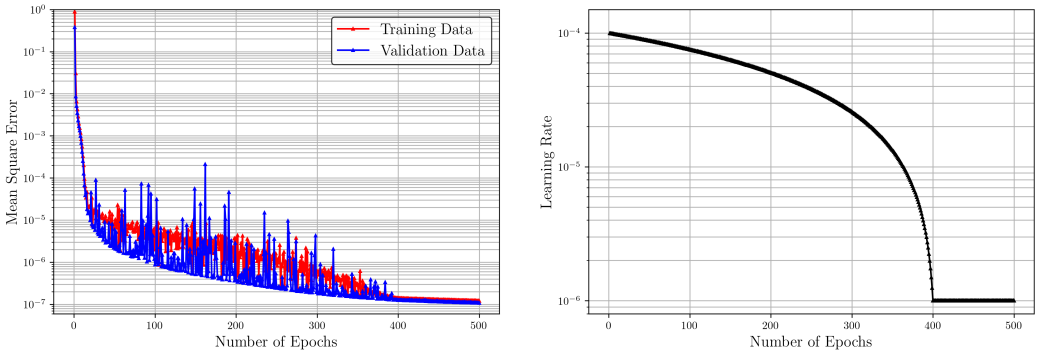


Figure A.4: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Manufactured Solution for $N_x = 128$ Spatial Elements

A.2. Unsteady 1D Viscous Burgers' Equation

Table A.5: Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 16$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 3,750			Number of Trainable Parameters: 3,865		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$ ReLU Activation Function	$(8, 15)$	**Linear**	$(\text{In}, \text{Out}) = (14, 160)$ ReLU Activation Function	(160)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(16, 8)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(16, 8)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(32, 5)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(8, 15)$
Linear	$(\text{In}, \text{Out}) = (160, 14)$	(14)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$	$(1, 16)$

Table A.6: Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 32$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 16,692			Number of Trainable Parameters: 16,921		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$ ReLU Activation Function	$(8, 31)$	**Linear**	$(\text{In}, \text{Out}) = (28, 320)$ ReLU Activation Function	(320)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(16, 16)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(32, 9)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(32, 9)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (2, 2, 1)$ ReLU Activation Function	$(16, 16)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(64, 5)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(8, 31)$
Linear	$(\text{In}, \text{Out}) = (320, 28)$	(28)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 1, 1)$	$(1, 32)$

Table A.7: Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 128$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 32,951			Number of Trainable Parameters: 33,289		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (6, 1, 2)$ ReLU Activation Function	$(8, 127)$	**Linear**	$(\text{In}, \text{Out}) = (47, 432)$ ReLU Activation Function	(432)
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (5, 2, 2)$ ReLU Activation Function	$(16, 64)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(32, 17)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 2, 2)$ ReLU Activation Function	$(32, 33)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (5, 2, 2)$ ReLU Activation Function	$(32, 33)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (5, 2, 2)$ ReLU Activation Function	$(32, 17)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (4, 2, 2)$ ReLU Activation Function	$(16, 64)$
Convolution	$(\text{Size}, \text{Stride}, \text{Padding}) = (3, 2, 1)$ ReLU Activation Function	$(48, 9)$	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (5, 2, 2)$ ReLU Activation Function	$(8, 127)$
Linear	$(\text{In}, \text{Out}) = (432, 47)$	(47)	**Transpose Convolution**	$(\text{Size}, \text{Stride}, \text{Padding}) = (6, 1, 2)$	$(1, 128)$

Table A.8: Convolutional Autoencoder Architecture of Unsteady 1D Viscous Burger's Equation for $N_y = 256$ Spatial Elements

Encoder			**Decoder**		
Number of Trainable Parameters: 36,055			Number of Trainable Parameters: 36,393		
Layer	**Parameters**	**Output**	**Layer**	**Parameters**	**Output**
Convolution	(Size, Stride, Padding) $= (6, 1, 2)$ ReLU Activation Function	$(8, 255)$	**Linear**	(In, Out) $= (47, 432)$ ReLU Activation Function	(432)
Convolution	(Size, Stride, Padding) $= (5, 2, 2)$ ReLU Activation Function	$(16, 128)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 17)$
Convolution	(Size, Stride, Padding) $= (4, 2, 2)$ ReLU Activation Function	$(32, 65)$	**Transpose Convolution**	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 33)$
Convolution	(Size, Stride, Padding) $= (5, 2, 2)$ ReLU Activation Function	$(32, 33)$	**Transpose Convolution**	(Size, Stride, Padding) $= (5, 2, 2)$ ReLU Activation Function	$(32, 65)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(32, 17)$	**Transpose Convolution**	(Size, Stride, Padding) $= (4, 2, 2)$ ReLU Activation Function	$(16, 128)$
Convolution	(Size, Stride, Padding) $= (3, 2, 1)$ ReLU Activation Function	$(48, 9)$	**Transpose Convolution**	(Size, Stride, Padding) $= (5, 2, 2)$ ReLU Activation Function	$(8, 255)$
Linear	(In, Out) $= (432, 47)$	(47)	**Transpose Convolution**	(Size, Stride, Padding) $= (6, 1, 2)$	$(1, 256)$

