Learning the Evolution of Time-dependent Partial Differential Equations



Oluwaseun Emmanuel Coker

EPSRC Centre for Doctoral Training in Fluid Dynamics

University of Leeds

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

February 2026

Intellectual Property and Publication Statements

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

This work was undertaken on ARC4 and Aire, part of the High Performance Computing facilities at the University of Leeds, UK.

The work in Chapter 4 of the thesis has appeared in a conference paper as follows:

Coker, O.E.; Jimack, P.K.; Wang, H.; Amirul, K." Scheduled Temporal Loss Weighting for Neural Operators" Machine Learning and the Physical Sciences Workshop at the Neural Information Processing Systems (NeurIPS 2025) Conference. Available from: https://ml4physicalsciences.github.io/2025

For the above paper, the candidate was responsible for conceptualising and implementing the method, and analysing the results. The candidate also led the writing, including both the original draft and subsequent revisions. The other authors contributed to the review and editing of the manuscript.

This thesis copy has been supplied on the understanding that it is copyright material, and that no quotation from the thesis may be published without proper acknowledgement. The right of Oluwaseun Emmanuel Coker to be identified as Author of this work has been asserted by her in accordance with the Copyright, Designs and Patents Act 1988.

© 2026 The University of Leeds and Oluwaseun Emmanuel Coker.

Acknowledgements

My PhD journey began with a last-minute application. I was actually getting ready for bed when I realised the deadline was at midnight, only two hours away. The decision to stop everything and complete the form set me on an incredible and rewarding path—one that I will cherish forever. I wrote my personal statement in under one hour, something I look back on as slightly wild, but at the time, the words flowed effortlessly because my motivation was clear and simple: *the quest for more knowledge.*

I must begin by thanking my supervisors, **Peter Jimack**, **He Wang**, and **Amirul Khan**, for guiding me through every phase of this research. **Peter**, you are incredibly thoughtful, kind, and wise; you always have the right words to both encourage and challenge me. **He**, thank you for your insightful ideas and for complementing the team with your machine learning expertise. **Amirul**, thank you for your support and the many interesting discussions in our meetings. I will certainly miss our tangential conversations regarding the direction and impact of Artificial Intelligence (AI).

I am immensely grateful to the EPSRC Centre for Doctoral Training (CDT) in Fluid Dynamics. **Claire Savy** is a true pillar of the CDT; thank you for always having the answers to my administrative queries, but more importantly, thank you for being so approachable and happy to chat, especially when I was frustrated with my work. **Yatin**, your positive attitude towards life is infectious, and I am grateful to have sat next to you. A special thank you goes to my CDT Cohort, whose friendship made this journey truly enjoyable. I will always cherish our "number types": **Kasia** (Pi), **Rhiannon** (Polar coordinate), **Cristina** (Simply number 10), **Jo** (Recurring number), **Mostafa** (Integer), **Meg** (Imaginary number), **Usamah** (Floating point), **Jose** (Exponential number), **Ahmad** (Prime number), and **Abhi** (Significant figure) and **Luke** (Complex number). While writing this, I am still not sure why I was labelled an Irrational number, but if I were to summarise our cohort with one number, it would be "Real".

My faith is a central part of this journey, and I have been truly blessed to be part of **Christ Church Central Leeds (CCCL)**. **Jonty**, your teaching is challenging and encouraging. **Nick**, I hope to emulate your service to the church and your family. **Tom S**, I thoroughly enjoyed our Sunday conversations on the latest developments in AI. **Mervyn** and **Jonathan**, I am grateful for our fellowship over lunch on campus. To everyone I have shared a home group with, the **Leaches**, the **Ilotts**, the **Algeos**, the **Elliotts**, the **Pickles**, **Anastasia**, **Liz**, **Shen**, **James**, **Luke**, **Fang Yi**, **Pan**, and **Brooke**, thank you for your fellowship and encouragement. Home group has been a major highlight over the last four years.

I must also thank **Djami**, **Princess**, and **Lanre**, whom I met during my first few weeks in Leeds; your friendship has brought so much joy to these last four years. **Djami**, you are thoughtful and kind; **Princess**, you are resourceful and caring; **Lanre**, you are hardworking and positive-minded.

I am incredibly indebted to my **parents**, who have always prioritised my education above their own interests. I would not be where I am today without their daily sacrifices. You inspire me to live a life of service, putting the needs of others above my own. Words cannot describe how grateful I am to you. To my sister, **Busola**, thank you for navigating life with me; I will always treasure our time spent reminiscing about our childhood. To my dear friends, **Timilehin**, **Olisa**, and **Femi**, you have been a constant source of encouragement and I deeply value our brotherhood.

To my dear wife, **Lisa**, when I think of my PhD journey, you are central to it. Thank you for your endless patience. You listened to my frustrations and provided the emotional support that kept me going. I am so lucky that I get to spend the rest of my life with you. I love you so much.

Above all, I praise the **Lord my God**. It has been a profound honour to study the world You created. I am humbled that You have created us with the ability to comprehend such complex patterns and mysteries. Thank You for Your providence throughout this journey, guiding my understanding and sustaining me through every challenge of this PhD.

Abstract

This thesis investigates the challenges of using deep learning models, specifically **Neural Operators**, to predict the long-term temporal evolution of complex physical systems governed by Partial Differential Equations (PDEs). While deep learning surrogate models can offer significant computational speedups over traditional numerical solvers, they frequently suffer from *autoregressive error accumulation* (exposure bias) and a lack of *temporal flexibility* when constrained to fixed timesteps during inference.

This research addresses these limitations through three primary contributions. First, we develop **Scheduled Temporal Loss Weighting (STLW)**, a novel curriculum learning strategy for constant-timestep models. STLW dynamically adjusts loss weights across a temporal rollout during training, prioritising short-term accuracy initially and gradually increasing the influence of long-term gradients. Experimental results demonstrate that STLW significantly reduces error propagation, achieving up to a 62.5% reduction in error for the viscous Burgers' equation compared to standard baselines.

Second, we introduce an **attention-based temporal conditioning mechanism** to break the dependence on fixed time-stepping. By integrating cross-attention with temporal embeddings, a single Fourier Neural Operator (FNO) can predict at variable timesteps. Third, this work demonstrates the feasibility of **variable-timestep sampling strategies**, establishing that "large-to-small" or "small-to-large" stepping can effectively bypass high-error phases in PDE evolution depending on the physical regime.

The proposed methods are rigorously validated across a diverse suite of benchmark PDEs, including Advection, Viscous and Inviscid Burgers', Diffusion-Reaction, Diffusion-Sorption, and Korteweg-de Vries (KdV) equations. Overall, the results demonstrate that combining architectural flexibility with advanced training strategies can yield superior global stability and generalisation in long-term prediction.

Abbreviations

AR Autoregressive

AR-CL Autoregressive Curriculum Learning

Av Advection

CL Curriculum Learning

DeepONet Deep Operator Network

DR Diffusion-Reaction Equation

DS Diffusion-Sorption Equation

EL Embedding Layer

F-FNO Factorised Fourier Neural Operator

FNO Fourier Neural Operator

iB Inviscid Burgers'

KdV Korteweg-de Vries

MLP Multi-Layer Perceptron

MSE Mean Squared Error

nRMSE Normalised Root Mean Squared Error

ODE Ordinary Differential Equation

PDE Partial Differential Equation

STE Sinusoidal Temporal Embedding

STLW Scheduled Temporal Loss Weighting

TF Teacher Forcing

U-NO U-shaped Neural Operator

vB Viscous Burgers'

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Challenges in Long-term Prediction . . . 2
1.3 Aims and Objectives . . . 3
1.4 Thesis Outline . . . 5

2 Literature Review **8**
2.1 Time-Dependent Partial Differential Equations . . . 9
2.2 Classical Numerical Solvers and Their Limitations . . . 11
2.3 Deep Learning for Partial Differential Equations . . . 12
2.4 Open Challenges for Learned PDE Solvers . . . 13
2.5 Physics-Informed Machine Learning . . . 15
2.5.1 Observational bias . . . 15
2.5.2 Learning bias . . . 16
2.5.3 Inductive bias . . . 16
2.6 Neural Operators for PDEs . . . 17
2.6.1 Deep Operator Networks (DeepONets) . . . 18
2.6.2 Neural Operators . . . 18
2.6.3 Neural Operator vs DeepONet . . . 20
2.6.4 Beyond Fourier Neural Operator (FNO) . . . 22
2.6.5 Graph-Based Approaches . . . 23
2.7 Temporal Modelling for PDE Evolution . . . 24
2.8 Summary and Research Gap . . . 25

3 Methods **27**
3.1 Problem Definition . . . 27
3.1.1 Operator Learning . . . 27
3.1.2 Autoregressive (Time-stepping) Formulation . . . 29

3.2 Fourier Neural Operator (FNO) . . . 30
3.2.1 Architecture . . . 30
3.2.2 Implementation . . . 32
3.3 Variants of FNO . . . 33
3.3.1 FNO-Decompose . . . 33
3.3.2 Factorised F-FNO . . . 33
3.4 Other Architectures . . . 34
3.4.1 U-NET . . . 34
3.4.2 U-shaped Neural Operator (U-NO) . . . 35
3.5 Code Implementation . . . 36
3.6 Partial Differential Equations . . . 36
3.6.1 Advection Equation (Av) . . . 37
3.6.2 Viscous Burgers' Equation (vB) . . . 37
3.6.3 Inviscid Burgers' Equation (iB) . . . 38
3.6.4 Diffusion-Reaction Equation (DR) . . . 38
3.6.5 Diffusion-Sorption (DS) . . . 39
3.6.6 Korteweg-De Vries Equation (KdV) . . . 39
3.7 Training Approach . . . 40
3.8 Metrics . . . 40
3.9 Baseline FNO Characterisation . . . 41
3.9.1 Training objective and baseline configuration . . . 41
3.9.2 Single-step mapping to autoregressive prediction . . . 42

4 Curriculum Learning for Constant-timestep Models 44
4.1 Introduction . . . 44
4.2 Problem Statement . . . 48
4.3 Related Work . . . 49
4.3.1 Long-term Prediction . . . 49
4.3.2 Neural Operators . . . 49
4.3.3 Curriculum Learning . . . 50
4.4 Baseline Training Strategies . . . 50
4.4.1 Gradient propagation during training . . . 52
4.4.2 Baseline Rollout Analysis: The Stability Gap . . . 54
4.5 Curriculum-based training . . . 59
4.5.1 Scheduler: Epoch-level . . . 60
4.5.2 Metric: Batch-level . . . 61
4.5.3 AR Curriculum . . . 64
4.6 Scheduled Temporal Loss Weighting (STLW) . . . 64
4.6.1 Implementation . . . 65
4.6.2 Curriculum Metric . . . 65

4.6.3 Weighting Transition function . . . 67
4.6.4 Training Algorithm . . . 69
4.6.5 Gradient Propagation . . . 70
4.6.6 STLW as a Learning Rate Scheduler . . . 71
4.6.7 STLW as a Continuous AR Curriculum . . . 72
4.7 Experiments . . . 73
4.7.1 Training Strategies . . . 74
4.7.2 Parameterisation of CL Strategies . . . 74
4.7.3 Datasets . . . 76
4.7.4 Training Procedure . . . 76
4.7.5 Analysis of Training Dynamics . . . 77
4.8 Results and Discussion . . . 80
4.8.1 Baseline Performance Analysis . . . 80
4.8.2 Comparative Effectiveness of Curriculum Learning . . . 80
4.8.3 Long-Term Accuracy and Stability . . . 82
4.8.4 Error Propagation Analysis . . . 82
4.8.5 Impact of Training Rollout Length . . . 83
4.8.6 Inference Rollout Analysis . . . 85
4.8.7 STLW as a Multi-Dimensional Learning Rate Scheduler . . . 86
4.8.8 Hyperparameter Analysis: Batch Size and Rollout Dynamics 88
4.8.9 Generalisability Across Architectures . . . 90
4.8.10 Sensitivity Analysis of the STLW Parameter . . . 92
4.9 Discussion . . . 95
4.10 Conclusion . . . 97

5 Attention Mechanism for Variable-timestep Models 99
5.1 Introduction . . . 99
5.2 Problem Statement . . . 100
5.3 Related Work . . . 102
5.3.1 Variable-timestep Modelling . . . 102
5.3.2 Neural Operators . . . 102
5.3.3 Attention Mechanisms . . . 102
5.4 Time Conditioning . . . 104
5.5 Time Embedding . . . 105
5.5.1 Non-learnt Time Embedding . . . 106
5.5.2 Learnt Time Embedding . . . 106
5.6 Attention-Based Time Conditioning . . . 108
5.6.1 Input Lifting and Embedding . . . 108
5.6.2 The Attention Mechanism . . . 109
5.6.3 Spectral Convolution and Output Projection . . . 110

5.7 Experiments . . . 112
5.7.1 Variable-timestep Sampling . . . 114
5.8 Results . . . 116
5.8.1 Experimental Configuration and Metrics . . . 116
5.8.2 Analysis of Temporal Evolution . . . 117
5.8.3 Advection . . . 118
5.8.4 Viscous Burgers . . . 123
5.8.5 Diffusion-Reaction . . . 129
5.8.6 Diffusion-Sorption . . . 134
5.8.7 Effect of model size . . . 139
5.8.8 Comparative Performance: Variable-timestep vs. Constant-timestep Models . . . 139
5.8.9 Advection: Time-dependent speed parameter . . . 148
5.8.10 Comparison with a Transformer-based Neural Operator . . . 155
5.8.11 Variable-timestep Model with Curriculum Learning . . . 157
5.9 Discussion . . . 159
5.9.1 Efficacy of Temporal Conditioning Mechanisms . . . 159
5.9.2 The Generalisation Penalty vs. Exposure Bias . . . 159
5.9.3 Variable-timestep model: Error Dynamics . . . 160
5.9.4 Analysis of Non-Stationary Systems . . . 162
5.9.5 Structural Stability and Scaling Laws . . . 162
5.9.6 Summary of Stability and Propagation . . . 162

6 Conclusion and Future Work **163**
6.1 Summary of main results . . . 163
6.1.1 Curriculum Learning for Constant-Time model . . . 163
6.1.2 Attention Mechanism for Variable-timestep model . . . 164
6.2 Variable-timestep Model with Curriculum Learning . . . 165
6.3 Thesis Contributions . . . 165
6.3.1 A Novel Training Strategy: Scheduled Temporal Loss Weighting (STLW) . . . 166
6.3.2 Novel Model Architecture: Attention-Based Temporal Conditioning . . . 166
6.3.3 Synthesis and Performance Correlation . . . 167
6.3.4 Experimental Validation and Open-Source Contribution . . . 167
6.4 Future Work . . . 168
6.4.1 Curriculum Learning for Constant-timestep Model . . . 168
6.4.2 Attention Mechanism for variable-timestep models . . . 168

References **182**

A Fourier Analysis **183**
A.1 Fourier Series and Transform 183
A.2 Properties of the Fourier transform 185
A.2.1 Derivatives of functions 185
A.2.2 Linearity 185
A.2.3 Parseval's Theorem 186
A.2.4 Convolution 186

List of Figures

3.1 Operator learning. $\mathcal{A}$ and $\mathcal{U}$ denote the spaces of admissible input (initial-condition) and output (solution) functions, respectively. The input $a \in \mathcal{A}$ is a function on the spatial domain $D \subset \mathbb{R}^d$, while the output $u \in \mathcal{U}$ is the corresponding solution on $[0, T] \times D$; each represents an individual function drawn from its respective function space. 28

3.2 (top) Fourier Neural Operator architecture. (bottom) composition of a single FNO layer. Taken from Kovachki *et al.* (2021). 32

4.1 Baseline training strategies of $\widehat{T}$ timestep: Teacher forcing (one-rollout) and autoregressive rollout (multi-rollout), where T represents the total timesteps of the data and $\widehat{T}$ represent the training batch timesteps. 52

4.2 Comparative analysis of rollout MSE across different neural operator architectures as a function of temporal horizon. The plot illustrates the scaling behaviour of the FNO, U-NO, and LSM models trained with equal input (history) and output (prediction) windows of lengths 1, 25, 50, and 125 steps. Note the superior performance of the FNO model, which maintains higher stability and lower error at longer horizons compared to U-NO and LSM. 59

4.3 Transition functions at the epoch scale: Analytical functions for $C(e)$, showing increasing (solid line) and decreasing (dashed line) variants. 61

4.4 Combinations of Teacher Forcing (TF) and Autoregressive Rollout (AR) across training epochs. 63

4.5 STLW: A schematic of the Scheduled Temporal Loss Weighting (STLW) Curriculum during training. The colour strength ranges from 0 (low) to 1 (high), indicating the weight assigned to each rollout during training. Later rollouts in the early training phase have low strength, whereas these weights gradually increase in magnitude. The weights are applied to the loss of its respective timestep and epoch. 66
4.6 A schematic of the scheduled temporal loss Weighting (STLW) Curriculum. For a dataset with a subsequence of length T ($T = 6$), a weight w is assigned to each timestep for $t > 0$, which controls the loss l contributed by each timestep (for example, at epoch = 60, the weight for each timestep is highlighted in black markers). The weighting function (curves), which determines the weight values, changes during training, and at the end of training (epoch = 100), the weighting function sets all weights to 1. 69
4.7 Schematic comparison between the discrete AR-CL and the Scheduled Temporal Weight Loss (STWL) curriculum (right). STWL acts as a continuous variant, with a weight applied at every timestep, represented by the colour intensity of the bars. 73
4.8 Training dynamics of baseline strategies. The training (dashed line) and validation (solid line) losses are shown for each strategy. Shaded areas represent the standard deviation. 78
4.9 Training dynamics of curriculum strategies compared to the AR baseline. STLW demonstrates enhanced stability and lower long-term error propagation. 79
4.10 Error propagation over time for the best baseline and the top two curriculum strategies for: (a) Advection (Av), (b) Viscous Burgers (vB), (c) Inviscid Burgers (iB), and (d) Korteweg–De Vries (KdV). 83
4.11 Effect of increasing training rollout length $\widehat{T}$ on the nRMSE of the best baseline and the top two curriculum strategies across the four PDE datasets. 84
4.12 Training dynamics under fixed and variable (scheduled) learning rates (η) across different curricula using the Adam optimiser. Dashed lines indicate training loss; solid lines indicate validation loss. . . . 88
4.13 Sensitivity of the A_c parameter for the viscous Burgers' (vB) equation. Horizontal lines indicate the baseline errors for AR and AR-CL. 93
4.14 Sensitivity of the A_c parameter for the KdV equation. The stability across the x-axis demonstrates the robustness of the weighting strategy. 94

5.1 Schematic of the standard Fourier Neural Operator (FNO) architecture featuring baseline temporal conditioning. The diagram illustrates the lifting of the input function u_t into a higher-dimensional space, followed by iterative layers where the time step Δt is integrated via one of the two baseline conditioning mechanisms, (a) addition or (b) concatenation. Each layer performs a spectral convolution in the frequency domain and a local linear transformation in the spatial domain, before a final projection to yield the prediction $u_{t+\Delta t}$. 105

5.2 Schematic of the Fourier Neural Operator (FNO) architecture integrated with temporal embedding modules. Unlike simple scalar conditioning, this model employs learnt or non-learnt embedding functions, $E_{\phi,t}$ and $E_{\phi,t+1}$, to transform the raw input and output timestamps into continuous vector representations, $\hat{t}$ and $\widehat{t+1}$. These embeddings are then incorporated into the spectral layers to guide the evolution of the hidden feature maps from the current state u_t to the predicted state $u_{t+\Delta t}$. 107

5.3 Schematic of the proposed Attention-based Fourier Neural Operator for variable-timestep prediction. The architecture processes inputs through three parallel streams: (i) a spectral convolution for frequency-domain patterns, (ii) a spatial convolution (1×1) acting as a learnable bias, and (iii) an attention mechanism that uses lifted temporal embeddings (T_t, T_{t+1}) to modulate the function embedding U_t. The outputs are summed and passed through a non-linear activation σ to produce the next hidden state U_{t+1}, which is projected back to the physical space u_{t+1}. 111

5.4 **Training convergence for the Viscous Burgers' equation over 200 epochs**. Each curve represents the mean Normalised Root Mean Squared Error (nRMSE) for one of the six strategies outlined in Table 5.1. The consistent downward trend and stabilisation across all strategies indicate robust convergence and provide a reliable baseline for the variable-timestep sampling evaluations. 114

5.5 Temporal error propagation for constant-timestep sampling ($\Delta t \in \{1, \ldots, 5\}$). A performance "elbow" is observed at $\Delta t > 2$, where the error increases significantly. 121

5.6 Impact of **increasing** variable-timestep sampling strategies on prediction accuracy. Strategies starting with small timesteps (e.g., 1-2 and 1-2-3) show improved stability over the constant $\Delta t = 1$ baseline, whereas strategies starting at larger intervals (2-3-4) suffer from high initial error injection. 121

5.7 Impact of **decreasing** variable-timestep sampling strategies. The results demonstrate a "correction" effect, in which transitioning from a large to a smaller timestep (e.g., 3-2-1) mitigates cumulative error more effectively than maintaining a constant large timestep. 122
5.8 Cumulative error growth during the autoregressive rollout for the Advection equation. The near-linear accumulation of error across all models reflects the system's non-dissipative nature, in which local prediction inaccuracies are advected forward without damping. 122
5.9 MLP-At temporal error propagation for constant timesteps. The non-linear error curves reflect the physical difficulty of capturing early-stage shock formation, with larger timesteps ($3\Delta t$) showing better long-term stability due to reduced rollouts. 126
5.10 Performance of **increasing** variable-timestep steps. While 1-2 and 1-2-3 improve upon the constant $1\Delta t$ baseline, they are still hindered by high error in the early timesteps. 127
5.11 Performance of **decreasing** variable-timestep steps. Decreasing timesteps leads to worse performance because the smooth solutions are more susceptible to error propagation. 127
5.12 Cumulative error sum for Burgers' equation. The higher error in the 1-2 sampling compared to the 2-1 sampling demonstrates the impact of the sampling strategy. 128
5.13 Temporal error propagation for constant-timestep sampling. The error increases over time as reaction-driven complexities accumulate, with the $1\Delta t$ model suffering the most from autoregressive error propagation. 131
5.14 Increasing variable-timestep sampling: Strategies like 1-2 and 1-2-3 improve upon the $1\Delta t$ baseline by transitioning to fewer, larger steps as the physical system's error sensitivity increases. 132
5.15 Decreasing variable-timestep sampling: The 3-2-1 and 2-1 sampling increase errors over the constant $3\Delta t$ baseline, as the introduction of smaller timesteps increases the total number of rollout steps and the associated exposure bias. 133
5.16 Cumulative error sum. Although 3-2-1 begins with lower error, the 1-2-3 strategy achieves a lower final cumulative total by using larger timesteps to "bypass" frequent sampling during the high-error later stages of the PDE evolution. 133
5.17 Temporal error propagation for constant timesteps. The high error at early timesteps highlights the difficulty of capturing initial sorption dynamics, with error profiles converging as the system stabilises. 136

5.18 Performance of **increasing** variable-timestep sampling. Variable-timestep sampling does not improve over constant-timestep sampling. 137
5.19 Performance of **decreasing** variable-timestep sampling. Variable-timestep sampling does not improve over constant-timestep sampling. 137
5.20 Cumulative error for Diffusion-Sorption. The higher error in the 1-2 1-2-3 sampling compared to 2-1 and 3-2-1 illustrates how a larger timestep during the high-error early phase can reduce overall error propagation. 138
5.21 Scaling: The rate of error reduction with an increase in the number of training samples and model dimension. 143
5.22 Advection: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable-timestep model evaluated at $1\Delta t$ (left). The data sample with the highest error is shown. 144
5.23 Viscous Burgers: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable model evaluated at $1\Delta t$ (left). Data sample with the smallest error. . . . 145
5.24 Diffusion-Reaction: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable model evaluated at $1\Delta t$ (left) 146
5.25 Diffusion-Sorption: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable model evaluated at $1\Delta t$ (left) 147
5.26 Temporal error propagation (nRMSE) for the advection equation with a **discontinuous step** speed function. The secondary axis (dashed black line) indicates the time-dependent speed $\beta(t)$, which shifts abruptly at $t \approx 66$ and $t \approx 134$. The plot illustrates how constant-timestep models ($1, 2$, and $3\Delta t$) experience sharp increases in error at these discontinuities. 149
5.27 Total nRMSE relative to the total number of timesteps (t_s) for the **discontinuous (step)** speed function. The plot compares constant-timestep sampling ($\Delta t = 1, 2, 3, 4, 5$) against variable-timestep sampling strategies across a range of computational budgets. The variable-timestep curve demonstrates a superior efficiency-to-accuracy ratio, consistently achieving lower error than its constant-timestep counterparts across most rollout counts, particularly at both low and high sampling rates. 150
5.28 Error propagation (nRMSE) for the advection equation with a **linear-increasing** speed function. The secondary axis (dashed black line) indicates the accelerating speed parameter $\beta(t)$. 151

5.29 Error per timestep for the **linear-increasing** advection speed. The plot illustrates the effectiveness of a 3-2 variable-sampling strategy in stabilising error growth. As the wave velocity $\beta(t)$ increases, the transition from a coarser resolution ($3\Delta t$) to a finer resolution ($2\Delta t$) at $t = 125$ effectively caps the rising error profile, maintaining a lower and more consistent error per step than the constant $3\Delta t$ baseline. 152
5.30 Error propagation (nRMSE) for the advection equation with a **linear-decreasing** speed function. The secondary axis (dashed black line) represents the decelerating speed parameter $\beta(t)$. . . . 153
5.31 Error per timestep for the **linear-decreasing** advection speed. The plot illustrates the effectiveness of a 2-3 variable-sampling strategy. By transitioning from a finer resolution ($2\Delta t$) during the high-velocity initial phase to a coarser resolution ($3\Delta t$) at $t = 100$ as the wave decelerates, the model maintains structural fidelity while reducing the total number of required recursions. 154

A.1 (top) Periodic function valid for Fourier series. (bottom) Non-periodic function valid for the Fourier transform. 185

Chapter 1

Introduction

1.1 Motivation

Partial Differential Equations (PDEs) serve as a fundamental mathematical language for describing change in the physical world. From the propagation of electromagnetic waves to the chaotic fluctuations of atmospheric turbulence, PDEs enable the modelling and prediction of complex systems across vast spatial and temporal scales (Strauss, 2007).

However, solving these equations remains challenging. Most governing equations for real-world phenomena, such as the Navier–Stokes equations, exhibit nonlinear behaviour across strongly coupled length and time scales, rendering analytical solutions intractable. We therefore rely on traditional numerical methods to find approximate solutions (e.g., finite element or finite volume methods (Ames, 2014)). These methods are highly reliable and, for a wide class of problems, come with rigorous convergence guarantees, so that high accuracy can be achieved by refining the discretisation. Their principal limitation is therefore not a lack of attainable accuracy but its *cost*: a single high-fidelity solve can be expensive, and this cost becomes prohibitive in "many-query" settings, where the governing equations must be solved repeatedly within an outer loop. Such settings include PDE-constrained optimisation and inverse design, uncertainty quantification, and real-time or control applications, where thousands of forward solves may be required. A complementary motivation is data-driven model discovery, in which learned surrogates are used to identify or close governing equations directly from observational or simulation data. These outer-loop tasks motivate the development of fast, reusable surrogate models that amortise the cost of repeated solves rather than replacing a single accurate numerical solve.

To address this limitation, a new paradigm has emerged: using Artificial Intelligence (AI) and Deep Learning (DL) as accelerated surrogate models. Unlike traditional solvers that iteratively resolve the physics at every grid point, neural network-based models, specifically **Neural Operators**, aim to learn the underlying functional mapping of the physical dynamics from data. By shifting the heavy computational burden to an offline training phase, these models can offer fast inference once trained. We note, however, that reported speedups vary widely and should be interpreted with care: such comparisons are sensitive to the choice of numerical baseline, the hardware used, and the accuracy tolerance demanded, and a systematic review has argued that many headline comparisons in this literature rely on weak baselines (McGreivy & Hakim, 2024). However, while DL has revolutionised fields such as image recognition (Krizhevsky *et al.*, 2017) and natural language processing (Manning & Schutze, 1999), its application to the time evolution of PDEs introduces unique challenges for long-term stability and error accumulation.

This thesis investigates the challenges faced by deep learning-based models, such as neural operators, in predicting the evolution of time-dependent PDEs. In particular, we focus on the transition from simple one-step predictions to robust long-term forecasting, with the ultimate goal of improving these models' stability and accuracy.

1.2 Challenges in Long-term Prediction

While deep learning models, particularly Neural Operators, have demonstrated remarkable success in single-step mappings, their application to the long-term evolution of time-dependent PDEs remains a challenge. In most practical engineering scenarios, we require predictions that span multiple characteristic time scales; however, several factors hinder current architectures' ability to maintain accuracy over these extended horizons.

- **Autoregressive Error Accumulation:** The prevailing approach for temporal prediction is the autoregressive "rollout" method, where the model's prediction at time t is fed back as the input for the next time step, $t + \Delta t$. This recursive process introduces a fundamental instability: any minor approximation error made in the initial steps may be propagated and amplified through subsequent iterations. In chaotic or turbulent systems, such as those governed by the Navier–Stokes equations, this error often grows exponentially, causing the predicted trajectory to diverge rapidly from the true physical solution (McCabe *et al.*, 2023). Where models are trained only on short-horizon (e.g., one-step-ahead) objectives, they remain un-

aware of this cumulative drift during training, leading to a "distribution shift" at test time, where the model encounters its own error-prone predictions. We emphasise that not all neural PDE solvers are trained this way: some learn the full spatio-temporal solution in a single pass, as in the space–time (FNO-3D) variant of the Fourier neural operator (Li *et al.*, 2020b), while others are explicitly trained over multiple unrolled steps, as in message-passing neural PDE solvers (Brandstetter *et al.*, 2022c). More recently, transformer-based neural operators have been proposed to model temporal dependencies over longer horizons (Cao, 2021; Herde *et al.*, 2024).

- **Spectral Instability:** Recent investigations have identified the accumulation of high-frequency signals as a primary source of this instability. In many complex PDEs, energy is transferred across scales; however, neural networks can struggle to capture high-frequency components with low amplitudes. When these small-scale features are incorrectly predicted, they can manifest as noise or errors that corrupt the larger-scale solution (Lippe *et al.*, 2023). Unlike classical numerical methods, which employ carefully designed filters or artificial viscosity to maintain stability, standard deep learning architectures do not, by default, incorporate any mechanism analogous to numerical dissipation or spectral filtering, and therefore have no built-in means of damping these non-physical high-frequency oscillations.

- **Constraint of Constant Time-Stepping:** Furthermore, existing autoregressive frameworks are largely restricted to constant time-step intervals. This lack of temporal flexibility is a significant limitation for real-world physical processes where the rate of change is not uniform. For instance, in problems involving heat conduction or chemical reactions, the system may evolve rapidly during initial transients and slowly as it approaches equilibrium. A model constrained to a fixed Δt can either be computationally inefficient during slow phases or numerically unstable during fast phases. Developing "variable-timestep" operator models is therefore essential to achieve the efficiency required to compete with adaptive classical solvers.

1.3 Aims and Objectives

The primary goal of this research is to enhance the long-term stability and temporal flexibility of Neural Operators for time-dependent PDEs. This aim is subdivided into three core areas, with objectives defined to bridge the gap between architectural innovation and robust training strategies.

Aim 1: Robust training for long-term stability (Constant-timestep models)

The first aim is to develop training strategies that mitigate "exposure bias" and error propagation in autoregressive models. While standard Neural Operators are effective for single-step mappings, they often diverge during long-term "roll-outs", where the model's previous predictions are used as inputs. We propose a curriculum learning approach to stabilise this process by transitioning the model from simpler to more complex temporal tasks.

Objectives

1. Conduct a comparative analysis of baseline training strategies, such as "teacher-forcing" and multi-step autoregressive rollouts, to quantify the trade-off between local accuracy and global stability.

2. Develop and implement *Scheduled Temporal Loss Weighting* (STLW), a novel curriculum learning strategy that dynamically weights the loss across the temporal rollout during training.

3. Validate the STLW framework across a diverse suite of benchmark PDEs, including Advection, Inviscid Burgers', and Korteweg-de Vries (KdV) equations, to assess its generalisability across different physical regimes.

Aim 2: Flexible Temporal Resolution (Variable-timestep)

The second aim is to break Neural Operators' dependence on fixed time-stepping schemes. Traditional autoregressive models are restricted to a constant Δt seen during training, making them inefficient for systems with multiscale dynamics or non-stationary systems (e.g., time-dependent PDE parameters). We therefore aim to develop a conditioning mechanism that enables a single model to predict at arbitrary time steps.

Objectives

1. Design and implement an *attention-based temporal conditioning* mechanism within the FNO architecture to modulate feature maps based on a requested temporal query Δt.

2. Compare various temporal embedding strategies, such as sinusoidal temporal embeddings (STE), embedding layers (EL), and multi-layer perceptrons (MLP), to identify the most effective representation for continuous time.

3. Investigate "variable-timestep sampling" strategies (e.g., Large-to-Small vs. Small-to-Large) to determine whether a combination of larger and smaller time steps can effectively bypass high-error phases and improve accuracy for selected benchmarks.

4. Demonstrate the efficacy of variable-timestep models in non-stationary systems, such as advection with time-dependent wave speed.

Aim 3: Synthesis of Model and Training Strategy

The final aim is to evaluate the potential benefit of combining architectural flexibility (variable-timestep) with advanced training protocols (curriculum learning). We seek to determine whether these two improvements are mutually reinforcing when applied to complex, dissipative, and reactive systems.

Objectives

1. Integrate the STLW curriculum learning framework with the attention-based variable-timestep model.

2. Benchmark the integrated variable-timestep model with STLW to quantify the "generalisation penalty" versus the gain in global stability.

3. Perform a scaling analysis to determine how increasing model capacity and data availability influence the convergence of variable-timestep models toward the accuracy of constant-timestep model baselines.

1.4 Thesis Outline

This thesis investigates novel training methodologies and architectural frameworks designed to enhance the predictive accuracy and stability of neural operators for time-dependent partial differential equations (PDEs). The research is structured to transition from foundational theory and baseline evaluations to the development of specific methodological and architectural solutions.

A comprehensive background and literature review is provided in Chapter 2 to establish the theoretical landscape; readers familiar with the field may find

this chapter a reference for specific technical contexts. Chapter 3 then establishes the methodological foundation on which the contribution chapters build, including the operator-learning and autoregressive time-stepping formulations, the benchmark equations, and the baseline training and evaluation protocols.

The core contributions are detailed in Chapters 4 and 5, which focus on a novel training strategy and model architecture, respectively. These are written as standalone chapters, each containing a concise review of specific related work. Finally, the conclusion synthesises these findings and presents initial results that combine the proposed techniques, offering a roadmap for future research.

The specific content of each chapter is outlined below:

- Chapter 2 examines the evolution of deep learning solvers for PDEs, tracing the transition from traditional numerical discretisation to operator learning. It explores the strengths and limitations of physics-informed neural networks (PINNs) and Fourier neural operators (FNOs), particularly regarding their ability to respect physical symmetries and conservation laws. By reviewing recent advances in time-series prediction and inductive biases, the chapter establishes the technical context for the stability and flexibility challenges addressed in subsequent chapters.

- Chapter 3 establishes the theoretical and mathematical foundation for the research, detailing the formulation of learning problems for time-dependent partial differential equations (PDEs). It provides a comprehensive overview of the Fourier neural operator (FNO) and its variants, such as Factorised F-FNO, as well as alternative architectures like U-Net and U-NO. The chapter also introduces the governing equations used as benchmarks throughout this work, ranging from advection and the viscous Burgers equation to the Korteweg-de Vries (KdV) equation, and defines the training protocols and error metrics, such as nRMSE, that serve as the quantitative basis for evaluating model performance.

- Chapter 4 introduces the first major contribution of the thesis: Scheduled Temporal Loss Weighting (STLW), a novel curriculum learning strategy designed to stabilise constant-timestep autoregressive models. The chapter first establishes, through a baseline rollout analysis, the exposure-bias "stability gap" between local one-step accuracy and long-term stability that motivates this approach. Addressing the challenge of exponential error propagation, STLW uses a continuous curriculum that dynamically reweights the loss across the temporal horizon as training progresses. The experimental results demonstrate that this approach bridges the gap between teacher-forcing and full-rollout training, yielding substantial improvements

in stability across various dissipative and dispersive systems, including a 62.5% reduction in error for the viscous Burgers equation.

- Chapter 5 shifts focus toward architectural flexibility, proposing a parallel attention-based conditioning mechanism that enables Fourier neural operators to perform variable-timestep prediction. By evaluating different temporal embedding strategies, such as embedding layers (EL) and multi-layer perceptron (MLP)-based embeddings, the research demonstrates how a single model can be conditioned to predict at arbitrary time steps (Δt) not seen during training. The chapter further explores optimal sampling strategies, revealing that while variable-timestep models may incur a minor generalisation penalty, they offer superior global structural fidelity and efficiency, particularly in non-stationary systems (e.g., advection with a time-dependent wave speed).

- Chapter 6 synthesises the findings of the thesis, emphasising the complementarity between STLW and the variable-timestep model. It reflects on the primary contributions to knowledge, including how the combination of curriculum learning and temporal conditioning mitigates key challenges in operator learning. The thesis concludes by outlining future research directions, including scaling these methods to three-dimensional turbulent flows, integrating hard physical constraints, and applying variable-timestep operators to irregular domains and complex geometries.

Chapter 2

Literature Review

The emergence of deep learning models for solving partial differential equations (PDEs) has shifted the paradigm of computational modelling from purely numerical discretisation to data-driven, physics-augmented inference. This chapter provides a comprehensive review of progress in this rapidly evolving field, with a specific focus on architectures that bridge the gap between classical numerical methods and modern machine learning. We examine the evolution of physics-informed frameworks, ranging from the mesh-free optimisation of physics-informed neural networks (PINNs) (Raissi, 2018) to the generalisation capabilities of operator-learning architectures, such as the Fourier Neural Operator (FNO) (Li *et al.*, 2020b) and DeepONets (Lu *et al.*, 2021).

The review follows a single line of argument, moving from the physical problem to the specific gap addressed by this thesis. Section 2.1 establishes the mathematical foundation, introducing time-dependent PDEs, the conditions required for a well-posed problem, and the classification of second-order equations. Section 2.2 examines the classical numerical methods used to solve them and the computational costs that limit their use when many repeated solves are required. Section 2.3 introduces deep learning as an alternative, and Section 2.4 sets out the challenges that distinguish learned PDE solvers from conventional machine learning applications. Section 2.5 describes how physical knowledge can be embedded into learning algorithms through observational, learning, and inductive biases, the last of which motivates operator learning. Section 2.6 reviews operator-learning architectures in detail, covering DeepONets, the neural operator family and the Fourier Neural Operator, a comparative analysis of the two paradigms (Section 2.6.3), and subsequent extensions (Sections 2.6.4 and 2.6.5). Section 2.7 then turns to the temporal dimension, reviewing time-series prediction and identifying the stability and flexibility limitations of autoregressive rollout. Section 2.8 draws

these threads together and states the resulting research gap.

2.1 Time-Dependent Partial Differential Equations

Partial Differential Equations (PDEs) are a class of differential equations that involve multiple independent variables and their partial derivatives. For example, the temperature of a rod with hot and cold ends depends "partially" on space and time and can therefore be described using a PDE. Differential equations are among the most powerful tools for modelling and predicting physical phenomena. They are fundamental to scientific understanding at micro-scales, such as quantum mechanics (Schrödinger's equation) (Berezin & Shubin, 2012) and molecular dynamics (the Fokker–Planck equation) (Risken & Risken, 1996), as well as at macro-scales, such as geophysical and astrophysical processes that span large spatial and temporal scales (Tan, 1992). The Navier–Stokes equations provide a salient example: they form a system of nonlinear partial differential equations that describe the evolution of fluids in space and time, and their importance stems from the ubiquity of fluid flow across science and engineering. Their mathematical intractability is underscored by the fact that the existence and smoothness of their solutions in three dimensions remains an open problem, recognised as one of the Millennium Prize Problems (Doering, 2009).

A key distinction between PDEs and ordinary differential equations (ODEs) is the number of independent variables: ODEs involve a single independent variable, whereas PDEs involve two or more. In both cases the unknown is a function - the *dependent variable* - whose value is determined by the independent variables, for instance space and time. Solving a PDE therefore means finding the function that, together with its partial derivatives, satisfies the equation throughout the domain of interest.

Solutions of PDEs that describe physical processes typically require a "well-posed problem", i.e., a PDE posed on a domain together with physically meaningful auxiliary conditions, namely conditions on the initial state of the system (initial conditions, prescribed in time) and on the behaviour of the solution at the domain boundary (boundary conditions, prescribed in space), that yield three properties: **existence, uniqueness, and stability**. Existence implies that at least one solution satisfies the auxiliary conditions, uniqueness means that at most one solution exists, and stability means that the solution depends continuously on the auxiliary conditions. For example, to make the heat equation on a rod well-posed, one must specify the initial condition and boundary conditions

at both ends of the rod.

PDEs can be classified in several ways based on their characteristics (e.g., the order of the highest-derivative terms, linearity, and the value of the discriminant). For second-order linear PDEs the last criterion can be made precise. Writing such an equation in two independent variables as

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G, \tag{2.1}$$

its type is determined by the discriminant $B^2 - AC$ of the second-order (principal) part: the equation is *elliptic* where $B^2 - AC < 0$, *parabolic* where $B^2 - AC = 0$, and *hyperbolic* where $B^2 - AC > 0$. This classification governs the qualitative behaviour of the solution and the auxiliary conditions required for well-posedness. The main classes are summarised below:

- **Elliptic PDEs:** Elliptic equations ($B^2 - AC < 0$) describe equilibrium or steady-state problems, in which local disturbances are smoothed out. They arise naturally in boundary value problems (BVPs) and exhibit strong regularising properties, yielding smooth solutions. The canonical example is Poisson's equation,

  $$-\Delta u = f, \tag{2.2}$$

  where $\Delta = \sum_i \partial^2/\partial x_i^2$ is the Laplacian, u is the unknown function and f is the forcing or source term (taking $f \equiv 0$ recovers Laplace's equation). The generic form $\mathcal{L}u = f$ alone does not characterise ellipticity, since parabolic and hyperbolic equations can be written in the same way; what distinguishes the three types is the structure of the second-order operator $\mathcal{L}$, as captured by the sign of the discriminant above.

- **Parabolic PDEs:** Parabolic equations ($B^2 - AC = 0$) are commonly used to model changes in time and are accompanied by initial and boundary conditions for a well-posed problem. The solutions tend to be smooth and regular and appropriate for modelling diffusion and heat conduction. The general form of a parabolic equation is:

  $$u_t = \mathcal{L}u + f, \tag{2.3}$$

  where u_t is the partial derivative of u with respect to time.

- **Hyperbolic PDEs:** Hyperbolic equations ($B^2 - AC > 0$) describe wave-like phenomena that propagate in space and time. Their solutions are often framed as initial value problems (IVPs). Disturbances propagate at a finite speed along characteristic lines; therefore, not all points in the domain feel the effect of the disturbance immediately.

- **Linear and Nonlinear PDEs:** In linear PDEs, the dependent variable and its partial derivatives appear only in linear combinations, whereas nonlinear PDEs can include products, powers, and other nonlinear functions of the dependent variable and its derivatives. Nonlinear PDEs exhibit more complex phenomena and are generally more difficult to solve.

- **First-Order and Second-Order PDEs:** First-order PDEs correspond to equations in which the highest derivative is first order, whereas second-order PDEs involve second-order derivatives. Examples of first-order PDEs include the Hamilton–Jacobi equation and the transport equation. Most PDEs that describe physical processes are second order, as is the case for elliptic, parabolic, and hyperbolic PDEs.

While partial differential equations (PDEs) are essential for describing many fundamental processes, obtaining exact solutions can be challenging. For most PDEs, deriving analytical solutions using traditional mathematical techniques is difficult. These difficulties arise from the complexity of the underlying processes, which often exhibit nonlinear behaviour across large time and length scales and involve multiple physical phenomena that can be strongly coupled. As a result, scientists and engineers rely on numerical methods to obtain approximate solutions. Although these numerical solutions are not exact, they provide valuable approximations that help us understand and predict the behaviour of complex systems, such as in fluid dynamics. These methods solve PDEs by discretising the equations in space and/or time (Ames, 2014).

2.2 Classical Numerical Solvers and Their Limitations

The rapid evolution of computer hardware and software has led to the emergence of powerful computer systems equipped with vast resources. This has significantly improved our ability to simulate complex systems with higher accuracy at finer resolutions. Nevertheless, while classical numerical methods have proven effective and reliable, their application is still hindered by several challenges. The primary concern is the high computational cost of these methods, driven by the fine discretisations required to resolve the smallest length scales, numerical stability constraints, high dimensionality, mesh generation, and modelling assumptions. The stability and accuracy of these methods for time-dependent PDEs can be sensitive to the choice of time-step size, especially when simulating

long-term or rapidly changing behaviour. Moreover, achieving robust performance often requires domain-specific expertise to select an appropriate method, as well as careful implementation and code optimisation. In summary, while the fundamental behaviour of many physical processes is relatively well understood, the key challenge lies in solving the resulting PDEs efficiently, particularly when many repeated solves are required.

These challenges are becoming more apparent as we approach the physical limits of semiconductor manufacturing at smaller scales (Vashchenko & Sinkevitch, 2008). This trend constrains the linear growth in computing power observed over the past decade, as predicted by Moore's law (Thompson & Parthasarathy, 2006). Furthermore, the energy demands of high-performance computing (HPC), along with its contribution to our carbon footprint, are an increasing concern as we strive to reduce climate impacts (Vital *et al.*, 2013). Given the projected growth in global energy demand (Raimi *et al.*, 2023), there is a pressing need to explore alternative approaches to reduce the computational costs of solving PDEs.

2.3 Deep Learning for Partial Differential Equations

Over the last decade, advances in specialised computer hardware (e.g., graphics processing units; GPUs) have had a profound impact on the development of artificial intelligence (AI) in applications such as image recognition (Krizhevsky *et al.*, 2017), reinforcement learning (Mnih *et al.*, 2013), and natural language processing (Manning & Schutze, 1999). These advances are largely attributable to the availability of large datasets, significant computing resources, and the development of efficient training algorithms. Central to this progress is the adoption of artificial neural networks (ANNs), which have given rise to a subfield within machine learning known as deep learning.

Neural networks aim to mimic the signal processing of biological neurons by passing input signals through "deep" sequences of linear and nonlinear operations. Their broad empirical success spans diverse domains, including healthcare (Esteva *et al.*, 2019), agriculture (Kamilaris & Prenafeta-Boldú, 2018), and finance (Heaton *et al.*, 2017). The sciences and engineering have likewise seen substantial growth in the application of deep learning to improve and accelerate the understanding of physical processes, with applications ranging from subatomic length scales (Luo *et al.*, 2023; Szabó & Castelnovo, 2020) to astronomical length scales (Aragon-Calvo, 2019), protein folding (Ruff & Pappu, 2021), molecular dynamics (Mardt *et al.*, 2018), carbon-capture storage (Wen *et al.*, 2022), and weather

forecasting (Ravuri *et al.*, 2021).

Recently, deep learning has shown promise in reducing the computational cost of solving PDEs. These methods have been applied to ordinary differential equations (ODEs) (Chen *et al.*, 2018), the Navier–Stokes equations (Raissi, 2018), Maxwell's equations (Zhang *et al.*, 2021), the shallow-water equations (Bihlo & Popovych, 2022), and Schrödinger's equation (Pfau *et al.*, 2020). Such equations can be addressed using deep learning methods in either a data-driven or a data-free setting.

When data are available (from experiments or simulations), models are trained to learn the dynamics present in the dataset and then used to infer solutions for new conditions not seen during training, thereby capturing the underlying dynamics of the PDE (Li *et al.*, 2020b). When data are unavailable, physics-informed approaches can incorporate the PDE residual as a constraint, thereby encouraging the model to satisfy the governing equations (Raissi *et al.*, 2019). Another successful direction is hybrid methods that combine deep learning with classical numerical solvers (Heinlein *et al.*, 2021).

The computational advantage of these methods is best understood as an *amortisation*: the cost of solving is shifted to an offline training phase, after which inference for each new instance is fast. The saving is therefore realised in many-query settings, where the up-front training cost is recovered over a large number of subsequent solves, rather than in a single solve. It is important to stress that a learned surrogate targets a fixed accuracy that is limited by optimisation and approximation error, and typically does not match the high accuracy obtainable by refining a classical solver. Specialised hardware (e.g., GPUs designed for parallel processing) can further reduce inference time. These methods are also flexible and can be adapted to different objectives: they have been used to generalise across different initial conditions (Li *et al.*, 2020b), boundary conditions (Farimani *et al.*, 2017), PDE parameters (Brandstetter *et al.*, 2022c), and even different equations (Wu *et al.*, 2022).

These capabilities are, however, accompanied by a distinct set of difficulties, which the following section sets out.

2.4 Open Challenges for Learned PDE Solvers

Machine learning is playing an increasingly important role in science and engineering, particularly for systems governed by partial differential equations (PDEs). A range of traditional machine learning models has been used to solve PDEs. For example, Recurrent Neural Networks (RNNs) and Transformers can be used to emulate time-dependent PDEs (Cao, 2021; del Águila Ferrandis *et al.*, 2021;

Vlachas *et al.*, 2022), and Generative Adversarial Networks (GANs) have been used to model stochastic differential equations (Gao & Ng, 2022). Among these approaches, Convolutional Neural Networks (CNNs) are especially popular, as they parameterise PDE operators between finite-dimensional Euclidean spaces (Kutyniok *et al.*, 2022; Roska *et al.*, 1995). Their popularity stems from their ability to capture local features in images, making them well-suited to predicting fluid-flow fields (Liu *et al.*, 2020; Wang *et al.*, 2020).

Despite the potential of deep learning methods to solve PDEs efficiently, several key challenges remain and must be addressed before they can be widely adopted. These challenges differ from those in traditional deep learning applications (e.g., image recognition), where data generation is relatively cheap, but they are similar to challenges faced by classical numerical methods. The systems of interest often exhibit complex, multiscale dynamics coupled with nonlinearities that span large spatiotemporal scales, such as turbulent flows. Therefore, to obtain accurate and reliable results, these methods must effectively capture interactions within and between dynamics at different spatial and temporal scales. Accurately predicting long-term behaviour is also challenging because errors accumulate over time. In addition, deep learning methods must handle complex boundary conditions and irregular domains, making the development of flexible methods important.