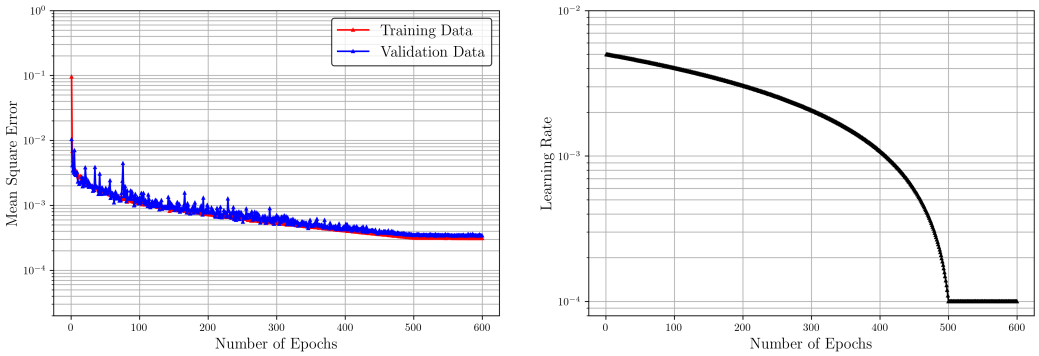


Figure A.5: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 16$ Spatial Elements

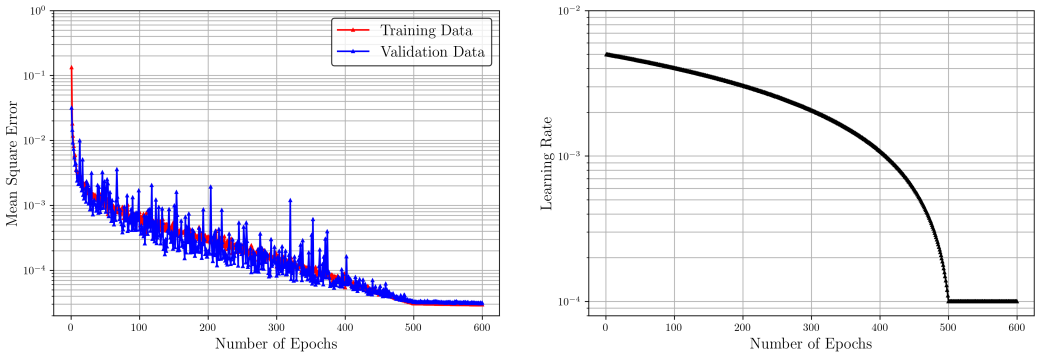


Figure A.6: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 32$ Spatial Elements

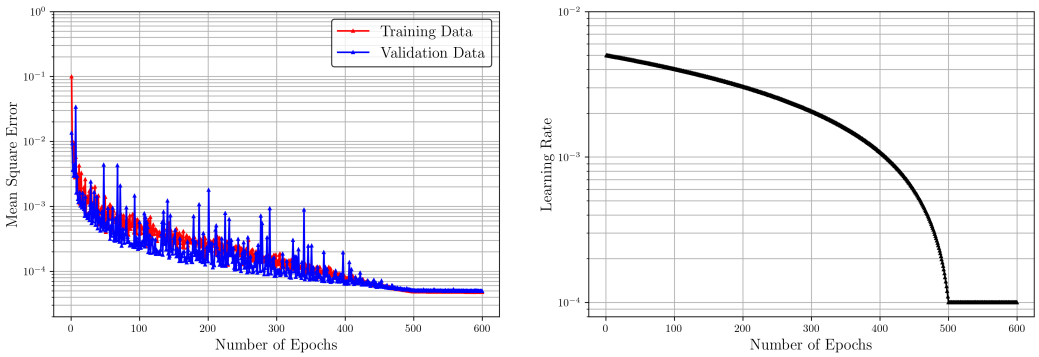


Figure A.7: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 128$ Spatial Elements

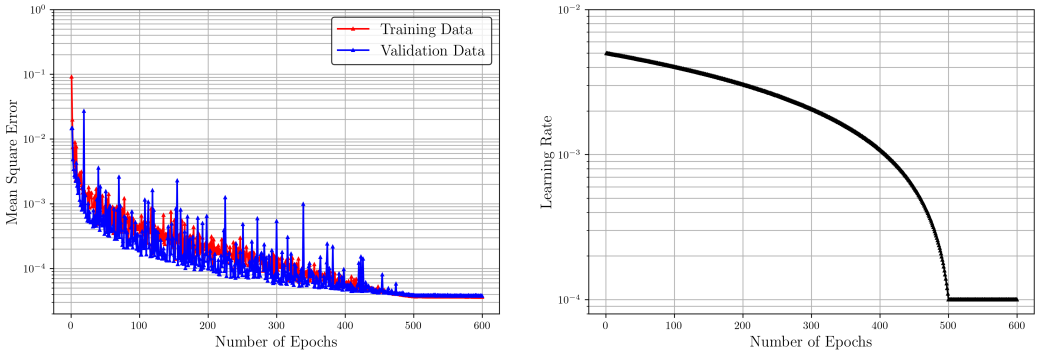


Figure A.8: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 1D Viscous Burgers' Equation for $N_y = 256$ Spatial Elements

A.3. Unsteady 2D Viscous Burgers' Equation

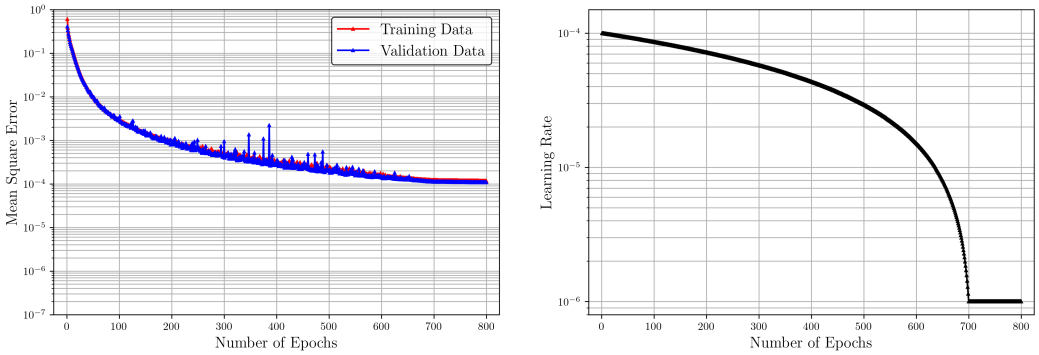


Figure A.9: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equation for $N_z \times N_y = 32 \times 16$ Spatial Elements

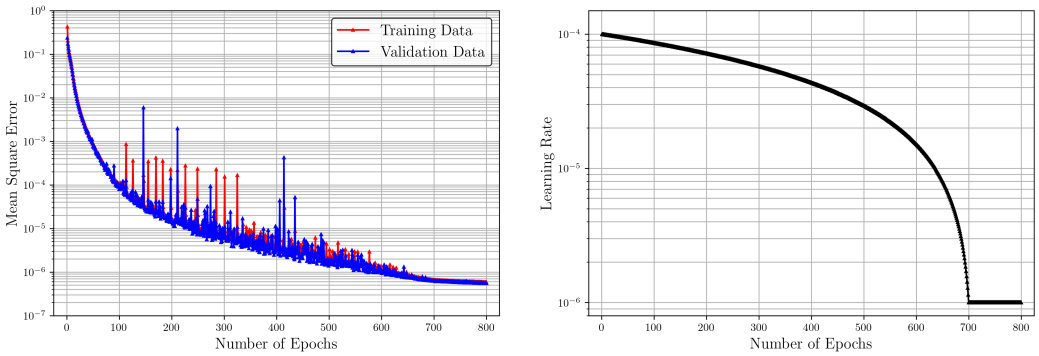


Figure A.10: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equation for $N_z \times N_y = 64 \times 32$ Spatial Elements

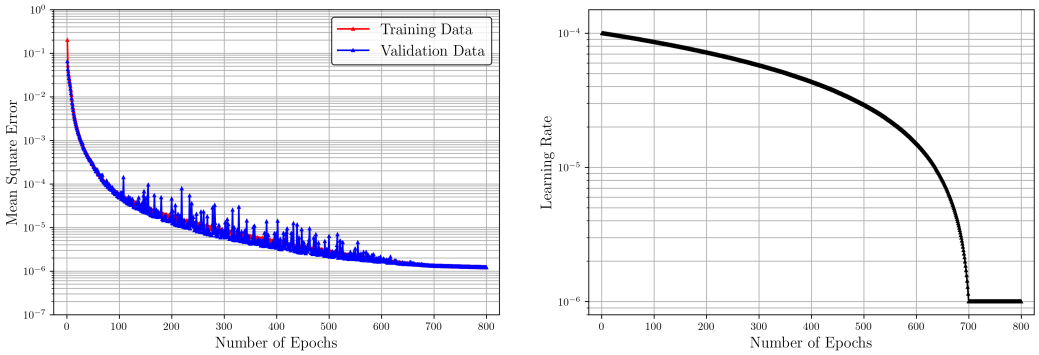


Figure A.11: Mean Square Error Loss and Learning Rate for Convolutional Autoencoder of Unsteady 2D Viscous Burgers' Equation for $N_z \times N_y = 256 \times 128$ Spatial Elements

B

Echo State Network Optimization

B.1. Manufactured Solution

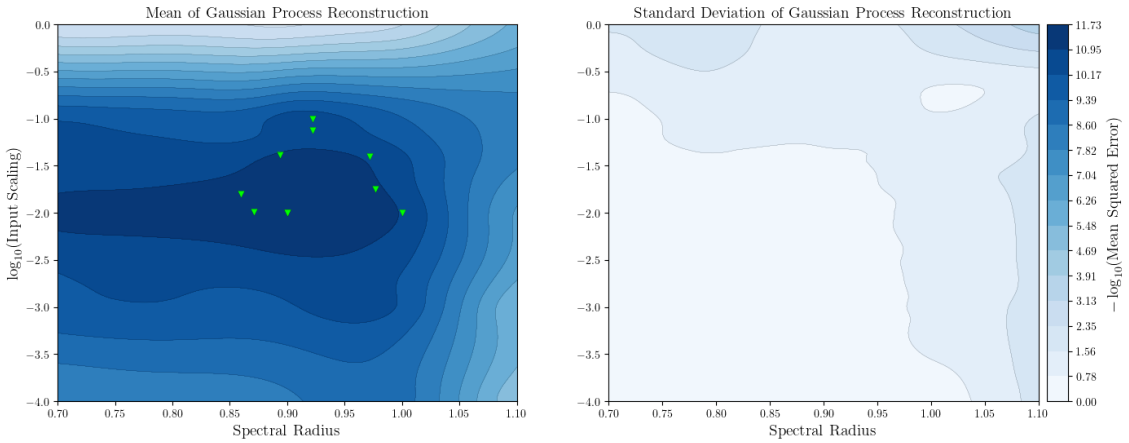


Figure B.1: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 8$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