Alongside these, three further concerns limit practical deployment: limited data availability, interpretability, and generalisation. In science and engineering, large datasets are not always readily accessible because obtaining them may require expensive field measurements, laboratory experiments, or computationally intensive simulations. The limited interpretability and poor generalisation of deep learning models remain a major concern, particularly when the goal is to quantify underlying physical processes for prediction and control.

For science and engineering applications, the central challenge is to learn functions that respect (or satisfy) prior knowledge about the problem using limited data, given the high cost of numerical simulations and experiments. In many physical processes, we observe symmetries and invariances that can lead to reduced problem formulations, and physical processes obey various laws, such as the laws of thermodynamics, optics, electromagnetism, and conservation. Therefore, enhancing neural networks' ability to generalise to unseen conditions requires methods that either respect symmetries or use physical principles to constrain predictions to physically plausible outcomes. Classical numerical methods for solving PDEs have addressed these challenges for decades; thus, ideas from these areas can significantly improve the performance of deep learning methods.

2.5 Physics-Informed Machine Learning

The traditional machine learning paradigm is to map the problem of interest onto standard machine learning algorithms such as support vector machines (Hearst *et al.*, 1998), decision trees (Myles *et al.*, 2004), or fully connected neural networks (Bishop, 1994). However, in recent years, a model-based approach has offered several advantages over traditional methods (Bishop, 2013). This approach seeks to derive models tailored to the problem, for example, by embedding explicit and/or inductive prior knowledge or modelling assumptions into the learning algorithm. Understanding the problem's assumptions and prior knowledge is crucial, and encoding them in the model is central to model-based machine learning, often referred to as "*physics-informed machine learning*" in science and engineering applications. This approach can offer significant benefits because we already have some initial understanding of the fundamental behaviour of physical systems.

An example of a model-based approach is the use of Convolutional Neural Networks (CNNs) in computer vision applications (Gu *et al.*, 2018; LeCun *et al.*, 1989). The convolution operation is translation-invariant; therefore, encoding this property in neural networks is desirable for image recognition or classification. Another example is the use of recurrence in neural networks, as in Recurrent Neural Networks (RNNs) for sequential prediction. In this case, the model assumes that a prediction at a future time depends only on the system's state at the previous time step, making recurrence advantageous for sequential or time-dependent problems (Medsker & Jain, 2001).

On the theme of physics-informed machine learning, Karniadakis *et al.* (2021) defined three ways to embed physics into data-driven methods: **observational**, **inductive**, and **learning** biases. Observational biases include data augmentation and preprocessing techniques that encourage the model to implicitly satisfy certain physical laws; inductive biases involve the careful design of machine learning architectures to capture prior knowledge; and learning biases involve the use of loss functions and constraints during training. Each is examined in turn below, since together they underpin the operator-learning architectures developed in the remainder of this thesis.

2.5.1 Observational bias

Observational bias involves augmenting training datasets to help the model capture underlying constraints. Brandstetter *et al.* (2022b) used Lie point symmetry data augmentation to improve neural PDE solver sample complexity by an order of magnitude. Yang *et al.* (2022) improved performance by generating synthetic data using a generative model that leverages physical knowledge.

2.5.2 Learning bias

The first well-known approach, popularised by Raissi *et al.* (2019), uses a learning bias to develop physics-informed neural networks (PINNs) for solving nonlinear partial differential equations. PINNs incorporate prior physical knowledge by minimising the residual of the governing differential equations. This approach adds an extra loss term derived from the PDE and uses automatic differentiation (Paszke *et al.*, 2017) to differentiate the neural network with respect to its input coordinates and parameters. PINNs have been applied widely and have shown encouraging results. However, this type of bias is typically a soft constraint and can struggle with convergence due to potentially conflicting objectives among loss terms in multi-objective optimisation (Rohrhofer *et al.*, 2022). PINNs are also known to exhibit slow convergence and spectral bias that favours low-frequency components, which can lead to inaccurate solutions that filter out high-frequency components of the true solution (Wang *et al.*, 2021).

2.5.3 Inductive bias

Inductive bias involves designing neural network architectures so that their predictions align with prior knowledge or information. Hansen *et al.* (2023) incorporated conservation constraints by combining conservation laws with a Bayesian update. Prantl *et al.* (2022) implemented a hard constraint by enforcing conservation of momentum using antisymmetric continuous convolutional layers. Geiger & Smidt (2022) proposed an equivariant neural network, Euclidean Neural Networks, that naturally operates on geometry and geometric tensors describing systems in 3D.

Recently, observational and inductive biases that encode hard constraints have received increased attention. For example, much of the spatiotemporal data in science and engineering, obtained from measurements, experiments, and simulations, can be assumed to lie on a continuous domain. Embedding this bias into neural networks can yield resolution-invariant models in which the error is independent of data resolution. The generalisation of neural networks to learning mappings between function spaces, often referred to as "*operator learning*", has become one of the most promising research directions. An operator is a mapping between function spaces, and a function space is an infinite-dimensional space consisting of a set of functions. An example of this bias is the family of architectures known as neural operators (Kovachki *et al.*, 2021). These networks learn in function space, leading to resolution invariance, i.e., network parameters and errors are independent of data resolution once a sufficient resolution is reached. In a supervised learning setting, where inputs and corresponding outputs are available, a neural operator learns the solution operator for a family of

PDEs. Learning in function space implies that network parameters should be independent of input data resolution, and that the trained network's error should be resolution-invariant if the operator is learned correctly. This contrasts with classical neural networks that map between finite-dimensional spaces, which may not generalise well across resolutions. The work of Kovachki *et al.* (2021) demonstrated that neural operators can be trained at lower resolution and evaluated directly at higher resolution without a detrimental loss in performance.

The following section examines these architectures, and the design choices that distinguish them, in detail.

2.6 Neural Operators for PDEs

The motivation for operator learning using neural networks is to discover a parametric operator that maps between function spaces. Learning mappings between function spaces has significant applications in science and engineering, particularly in fluid dynamics, which is the focus of this work. For example, a key assumption in the derivation of the governing equations of fluid dynamics in differential form is the "*continuum assumption*". Fluids are assumed to be continuous, i.e. macroscopic quantities of interest, such as pressure and velocity, are taken to vary continuously from one point to another in the fluid domain. Although fluids are composed of individual molecules at the microscopic scale, the continuum assumption is valid at macroscopic scales and allows the use of control-volume and fluid-parcel approaches in mathematical modelling. A model of these fluid systems can take the form of differential equations, in which the boundary and initial conditions required to obtain a unique solution are continuous functions of space and time. The solution to these equations and quantities of interest can be continuous functions either in space and/or in time. For example, temperature over the Earth's surface can be assumed to be a continuous function of position, and wind speed measured with an anemometer can be assumed to vary continuously over time.

The formulation of infinite-dimensional neural networks is motivated by the need to approximate the nonlinear operator of a family of PDEs that map between input and output function spaces. We focus on two popular approaches in the literature: DeepONets and Neural Operator. The work of Chen & Chen (1995) proposed a universal operator approximation theorem, which states that a neural network with a single hidden layer can approximate any nonlinear continuous functional and nonlinear operator. This theorem inspired the first deep neural network architecture for learning nonlinear operators, DeepONets (Lu *et al.*, 2021). The second approach to learning nonlinear operators, Neural Operator (Li

et al., 2020c), learns integral operators that can approximate the non-local effects in the solution operator of PDEs. These two approaches to operator regression aim to break the curse of dimensionality by learning in infinite-dimensional input and output spaces, enabling the dynamics of complex nonlinear systems to be learned without retraining the neural network.

2.6.1 Deep Operator Networks (DeepONets)

The Deep Operator Network proposed by Lu *et al.* (2021) is composed of two neural networks: a branch net for the input space and a trunk net for the output space. The architecture is motivated by the proposed universal operator approximation theorem by Chen & Chen (1995), and the output function of the DeepONet is formulated as:

$$\mathcal{G}(u)(\xi) \approx \sum_{k=1}^{p} b_k(u) t_k(\xi) + b_0 \tag{2.4}$$

where $\mathcal{G}$ is the operator, u is the input function and ξ is any point in the output domain $\mathcal{G}(u)$. b_k is the output of the branch net, t_k is the output of the trunk net and b_0 is the bias.

The branch net can be any classical neural network architecture, such as a CNN or an RNN. It takes in the discretised input function u, i.e. $\{u(x_1), u(x_2), \dots u(x_m)\}$ and returns the coefficients b_k with p values. The trunk net is usually a fully connected network that takes the evaluation point ξ in the output space and returns the basis vector with p values. Thus, the approximation of the output function is the product of the coefficients b_k and the basis vectors t_k.

DeepONet was shown to be universal, i.e it can approximate a large class of nonlinear operators despite its underlying finite-dimensional setting. DeepONet has found applications in predicting linear instability waves in high-speed boundary layers (Di Leoni *et al.*, 2021), solving PDEs with multiple input functions (Tan & Chen, 2022) and nanoscale heat transport (Lu *et al.*, 2022b), to name a few. Extensions to the original DeepONet architecture include POD-DeepONets by Lu *et al.* (2022a), which uses a proper orthogonal decomposition (POD) basis as input to the trunk net and computes coefficients of the POD basis from the branch net. A physics-informed extension, inspired by PINNs, was proposed by Goswami *et al.* (2022) to address cases where training data are sparse.

2.6.2 Neural Operators

The approach of neural operator architectures, originally proposed in Li *et al.* (2020c), is inspired by Green's function for solving linear partial differential equa-

tions. For example, let

$$\mathcal{L}u(x) = f(x) \tag{2.5}$$

where $\mathcal{L}$ is a linear partial differential operator, $u(x)$ is the solution function and $f(x)$ is the forcing term. The solution function $u(x)$ can be expressed in term of the Green's function $G(x, y)$ as

$$u(y) = \int G(x, y)f(x) \tag{2.6}$$

The Green's function for a neural operator is learned, and by composing the integral transform (2.6) with a nonlinear activation function, a nonlinear operator can be learned.

The neural operator architecture follows the classical approach to developing neural networks by composing a nonlinear activation function with linear integral operators. The architecture is composed of three successive operations: lifting, iterative kernel integration, and projection. The lifting operation maps the low-dimensional input function to a high-dimensional space. The iterative kernel integration (2.6) is computed by learning a kernel, which is synonymous with learning the Green's function. Finally, the projection operation maps back to the output domain. Kovachki *et al.* (2021) proposed a universal approximation theorem showing that this architecture can learn any given nonlinear continuous operator. Furthermore, it was shown that the network is resolution-invariant: its parameters and error remain constant as resolution increases when the operator is correctly learned, which is a desirable property. This architecture has shown promising results and has been used for multiphase flow (Wen *et al.*, 2022), weather forecasting (Pathak *et al.*, 2022), and seismic wave equations (Li *et al.*, 2022a), to name a few.

There are four methods for parameterising or learning the integral operator. The first method proposed by Li *et al.* (2020c), Graph Neural Operator (GNO), uses a graph neural network to learn a kernel via the Nystrom approximation to reduce the cost of computing the integral. This amounts to a uniform sampling on the domain of the output function, such that the points randomly sampled J' are much less than the points J in the domain. This approach has a computational complexity of $\mathcal{O}(JJ')$ and proves to be computationally expensive, scaling poorly in comparison to other methods. Also, You *et al.* (2022) showed instabilities for very deep layers and proposed ways to alleviate such limitations.

The second method, Multipole Graph Neural Operator (MGNO) by Li *et al.* (2020d), uses multi-scale decomposition of the kernel to achieve linear complexity of approximately $\mathcal{O}(J)$. The kernel is decomposed into different interaction ranges, and a hierarchy of low-rank structures is imposed on the long-range interactions. Another way to achieve linear complexity, as proposed by Kovachki

et al. (2021), is to assume the kernel is of a tensor product form, called a Low-Rank Operator (LNO). This approach was shown to be similar to the DeepONet framework but formulated to be consistent in function space.

The most efficient and popular method, the Fourier Neural Operator (FNO), proposed by Li *et al.* (2020b), uses spectral-domain multiplication to avoid explicitly computing the integral operator. It imposes a convolution integral operator, since convolution in the spatial domain corresponds to multiplication in Fourier space. This approach benefits from the fast implementation of the Fourier transform (FFT), leading to a complexity of $\mathcal{O}(J \log J)$. The kernel is complex-valued and is learned in Fourier space.

In summary, GNO and MNO use Graph Neural Networks (GNNs) to approximate the sampling and integration scheme, with computational complexity that scales with the number of sampled grid points. By enforcing additional structure on the kernel, LNO and FNO achieve lower complexity. The kernel networks employed in LNO, GNO, and MGNO can perform well in the small-parameter regime, whereas FNO typically requires more parameters for strong performance. However, with sufficient parameters, FNO outperforms the other methods across the test cases investigated by Kovachki *et al.* (2021).

2.6.3 Neural Operator vs DeepONet

Neural operators and DeepONets have found substantial success in science and engineering applications, and either architecture can be used for many operator-learning problems. However, it is unclear which architecture is preferable or which has advantages over the other, especially as the most thorough comparison of the two (Lu *et al.*, 2022a) was carried out by the group that developed DeepONet, who questioned the methodology of the earlier comparison reported by the originators of FNO (Li *et al.*, 2020b), in which FNO had appeared the more accurate of the two.

From a theoretical perspective, DeepONet is often viewed as a linear approximation method because the nonlinear operator is approximated by a linear span, i.e., the trunk net provides a linear basis of the output space $\{t_1, t_2, \ldots, t_k\}$, where k is the number of basis functions. Therefore, the accuracy of the approximation can depend on the number of basis functions k (Pinkus, 2012). DeepONets may be limited by this property, since some PDEs require a large value of k to achieve good accuracy (Cohen & DeVore, 2015).

Nonlinear approximation methods can offer advantages over linear approximations, and neural operators adopt this perspective. For function approximation, these benefits are well understood (DeVore *et al.*, 1989), whereas the corresponding theory for operators is still in its early stages (Bonito *et al.*, 2021). Theoret-

ically, both operator networks have been shown to approximate a wide class of nonlinear operators with a desired accuracy, making them "universal".

The architectures take different approaches. DeepONet uses standard neural architectures, such as CNNs, RNNs, and GNNs, for the branch net and fully connected networks for the trunk net. This offers flexibility and design choices depending on the problem. However, because they rely on discretised inputs, DeepONets are not resolution-invariant, as the number of parameters can depend on the discretisation of the input space. In contrast, neural operators propose a dedicated neural layer inspired by the classical neural network design of a linear transform (a linear operator) followed by a nonlinear activation function. This provides less flexibility for feature engineering, but the resulting formulation is consistent in function spaces and can yield a resolution-invariant network.

Since the learning process is implemented numerically, the input and output function spaces must be finite-dimensionalised, which introduces biases and limitations that depend on how the function space is represented. DeepONet uses different discretisations for the input and output spaces: the branch net takes element-wise values of the input function, and the trunk net evaluates the output at arbitrary locations. The original architecture proposed by Liu *et al.* (2020) requires the input values to be sampled at the same locations for all training examples, while the output can be evaluated at arbitrary locations. This representation is similar to the low-rank neural operator (LNO) and is among the most general and flexible, enabling broad applicability; however, it can also limit representation power.

Lu *et al.* (2022a) introduced several extensions to the baseline DeepONet architecture to obtain comparable results against FNO in some test cases, including using POD modes of the training data in the trunk net, adding extra features to both the branch and trunk nets, and modifying the trunk net to enforce Dirichlet and periodic boundary conditions.

The way the function space is finite-dimensionalised varies considerably across neural operators, and with it their flexibility. Graph-based neural operators such as GNO and MGNO discretise the function space using graph nodes, which offers a highly general representation (second only to LNO and DeepONet) and does not require the input and output to share a discretisation. By contrast, the Fourier Neural Operator (FNO) is the least flexible because it is discretised in Fourier space, which typically requires a uniform grid in real space and periodic functions; this shared, structured discretisation is a property of the FNO specifically, rather than of neural operators in general. Not surprisingly, it has strong representational power and performs well on test cases where these conditions hold (Kovachki *et al.*, 2021; Lu *et al.*, 2022a).

Lu *et al.* (2022a) compared both networks by evaluating 16 test cases of varying complexity. In general, the original FNO outperformed the original DeepONet

in most case studies, given FNO's limitations. They also proposed extensions to the original DeepONet architecture that improved performance, especially POD-DeepONet.

The DeepONet variants were shown to perform better on more challenging problems; however, important details such as computational complexity, scaling behaviour, and ablation studies were not reported. Moreover, some test cases in which DeepONet variants performed substantially better were outside the limitations of FNO (e.g., Darcy flows with complex geometries, unstructured grids, and discontinuous functions). Fourier transforms typically require a regular, equispaced grid. Although FNO variants can use interpolation to handle complex geometries and irregular grids, interpolation schemes can introduce additional bias and potentially large interpolation errors for nonlinear PDEs. For the wave-equation case study, a square (discontinuous) wave was used as the initial condition, even though Fourier transforms are known to represent discontinuities poorly (the Gibbs phenomenon).

Furthermore, results from Kissas *et al.* (2022) on operators with coupled attention—evaluated on five test cases within the limits of Fourier transforms—showed that the original FNO architecture performed better than the original DeepONet in most cases.

In summary, the trade-off between flexibility and representation power is evident between FNO and DeepONet. FNO is constrained by its use of the Fourier transform and therefore tends to perform better than DeepONet on a subset of problems where its assumptions are satisfied. DeepONet is more flexible due to its linear-approximation form and generally works well across a broader range of problems. DeepONet is reviewed here to situate the Fourier Neural Operator within the broader operator-learning landscape; the empirical contributions of this thesis, which concern temporal stability and variable-timestep prediction, are developed on the FNO family and related grid-based neural operators, and DeepONet is therefore used for context rather than as an experimental baseline.

2.6.4 Beyond Fourier Neural Operator (FNO)

The use of the Fourier transform in deep learning can be traced back to its use in proving the universal approximation theorem of neural networks (Hornik *et al.*, 1989). It has been used to implement faster convolutional networks (Li *et al.*, 2020a), to study training and generalisation in neural networks (Xu, 2018), and for time-series prediction (Busseti *et al.*, 2012). In our setting, Fourier transforms enable fast implementations of neural operators but reduce the flexibility of the problems that can be addressed.

One way to alleviate some of these limitations is to use Fourier continuation,

proposed by Bruno *et al.* (2007) for non-uniform geometries. Several variants of the FNO architecture have been proposed to obtain better performance. Tran *et al.* (2021) took advantage of the separable nature of Fourier transforms in space, coupled with a shared kernel integral operator to obtain better performance with a smaller number of parameters. Kissas *et al.* (2022) leveraged the recent success of the attention mechanism to propose a new architecture in which input functions are mapped to a set of averaged, weighted features, improving FNO performance. To alleviate the limitation of irregular geometries, Li *et al.* (2022b) proposed an architecture that deforms the irregular domain into a uniform latent mesh using a neural network on which the Fourier transform can be applied, and Wen *et al.* (2022) combined advances in CNN with FNO to obtain better performance for multiphase flow applications.

A more recent and novel approach by Brandstetter *et al.* (2022a) addresses the correlation between flow fields, such as pressure (scalar) and velocity (vector), that can arise in physical processes over time. They proposed a Clifford neural layer based on Clifford algebra that captures flow-field correlations using multivector fields. Their approach replaces Fourier operations with their Clifford counterparts, showing better performance for multi-physics problems. Tripura & Chakraborty (2022) proposed the Wavelet Neural Operator that uses wavelet transforms for accurate spatiotemporal tracking of flow features. Cao *et al.* (2023) implemented a Laplace Neural Operator that uses Laplace transforms to handle non-periodic functions and transient response.

2.6.5 Graph-Based Approaches

Recent advances in graph representation learning could offer better accuracy and performance than the graph neural operator proposed by Li *et al.* (2020c). As noted by Kovachki *et al.* (2021), the full-integration method using GNO without approximation can still outperform FNO, albeit at a higher computational cost. Therefore, it is of interest to explore new approximation techniques for graph learning that reduce computational cost with little or no loss in accuracy.

Furthermore, many problems in science and engineering, such as fluid and solid dynamics, can be naturally interpreted as graphs. In the case of fluids, fluid particles and their interactions can be represented as nodes and edges, respectively. For mesh-based numerical simulations, graphs can directly represent both structured and unstructured meshes, giving them greater flexibility than CNNs. They can be used to represent nodes in FDM, elements in FEM, and control volumes in FVM. Graph Neural Networks have been used to predict aerodynamics and structural mechanics by leveraging adaptive meshing techniques to improve performance (Pfaff *et al.*, 2020). Multiscale techniques have also been used to

improve GNN performance in settings where non-local dynamics are important (e.g., weather forecasting) (Lino *et al.*, 2022). Iakovlev *et al.* (2020) proposed a continuous-time graph model that uses message passing to parameterise the governing equations.

Message passing remains the most widely used information-propagation scheme for graph-structured data. The scheme updates node representations by exchanging and aggregating information from neighbouring nodes via edge connections. Message-passing models can capture local structure and, to some extent, global information, but they can struggle with over-smoothing (where node features become very similar) (Chen *et al.*, 2020). They can also capture long-range dependencies poorly, be computationally expensive to train, and scale poorly (Alon & Yahav, 2020).

To alleviate these limitations, graph transformers have shown promise as an alternative, albeit with increased computational cost due to quadratic scaling (Yun *et al.*, 2019). Instead of using only a node's neighbourhood for updates, graph transformers apply self-attention over all nodes in the graph.

The architectures reviewed above are concerned principally with spatial representation. Learning the evolution of time-dependent PDEs additionally requires an explicit treatment of the temporal dimension, to which we now turn.

2.7 Temporal Modelling for PDE Evolution

Learning the evolution of time-dependent PDEs requires time-series analysis of discrete-time samples of the underlying dynamics. Time-series prediction has been studied for decades using statistical methods (Box & Jenkins, 1968) and, more recently, through deep learning (Lim & Zohren, 2021). Statistical models such as autoregressive integrated moving average (ARIMA) (Nelson, 1998), state-space models (Aoki, 2013), and Gaussian process models (Roberts *et al.*, 2013) have been widely used for time-series data.

Deep learning models provide an alternative approach for learning temporal dependencies directly from data. Different architectures have been used for time-series prediction; the most common include convolution-based, recurrent-based, and attention-based approaches. Convolution-based approaches use Convolutional Neural Networks (CNNs) to learn temporal dependencies from past time steps to predict future states. This yields a time-invariant model, i.e., the kernels remain fixed across time. However, many CNN-based approaches are primarily spatial and rely on autoregressive temporal prediction (Qu *et al.*, 2022). In general, CNNs are not naturally suited to time-series prediction without additional pre-processing.

Recurrent neural networks (RNNs) have long been a standard model for time-series data (Koutnik *et al.*, 2014). They process temporal data using recurrent connections that allow information to flow from one step to the next. Temporal dependencies are captured through a hidden state that aggregates information from previous time steps. Vlachas *et al.* (2022) used RNNs to learn multiscale PDE dynamics by evolving the model on two different temporal scales. Limitations of vanilla RNNs in modelling long-range dependencies led to variants such as long short-term memory networks (LSTMs) and gated recurrent units (GRUs) (Dey & Salem, 2017; Yu *et al.*, 2019). These architectures improve gradient flow and reduce vanishing and exploding gradients.

Attention-based networks, such as transformers (Vaswani *et al.*, 2017), address the sequential training bottleneck of recurrent networks by processing time-series data in parallel. The attention mechanism can capture long-range dependencies by dynamically weighting relevant parts of the history, allowing the model to "attend" to important past states for accurate future prediction. Attention mechanisms have been used for PDE prediction and have been shown to improve long-term prediction (Li *et al.*, 2022c; Peng *et al.*, 2022).

2.8 Summary and Research Gap

This chapter has reviewed the mathematical, numerical, and machine-learning foundations relevant to the prediction of time-dependent partial differential equations. Beginning with classical numerical methods, the discussion highlighted their robustness and physical fidelity, while also emphasising their significant computational cost, limited scalability, and reliance on carefully tuned discretisation strategies. These limitations motivate the search for alternative approaches that can accelerate simulation without sacrificing accuracy.

Recent advances in deep learning have enabled powerful data-driven and physics-informed methods for PDE modelling. Physics-Informed Neural Networks offer a principled means of embedding governing equations directly into the learning process, but remain constrained by optimisation difficulties, limited scalability, and poor performance in long-time or high-dimensional regimes. In contrast, operator learning methods, particularly Neural Operators and the Fourier Neural Operator, provide an efficient and discretisation-invariant framework for learning solution operators across families of PDEs. These models demonstrate strong spatial generalisation and fast inference, making them attractive candidates for surrogate modelling.

However, the literature also reveals persistent challenges in the temporal domain. Most neural operator architectures rely on fixed-step autoregressive predic-

tion, which is prone to error accumulation, exposure bias, and spectral instability during long-term rollouts. Furthermore, the restriction to a single timestep limits their applicability to systems with non-uniform or multiscale temporal dynamics. While time-series modelling techniques and variable-time formulations have been explored in the broader machine learning literature, existing approaches often struggle to scale efficiently to high-dimensional PDEs or to maintain stability over extended prediction horizons.

Collectively, these observations expose a clear gap between the expressive capacity of modern neural architectures and the stringent stability and flexibility requirements of long-term PDE prediction. Addressing this gap requires approaches that improve temporal robustness and adaptability without sacrificing the computational advantages of neural operators.

Motivated by these challenges, the next chapter formalises the problem setting considered in this thesis. It introduces the governing equations, datasets, baseline neural operator architectures, and evaluation techniques used throughout the study, providing a rigorous foundation for developing and assessing the proposed training strategies and variable-time modelling approaches.

Chapter 3

Methods

This chapter details the methodological framework and mathematical formulations employed to learn the evolution of time-dependent partial differential equations (PDEs). This work uses Neural Operators to learn continuous mappings between functional spaces, enabling accelerated inference.

The Chapter begins by formalising the operator learning problem and the challenges associated with autoregressive modelling, specifically the accumulation of errors over long-term rollouts. We then describe the architectural configurations of the Fourier Neural Operator (FNO) and its variants, which form the core of our experimental investigations. Finally, we introduce a suite of benchmark PDEs, ranging from simple advection to complex nonlinear systems such as the Korteweg-de Vries (KdV) equation, and define the metrics used to evaluate model performance and stability.

3.1 Problem Definition

This section formalises the mathematical framework for learning the temporal evolution of physical systems. We define the problem as an operator learning task, where the goal is to map a system's state from an initial time to a subsequent state.

3.1.1 Operator Learning

We define our operator learning problem by considering a class of time-dependent PDEs on $t \in [0, T]$ and $x \in D$, where $D \subset \mathbb{R}^d$ is a bounded spatial domain, subject

to a prescribed boundary datum on ∂D, in the following form:

$$\begin{aligned} \mathcal{G}^\dagger(a)(t,x) &= u(t,x) \qquad (t,x) \in [0,T] \times D, \\ u(0,x) &= a(x) \qquad x \in D, \\ u(t,x) &= b(t,x) \qquad (t,x) \in [0,T] \times \partial D \end{aligned} \tag{3.1}$$

where $a \in \mathcal{A}$ is the initial-condition function, $b \in \mathcal{B}$ is the prescribed boundary datum defined on $[0,T] \times \partial D$, and $u \in \mathcal{U}$ is the solution function. The spaces $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{U}$ are Banach spaces of functions defined on D, $[0,T] \times \partial D$ and $[0,T] \times D$, respectively. The operator $\mathcal{G}^\dagger : \mathcal{A} \to \mathcal{U}$ is the *solution operator* of the PDE: for a fixed boundary datum it maps the initial condition a to the corresponding solution u. It is therefore not itself a differential operator, but rather the (generally nonlinear and non-local) operator that the governing equation implicitly defines. Figure 3.1 gives a visual description of the mapping from $\mathcal{A}$ to $\mathcal{U}$.

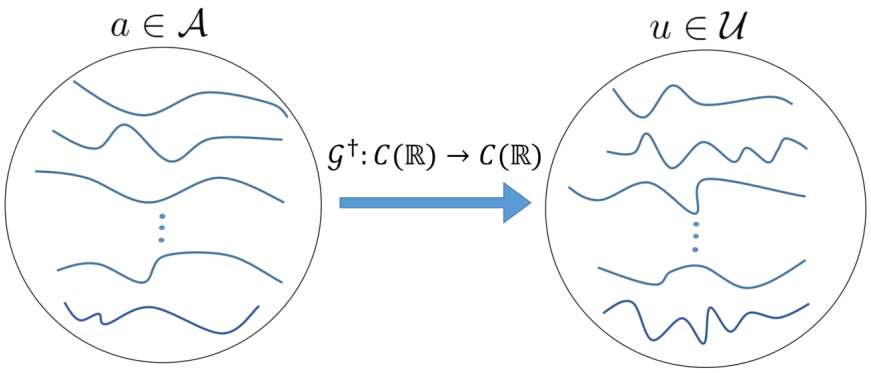


Figure 3.1: Operator learning. $\mathcal{A}$ and $\mathcal{U}$ denote the spaces of admissible input (initial-condition) and output (solution) functions, respectively. The input $a \in \mathcal{A}$ is a function on the spatial domain $D \subset \mathbb{R}^d$, while the output $u \in \mathcal{U}$ is the corresponding solution on $[0,T] \times D$; each represents an individual function drawn from its respective function space.

The goal is to accurately learn the solution operator that maps any $a \in \mathcal{A}$ to its corresponding $u \in \mathcal{U}$. Therefore we introduce a nonlinear parametric model $\mathcal{G}_\theta$ with parameters θ

$$\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}, \quad \theta \in \mathbb{R}^p, \tag{3.2}$$

where θ are the parameters of the model in $\mathbb{R}^p$. The learning problem aims to find the optimum θ^*, such that $\mathcal{G}_{\theta^*} \approx \mathcal{G}^\dagger$.

The mapping is formulated by using a finite collection of input-output function pairs $\{a^{(i)}, u^{(i)}\}_{i=1}^{N}$, where each input $a^{(i)} \sim \mu$ is an independent and identically distributed sample drawn from a probability measure μ supported on the input function space $\mathcal{A}$. The approach to evaluating the error is an empirical test-train method, where the difference between $\mathcal{G}_\theta(a^{(i)})$ and $\mathcal{G}^\dagger(a^{(i)})$ is minimised using a fixed number of training samples, and the model performance is evaluated from new test samples that have not been seen during training.

3.1.2 Autoregressive (Time-stepping) Formulation

The solution operator $\mathcal{G}^\dagger$ defined above maps an initial condition directly to the full space–time solution. For the time-dependent problems studied in this thesis, however, the solution is advanced *incrementally* in time: the models of Chapters 4 and 5 take the current state as input and predict the state a fixed step Δt later. We therefore introduce the autoregressive (time-stepping) formulation used throughout those chapters. We consider a class of time-dependent PDEs on $t \in [0, T]$ and $x \in D$ with boundary condition B of the form:

$$\begin{aligned} \partial_t u(t,x) &= \mathcal{F}^\dagger\left(u(t,x)\right) \qquad (t,x) \in [0,T] \times D, \\ u(0,x) &= u_0(x) \qquad x \in D, \\ u(t,x) &= B \qquad (t,x) \in [0,T] \times \partial D, \end{aligned} \tag{3.3}$$

where $u : [0,T] \times D \subset \mathbb{R}^d$, with initial condition $u_0(x)$ at $t = 0$ and boundary condition $u(t,x) = B$ on ∂D, and $\mathcal{F}^\dagger$ is the (spatial) differential operator on the right-hand side. The aim is to learn an autoregressive model that takes $u(t)$ as input and predicts the next state $u(t + \Delta t)$. We define a nonlinear parametric autoregressive model $\mathcal{F}_\theta$ with parameters θ:

$$u\left(t + \Delta t\right) = \mathcal{F}_\theta\left(\Delta t, u(t)\right) \quad \theta \in \mathbb{R}^p, \tag{3.4}$$

where the learning problem seeks the optimum θ^* such that $\mathcal{F}_{\theta^*} \approx \mathcal{F}^\dagger$, using a finite collection of input–output pairs $\{u(t)^{(i)}, u(t + \Delta t)^{(i)}\}_{i=1}^{N}$ and the same empirical train–test evaluation described above.

More generally, the single-state map (3.4) is the special case $H = 1$, $P = 1$ of a windowed formulation in which the model is conditioned on a *history* of H past states and predicts a *horizon* of P future states in a single forward pass,

$$\left(u(t + \Delta t), \ldots, u(t + P\,\Delta t)\right) = \mathcal{F}_\theta(\Delta t,\ u(t),\ u(t - \Delta t), \ldots, u(t - (H-1)\Delta t))\,, \tag{3.5}$$

after which the predicted states are fed back as inputs to advance the solution autoregressively. The history length H controls how much temporal context the

operator receives, while the prediction horizon P controls how many steps are supervised jointly during training; the effect of both is studied empirically in Section 4.4.2.

3.2 Fourier Neural Operator (FNO)

The Fourier Neural Operator (FNO) architecture proposed by Kovachki *et al.* (2021) is presented below, and implementation of the FNO model is discussed below.

3.2.1 Architecture

For L layers in a Fourier Neural Operator (FNO), the architecture proposed by Kovachki *et al.* (2021) in Figure 3.2 is written as

$$\mathcal{G}_\theta := \mathcal{Q} \circ \sigma_L(W_{L-1} + \mathcal{K}_{L-1} + b_{L-1}) \circ \cdots \circ \sigma_1(W_0 + \mathcal{K}_0 + b_0) \circ \mathcal{P}. \tag{3.6}$$

The architecture has three main components defined as follows;

1. **Encoder** $\mathcal{P}$
   The lifting stage maps the dimension of the input function to a higher dimension $a \mapsto v_0$, where $a \in \mathbb{R}^{d_a}$ and $v_0 \in \mathbb{R}^{d_{v_0}}$. It is a fully local operation, where $d_{v_0} > d_a$.

2. **Processsor** $\sigma_L(W_{L-1} + \mathcal{K}_{L-1} + b_{L-1})$
   Each iteration maps the hidden representation to the next $v_l \mapsto v_{l+1}$, where $v_l \in \mathbb{R}^{d_{v_l}}$ and $v_{l+1} \in \mathbb{R}^{d_{v_{l+1}}}$. L is the number of iterations/layers. Each layer is composed of a sum of a local linear operator W_l, a non-local integral kernel operator $\mathcal{K}_l$, and a bias function b_l. Finally, the sum is passed through a local nonlinear activation function σ_l.

3. **Decoder** $\mathcal{Q}$
   The projection stage maps the last hidden representation to the dimension of the output function, $v_L \mapsto u$, where $v_L \in \mathbb{R}^{d_{v_L}}$ and $u \in \mathbb{R}^{d_u}$. It is a fully local operation, where $d_{v_L} > d_u$.

The local lifting $\mathcal{P}$ and projection $\mathcal{Q}$ maps $\mathbb{R}^{d_a} \to \mathbb{R}^{d_{v_0}}$ and $\mathbb{R}^{d_{v_L}} \to \mathbb{R}^{d_u}$, respectively. The lifting stage transforms the input data into a higher-dimensional space so that the non-local part of the operator can be captured, while the projection maps the last layer to the output space. The local operators $(\mathcal{P}, \mathcal{Q})$ are

element-wise operations, such that the transformation of a point does not depend on other points in the domain, i.e. $(\mathcal{P}(a))(x) = (\mathcal{P}a(x))$ for any $x \in D$.

The non-local component is learned by the processor through iterative kernel layers. Each layer is composed of the sum of a local linear operator W_l, a non-local linear integral kernel operator $\mathcal{K}_l$, and a local bias function b_l into a local nonlinear activation function σ_l. Thus, the parameters of the neural operator $\theta \in \mathbb{R}^p$ are composed of the parameters of $\mathcal{P}, \mathcal{Q}$, b_l, W_l, and $\mathcal{K}_l$.

The integral kernel operator $\mathcal{K}_l$ is a non-local integral transformation of the function v_l from its original space y to a new space x using the kernel function $\kappa^{(l)}$. Kovachki *et al.* (2021) expressed $\mathcal{K}_l$ as:

$$(\mathcal{K}_l(v_l))(x) = \int_D \kappa^{(l)}(x, y) v_l(y)\, d\nu_l(y) \quad \forall x \in D. \tag{3.7}$$

For a full single hidden layer, equation 3.6 becomes

$$(\mathcal{G}_\theta(a))(x) = \mathcal{Q}\left(\sigma\left(W_0\mathcal{P}(a(y)) + \int_D \kappa^{(0)}(x, y)\mathcal{P}(a(y))\, dy + b_0(y)\right)\right). \tag{3.8}$$

The operations, $\mathcal{P}, \mathcal{Q}, W_0, b_0$ in (3.8) are defined using a multi-layer perceptron (MLP). By assuming $\kappa(x, y) = \kappa(x - y)$, a convolution operator is obtained

$$(\mathcal{K}(v))(x) := \int_D \kappa(x - y)v(y)\, dy, \quad \forall x \in D. \tag{3.9}$$

Therefore, we can efficiently parameterise $\mathcal{K}$ directly in the Fourier space since a convolution operation of two functions in Fourier space is a product of the two Fourier-transformed functions (see appendix A for a summary of the relevant Fourier analysis). Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ be the Fourier transform and inverse Fourier transform, respectively. Thus,

$$(\mathcal{K}(v))(x) = \mathcal{F}^{-1}(\mathcal{F}(\kappa) \cdot \mathcal{F}(v))(x), \quad \forall x \in D. \tag{3.10}$$

The Fourier transform of the kernel $\mathcal{F}(\kappa)$ can be parameterised as a weight matrix R_ϕ

$$(\mathcal{K}(v))(x) = \mathcal{F}^{-1}(R_\phi \cdot \mathcal{F}(v))(x), \quad \forall x \in D. \tag{3.11}$$

R_ϕ is a parameterised Fourier transform of the periodic function κ. Each frequency mode k is parameterised as $R_\phi(k) \in \mathbb{C}^{m\times n}$ and $(\mathcal{F}v)(k) \in \mathbb{C}^n$. For regularisation and efficient computation, a finite set of k_{max} modes is used and R_ϕ becomes a complex-valued tensor of shape $(k_{max} \times m \times n)$. Finally, the multiplication of $R \in \mathbb{C}^{k_{\max}\times d_v \times d_v}$ and $\mathcal{F}v \in \mathbb{C}^{k_{\max}\times d_v}$ in Fourier space becomes

$$(R \cdot (\mathcal{F}v))_{k,l} = \sum_{j=1}^{n} R_{k,l,j}(\mathcal{F}v_l)_{k,j}, \quad k = 1, ..., k_{\max}, \quad j = 1, ..., m. \tag{3.12}$$

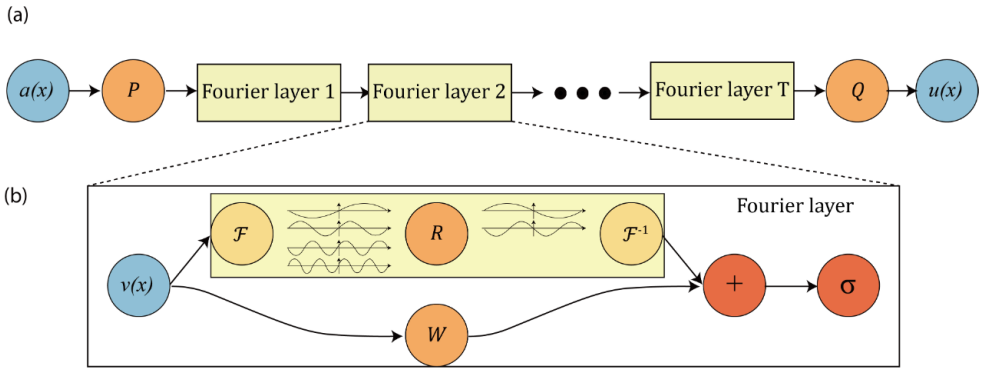


Figure 3.2: (top) Fourier Neural Operator architecture. (bottom) composition of a single FNO layer. Taken from Kovachki *et al.* (2021).

3.2.2 Implementation

The FNO implementation is composed of a linear layer for the encoder and a nonlinear MLP for the decoder. The iterative layer has 4 layers, each composed of a nonlinear activation applied to the sum of spectral and spatial convolutions.

Index	**Name**	**Implementation**
1	Encoder	**Linear**(*in_channel*, *d_model*)
2	Layer (4x)	**GeLU** (**SpectralConv**(*d_model*, *d_model*) + **Conv**(*d_model*, *d_model*))
3	Decoder	**MLP**(*d_model*, *in_channel*)

Table 3.1: FNO model with 4 layers

The components in Table 3.1 are defined as follows. **Linear** denotes a linear layer with bias and **GeLU** the Gaussian Error Linear Unit activation function. The spectral convolution **SpectralConv** implements $\mathcal{F}^{-1}(\mathcal{F}(R \cdot \textbf{Projection}(x)))$, in which R is a complex-valued weight matrix with random initialisation of shape $[1, \mathit{modes}, \mathit{d_model}, \mathit{d_model}]$, while **Conv** is a convolution layer with a fixed kernel size of one.

3.3 Variants of FNO

We define two FNO variants: FNO-Decompose, which uses real-valued weights to represent complex weights, and Factorised-FNO (Tran *et al.*, 2021), which has an improved representation layer.

3.3.1 FNO-Decompose

The FNO-decompose architecture is similar to the FNO; however, the complex weights in the FNO are implemented as two real weights in the network. Therefore, the model parameters are twice those of FNO. The table below shows the model's structure.

The components in Table 3.2 follow those of the standard FNO: **Linear** is a linear layer with bias and **GeLU** the GELU activation function. The decomposed spectral convolution **SpectralConvDecompose** implements $\mathcal{F}^{-1}(\mathcal{F}(R \cdot \textbf{Projection}(x)))$, where R is now a real-valued weight matrix with random initialisation of shape [2, *modes*, *d_model*, *d_model*] (the leading dimension of two storing the real and imaginary parts), and **Conv** is a convolution layer with a fixed kernel size of one.

Index	Name	Implementation
1	Projection	**Linear**(*in_channel*, *d_model*)
2	Layer (4x)	**GeLU** (**SpectralConvDecompose**(*d_model*, *d_model*) + **Conv**(*d_model*, *d_model*))
3	Projection	**Linear**(*d_model*, *in_channel*)

Table 3.2: FNO-Decompose model with 4 layers

3.3.2 Factorised F-FNO

The factorised Fourier Neural Operator (F-FNO) as outlined in Table 3.3 includes an improved representation layer that learns features in the Fourier space independently for each dimension, a process called Fourier factorisation (Tran

et al., 2021). The F-FNO places residual connections after activation, enabling our neural operator to benefit from a deeply stacked network.

The components in Table 3.3 are defined as follows. $\mathbf{MLP}_1$ and $\mathbf{MLP}_2$ are two-layer multi-layer perceptrons, each formed by the sequence **Linear** $\rightarrow$ **Dropout** $\rightarrow$ **ReLU** $\rightarrow$ **LayerNorm** with a residual connection. The decomposed spectral convolution **SpectralConvDecompose** implements $\mathcal{F}^{-1}(\mathcal{F}(R \cdot \mathbf{Projection}(x)))$, where R is a real-valued weight matrix with random initialisation of shape $[2, \mathit{modes}, \mathit{d_model}, \mathit{d_model}]$.

Index	Name	Implementation
1	Projection	**Linear**(*in_channel*, *d_model*)
2	Layer (4x)	$\mathbf{MLP}_1 \circ \mathbf{MLP}_2$ $\circ$ **SpectralConvDecompose**(*d_model*, *d_model*)
3	Projection	**Linear**(*d_model*, *in_channel*)

Table 3.3: Factorised-FNO model

3.4 Other Architectures

We define other architectures used in this work for comparison with FNO: U-NET Gupta & Brandstetter (2022) and the U-shaped Neural Operator (U-NO) (Rahman *et al.*, 2023).

3.4.1 U-NET

U-Nets serve as foundational frameworks for tasks such as image synthesis, segmentation, and modelling. Their design features a symmetrical structure where a contraction path (downsampling) is paired with an expansion path (upsampling), linked by shortcut connections that bridge corresponding feature maps. This configuration (Table 3.4) enables a more balanced integration of fine-grained and high-level features than architectures such as Fourier Neural Operators (FNOs).

The components in Table 3.4 are defined as follows. A **ConvBlock** is a two-layer network formed by the sequence **Conv** $\rightarrow$ **GroupNorm** $\rightarrow$ **GeLU** $\rightarrow$ **LayerNorm**, with kernel size three and padding one. The downsampling block **DownBlock_n** at resolution n applies **MaxPool(2)** followed by a **ConvBlock**, while the upsampling block **UpBlock_n** concatenates a **ConvTranspose** with

the corresponding **DownBlock_n** and applies a **ConvBlock**, with a residual connection from the matching downsampled resolution.

Index	Name	Implementation
1	Projection	**ConvBlock**(*in_channel*, *out_channel*)
2	Number of resolutions (N)	n $\in 1, ..., N$
3	Downsampling Block	**DownBlock_n**(*n*out_channel*, *out_channel*$\times$*((n+1)*$\times$*factor*)
4	Upsampling Block	**UpBlock_n**(*out_channel*$\times$*((n+1)*$\times$*factor*, *n*out_channel*)
5	Projection	**ConvBlock**(*out_channel*, *in_channel*)

Table 3.4: U-NET model

3.4.2 U-shaped Neural Operator (U-NO)

The U-shaped Neural Operator (U-NO) as outlined in Table 3.5 is a hybrid architecture that combines the multi-scale structure of a U-Net with the mathematical foundation of Neural Operators (specifically, Fourier Neural Operators) (Rahman *et al.*, 2023).

The components in Table 3.5 are defined as follows. An **MLP** is a two-layer network formed by the sequence **Linear** $\rightarrow$ **GeLU**. The **OperatorBlock_n** at a downsampled resolution applies **MaxPool(2)** followed by a **ConvBlock**, while the upsampling block **UpBlock_n** concatenates a **ConvTranspose** with the corresponding **DownBlock_n** and applies a **ConvBlock**, with a residual connection from the matching downsampled resolution.

Index	Name	Implementation
1	Projection	**MLP**(*in_channel*, *out_channel*)
2	Number of resolutions (N)	n $\in 1, ..., N$
3	Operator Block	**OperatorBlock_n**(*n*out_channel*, *out_channel*$\times$*((n+1)*$\times$*factor*)
5	Projection	**MLP**(*out_channel*, *in_channel*)

Table 3.5: U-Shaped Neural Operator (U-NO) model

3.5 Code Implementation

The result chapters (4 and 5) are based on these GitHub repositories https://github.com/seuncoker/constant_autoregression and https://github.com/seuncoker/variable_time_autoregression, respectively.

The training protocols and models are provided. The original FNO architecture by (Li *et al.*, 2020b) and all current extensions of the FNO were implemented in PyTorch. PyTorch is more Pythonic and feels similar to NumPy, making it a popular choice for experimentation and research in the research community. The original FNO implementation was not flexible for hyperparameter tuning; therefore, it was adapted to meet current needs, and its existing code was modified to support further research and experiments.

3.6 Partial Differential Equations

We define the partial differential equations used in this work and how the datasets for training, evaluating, and testing our models were generated. We use benchmark datasets by Takamoto *et al.* (2022) for Advection (Av), viscous Burgers' (vB), Diffusion-Reaction (DR) and Diffusion-Sorption (DS), and Brandstetter *et al.* (2022c) for inviscid Burgers' (iB) and Korteweg-De Vries (KdV).

3.6.1 Advection Equation (Av)

The advection equation models pure advection behaviour without diffusion.

$$\begin{aligned} \partial_t u(t,x) + \beta \partial_x u(t,x) = 0, \quad x \in (0,1), t \in (0,1], \\ u(0,x) = u_0(x), \quad x \in (0,1), \end{aligned} \tag{3.13}$$

where $\beta = 0.5$ is the advection speed. Note that the exact solution of the system is given as: $u(t,x) = u_0(x - \beta t)$. The boundary conditions are periodic, and the initial conditions are generated by a superposition of sinusoidal waves as:

$$u_0(x) = \sum_{k_i = k_1, \cdots, K_N} A_i \sin(k_i x + \phi_i), \tag{3.14}$$

where $k_i = 2\phi\{ni\}/L_x$ are wave numbers whose $\{ni\}$ are integer numbers selected randomly in $[1, n_{max}]$, N is the integer determining how many waves to be added, L_x is the calculation domain size, A_i is a random float number uniformly chosen in $[0, 1]$, and ϕ_i is the randomly chosen phase in $(0, 2\pi)$. In all examples included in this work, we select $k_{max} = 2$ and $N = 2$. For the numerical setup, the spatial domain $[0, 1)$ is partitioned into $n_x = 1024$ equidistant points, while the temporal interval $[0, 1]$ is discretised using $n_t = 201$ uniform steps. The numerical solution was computed using a 2nd-order upwind finite difference scheme in both time and space. For training and inference, the spatial domain is downsampled to $n_x = 256$.

3.6.2 Viscous Burgers' Equation (vB)

The viscous Burgers' equation is a non-linear parabolic partial differential equation (PDE) with applications in fluid dynamics, nonlinear acoustics, and gas dynamics. It is the simplest PDE that combines nonlinearity and diffusivity. The equation often appears in simplified models of complex systems, thereby providing insight into the general behaviour of complex processes. The one-dimensional version of a viscous fluid is given as

$$\begin{aligned} \partial_t u(t,x) + u(t,x)\partial_x u(t,x) = \nu \partial_{xx} u(t,x) \quad x \in (0,1),\ t \in (0,2], \\ u(0,x) = u_0(x), \qquad x \in (0,1), \end{aligned} \tag{3.15}$$

where u is scalar velocity and is a function of space x and time t. The parameter $\nu = 0.01$ is the viscosity term.

The initial conditions are generated as in the advection equation. The numerical solution was calculated with the temporally and spatially 2nd-order upwind difference scheme for the advection term, and the central difference scheme for the

diffusion term. For the numerical setup, the spatial domain (0, 1) is partitioned into $n_x = 1024$ equidistant points, while the temporal interval $[0, 1]$ is discretised using $n_t = 201$ uniform steps. For training and inference, the spatial domain is downsampled to $n_x = 256$.

3.6.3 Inviscid Burgers' Equation (iB)

The inviscid Burgers' equation is similar to the viscous Burgers' equation (3.15); however, a source term is introduced, and there is no diffusion, i.e., $\nu = 0$.

$$\begin{aligned} \partial_t u(t,x) + u(t,x)\partial_x u(t,x) &= \delta(t,x), \\ u(0,x) = \delta(0,x), \quad \delta(t,x) &= \sum_{j=1}^{J} A_j \sin(w_j t + 2\pi\, l_j\, x/L + \phi_j), \end{aligned} \tag{3.16}$$

where u velocity is a function of space x and time t.