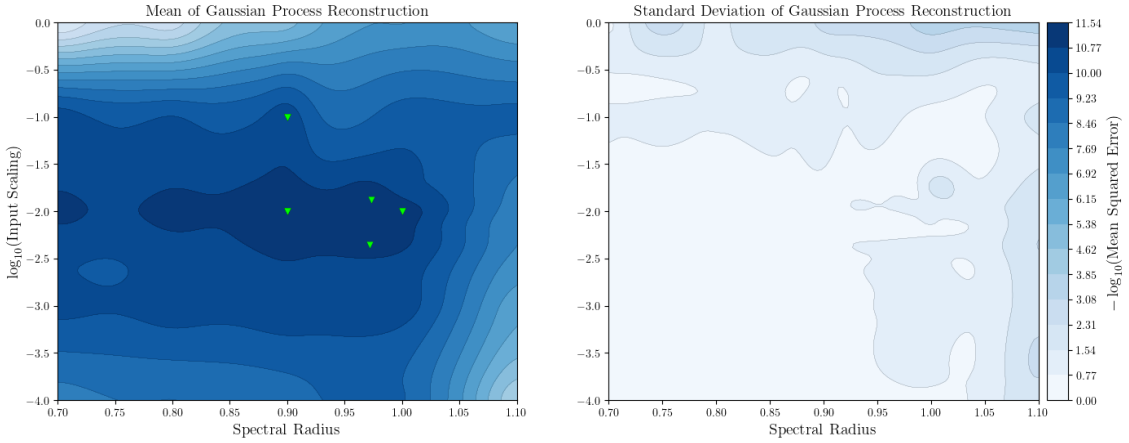


Figure B.2: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 16$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

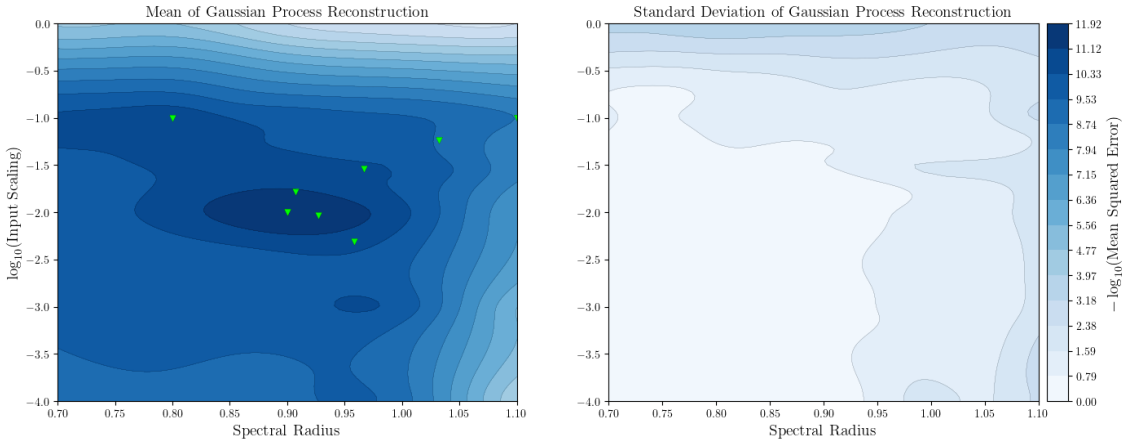


Figure B.3: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 64$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

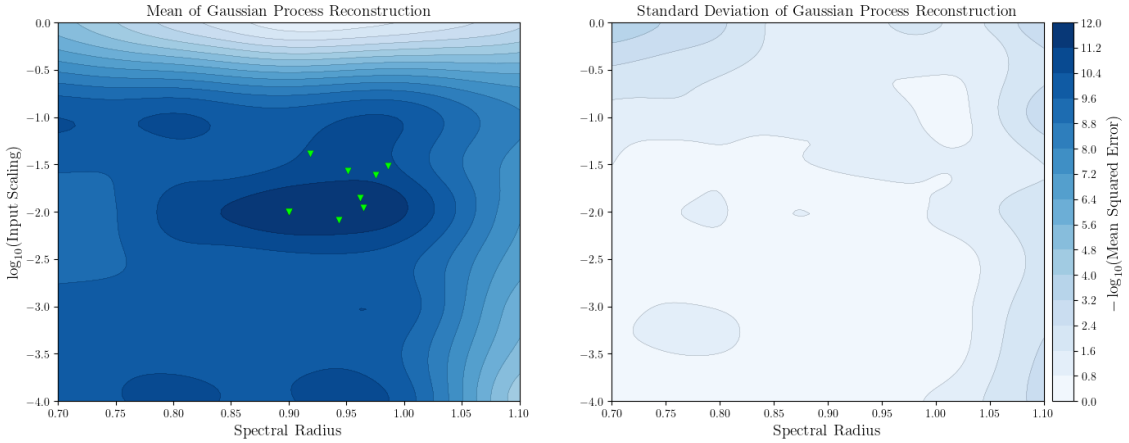


Figure B.4: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Manufactured Solution for $N_x = 128$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

B.2. Unsteady 1D Viscous Burgers' Equation

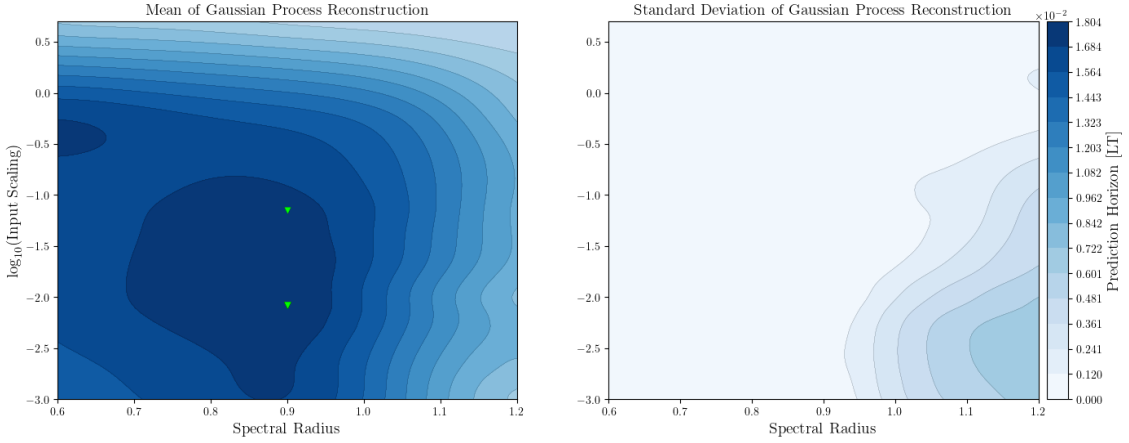


Figure B.5: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 16$ Spatial Elements and $N_r = 512$ Reservoir Neurons

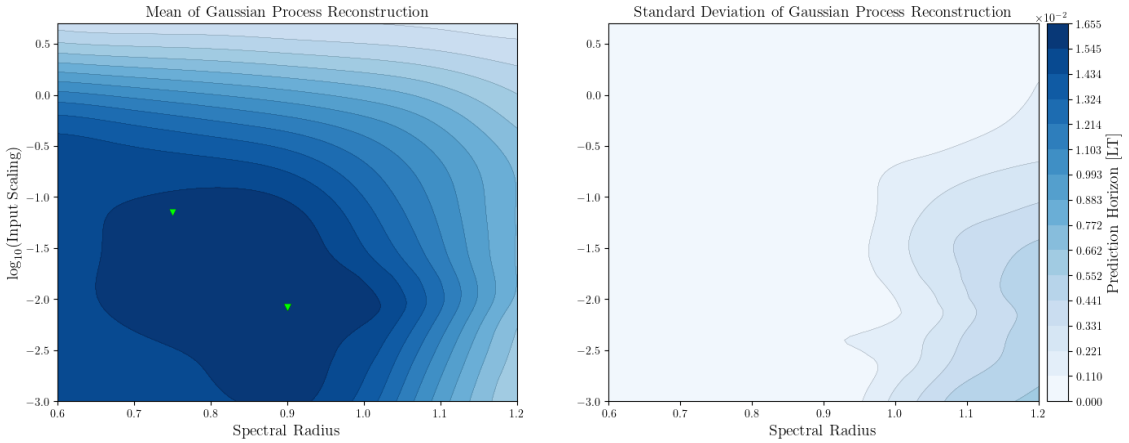


Figure B.6: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 32$ Spatial Elements and $N_r = 512$ Reservoir Neurons

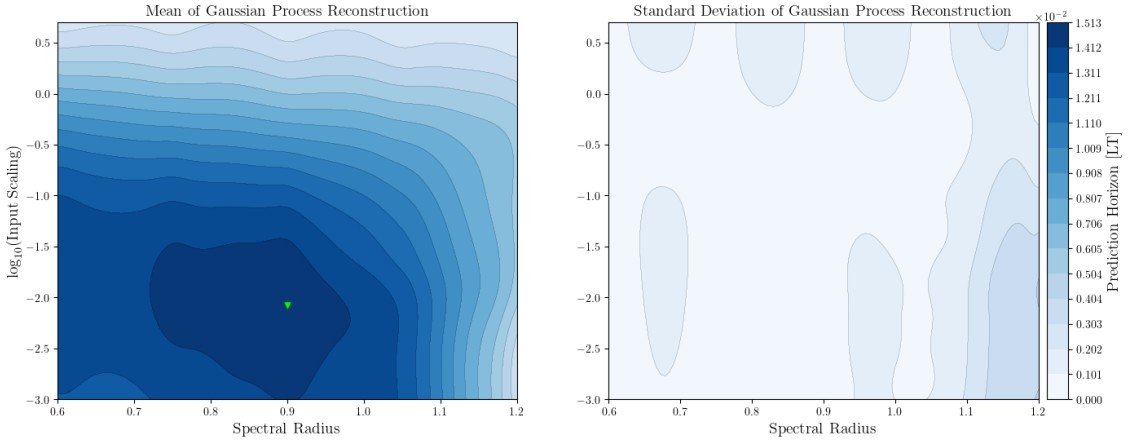


Figure B.7: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 128$ Spatial Elements and $N_r = 512$ Reservoir Neurons