The forcing function, denoted by δ, is characterised by parameters $J = 5$ and $L = 16$. Its constituent coefficients are determined through uniform sampling, specifically $A_j \sim U[-0.5, 0.5]$, $w_j \sim U[-0.4, 0.4]$, and $\phi_j \sim U[0, 2\pi)$, with l_j chosen from the set $\{1, 2, 3\}$. For the numerical setup, the spatial domain $(0, 16)$ is partitioned into $n_x = 200$ equidistant points, while the temporal interval $(0, 4]$ is discretised using $n_t = 250$ uniform steps.

3.6.4 Diffusion-Reaction Equation (DR)

Here, we consider a one-dimensional diffusion-reaction PDE that combines a diffusion process with rapid evolution driven by a source term. The equation is expressed as:

$$\begin{aligned} \partial_t u(t,x) - \nu \partial_{xx} u(t,x) - \rho u(1-u) = 0, \quad x \in (0,1),\, t \in (0,1], \\ u(0,x) = u_0(x), x \in (0,1), \end{aligned} \tag{3.17}$$

where viscosity $\nu = 0.0005$ and density $\rho = 0.01$. The variable u could grow exponentially due to the force term, which depends on u. As in the 1D advection equation, we use periodic boundary conditions and the same initial condition (Equation 3.14); however, $k_{max} = 3$ and $N = 3$. The numerical solution was computed using a 2nd-order central-difference scheme in both time and space. For the source term part, we use the piecewise-exact solution (PES) method. For the domain setup, the spatial domain [0, 1] is partitioned into $n_x = 1024$ equidistant points, while the temporal interval $[0, 1]$ is discretised using $n_t = 201$ uniform steps. For training and inference, the spatial domain is downsampled to $n_x = 256$.

3.6.5 Diffusion-Sorption (DS)

The diffusion-sorption equation models a diffusion process that is retarded by sorption. The equation is written as

$$\partial_t u(t.x) = D/R(u)\partial_{xx} u(t,x), \tag{3.18}$$

where D is the effective diffusion coefficient, R is the retardation factor representing the sorption that hinders the diffusion process.

This equation is retarded by the retardation factor R, which is dependent on u based on the Freundlich sorption isotherm:

$$R(u) = 1 + \frac{1-\phi}{\phi}\rho_s k n_f u^{n_f - 1}, \tag{3.19}$$

where $\phi = 0.29$ is the porosity of the porous medium, $\rho_s = 2880$ is the bulk density, $k = 3.5 \times 10^{-4}$ is the Freundlich's parameter, $n_f = 0.874$ is the Freundlich's exponent, and the effective diffusion coefficient $D = 5 \times 10^{-4}$.

The initial condition is generated with a uniform distribution $u(0,x) \sim \mathcal{U}(0, 0.2)$ for $x \in (0,1)$. The data is discretised into $n_x = 1024$ and $n_t = 501$, and the downsampled version is used for the models' training, with $n_x = 256$, $n_t = 101$. The spatial discretisation is performed using the finite volume method (**?**), and the time integration is performed using a fourth-order Runge-Kutta method.

3.6.6 Korteweg-De Vries Equation (KdV)

The Korteweg-de Vries equation describes waves on the surface of shallow water. It exhibits typical behaviours, such as a large number of explicit solutions, in particular soliton solutions (nonlinear, self-reinforcing, localised wave packets that are strongly stable). The equation is given as;

$$\partial_t u + \partial_{xxx} u - 6\, u\, \partial_x u = 0 \quad x \in \mathbb{R}^d,\ t \geq 0, \tag{3.20}$$

The nonlinear convective term $u\, u_x$ and the dispersive term u_{xxx} require careful attention from the traditional solvers. The numerical solutions are obtained in a periodic domain using the method of lines, with spatial derivatives computed via a pseudospectral method. In the pseudospectral method, the derivatives are computed in the frequency domain by applying the fast Fourier transform (FFT), multiplying by the appropriate values and converting back to the spatial domain with the inverse FFT. For integration in time, an implicit Runge-Kutta method of Radau IIA family, order 5 is used (**?**). The initial conditions are sampled from

a distribution over truncated Fourier series with random coefficients $\{A_j, l_j, \phi_j\}$:

$$u_0(x) = \sum_{j=1}^{J} A_j \sin(2\pi\, l_j\, x/L + \phi_j), \tag{3.21}$$

where $A_j \sim U[-0.5, 0.5]$, and $\phi_j \sim U[0, 2\pi)$, with l_j chosen from the set $\{1, 2, 3\}$. The domain is discretised uniformly over $n_x = 256$ points with periodic boundaries and set $t \in [0, 1]$ at $n_t = 640$.

3.7 Training Approach

The training, validation, and test data consist of **1024**, **128**, and **128** samples, respectively. During training, we implement **multi-rollout** (see 4.4.2 for further details) to improve long-term stability. In addition, we implement **multistep prediction** during inference by increasing the model's output size, i.e., predicting multiple time steps per rollout (see 4.4.2 for further details). We use the **Adam** optimiser to explore different initial learning rates ($\{1e-2, 1e-3, 1e-4\}$), and an initial learning rate of **1e − 3** with **scheduled decay** performed best. The model was tested at each epoch during training, and we implemented early stopping to prevent overfitting. We monitor the validation loss and use the epoch with the lowest validation loss for testing.

3.8 Metrics

Throughout this work, model performance is evaluated using the normalised root mean squared error (nRMSE), following the convention adopted in the PDEBench benchmark suite (Takamoto *et al.*, 2022). The nRMSE is a dimensionless error measure obtained by normalising the root mean squared error by the norm of the ground-truth solution, which makes errors comparable across PDEs that span very different physical scales. For a predicted field $\tilde{u}_{pred}$ and a ground-truth field u_{true} it is defined as

$$\text{nRMSE} = \frac{\|\tilde{u}_{pred} - u_{true}\|_2}{\|u_{true}\|_2} = \frac{\sqrt{\sum_{j=1}^{J} \left(\tilde{u}_{pred}^{(j)} - u_{true}^{(j)}\right)^2}}{\sqrt{\sum_{j=1}^{J} \left(u_{true}^{(j)}\right)^2}}, \tag{3.22}$$

where J is the spatial resolution, $\|\cdot\|_2$ denotes the L_2 norm, and u_{true} and $\tilde{u}_{pred}$ are the ground-truth and predicted values, respectively. Because the error is

normalised, it can be interpreted directly as a relative (percentage) error, and its quadratic form makes it sensitive to large pointwise discrepancies in the solution.

3.9 Baseline FNO Characterisation

Before developing the training and architectural contributions of the following chapters, we briefly characterise the baseline single-step FNO on the viscous Burgers' equation (Section 3.6.2). The purpose is threefold: to fix the reference configuration used as a starting point in later chapters, to confirm that our implementation reproduces the hyperparameter-scaling behaviour reported for the FNO in the literature (Kovachki *et al.*, 2021; Li *et al.*, 2020b), and to expose the limitation of single-step mapping that motivates the autoregressive formulation used thereafter. These are validation studies rather than novel results, and we therefore report only the outcomes that carry forward and omit the full experimental detail.

3.9.1 Training objective and baseline configuration

We first compared training objectives on the low-frequency ("easy") viscous Burgers' data of Li *et al.* (2020b), whose initial conditions are drawn from a Gaussian measure $\mu = \mathcal{N}(0, C)$ with covariance $C = 625\,(-\Delta + 25I)^{-2}$ and are dominated by high-magnitude, low-frequency components. Consistent with the scale-invariance argument that motivated the choice of nRMSE as the error metric, training with the normalised root mean squared error (nRMSE) objective outperformed the mean-squared error (MSE): the test nRMSE fell from 0.0015 to 0.0006 (a 60% reduction), and the nRMSE-trained model remained superior even when evaluated under MSE ($0.0030 \rightarrow 0.0021$). We therefore adopt nRMSE as the training objective throughout the thesis. Varying the remaining hyperparameters reproduced the expected FNO scaling: the error is resolution-invariant above a minimum discretisation of 64 points; it is minimised at a channel width $d_v = 128$; the number of Fourier modes required tracks the viscosity ($k_{\max} = 8$ for $\nu = 10^{-2}$ and $k_{\max} = 16$ for $\nu = 10^{-3}$, since lower viscosity carries more high-frequency content, with additional modes causing over-fitting); and the error decreases monotonically with the number of training samples, whereas additional layers help only up to roughly three for the higher viscosity. A GeLU activation outperformed ReLU. These trends match those documented for the FNO (Kovachki *et al.*, 2021; Li *et al.*, 2020b) and are summarised in Table 3.6.

Hyperparameter	Baseline choice / observed behaviour
Training objective	nRMSE (60% lower test error than MSE)
Spatial resolution	Resolution-invariant above 64 points
Channel width d_v	128 (minimum test error)
Fourier modes $k_{\max}$	8 for $\nu = 10^{-2}$; 16 for $\nu = 10^{-3}$
FNO layers	≈ 3 (high ν); up to 8 (low ν)
Training samples	Error decreases monotonically with sample count
Activation	GeLU (outperforms ReLU)

Table 3.6: Baseline single-step FNO configuration on the low-frequency viscous Burgers' data, and the hyperparameter-scaling behaviour used to select it. The trends reproduce those reported for the FNO (Kovachki *et al.*, 2021; Li *et al.*, 2020b).

Augmenting the input with the viscosity, and subsequently the evaluation time, as additional coordinates allowed a single FNO to map any initial condition to its solution across a range of viscosities and times, with test errors of 0.0016 and 0.0018 respectively. The model interpolated accurately within the training distribution, including at viscosities unseen during training, but extrapolation degraded most for short times and low viscosities, where high-frequency content dominates. The single-step FNO therefore approximates the parametric solution operator well inside its training distribution but does not, by itself, extrapolate reliably.

3.9.2 Single-step mapping to autoregressive prediction

The single-step FNO is highly accurate on the low-frequency data above, but this masks a limitation that the harder, PDEBench-style benchmark of Takamoto *et al.* (2022) makes explicit. On that data, whose initial conditions superpose higher-frequency modes, the test error is one to two orders of magnitude larger and deteriorates sharply as the viscosity decreases (Table 3.7). Of all the hyperparameters above, only increasing the training set gave a material improvement, and even a fourfold increase (from 1000 to 4000 samples, reducing the test error from 0.119 to 0.055) left it an order of magnitude above the easy case. This gap, together with the accumulation of error that arises when a single-step map is applied recursively over long horizons, motivates the shift from single-step mapping to *autoregressive* prediction that underpins the remainder of the thesis: the

training strategy of Chapter 4 and the architecture of Chapter 5.

Dataset	ν	**Train MSE**	**Test MSE**
Easy (Li *et al.*, 2020b)	0.1	6.0×10^{-5}	1.26×10^{-3}
	0.01	6.0×10^{-5}	1.32×10^{-3}
Hard (Takamoto *et al.*, 2022)	0.1	4.03×10^{-4}	2.65×10^{-2}
	0.01	1.57×10^{-3}	1.19×10^{-1}

Table 3.7: Single-step FNO train/test error on the low-frequency ("easy") and PDEBench-style ("hard") viscous Burgers' benchmarks. The one-to-two order-of-magnitude gap, which widens as the viscosity decreases, is what motivates the autoregressive treatment developed in the following chapters.

Chapter 4

Curriculum Learning for Constant-timestep Models

In this chapter, we examine the challenges of training constant-timestep models, i.e., deep autoregressive models that predict at a fixed timestep. These models utilise previous predictions to produce next-step predictions, thereby recovering the time evolution of the PDE. We identify a bottleneck in baseline training strategies and propose a novel curriculum-based approach that progressively increases the complexity of learning. This improves prediction accuracy and stability across the various partial differential equations investigated.

4.1 Introduction

Neural operators offer a promising approach to address the computational cost of traditional numerical methods for solving time-dependent partial differential equations (PDEs), especially for modelling fluid flow (Kovachki *et al.*, 2024). They learn an operator that maps input to output functions, meaning they can learn a mapping that is independent of data discretisation and predict the solution function for unseen conditions during inference at different resolutions. This offers greater flexibility than traditional neural networks (e.g., Convolutional Neural Networks (CNNs)) and the potential to achieve a significant speed-up over the classical approach, especially in tasks like design optimisation that require evaluation across several design parameters (Lu *et al.*, 2022b).

Deep autoregressive models based on neural operators have emerged as a

popular choice for modelling the time evolution of partial differential equations (PDEs). These **constant-timestep models** operate explicitly by using the solution from a previous state to predict the next at a fixed timestep. Unlike recurrent neural networks (RNNs), which carry an internal hidden state, that is, a learned latent vector recurrently propagated from one timestep to the next to summarise the sequence history, autoregressive neural operators maintain no such persistent latent state: each prediction is conditioned only on the (explicit) physical solution field fed in at that step. This absence of a recurrent hidden state makes them significantly easier to train by bypassing the vanishing gradient problem typically caused by backpropagating through long recurrent sequences (Pascanu *et al.*, 2013). Furthermore, they are less susceptible to **exposure bias**, the compounding error that arises from discrepancies between training-phase ground truths and testing-phase predictions.

Despite their advantages, long-term autoregressive prediction of time-dependent PDEs using neural operators remains challenging for several reasons:

- **Multiscale Complexity:** PDE dynamics can be complex, often involving a sophisticated interplay of multiple spatial and temporal scales, alongside inherently unsteady long-term behaviours.
- **Gradient Accumulation:** Training for autoregressive stability is computationally intensive; "long-rollout" training can trigger gradient accumulation issues, leading to numerical instability and out-of-memory (OOM) errors (List *et al.*, 2025).
- **Exponential Error Propagation:** During inference, small per-step inaccuracies can compound over large autoregressive rollouts. This error propagation can grow exponentially, causing the predicted trajectory to diverge significantly from the ground truth (Gonzalez *et al.*, 2023; McCabe *et al.*, 2023).

To mitigate these issues, recent research has focused on adapting the training process to improve robustness. Because error propagation is largely **model-agnostic**, even advanced architectures require explicit design choices to handle noise and error accumulation. Current strategies simulate the "exposure bias" during training. For instance, Lippe *et al.* (2024) introduced a noise-removal refinement process to improve accuracy, while Sanchez-Gonzalez *et al.* (2021) demonstrated that injecting noise directly into the input during training significantly improves the model's long-term stability.

In this chapter, we focus on a promising training approach based on **curriculum learning**, which has improved accuracy and generalisation in sequence-to-sequence tasks using RNNs (Vlachas & Koumoutsakos, 2024). This strategy

utilises a metric to distinguish between "easy" and "hard" data samples, and employs a scheduled curriculum to prioritise simpler tasks before introducing complexity. In the context of modelling sequential data with RNNs, current scheduling curricula transition from **teacher forcing**, where the model is "fed" ground-truth inputs to simplify the learning objective, to **autoregressive rollout**, which is more challenging but better aligns with long-rollout stability. While successful in the RNN domain, this training paradigm remains largely under-investigated for autoregressive neural operators. We investigate existing curriculum learning strategies for autoregressive neural operators (FNO Li *et al.* (2021) and U-NO Rahman *et al.* (2023)) in the context of time-dependent PDEs. Based on this preliminary investigation, we observe that existing curricula can overfit to early dynamics (small rollout), resulting in poor performance on later dynamics (large rollout).

Therefore, we propose a new training strategy, called **Scheduled Temporal Loss Weighing (STLW)**. It is a curriculum-based learning approach that performs only autoregressive rollout training, thereby avoiding the easy, suboptimal teacher-forcing training. Existing methods that use only autoregressive rollout training focus on optimising short-term gradient accumulation in the early stages of training to avoid instabilities from long-term gradients, thereby neglecting long-term gradient accumulation. Instead, STLW prioritises short-term gradient accumulation without neglecting long-term gradients during the initial training phase. We achieve this by assigning a weight to the loss at each timestep of the autoregressive rollout, governed by a transition function that dynamically rebalances the weights as training progresses.

Specifically, STLW assigns lower weights to later timesteps during early training, reducing their contribution to the total loss magnitude while maintaining their influence on the total gradient accumulation. As training progresses, these weights increase according to a smooth, continuous schedule, eventually converging to a uniform weighting (i.e., weight ≈ 1) by the end of the training process (see Figure 4.6). This scheduling facilitates a more stable optimisation landscape, ultimately yielding higher accuracy and superior generalisation compared to existing methods. In summary, the key contributions of this chapter are as follows:

1. **Consolidation** of existing curriculum learning strategies for modelling PDE time evolution via autoregressive neural operators (constant-timestep models).

2. **Development** of Scheduled Temporal Weighted Loss (STLW), a novel approach that bridges the gap between training using easy one-rollout and hard full-rollout.

3. **Benchmarking** of STLW against existing curriculum learning strategies across multiple PDE datasets.
4. **Demonstration** of the superior accuracy and robustness provided by the STLW framework.

4.2 Problem Statement

For completeness, and to fix the notation used throughout this chapter, we briefly recapitulate the explicit time-stepping formulation introduced in the Methods chapter (Chapter 3). We formulate the learning problem by considering explicit time-stepping for a general class of time-dependent PDEs over the domain $(t, x) \in (0, T] \times D$. The governing equations, subject to boundary conditions B, are defined as:

$$\begin{aligned} \partial_t u(t,x) &= \mathcal{F}^\dagger\left(u(t,x)\right) \qquad (t,x) \in [0,T] \times D, \\ u(0,x) &= u_0(x) \qquad x \in D, \\ u(t,x) &= B \qquad (t,x) \in (0,T] \times \partial D, \end{aligned} \tag{4.1}$$

where $u : [0, T] \times D \to \mathbb{R}^d$ is the solution function, $u_0(x)$ is the initial condition at $t = 0$ and , and $\mathcal{F}^\dagger$ denotes a spatial differential operator.
Our objective is to learn a forward-explicit autoregressive neural operator, $\mathcal{F}_\theta$ with parameters $\theta \in \mathbb{R}^p$, that maps an input function $u(t)$ to the predicted solution at the subsequent timestep:

$$\tilde{u}\left(t + \Delta t\right) = \mathcal{F}_\theta\left(u(t)\right) \quad \theta \in \mathbb{R}^p. \tag{4.2}$$

The learning problem seeks to identify the optimal parameters θ^* by minimising the discrepancy between the ground truth u and the prediction $\tilde{u}$. Training is performed using a finite set of time-evolution trajectories generated by a high-accuracy numerical scheme. Each trajectory is represented as an evenly spaced temporal sequence with a uniform spatial resolution. To assess performance, we use an empirical train-test split and evaluate the model on unseen test samples.

During inference, the trained model $\mathcal{F}_{\theta^*}$ is applied recursively to generate future solutions at a constant timestep Δt. We can therefore reformulate the model from Equation 4.2 to define a sequence of T autoregressive rollouts, yielding the solutions $\{\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_T\}$:

$$\tilde{u}_t = \mathcal{F}_{\theta^*}(\tilde{u}_{t-1}), \quad 1 \leq t \leq T. \tag{4.3}$$

4.3 Related Work

Recent efforts to enhance the stability and accuracy of PDE solvers can be broadly categorised into architectural modifications, training paradigms, and the integration of inductive biases. While many of these methods focus on short-term accuracy, maintaining long-term stability remains a primary bottleneck. This section reviews the current landscape of long-term prediction strategies and the evolution of neural operators as resolution-independent models.

4.3.1 Long-term Prediction

Approaches to encourage stable, long-term prediction of PDEs generally fall into three categories. The first involves integrating additional modules into existing architectures. Examples include the use of Physics-Informed Neural Networks (PINNs) to account for long-term temporal dependencies via a physics-based loss (Wang & Perdikaris, 2023), Recurrent Neural Networks (RNNs) to capture temporal dynamics (Michałowska *et al.*, 2024), transfer learning for enhanced stability (Xu *et al.*, 2023), and denoising techniques to refine predictions (Lippe *et al.*, 2023).

The second category focuses on training strategies designed to improve robustness without altering the underlying model. These include adversarial training to defend against error accumulation (Sanchez-Gonzalez *et al.*, 2021), temporal bundling, which predicts multiple timesteps simultaneously to reduce the cumulative number of forward passes (Brandstetter *et al.*, 2022c), latent evolution coupled with latent loss (Wu *et al.*, 2022), and the "push-forward" trick, which exposes the model to its own noise during training (Brandstetter *et al.*, 2022c).

The third category modifies existing frameworks with inductive biases, such as attention mechanisms for capturing multi-scale temporal dependencies (Peng *et al.*, 2022), U-Net-based architectures for resolving small-scale dynamics (Li *et al.*, 2023), or implicit neural networks that guarantee stability, often at the expense of accuracy (Migus *et al.*, 2023). Our work aligns with the second approach, focusing on enhancing training paradigms for existing architectures.

4.3.2 Neural Operators

A detailed treatment of operator-learning models, including the Fourier Neural Operator (FNO) (Li *et al.*, 2021) and the U-shaped Neural Operator (U-NO) (Rahman *et al.*, 2023), is given in the Literature Review (Chapter 2); we adopt these architectures here because they generalise across spatial resolutions, unlike grid-tied convolutional models. Research into the effectiveness of such operators

specifically for autoregressive time-stepping remains limited: standard architectures often struggle to capture the long-term dynamics of complex PDEs and typically require extensive modification to maintain accuracy over long horizons. In this study, we therefore use these state-of-the-art neural operators as the backbone on which to demonstrate the effectiveness of our proposed training strategy.

4.3.3 Curriculum Learning

Curriculum Learning (CL), first proposed by Bengio *et al.* (2009), is based on the hypothesis that training a model by transitioning from "easy" to "hard" samples improves convergence speed and generalisation. A CL framework is defined by a "curriculum metric" that determines sample difficulty and a "scheduler" that controls the progression of complexity during training. CL is broadly classified into automatic and predefined approaches; the latter, which relies on prior knowledge of task complexity or noise, is the focus of this work.

The Application of CL to PDEs is an emerging area, with most studies transitioning between teacher forcing (easy) and autoregressive rollout (difficult). Teutsch & Mäder (2022) investigated various CL schedulers for RNNs, demonstrating that CL-based training outperforms standard methods, though identifying an optimal schedule often requires an expensive grid search. Vlachas & Koumoutsakos (2024) further highlighted the benefits of CL for extended rollouts, effectively mitigating error propagation. Similarly, Takamoto *et al.* (2023) proposed a scheduler that significantly reduced errors during autoregressive inference.

Despite its prevalence, teacher forcing can lead to suboptimal long-term performance due to exposure bias (Hess *et al.*, 2023). Consequently, we avoid teacher forcing entirely, formulating our curriculum approach exclusively within the autoregressive rollout regime.

4.4 Baseline Training Strategies

Training a neural operator for long-term prediction presents significant technical challenges. Consider a dataset comprising $T+1$ evenly spaced temporal samples ($0 \leq t \leq T$). An autoregressive model, as defined in Equation 4.3, is trained to predict T timesteps during inference. This process requires an initial condition (or ground-truth solution) $u_{t=0}$ to predict the subsequent state $\tilde{u}_1$, which is then fed back into the model recursively to predict $\tilde{u}_2$ through $\tilde{u}_T$. This iterative approach is called autoregressive prediction.

To facilitate this, the autoregressive rollout (AR) strategy is frequently employed during training (Brandstetter *et al.*, 2022c; List *et al.*, 2025). This approach predicts T timesteps by performing T successive rollouts, where each predicted solution serves as the input for the next step. The total loss used to optimise the model weights is the summation of losses across all T predictions. Because each prediction depends on the preceding one, the resulting computational graph is deep, leading to significant gradient accumulation. Consequently, full autoregressive rollout for large T is often computationally prohibitive due to memory constraints and high processing costs (List *et al.*, 2025).

To mitigate these issues, a common strategy is to train on shorter subsequences of length $\widehat{T}+1$, where $\widehat{T}$ is significantly smaller than the total sequence length ($\widehat{T} \ll T$). To effectively utilise the available data, multiple batches of $(\widehat{T}+1)$-length subsequences are sampled, each starting at a different temporal index within the original sequence. Based on this, we define two baseline training strategies, illustrated in Figure 4.1: Teacher Forcing (TF) and Autoregressive Rollout (AR). In this figure, the model is depicted as a circle, colour-coded by the input source used at each prediction step: green indicates the model is conditioned on ground-truth data, while blue indicates it utilises its own previous prediction.

Teacher Forcing (TF) is a one-step rollout strategy ($\widehat{T}=1$) that predicts only the immediate next timestep. For any $t \in \{0, \ldots, T-1\}$, TF takes the ground-truth solution u_t to predict $\tilde{u}_{t+1}$. This approach is computationally efficient as it avoids the gradient accumulation associated with multi-step rollouts ($\widehat{T}>1$). However, TF can be unstable during inference due to *exposure bias*, the amplification of small errors or noise that the model did not encounter during training. This often leads to solution divergence over long horizons. To address this, Gaussian noise of varying magnitudes can be added to the ground-truth inputs to improve robustness; we refer to this variant as teacher forcing with noise (TF+N).

Fixed-length Autoregressive Rollout (AR) involves predicting a short subsequence where $1 < \widehat{T} \ll T$. This approach reduces computational overhead relative to a full rollout of length T while partially mitigating the discrepancy between training and inference phases inherent in teacher forcing.

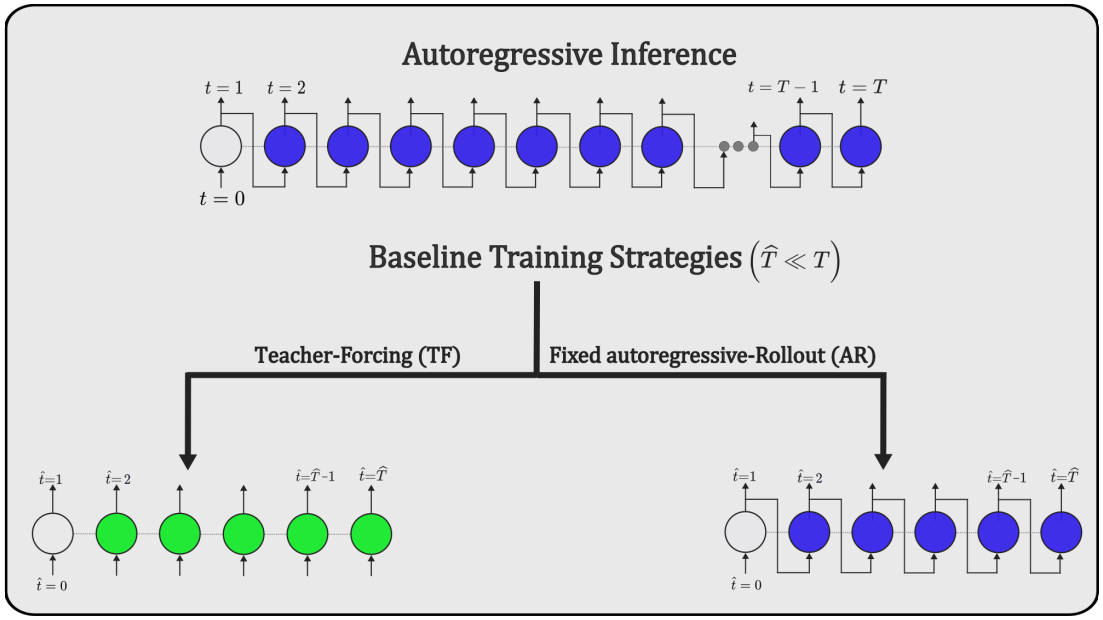


Figure 4.1: Baseline training strategies of $\widehat{T}$ timestep: Teacher forcing (one-rollout) and autoregressive rollout (multi-rollout), where T represents the total timesteps of the data and $\widehat{T}$ represent the training batch timesteps.

4.4.1 Gradient propagation during training

For a training batch consisting of $\widehat{T}$ rollouts from a dataset of $T+1$ temporal samples, the training procedure is defined as:

$$\theta^* = \arg\min_{\theta} \left[\sum_{t=0}^{T-\widehat{T}} \sum_{\hat{t}=1}^{\widehat{T}} l\left(u_{t+\hat{t}}, \tilde{u}_{t+\hat{t}}\right) \right]. \tag{4.4}$$

We iterate over each timestep from $t=0$ to $T-\widehat{T}$. The initial solution u_t is fed into the model to autoregressively predict the subsequent $\widehat{T}$ steps, where the total loss is the sum of the individual losses at each timestep $\hat{t}$. For a given starting point t, the loss L_t is defined as:

$$L_t = \sum_{\hat{t}=1}^{\widehat{T}} l_{\hat{t}} = \sum_{\hat{t}=1}^{\widehat{T}} l(u_{\hat{t}}, \tilde{u}_{\hat{t}}). \tag{4.5}$$

For convenience, assuming $\widehat{T} \to T$ such that $\hat{t} = t$, we can simplify Equation 4.5 to:

$$L = \sum_{t=1}^{T} l_t = \sum_{t=1}^{T} l(u_t, \tilde{u}_t). \tag{4.6}$$

The gradients associated with this loss, $\frac{\partial L_t}{\partial \theta}$, are obtained by backpropagating through all steps. The gradient of the loss at each timestep t can be expressed via the chain rule as:

$$\frac{\partial l_t}{\partial \theta} = \sum_{tt=1}^{t} \left[\frac{\partial l_t}{\partial \tilde{u}_t} \left(\prod_{ttt=1}^{t-tt} \frac{\partial \tilde{u}_{t-ttt+1}}{\partial \tilde{u}_{t-ttt}} \right) \frac{\partial \tilde{u}_{tt}}{\partial \theta} \right]. \tag{4.7}$$

To illustrate, when $T = 3$, the total loss is $L_3 = l_1 + l_2 + l_3$. Using Equation 4.7, the gradients for each component with respect to the model parameters θ are:

$$\frac{\partial l_1}{\partial \theta} = \frac{\partial l_1}{\partial \tilde{u}_1} \frac{\partial \tilde{u}_1}{\partial \theta}, \tag{4.8}$$

$$\frac{\partial l_2}{\partial \theta} = \frac{\partial l_2}{\partial \tilde{u}_2} \frac{\partial \tilde{u}_2}{\partial \theta} + \frac{\partial l_2}{\partial \tilde{u}_2} \left(\frac{\partial \tilde{u}_2}{\partial \tilde{u}_1} \right) \frac{\partial \tilde{u}_1}{\partial \theta}, \tag{4.9}$$

$$\frac{\partial l_3}{\partial \theta} = \frac{\partial l_3}{\partial \tilde{u}_3} \frac{\partial \tilde{u}_3}{\partial \theta} + \frac{\partial l_3}{\partial \tilde{u}_3} \left(\frac{\partial \tilde{u}_3}{\partial \tilde{u}_2} \right) \frac{\partial \tilde{u}_2}{\partial \theta} + \frac{\partial l_3}{\partial \tilde{u}_3} \left(\frac{\partial \tilde{u}_3}{\partial \tilde{u}_2} \frac{\partial \tilde{u}_2}{\partial \tilde{u}_1} \right) \frac{\partial \tilde{u}_1}{\partial \theta}. \tag{4.10}$$

The behaviour of the loss gradient depends significantly on the product of Jacobians between consecutive solutions (e.g., $\tilde{u}_k$ and $\tilde{u}_{k-1}$). Consequently, there is a direct relationship between the gradient stability and the underlying PDE dynamics. For chaotic systems, Mikhaeil *et al.* (2022) demonstrated that the geometric mean of the Jacobian eigenvalues can exceed unity, leading to:

$$\left\| \prod_{ttt=1}^{t-tt} \frac{\partial \tilde{u}_{t-ttt+1}}{\partial \tilde{u}_{t-ttt}} \right\| > 1. \tag{4.11}$$

As T increases, the gradient magnitude grows accordingly; for large rollouts, this leads to unstable training and gradient divergence. In the limit $T \to \infty$, the Jacobian magnitude approaches infinity:

$$\lim_{T \to \infty} \left\| \frac{\partial u_T}{\partial u_1} \right\| = \infty. \tag{4.12}$$

This induces the **exploding gradient** phenomenon, which severely hinders the training of autoregressive models. Furthermore, theoretical evidence suggests that these challenges are inherent to the temporal dependencies and cannot be fully mitigated by architectural changes alone. Therefore, strategic management of gradients during training is critical.

4.4.2 Baseline Rollout Analysis: The Stability Gap

Having defined the teacher-forcing (TF) and autoregressive-rollout (AR) baselines above, we now establish empirically the "stability gap" that motivates the curriculum developed in this chapter. Using the viscous Burgers' equation as a representative dissipative benchmark, we show (i) that a model trained with a single-step objective attains high one-step accuracy yet diverges under autoregressive inference, and (ii) that, among candidate operator backbones, the Fourier Neural Operator (FNO) with a balanced history/horizon window provides the strongest baseline. The higher-frequency, more diverse initial-condition regime adopted for the main experiments (Chapter 3) is substantially more challenging than the low-frequency configurations used in early operator studies, which makes closing this stability gap the central objective of the remainder of the chapter. The FNO backbone and the resulting "Many-to-Many" training window identified here are used throughout the experiments of Section 4.7.

Training: one-step versus multi-step rollout

We empirically demonstrate the advantages of training with multi-step rollouts compared to the standard one-step approach. As reported in Table 4.1, while models trained using a one-step objective achieve a comparably low Mean Squared Error (MSE) during teacher-forced (one-step) inference, their performance deteriorates significantly during a full autoregressive rollout.

This "stability gap" is a direct result of exposure bias: in one-step training, the model is always conditioned on the ground-truth manifold, whereas during inference, small numerical errors in the initial predictions $(t+1, t+2)$ compound. These errors push the model into out-of-distribution states from which it cannot recover, leading to the catastrophic error growth observed in the baseline configurations.

To mitigate this, the multi-step rollout training strategy was implemented, with its effects detailed in Table 4.2. By increasing the training horizon from a single step to 16, we introduce a temporal inductive bias that penalises the model for error accumulation during training. The full-rollout MSE column of Table 4.2 demonstrates a decrease in inference error as the training rollout increases, confirming that the FNO learns a more stable representation of the underlying dynamical system when optimised over longer sequences.

However, this gain in stability comes at a clear computational cost. The training-time-per-epoch column in Table 4.2 tracks the training time per epoch, which scales linearly with the rollout length. This creates a computational-to-accuracy trade-off, while the 16-step rollout provides the highest fidelity.

Architecture	One-step (teacher-forced) MSE	Full-rollout MSE
FNO	$1e-6$	0.381
U-NO	$1.0e-6$	0.319
LSM	$1.9e-6$	0.361

Table 4.1: **Comparison of one-step prediction error versus full-autoregressive-rollout error.** The table reports the discrepancy between local accuracy (predicting $t+1$ given the ground-truth state) and global stability for the neural operator architectures considered. While all models achieve a comparably low Mean Squared Error (MSE) on the one-step teacher-forced task, the divergence in the full-rollout error highlights the impact of error accumulation and exposure bias over long-term temporal evolution. The full-rollout values correspond to the single-input/single-output ($H = 1$, $P = 1$) configuration; the teacher-forced one-step entries are to be populated from the source experiment logs.

Training rollout length	Full-rollout MSE	Training time per epoch (s)
1	0.38	181
2	0.33	239
4	0.37	415
8	0.31	830
16	0.28	1367

Table 4.2: **Trade-off between inference accuracy and computational cost across different training rollout lengths.** The full-rollout Mean Squared Error (MSE) decreases consistently as the training horizon increases from 1 to 16 steps, while the training time per epoch increases approximately linearly with the rollout length. Longer rollouts therefore yield gains in stability and accuracy at the expense of computational overhead, which supports the use of a larger training window.

Inference: multi-step prediction

We provide a justification for using a multi-step prediction strategy. We analyse how the temporal window size, both in terms of historical context (input) and prediction horizon (output), affects the stability and accuracy of the Fourier Neural Operator (FNO). The results reveal a relationship between history and future rollout performance, discussed below.

Table 4.3 demonstrates that the best performance is achieved when the model uses a large historical context to predict a long future horizon simultaneously. The configuration with **25 input steps and 25 output steps** yields the lowest rollout MSE (**0.012**). Conversely, the model exhibits extreme instability when a long history is used to predict a single future step, as evidenced by a worst-case error of **167.708** in the 25-input, 1-output configuration.

1. **The Multi-Step Prediction Advantage:** A clear trend emerges: for any given number of input timesteps, increasing the number of output timesteps during training consistently reduces the rollout error. For instance, with 10 input steps, increasing the prediction horizon from 1 to 25 steps reduces the error from 1.037 to 0.0496.

This suggests that training the model to predict multiple steps at once acts as a form of **temporal regularisation**. By forcing the model to optimise for a sequence rather than a single point, the FNO is incentivised to learn the underlying continuous global dynamics of the PDE rather than high-frequency local gradients that may lead to numerical instability during autoregressive rollout.

2. **The Instability of "Many-to-One" Mapping:** The most striking result is the failure of the model when the input history significantly outweighs the prediction horizon (the "Many-to-One" approach). While one might expect more history to provide better context, the error jumps from 0.381 (1 input) to 167.708 (25 inputs) when the output is kept at a single step.

   This divergence is likely due to the model becoming overly sensitive to the temporal correlations in the input sequence. Without a corresponding long-term prediction loss to "anchor" the learning process, the model may overfit to the 25-step input window in a way that creates a catastrophic feedback loop during inference. This highlights a critical limitation: **more history is beneficial only if the training objective also encompasses long-term evolution.**

3. **Balance Between History and Horizon:** The data suggests a **temporal balance** is required. As the history (input) increases, the model's capacity to represent the system's state increases, but it requires a larger prediction horizon (output) to translate that capacity into stable rollouts. When the history and horizon are balanced at 25 steps each, the model achieves an order-of-magnitude improvement over the 1-to-1 baseline (0.012 vs 0.381). This suggests that, for complex PDEs, capturing long-range dependencies in both directions is essential to maintaining the spectral integrity of the neural operator.

In summary, increasing history alone is insufficient for stability. The FNO requires a "Many-to-Many" training strategy to effectively map the evolution of the PDE. The catastrophic failure in the 25-to-1 case indicates that providing more information to a neural operator without a sufficiently challenging training horizon can lead to severe out-of-distribution errors during autoregressive inference.

Input timesteps (History)	**Output timesteps (Prediction)**		
	1	10	25
1	0.381	0.380	0.328
10	1.037	0.100	0.0496
25	<u>167.708</u>	0.017	**0.012**

Table 4.3: Impact of temporal training windows on FNO rollout stability. The table compares rollout Mean Squared Error (MSE) for models trained with varying input history (H) and output prediction horizons (P). The results highlight that increasing the prediction horizon acts as a temporal regularizer, with the "Many-to-Many" configuration ($H = 25, P = 25$) achieving the lowest error (bold), while the "Many-to-One" configuration ($H = 25, P = 1$) results in catastrophic instability (<u>underlined</u>).

Figure 4.2 compares the Mean Squared Error (MSE) of three different models—Fourier Neural Operator (FNO), U-shaped Neural Operator (U-NO), and Latent Spectral Model (LSM), as the number of multisteps is increased. In this experiment, the number of input time steps (history) and the number of future time steps to predict are kept equal at 1, 25, 50, and 125.

At the smallest window of 1 step, all three models exhibit a relatively high MSE in the range of 10^{-1}. FNO has an error of approximately 0.381, LSM is at 0.361, and U-NO is slightly lower at 0.319. As the history and prediction window increases to 25 and 50, all three models show a dramatic improvement, with errors dropping into the 10^{-2} range. The most notable trend appears at 125 multisteps. While U-NO and LSM see a modest further decrease in error to 0.018 and 0.015, respectively, the FNO model's error continues to reduce, reaching a minimum of 0.002. This is an order of magnitude lower than the other two models.

The results in Figure 4.2 provide a strong justification for selecting the FNO model. While all models benefit from a "Many-to-Many" training strategy, FNO demonstrates superior scaling behaviour. Its capacity to reduce error does not plateau in the same way as U-NO or LSM when faced with a long temporal horizon (125 steps). This indicates that the FNO architecture can capture and utilise long-term temporal dependencies across a large historical context to make highly accurate, stable long-range predictions.

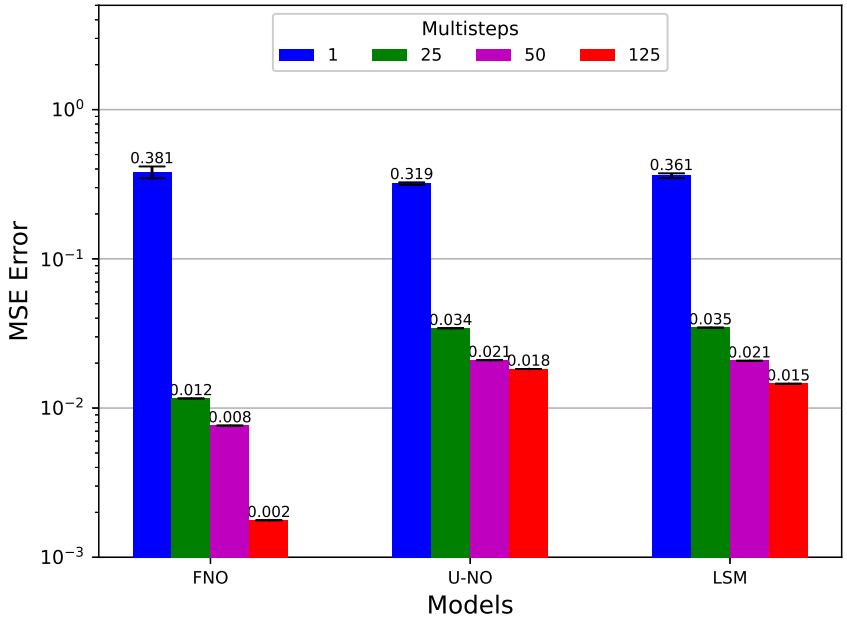


Figure 4.2: Comparative analysis of rollout MSE across different neural operator architectures as a function of temporal horizon. The plot illustrates the scaling behaviour of the FNO, U-NO, and LSM models trained with equal input (history) and output (prediction) windows of lengths 1, 25, 50, and 125 steps. Note the superior performance of the FNO model, which maintains higher stability and lower error at longer horizons compared to U-NO and LSM.

4.5 Curriculum-based training

Curriculum learning (CL) is a training strategy that can improve the stability and accuracy of autoregressive models by mimicking the incremental nature of human learning (Bengio *et al.*, 2009). The primary objective of CL is to start with simple tasks and gradually introduce more challenging samples, thereby increasing the difficulty of the optimisation objective. Formulating a curriculum requires defining a *curriculum metric* and a *scheduler*: the metric quantifies the difficulty of a given sample, while the scheduler dictates how difficulty increases during training.

In the context of sequence prediction, the "easy" task is typically defined as Teacher Forcing (TF). Under this regime, the model $\mathcal{F}_\theta$ generates a prediction

$\tilde{u}_{\hat{t}}$ using the ground-truth solution $u_{\hat{t}-1}$ as input. Conversely, the "difficult" task is the autoregressive (AR) rollout, where the model $\mathcal{F}_\theta$ predicts $\tilde{u}_{\hat{t}}$ based on its own previous prediction $\tilde{u}_{\hat{t}-1}$. This latter approach exposes the model to its own errors during training, better aligning the training objective with the inference phase (see Figure 4.1).

Consequently, the curriculum metric is often defined as the proportion of TF to AR steps within a fixed training rollout of length $\widehat{T}$. A higher AR-to-TF step ratio indicates a more challenging task, whereas a higher proportion of TF steps simplifies the learning objective. Several studies have proposed curriculum strategies to manage the transition from TF to AR. In the following sections, we expand upon this paradigm by categorising transitions at two levels: per-batch (within a training rollout) and per-epoch.

4.5.1 Scheduler: Epoch-level

For each epoch e (one complete pass through the dataset), we define a parameter $\alpha_e \in [0, 1]$ representing the AR ratio, the number of autoregressive steps relative to the training rollout length $\widehat{T}$. An $\alpha_e = 1$ corresponds to full AR, while $\alpha_e = 0$ indicates full TF across all rollouts. We define a curriculum transition function $C : \mathbb{N} \to [\alpha_{start}, \alpha_{end}]$ that governs the evolution of α_e. This function C can take various analytical forms, such as linear, sigmoid, or exponential.

$$\alpha_e = C(e). \tag{4.13}$$

As illustrated in Figure 4.3, this framework allows for two primary training trajectories:

TF–AR Curriculum ($\alpha_{start} \leq \alpha_e \leq \alpha_{end}$)

In the increasing configuration ($\alpha_{start} < \alpha_{end}$), the curriculum transitions from a high proportion of TF (low AR) at the beginning of training to a high proportion of AR (low TF) as training progresses.

AR–TF Curriculum ($\alpha_{start} \geq \alpha_e \geq \alpha_{end}$)

In the decreasing configuration ($\alpha_{start} > \alpha_{end}$), the model transitions from an initially high AR ratio to a higher TF proportion by the final epoch.

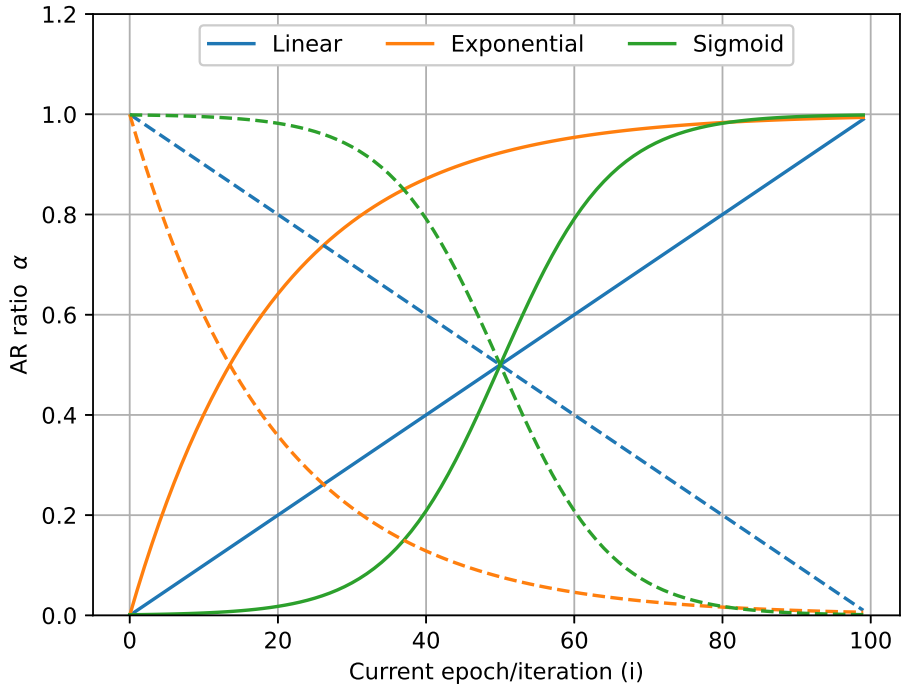


Figure 4.3: Transition functions at the epoch scale: Analytical functions for $C(e)$, showing increasing (solid line) and decreasing (dashed line) variants.

4.5.2 Metric: Batch-level

Once the transition function C determines α_e, we must define how to distribute AR and TF steps across the $\widehat{T}$ rollouts within each training batch to satisfy α_e. For any timestep t, the input to the model is determined by a decision function $\phi_{e,t}$. If $\phi_{e,t} = 1$, the model utilises the previous prediction (AR); otherwise, it utilises the ground-truth solution (TF):

$$\tilde{u}_{t+1} = \begin{cases} \mathcal{F}_\theta(\tilde{u}_t), & \text{if } \phi_{e,t} = 1 \quad \text{(AR)} \\ \mathcal{F}_\theta(u_t), & \text{otherwise} \quad \text{(TF)}. \end{cases} \tag{4.14}$$

We classify the strategies for determining $\phi_{e,t}$ into two categories: deterministic and probabilistic. Figure 4.4 illustrates the resulting combinations of AR (blue) and TF (green).

Deterministic (D)

This approach explicitly sets a transition point between TF and AR within each sequence.

- **TF–AR-D-CL:** A deterministic curriculum starting as full TF and shifting towards AR over the course of training.
- **AR–TF-D-CL:** The inverse case, transitioning from initial AR to TF at the final epoch.

In the TF–AR curriculum, α_e determines the number of initial TF steps before switching to AR for the remainder of the rollout. The decision function ϕ is defined as:

$$\phi_{e,t} = \begin{cases} 1, & \text{if } \alpha_e \geq \frac{t}{\widehat{T}} \quad \text{(AR)} \\ 0, & \text{otherwise.} \end{cases} \tag{4.15}$$

Probabilistic (P)

In the probabilistic variant, α_e represents the probability that any given step in the rollout is executed in AR mode.

- **TF–AR-P-CL:** The probability of sampling AR steps increases over the training duration.
- **AR–TF-P-CL:** The sampling probability shifts from AR at the start to TF by the end of training.

The decision function $\phi_{e,t}$ is a discrete random variable drawn from a Bernoulli distribution:

$$\phi_{e,t} \sim \text{Bernoulli}(\alpha_e). \tag{4.16}$$

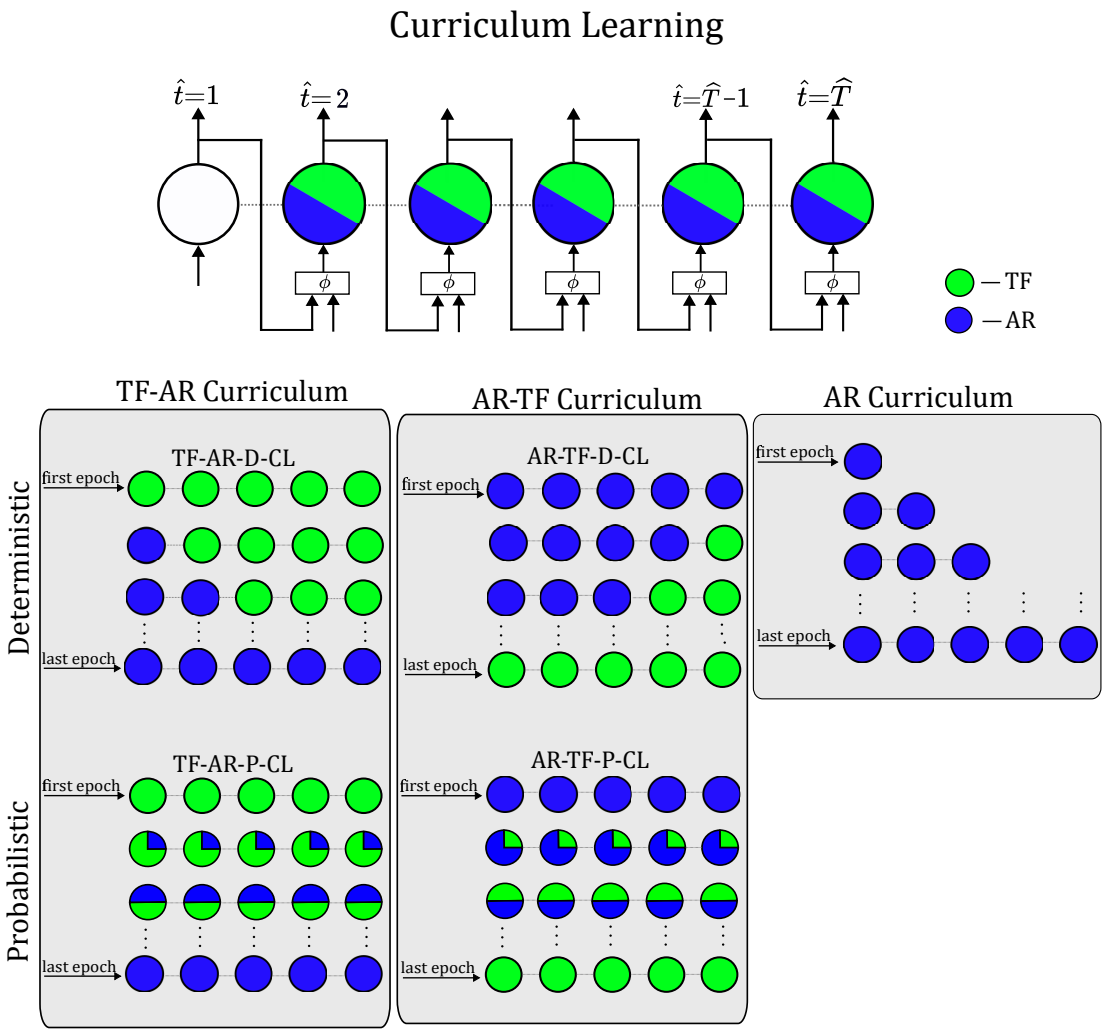


Figure 4.4: Combinations of Teacher Forcing (TF) and Autoregressive Rollout (AR) across training epochs.