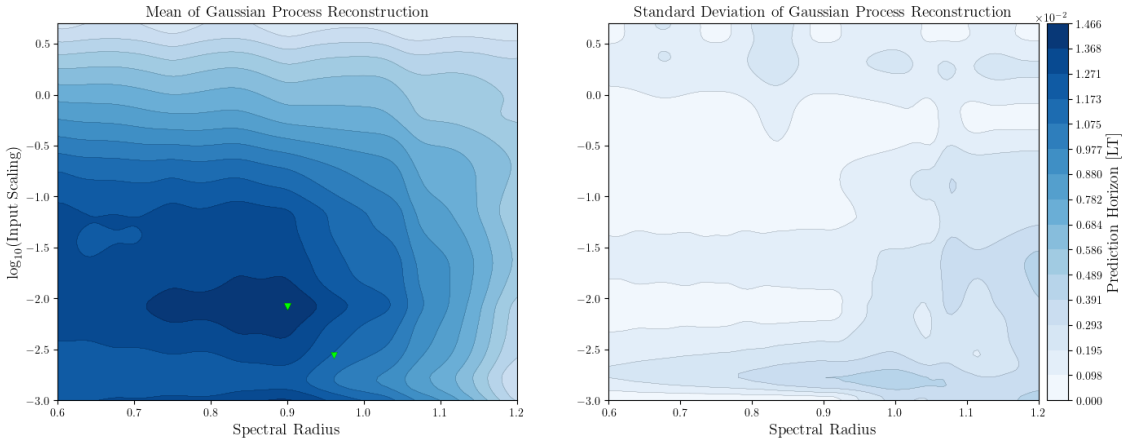


Figure B.8: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks of Unsteady 1D Viscous Burgers' Equation for $N_y = 256$ Spatial Elements and $N_r = 512$ Reservoir Neurons

C

Convolutional Autoencoder - Echo State Network Optimization

C.1. Manufactured Solution

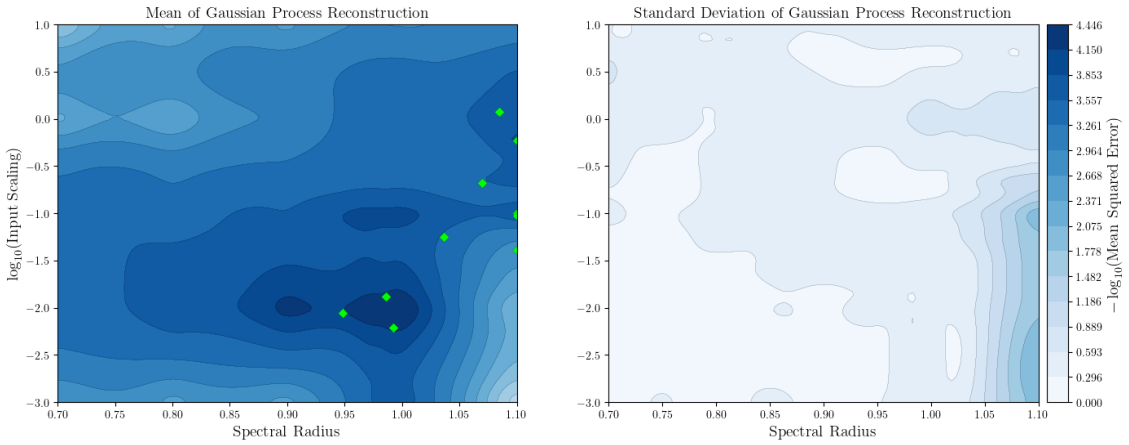


Figure C.1: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 8$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

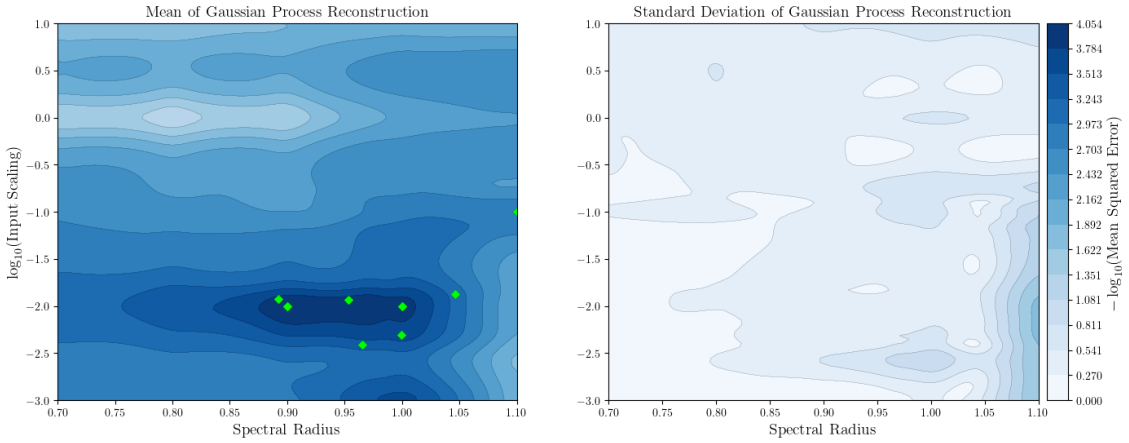


Figure C.2: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 16$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