Based on the epoch and batch scales, we can define different curricula, as shown in Figure 4.4. This includes choosing between increasing and decreasing α, corresponding to the TF-AR and AR-TF curricula, respectively. The per-epoch option includes deterministic (D) and probabilistic (P) modes that determine whether to select TF or AR at each timestep.

4.5.3 AR Curriculum

An alternative curriculum strategy involves using exclusively AR steps while varying the training rollout length, $\widehat{T}$ (AR-CL). In this configuration, the curriculum metric is redefined since Teacher Forcing is omitted. The distinction between "easy" and "hard" tasks is instead governed by the number of training rollouts, $\widehat{T}$. Given the relationship between the rollout length and the loss gradient discussed in Section 4.4.1, we can state that:

> As the training rollout $\widehat{T}$ increases, the magnitude of the loss gradient increases, thereby making the optimisation objective more difficult.

In this context, the AR ratio α represents the percentage of the maximum training rollout $\widehat{T}$. We define the **AR Curriculum** as the case where α increases over time, progressively extending the number of training rollouts as illustrated in Figure 4.4.

4.6 Scheduled Temporal Loss Weighting (STLW)

We present our proposed approach, Scheduled Temporal Loss Weighting (STLW): a curriculum learning strategy designed for the long-term autoregressive rollout of time-dependent PDEs. The core principle of temporal weighting is to enable the model to attend to all timesteps within the training batch simultaneously, albeit with varying degrees of attention governed by dynamic weights.

In autoregressive inference, errors from earlier timesteps accumulate and propagate through the sequence, potentially leading to exponential error growth. For PDEs with complex dynamics evolving over long timescales, such as transitions between smooth and sharp gradients, we hypothesise that prioritising earlier timesteps during the initial training phase reduces overall error propagation and enhances stability. However, it remains essential to maintain some contribution from all temporal dynamics throughout training to prevent overfitting to early states and to mitigate the gradient accumulation issues associated with long rollouts.

STLW alleviates the limitations of both short- and long-rollout training by treating the sequence length as a continuum of difficulty. We prioritise "easy" samples (early timesteps with lower cumulative error) before "hard" samples (later timesteps with higher error propagation).

4.6.1 Implementation

To achieve our training strategy, we control the loss contributed to each predicted timestep during autoregressive training rollouts. For sequence of length $T + 1$, where $t \in 0, 1, 2, \ldots, T$, we assign a weight $w \in [0, 1]$ to the loss l at each predicted timestep $t > 0$. As illustrated in Figure 4.5, the weights are represented by the colour intensity; later timesteps begin with low weights and gradually transition toward full weighting. The total loss is defined as the sum of these weighted individual losses.

For a training rollout of length $\widehat{T}$ at epoch e, where $\widehat{T} \ll T$, and a starting index $t \in \{0, \ldots, T - \widehat{T}\}$, the total weighted loss $\widehat{L}_t$ to be backpropagated is:

$$\widehat{L}_t = \sum_{\hat{t}=1}^{\widehat{T}} w_{e,\,t+\hat{t}} \cdot l(u_{t+\hat{t}},\, \tilde{u}_{t+\hat{t}}), \qquad \hat{t} \in \{1, \ldots, \widehat{T}\}. \tag{4.17}$$

4.6.2 Curriculum Metric

We define the curriculum metric $\bar{w}_e$ as the mean of all weights w assigned at epoch e. For a sequence of length T, the mean weight is:

$$\bar{w}_e = \frac{1 + \sum_{t=2}^{T} w_{e,t}}{T}. \tag{4.18}$$

The simplest training task ($\bar{w}_e \approx 1/T$) occurs when weights for all steps $t > 1$ are heavily damped ($w_t \to 0$), whereas the most difficult task ($\bar{w}_e = 1$) occurs when all weights are undamped ($w_t = 1$). In our framework, the first rollout step is always undamped ($w_1 = 1$); therefore, the scheduler transitions from $\bar{w}_e = \frac{1}{T}$ to $\bar{w}_e = 1$.

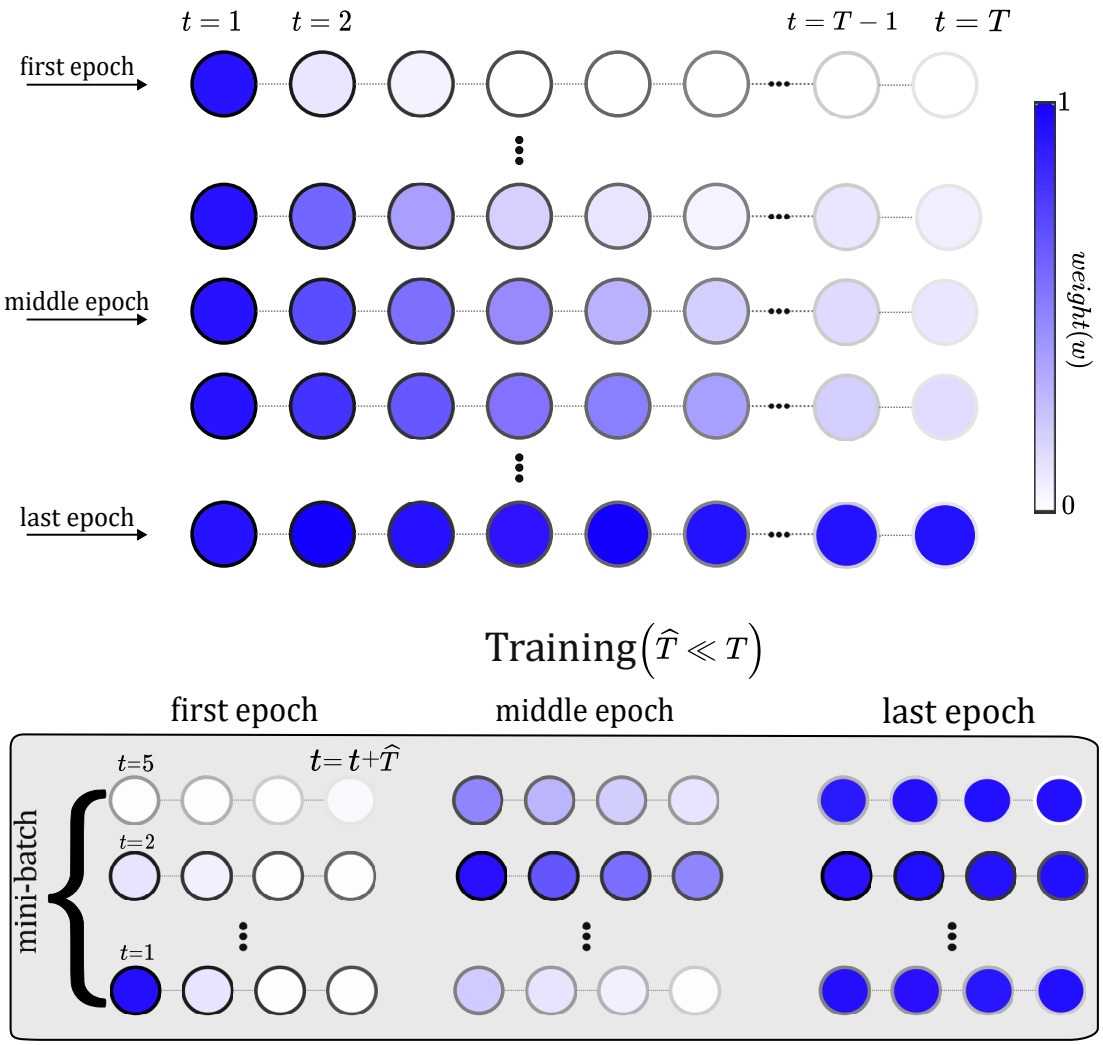


Figure 4.5: STLW: A schematic of the Scheduled Temporal Loss Weighting (STLW) Curriculum during training. The colour strength ranges from 0 (low) to 1 (high), indicating the weight assigned to each rollout during training. Later rollouts in the early training phase have low strength, whereas these weights gradually increase in magnitude. The weights are applied to the loss of its respective timestep and epoch.

4.6.3 Weighting Transition function

We employ a predefined transition function $C(e,t)$ to assign weight values across epochs and timesteps:

$$w_{e,t} = C(e,t), \qquad t \in \{1,\dots,T\}. \tag{4.19}$$

The weights are structured such that $w_{e,t-1} \leq w_{e,t} \leq w_{e,t+1}$ is not necessarily the case initially; rather, we prioritise $w_{e,t} < 1$ for later t during early epochs.

The choice of an exponentially decaying weight profile is motivated directly by the structure of error accumulation in autoregressive rollouts. Let $\epsilon_t = \|\tilde{u}_t - u_t\|$ denote the prediction error at rollout step t. Because each state is produced by feeding the previous prediction back into the operator $\mathcal{F}_\theta$, a first-order perturbation analysis gives $\epsilon_t \lesssim L\,\epsilon_{t-1} + \delta$, where L is the local Lipschitz constant of $\mathcal{F}_\theta$ and δ is the intrinsic per-step error. Unrolling this recursion yields

$$\epsilon_t \;\lesssim\; \delta \sum_{k=0}^{t-1} L^k \;=\; \delta\,\frac{L^t - 1}{L - 1}, \tag{4.20}$$

so that for $L > 1$ the per-step error grows *geometrically* (i.e. exponentially) with the rollout index t. Under a uniform loss weighting, the later timesteps therefore dominate both the loss magnitude and, more importantly, the gradient, precisely when the model is least accurate (early in training) and these large late-horizon gradients are most destabilising. To counteract this, we seek weights w_t that approximately *equalise* the weighted error contribution $w_t\,\epsilon_t$ across the rollout. Substituting the exponential growth $\epsilon_t \sim e^{\beta t}$ implied by (4.20) (with $\beta = \ln L$) into the equalisation condition $w_t\,\epsilon_t \approx \text{const}$ gives $w_t \sim e^{-\beta t}$, i.e. an *exponentially decreasing* weight profile is the natural choice for redistributing the loss uniformly over the horizon. As training proceeds and the intrinsic error δ (and effective L) decrease, the accumulation in (4.20) weakens and the weights are relaxed back towards uniformity ($w_t \to 1$), recovering the standard full-rollout objective. Following an empirical investigation, we selected a Gaussian-style exponential function that realises this decaying profile while fixing the first rollout at unit weight ($w_1 = 1$):

$$C(e,t) = \exp\left(-C_e \cdot t^2\right). \tag{4.21}$$

The decay parameter C_e is defined as:

$$C_e(e) = \exp\left(-A \cdot e \cdot \frac{T}{E}\right), \tag{4.22}$$

where E is the total number of epochs and $e \in \{1,\dots,E\}$. The constant A controls how quickly the weighting profile relaxes towards uniformity and must

be adapted to the sequence length T: longer rollouts accumulate error over more steps (cf. Equation 4.20) and therefore require a slower relaxation. We determined the dependence of A on T through a dedicated empirical study. For each of a set of representative sequence lengths $T \in \{8, 16, 32, 64\}$, we performed a one-dimensional grid search over candidate values of A, training the model to convergence at each value and recording the $A^\star(T)$ that minimised the held-out validation rollout error. Plotting $\ln A^\star$ against T revealed a decaying power-law trend, which we fitted by least squares to obtain the closed-form relation

$$A = \exp\left(5.0 \cdot T^{-0.723}\right). \tag{4.23}$$

The fitted exponent (-0.723) and prefactor (5.0) are specific to the benchmarks studied here; the relation should be regarded as an empirical interpolation that removes the need to re-tune A for each new T within the tested range, rather than as a universal law. In practice, A may alternatively be treated as a single tunable hyperparameter and selected directly by validation; Equation 4.23 simply provides a convenient default that generalised well across the sequence lengths considered.

Figure 4.6 visualises this weighting function over 100 epochs. In the early stages, the function assigns high weights to the initial timesteps and decays rapidly thereafter. As training progresses, the curve flattens until all weights converge to unity ($w_t \approx 1$). This specific Gaussian formulation, centred at $t = 1$, ensures a smooth transition that scales robustly across different PDE benchmarks.

In summary, the transition function C (4.19) determines the weight values applied to all predictions. We chose an exponential function that forms a Gaussian curve, centred at $t = 1$, since the weight of the first rollout is fixed ($w_1 = 1$). The value of C_e (4.22) determines the width of the curve. As epoch e increases, the width of the curve (C_e) increases based on an exponential function that depends on the total timesteps T and the total number of epochs E. Finally, we define an empirical function A to account for training data with different T values, thereby maintaining a smooth transition.

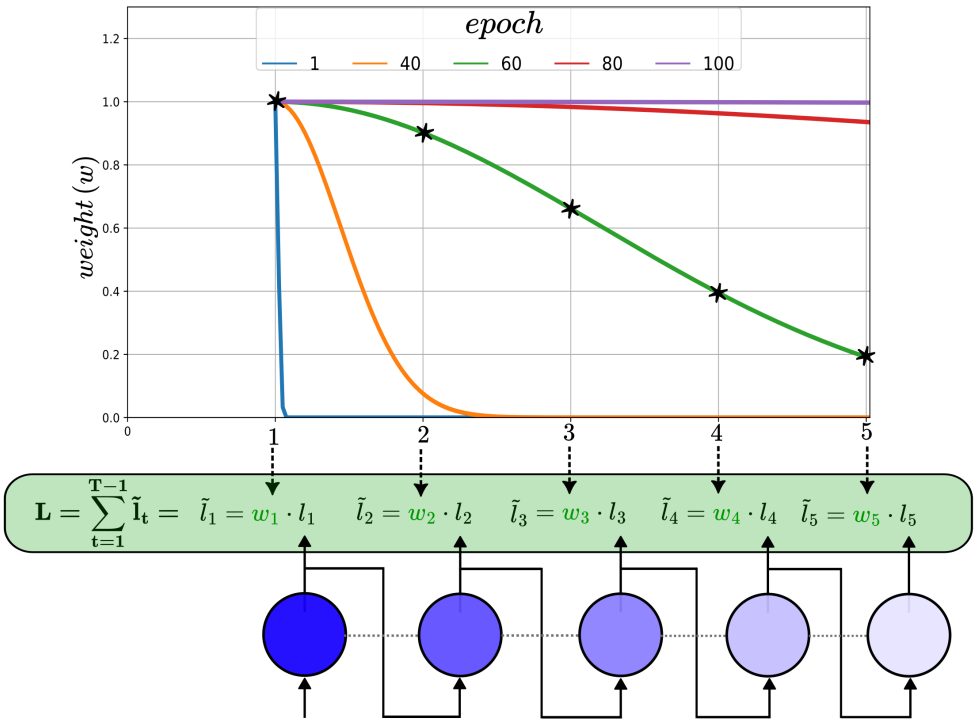


Figure 4.6: A schematic of the scheduled temporal loss Weighting (STLW) Curriculum. For a dataset with a subsequence of length T ($T = 6$), a weight w is assigned to each timestep for $t > 0$, which controls the loss l contributed by each timestep (for example, at epoch = 60, the weight for each timestep is highlighted in black markers). The weighting function (curves), which determines the weight values, changes during training, and at the end of training (epoch = 100), the weighting function sets all weights to 1.

4.6.4 Training Algorithm

The pseudocode for the training process is summarised in Algorithm 4.4. We train using mini-batches of randomly sampled initial timesteps across multiple iterations per epoch. At the start of each epoch the temporal weights are recomputed from Equations 4.21–4.23; within each iteration the backpropagated loss is the sum of the weighted per-timestep losses over the rollout, and the model parameters θ are updated once per iteration.

Algorithm 1: Scheduled Temporal Loss Weighting (STLW) Curriculum

Input: epochs E, sequence length T, training rollout $\widehat{T}$, iterations/epoch N_{iter}, model $\mathcal{F}_\theta$, loss

for $e = 1, \ldots, E$:

compute temporal weights $\{w_{e,t}\}_{t=1}^{T}$ from Eq. 4.21–4.23

for $iter = 1, \ldots, N_{iter}$:

sample start time $t_0 \sim \{0, \ldots, T - \widehat{T}\}$ and set $\tilde{u}_{t_0} = u_{t_0}$

roll out $\widehat{T}$ steps via $\tilde{u}_{t_0+\hat{t}} = \mathcal{F}_\theta(\tilde{u}_{t_0+\hat{t}-1})$ and accumulate

$L_{total} = \sum_{\hat{t}=1}^{\widehat{T}} w_{e,\,t_0+\hat{t}}\, \mathcal{L}\left(\tilde{u}_{t_0+\hat{t}}, u_{t_0+\hat{t}}\right)$

update $\theta \leftarrow \text{Optimiser}(\nabla_\theta L_{total})$

Table 4.4: Pseudocode for the Scheduled Temporal Loss Weighting (STLW) curriculum. At each epoch the temporal weights are refreshed; within each iteration a random subsequence of length $\widehat{T}$ is rolled out autoregressively and the parameters are updated on the summed weighted loss.

4.6.5 Gradient Propagation

Under the STLW framework, we can reformulate the standard optimisation objective from Equation 4.4 as follows:

$$\theta^* = \arg\min_\theta \left[\sum_{t=0}^{T-\widehat{T}} \sum_{\hat{t}=1}^{\widehat{T}} w_{e,t+\hat{t}} \cdot l\left(u_{t+\hat{t}}, \tilde{u}_{t+\hat{t}}\right) \right]. \tag{4.24}$$

For convenience, assuming $\widehat{T} \to T$ such that $\hat{t} = t$, the total weighted loss is:

$$\widehat{L} = \sum_{t=1}^{T} \hat{l}_t = \sum_{t=1}^{T} w_{e,t} \cdot l(u_t, \tilde{u}_t). \tag{4.25}$$

The gradients of this loss, $\frac{\partial \widehat{L}}{\partial \theta}$, are computed by backpropagating through all timesteps. The gradient of the weighted loss at each timestep t can be derived

by modifying the unweighted formulation from Equation 4.7:

$$\begin{aligned}\frac{\partial \hat{l}_t}{\partial \theta} &= \sum_{tt=1}^{t}\left[\frac{\partial \hat{l}_t}{\partial \tilde{u}_t}\left(\prod_{ttt=1}^{t-tt}\frac{\partial \tilde{u}_{t-ttt+1}}{\partial \tilde{u}_{t-ttt}}\right)\frac{\partial \tilde{u}_{tt}}{\partial \theta}\right] \\ &= w_{e,t}\left(\sum_{tt=1}^{t}\left[\frac{\partial l_t}{\partial \tilde{u}_t}\left(\prod_{ttt=1}^{t-tt}\frac{\partial \tilde{u}_{t-ttt+1}}{\partial \tilde{u}_{t-ttt}}\right)\frac{\partial \tilde{u}_{tt}}{\partial \theta}\right]\right) \\ &= w_{e,t}\frac{\partial l_t}{\partial \theta}.\end{aligned} \tag{4.26}$$

Comparing the weighted loss gradient with the standard formulation in Equation 4.7 reveals that the weights are directly backpropagated through the optimisation process. Effectively, the weighted gradient is the product of the standard loss gradient and the scheduled weight. Consequently, STLW can be interpreted as a scheduled gradient-weighting mechanism. In the early stages of training, high priority (minimal damping) is assigned to earlier timesteps, while lower priority (heavy damping) is assigned to later timesteps, mitigating the impact of exploding gradients while the model is still learning the fundamental dynamics.

4.6.6 STLW as a Learning Rate Scheduler

The learning rate (η) is a critical hyperparameter in gradient descent optimisation, as it controls the step size, prevents overshooting, and ensures convergence. Our proposed method can be interpreted as a spatio-temporal extension of a learning rate scheduler. Standard gradient descent is defined as:

$$\theta_{e+1} = \theta_e - \eta_e \frac{\partial l}{\partial \theta}. \tag{4.27}$$

It is common practice to employ a scheduler (e.g., linear, exponential, or cosine decay) to vary η during training to improve stability and accuracy. Typically, the learning rate decreases as the epoch e progresses. Our method can be reformulated in a similar manner:

$$\begin{aligned}\theta_{e+1} &= \theta_e - \eta_e\left(w_{e,t}\frac{\partial l_t}{\partial \theta}\right), \\ \theta_{e+1} &= \theta_e - \widehat{\eta}_{e,t}\frac{\partial l_t}{\partial \theta}, \quad \text{where} \quad \widehat{\eta}_{e,t} = \eta_e \cdot w_{e,t}.\end{aligned} \tag{4.28}$$

Through this formulation, our approach can be viewed as adding an extra dimension to the standard learning rate. This allows for adaptive step sizes across different timesteps t, thereby leveraging the stabilising benefits of a learning rate scheduler while maintaining temporal granularity.

Despite this mathematical equivalence, we categorise our approach as a form of curriculum learning rather than a standard scheduler. This distinction arises because STLW weights increase as training progresses to introduce more complexity, whereas standard learning rates typically decrease to facilitate fine-tuning near the end of training.

4.6.7 STLW as a Continuous AR Curriculum

Our approach can be interpreted as a continuous generalisation of the discrete AR-Curriculum (see Figure 4.7). Training with a fixed autoregressive rollout, where the sequence length $\widehat{T}$ is large, is inherently prone to instability. While a discrete curriculum improves stability by incrementally increasing the rollout length, it may lead to overfitting to early-stage dynamics. This is particularly problematic for PDEs with evolving temporal behaviours, as the model remains "blind" to later dynamics until the curriculum progresses.

The proposed continuous approach alleviates this by including all timesteps in the loss function from the onset of training. Rather than completely ignoring later timesteps, STWL limits their influence via the weighting function, ensuring the model maintains a global view of the trajectory while prioritising stability through damping. The distinction between these strategies can be formalised as:

$$\text{Fixed AR: } \widehat{L} = \sum_{t=1}^{T} w_{e,t} \cdot l(u_t, \tilde{u}_t), \qquad w_{e,t} = 1, \tag{4.29}$$

$$\text{Discrete AR-CL: } \widehat{L} = \sum_{t=1}^{\widehat{T}_e} l(u_t, \tilde{u}_t), \qquad \widehat{T}_e < T, \tag{4.30}$$

$$\text{Continuous AR-CL (STWL): } \widehat{L} = \sum_{t=1}^{T} w_{e,t} \cdot l(u_t, \tilde{u}_t), \qquad w_{e,t} \in (0, 1]. \tag{4.31}$$

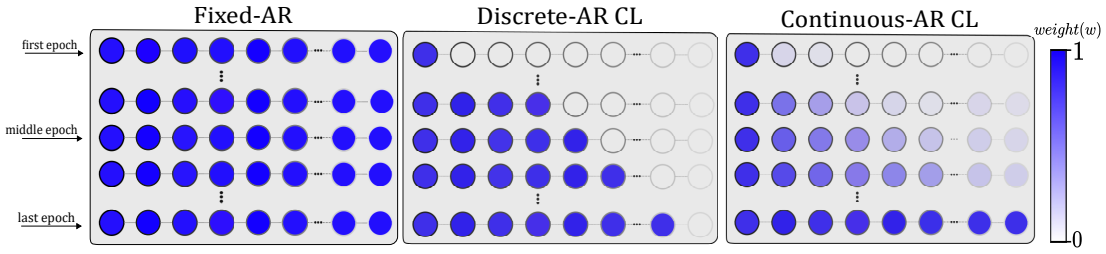


Figure 4.7: Schematic comparison between the discrete AR-CL and the Scheduled Temporal Weight Loss (STWL) curriculum (right). STWL acts as a continuous variant, with a weight applied at every timestep, represented by the colour intensity of the bars.

4.7 Experiments

To validate the effectiveness of our proposed STWL training strategy, we evaluate it against both baseline and established curriculum learning methods. We investigate these strategies across four distinct datasets representing a diverse range of PDE dynamics. Our analysis aims to address the following research questions:

- **Q1 Baseline training strategies:** How do the baseline strategies (TF, TF+N, and AR) compare in terms of stability and long-term accuracy?
- **Q2 Curriculum learning performance:** How does our proposed STLW curriculum compare with existing CL strategies? Do curriculum-based approaches consistently outperform the strongest baseline methods?
- **Q3 Impact of training rollout:** What is the effect of the training rollout length, $\widehat{T}$, on model performance and the associated computational cost?
- **Q4 Generalisation to inference rollouts:** What is the impact of extending the temporal horizon during inference for models trained with different strategies?
- **Q5 Curriculum robustness:** How sensitive are curriculum learning strategies to different random seed initialisations, and what is their overall stability across multiple runs?
- **Q6 Sensitivity to curriculum parameterisation:** To what extent does the choice of transition function and its parameters influence the final model performance?

4.7.1 Training Strategies

We evaluate our proposed **Scheduled Temporal Loss Weighting (AR-STLW-CL)**, which dynamically weights the loss of individual timesteps throughout the curriculum, against several baseline and established curriculum learning (CL) strategies. These are categorised as follows:

1. **Baseline Training Strategies (Static)**
   - **Teacher Forcing (TF):** The model exclusively receives ground-truth data as input for each prediction step, effectively training on one-step rollouts.
   - **Teacher Forcing with Noise (TF+N):** Similar to TF, but with Gaussian noise injected into the ground-truth inputs to enhance robustness against error accumulation during inference.
   - **Fixed Autoregressive (AR):** The model consistently uses its own previous predictions as input, directly simulating its behaviour during the inference phase.
2. **Deterministic Curricula (D-CL)** These strategies follow a fixed, non-random schedule to transition the input source or task complexity:
   - **TF–AR-D-CL:** Initiates training with full Teacher Forcing and transitions to full Autoregressive rollouts by the final epoch.
   - **AR–TF-D-CL:** The inverse approach, beginning with full AR rollouts and transitioning toward TF.
   - **AR-CL:** A curriculum that progressively increases the length of the autoregressive rollout ($\widehat{T}$) without employing Teacher Forcing.
3. **Probabilistic Curricula (P-CL)** These strategies use a probability α_e to determine the training mode for each rollout:
   - **TF–AR-P-CL:** Rollouts are initially sampled as TF, with the probability of sampling an AR rollout increasing throughout the training duration.
   - **AR–TF-P-CL:** Rollouts start as predominantly AR, with the sampling distribution shifting toward TF as training progresses.

4.7.2 Parameterisation of CL Strategies

Table 4.5 summarises the parameters and values utilised across our experiments. The baseline strategies (TF, TF+N, and AR) do not employ a curriculum; hence,

the AR ratio, training rollout length, and loss weighting remain static. For the AR-CL approach, we investigated increasing the training rollout length, $\widehat{T}$, during training using both linear and exponential schedules.

For the deterministic and probabilistic approaches involving TF–AR (increasing α), we explored various starting ratios $\alpha_{start} \in \{0.0, 0.25, 0.5, 0.75\}$, all transitioning toward $\alpha_{end} = 1.0$. Conversely, for AR–TF (decreasing α), we explored $\alpha_{start} \in \{1.0, 0.75, 0.5, 0.25\}$ transitioning toward $\alpha_{end} = 0.0$.

For our proposed AR-STLW-CL, we varied the A parameter from Equation 4.22, defined as:

$$A = \exp\left(A_c \cdot T^{-0.723}\right). \tag{4.32}$$

We evaluated values of A_c within the range $[4.0, 6.0]$. Initial exploratory investigations indicated that the exponential transition function provided superior performance and stability compared to linear or sigmoid alternatives; consequently, the results presented hereafter exclusively use the exponential transition profile.

	Strategy	**Parameter and Transition**		
		AR ratio $\alpha_{start} \rightarrow \alpha_{end}$	Training rollout $\widehat{T}$	Weighting w
Baseline	TF	0.0	Constant	Constant, 1.0
	TF+N	0.0	Constant	Constant, 1.0
	AR	1.0	Constant	Constant, 1.0
Baseline CL	AR-CL	1.0	Variable, Increasing {Linear, Exponential}	Constant, 1.0
	TF–AR-D-CL	$\{0.0, 0.25, 0.5, 0.75\} \nearrow 1.0$ {Linear, Exponential}	Constant	Constant, 1.0
	TF+N–AR-D-CL	$\{0.0, 0.25, 0.5, 0.75\} \nearrow 1.0$ {Linear, Exponential}	Constant	Constant, 1.0
	AR–TF-P-CL	$\{1.0, 0.75, 0.5, 0.25\} \searrow 0.0$ {Linear, Exponential}	Constant	Constant, 1.0
	TF–AR-P-CL	$\{0.0, 0.25, 0.5, 0.75\} \nearrow 1.0$ {Linear, Exponential}	Constant	Constant, 1.0
Ours	AR-STLW-CL	1.0, $4.0 \leq A_c \leq 6.0$	Constant	Variable, Increasing $w \in [0, 1]$, Exponential

Table 4.5: Curriculum learning parameters used in exploratory and final experiments.

4.7.3 Datasets

To evaluate the proposed STLW approach, we conducted experiments on four one-dimensional (1D) time-dependent PDEs. These datasets were selected to represent a diverse range of physical behaviours, from linear transport to nonlinear dispersive waves. Specifically, we utilise Advection (Av) and viscous Burgers (vB) from the PDEBench suite (Takamoto *et al.*, 2022), inviscid Burgers with a forcing term (iB) from (Brandstetter *et al.*, 2022c), and the Korteweg–De Vries (KdV) equation from (Brandstetter *et al.*, 2022b).

Advection (Av): A first-order hyperbolic PDE that models the linear transport of a scalar quantity by the bulk motion of a fluid. This dataset serves as a baseline for assessing the ability of curriculum strategies to maintain stability over 200 timesteps with relatively simple dynamics.

Burgers' Equations (vB and iB): The dynamics of the Burgers' equation are governed by the balance between nonlinear advection and dissipation. We investigate two distinct regimes:

- **Viscous Burgers (vB):** Transitions from sharp, high-amplitude features to smooth, low-amplitude solutions over 200 timesteps.
- **Inviscid Burgers with forcing (iB):** Exhibits the opposite behaviour, transitioning from smooth initial states to sharp, high-amplitude solutions over 250 timesteps.

These cases test the efficacy of the STLW strategy in capturing evolving dynamics where the solution complexity changes significantly over the temporal horizon.

Korteweg–De Vries (KdV): This equation models nonlinear, dispersive, and non-dissipative waves on shallow-water surfaces. The KdV equation is significantly more challenging than the Advection case due to its dispersive nature. We utilise this dataset to evaluate the robustness of our curriculum strategies over a substantially longer sequence of 640 timesteps.

4.7.4 Training Procedure

All models were implemented using the Fourier Neural Operator (FNO) with 16 Fourier modes, 32 hidden dimensions, and 4 layers (see Section 3.2 for architectural details). This configuration was chosen following an exploratory grid search to optimise the trade-off between predictive accuracy and computational efficiency.

Unless otherwise specified, we employ the Adam optimiser with an initial learning rate of 10^{-3}, which is decayed by a factor of 0.95 after each epoch. Training is performed for 100 epochs with an nRMSE loss and a batch size of 64.

To manage memory constraints while capturing temporal dependencies, we use a fixed training rollout of $\widehat{T} = 24$. We employ a multistep prediction of 5, i.e., the input (history) and output (horizon) are equal. For each epoch, we randomly sample the starting temporal index for each trajectory within a batch, repeating this process 100 times per epoch (iterations). To ensure statistical reliability, each experiment is executed five times with different random weight initialisations; we report the resulting mean and standard deviation.

4.7.5 Analysis of Training Dynamics

We evaluate the training and validation loss curves to distinguish between short-term and long-term predictive performance. Figure 4.8 illustrates the training dynamics of the baseline strategies (TF, TF+N, and AR) for two datasets: vB and KdV. We report both the one-step (short-term) and full-rollout (long-term) validation losses, with the shaded regions indicating the standard deviation across five runs.

We observe that TF and TF+N achieve the lowest one-step rollout loss. The narrower shaded region for TF+N suggests that adding Gaussian noise enhances the robustness of the teacher-forcing strategy. However, these strategies exhibit significant instability when evaluated using the full rollout loss. Conversely, the Fixed AR strategy remains more stable and accurate over long horizons, despite having a higher one-step error.

The integration of curriculum learning (CL) further enhances these results. As shown in Figure 4.9, both the discrete AR-CL and our proposed AR-STLW-CL outperform the baseline AR strategy across both loss metrics. Interestingly, while AR-CL achieves a slightly lower one-step error, AR-STLW-CL demonstrates superior performance on the full-rollout loss. This result supports our hypothesis that the continuous weighting mechanism in STLW is better suited for mitigating error accumulation during long-term autoregressive inference.

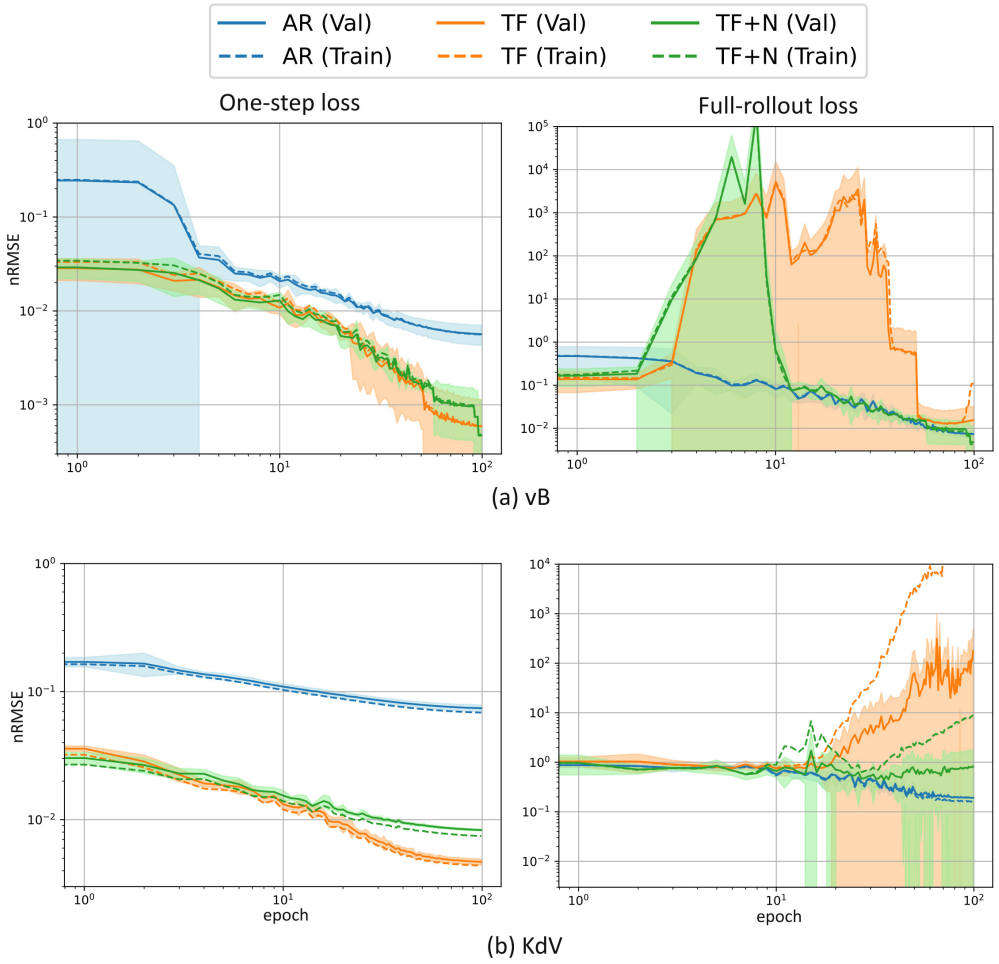


Figure 4.8: Training dynamics of baseline strategies. The training (dashed line) and validation (solid line) losses are shown for each strategy. Shaded areas represent the standard deviation.

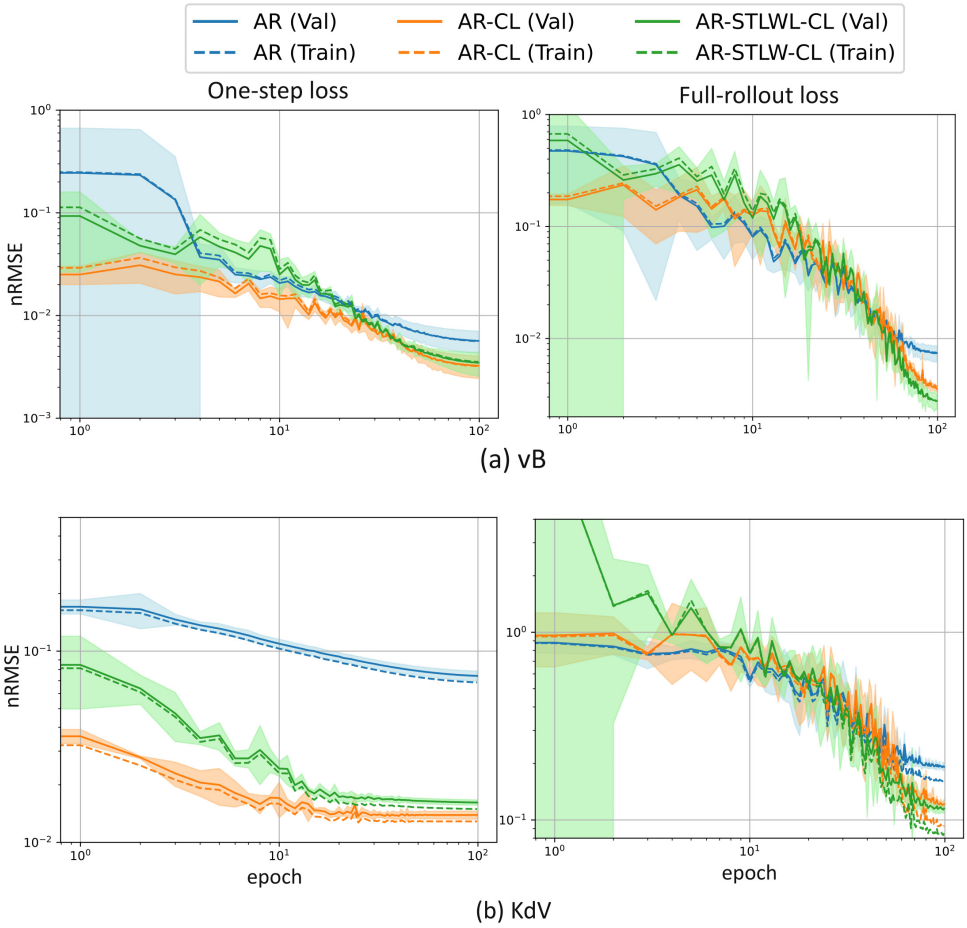


Figure 4.9: Training dynamics of curriculum strategies compared to the AR baseline. STLW demonstrates enhanced stability and lower long-term error propagation.

4.8 Results and Discussion

Table 4.6 presents the mean normalised Root Mean Squared Error (nRMSE) across four PDE datasets, evaluated over nine distinct training strategies. Each row reflects the curriculum parameter configuration that yielded the lowest nRMSE during our exploratory investigation (as detailed in Table 4.5). For each dataset, the best-performing strategy is highlighted in **bold**, and the second-best is underlined.

4.8.1 Baseline Performance Analysis

We first evaluate the static baseline strategies: TF, TF+N, and AR. Our results indicate that the AR strategy significantly outperforms TF in three out of the four datasets (Av, vB, and KdV). In contrast, TF outperforms AR on the iB dataset. To provide a rigorous comparison, we use the best-performing baseline for each dataset as the reference point to calculate the relative improvement (*rel. impro.*) of the curriculum strategies.

4.8.2 Comparative Effectiveness of Curriculum Learning

The empirical results demonstrate that curriculum learning strategies consistently enhance model performance. Our proposed **AR-STLW-CL** method achieved the lowest error (and consequently the greatest relative improvement) on three of the four datasets: vB, iB, and KdV, with improvements of **62.5%**, **12.0%**, and **40.3%**, respectively. The AR-CL strategy ranked second in three cases (Av, iB, and KdV), with relative improvements of 16.8%, 11.4%, and 36.9%.

Among the deterministic strategies, TF–AR-D-CL and TF+N–AR-D-CL showed strong performance on the viscous Burgers' (vB) dataset, ranking as the joint best and second-best strategies. Conversely, probabilistic approaches (AR–TF-P-CL and TF–AR-P-CL) generally underperformed. Notably, AR–TF-P-CL (decreasing complexity) often led to performance degradation, failing to surpass the best baseline strategies in several instances. This confirms that increasing the task difficulty over time, rather than decreasing it, is critical for training stability in autoregressive neural operators.

	Strategy	Best Curriculum	nRMSE		
		$\alpha_{start} \rightarrow \alpha_{end}$	$mean^{\pm s.d}$ ↓	rel. impro. ↑	last 10% ↓
Advection (Av)	TF	0.0	$0.0375^{\pm 1.76e-2}$	–	0.0883
	TF+N	0.0	$0.0165^{\pm 5.68e-3}$	–	0.0446
	AR	1.0	$0.00353^{\pm 8.70e-5}$	–	0.00536
	AR-CL	1.0	$\underline{0.00293}^{\pm \underline{8.98e-5}}$	16.8%	<u>0.00450</u>
	TF–AR-D-CL	0.0 ↗ 1.0	$\mathbf{0.00272}^{\pm 9.58e-5}$	**22.6%**	**0.00423**
	TF+N–AR-D-CL	0.0 ↗ 1.0	$0.00296^{\pm 1.43e-4}$	15.9%	0.00455
	AR–TF-P-CL	1.0 ↘ 0.0	$0.00538^{\pm 1.18e-3}$	−52.4%	0.01460
	TF–AR-P-CL	0.0 ↗ 1.0	$0.00294^{\pm 9.48e-5}$	16.5%	0.00473
	AR-STLW-CL	1.0, $A_c = 5.0$	$0.00303^{\pm \mathbf{0.0}}$	14.1%	0.00485
Viscous Burgers (vB)	TF	0.0	$0.0632^{\pm 4.23e-2}$	–	0.1040
	TF+N	0.0	$0.0125^{\pm 2.39e-2}$	–	0.0154
	AR	1.0	$0.00749^{\pm 1.15e-3}$	–	0.00771
	AR-CL	1.0	$0.00376^{\pm \mathbf{1.30e-4}}$	49.7%	0.00623
	TF–AR-D-CL	0 ↗ 1	$\mathbf{0.00280}^{\pm 4.83e-4}$	**62.5%**	**0.00363**
	TF+N–AR-D-CL	0.0 ↗ 1.0	$\underline{0.00291}^{\pm 3.59e-4}$	<u>61.1%</u>	<u>0.00422</u>
	AR–TF-P-CL	1.0 ↘ 0.0	$0.0112^{\pm 8.29e-4}$	−44.9%	0.01427
	TF–AR-P-CL	0.0 ↗ 1.0	$0.00413^{\pm \underline{2.13e-4}}$	56.3%	0.00501
	AR-STLW-CL	1.0, $A_c = 5.0$	$\mathbf{0.00280}^{\pm 4.66e-4}$	**62.5%**	0.00566
Inviscid Burgers (iB)	TF	0.0	$0.274^{\pm 3.9e-3}$	–	0.625
	TF+N	0.0	$0.278^{\pm 8.1e-3}$	–	0.628
	AR	1.0	$0.306^{\pm 8.7e-3}$	–	0.683
	AR-CL	1.0	$\underline{0.243}^{\pm 3.4e-3}$	<u>11.4%</u>	<u>0.559</u>
	TF–AR-D-CL	0.0 ↗ 1.0	$0.252^{\pm 4.4e-3}$	8.2%	0.571
	TF+N–AR-D-CL	0.0 ↗ 1.0	$0.250^{\pm 3.1e-3}$	8.7%	0.573
	AR–TF-P-CL	1.0 ↘ 0.0	$0.291^{\pm 1.4e-2}$	−6.1%	0.660
	TF–AR-P-CL	0.0 ↗ 1.0	$0.266^{\pm \mathbf{1.7e-3}}$	3.0%	0.603
	AR-STLW-CL	1.0, $A_c = 5.0$	$\mathbf{0.241}^{\pm \underline{2.8e-3}}$	**12.0%**	**0.558**
KdV	TF	0.0	$0.541^{\pm 9.50e-2}$	–	0.896
	TF+N	0.0	$0.347^{\pm 1.01e-1}$	–	0.623
	AR	1.0	$0.189^{\pm 5.29e-3}$	–	0.311
	AR-CL	1.0	$\underline{0.119}^{\pm 5.24e-3}$	<u>36.9%</u>	<u>0.226</u>
	TF–AR-D-CL	0.0 ↗ 1.0	$0.159^{\pm \mathbf{4.13e-3}}$	15.9%	0.290
	TF+N–AR-D-CL	0.0 ↗ 1.0	$0.145^{\pm \underline{4.74e-3}}$	23.3%	0.267
	AR–TF-P-CL	1.0 ↘ 0.0	$0.226^{\pm 2.0e-2}$	−19.1%	0.411
	TF–AR-P-CL	0.0 ↗ 1.0	$0.189^{\pm 7.29e-3}$	0%	0.342
	AR-STLW-CL	1.0, $A_c = 5.0$	$\mathbf{0.113}^{\pm 5.02e-3}$	**40.3%**	**0.214**

Table 4.6: **Accuracy**: Normalised Root Mean Squared Error (nRMSE) using FNO. **Bold** and <u>underline</u> denote the best and second-best results, respectively.

4.8.3 Long-Term Accuracy and Stability

To assess long-term predictive performance, we computed the nRMSE over the final 10% of the temporal horizon (the "Last 10%" column in Table 4.6). Excluding the probabilistic variants, curriculum learning consistently enhanced accuracy at later timesteps. **AR-STLW-CL** demonstrated the lowest terminal error for two datasets (iB and KdV), while TF–AR-D-CL achieved the lowest error for Av and vB.

We also evaluated the stability of these strategies by examining the standard deviation across five independent runs per configuration. Our findings reveal that non-probabilistic curriculum strategies exhibit smaller standard deviations than static baselines, indicating greater robustness to weight initialisation. Specifically, AR-STLW-CL demonstrated the highest stability for the Av dataset, while AR-CL, TF–AR-P-CL, and TF–AR-D-CL showed the smallest variations for vB, iB, and KdV, respectively.

4.8.4 Error Propagation Analysis

Figure 4.10 illustrates the accumulation of error over time for the three top-performing curricula for each dataset. These trajectories clearly demonstrate the benefit of curriculum-based training across all physical regimes. While **AR-STLW-CL** generally achieves the lowest overall nRMSE, the plots indicate that its performance advantage over the top discrete curricula remains consistent throughout the rollout, effectively "flattening" the error growth curve compared to the best baseline.

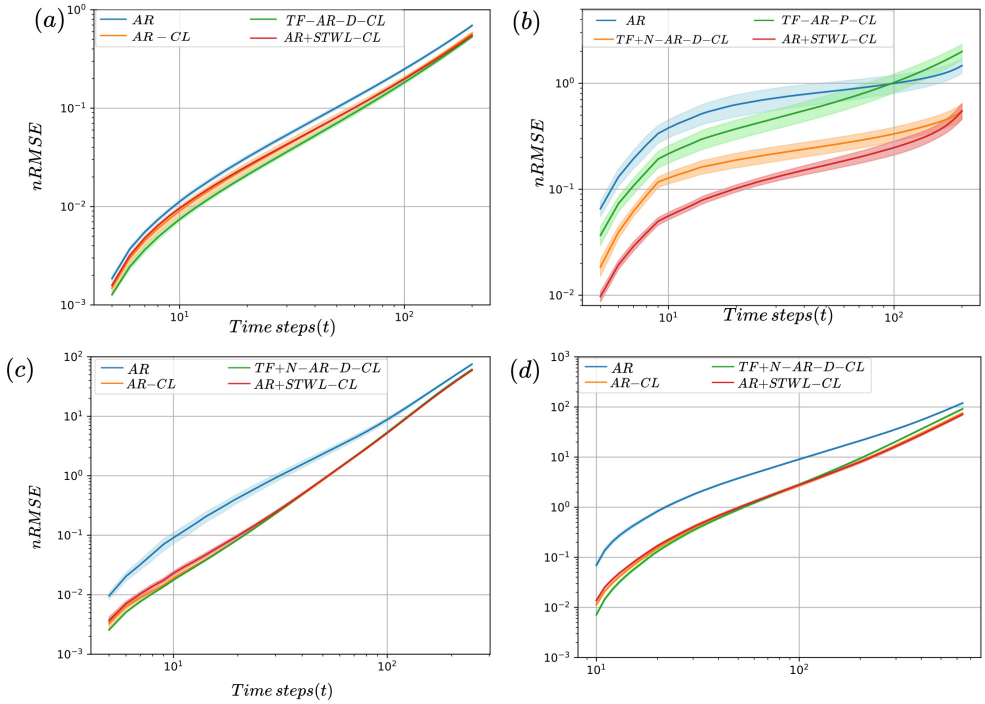


Figure 4.10: Error propagation over time for the best baseline and the top two curriculum strategies for: (a) Advection (Av), (b) Viscous Burgers (vB), (c) Inviscid Burgers (iB), and (d) Korteweg–De Vries (KdV).