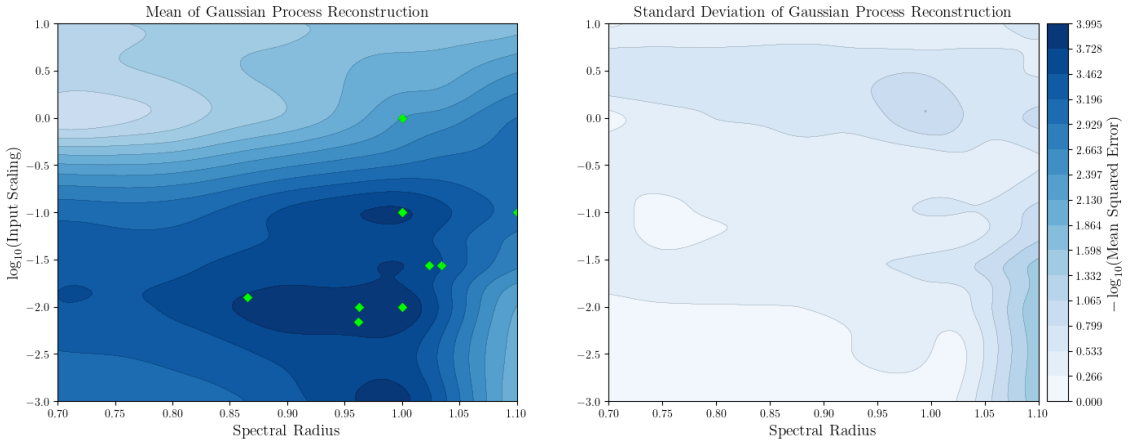


Figure C.3: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 64$ Spatial Elements and $N_r = 1024$ Reservoir Neurons

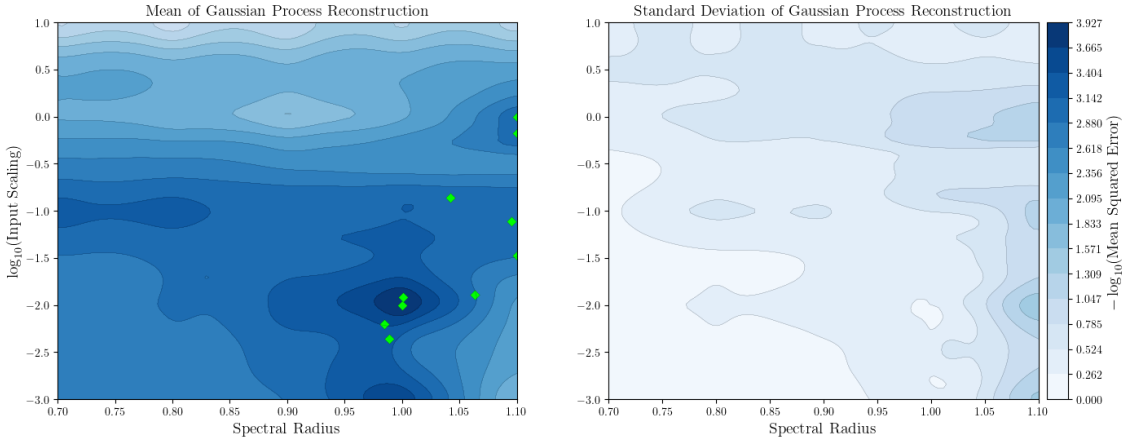


Figure C.4: Averaged Mean Square Error Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Manufactured Solution for $N_x = 8$ Spatial Elements and $N_r = 128$ Reservoir Neurons

C.2. Unsteady 1D Viscous Burgers' Equation

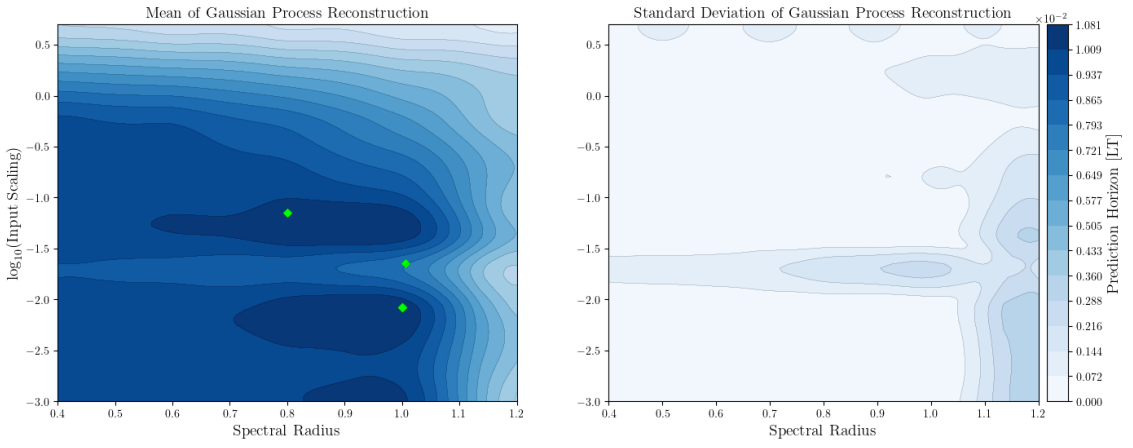


Figure C.5: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 16$ Spatial Elements and $N_r = 512$ Reservoir Neurons

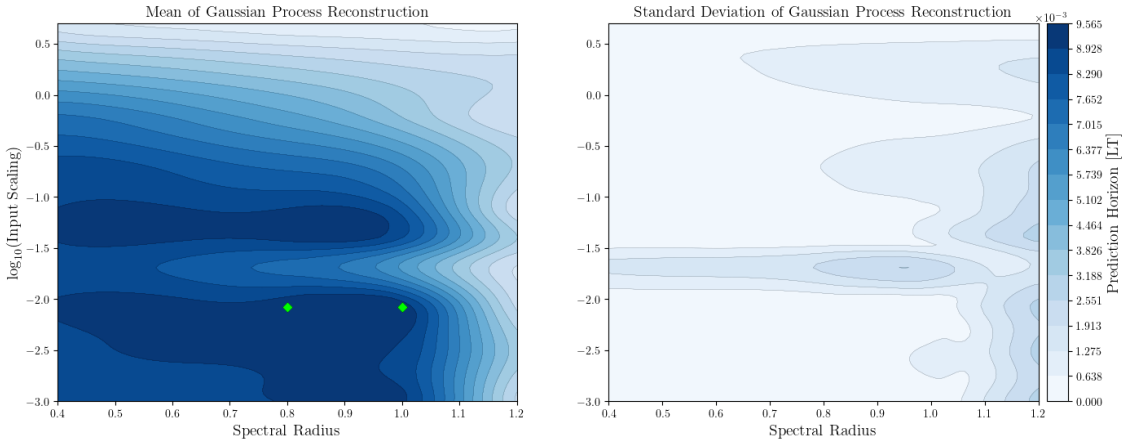


Figure C.6: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 32$ Spatial Elements and $N_r = 512$ Reservoir Neurons

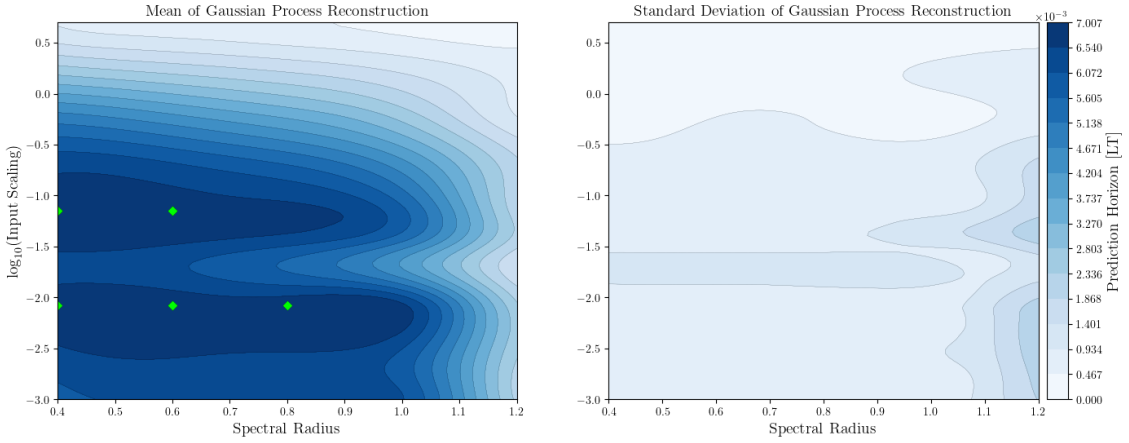


Figure C.7: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 128$ Spatial Elements and $N_r = 512$ Reservoir Neurons

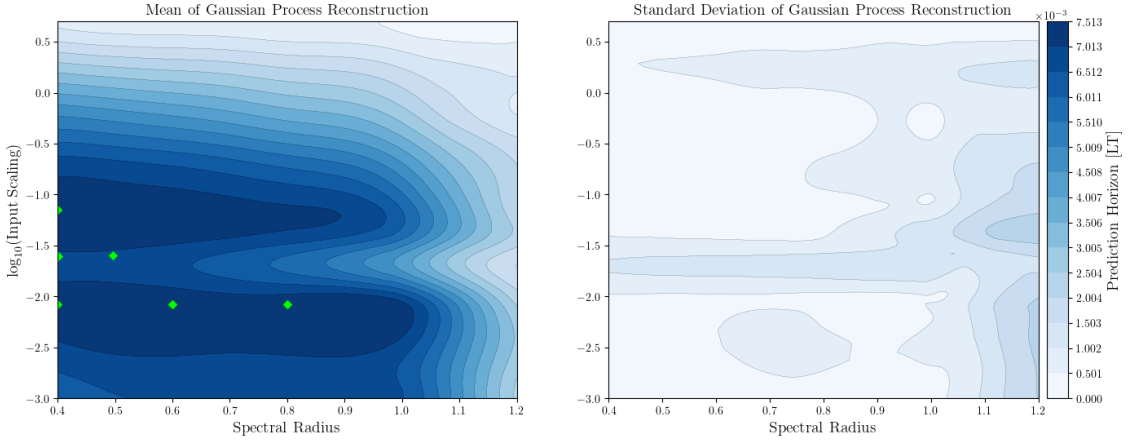


Figure C.8: Averaged Prediction Horizon ($k = 1 \times 10^{-1}$) Gaussian Process Reconstruction and Optimal Hyper-Parameters for Echo State Networks Applied to Latent Space of Unsteady 1D Viscous Burgers' Equation for $N_y = 256$ Spatial Elements and $N_r = 512$ Reservoir Neurons