4.8.5 Impact of Training Rollout Length

As established in Section 4.6.7, AR-STLW-CL can be conceptualised as a continuous generalisation of the discrete AR-CL strategy. While discrete AR-CL progressively increases the number of training rollouts $\widehat{T}$, our continuous approach applies temporal weights across the entire sequence length. We investigate the impact of varying the training rollout length $\widehat{T} \in \{2, 4, 8, 16, 32\}$ on the performance of the fixed-AR, discrete AR-CL, and continuous AR-STLW-CL strategies.

Figure 4.11 presents the nRMSE as a function of $\widehat{T}$. Our observations indicate that **AR-STLW-CL** consistently achieves lower errors and greater stability across all rollout lengths tested. Notably:

- **Advection (Av):** AR-STLW-CL remains stable even at low $\widehat{T}$ values where other methods fluctuate.
- **Viscous Burgers (vB):** While errors generally reach a minimum at $\widehat{T} = 8$,

AR-STLW-CL consistently maintains a lower error profile across the entire range.

- **iB and KdV:** Whereas the error for the fixed-AR strategy begins to diverge as $\widehat{T}$ increases beyond 8, both AR-CL and AR-STLW-CL maintain a downward trend, with AR-STLW-CL achieving the most significant reduction in error.

In summary, AR-STLW-CL demonstrates enhanced scalability and stability across different training rollouts, highlighting its effectiveness in mitigating gradient-related instabilities associated with long-horizon training.

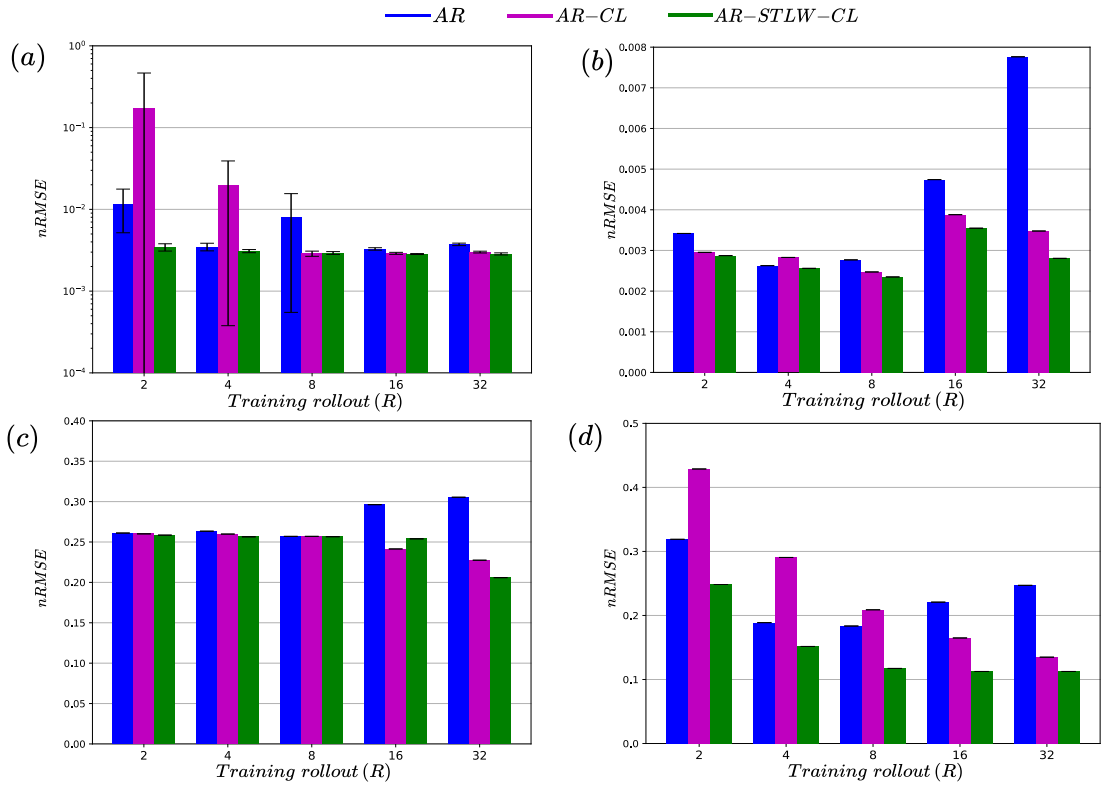


Figure 4.11: Effect of increasing training rollout length $\widehat{T}$ on the nRMSE of the best baseline and the top two curriculum strategies across the four PDE datasets.

4.8.6 Inference Rollout Analysis

To evaluate the robustness of **AR-STLW-CL** for long-term forecasting, we analyse its performance across varying inference horizons (T). Table 4.7 details the mean nRMSE and standard deviation for the Advection (Av) and Korteweg–De Vries (KdV) datasets.

In this analysis, each reported value corresponds to a model trained with a specific output size (S), defined as the number of timesteps predicted per forward pass. For the Av dataset, we evaluate the configurations $(T, S) = (40, 5)$ and $(200, 1)$. For the KdV dataset, we examine increasingly long horizons: $(T, S) \in \{(128, 5), (320, 2), (640, 1)\}$.

The results indicate that AR-STLW-CL consistently outperforms alternative strategies across nearly all temporal horizons. Notably, the performance gap between our curriculum and the baselines becomes more pronounced as the rollout length T increases. While the deterministic curriculum (TF–AR-D-CL) remains competitive at shorter horizons (e.g., $T = 40$ for Av), it exhibits significant instability or divergence in long-term regimes ($T = 200$).

Strikingly, at the longest horizon for KdV ($T = 640$), most baseline and discrete-curriculum strategies either fail to converge or produce errors several orders of magnitude higher than those of our approach. In contrast, AR-STLW-CL maintains superior stability and a significantly lower error profile. This confirms that our continuous weighting strategy effectively mitigates the compounding error and gradient instabilities typically associated with extended autoregressive rollouts.

	Strategy	Inference Rollout T		
		40		**200**
Advection (Av)	TF	$0.0375^{\pm 1.76e-2}$		$0.486^{\pm 4.53e-2}$
	TF+N	$0.0165^{\pm 5.68e-3}$		NaN
	AR	$0.00353^{\pm 8.70e-5}$		$859.6^{\pm 1913.4}$
	AR-CL	$\underline{0.00293}^{\pm 8.98e-5}$		$\underline{0.0551}^{\pm 1.01e-1}$
	TF–AR-D-CL	$\mathbf{0.00272}^{\pm 9.58e-5}$		$10.9^{\pm 23.1}$
	TF+N–AR-D-CL	$0.00296^{\pm 1.43e-4}$		$1.057^{\pm 1.93}$
	AR–TF-P-CL	$0.00538^{\pm 1.18e-3}$		$0.196^{\pm 1.12e-1}$
	TF–AR-P-CL	$0.00294^{\pm 9.48e-5}$		$29.9^{\pm 65.7}$
	AR-STLW-CL	$0.00303^{\pm 0.0}$		$\mathbf{0.0135}^{\pm 8.72e-3}$

	Strategy	Inference Rollout T		
		128	**320**	**640**
KdV	TF	$0.541^{\pm 9.50e-2}$	–	–
	TF+N	$0.347^{\pm 1.01e-1}$	$0.408^{\pm 1.14e-2}$	$0.810^{\pm 1.37e-1}$
	AR	$0.189^{\pm 5.29e-3}$	$0.298^{\pm 6.62e-2}$	$0.518^{\pm 1.94e-1}$
	AR-CL	$\underline{0.119}^{\pm 5.29e-3}$	$\underline{0.196}^{\pm 6.61e-3}$	$2.49e5^{\pm 5.58e5}$
	TF–AR-D-CL	$0.159^{\pm 4.13e-3}$	$0.244^{\pm 1.42e-2}$	$\underline{0.366}^{\pm 1.29e-2}$
	TF+N–AR-D-CL	$0.145^{\pm 4.74e-3}$	–	–
	AR–TF-P-CL	$0.226^{\pm 2.00e-2}$	–	–
	TF–AR-P-CL	$0.189^{\pm 7.29e-3}$	–	–
	AR-STLW-CL	$\mathbf{0.113}^{\pm 5.02e-3}$	$\mathbf{0.170}^{\pm 1.09e-2}$	$\mathbf{0.307}^{\pm \mathbf{2.85e-2}}$

Table 4.7: **Inference Rollout Performance**: nRMSE across varying temporal horizons T. **Bold** and underline denote the best and second-best results. "NaN" and extremely high values indicate model divergence.

4.8.7 STLW as a Multi-Dimensional Learning Rate Scheduler

Our proposed curriculum, **AR-STLW-CL**, can be conceptualised as a **two-dimensional learning rate scheduler**. While conventional schedulers modulate the learning rate η_e solely as a function of the training epoch (e), AR-STLW-CL introduces a temporal weighting function $w_{e,t}$ that varies across both the epoch and the specific prediction timestep $t \in \{1, \ldots, T\}$. This allows the model to selectively prioritise different stages of the temporal rollout at different

points in the training process (see Section 4.6.6 for the formal derivation).

To isolate the impact of temporal weighting from standard learning rate effects, we evaluate a variant of our method with a constant learning rate (AR-STLW-CL + fixed η) and compare it directly with a standard baseline (AR + fixed η). Additionally, we evaluate discrete AR-CL with a fixed η and a standard AR approach with a variable learning rate (AR + var. η), where the latter utilises a decay factor of 0.98 per epoch. While our complete framework adaptively schedules both the temporal weights w and the learning rate η, these comparisons highlight the unique contribution of the STLW mechanism.

Comparative Analysis of Results

To evaluate the efficacy of this interpretation, Figure 4.12 illustrates the test loss curves for the viscous Burgers' (vB) equation under two learning rate regimes: $\eta = 10^{-4}$ (left) and $\eta = 10^{-5}$ (right). We draw the following conclusions:

- **Impact of Temporal Weighting:** Comparing the fixed learning rate variants (AR-STLW-CL + fixed η vs. AR + fixed η) reveals that the inclusion of our curriculum leads to a significant reduction in terminal test loss, demonstrating that scheduled weighting independently improves convergence.

- **Implicit Proxy for Learning Rate Decay:** The behaviour of the fixed-η STLW curriculum closely mirrors that of the standard variable learning rate approach (AR + var. η). While the variable η approach eventually achieves a slightly lower loss, the qualitative similarity between the curves suggests that temporal loss weighting acts as an implicit proxy for learning rate decay, providing a similar regularising effect.

- **Stability vs. Discrete Transitions:** This relationship is further highlighted by the distinct behaviour of AR-CL + fixed η. Unlike the smooth convergence seen with STLW or variable η, the discrete curriculum produces a markedly different, more erratic loss trajectory. This suggests that continuous weighting (STLW) is a more fluid and stable optimisation strategy than discrete transitions in rollout length.

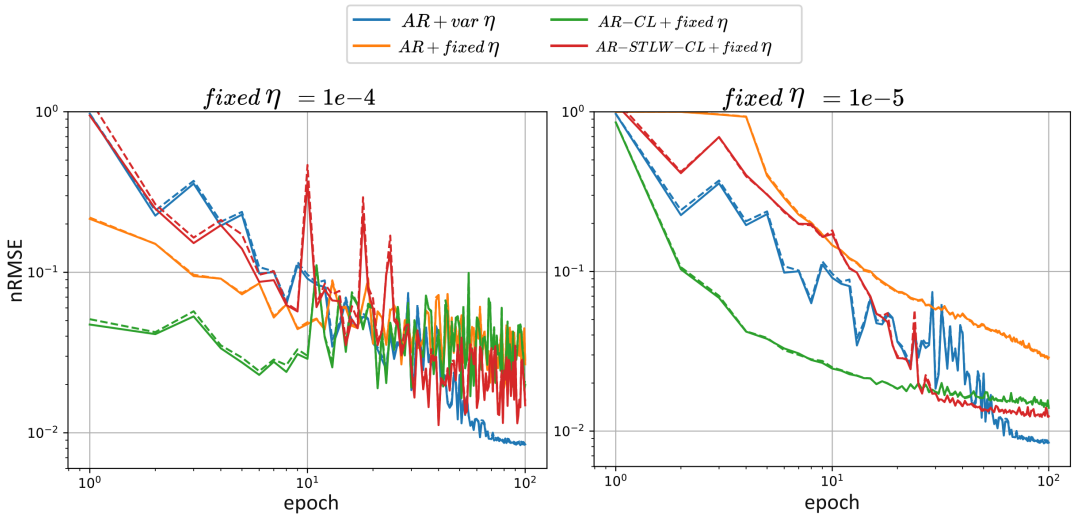


Figure 4.12: Training dynamics under fixed and variable (scheduled) learning rates (η) across different curricula using the Adam optimiser. Dashed lines indicate training loss; solid lines indicate validation loss.

4.8.8 Hyperparameter Analysis: Batch Size and Rollout Dynamics

The training **batch size** (B), **rollout length** (R), and **iterations per epoch** (I) are critical hyperparameters that govern the optimisation landscape (see Algorithm 4.4). The batch size dictates the degree of parallelisation and gradient smoothing, while the rollout length determines the temporal depth of the back-propagation chain. Finally, the number of iterations per epoch determines the diversity of sampled initial conditions. Collectively, these parameters define the total volume of spatio-temporal data processed per gradient update.

To isolate the specific effects of these hyperparameters, we conducted a controlled study by fixing the product of the rollout length and iterations ($R \times I = 256$). This constraint ensures a fair comparison by maintaining a constant number of temporal state-updates per epoch across all configurations. The results are summarised in Table 4.8, which presents the test nRMSE for the **AR** (top), **AR-CL** (middle), and **AR-STLW-CL** (bottom) strategies.

Our analysis yields the following insights:

- **Baseline Regimes:** The standard AR approach is most effective in "low-

resource" regimes, characterised by smaller batch sizes and shorter rollouts, where the gradient variance is high but the error accumulation is limited.

- **Curriculum Scalability: AR-STLW-CL** demonstrates superior scalability in "high-resource" regimes (large batch sizes). Unlike the baselines, it remains robust as rollout length increases, effectively balancing gradient stability and temporal depth.

- **Optimisation Equilibrium:** The results indicate a delicate equilibrium between these parameters; blindly increasing the batch size or rollout does not guarantee convergence. For the viscous Burgers' (vB) dataset, we identified an optimal performance pocket at a batch size of 16 and a rollout of 4.

Configuration		(Rollout R, Iteration I)					
		(1, 256)	**(2, 128)**	**(4, 64)**	**(8, 32)**	**(16, 16)**	**(32, 9)**
Batch	**Strategy**	**Test nRMSE**					
4	AR	**0.0020**	**0.0019**	0.0023	0.0026	0.0042	0.0045
	AR-CL	**0.0020**	0.0027	**0.0017**	**0.0019**	**0.0023**	0.0036
	AR-STLW-CL	0.0080	0.0053	0.0033	0.0027	0.0033	**0.0035**
8	AR	**0.0026**	0.0022	0.0023	**0.0019**	0.0034	0.0072
	AR-CL	**0.0026**	**0.0021**	0.0019	0.0022	**0.0026**	**0.0039**
	AR-STLW-CL	0.0046	0.0026	**0.0017**	0.0020	0.0035	0.0042
16	AR	**0.0024**	**0.0013**	**0.0013**	0.0024	0.0045	0.0106
	AR-CL	**0.0024**	0.0015	**0.0014**	0.0024	**0.0032**	**0.0056**
	AR-STLW-CL	0.0030	0.0025	0.0017	**0.0022**	0.0035	0.0066
32	AR	0.0025	0.0016	**0.0015**	0.0029	0.0068	0.0141
	AR-CL	0.0025	**0.0014**	0.0016	0.0029	0.0052	**0.0082**
	AR-STLW-CL	**0.0018**	0.0016	**0.0015**	**0.0025**	**0.0049**	0.0084
64	AR	0.0028	0.0016	0.0022	0.0034	0.0092	0.0235
	AR-CL	0.0028	0.0015	**0.0019**	**0.0032**	**0.0063**	0.0110
	AR-STLW-CL	**0.0011**	**0.0013**	0.0020	0.0037	0.0072	**0.0091**
128	AR	0.0056	0.0022	0.0036	0.0066	0.0160	0.0373
	AR-CL	0.0056	0.0027	0.0032	0.0050	0.0099	0.0170
	AR-STLW-CL	**0.0016**	**0.0018**	**0.0032**	**0.0059**	**0.0091**	**0.0123**

Table 4.8: **Hyperparameter Sensitivity**: nRMSE as a function of Batch Size and Rollout/Iteration configuration. Each sub-row represents AR, AR-CL, and AR-STLW-CL, respectively. **Bold** indicates the best strategy per cell; blue indicates the best configuration per curriculum.

4.8.9 Generalisability Across Architectures

To investigate the architectural robustness of our approach, we compared the training strategies across three distinct neural operator frameworks: **FNO**, **U-Net**, and **U-NO**. These models span a significant parameter range—from 140K to 1.5M parameters, allowing us to verify that the benefits of the STLW curriculum are not model-specific but generalise across different scales of neural operator complexity. To situate these convolutional and spectral operators against the recent family of attention-based solvers, we include a transformer-based neural operator, the Operator Transformer (OFormer) (Li *et al.*, 2022c) in Tables 4.9

and 4.10.

Analysis of Viscous Burgers' (vB) Results

Table 4.9 summarises the results for the viscous Burgers' equation. Our observations indicate that **AR-STLW-CL consistently achieves the lowest test loss** across all model architectures and rollout lengths. In the unit rollout regime ($R = 1$), the performance gain from our curriculum is particularly significant, substantially outperforming both the static baseline and the discrete AR-CL strategy.

As the rollout length increases to $R = 32$, the task becomes considerably more challenging due to accelerated error accumulation. Even in this high-complexity regime, AR-STLW-CL maintains its superiority, either matching or exceeding the performance of alternative strategies. This demonstrates that our weighted-loss approach provides a stable optimisation path that is independent of the underlying model's capacity or parameter count.

Analysis of Korteweg–De Vries (KdV) Results

The results for the KdV equation, presented in Table 4.10, further underscore the advantages of our strategy in dispersive physical systems. While the standard AR baseline and discrete AR-CL perform competitively at the smallest rollout ($R = 1$), the benefits of **AR-STLW-CL become highly pronounced as the training rollout length increases to** $R = 32$.

In this long-rollout regime, AR-STLW-CL yields the best results for all investigated models (FNO, U-Net, U-NO and OFormer). This suggests a critical takeaway: while standard teacher-forcing or fixed-autoregressive approaches may suffice for short-horizon tasks, scheduled temporal loss weighting is essential for achieving optimal stability and accuracy in long-term autoregressive training across diverse architectures.

(B, R, I)	Training Strategy	Model (Parameter Count)			
		FNO (140K)	**U-Net (680K)**	**U-NO (1.5M)**	**OFormer (667K)**
(128,1,256)	AR	0.0056	0.0023	0.0037	0.0012
	AR-CL	0.0056	0.0023	0.0037	0.0012
	AR-STLW-CL	**0.0016**	**0.0017**	**0.0015**	**0.00089**
(128,32,8)	AR	0.0373	0.0308	0.0119	0.0101
	AR-CL	0.0170	**0.0094**	0.0154	0.0093
	AR-STLW-CL	**0.0123**	**0.0094**	**0.0088**	**0.0054**

Table 4.9: Predictive accuracy (nRMSE) across different models for the viscous Burgers' (vB) equation.

(B, R, I)	Training Strategy	Model (Parameter Count)			
		FNO (140K)	**U-Net (680K)**	**U-NO (1.5M)**	**OFormer (667K)**
(128,1,256)	AR	**0.411**	**0.109**	**0.253**	**0.156**
	AR-CL	**0.411**	**0.109**	**0.253**	**0.156**
	AR-STLW-CL	0.495	0.186	0.385	0.246
(128,32,8)	AR	0.468	0.337	0.510	0.405
	AR-CL	0.367	0.335	0.466	0.398
	AR-STLW-CL	**0.350**	**0.257**	**0.441**	**0.331**

Table 4.10: Predictive accuracy (nRMSE) across different models for the Korteweg–De Vries (KdV) equation.

4.8.10 Sensitivity Analysis of the STLW Parameter

We conducted a sensitivity analysis of the STLW parameter A_c, which is a critical hyperparameter in the **AR-STLW-CL** framework. This parameter governs the rate at which temporal weights approach unity (1.0) as training progresses. Specifically, higher A_c values facilitate a rapid transition to full-sequence weighting, while lower values maintain a more gradual emphasis on earlier timesteps for a longer duration.

Figures 4.13 and 4.14 illustrate the nRMSE as a function of A_c for the viscous Burgers' and Korteweg–De Vries equations, respectively. For context, we include

horizontal reference lines representing the performance of the discrete curriculum (AR-CL) and the standard static baseline (AR).

The results demonstrate that **AR-STLW-CL remains effective across a wide range of A_c values**. While an optimal "sweet spot" exists for each physical system, the model's performance remains stable and consistently matches or exceeds that of both the AR and AR-CL baselines. This suggests that the proposed method is not excessively sensitive to precise hyperparameter tuning. From a practical perspective, this robustness is a significant advantage, as it reduces the computational overhead typically required for exhaustive hyperparameter optimisation in complex PDE solvers.

Ultimately, these findings reinforce that **AR-STLW-CL** scales more effectively than its discrete counterparts as the number of training rollouts $\widehat{T}$ increases, consistently yielding lower error rates and enhanced numerical stability.

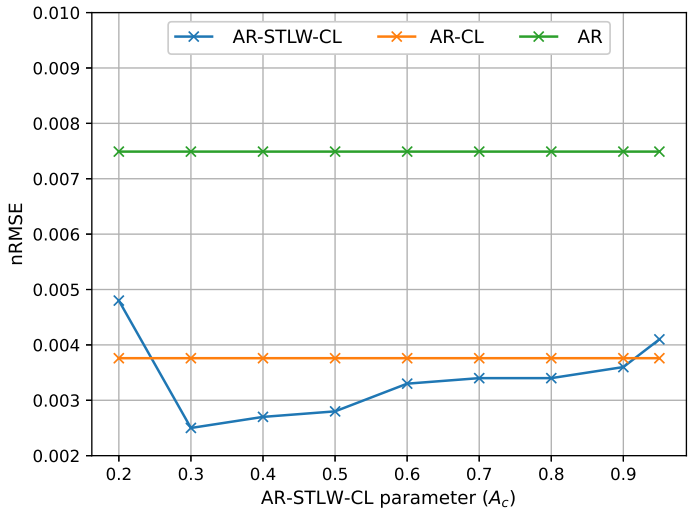


Figure 4.13: Sensitivity of the A_c parameter for the viscous Burgers' (vB) equation. Horizontal lines indicate the baseline errors for AR and AR-CL.

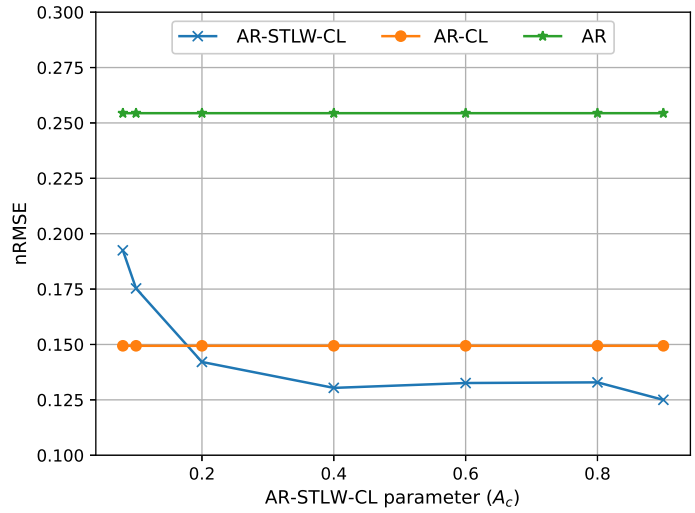


Figure 4.14: Sensitivity of the A_c parameter for the KdV equation. The stability across the x-axis demonstrates the robustness of the weighting strategy.

4.9 Discussion

In this section, we synthesise our empirical findings to address the six core research questions guiding this investigation into scheduled temporal loss weighting.

- **Q1: Baseline Teaching Strategies**
  How do the baseline strategies (TF and AR) compare with each other?

  Among the non-curriculum baselines (AR, TF, and TF+N), we observed that adding Gaussian noise (TF+N) consistently improved Teacher Forcing performance across all datasets by mitigating the distribution shift between training and inference. However, Autoregressive (AR) training proved to be the superior baseline, substantially outperforming both TF variants.

  This finding aligns with literature suggesting that AR training is particularly effective for systems that are not highly chaotic Teutsch & Mäder (2022), as it forces the model to learn to correct its own errors during the rollout. Since the PDEs investigated here (Advection, Burgers, and KdV) are governed by well-behaved physical structures rather than chaotic turbulence, AR's ability to maintain long-term temporal coherence provides a significant advantage.

- **Q2: Effectiveness of Curriculum Learning**
  How does our proposed curriculum compare with other approaches, and does it consistently outperform baselines?

  Our results demonstrate that curriculum learning (CL) significantly enhances both the accuracy and numerical stability of long-term predictions, corroborating the foundational work of Bengio *et al.* (2009) and recent applications in neural operators Takamoto *et al.* (2023); Vlachas & Koumoutsakos (2024).

  The proposed **AR-STLW-CL** framework demonstrated superior potential compared to curricula that transition between TF and AR. Notably, strategies that move toward TF (AR–TF) were the least effective, which is expected given TF's poor baseline performance on these datasets. This observation contrasts with Teutsch & Mäder (2022), who favoured increasing TF for chaotic systems. This discrepancy underscores a critical insight: the optimal curriculum direction is system-dependent. For non-chaotic physical systems, the "hard" task is usually full AR rollout; starting with simplified AR tasks (via STLW) is thus more effective than incorporating TF.

- **Q3: Impact of Training Rollout Length**
  What is the effect of training rollout on model performance and computational cost?

  Our analysis reveals that increasing the training rollout length $\widehat{T}$ does not guarantee monotonic improvements. For the AR baseline, we identified a performance "plateau" or minimum error at $\widehat{T} = 8$. Beyond this point, error accumulation during training often becomes counterproductive.

  In contrast, curriculum approaches, particularly **AR-STLW-CL**, effectively managed the increased difficulty of longer rollouts, limiting the rate of error growth. While KdV showed continued improvement beyond $\widehat{T} = 8$, researchers must weigh these marginal gains against the linear increase in computational cost (memory and time) associated with deeper rollouts.

- **Q4: Robustness in Inference Rollouts**
  What is the impact of increasing the rollout during testing for different training strategies?

  Curriculum learning significantly bolsters a model's ability to generalise to inference horizons far beyond those seen during training. In our most rigorous tests (Av and KdV), **AR-STLW-CL** emerged as the most resilient strategy. While standard AR models often diverged or produced "NaN" values during extended testing, STLW-trained models maintained physical plausibility and stable error profiles, effectively mitigating the "compounding error" problem inherent in autoregressive forecasting.

- **Q5: Robustness to Stochasticity**
  How sensitive are curriculum strategies to different random seed initialisations?

  Assessed over five independent runs, curriculum strategies generally exhibited lower variance than baselines, suggesting they smooth the optimisation landscape. While no single curriculum strategy proved to be the "most stable" across every dataset, the reduction in standard deviation compared to the AR baseline indicates that CL strategies are less sensitive to weight initialisation, making them more reliable for production-scale deployments.

- **Q6: Parameter Sensitivity**
  What is the effect of curriculum parameterisation on model performance?

A key advantage of the **AR-STLW-CL** framework is its relative insensitivity to the curvature parameter A_c. While A_c influences the transition speed of the temporal weights, the model performs robustly across a wide spectrum of values. This suggests that the "soft" transition provided by continuous weighting is fundamentally more stable than the "hard" transitions of discrete AR-CL or static AR, regardless of the exact scheduling speed. This minimises the need for expensive hyperparameter tuning, making it easier to adopt the STLW method.

4.10 Conclusion

This chapter has investigated the challenges of training neural operators for long-term forecasting of partial differential equations, with a focus on instabilities arising from autoregressive rollouts. We introduced **Scheduled Temporal Loss Weighting (AR-STLW-CL)**, a novel curriculum learning framework that conceptualises the training process as a continuous transition from short-term to long-term temporal dependencies.

Our empirical evaluation across four distinct physical systems—Advection, Viscous Burgers', Inviscid Burgers', and Korteweg–De Vries—demonstrates that AR-STLW-CL consistently outperforms standard Teacher Forcing and discrete curriculum strategies. Specifically, the proposed method achieved relative improvements in nRMSE of up to **62.5%** over the best baseline strategies. Beyond predictive accuracy, our analysis revealed several key advantages of the STLW approach:

1. **Stability:** Unlike standard autoregressive baselines that often diverge during extended inference horizons, STLW-trained models maintain better stability, effectively mitigating the accumulation of compounding errors.

2. **Architectural Agnosticism:** The framework demonstrated robust performance across diverse architectures, including FNO, U-Net, and U-NO, proving that the benefits of continuous temporal weighting are independent of model scale or parameter count.

3. **Practicality and Robustness:** Sensitivity analysis confirmed that the framework is relatively insensitive to the curvature parameter A_c. This robustness, combined with its superior performance in high-batch-size regimes, makes it an efficient, "plug-and-play" solution for researchers that requires minimal hyperparameter tuning.

4. **Multi-dimensional Scheduler:** We established that STLW acts as a multi-dimensional learning rate scheduler, providing an implicit regularising effect that yields smoother convergence trajectories compared to the "hard" transitions seen in discrete curricula.

In conclusion, the transition from discrete rollout steps to a continuous, weighted loss function represents a significant advancement in curriculum learning for time series modelling. By providing a more continuous optimisation path, **AR-STLW-CL** enables neural operators to learn the underlying physics of complex systems with greater precision and reliability. Future work could explore applying this framework to higher-dimensional chaotic fluid dynamics and integrating it with physics-informed constraints to further enhance model conservation properties.

Chapter 5

Attention Mechanism for Variable-timestep Models

5.1 Introduction

The numerical solution of partial differential equations (PDEs) is a cornerstone of scientific and engineering efforts, and obtaining accurate, efficient solutions is crucial for safety-critical applications (Zhang *et al.*, 2023). While current deep learning-based PDE solvers, specifically autoregressive neural operators, have shown great promise, their reliance on fixed-time-stepping schemes presents significant limitations. These models are often inefficient when dealing with multiscale dynamics or time-dependent solutions that tend to a steady-state, where different phases of the physical evolution would benefit from varying temporal resolutions.

An important challenge in autoregressive modelling is the accumulation of error. As established in Chapter 4, the primary cause of divergence in these models is the propagation of error from earlier to later timesteps over a large number of rollouts. Fixed or constant-timestep methods are inherently constrained by the temporal resolution of their training data; if a model is trained with a fixed timestep Δt, it lacks the flexibility to take larger steps during slower dynamics to reduce the number of rollouts or smaller steps during faster dynamics to maintain stability. Consequently, these models often struggle with long-term temporal evolution and time extrapolation (McCabe *et al.*, 2023).

To address these limitations, this chapter proposes a variable-timestep prediction framework. We argue that variable-timestep strategies, which dynamically

combine smaller and larger timesteps, can mitigate error accumulation by reducing the total number of rollouts required for a given prediction horizon. While "continuous-time" models, such as Neural Ordinary Differential Equations (Neural ODEs), offer another approach by modelling the derivative of the state, they are often computationally expensive due to the requirements of the underlying ODE solvers (Chen *et al.*, 2018).

Instead, we enhance the Fourier Neural operator (Li *et al.*, 2020b) by introducing a novel **Attention-based Temporal Conditioning** mechanism. Unlike constant-timestep rollout, which constrains a model to a specific temporal discretisation, our approach leverages temporal embeddings and a cross-attention mechanism to modulate the model's feature maps based on the requested timestep Δt. This allows the model to capture temporal dependencies, enabling accurate predictions across a continuous range of timesteps.

The contributions of this chapter are as follows:

- We introduce an attention-based conditioning architecture that enables variable-timestep sampling using Fourier Neural Operators.
- We evaluate various temporal embedding strategies, including Sinusoidal Temporal Embeddings (STE), Embedding Layer (EL) and Multi-Layer Perceptron (MLP).
- We demonstrate the effectiveness of this approach across four benchmark PDEs, showing that variable-timestep sampling can effectively manage the trade-off between efficiency and long-term stability.

5.2 Problem Statement

We consider the task of learning an explicit time-stepping operator for a general class of time-dependent PDEs over the domain $(t, x) \in [0, t_s] \times D$. The formulation mirrors that of the preceding chapters; we restate it briefly here only to introduce the temporal-embedding and conditioning notation specific to the variable-timestep setting. The system, subject to Dirichlet boundary conditions B, is defined as:

$$\begin{aligned} \partial_t u(t,x) &= \mathcal{F}^\dagger\left(u(t,x)\right), \qquad (t,x) \in (0,t_s] \times D, \\ u(0,x) &= u_0(x), \qquad x \in D, \\ u(t,x) &= B, \qquad (t,x) \in (0,t_s] \times \partial D, \end{aligned} \tag{5.1}$$

where $u : [0, t_s] \times D \to \mathbb{R}^d$ is the solution state, $u_0(x)$ is the initial condition at $t = 0$, and $\mathcal{F}^\dagger$ denotes a spatial differential operator.

Our objective is to learn a nonlinear parametric operator $\mathcal{F}_\theta$ that maps an input function $u(t_1)$ to a predicted solution $\tilde{u}(t_2)$ for any $t_2 > t_1$. We formulate this autoregressive model as:

$$\tilde{u}(t_2) = \mathcal{F}_\theta \left(\mathbf{C} \left(u(t_1), \mathbf{E}_\phi(t_1), \mathbf{E}_\phi(t_2) \right) \right), \qquad \theta \in \mathbb{R}^p, \tag{5.2}$$

where t_1 and t_2 represent the temporal coordinates of the input and output states, respectively. Here, $\mathbf{E}_\phi$ denotes a temporal embedding, which may be a fixed mapping or a learned representation with parameters ϕ, and $\mathbf{C}$ represents a conditioning mechanism that integrates the state u with the temporal information. The optimisation problem seeks to find the optimal parameters θ^* that minimise the discrepancy between the predicted and ground-truth states.

Training is performed on a finite collection of PDE trajectories originating from diverse initial conditions. While these trajectories are often provided as sequences with constant-timesteps, we employ a **variable-timestep sampling** strategy during training to simulate unevenly spaced sequences. This ensures the model learns to generalise across varying timesteps.

During training, the model is conditioned on the absolute timestamps (t_1, t_2). The loss is minimised using empirical risk minimisation over a training set, while the generalisation error is evaluated on a disjoint test set of unseen initial conditions. At inference, the trained operator $\mathcal{F}_{\theta^*}$ is applied recursively to generate long-term temporal evolutions.

Following the windowed history/horizon formulation of Chapter 3, the input at t_1 is not a single field but a block of the H most recent states, $\{u_{t_1-H+1}, \ldots, u_{t_1-1}, u_{t_1}\}$, which we denote compactly by the lowercase symbol $u(t_1)$; likewise $u(t_2)$ denotes the block of P predicted states $\{u_{t_2}, \ldots, u_{t_2+P-1}\}$. Throughout this chapter we therefore let a lowercase u_t stand for such a (possibly multi-timestep) window of the trajectory at the point t, consistent with the schematics of Figures 5.2 and 5.3. Temporal context enters the operator through two channels: (i) this window of past states, which carries the recent dynamics, and (ii) the embedded source and target timestamps $\mathbf{E}_\phi(t_1)$ and $\mathbf{E}_\phi(t_2)$, which tell the operator *where* on the trajectory the input lies and *how far* ahead the requested output is. It is precisely this pairing of a time-stamped state history with an explicitly requested target time that motivates an attention mechanism: rather than injecting a fixed temporal signal by concatenation, attention lets the requested target time t_2 query the time-stamped state history and weight the resulting transition accordingly (Section 5.7), which a static conditioning signal cannot do adaptively.

5.3 Related Work

5.3.1 Variable-timestep Modelling

In contrast to traditional numerical methods for time-dependent problems, where adaptive time-stepping is a mature and widely used technique (Söderlind, 2002), effective variable-timestep prediction for PDEs remains relatively under-explored in the context of neural operators. Much of the existing literature focuses on constant-timestep prediction, with advancements primarily targeting improved learning strategies (Lippe *et al.*, 2023; Takamoto *et al.*, 2023) and architectural refinements (McCabe *et al.*, 2023; Rahman *et al.*, 2023). For variable-timestep prediction, current approaches typically utilise embedding modules to transform temporal information into a latent representation. These embeddings are then integrated with the input function through either concatenation (Dang *et al.*, 2022) or additive operations (Gupta & Brandstetter, 2022).

5.3.2 Neural Operators

A general account of neural operators is given in the Literature Review (Chapter 2); here we note only the points pertinent to variable-timestep prediction. Among existing operators, DeepONet (Di Leoni *et al.*, 2021) has been adapted for irregular temporal sampling, for example by Diab & Al Kobaisi (2025), who proposed a temporal-branch modification to facilitate variable-timestep prediction. The present work instead builds on the FNO (Li *et al.*, 2020b), chosen for its efficacy in capturing global spectral information and its iterative architecture of learnt integral kernels, onto which we graft the temporal-conditioning mechanisms developed below.

5.3.3 Attention Mechanisms

The attention mechanism, famously introduced within the Transformer architecture (Vaswani *et al.*, 2017), has proven highly effective at capturing long-range dependencies. Consequently, several works have adapted attention-based layers to PDEs (Dang *et al.*, 2022; Kissas *et al.*, 2022; Li *et al.*, 2022c; Peng *et al.*, 2022; Takamoto *et al.*, 2022). These applications generally categorise attention into spatial or temporal contexts. For example, Li *et al.* (2022c) utilised attention to encode spatial relationships between input and query points, while Han *et al.* (2022) employed a graph neural network with an attention mechanism to capture temporal dependencies.

Transformer-based neural operators. More recently, a family of transformer-based neural operators has emerged that uses attention as the primary backbone for solving PDEs rather than as an auxiliary module. An early example is the Operator Transformer (OFormer) (Li *et al.*, 2022c), which couples self-attention over the input function with a cross-attention decoder and point-wise MLPs, making few assumptions about the sampling pattern of the input or query locations and thereby learning a discretisation-invariant solution operator; it employs a linear (Galerkin-type) attention to avoid the quadratic cost of full self-attention, and evolves time-dependent systems through a latent time-marching scheme with a fixed step size. The Convolutional Vision Transformer (CViT) couples a convolutional tokeniser with a vision-transformer encoder to learn PDE solution operators (Wang *et al.*, 2024); Transolver introduces a physics-aware tokenisation that applies attention over learnt physical states rather than raw mesh points, improving scalability on complex geometries (Wu *et al.*, 2024); and the foundation-model line of work, including Poseidon (Herde *et al.*, 2024) and Walrus (McCabe *et al.*, 2025), pre-trains large attention-based operators across families of PDEs to enable transfer to downstream systems. These architectures demonstrate the representational power of self-attention over spatial (and, in some cases, spatio-temporal) tokens.

Our use of attention differs from these approaches in both scope and placement. Rather than replacing the operator backbone with a transformer, we retain the spectral (FNO) backbone and introduce attention solely as a lightweight *temporal-conditioning* mechanism: attention is computed between the lifted function embedding and the source/target time embeddings, so that the requested timestep modulates the state transition. Whereas OFormer, CViT, Transolver, Poseidon, and Walrus apply self-attention across spatial tokens to model spatial correlations (typically at substantial computational cost), our attention operates on the temporal axis to enable variable-timestep prediction, while leaving the resolution-invariant spectral convolution to handle spatial structure. The contrast is sharpest with OFormer: it marches time in a learnt latent space with a *fixed* step, so variable-timestep prediction is not native to its design, whereas variable-timestep conditioning is the specific capability our mechanism is built to provide. In this work we therefore leverage the attention mechanism specifically to condition the model on temporal dependencies through a novel conditioning framework, and we adopt OFormer as a representative transformer-based operator for empirical comparison on the harder, non-stationary advection problem (Section 5.8.10).

5.4 Time Conditioning

The efficacy of variable-timestep modelling with neural operators depends fundamentally on how temporal information is integrated into the network architecture. This process, termed *time conditioning*, enables the model to interpret and leverage the system's temporal evolution, and we separate it into two stages: *time embedding*, which transforms the raw temporal scalars into latent vectors (classified as learnt/non-learnt and discrete/continuous, and described in the following section), and a *conditioning mechanism*, the architectural operation that incorporates those embeddings into the hidden states. The two conditioning mechanisms most commonly used in the literature are addition and concatenation (Dang *et al.*, 2022; Gupta & Brandstetter, 2022), both of which are illustrated within a standard FNO architecture in Figure 5.1. We describe both below for completeness, focusing on concatenation, which serves as the baseline conditioning mechanism throughout this chapter, before introducing the attention-based mechanism that is the central contribution of the chapter.

Addition. Additive conditioning sums the temporal embedding directly with the spatial feature representation, and therefore requires the two tensors to share identical dimensions. It can be regarded as a constrained form of concatenation in which the weight matrices associated with the summed inputs are implicitly tied, and is the operation underlying residual connections (He *et al.*, 2016); additive temporal conditioning has also been employed in the neural-operator literature, for example by Gupta & Brandstetter (2022). Conceptually, it treats time as a "delta" that modifies an existing state rather than generating new features.

Concatenation. Concatenation instead appends the temporal information, whether a scalar or an embedding vector, as additional feature channels, and is the baseline conditioning mechanism adopted in this work (Dang *et al.*, 2022). If a layer processes an input $\mathbf{u}(\mathbf{t}) \in \mathbb{R}^{C \times N_x \times N_y}$ and the temporal embedding is expanded to $\mathbf{E} \in \mathbb{R}^{C' \times N_x \times N_y}$, concatenating them along the channel dimension yields a tensor in $\mathbb{R}^{(C+C') \times N_x \times N_y}$. In contrast to addition, concatenation places no dimensional constraint on the embedding and preserves both signals in full rather than superposing them: it expands the feature space, allowing the subsequent spectral and pointwise layers to learn arbitrary relationships between the spatial and temporal features, at the cost of a larger input dimensionality and hence a higher parameter count and memory footprint in the following layers. Wu *et al.* (2022), for instance, used concatenation to condition an operator on static PDE parameters alongside the solution function.

Both addition and concatenation are established, general-purpose conditioning strategies, and we have described addition here for completeness. In the

experiments that follow, however, we retain *concatenation* as the sole baseline conditioning mechanism against which the proposed attention mechanism is evaluated: all reported results therefore use either concatenation or attention, and additive conditioning is not evaluated further. Concatenation is preferred as the baseline because it imposes no dimensional constraint on the temporal embedding and keeps the temporal and spatial information separable, making it a more faithful reference point for a mechanism, such as attention, whose purpose is to modulate the hidden state adaptively rather than merely augment it.

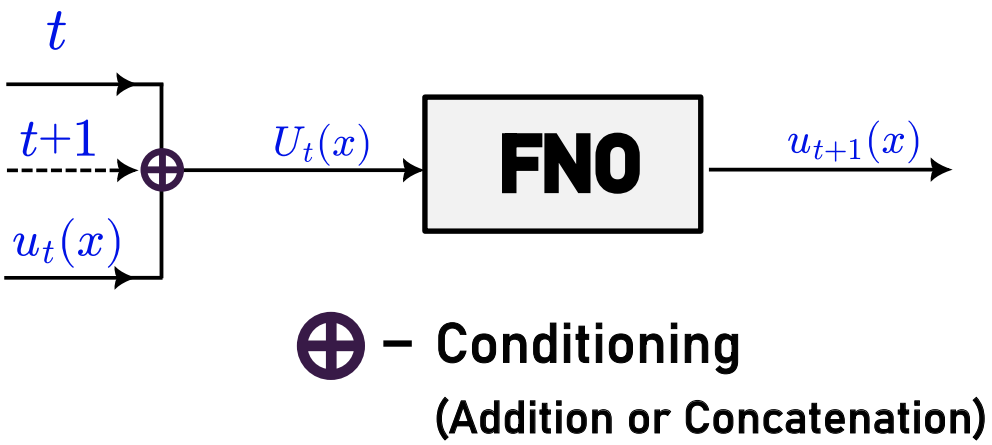


Figure 5.1: Schematic of the standard Fourier Neural Operator (FNO) architecture featuring baseline temporal conditioning. The diagram illustrates the lifting of the input function u_t into a higher-dimensional space, followed by iterative layers where the time step Δt is integrated via one of the two baseline conditioning mechanisms, (a) addition or (b) concatenation. Each layer performs a spectral convolution in the frequency domain and a local linear transformation in the spatial domain, before a final projection to yield the prediction $u_{t+\Delta t}$.

5.5 Time Embedding

Beyond a simple identity transformation of the timestamps, more sophisticated encoding schemes can represent temporal information, particularly for models that lack inherent sequence awareness. As illustrated in Figure 5.2, we denote the input and output temporal embeddings as $\hat{t}$ and $\widehat{t+1}$, respectively, where the raw temporal values are processed by the encoding layers $E_{\phi,t}$ and $E_{\phi,t+1}$. We

categorise these embeddings into two primary dimensions: *learnt vs non-learnt* and *continuous vs discrete*. For instance, a Multi-Layer Perceptron (MLP) constitutes a learnt, continuous embedding; the mapping is optimised via learnable parameters during training and can evaluate any continuous timestamp, even if training was restricted to discrete intervals. This work explores various combinations of these properties, omitting only non-learnt continuous embeddings, which are reserved for future research.

5.5.1 Non-learnt Time Embedding

Non-learnt embeddings, such as the Sinusoidal Temporal Embedding (STE) originally popularised in the Transformer architecture (Vaswani *et al.*, 2017), provide high-dimensional, deterministic representations of temporal structures. Closely related constructions, in which a scalar coordinate is lifted through a bank of fixed sinusoids of geometrically spaced frequencies, have been shown to let coordinate networks represent high-frequency functions that they would otherwise fail to capture (Tancik *et al.*, 2020). This method timestamps input functions by integrating pre-defined sinusoidal waves of varying frequencies into the feature maps. This mechanism is often crucial for models to generalise to unseen timesteps by leveraging the periodic properties of the basis functions.

5.5.2 Learnt Time Embedding

As an alternative to fixed sinusoidal patterns, temporal positions can be encoded using learnable embeddings, an approach exemplified by learned temporal representations such as Time2Vec (Kazemi *et al.*, 2019), which augments a learnable linear term with learnable periodic components. In this approach, the neural network optimises the representation for each timestep during training. While more flexible and potentially better suited to complex temporal patterns, learnable embeddings may be prone to overfitting the specific time range present in the training data. In this work, we evaluate two specific implementations: an Embedding Layer (EL) and an MLP. The EL utilises a one-hot encoding strategy, which restricts the temporal space to discrete values, whereas the MLP facilitates the modelling of a continuous temporal domain.

Mathematical formulation of the embedding-based model. The architecture in Figure 5.2 can be written explicitly as follows. The raw input and output timestamps t and $t + \Delta t$ are first mapped to embedding vectors by the

encoders $E_{\phi,t}$ and $E_{\phi,t+1}$,

$$\hat{t} = E_{\phi,t}(t) \in \mathbb{R}^{d_e}, \qquad \widehat{t+1} = E_{\phi,t+1}(t+\Delta t) \in \mathbb{R}^{d_e}, \tag{5.3}$$

where d_e is the embedding dimension and ϕ denotes the (optional) learnable embedding parameters. The input field is lifted to a hidden representation $v_0 = \mathcal{P}(u_t) \in \mathbb{R}^{d \times N_x \times N_y}$ by a pointwise operator $\mathcal{P}$, and each Fourier layer $\ell = 0, \ldots, L-1$ updates the hidden state while conditioning on the temporal embeddings via the chosen mechanism $\mathbf{C}$ (concatenation or the proposed attention; Section 5.7),

$$v_{\ell+1} = \sigma\Big(\mathbf{C}\Big((\mathcal{K}_\ell v_\ell + W_\ell v_\ell),\ \hat{t},\ \widehat{t+1}\Big)\Big), \tag{5.4}$$

where $\mathcal{K}_\ell$ is the spectral convolution (a learnt integral kernel applied in Fourier space), W_ℓ is the pointwise (1×1) linear transform, and σ is a non-linear activation. After L layers, a projection operator Q returns the prediction to physical space, $\tilde{u}_{t+\Delta t} = Q(v_L)$. The temporal embeddings $\hat{t}$ and $\widehat{t+1}$ are shared across all layers, so that every layer is informed of both the current time and the requested target time; setting $\mathbf{C}$ to the identity on its first argument (i.e. ignoring $\hat{t}, \widehat{t+1}$) recovers the standard FNO without temporal conditioning.

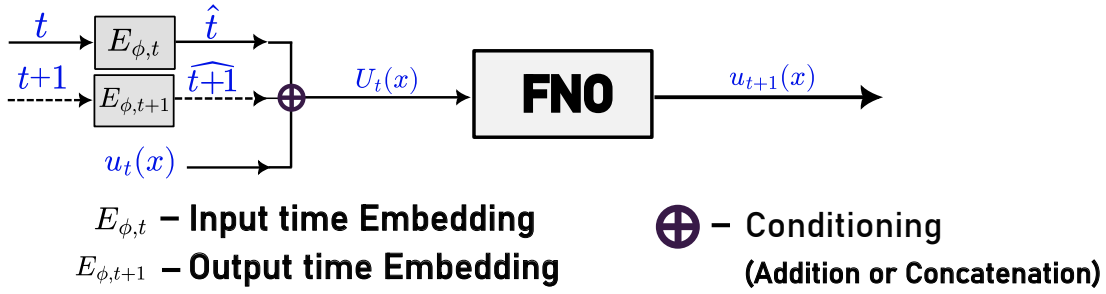


Figure 5.2: Schematic of the Fourier Neural Operator (FNO) architecture integrated with temporal embedding modules. Unlike simple scalar conditioning, this model employs learnt or non-learnt embedding functions, $E_{\phi,t}$ and $E_{\phi,t+1}$, to transform the raw input and output timestamps into continuous vector representations, $\hat{t}$ and $\widehat{t+1}$. These embeddings are then incorporated into the spectral layers to guide the evolution of the hidden feature maps from the current state u_t to the predicted state $u_{t+\Delta t}$.

5.6 Attention-Based Time Conditioning

In this section, we present our proposed attention-based temporal conditioning framework within the standard FNO architecture. This approach enables flexible variable-timestep prediction and sampling, as illustrated in Figure 5.3. The modified FNO architecture incorporates an attention module alongside the two fundamental components of the standard FNO: spectral convolutions and spatial convolutions. Each layer of the proposed model is defined as:

$$U_{t+1} := \sigma\left(\text{Attention}(U_t, T_t, T_{t+1}) + \text{SpectralConv}(U_t) + \text{SpatialConv}(U_t)\right), \tag{5.5}$$

where U_t and U_{t+1} denote the hidden function embeddings at the current and subsequent states, respectively. T_t and T_{t+1} represent the corresponding temporal embeddings, and σ denotes a non-linear activation function.

The central philosophy of this design is the parallel modelling of spatial and temporal dependencies. Specifically, the spectral convolution efficiently captures spatial patterns in the frequency domain, while the attention mechanism is leveraged to model temporal dependencies and modulate the state transition based on the requested timestep (Kissas *et al.*, 2022; Peng *et al.*, 2022). The spatial convolution, implemented as a point-wise (1×1) kernel, serves as a learnable bias or skip connection. A key refinement in our architecture is the sharing of temporal embeddings across all iterative layers, a design choice that has been shown to enhance training stability and predictive accuracy.

5.6.1 Input Lifting and Embedding

The spatial and spectral modules operate on the function embedding $U_t(x)$, whereas the attention block integrates the lifted state $U_t(x)$ with the temporal embeddings T_t and T_{t+1}:

$$\begin{aligned} U_t &= L_{\phi,u}(u_t), \\ T_t &= L_{\phi,t}(\hat{t}), \\ T_{t+1} &= L_{\phi,t+1}(\widehat{t+1}), \end{aligned} \tag{5.6}$$

where $\hat{t}$ and $\widehat{t+1}$ are generated by the initial embedding layers $E_{\phi,t}$ and $E_{\phi,t+1}$. These primary embeddings transform the raw timestamps into continuous or discrete vector spaces using an MLP (learnt, continuous), an EL (learnt, discrete), or an STE (non-learnt, discrete). The lifting maps $L_{\phi,u}$, $L_{\phi,t}$, and $L_{\phi,t+1}$ are linear transformations that project these representations into a higher-dimensional space of width d.

For concreteness, let N denote the number of spatial points (so $N = N_x \times N_y$ for a two-dimensional grid). The lifted function embedding is then $U_t \in \mathbb{R}^{N\times d}$, i.e. one d-dimensional feature vector per spatial location, while the temporal embeddings $T_t, T_{t+1} \in \mathbb{R}^{1\times d}$ are single d-dimensional vectors that are broadcast across all N spatial tokens so that they are conformable with U_t.

These shapes are reconciled with the windowed history/horizon formulation of Chapter 3, in which the input carries a history of H states and the output a horizon of P states, through the channel dimension. Writing c for the number of physical field components of a single state ($c = 1$ for a scalar field), the H history states are stacked along the channel axis of u_t prior to lifting, so that $L_{\phi,u}\colon \mathbb{R}^{N\times(Hc)} \to \mathbb{R}^{N\times d}$ and the hidden state $U_t \in \mathbb{R}^{N\times d}$ encodes the temporal history within its d channels rather than as a separate axis. Symmetrically, the P horizon states are recovered from the final hidden state by the output projection of Equation 5.9, $L_{\phi,u_{t+1}}\colon \mathbb{R}^{N\times d} \to \mathbb{R}^{N\times(Pc)}$. The attention module of the following subsection therefore operates over the N spatial tokens at fixed width d: the history and horizon never appear as explicit attention dimensions, and temporal context is supplied entirely through the timestamp embeddings T_t and T_{t+1}. The layer subscripts t and $t{+}1$ denote the source and target temporal coordinates t_1 and t_2 of the problem statement, not a unit step.

5.6.2 The Attention Mechanism

The attention module maps a query (derived from the target temporal state T_{t+1}) and a set of key - value pairs (derived from the current state U_t and T_t) to a modulated output. The output is computed as a weighted sum of the values, with weights determined by a compatibility function between the query and the corresponding keys. Our architecture formulates the Query (Q), Key (K), and Value (V) matrices as:

$$\begin{aligned} Q &= W_q(T_{t+1} + U_t), \\ K &= W_k(T_t + U_t), \\ V &= W_v(U_t), \\ \text{Attention}(Q,K,V) &= \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \end{aligned} \tag{5.7}$$

where d denotes the model width. The learnable projections satisfy $W_q, W_k, W_v \in \mathbb{R}^{d\times d}$, and with $U_t \in \mathbb{R}^{N\times d}$ and the broadcast temporal embeddings $T_t, T_{t+1} \in \mathbb{R}^{N\times d}$, the resulting matrices are $Q, K, V \in \mathbb{R}^{N\times d}$. The compatibility product $QK^T \in \mathbb{R}^{N\times N}$ is a spatial attention map whose rows are normalised by the softmax, and the modulated output $\text{softmax}(QK^T/\sqrt{d})\,V \in \mathbb{R}^{N\times d}$ therefore has

the same shape as the function embedding U_t, allowing it to be summed with the spectral and spatial convolution outputs in Equation 5.5. The scaling by $\sqrt{d}$ follows the standard scaled dot-product attention of Vaswani *et al.* (2017) and keeps the softmax logits well conditioned as d grows. Empirically, we observed that incorporating the function embedding U_t into both the query and key vectors significantly improved convergence during training.

5.6.3 Spectral Convolution and Output Projection

The spectral convolution transforms the input embedding into the frequency domain via the Fast Fourier Transform (FFT) to apply a complex weight tensor W_r:

$$\text{SpectralConv}(U_t) = \mathcal{F}^{-1}\left(W_r \cdot (\mathcal{F}U_t)\right)(x), \tag{5.8}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier and Inverse Fourier transforms, respectively. Finally, the updated hidden representation U_{t+1} is projected back to the physical space to yield the prediction u_{t+1}:

$$u_{t+1} = L_{\phi,u_{t+1}}(U_{t+1}). \tag{5.9}$$

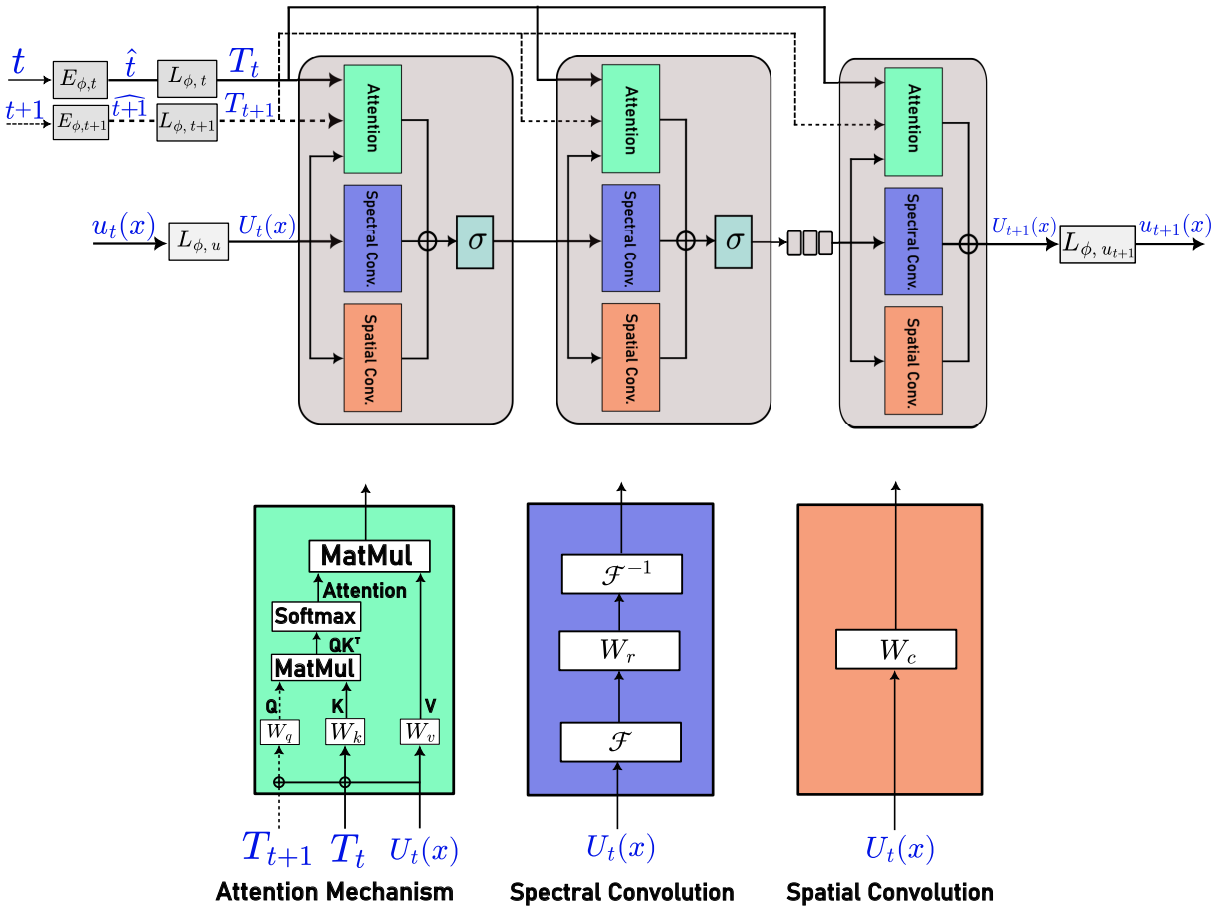


Figure 5.3: Schematic of the proposed Attention-based Fourier Neural Operator for variable-timestep prediction. The architecture processes inputs through three parallel streams: (i) a spectral convolution for frequency-domain patterns, (ii) a spatial convolution (1×1) acting as a learnable bias, and (iii) an attention mechanism that uses lifted temporal embeddings (T_t, T_{t+1}) to modulate the function embedding U_t. The outputs are summed and passed through a non-linear activation σ to produce the next hidden state U_{t+1}, which is projected back to the physical space u_{t+1}.

5.7 Experiments

This section details the experimental setup utilised to evaluate six distinct temporal conditioning and embedding strategies. We categorise these strategies by their conditioning mechanism, **Concatenation (Co)** and our proposed **Attention (At)**, each combined with three embedding types: **Sinusoidal Temporal Embedding (STE)**, **Embedding Layer (EL)**, and **Multi-Layer Perceptron (MLP)**. Table 5.1 summarises the resulting combinations (non-learnt continuous embeddings are omitted).

The performance of these strategies is benchmarked across four fundamental PDEs: Advection (Av), Viscous Burgers' (v-B), Diffusion-Reaction (DR), and Diffusion-Sorption (DS). Detailed mathematical formulations for each system are provided in Section 3.6. These benchmarks were selected to assess the models' capacity to handle linear translation, nonlinear evolution, and complex source-driven dynamics.

Specifically, the Advection equation serves as a baseline for simple translation (Section 3.6.1), while the viscous Burgers' equation introduces nonlinear dynamics where advection and diffusion are balanced (Section 3.6.2). The Diffusion-Sorption equation models coupled processes in which molecules permeate a material while being sorbed into its structure (Section 3.6.5), and the Diffusion-Reaction equation describes systems in which substances simultaneously diffuse and interact chemically (Section 3.6.4).

To ensure a rigorous comparison, all models were standardised with 16 Fourier modes, a hidden dimension $d = 32$, and 4 iterative layers (see Section 3.2.2). Hyperparameter exploration was conducted on the number of modes and hidden dimensions to optimise the trade-off between predictive accuracy and computational efficiency. Training was performed using the Adam optimiser with a batch size of 128 and a temporal rollout of $R = 8$. We employed an initial learning rate of 10^{-3}, modulated by a step-size scheduler that applies a decay factor of 0.95 per epoch.

For each sample within a training batch, we stochastically generate a rollout sequence $\widehat{T}$. For a given trajectory of T timesteps, we subsample $\widehat{T}$ intervals; this operation is performed 100 times per epoch. We employ a multistep prediction of 5, i.e., the input (history) and output (horizon) are equal. Models are trained for 200 epochs using a normalised Root Mean Square Error (nRMSE) loss function, as defined in Equation 3.22. The training process employs a variable-timestep sampling strategy (Algorithm 1), which exposes the model to a continuous range of temporal scales. Further details regarding this strategy are provided in Section 5.7.1.

As shown in Figure 5.4, each method was trained until convergence was achieved. All six strategies exhibited stable convergence within 200 epochs, using

the viscous Burgers' equation as a representative case. Notably, the **Attention-based (At)** configurations demonstrated a marginally smoother loss trajectory compared to the Concatenation (Co) baselines. This suggests that the attention mechanism facilitates more stable gradient flow during the early stages of learning, when the model is first exposed to variable timesteps.

Strategy	**Conditioning**		**Embedding**			
	Concatenation	Attention	Non-learnt	Learnt	Discrete	Continuous
STE-Co	✓		✓		✓	
STE-At (**ours**)		✓	✓		✓	
EL-Co	✓			✓	✓	
EL-At (**ours**)		✓		✓	✓	
MLP-Co	✓			✓		✓
MLP-At (**ours**)		✓		✓		✓

Table 5.1: **Taxonomy of the six temporal conditioning and embedding strategies.** Checkmarks (✓) indicate the architectural combinations of conditioning mechanisms and embedding properties evaluated across the benchmark PDEs.

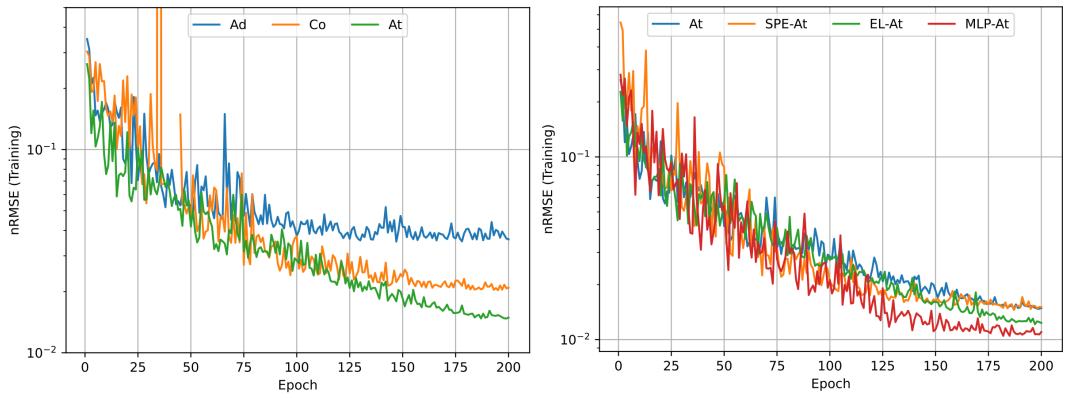


Figure 5.4: **Training convergence for the Viscous Burgers' equation over 200 epochs**. Each curve represents the mean Normalised Root Mean Squared Error (nRMSE) for one of the six strategies outlined in Table 5.1. The consistent downward trend and stabilisation across all strategies indicate robust convergence and provide a reliable baseline for the variable-timestep sampling evaluations.

5.7.1 Variable-timestep Sampling

During the training phase, we perform rollouts with a variable time step. The objective of our variable-timestep sampling strategy is to extract a randomly sampled, ordered subset $\widehat{T}$ of a fixed size $R+1$ from the temporal domain $T = \{0, 1, \ldots, t_s\}$. This subset is constructed such that the intervals between consecutive elements are non-constant. Here, R denotes the number of autoregressive rollouts—the count of recursive model applications where the prediction at one step serves as the input for the next. This sampling strategy is critical because the underlying temporal assumptions significantly affect the model's ability to predict variable-timestep outcomes.

To implement this, we independently sample a start time t_{start} and an end time t_{end} from T for each iteration, subject to the constraint $t_{\text{end}} \geq t_{\text{start}} + R$. Within the interval $[t_{\text{start}}, t_{\text{end}}]$, we randomly select $R-1$ elements to form the subset $\widehat{T}$. The magnitude of R directly influences the sampling distribution; specifically, as R increases, the probability of observing large temporal gaps between consecutive elements in $\widehat{T}$ decreases. In this work, we keep R constant throughout training. To enhance data diversity, t_{start} is sampled independently for each trajectory within a single batch. This stochastic approach ensures the model is exposed to a broad spectrum of temporal scales, as detailed in Algorithm 1.

Algorithm 1: Training with Variable-timestep Sampling

Input: Epochs E, iterations per epoch I, time domain $T = \{0, 1, \ldots, t_s\}$, training samples $\{u_t\}_{t \in T}$, rollout count R, model $\mathcal{F}_\theta$, loss criterion $\mathcal{L}$.

for $e = 1$ **to** E **do**
  for $i = 1$ **to** I **do**
    $t_{\text{start}} \sim \text{Uniform}(0, t_s - R)$
    $t_{\text{end}} \sim \text{Uniform}(t_{\text{start}} + R, t_s)$
    $\widehat{T} = \text{Sort}(\text{SampleWithReplacement}(\text{range}(t_{\text{start}}, t_{\text{end}}), \text{size} = R + 1))$
    Total Loss $= 0$
    $\tilde{u}_{t_0} = u_{t_{\text{start}}}$
    for $j = 1$ **to** R **do**
      $t_{curr} = \widehat{T}[j-1], \quad t_{next} = \widehat{T}[j]$
      $\tilde{u}_{t_{next}} = \mathcal{F}_\theta(\tilde{u}_{t_{curr}}, t_{curr}, t_{next})$
      Total Loss $=$ Total Loss $+ \mathcal{L}(\tilde{u}_{t_{next}}, u_{t_{next}})$
    end for
    $\theta \leftarrow \text{Optimiser}(\text{Total Loss})$
  end for
end for

Table 5.2: **Variable-timestep Sampling Strategy for Autoregressive Training:** This procedure generates non-uniform temporal rollout sequences. For each iteration, a random temporal window $[t_{\text{start}}, t_{\text{end}}]$ is selected, from which an ordered subset $\widehat{T}$ of size $R + 1$ is sampled. This ensures the intervals between consecutive elements are non-constant, exposing the model to diverse temporal scales (Δt) to improve long-term stability and capture multiscale dynamics.

5.8 Results

We evaluate the performance of various time conditioning and embedding methods across several time-sampling strategies for each benchmark equation: Advection (Section 5.8.3), Burgers' (Section 5.8.4), Diffusion-Reaction (Section 5.8.5), and Diffusion-Sorption (Section 5.8.6). Our findings are summarised in Table 5.3, which identifies the optimal conditioning and embedding strategies and the most effective sampling regimes for each system.

The results demonstrate that **Attention conditioning** consistently outperforms all other mechanisms across all equations. Regarding embedding types, the **Embedding Layer (EL)** proved most effective for three of the four systems, while the **Multi-Layer Perceptron (MLP)** yielded the best performance for the viscous Burgers' equation. Analysis of variable-timestep-stepping reveals that for Advection and Diffusion-Reaction, a "small-to-large" strategy (increasing the timestep during evolution, e.g., $\Delta t = 1 \rightarrow 2$) is optimal. Conversely, for the viscous Burgers' and Diffusion-Sorption equations, a "large-to-small" strategy (decreasing the timestep over time, e.g., $\Delta t = 3 \rightarrow 2 \rightarrow 1$) produces superior results. Detailed discussion and physical justifications for these observations are provided in the following subsections.

5.8.1 Experimental Configuration and Metrics

For each PDE, we evaluate six distinct conditioning and embedding combinations (as detailed in Table 5.1) under both constant and variable-timestep sampling. Constant-timestep sampling is investigated at intervals of $1\Delta t$, $2\Delta t$, and $3\Delta t$, where Δt represents the smallest temporal resolution of the dataset. For variable-timestep sampling, we examine transitions from small-to-large timesteps ($1\Delta t \rightarrow 2\Delta t$ and $1\Delta t \rightarrow 2\Delta t \rightarrow 3\Delta t$) and large-to-small timesteps ($2\Delta t \rightarrow 1\Delta t$ and $3\Delta t \rightarrow 2\Delta t \rightarrow 1\Delta t$). To ensure a balanced comparison, temporal sequences are split evenly between the selected timesteps. For instance, in a simulation of $200\,t_s$ using a $1\Delta t \rightarrow 2\Delta t$ strategy, $1\Delta t$ is employed for the initial 100 steps, followed by $2\Delta t$ for the remainder of the rollout.

Performance is quantified using two primary metrics: **total temporal error** and **final-time error**. These metrics assess the overall predictive trajectory and the accuracy of the terminal state, respectively. Given that different sampling strategies result in varying total step counts, we compute the total error by integrating the normalised root mean squared error (nRMSE) over the prediction horizon. For final-time accuracy, we report the nRMSE only at the last timestep. In the following tables, the minimum error per strategy (row) is indicated in **bold**, the minimum error per sampling regime (column) is underlined, and the

overall optimal configuration is highlighted in blue.

5.8.2 Analysis of Temporal Evolution

For each system, we plot the cumulative error over time to illustrate how different variable-timestep schemes, even those with identical total step counts, can yield divergent error profiles. We also provide error plots for constant-timestep sampling ($\Delta t \in \{1, 2, 3, 4, 5\}$) and compare the temporal evolution of error between constant- and variable-timestep regimes. This allows for a clear visualisation of the impact of switching strategies mid-rollout.

In Section 5.8.8, we benchmark a single variable-timestep model against specialised constant-timestep models (the latter being trained exclusively on a single fixed Δt). By treating constant-timestep models as a "gold standard" or an upper bound on accuracy, we quantify the performance trade-off associated with the flexibility of our variable-timestep architecture. Finally, Section 5.8.9 assesses the model's ability to adapt to non-stationary systems, i.e., dynamic physical parameters, specifically where the wave speed $\beta(t)$ in the Advection equation varies over time (step, linear-increasing, and linear-decreasing). This simple test evaluates the conditioning mechanism's capacity to capture the non-trivial interaction between the temporal step size and time-varying physics.

Dataset	**Variable-Timestep Model**		
	Best Strategy	**Best Sampling**	**Best Variable Strategy**
Advection (Av)	EL-At	Variable	$\Delta t = 1 \rightarrow 2$
Viscous Burgers' (vB)	MLP-At	Constant and Variable	$\Delta t = 3 \rightarrow 2 \rightarrow 1$
Diff.-Reaction (DR)	EL-At	Constant	$\Delta t = 1 \rightarrow 2 \rightarrow 3$
Diff.-Sorption (DS)	EL-At	Constant	$\Delta t = 3 \rightarrow 2 \rightarrow 1$

Table 5.3: Summary of optimal configurations and sampling strategies for each dataset based on total temporal error.

5.8.3 Advection

The 1D advection equation is the simplest test case, in which the solution is purely translated at a constant speed parameter $\beta = 0.5$. Because the system lacks dissipative mechanisms like diffusion, it can be sensitive to the accumulation of numerical or model-induced errors during long-term autoregressive rollouts.

Overall, the **Attention-based (At)** conditioning mechanism combined with the **Embedding Layer (EL)** architecture achieved the most robust performance. As shown in Table 5.4, the **EL-At** configuration using a **1-2-3** variable-timestep sampling strategy yielded the lowest total error. However, when evaluating the accuracy of the final state (Table 5.5), the **1-2** variable strategy with EL-At was superior. Across all tests, Attention (At) consistently outperformed Concatenation (Co).

Analysis of Constant-Time Sampling

Figure 5.5 illustrates the error propagation for models evaluated at fixed temporal intervals ($\Delta t = 1$ to 5).

- **Stability at Small Timesteps:** The error propagation for $1\Delta t$ and $2\Delta t$ remains the lowest and follows nearly identical linear trajectories. This suggests that the model accurately captures wave translation at these time resolutions.

- **The $2\Delta t$ Threshold:** A significant performance degradation is observed once the timestep exceeds $2\Delta t$. The error curves for $3\Delta t$, $4\Delta t$, and $5\Delta t$ are significantly worse than those for $1\Delta t$ and $2\Delta t$, with the most pronounced increase occurring between $3\Delta t$ and $4\Delta t$. In the context of advection, this indicates that the model's ability to map the function over longer temporal steps diminishes, likely due to the spectral layers' limitations in capturing high-frequency shifts at large Δt.

Analysis of Variable-Timestep Sampling

The efficacy of variable-timestep sampling is highly dependent on whether the sampling includes the "high-error" timesteps identified in the constant-timestep sampling error curves (i.e., $\Delta t > 2$).

- **Increasing timestep sampling (Figure 5.6):** Transitioning from $1\Delta t \rightarrow 2\Delta t$ and $1\Delta t \rightarrow 2\Delta t \rightarrow 3\Delta t$ (i.e., Small-to-Large) shows an improvement over constant $1\Delta t$. This demonstrates that once the model has established a stable trajectory over small timesteps, it can transition to larger steps

to complete the rollout with less cumulative error. Also, we observe that $1\Delta t \rightarrow 2\Delta t$ performs better than $1\Delta t \rightarrow 2\Delta t \rightarrow 3\Delta t$, most likely due to the use of "error-prone" $3\Delta t$ in the later sampling.

However, strategies starting with a larger steps (e.g., 2-3 or 2-3-4) do not show improvement over constant $2\Delta t$, as the sampling strategy contains timesteps (i.e., $3\Delta t$ and $4\Delta t$) that have high-error (as seen in Figure 5.5), which compromises the entire rollout's integrity.

- **Decreasing timestep sampling (Figure 5.7):** Conversely, decreasing the timestep during the rollout can also be beneficial (i.e., Large-to-Small). The $2\Delta t \rightarrow 1\Delta t$ strategy improves upon a constant $2\Delta t$ rollout. Notably, the $3\Delta t \rightarrow 2\Delta t \rightarrow 1\Delta t$ strategy significantly outperforms the $3\Delta t \rightarrow 2\Delta t$ and $3\Delta t$ sampling. This suggests that "correcting" the trajectory with smaller, more accurate steps toward the end of a rollout can mitigate some of the error propagation introduced by the early, larger timesteps.

Cumulative Error and Accuracy Trade-offs: Figure 5.8 further confirms existing trends by showing the cumulative sum of errors. For a fixed sampling budget (i.e., different sampling with the same number of total timesteps 125 t_s), more samples from the lower-error (earlier) region ultimately reduce the total error, as seen in the case where the cumulative error for the $1\Delta t \rightarrow 2\Delta t$ and strategy is lower than that for the $2\Delta t \rightarrow 1\Delta t$ strategy. The difference is more pronounced in the comparison between $1\Delta t \rightarrow 2\Delta t \rightarrow 3\Delta t$ and $3\Delta t \rightarrow 2\Delta t \rightarrow 1\Delta t$ sampling. Starting with a larger timestep ($3\Delta t$) results in much higher total error than with a smaller timestep. We observe a significant increase in the slope of $3\Delta t \rightarrow 2\Delta t \rightarrow 1\Delta t$ at approximately $t \approx 137$, where the timestep transitions from $2\Delta t$ to $3\Delta t$, while there is no visible change in the slope when transitioning from $2\Delta t$ to $1\Delta t$ in the $1\Delta t \rightarrow 2\Delta t \rightarrow 3\Delta t$ sampling.

TIME CONDITION AND EMBED STRATEGY		Variable-Timestep MODEL						
		CONSTANT-TIME SAMPLING			Variable-Timestep SAMPLING			
		1 (201 t_s)	2 (101 t_s)	3 (67 t_s)	1-2 (150 t_s)	2-1 (150 t_s)	1-2-3 (125 t_s)	3-2-1 (125 t_s)
Advection (Av)	STE-Co	187.00	**49.39**	83.58	156.66	61.69	130.87	6.73
	STE-At (**ours**)	6.00	9.26	4.65	5.06	8.00	4.62	**4.44**
	EL-Co	2.31	1.98	2.76	1.89	**1.79**	1.85	2.46
	EL-At (**ours**)	1.20	1.14	1.93	1.07	1.08	**1.03**	1.29
	MLP-Co	2.42	2.44	3.42	2.35	**2.33**	2.45	3.15
	MLP-At (**ours**)	2.12	3.87	3.81	1.87	3.90	**1.73**	3.59

Table 5.4: Normalised Root Mean Squared Error (nRMSE) averaged over the full temporal rollout for the Advection equation: The values in **bold** and underline are the smallest error per row and column, respectively. The smallest error is highlighted in blue.

TIME CONDITION AND EMBED STRATEGY		Variable-Timestep MODEL						
		CONSTANT-TIME SAMPLING			Variable-Timestep SAMPLING			
		1 (201 t_s)	2 (101 t_s)	3 (67 t_s)	1-2 (150 t_s)	2-1 (150 t_s)	1-2-3 (125 t_s)	3-2-1 (125 t_s)
Advection (Av)	STE-Co	1.13	0.61	1.19	**0.33**	0.45	0.37	0.057
	STE-At (**ours**)	0.059	0.076	0.036	0.031	0.041	**0.027**	0.032
	EL-Co	0.018	0.018	0.022	**0.011**	0.011	0.013	0.013
	EL-At (**ours**)	0.011	0.010	0.019	**0.008**	0.009	0.009	0.009
	MLP-Co	0.021	0.016	0.032	0.017	0.016	0.016	0.018
	MLP-At (**ours**)	0.021	0.023	0.024	0.012	0.021	**0.011**	0.018

Table 5.5: Mean normalised root mean squared error (nRMSE) at **final** time: The value in **bold** and underline are the smallest error per row and column, respectively. The smallest error is highlighted in blue.

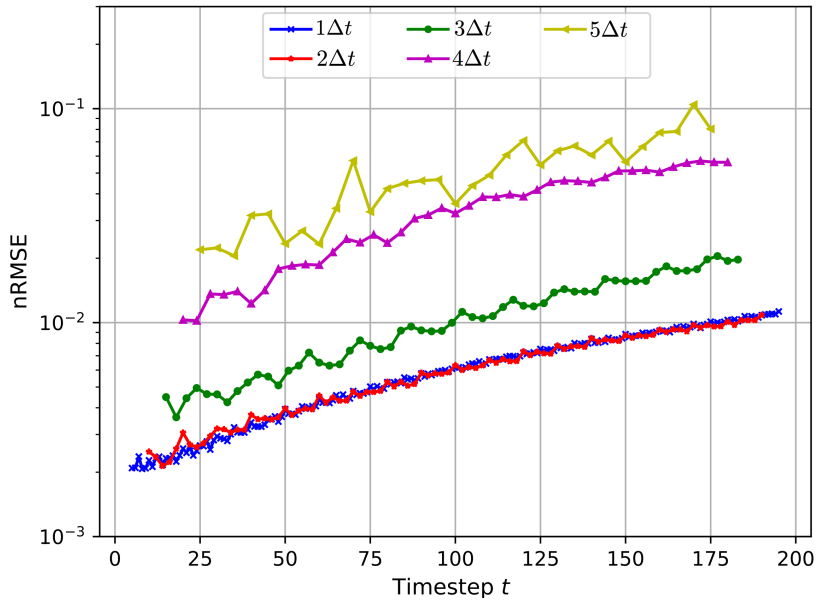


Figure 5.5: Temporal error propagation for constant-timestep sampling ($\Delta t \in \{1, \ldots, 5\}$). A performance "elbow" is observed at $\Delta t > 2$, where the error increases significantly.

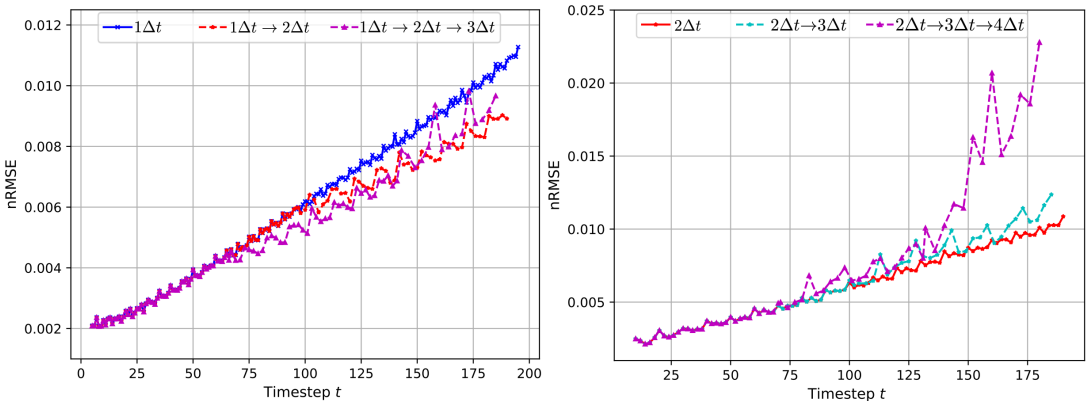


Figure 5.6: Impact of **increasing** variable-timestep sampling strategies on prediction accuracy. Strategies starting with small timesteps (e.g., 1-2 and 1-2-3) show improved stability over the constant $\Delta t = 1$ baseline, whereas strategies starting at larger intervals (2-3-4) suffer from high initial error injection.

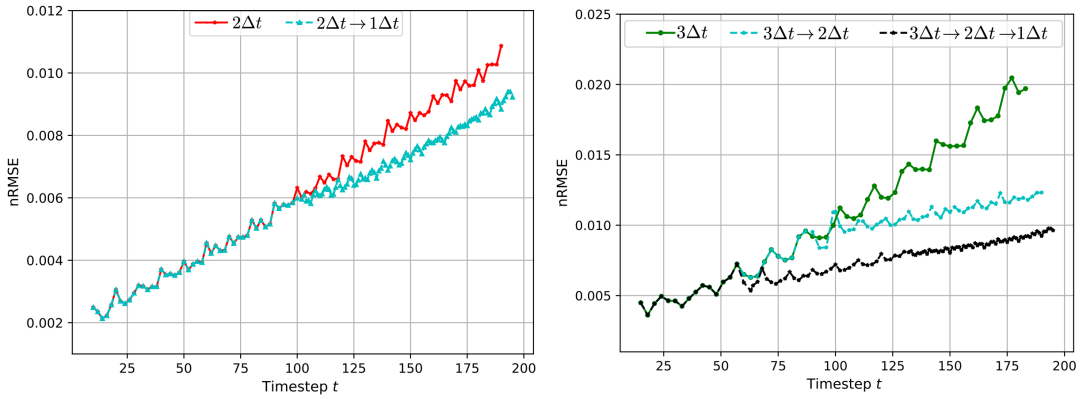


Figure 5.7: Impact of **decreasing** variable-timestep sampling strategies. The results demonstrate a "correction" effect, in which transitioning from a large to a smaller timestep (e.g., 3-2-1) mitigates cumulative error more effectively than maintaining a constant large timestep.

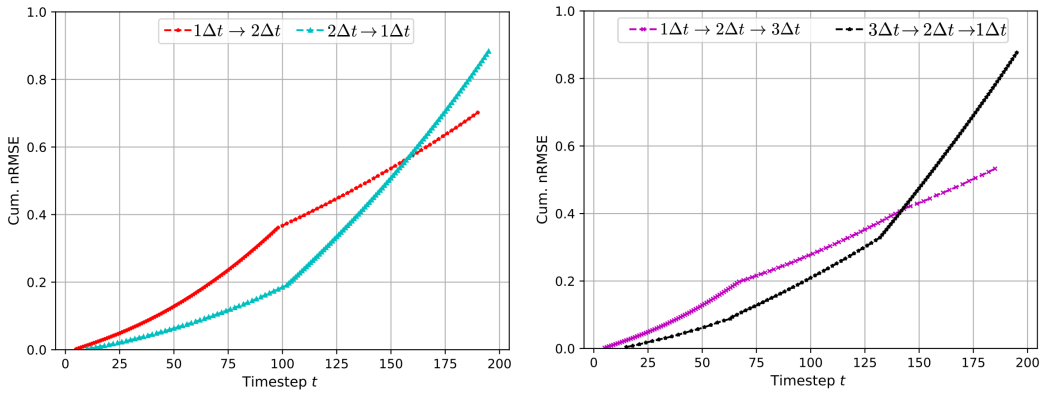


Figure 5.8: Cumulative error growth during the autoregressive rollout for the Advection equation. The near-linear accumulation of error across all models reflects the system's non-dissipative nature, in which local prediction inaccuracies are advected forward without damping.

5.8.4 Viscous Burgers

The Viscous Burgers' equation (viscosity, $\nu = 0.01$) introduces nonlinear shock formation and dissipative dynamics, providing a more rigorous test of the temporal conditioning mechanisms. Unlike the pure translation seen in advection, the model must account for changing gradients and the smoothing effect of viscosity over time.

The **MLP-based Attention (MLP-At)** architecture emerged as the top performer. As shown in Table 5.6, the MLP-At model using a $\mathbf{3\Delta t}$ (67 t_s) constant-timestep or $\mathbf{3\Delta t} \rightarrow \mathbf{2\Delta t} \rightarrow \mathbf{1\Delta t}$ ($125t_s$) variable sampling strategy achieved the lowest total error across the entire rollout.

For constant-timestep sampling, the total error decreases as the timestep increases or the number of rollouts decreases, so only 67 steps are required for a $3\Delta t$ interval. However, the best performing variable-timestep sampling (3-2-1) demonstrates superior resilience; it can achieve an equivalent error level while utilising almost twice as many rollout steps. This highlights the unique flexibility of the variable-timestep approach, which maintains high predictive accuracy even as temporal sampling increases, a task that typically increases error propagation.

A key trend across all tables is that **Attention (At)** outperforms Concatenation (Co), suggesting that the attention mechanism is better at "weighting" temporal information relative to the temporal information against the complex spatial features of the shock front. While variable-timestep strategies remain competitive, constant-timestep strategies (specifically $3\Delta t$) showed high robustness in final-time accuracy (Table 5.7).

Analysis of Constant-Time Sampling and Nonlinear Error Curve

Figure 5.9 displays a nonlinear error profile with different early and late behaviour influenced by the PDE dynamics:

- **Nonlinear Error and Early-Stage Difficulty:** Unlike the linear error growth seen in advection, the error here is nonlinear and significantly higher during the initial timesteps. This corresponds to the time phase where the solution forms a sharp shock front. The high-frequency spectral components generated during this steepening phase pose a significant challenge for the FNO's spectral layers. As time progresses, the viscous term (νu_{xx}) dominates, causing the shock to dissipate into a smooth, low-gradient solution. This transition to lower-frequency dynamics, which is easier for the FNO model to predict, explains the plateauing and eventual stabilisation of the error in later stages.

- **Early error propagation:** The constant $1\Delta t$ strategy performs best during the first 25 timesteps. In this early "fast-evolution" phase, the rapid change over time requires fine temporal resolution to be accurately resolved, especially in the presence of high spatial gradients. However, as the solution dynamics slow down for smooth solutions ($t > 25$), larger timesteps become more effective.

- **Later Error propagation:** The constant $1\Delta t$ strategy exhibits significantly higher errors in the later stages compared to $2\Delta t$ and $3\Delta t$. This is a clear manifestation of exposure bias in autoregressive modelling. Because the $1\Delta t$ model must perform three times as many rollout steps as the $3\Delta t$ model, the accumulation of small, per-step errors eventually outweighs the accuracy gained from a finer temporal mesh. By contrast, the $3\Delta t$ strategy maintains the lowest error at the end of the rollout by reaching the target state in fewer rollouts, thereby minimising error propagation. However, beyond $3\Delta t$ (i.e., $4\Delta t$ and $5\Delta t$), the error increases, indicating that $3\Delta t$ is the optimal timestep.

Analysis of Variable-Timestep Sampling and Directionality

The results for variable-timestep sampling reveal the benefit of variable-timestep sampling.

- **Increasing timestep sampling (Figure 5.10)** : This figure illustrates the impact of transitioning from small-to-large during the later stages of the evolution, where the dynamics are dominated by a smooth, low-gradient, and low-magnitude dissipative solution. Starting with a $1\Delta t$ baseline (top-left), the 1-2 and 1-2-3 variable-timestep sampling significantly reduce the final error compared to a constant $1\Delta t$ rollout. For samples starting with $2\Delta t$ (top-right) and $3\Delta t$ (bottom-middle), while the variable-timestep strategies do not outperform their respective constant-timestep counterparts, they achieve comparable accuracy with significantly fewer total rollouts. This demonstrates that once the initial shock has dissipated, the model can remain stable using coarser temporal resolutions.

- **Decreasing timestep sampling (Figure 5.11)**: In contrast, Figure 5.11 shows that reducing the timestep over time (e.g., 2-1 or 3-2-1) has a negligible or even negative impact on the final error. Because the late-stage solution is smooth and physically less demanding, the larger steps ($2\Delta t$ and $3\Delta t$) are already highly effective. Transitioning to a finer $1\Delta t$ resolution late in the rollout increases the total number of autoregressive steps, thereby allowing more opportunities for error propagation.

Cumulative Error and Rollout Stability

Figure 5.12 provides a view of these dynamics by plotting the cumulative error sum. A counterintuitive observation is that 1-2 sampling results in higher cumulative error than 2-1 sampling. This is because the 1-2 strategy requires more frequent sampling during the high-error, high-gradient initial phase. Although a larger timestep is suitable for the later stages, the error accumulated during the early "shock formation" phase. Consequently, as shown in the total error summary (Table 5.6), the **Large-to-Small** (3-2-1) strategy consistently outperforms the **Small-to-Large** (1-2-3) approach. The 1-2-3 strategy suffers from higher *exposure bias* because it relies on a greater number of small-step evaluations in the high-error phase. By using fewer, larger steps ($3\Delta t$) during the early stages, the model "leaps" over the region of highest instability, resulting in a lower total error. This confirms that, for the Viscous Burgers' equation, the optimal variable-timestep strategy minimises the number of rollouts during periods of high physical complexity.

	TIME	**Variable-Timestep MODEL**						
	CONDITION-EMBED	**CONSTANT-TIME**			**Variable-Timestep**			
	STRATEGY	1 (201 t_s)	2 (101 t_s)	3 (67 t_s)	1-2 (150 t_s)	2-1 (150 t_s)	1-2-3 (125 t_s)	3-2-1 (125 t_s)
Vis. B (vB)	STE-Co	6.04	**1.64**	2.41	5.87	1.86	5.74	2.15
	STE-At (**ours**)	2.24	1.97	**1.24**	2.27	1.80	2.34	1.33
	EL-Co	3.9	1.7	2.3	3.9	**1.6**	3.9	10.5
	EL-At (**ours**)	1.6	1.0	1.0	1.5	1.1	1.5	1.0
	MLP-Co	2.2	**1.2**	1.6	2.2	1.3	2.2	1.5
	MLP-At (**ours**)	1.8	0.93	**0.84**	1.7	0.96	1.6	**0.84**

Table 5.6: Total nRMSE for the Viscous Burgers' equation: The values in **bold** and underline are the smallest error per row and column, respectively. The smallest error is highlighted in blue. MLP-At with a 3-2-1 sampling strategy achieves the lowest error, highlighting the advantage of decreasing the timestep when dealing with non-linear shock formation.

	TIME	Variable-Timestep MODEL						
	CONDITION-EMBED	CONSTANT-TIME			Variable-Timestep			
		1	2	3	1-2	2-1	1-2-3	3-2-1
	STRATEGY	(201 t_s)	(101 t_s)	(67 t_s)	(150 t_s)	(150 t_s)	(125 t_s)	(125 t_s)
Vis. B (vB)	STE-Co	0.021	**0.0047**	0.0062	0.016	0.0087	0.017	0.0055
	STE-At (**ours**)	0.0066	0.0081	0.0039	0.0069	0.0051	0.0065	**0.0038**
	EL-Co	0.0087	0.0030	0.0045	0.0081	**0.0030**	0.0079	0.0359
	EL-At (**ours**)	0.0059	0.0034	**0.0034**	0.0043	0.0048	0.0043	0.0039
	MLP-Co	0.0077	0.0041	**0.0035**	0.0067	0.0041	0.0067	0.0041
	MLP-At (**ours**)	0.01	0.0037	**0.0024**	0.0085	0.0042	0.0071	0.0031

Table 5.7: Final time error for Viscous Burgers': The values in **bold** and underline are the smallest error per row and column, respectively. The smallest error is highlighted in blue. The constant $3\Delta t$ strategy performs well, as it minimises the number of autoregressive steps required to reach the final state.

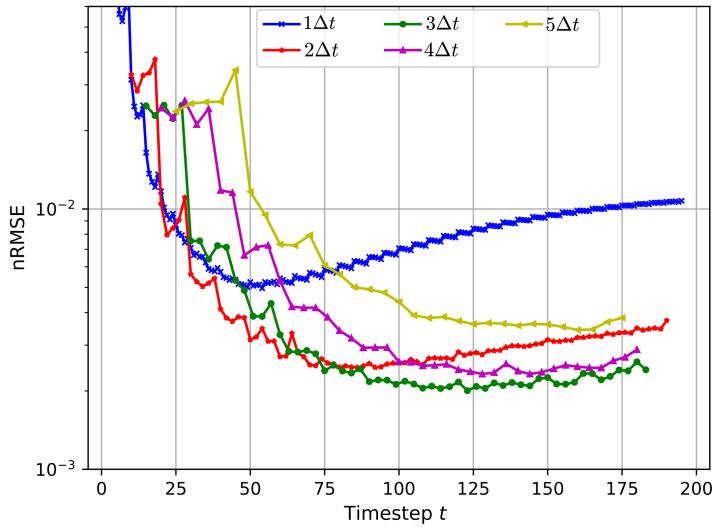


Figure 5.9: MLP-At temporal error propagation for constant timesteps. The non-linear error curves reflect the physical difficulty of capturing early-stage shock formation, with larger timesteps ($3\Delta t$) showing better long-term stability due to reduced rollouts.

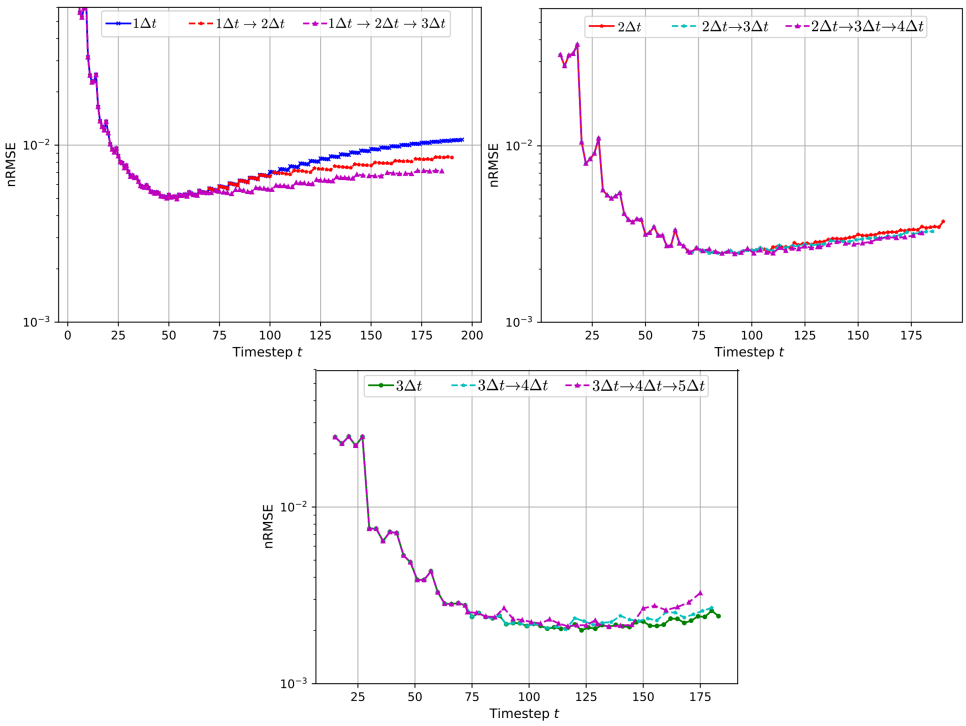


Figure 5.10: Performance of **increasing** variable-timestep steps. While 1-2 and 1-2-3 improve upon the constant $1\Delta t$ baseline, they are still hindered by high error in the early timesteps.

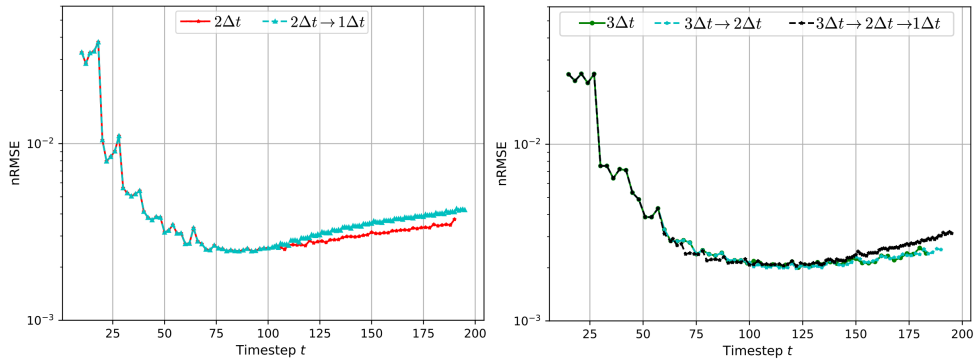


Figure 5.11: Performance of **decreasing** variable-timestep steps. Decreasing timesteps leads to worse performance because the smooth solutions are more susceptible to error propagation.

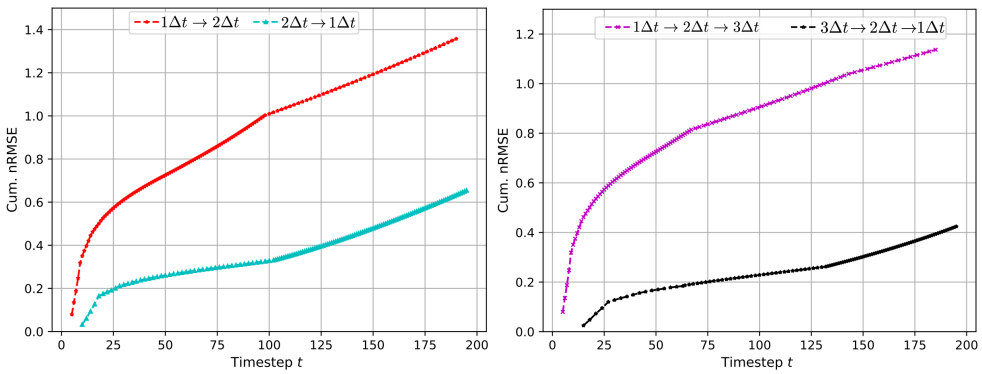


Figure 5.12: Cumulative error sum for Burgers' equation. The higher error in the 1-2 sampling compared to the 2-1 sampling demonstrates the impact of the sampling strategy.

5.8.5 Diffusion-Reaction

The Diffusion-Reaction equation presents a case in which temporal stability is challenged by potentially exponential growth driven by the reaction term.

The **Embedding Layer with Attention (EL-At)** architecture provided the most accurate predictions. As shown in Tables 5.8 and 5.9, the constant $3\Delta t$ sampling strategy achieved the lowest total and final-time errors. A consistent trend across all evaluations is the superiority of the **Attention (At)** mechanism over Concatenation, reinforcing its ability to dynamically weigh temporal features. Notably, in this system, constant-timestep sampling with a fixed step size of $3\Delta t$ outperformed variable-timestep strategies, suggesting that the benefits of a fixed, optimised temporal step outweigh the flexibility of variable-timestep sampling for this specific PDE.

Analysis of Constant-Time Sampling and Late-Stage Complexity

Figure 5.13 illustrates the temporal evolution of error for fixed timesteps:

- **Late-Stage Error Growth:** Unlike the Burgers' equation, the error for Diffusion-Reaction is lowest during the initial phases and grows non-linearly over time. This indicates that while the model effectively captures the initial diffusive spreading, the cumulative effect of the reaction term introduces complexities that are harder to resolve as the rollout progresses.

- **Optimal Sampling ($3\Delta t$):** The $1\Delta t$ strategy exhibits the highest error at later timesteps. This is attributed to the high frequency of rollouts: in a system where errors accumulate over time, increasing the number of autoregressive steps exacerbates overall divergence from the ground truth. The $3\Delta t$ strategy achieves the best balance, providing enough temporal density to capture the reaction dynamics while minimising the exposure bias inherent in long rollouts.

Variable-Timestep Sampling and Cumulative Error Analysis

The performance of variable-timestep strategies is heavily influenced by the sampling region in reference to error propagation:

- **Increasing Vs Decreasing strategy (Figures 5.14 & 5.15):** Increasing the timestep (e.g., 1-2-3) show improvement over the constant $1\Delta t$ baseline. Conversely, decreasing strategies like the 3-2-1 sampling, significantly increases error upon constant $3\Delta t$ result. This is because the $3\Delta t$ timestep is already highly optimised for this equation; adding smaller, more error-prone steps ($1\Delta t$) into the rollout degrades the overall performance.

- **Cumulative Error (Figure 5.16):** While the "small-to-large" strategy (1-2-3) starts with higher error than 3-2-1, it ends with a lower total cumulative error. This is because 1-2-3 utilises larger timesteps ($3\Delta t$) during the final stages of the rollout, when the system is most prone to error. By reducing the number of samples taken during this high-error late phase, the 1-2-3 strategy effectively limits the total error compared to 3-2-1.

	TIME	**Variable-Timestep MODEL**						
	CONDITION-EMBED	**CONSTANT-TIME**			**Variable-Timestep**			
	STRATEGY	1 (201 t_s)	2 (101 t_s)	3 (67 t_s)	1-2 (150 t_s)	2-1 (150 t_s)	1-2-3 (125 t_s)	3-2-1 (125 t_s)
Diff-React (DR)	STE-Co	24.0	17.0	**10.4**	26.86	15.0	22.6	13.8
	STE-At (**ours**)	13.9	9.1	**7.9**	12.0	10.7	10.8	9.3
	EL-Co	11.2	9.4	**9.3**	10.8	9.6	10.4	9.5
	EL-At (**ours**)	9.4	4.8	**3.4**	7.8	6.4	6.8	5.1
	MLP-Co	15.1	10.2	**9.3**	12.7	12.0	11.4	11.2
	MLP-At (**ours**)	11.4	8.0	**7.1**	10.2	9.0	9.4	8.3

Table 5.8: Total nRMSE for the Diffusion-Reaction equation: The EL-At model with a constant $3\Delta t$ strategy achieves the best performance, suggesting that minimising rollouts is vital for stability in reactive systems. For variable-timestep sampling, the "large-to-small" strategy performs better.

	TIME	**Variable-Timestep MODEL**						
	CONDITION-EMBED	**CONSTANT-TIME**			**Variable-Timestep**			
	STRATEGY	1 (201 t_s)	2 (101 t_s)	3 (67 t_s)	1-2 (150 t_s)	2-1 (150 t_s)	1-2-3 (125 t_s)	3-2-1 (125 t_s)
Diff-React (DR)	STE-Co	0.21	0.20	**0.08**	0.31	0.17	0.21	0.15
	STE-At (**ours**)	0.14	0.087	**0.081**	0.11	0.12	0.098	0.10
	EL-Co	0.09	**0.07**	0.13	0.12	0.12	0.12	0.11
	EL-At (**ours**)	0.10	0.052	**0.031**	0.077	0.077	0.066	0.065
	MLP-Co	0.19	0.13	0.11	0.12	0.15	**0.11**	0.14
	MLP-At (**ours**)	0.12	0.094	**0.066**	0.10	0.10	0.090	0.093

Table 5.9: Final time error for Diffusion-Reaction: Constant $3\Delta t$ remains the most robust strategy, while in the variable-timestep sampling, there is no clear superior sampling strategy for our Attention-based options.

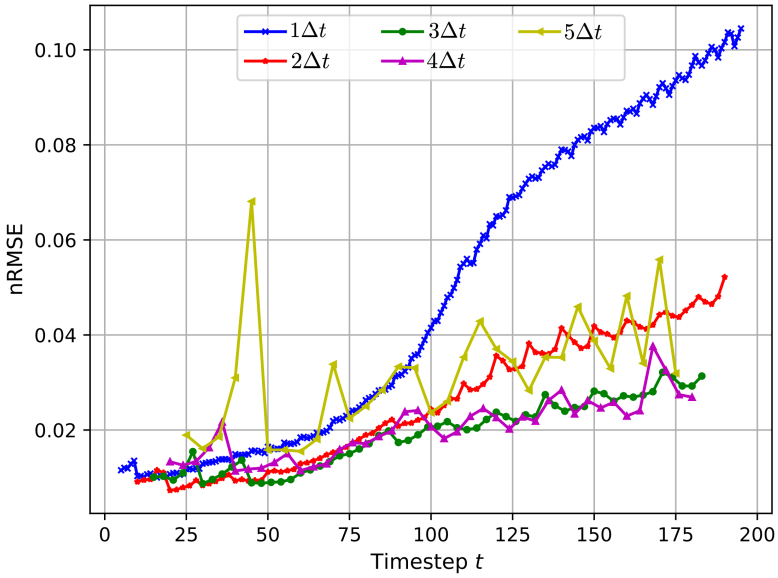


Figure 5.13: Temporal error propagation for constant-timestep sampling. The error increases over time as reaction-driven complexities accumulate, with the $1\Delta t$ model suffering the most from autoregressive error propagation.

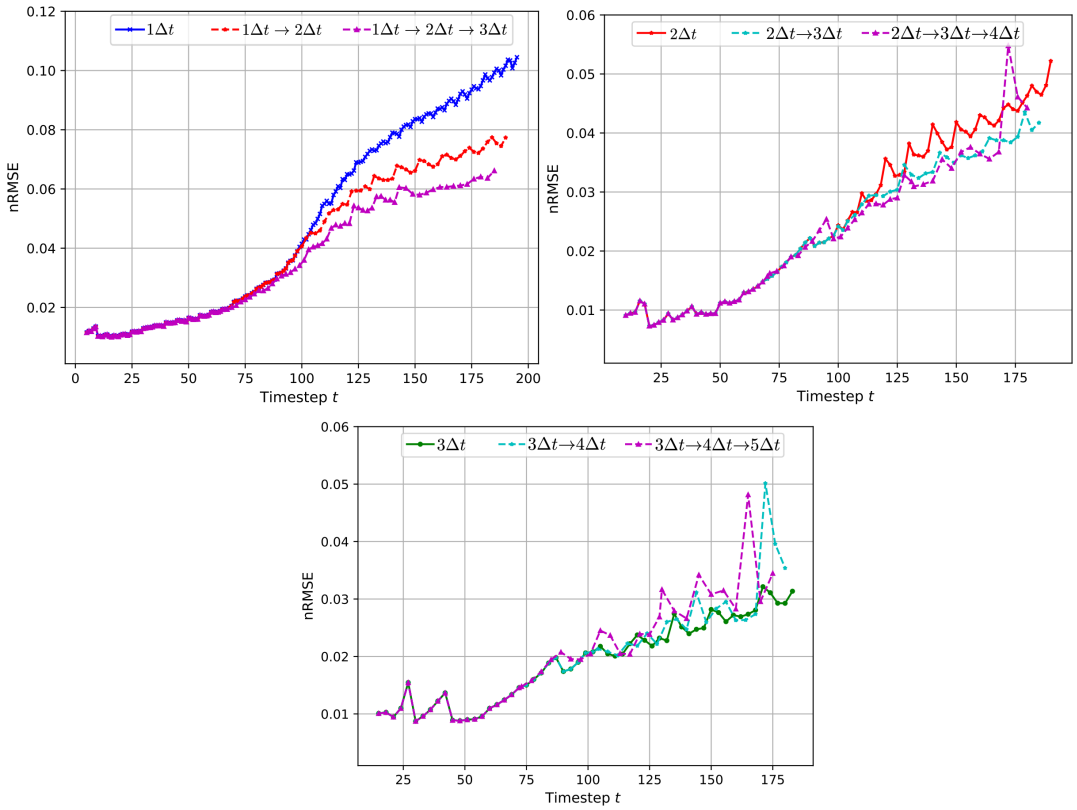


Figure 5.14: Increasing variable-timestep sampling: Strategies like 1-2 and 1-2-3 improve upon the $1\Delta t$ baseline by transitioning to fewer, larger steps as the physical system's error sensitivity increases.

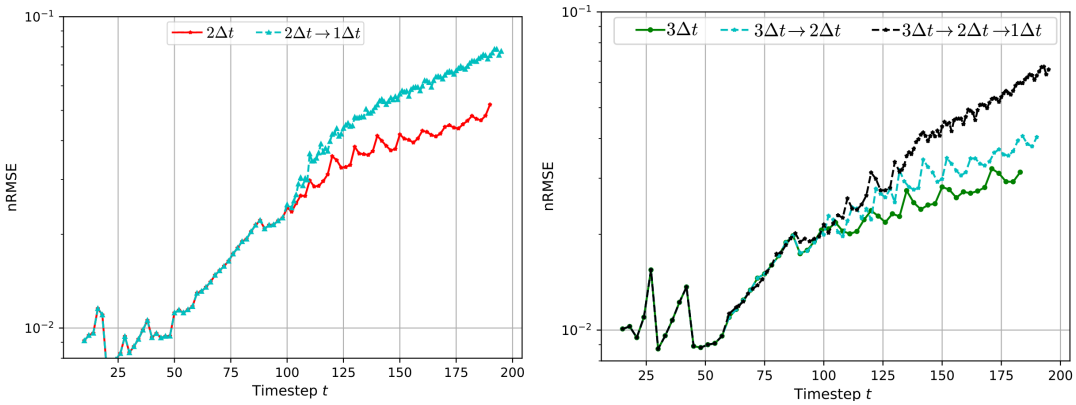


Figure 5.15: Decreasing variable-timestep sampling: The 3-2-1 and 2-1 sampling increase errors over the constant $3\Delta t$ baseline, as the introduction of smaller timesteps increases the total number of rollout steps and the associated exposure bias.

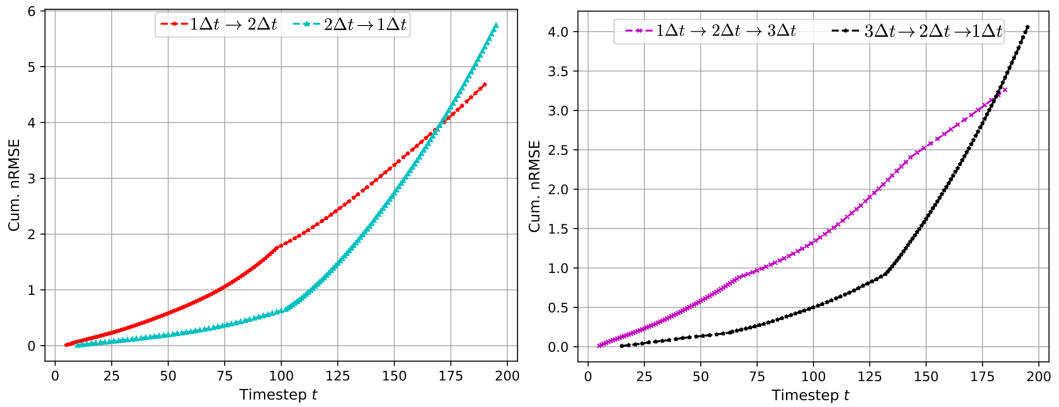


Figure 5.16: Cumulative error sum. Although 3-2-1 begins with lower error, the 1-2-3 strategy achieves a lower final cumulative total by using larger timesteps to "bypass" frequent sampling during the high-error later stages of the PDE evolution.

5.8.6 Diffusion-Sorption

The Diffusion-Sorption equation models the transport of solutes in porous media, where interactions between the liquid and solid phases (sorption) introduce non-linear retardation. This process typically creates high-gradient "fronts" early in the simulation that stabilise as the media reaches equilibrium

Analysis of the results shows that the **Embedding Layer with Attention (EL-At)** architecture is the most effective for this system. As detailed in Table 5.10, the constant $3\Delta t$ strategy achieved the lowest total error. While constant-timestep sampling generally demonstrates higher accuracy in total error metrics, the distinction between constant and variable-timestep strategies becomes negligible when evaluating the final-time error (Table 5.11). Across all tests, the **Attention (At)** mechanism consistently outperformed Concatenation, highlighting its ability to adapt to the sorption process's shifting gradients.

An observation in this PDE is the relationship between sampling density and model stability. In constant-timestep sampling, accuracy depends on the number of rollout steps; i.e., increasing the timestep (decreasing the number of rollouts) reduces error because there is less error propagation. Variable-timestep sampling, however, offers a structural advantage that enables the model to remain competitive in large rollouts without suffering significant error propagation.

Analysis of Constant-Time Sampling and Initial Dynamics

Figure 5.17 illustrates the temporal error propagation for fixed timestep and reveals that the early-time phase is the primary driver of total error:

- **Early dynamics:** There is a clear separation in performance during the initial timesteps. Smaller constant timesteps achieve higher accuracy during the early phase of the rollout ($t < 20$) because they better resolve discontinuities and sharp gradients caused by rapid chemical interactions when the solute first encounters the solid matrix.

- **Late-Stage Convergence:** As the rollout progresses, the differences between various constant timesteps ($1\Delta t$ through $5\Delta t$) diminish. Once the initial "sorption shock" has passed and the system enters a more linear, diffusion-dominant regime, the model's error profiles converge, suggesting that temporal resolution becomes less critical in the later stages of evolution, allowing for larger timesteps.

Analysis of Variable-Timestep Sampling and Cumulative Error

The variable-timestep sampling results (Figures 5.18 & 5.19) show negligible difference and the cumulative error (Figure 5.20) reveals a familiar pattern:

- **Increasing and Decreasing (Figures 5.18 and 5.18)**: For increasing variable-timestep sampling (5.18), the figure shows the effect of transitioning to larger timesteps during the later stages of the evolution, where the dynamics are dominated by a smooth, low-gradient solution. Starting with a $1\Delta t$ baseline (left) and a $2\Delta t$ baseline (top-right), the variable-timestep strategies do not outperform their respective constant-timestep counterparts but achieve comparable accuracy with significantly fewer total rollouts. This demonstrates that once the initial discontinuity has dissipated, the model can remain stable at coarser temporal resolutions.

  For decreasing variable-timestep sampling (5.19), we observe a similar behaviour: decreasing variable-timestep sampling does not significantly affect the rollout accuracy, since all constant timesteps perform similarly in the later phase (figure 5.17).

- **Cumulative Error:** Figure 5.20 shows that starting with larger timesteps ($2\Delta t$ and $3\Delta t$) and moving to smaller ones ($1\Delta t$ and $2\Delta t \rightarrow 1\Delta t$), respectively, results in a lower cumulative error than the reverse case, i.e., stating with smaller timesteps. Although smaller timesteps are more accurate in the early phase, the higher number of rollouts in the high-error region leads to larger cumulative errors. This suggests that the early dynamics of the PDE are the most difficult for the model to capture. By using larger timesteps during the high-error early phase and smaller timesteps in the later stage, the model can reduce the total cumulative error.

	TIME	**Variable-Timestep MODEL**						
	CONDITION-EMBED	**CONSTANT-TIME**			**Variable-Timestep**			
	STRATEGY	1 (201 t_s)	2 (101 t_s)	3 (67 t_s)	1-2 (150 t_s)	2-1 (150 t_s)	1-2-3 (125 t_s)	3-2-1 (125 t_s)
Diff-Sorp (DS)	STE-Co	0.70	0.29	0.33	0.68	**0.28**	0.64	0.33
	STE-At (**ours**)	0.20	0.19	**0.16**	0.22	0.22	0.19	0.18
	EL-Co	0.19	**0.16**	0.17	0.18	0.17	0.17	0.17
	EL-At (**ours**)	0.14	0.12	**0.11**	0.13	0.12	0.13	0.12
	MLP-Co	0.20	0.14	**0.13**	0.19	0.15	0.17	0.14
	MLP-At (**ours**)	0.13	0.12	0.12	0.13	0.12	0.13	0.12

Table 5.10: Total nRMSE for the Diffusion-Sorption equation. The EL-At architecture with constant $3\Delta t$ sampling achieves the lowest error.

	TIME	Variable-Timestep MODEL						
	CONDITION-EMBED	CONSTANT-TIME			Variable-Timestep			
		1	2	3	1-2	2-1	1-2-3	3-2-1
	STRATEGY	(201 t_s)	(101 t_s)	(67 t_s)	(150 t_s)	(150 t_s)	(125 t_s)	(125 t_s)
Diff-Sorp (DS)	STE-Co	0.0038	0.0013	0.0013	0.0092	0.0013	0.0027	0.0014
	STE-At (**ours**)	0.0013	0.0014	0.0010	0.0020	0.0027	**0.0011**	0.0019
	EL-Co	0.0014	0.0010	0.0010	0.0011	0.0012	0.0010	0.0012
	EL-At (**ours**)	0.0013	0.0011	**0.0008**	0.0011	0.0011	0.0009	0.0010
	MLP-Co	0.0020	**0.0009**	0.0010	0.0012	0.0015	0.0096	0.0014
	MLP-At (**ours**)	0.0009	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008

Table 5.11: Prediction error at the final timestep for Diffusion-Sorption: There is no conclusive winner between constant and variable strategies, indicating that multiple temporal paths can converge to an accurate final state.

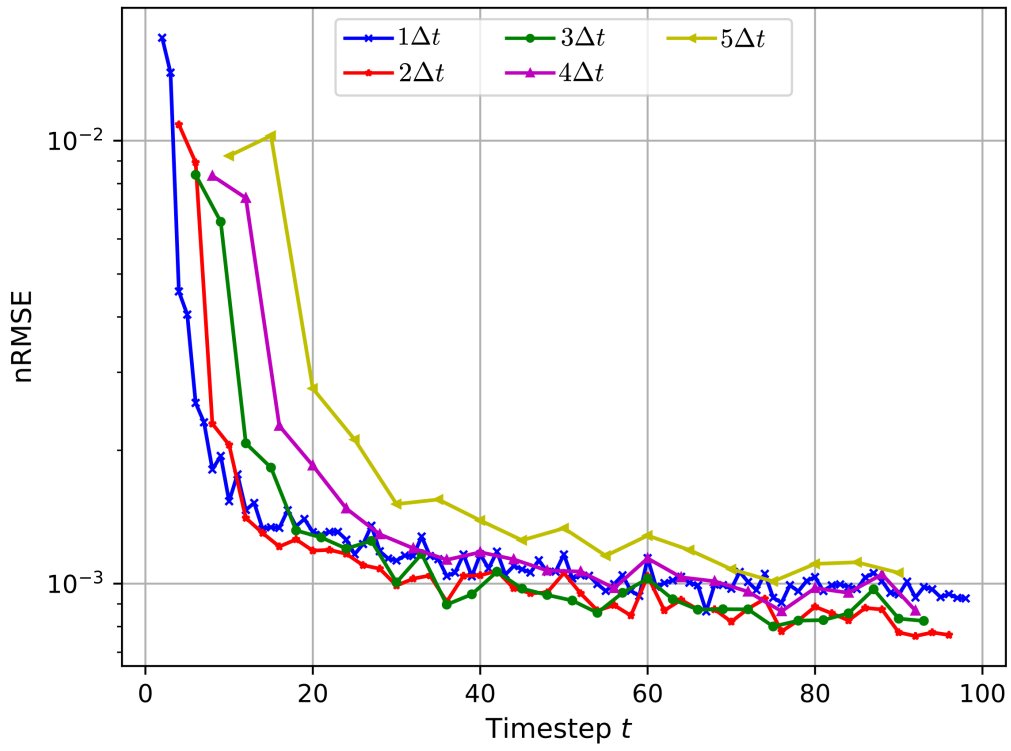


Figure 5.17: Temporal error propagation for constant timesteps. The high error at early timesteps highlights the difficulty of capturing initial sorption dynamics, with error profiles converging as the system stabilises.

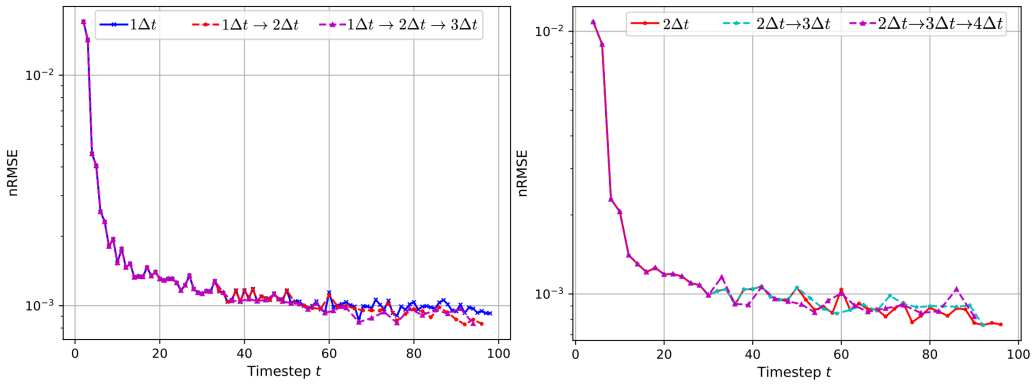


Figure 5.18: Performance of **increasing** variable-timestep sampling. Variable-timestep sampling does not improve over constant-timestep sampling.

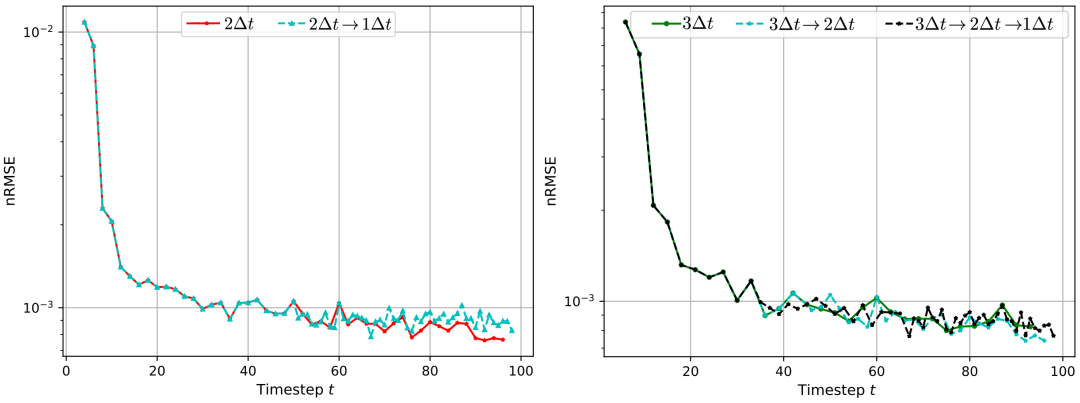


Figure 5.19: Performance of **decreasing** variable-timestep sampling. Variable-timestep sampling does not improve over constant-timestep sampling.

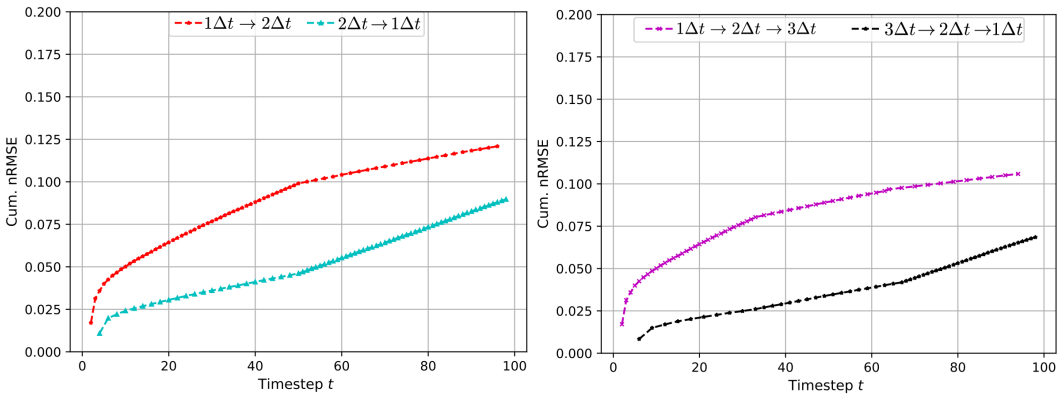


Figure 5.20: Cumulative error for Diffusion-Sorption. The higher error in the 1-2 1-2-3 sampling compared to 2-1 and 3-2-1 illustrates how a larger timestep during the high-error early phase can reduce overall error propagation.

5.8.7 Effect of model size

As shown in the Table below (5.12), we investigate the effect of model size using the viscous Burgers' equation. We double the hidden dimension d of our best-performing embedding method (EL) for the attention (At) and concatenation (Co) conditioning approaches, i.e., EL-At and EL-Co. The results show that Attention scales well and outperforms Concatenation across all time-sampling strategies. However, the Concatenation option is more computationally efficient (30% faster) due to lower memory (8% less) usage than our Attention-based approach.

	Time per Epoch (sec)	Memory (MB)	**Constant-timestep Sampling**			**Variable-timestep Sampling**			
			1	2	3	1-2	2-1	1-2-3	3-2-1
			201 t_s	101 t_s	67 t_s	150 t_s	150 t_s	125 t_s	125 t_s
EL-At	-		**1.8**	**0.93**	**0.84**	**1.7**	**0.96**	**1.6**	**0.83**
EL-Co	-		2.2	1.2	1.6	2.2	1.3	2.2	1.5
EL-At 2x param	71.8	2.49	**0.77**	**0.48**	**0.49**	**0.76**	**0.48**	**0.76**	**0.49**
EL-Co 2x param	**54.7**	**2.30**	0.98	0.64	0.69	0.95	0.65	0.96	0.68

Table 5.12: Effect of model size of the top two conditioning and embedding approaches.

5.8.8 Comparative Performance: Variable-timestep vs. Constant-timestep Models

This section evaluates the "generalisation penalty" incurred by using a single variable-timestep model rather than multiple specialised constant-timestep models. While the constant-timestep models are trained and optimised for a single, fixed temporal resolution (representing an accuracy "limit"), the variable-timestep model is a single model capable of predicting across a range of intervals ($\Delta t \in \{1, 2, 3, 4, 5\}$). Tables 5.13 and 5.14 compare these approaches across our four benchmark PDEs using two metrics: accuracy (nRMSE) and correlation (Pearson's Correlation).

Analysis of Predictive Accuracy (Table 5.13)

1. **Specialised Superiority (The Generalisation Penalty)**: In three out of the four datasets - Advection, Viscous Burgers', and Diffusion-Sorption, the constant-timestep models outperform the variable-timestep model.

    - Advection & Burgers': The specialised constant-timestep models can be about one order of magnitude better than the variable-timestep model in some intervals (e.g., vB at $1\Delta t$). This suggests that when a model is trained on a single Δt, it can perfectly over-fit the spectral transformations required for that specific timestep.
    - Diffusion-Sorption: The performance gap is minimal (0.0010 vs 0.0014), indicating that for certain highly stable dissipative systems, the variable-timestep model can almost match specialised performance.

2. **The Diffusion-Reaction Exception** We find that the single variable-timestep model consistently outperforms the constant-timestep models across all tested timesteps.

    - **Performance Gain:** At $3\Delta t$, the variable-timestep model achieves an error of 0.0192, which is nearly 50% lower than the specialised model's 0.0386.
    - **Cross-Temporal Learning:** This suggests that for complex, non-linear reaction terms, training on a variety of timesteps acts as a form of data augmentation or multi-task learning. Therefore, the variable-timestep model can learn a more robust representation of the underlying physics by seeing how the reaction evolves over different scales, whereas the constant-timestep model is limited to a single "snapshot" of the dynamics.

3. **Error Scaling with Δt**: The table reveals distinct scaling behaviours:

    - **Constant-timestep models:** Generally show decreasing error as Δt increases (e.g., Burgers' 0.0028 $\rightarrow$ 0.0013). This is attributed to the reduced number of autoregressive steps required to reach the same final time, thereby minimising error accumulation.
    - **Variable-timestep model:** Follows a U-shaped or increasing error curve in the Advection and Burgers' cases. For example, in Advection, error rises sharply from 0.0061 ($2\Delta t$) to 0.0499 ($5\Delta t$). This indicates that while the model generalises well to small steps, the complexity of mapping large timesteps in a single step challenges the conditioning mechanism.

DATASET	MODEL	CONSTANT-TIME SAMPLING				
		$1\Delta t$	$2\Delta t$	$3\Delta t$	$4\Delta t$	$5\Delta t$
Advection (Av)	Constant-timestep (5x)	**0.0034**	**0.0028**	**0.0027**	**0.0027**	**0.0025**
	Variable-timestep (1x)	0.0062	0.0061	0.010	0.0337	0.0499
Vis. Burgers' (vB)	Constant-timestep (5x)	**0.0028**	**0.0025**	**0.0021**	**0.0013**	**0.0013**
	Variable-timestep (1x)	0.0099	0.0051	0.0048	0.0062	0.0085
Diff-React (DR)	Constant-timestep (5x)	0.0515	0.0399	0.0386	0.0349	0.0361
	Variable-timestep (1x)	**0.0489**	**0.0258**	**0.0192**	**0.0203**	**0.0316**
Diff-Sorpt (DS)	Constant-timestep (5x)	**0.0011**	**0.0010**	**0.0010**	**0.0010**	**0.0010**
	Variable-timestep (1x)	0.0016	0.0014	0.0014	0.0015	0.0018

Table 5.13: Accuracy: Comparison between five constant-timestep models and one variable-timestep model. The five constant-timestep models correspond to models trained and tested at $\Delta t = 1, 2, 3, 4$, and 5, respectively. A single variable-timestep model is evaluated at the same timesteps for comparison.

Analysis of Correlation (Table 5.14)

Numerical error (nRMSE) only provides a partial view of performance; Table 5.14 utilises Pearson's Correlation (R) to assess how well the models preserve the physical structure and phase of the solution.

- **Preservation of Dynamics:** Across all PDEs, the variable-timestep model maintains exceptionally high correlation coefficients (typically $R > 0.98$). This indicates that even when minor numerical offsets occur at larger timesteps, the model correctly captures the essential physics—such as the precise location of shock fronts or the shape of a diffusing wave.

- **Structural Robustness:** The stability of the R values between $1\Delta t$ and $5\Delta t$ within the same variable model confirms that the architecture has learnt a consistent underlying evolution law. The model is not memorising transitions but is successfully approximating the continuous temporal flow of the PDE.

DATASET	MODEL	CONSTANT-TIME SAMPLING	
		$1\Delta t$	$5\Delta t$
Advection (Av)	Constant-timestep (2x)	1.0	1.0
	Variable-timestep (1x)	0.99	1.0
Viscous Burgers' (vB)	Constant-timestep (2x)	1.0	1.0
	Variable-timestep (1x)	1.0	1.0
Diff-React (DR)	Constant-timestep (2x)	0.99	0.99
	Variable-timestep (1x)	0.99	0.99
Diff-Sorpt (DS)	Constant-timestep (2x)	1.0	1.0
	Variable-timestep (1x)	1.0	1.0

Table 5.14: Pearson's Correlation: Comparison between two constant-timestep models and one variable-timestep model. The two constant-timestep models correspond to models trained and tested at $1\Delta t$ and $5\Delta t$, respectively. A single variable-timestep model is evaluated at the same timesteps for comparison.

Summary

A cross-comparison of Tables 5.13 and 5.14 reveals an important distinction between numerical precision and physical stability. While the variable-timestep model exhibits a "generalisation penalty" in terms of nRMSE for certain datasets, the Pearson correlation remains near-unity ($R \approx 1.0$) across all temporal scales. This suggests that the attention-based conditioning successfully preserves the structural topology of the PDE solutions.

Figure 5.21 establishes the scaling properties of our variable-timestep framework, demonstrating that although a generalisation penalty exists at low data volumes, it can be mitigated by increasing the training scale. Most importantly, the scaling law confirms that the flexibility of a single model does not come at the cost of structural stability; as long as sufficient data is provided, the variable-timestep model can recover the performance of a constant-timestep model while offering better parameter efficiency.

Finally, we visualise the solution of our four benchmark PDEs from Figure 5.22 - 5.25, and show the solution of the constant-timestep and variable-timestep models at $1\Delta t$. The solutions qualitatively look similar and thus are consistent with the correlation table. Consequently, the variable-timestep framework offers a robust and computationally efficient alternative to specialised solvers, particularly in non-linear systems such as Diffusion-Reaction, where multi-scale temporal training enhances predictive performance.

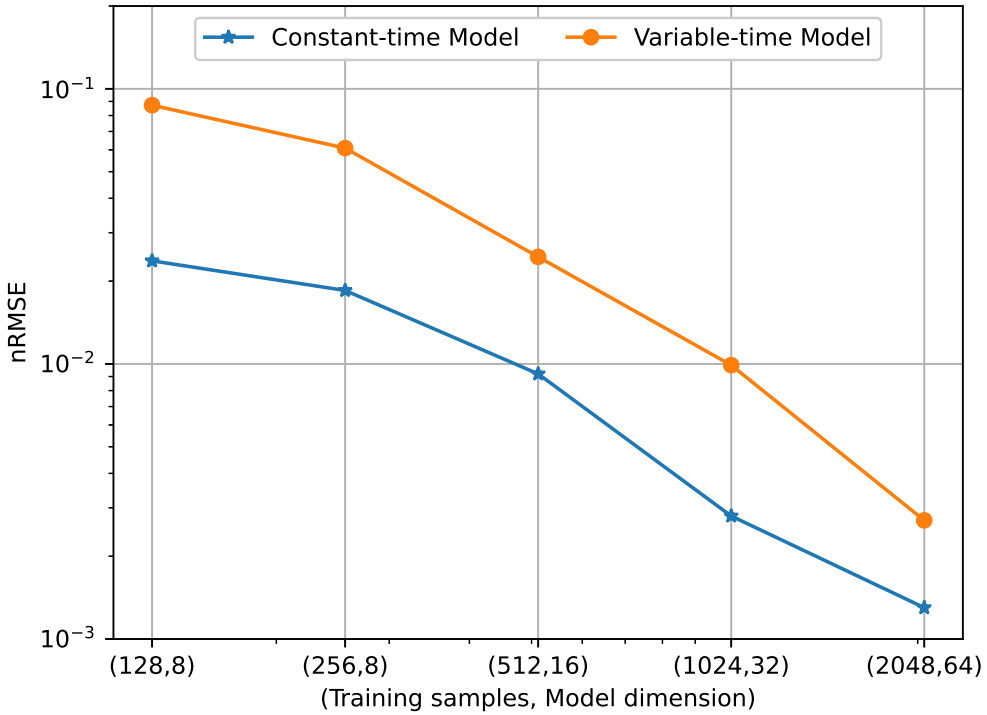


Figure 5.21: Scaling: The rate of error reduction with an increase in the number of training samples and model dimension.

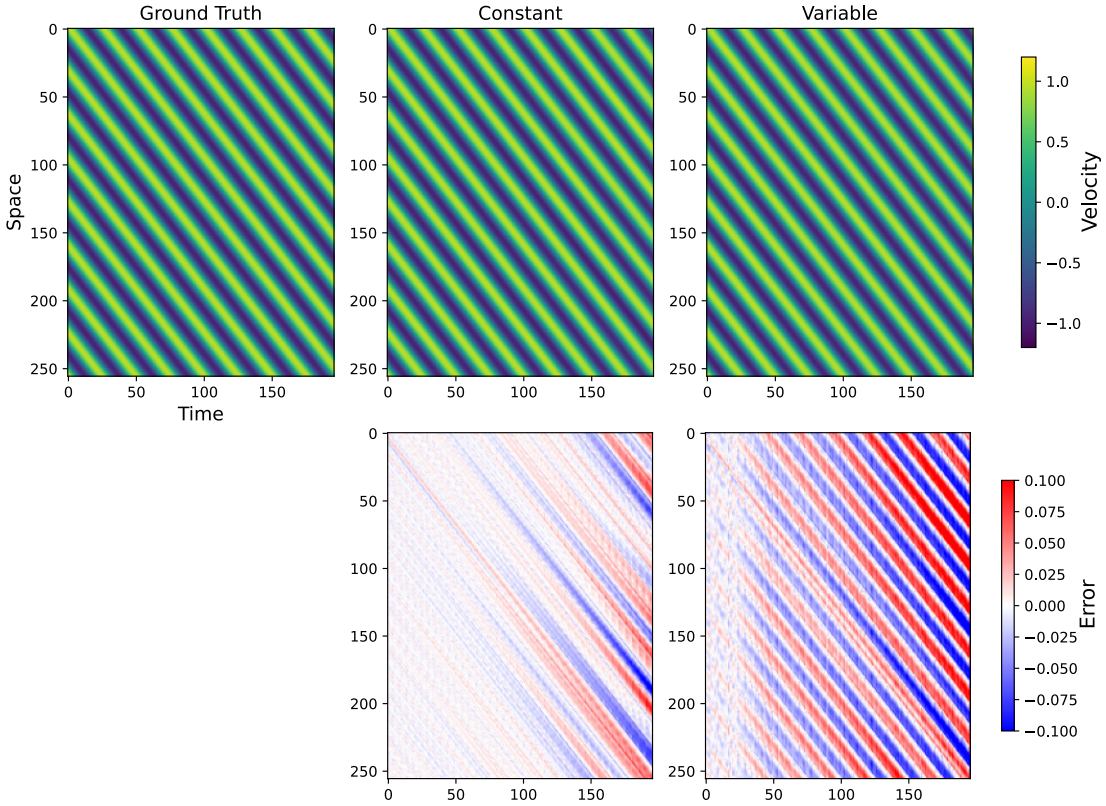


Figure 5.22: Advection: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable-timestep model evaluated at $1\Delta t$ (left). The data sample with the highest error is shown.

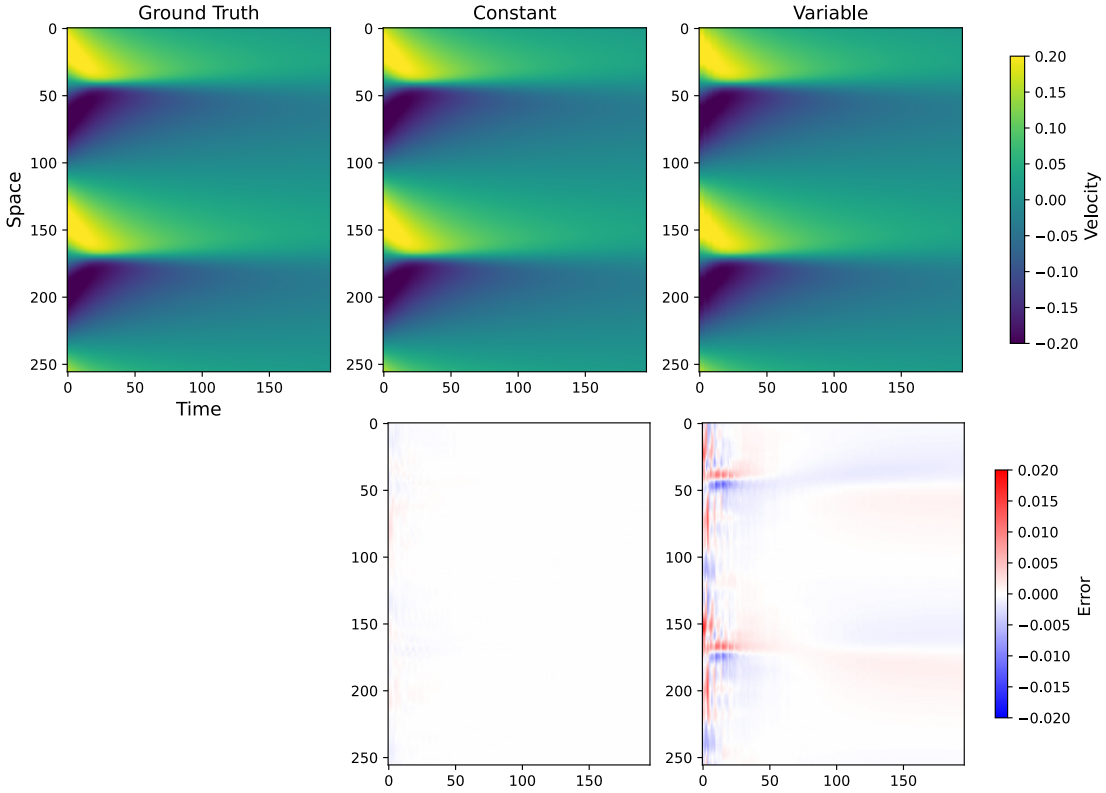


Figure 5.23: Viscous Burgers: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable model evaluated at $1\Delta t$ (left). Data sample with the smallest error.

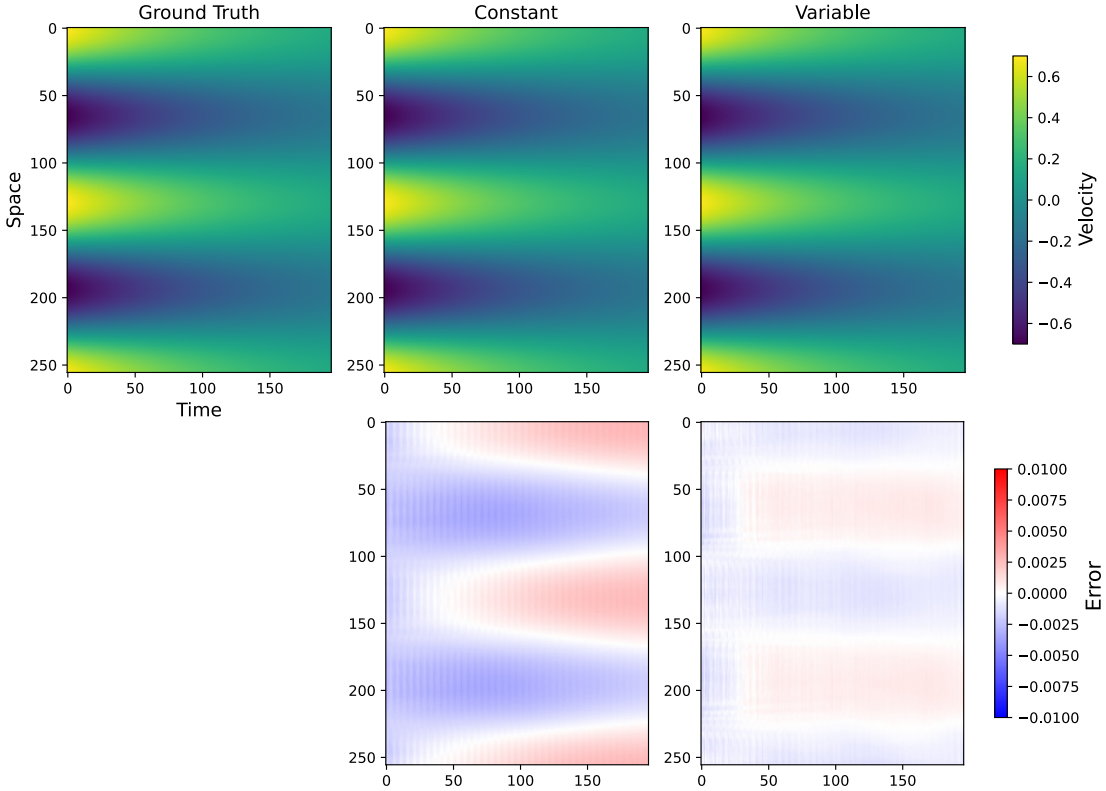


Figure 5.24: Diffusion-Reaction: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable model evaluated at $1\Delta t$ (left)

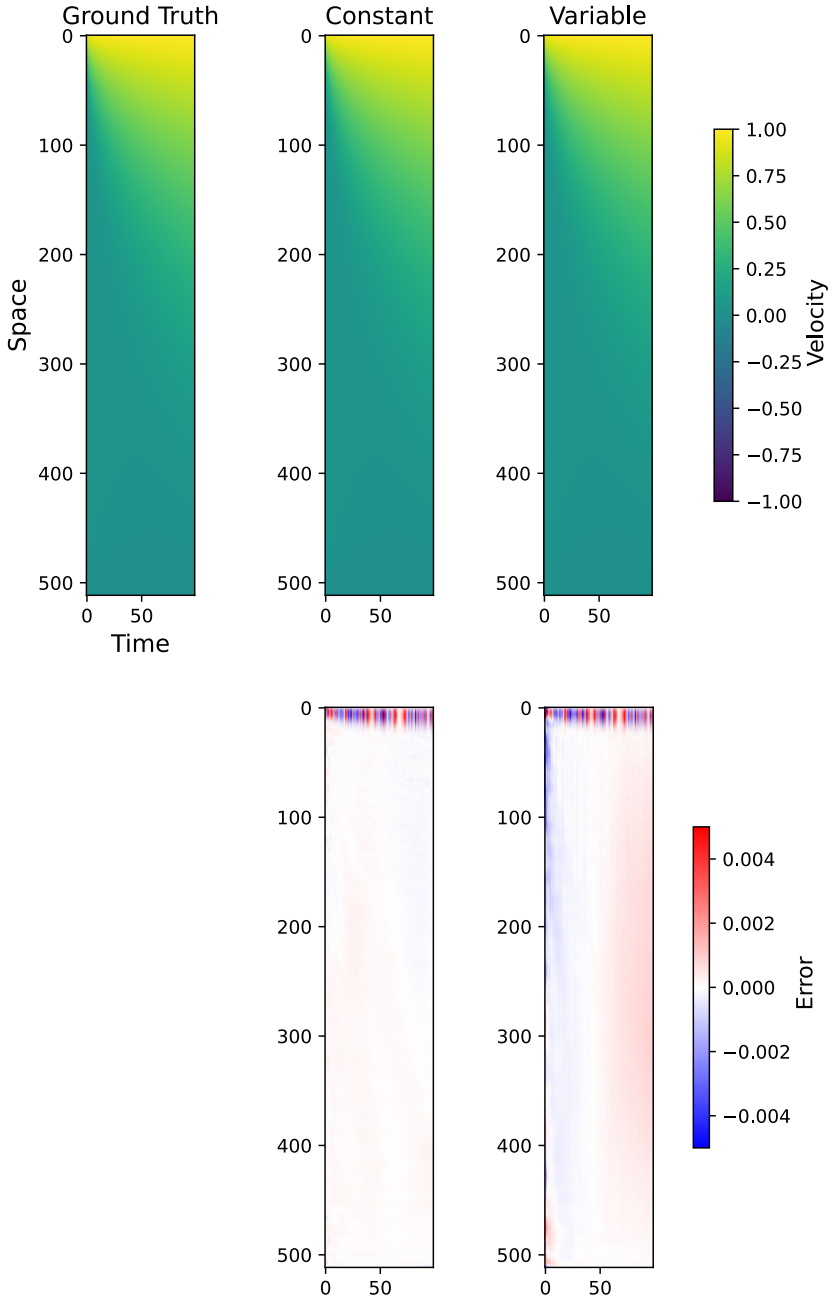


Figure 5.25: Diffusion-Sorption: Visualisation of prediction (top) and error (bottom) for $1\Delta t$ constant-timestep model (middle) and variable model evaluated at $1\Delta t$ (left)

5.8.9 Advection: Time-dependent speed parameter

This section provides an initial evaluation of the variable-timestep model's capacity to adapt to dynamic physical conditions by considering the simple advection problem (3.6.1), in which the speed $\beta(t)$ is time-varying. This requires our attention-based temporal conditioning mechanism to correlate the timestep Δt with a non-stationary solution field. To achieve this, we vary the speed parameter in the advection equation using three predefined functions: a step function, a linearly increasing function, and a linearly decreasing function. The quantitative performance across the three velocity functions is summarised in Table 5.15, comparing the total time error for the constant-timestep sampling ($1\Delta t$, $2\Delta t$ and $3\Delta t$) and the best performing variable-timestep sampling for each predefined function, where variable-timestep sampling achieves the smallest error on Linear-increasing and Linear-decreasing functions.

	C-parameter	**Constant-timestep Sampling**			**Variable-timestep Sampling**		
		1	2	3	2-1-2	3-2	2-3
		201 t_s	101 t_s	67 t_s	155 t_s	80 t_s	80 t_s
Av	**Step**	0.0090	**0.0067**	0.011	0.0070	-	-
	Linear-increasing	0.0069	0.0051	0.0052	-	**0.0047**	-
	Linear-decreasing	0.011	0.0073	0.0091	-	-	**0.0071**

Table 5.15: **Comparison of nRMSE for Advection with time-dependent speed $\beta(t)$.** The results evaluate constant-timestep models against the best-performing variable-sampling sequences for discontinuous (step), linear-increasing, and linear-decreasing velocity functions. **Bold** values indicate the sampling strategy that achieves the lowest error per row, highlighting the superior adaptability of the variable-timestep model with variable-timestep sampling in non-stationary environments.

Discontinuous (Step) Speed

Figure 5.26 shows the error profile exhibits sharp increases at the points of discontinuity ($t \approx 66$ and $t \approx 134$). Constant $2\Delta t$ sampling yields the lowest-trajectory error over the full evolution. An interesting trade-off is observed between $1\Delta t$ and $3\Delta t$ sampling: while the $3\Delta t$ model initially has lower error due to fewer autoregressive rollouts, its performance degrades sharply once it reaches the discon-

tinuity. Consequently, the coarse resolution of the $3\Delta t$ steps becomes insufficient to resolve the abrupt physical changes, leading to significantly higher errors in the later stages of the evolution compared to the $1\Delta t$ baseline.

While constant $2\Delta t$ sampling achieves the lowest error among fixed-step models, variable-timestep strategies can outperform it when accuracy is evaluated per total number of rollouts (t_s). Figure 5.27 provides a comparative analysis of efficiency by plotting error against the total number of rollouts (t_s). While constant $2\Delta t$ (at $100t_s$) represents a local optimum for fixed-step models, the variable-timestep sampling curve stays generally below the constant-timestep points. This indicates that by strategically distributing the temporal resolution, for instance, using finer steps specifically during the velocity discontinuities, the variable-timestep model can achieve a higher level of accuracy for the same computational cost (t_s) than a constant-timestep model.

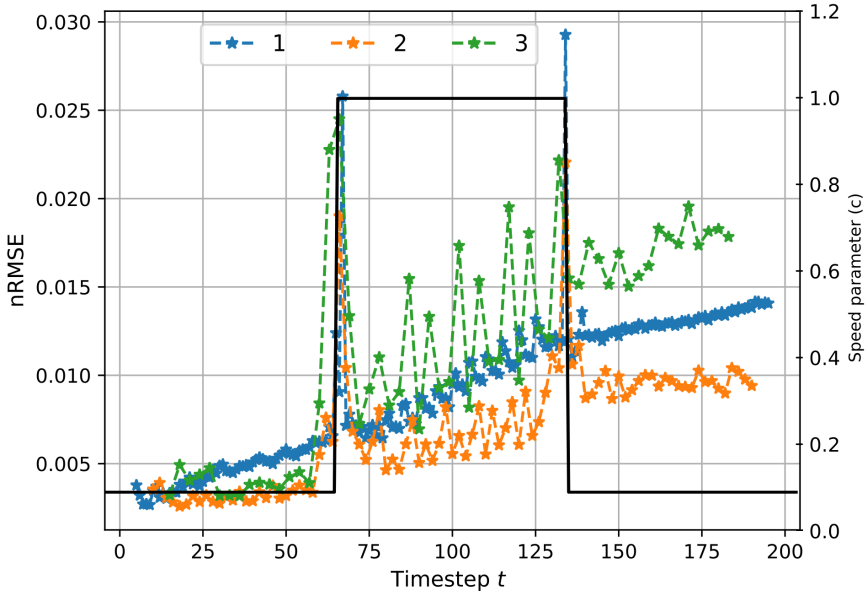


Figure 5.26: Temporal error propagation (nRMSE) for the advection equation with a **discontinuous step** speed function. The secondary axis (dashed black line) indicates the time-dependent speed $\beta(t)$, which shifts abruptly at $t \approx 66$ and $t \approx 134$. The plot illustrates how constant-timestep models ($1, 2$, and $3\Delta t$) experience sharp increases in error at these discontinuities.

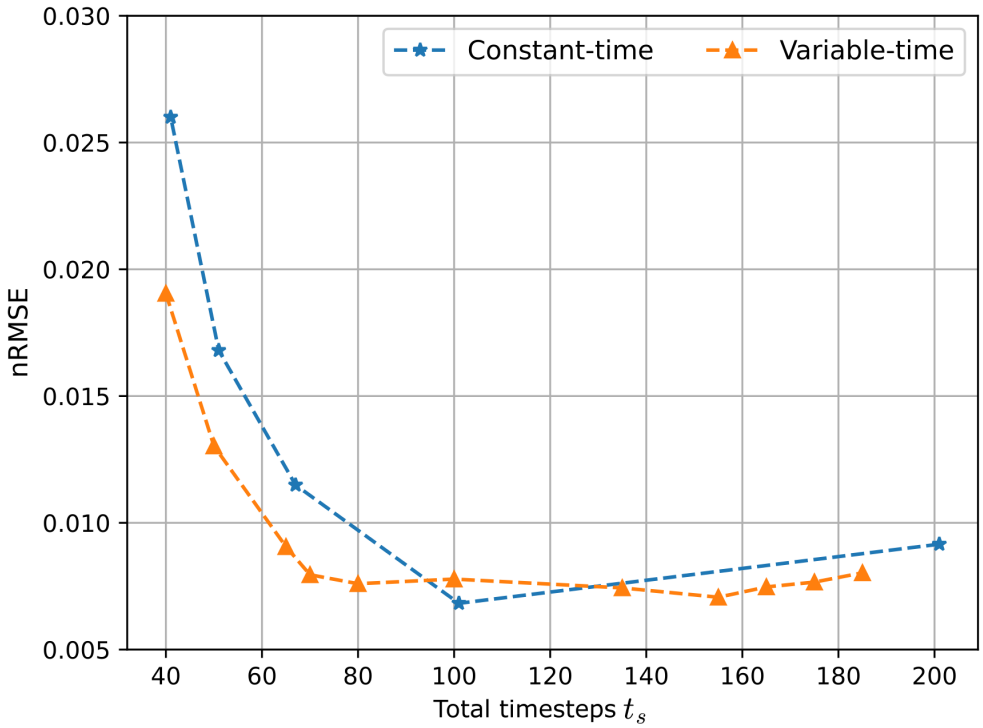


Figure 5.27: Total nRMSE relative to the total number of timesteps (t_s) for the **discontinuous (step)** speed function. The plot compares constant-timestep sampling ($\Delta t = 1, 2, 3, 4, 5$) against variable-timestep sampling strategies across a range of computational budgets. The variable-timestep curve demonstrates a superior efficiency-to-accuracy ratio, consistently achieving lower error than its constant-timestep counterparts across most rollout counts, particularly at both low and high sampling rates.

Linear-Increasing Speed

Figure 5.28 illustrates a steady error accumulation that becomes steeper as the wave accelerates. While $3\Delta t$ is more effective during the lower-velocity initial phase ($t < 125$), $2\Delta t$ provides better stability as the velocity reaches its maximum. In Figure 5.29, we show that the variable-timestep strategy ($3\Delta t \rightarrow 2\Delta t$) can capitalise on this by transitioning resolution to match the physical dynam-

ics. We observe the model's ability to perform "active error mitigation". In the linear-increasing case, a constant $3\Delta t$ sampling rate begins to suffer from increased error propagation as the wave speed increases. By switching to $2\Delta t$ at the midpoint, the variable-timestep model demonstrates a "refining" capability, increasing sampling frequency exactly when the physical dynamics evolves faster to ensure the long-term stability of the rollout, leading to a lower error than either $2\Delta t$ and $3\Delta t$ as seen in Table 5.15.

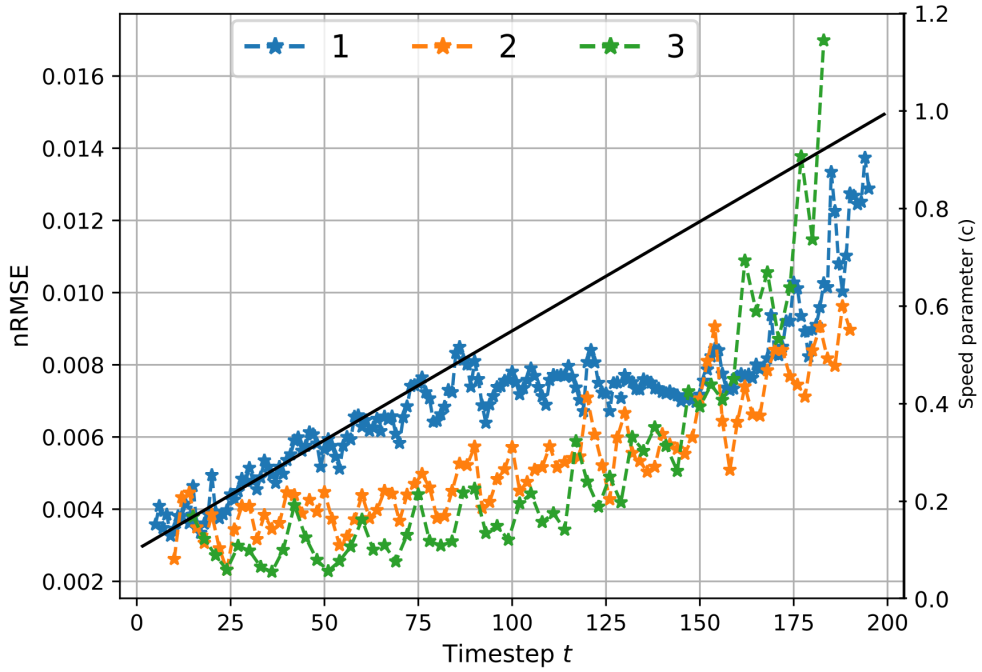


Figure 5.28: Error propagation (nRMSE) for the advection equation with a **linear-increasing** speed function. The secondary axis (dashed black line) indicates the accelerating speed parameter $\beta(t)$.

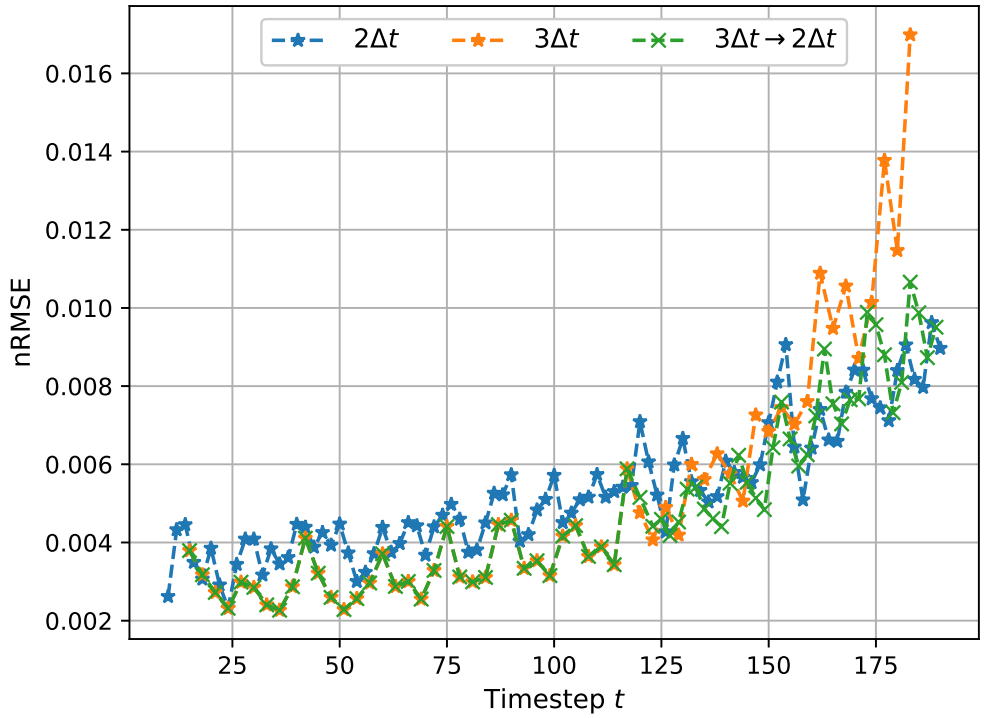


Figure 5.29: Error per timestep for the **linear-increasing** advection speed. The plot illustrates the effectiveness of a 3-2 variable-sampling strategy in stabilising error growth. As the wave velocity $\beta(t)$ increases, the transition from a coarser resolution ($3\Delta t$) to a finer resolution ($2\Delta t$) at $t = 125$ effectively caps the rising error profile, maintaining a lower and more consistent error per step than the constant $3\Delta t$ baseline.

Linear-Decreasing Speed

For the decelerating speed case, Figure 5.30 shows that $1\Delta t$ and $2\Delta t$ errors increase over time, whereas $3\Delta t$ error actually reduces alongside the decreasing wave speed. While $2\Delta t$ is optimal during the high-speed initial phase, its advantage diminishes as the speed reaches its minimum. In Figure 5.31 we show that by utilising a 2-3 variable-timestep strategy, the model captures the high-velocity dynamics early on and coarsens the resolution as the physics become less demand-

ing. This demonstrates the variable-timestep model's ability to "capitalise" on decelerating dynamics. Unlike constant-timestep models that can waste computational budget by maintaining a single unoptimised timestep, the variable-timestep framework allows for temporal coarsening, maintaining high Pearson correlation while reducing the number of rollouts.

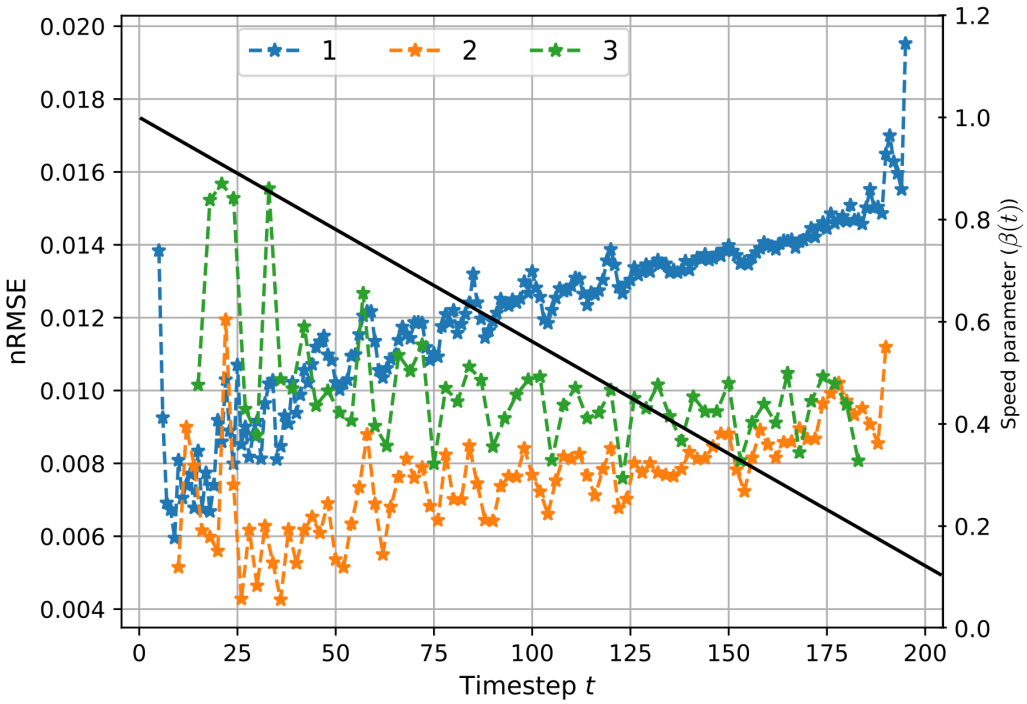


Figure 5.30: Error propagation (nRMSE) for the advection equation with a **linear-decreasing** speed function. The secondary axis (dashed black line) represents the decelerating speed parameter $\beta(t)$.

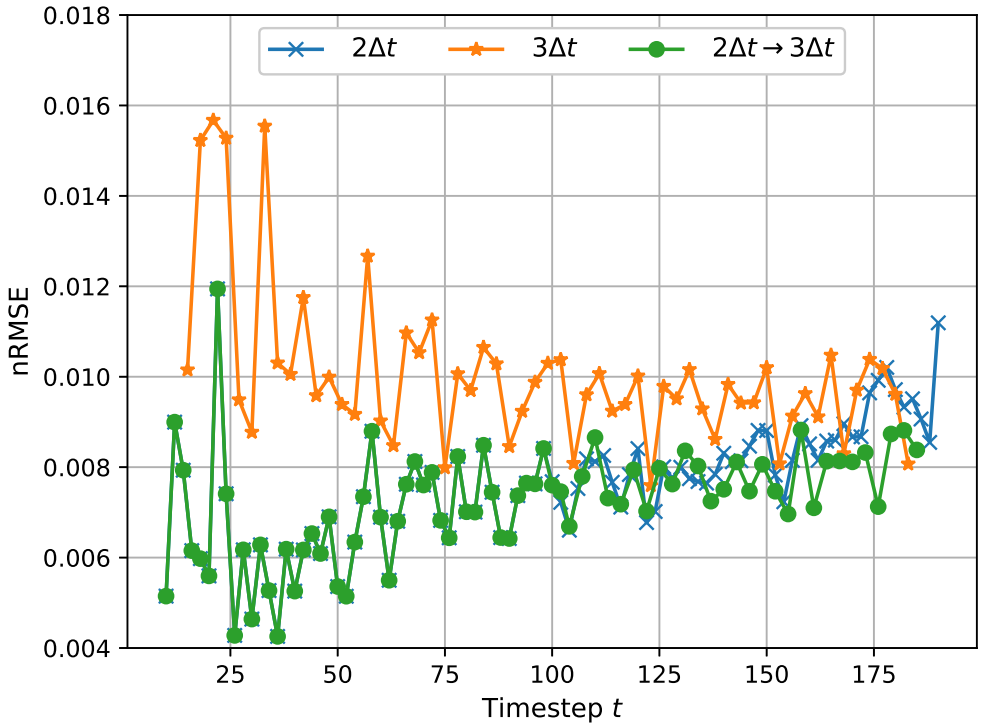


Figure 5.31: Error per timestep for the **linear-decreasing** advection speed. The plot illustrates the effectiveness of a 2-3 variable-sampling strategy. By transitioning from a finer resolution ($2\Delta t$) during the high-velocity initial phase to a coarser resolution ($3\Delta t$) at $t = 100$ as the wave decelerates, the model maintains structural fidelity while reducing the total number of required recursions.

5.8.10 Comparison with a Transformer-based Neural Operator

To compare our attention-conditioned FNO against the recent family of transformer-based neural operators, we additionally benchmark it against the Operator Transformer (OFormer) (Li *et al.*, 2022c) on the most challenging system considered in this chapter, the advection equation with a time-dependent speed parameter $\beta(t)$ (Section 5.8.9). This non-stationary setting is deliberately chosen as the harder problem: it requires the temporal-conditioning mechanism to correlate the requested interval Δt with a solution field whose dynamics themselves change in time, which stresses precisely the component that distinguishes a transformer-based operator (full spatial self-attention) from our lightweight temporal attention.

Table 5.16 reports the total-rollout nRMSE for both architectures on the three velocity profiles. For our model, each entry is the best-performing sampling strategy for that profile, taken from Table 5.15. Across all three profiles, OFormer attains the lower error: the margin is largest for the linear-increasing profile, where its nRMSE is roughly half of ours, and smallest for the linear-decreasing profile, where the two are close. This ordering is consistent with the architectural distinction drawn in Section 5.7 and the Related Work: OFormer applies full (Galerkin-type) self-attention over the spatial domain, which is well suited to resolving the fine spatial structure of the advected field, whereas our model applies attention only along the temporal axis and delegates spatial representation to the resolution-invariant spectral convolution.

Model	Time-dependent advection speed $\beta(t)$		
	Step	Linear-increasing	Linear-decreasing
Attention-FNO (ours)	0.0067	0.0047	0.0071
OFormer (Li *et al.*, 2022c)	**0.0045**	**0.0021**	**0.0063**

Table 5.16: Comparison of the proposed attention-conditioned FNO against the Operator Transformer (OFormer) (Li *et al.*, 2022c) on the harder, non-stationary advection problem with a time-dependent speed parameter $\beta(t)$. Values are total-rollout nRMSE (lower is better); for the attention-conditioned FNO each entry is the best-performing sampling strategy for that profile, drawn from Table 5.15. The lowest error in each column is shown in **bold**. OFormer attains the lower error on all three profiles, but at higher computational cost (see text).

This accuracy advantage comes at a computational cost. OFormer is appreciably slower to run than our attention-conditioned FNO, as its full self-attention backbone attends over all spatial tokens, whereas our mechanism confines attention to the temporal axis and leaves spatial structure to the spectral convolution. Our model therefore occupies the expected position in the accuracy and efficiency trade-off: it concedes a modest amount of accuracy, all four figures remain below 1% nRMSE, in return for a lighter attention mechanism and faster inference. This mirrors the parameter-efficiency argument developed in Section 5.8.8, where the flexibility of a single variable-timestep model was shown to carry a bounded accuracy penalty relative to specialised baselines.

A further distinction concerns temporal flexibility. In its original formulation, OFormer advances time-dependent systems through a latent time-marching scheme with a fixed step size, so variable-timestep prediction is not a native capability of the architecture; the comparison here therefore evaluates it in precisely the regime our method is designed for. That our lightweight temporal conditioning remains within the same order of magnitude as a full transformer-based operator—at lower computational cost and with native support for variable-timestep rollout—is the intended contribution, rather than an unqualified reduction in error. The result also delineates where each design is preferable: where fine spatial fidelity is paramount and compute is abundant, full spatial self-attention is advantageous; where efficiency and temporal adaptivity are the priority, temporal conditioning of a spectral backbone is the more economical choice.

5.8.11 Variable-timestep Model with Curriculum Learning

Tables 5.17 and 5.18 provide compelling evidence of the advantage of combining the Scheduled Temporal Loss Weighting (STLW) curriculum learning introduced in Chapter 4 with the variable-timestep attention-based model developed in this chapter. By comparing the performance across three PDEs, Advection (A1), Viscous Burgers (B1), and Diffusion-Sorption (DS1), we conclude that the proposed curriculum (STLW) acts as a foundational stabiliser, improving the variable-timestep model across all sampling strategies.

These results bridge the two contributions by showing that the architectural innovation of this chapter and the training strategy of Chapter 4 are not mutually exclusive but complementary. While attention-based conditioning enables variable-timestep prediction, the STLW curriculum provides the stability required to realise that capacity during long-term inference. The "Variable-timestep + STLW" configuration represents the most robust framework presented in this thesis for solving time-dependent PDEs efficiently.

		Variable-timestep Model						
		Constant-timestep Sampling			**Variable-timestep Sampling**			
		1	2	3	1-2	2-1	1-2-3	3-2-1
		201 t_s	101 t_s	67 t_s	150 t_s	150 t_s	125 t_s	125 t_s
A1	w/o STLW	1.20	1.14	1.93	1.07	1.08	1.03	1.29
	w STLW	**1.07**	**0.83**	**1.13**	**0.99**	**0.83**	**0.95**	**1.05**
B1	w/o STLW	1.8	0.93	0.84	1.7	0.96	1.6	0.84
	w STLW	**0.78**	**0.47**	**0.43**	**0.75**	**0.50**	**0.71**	**0.45**
DS1	w/o STLW	0.14	0.12	0.11	0.13	0.12	0.13	0.12
	w STLW	**0.12**	**0.11**	**0.10**	**0.11**	**0.12**	**0.11**	**0.11**

Table 5.17: Total Error: Effect of Scheduled Temporal Loss weighting (STLW) on variable-timestep models. For each PDE, we present the error with (w) and without (w/o) the STLW curriculum. STLW curriculum results (highlighted in **bold**) have the lowest errors.

		Variable-timestep Model						
		Constant-timestep Sampling			Variable-timestep Sampling			
		1 201 t_s	2 101 t_s	3 67 t_s	1-2 150 t_s	2-1 150 t_s	1-2-3 125 t_s	3-2-1 125 t_s
A1	w/o STLW	0.011	0.010	0.019	0.008	0.009	0.009	0.009
	w STLW	**0.009**	**0.0064**	**0.0086**	**0.0070**	**0.0069**	**0.0075**	**0.0071**
B1	w/o STLW	0.01	0.0037	**0.0024**	0.0085	0.0042	0.0071	0.0031
	w STLW	**0.0044**	**0.0033**	0.0025	**0.0035**	**0.0033**	**0.0035**	0.0031
DS1	w/o STLW	0.0013	0.0011	0.0008	0.0011	0.0011	0.0009	**0.0010**
	w STLW	**0.0014**	**0.0011**	**0.0009**	**0.0010**	**0.0014**	**0.0008**	0.0013

Table 5.18: Final time Error: Effect of Scheduled Temporal Loss weighting (STLW) on variable-timestep models. For each PDE, we present the error with (w) and without (w/o) the STLW curriculum. STLW curriculum results (highlighted in **bold**) have the lowest errors.

5.9 Discussion

The primary objective of this chapter was to evaluate the efficacy of variable-timestep sampling and prediction using neural operators, specifically addressing the limitations of constant-timestep models. By introducing an attention-based temporal conditioning mechanism, we aimed to give the model the flexibility to capture long-term dependencies without the rigid constraints imposed by a constant-timestep model that only predicts at a fixed timestep, Δt.

The experimental results across four distinct physical systems - Advection, Viscous Burgers, Diffusion-Reaction, and Diffusion-Sorption - provide a comprehensive evaluation of the proposed attention-based variable-timestep model. This discussion interprets the "variable-timestep advantage" through the lens of error propagation, physical complexity, and computational efficiency.

5.9.1 Efficacy of Temporal Conditioning Mechanisms

The results consistently indicate that **Attention-based conditioning (At)** outperforms concatenation (Co) across all tested PDEs. This superiority stems from the mechanism's ability to dynamically modulate the hidden state's feature maps based on the specific temporal query (Δt), rather than merely appending a static scalar.

However, a critical trade-off exists: the **Concatenation (Co)** approach is approximately **30% faster** and uses **8% less memory** than the attention mechanism (Table 5.12). While Attention provides higher precision, the memory and computational overhead may limit performance for extremely high-dimensional 3D problems, where GPU memory is expensive.

Regarding embeddings, the **Embedding Layer (EL)** proved the most robust across systems. The exception being **Viscous Burgers'**, where **MLP** performed best, suggests that for systems with high-frequency shocks and steep gradients, a continuous, non-linear mapping (MLP) may be better suited to capture the rapid transition between different temporal scales than the discrete nature of a simple embedding layer.

5.9.2 The Generalisation Penalty vs. Exposure Bias

An important finding in Section 5.8.8 is the "Generalisation Penalty": specialised constant-timestep models often achieve lower errors at their specific training timestep than the single, flexible variable-timestep model. However, this penalty is arguably a necessary cost for **parameter efficiency**, as the variable-timestep

model is **5x more efficient** than maintaining an ensemble of specialised constant-timestep models, i.e., **five** constant-timestep models-to-**one** variable-timestep model.

Furthermore, the variable-timestep model exhibits superior resilience to exposure bias in long-term rollouts. In systems like the Viscous Burgers' equation, the $1\Delta t$ constant-timestep model - despite having the finest resolution - eventually produces higher errors than the $3\Delta t$ model because the frequency of rollout steps (three times as many) leads to larger error propagation. By allowing for larger temporal steps, the variable-timestep model maintains a more stable rollout.

5.9.3 Variable-timestep model: Error Dynamics

Our results indicate that the behaviour of the error over time is a strong predictor of the optimal variable-timestep sampling strategy. As summarised in the Table 5.19, there is a positive correlation between the error behaviour over time and the "direction" of the most effective sampling strategy. We discuss the error behaviour for each PDE below:

- **Advection**: In the case of pure advection, where the solution is linearly translated in space, the error increases linearly over time as a result of pure propagation through successive rollouts. This leads to increasing errors over time.

- **Viscous Burgers'**: In this dissipative system, the spatial gradients of the solution reduce over time, the initial shock dissipates, leading to a low-gradient, low-magnitude solution. The nature of the solution dominates over the effect of error propagation, leading to a decreasing error over time.

- **Diffusion-Reaction**: In the diffusion-reaction case, the non-linear source term plays an important role as the solution converges towards a steady state. This leads to an increasing error curve over time.

- **Diffusion-Sorption**: The diffusion-sorption solution starts with a discontinuous initial condition, which diffuses over time. The challenge of accurately capturing discontinuity early in the evolution leads to higher error than later.

DATASET	Error Behaviour	Best variable-timestep sampling Strategy
Advection (Av)	Increasing	Increasing
Viscous Burgers' (vB)	Decreasing	Decreasing
Diff-Reaction (DR)	Increasing	Increasing
Diff-Sorpt (DS)	Decreasing	Decreasing

Table 5.19: Correlation between behaviour of error over time and the best variable-sampling strategy.

Ultimately, these observations suggest that the model's predictive accuracy is intrinsically linked to the error behaviour from which it is sampled. Sampling from a temporal region characterised by high local errors will invariably yield higher cumulative errors in the rollout, and vice versa.

Strategy Directionality: Temporal Coarsening vs. Refinement

We can group the variable-timestep strategies into temporal coarsening (decreasing) and refinement (increasing). The optimal variable-sampling strategy is linked to the physical evolution of the system:

- **Temporal Coarsening (small-to-large)**: For Advection and Diffusion-Reaction, a "Small-to-Large" strategy (e.g., $1\Delta t \rightarrow 2\Delta t \rightarrow 3\Delta t$) is optimal. In these systems, establishing a precise initial trajectory is vital. As errors naturally accumulate, transitioning to larger steps ($3\Delta t$) reduces the number of rollouts, effectively "capping" the cumulative error growth that would otherwise accelerate under a fixed $1\Delta t$ regime.

- **Temporal Refinement (large-to-small)**: Conversely, Viscous Burgers' and Diffusion-Sorption benefit from a "Large-to-Small" strategy. These systems are most dynamic at $t = 0$ due to high spatial gradients or initial shocks. For long-term accuracy, using a large $3\Delta t$ "leap" at the start allows the model to bypass the most unstable region of the evolution, thereby avoiding the exponential amplification of early-stage errors. Once the solution has dissipated to a smoother state, the model can safely refine its resolution in smaller steps to maintain local fidelity.

5.9.4 Analysis of Non-Stationary Systems

The evaluation of the time-dependent speed parameter in Advection (Section 5.8.9) provides a rudimentary proof-of-concept for **adaptive solvers**. The variable-timestep model proved superior in adapting to **linear-increasing (3-2 strategy)** and **linear-decreasing (2-3 strategy)** speeds. By transitioning to a finer resolution ($2\Delta t$) as wave velocity increases, the model caps the rising error profile more effectively than a rigid, fixed-step model.

However, a critical limitation was observed with the **discontinuous "Step" function**. Constant-timestep models exhibited sharp increases in error at the point of discontinuity. This highlights a potential "resolution risk"; while large temporal jumps are efficient for smooth evolution, they may lack the resolution to capture abrupt physical transitions, potentially leading to significant inaccuracies if the timestep "overshoots" a critical event.

5.9.5 Structural Stability and Scaling Laws

Despite the generalisation penalty in accuracy (nRMSE), the **Pearson correlation remains near-unity ($\mathbf{R \approx 1.0}$)** across all scales, implying that the attention-based model successfully preserves the **structural topology** and qualitative physics of the PDE solutions, even when accuracy is lower than that of the constant-timestep model with high accuracy.

The Scaling Law (Figure 5.21) further confirms that this generalisation penalty can be mitigated. As training data size and model dimensions increase, the variable-timestep model's performance converges toward the "gold standard" of specialised constant-timestep models. This suggests that the framework is limited more by data availability than by architectural constraints.

5.9.6 Summary of Stability and Propagation

In conclusion, while the variable-timestep models introduce a slight generalisation penalty in terms of accuracy, it provides a robust mechanism for long-term stability. By aligning the sampling strategy with the error dynamics of the PDE, either through temporal coarsening for PDEs with growing complexity or Temporal Refinement for PDEs with early-stage volatility, the model achieves a balance of global stability and parameter efficiency that a constant-timestep model cannot replicate. This work has not made any effort to develop a systematic strategy for selecting and changing the variable dt - rather, it sought to demonstrate the feasibility and potential benefits of being able to use a variable-timestep.

Chapter 6

Conclusion and Future Work

In Chapter 4, we addressed the fundamental challenges of training constant-timestep models for long-term autoregressive prediction of time-dependent partial differential equations (PDEs). Standard training approaches often struggle with gradient accumulation and the compounding of errors, known as exposure bias, during long inference rollouts. To mitigate these issues, we proposed **Scheduled Temporal Loss Weighting (AR-STLW-CL)**, a novel continuous curriculum learning framework that bridges the gap between simple one-step training and complex multi-step autoregressive rollouts.

In systems with dynamics that fluctuate between fast and slow regimes, a fixed timestep often leads to a trade-off between computational inefficiency and numerical instability. To overcome this, Chapter 5 shifts the focus toward temporal flexibility by exploring variable-timestep prediction. Specifically, we demonstrate the efficacy of an **Attention-based temporal conditioning mechanism** for enabling variable-timestep prediction in Fourier Neural Operators (FNOs). By replacing conventional concatenation-based conditioning with a parallel attention module, the model can dynamically weight temporal features, leading to more robust, stable autoregressive rollouts across different physical systems.

6.1 Summary of main results

6.1.1 Curriculum Learning for Constant-Time model

The experimental evaluation across four diverse PDE datasets (Advection, Viscous Burgers, Inviscid Burgers, and Korteweg-De Vries) demonstrated the follow-

ing:

- **Superiority Over Baselines:** AR-STLW-CL consistently outperformed standard baselines, including Teacher Forcing (TF) and fixed Autoregressive (AR) training. It achieved substantial relative improvements in accuracy, notably reducing the normalised Root Mean Squared Error (nRMSE) by 62.5% for the Viscous Burgers equation and 40.3% for the KdV equation compared with the best-performing baseline.

- **Stability in Long-Term Rollouts:** Our approach proved to be the most effective at maintaining stability during extended inference rollouts, where other methods typically suffer from exponential error growth.

- **Enhanced Scaling with Training Rollouts:** While standard AR training showed performance degradation when the training rollout length exceeded 8, AR-STLW-CL effectively managed larger rollouts, either limiting error growth or continuing to improve accuracy as training rollout length increased.

- **Robustness to Parameterisation:** The sensitivity analysis of the curriculum parameter A_c revealed that the framework is highly robust. Whether the transition from early-step focus to full rollout was fast or slow, AR-STLW-CL consistently maintained lower error rates than standard AR and discrete curriculum baselines.

6.1.2 Attention Mechanism for Variable-timestep model

The investigation into temporal conditioning strategies consistently identified the **Attention-based mechanism (At)**, particularly when paired with an **Embedding Layer (EL)**, as the most robust architecture across diverse physical systems. While the attention mechanism incurs a slight computational overhead, approximately 30% slower than simple concatenation, it provides the necessary expressive capacity to modulate spectral features according to the requested timestep Δt.

A pivotal finding of this work is the potential **correlation between temporal error dynamics and optimal sampling strategies**:

- **Temporal Coarsening (small-to-large):** In systems like Advection and Diffusion-Reaction, where error accumulates over time, starting with fine temporal resolution and transitioning to coarser steps (e.g., $1\Delta t \rightarrow 2\Delta t \rightarrow 3\Delta t$) effectively caps error propagation by reducing the total number of rollouts.

- **Temporal Refinement (large-to-small):** Conversely, for systems with early-stage volatility, such as Viscous Burgers' and Diffusion-Sorption, a "large-to-small" strategy is superior. By using a large temporal "leap" to bypass initial high-gradient shocks, the model avoids the early amplification of exposure bias.

Furthermore, the results demonstrate that while variable-timestep models may face a "Generalisation Penalty" in accuracy (nRMSE) compared to constant-timestep models, they maintain high **structural fidelity** ($R \approx 1.0$) and superior global stability. This trade-off is particularly advantageous in non-stationary systems, where the variable-timestep model's ability to adjust its resolution to match time-dependent speeds ($\beta(t)$) results in a significantly better efficiency-to-accuracy ratio than any fixed-step baseline.

6.2 Variable-timestep Model with Curriculum Learning

The two contributions of this thesis are complementary rather than independent. The combined evaluation, in which the Scheduled Temporal Loss Weighting (STLW) curriculum of Chapter 4 is applied to the variable-timestep attention-based model of Chapter 5, is reported in full in Section 5.8.11 (Tables 5.17 and 5.18). Those results show that the STLW curriculum acts as a foundational stabiliser for the variable-timestep model across Advection, Viscous Burgers, and Diffusion-Sorption, and across all sampling strategies: architectural innovation enables variable-timestep prediction, while the curriculum provides the stability required to realise that capacity during long-term inference. The "Variable-timestep + STLW" configuration is therefore the most robust framework presented in this thesis for solving time-dependent PDEs efficiently.

6.3 Thesis Contributions

The primary contribution of this research is the development of a unified framework that addresses the dual challenges of temporal instability and architectural rigidity in neural operators for PDEs. By synthesising novel training protocols with flexible architectures, this work provides a robust alternative to traditional numerical solvers. The specific contributions are detailed below.

6.3.1 A Novel Training Strategy: Scheduled Temporal Loss Weighting (STLW)

A major impact of this work, as detailed in Chapter 4, is the development of Scheduled Temporal Loss Weighting (STLW). This curriculum learning framework is designed to bridge the gap between the simplicity of *teacher-forcing* and the complexity of full multi-step autoregressive rollouts.

- **Dynamic Loss Weighting:** STLW introduces a continuous curriculum that assigns lower weights to later timesteps in an autoregressive rollout during early training stages. These weights are dynamically adjusted through a smooth, continuous schedule, allowing the model to stabilise the learning of complex temporal dependencies before tackling long-term horizons.
- **Mitigation of Overfitting:** The strategy avoids the suboptimal nature of standard teacher-forcing, which often overfits to ground-truth data, by providing a *soft* transition toward full autoregressive autonomy.
- **Empirical Gains:** The method is model-agnostic and demonstrated significant performance improvements across Fourier Neural Operators (FNO) and U-shaped Neural Operators (U-NO). Experimental results showed a 62.5% reduction in nRMSE for the viscous Burgers' equation and 40.3% for the KdV equation.

6.3.2 Novel Model Architecture: Attention-Based Temporal Conditioning

In Chapter 5, this thesis introduces an architecture that enables variable-timestep stepping in FNOs, addressing the inherent limitation of traditional models, which are restricted to fixed temporal scales.

- **Novel Conditioning Mechanism:** We replace standard concatenation-based conditioning with a parallel attention-based temporal conditioning mechanism. This allows the model to dynamically modulate temporal features based on the queried timestep Δt, ensuring the operator remains valid across a continuous temporal domain.
- **Temporal Embedding Evaluation:** The research provides a systematic evaluation of temporal embedding strategies, including Sinusoidal Temporal Embedding (STE), Embedding Layers (EL), and MLP-based embeddings, and identifies the optimal configurations for spectral transformation.

- **Global Stability and Efficiency:** While these models may incur a slight *generalisation penalty* in local accuracy compared to specialised constant-step models, they offer a superior efficiency-to-accuracy ratio. By reducing the total number of rollout operations, they effectively mitigate the accumulation of errors (exposure bias) inherent in long-term forecasting.

6.3.3 Synthesis and Performance Correlation

A key outcome of this research is the identification of the complementarity between STLW and variable-timestep conditioning. Combining these approaches creates a robust framework where the curriculum provides the stability necessary to realise the potential of flexible temporal prediction. Furthermore, this work establishes a correlation between error curve dynamics and optimal sampling strategies for our benchmarks:

- **Temporal Coarsening (Small-to-Large):** Found to be effective for dissipative systems (e.g., Advection, Diffusion-Reaction) to minimise error propagation over extended durations.
- **Temporal Refinement (Large-to-Small):** Proved superior for systems with early-stage volatility, such as viscous Burgers', where large initial timesteps allow the model to bypass high-gradient shocks and avoid early exposure bias.

6.3.4 Experimental Validation and Open-Source Contribution

Beyond methodological innovations, this thesis provides a rigorous benchmarking suite across dissipative and dispersive PDEs, including Advection, Burgers', Diffusion-Reaction, Diffusion-Sorption, and KdV equations. The research specifically demonstrates that variable-timestep models outperform fixed-step baselines in non-stationary systems where PDE parameters vary over time. Finally, the code developed for this work has been expanded into a flexible, open-source framework to facilitate future hyperparameter tuning and variable-timestep experimentation in the wider research community.

6.4 Future Work

6.4.1 Curriculum Learning for Constant-timestep Model

Building upon these findings, several directions emerge for extending the Scheduled Temporal Loss Weighting framework. The most immediate concerns the curriculum parameterisation itself: although the sensitivity study demonstrated that the framework is robust to the choice of A_c, the transition schedule is currently fixed in advance, and a natural next step would be to make it *adaptive*, tying the rate at which the temporal weights relax towards uniformity to quantities observed during training, such as the real-time validation loss or the magnitude of the accumulated gradients. Such a closed-loop curriculum would remove the remaining manual tuning and allow the schedule to respond to the difficulty of each particular system. A second direction concerns the class of dynamics studied. The present work deliberately focused on non-chaotic systems, for which fully autoregressive training is generally superior; chaotic dynamics, where teacher-forcing can instead hold a comparative advantage, would provide a more stringent test of the curriculum, and understanding how STLW behaves in that regime is a valuable open question. Beyond this, scalability remains to be established: extending the continuous curriculum to two- and three-dimensional fluid-dynamics problems would test its effectiveness for higher-dimensional spatial operators, where memory pressure is considerably greater. This last point is closely tied to computational efficiency, since the benefits of longer training rollouts must ultimately be weighed against their increased memory footprint and runtime, and a more careful characterisation of this accuracy-versus-cost trade-off would help practitioners choose an appropriate rollout length for a given budget.

6.4.2 Attention Mechanism for variable-timestep models

The success of attention-based temporal conditioning lays the foundation for more flexible, parameter-efficient neural operators. By treating time as a learnable query, neural operators can behave more like classical adaptive integrators, albeit with the speed of a deep learning inference. Several avenues follow naturally from this view. One is to revisit the temporal embedding itself: the present study evaluated learnt embeddings and fixed sinusoidal encodings but reserved non-learnt *continuous* encodings for future investigation, and deterministic continuous embeddings of this kind may improve generalisation beyond the temporal range seen during training. A more ambitious direction is to close the loop on the timestep selection. At present the sampling strategy is prescribed, whereas a genuinely adaptive operator would select its own Δt at inference, choosing finer

steps where local error estimates or physical gradients are large and coarser steps where the dynamics are smooth, in direct analogy with classical adaptive numerical solvers. Finally, as with the curriculum work, the attention-based architecture should be extended to two- and three-dimensional systems, where multiscale temporal phenomena are more pronounced and where the ability to allocate temporal resolution where it is most needed is likely to yield the greatest efficiency gains.

References

Alon, U. & Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*. 24

Ames, W.F. (2014). *Numerical methods for partial differential equations*. Academic press. 1, 11

Aoki, M. (2013). *State space modeling of time series*. Springer Science & Business Media. 24

Aragon-Calvo, M.A. (2019). Classifying the large-scale structure of the universe with deep neural networks. *Monthly Notices of the Royal Astronomical Society*, **484**, 5771–5784. 12

Bengio, Y., Louradour, J., Collobert, R. & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48. 50, 59, 95

Berezin, F.A. & Shubin, M. (2012). *The Schrödinger Equation*, vol. 66. Springer Science & Business Media. 9

Bihlo, A. & Popovych, R.O. (2022). Physics-informed neural networks for the shallow-water equations on the sphere. *Journal of Computational Physics*, **456**, 111024. 13

Bishop, C.M. (1994). Neural networks and their applications. *Review of scientific instruments*, **65**, 1803–1832. 15

Bishop, C.M. (2013). Model-based machine learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **371**, 20120222. 15

Bonito, A., Cohen, A., DeVore, R., Guignard, D., Jantsch, P. & Petrova, G. (2021). Nonlinear methods for model reduction. *ESAIM: Mathematical Modelling and Numerical Analysis*, **55**, 507–531. 20

Box, G.E. & Jenkins, G.M. (1968). Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **17**, 91–109. 24

Brandstetter, J., Berg, R.v.d., Welling, M. & Gupta, J.K. (2022a). Clifford neural layers for pde modeling. *arXiv preprint arXiv:2209.04934*. 23

Brandstetter, J., Welling, M. & Worrall, D.E. (2022b). Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, 2241–2256, PMLR. 15, 76

Brandstetter, J., Worrall, D. & Welling, M. (2022c). Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*. 3, 13, 36, 49, 51, 76

Bruno, O.P., Han, Y. & Pohlman, M.M. (2007). Accurate, high-order representation of complex three-dimensional surfaces via fourier continuation analysis. *Journal of computational Physics*, **227**, 1094–1125. 23

Busseti, E., Osband, I. & Wong, S. (2012). Deep learning for time series modeling. *Technical report, Stanford University*, 1–5. 22

Cao, Q., Goswami, S. & Karniadakis, G.E. (2023). Lno: Laplace neural operator for solving differential equations. *arXiv preprint arXiv:2303.10528*. 23

Cao, S. (2021). Choose a transformer: Fourier or galerkin. *Advances in Neural Information Processing Systems*, **34**, 24924–24940. 3, 13

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J. & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, 3438–3445. 24

Chen, R.T., Rubanova, Y., Bettencourt, J. & Duvenaud, D.K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, **31**. 13, 100

Chen, T. & Chen, H. (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, **6**, 911–917. 17, 18

Cohen, A. & DeVore, R. (2015). Approximation of high-dimensional parametric pdes. *Acta Numerica*, **24**, 1–159. 20

Cooley, J.W. & Tukey, J.W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, **19**, 297–301. 183

Dang, Y., Hu, Z., Cranmer, M., Eickenberg, M. & Ho, S. (2022). Tnt: Vision transformer for turbulence simulations. *arXiv preprint arXiv:2207.04616*. 102, 104

del Águila Ferrandis, J., Triantafyllou, M.S., Chryssostomidis, C. & Karniadakis, G.E. (2021). Learning functionals via lstm neural networks for predicting vessel dynamics in extreme sea states. *Proceedings of the Royal Society A*, **477**, 20190897. 13

DeVore, R.A., Howard, R. & Micchelli, C. (1989). Optimal nonlinear approximation. *Manuscripta mathematica*, **63**, 469–478. 20

Dey, R. & Salem, F.M. (2017). Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, 1597–1600, IEEE. 25

Di Leoni, P.C., Lu, L., Meneveau, C., Karniadakis, G. & Zaki, T.A. (2021). Deeponet prediction of linear instability waves in high-speed boundary layers. *arXiv preprint arXiv:2105.08697*. 18, 102

Diab, W. & Al Kobaisi, M. (2025). Temporal neural operator for modeling time-dependent physical phenomena. *Scientific Reports*, **15**, 32791. 102

Doering, C.R. (2009). The 3d navier-stokes problem. *Annual Review of Fluid Mechanics*, **41**, 109–128. 9

Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S. & Dean, J. (2019). A guide to deep learning in healthcare. *Nature medicine*, **25**, 24–29. 12

Farimani, A.B., Gomes, J. & Pande, V.S. (2017). Deep learning the physics of transport phenomena. *arXiv preprint arXiv:1709.02432*. 13

Fourier, J.B.J. (1878). *The analytical theory of heat*. The University Press. 183

Gao, Y. & Ng, M.K. (2022). Wasserstein generative adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, **463**, 111270. 14

Geiger, M. & Smidt, T. (2022). e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*. 16

Gonzalez, F., Demoulin, F.X. & Bernard, S. (2023). Towards long-term predictions of turbulence using neural operators. *arXiv preprint arXiv:2307.13517*. 45

Goswami, S., Bora, A., Yu, Y. & Karniadakis, G.E. (2022). Physics-informed neural operators. *arXiv preprint arXiv:2207.05748*. 18

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. *et al.* (2018). Recent advances in convolutional neural networks. *Pattern recognition*, **77**, 354–377. 15

Gupta, J.K. & Brandstetter, J. (2022). Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*. 34, 102, 104

Han, X., Gao, H., Pfaff, T., Wang, J.X. & Liu, L.P. (2022). Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*. 102

Hansen, D., Maddix, D.C., Alizadeh, S., Gupta, G. & Mahoney, M.W. (2023). Learning physical models that can respect conservation laws. *arXiv preprint arXiv:2302.11002*. 16

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645, Springer. 104

Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J. & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, **13**, 18–28. 15

Heaton, J.B., Polson, N.G. & Witte, J.H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, **33**, 3–12. 12

Heinlein, A., Klawonn, A., Lanser, M. & Weber, J. (2021). Combining machine learning and domain decomposition methods for the solution of partial differential equations—a review. *GAMM-Mitteilungen*, **44**, e202100001. 13

Herde, M., Raonić, B., Rohner, T., Käppeli, R., Molinaro, R., de Bézenac, E. & Mishra, S. (2024). Poseidon: Efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, arXiv:2405.19101. 3, 103

Hess, F., Monfared, Z., Brenner, M. & Durstewitz, D. (2023). Generalized teacher forcing for learning chaotic dynamics. *arXiv preprint arXiv:2306.04406*. 50

Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, **2**, 359–366. 22

Iakovlev, V., Heinonen, M. & Lähdesmäki, H. (2020). Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*. 24

Kamilaris, A. & Prenafeta-Boldú, F.X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, **147**, 70–90. 12

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S. & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, **3**, 422–440. 15

Kazemi, S.M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P. & Brubaker, M. (2019). Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*. 106

Kissas, G., Seidman, J.H., Guilhoto, L.F., Preciado, V.M., Pappas, G.J. & Perdikaris, P. (2022). Learning operators with coupled attention. *Journal of Machine Learning Research*, **23**, 1–63. 22, 23, 102, 108

Koutnik, J., Greff, K., Gomez, F. & Schmidhuber, J. (2014). A clockwork rnn. In *International conference on machine learning*, 1863–1871, PMLR. 25

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A. & Anandkumar, A. (2021). Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*. 12, 16, 17, 19, 20, 21, 23, 30, 31, 32, 41, 42

Kovachki, N.B., Lanthaler, S. & Stuart, A.M. (2024). Operator learning: Algorithms and analysis. *Handbook of Numerical Analysis*, **25**, 419–467. 44

Krizhevsky, A., Sutskever, I. & Hinton, G.E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, **60**, 84–90. 2, 12

Kutyniok, G., Petersen, P., Raslan, M. & Schneider, R. (2022). A theoretical analysis of deep neural networks and parametric pdes. *Constructive Approximation*, **55**, 73–125. 14

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. & Jackel, L.D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**, 541–551. 15

Li, B., Wang, H., Yang, X. & Lin, Y. (2022a). Solving seismic wave equations on variable velocity models with fourier neural operator. *arXiv preprint arXiv:2209.12340*. 19

Li, S., Xue, K., Zhu, B., Ding, C., Gao, X., Wei, D. & Wan, T. (2020a). Falcon: A fourier transform based approach for fast and secure convolutional neural network predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8705–8714. 22

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. & Anandkumar, A. (2020b). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*. 3, 8, 13, 20, 36, 41, 42, 43, 100, 102

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. & Anandkumar, A. (2020c). Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*. 17, 18, 19, 23

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K. & Anandkumar, A. (2020d). Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, **33**, 6755–6766. 19

Li, Z., Kovachki, N.B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A. & Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*. 46, 49

Li, Z., Huang, D.Z., Liu, B. & Anandkumar, A. (2022b). Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*. 23

Li, Z., Meidani, K. & Farimani, A.B. (2022c). Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*. 25, 90, 102, 103, 155, 156

Li, Z., Peng, W., Yuan, Z. & Wang, J. (2023). Long-term predictions of turbulence by implicit u-net enhanced fourier neural operator. *Physics of Fluids*, **35**. 49

Lim, B. & Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, **379**, 20200209. 24

Lino, M., Fotiadis, S., Bharath, A.A. & Cantwell, C. (2022). Towards fast simulation of environmental fluid mechanics with multi-scale graph neural networks. *arXiv preprint arXiv:2205.02637*. 24

Lippe, P., Veeling, B.S., Perdikaris, P., Turner, R.E. & Brandstetter, J. (2023). Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *arXiv preprint arXiv:2308.05732*. 3, 49, 102

Lippe, P., Veeling, B., Perdikaris, P., Turner, R. & Brandstetter, J. (2024). Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, **36**. 45

List, B., Chen, L.W., Bali, K. & Thuerey, N. (2025). Differentiability in unrolled training of neural physics simulators on transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, **433**, 117441. 45, 51

Liu, B., Tang, J., Huang, H. & Lu, X.Y. (2020). Deep learning methods for super-resolution reconstruction of turbulent flows. *Physics of Fluids*, **32**, 025105. 14, 21

Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G.E. (2021). Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, **3**, 218–229. 8, 17, 18

Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z. & Karniadakis, G.E. (2022a). A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, **393**, 114778. 18, 20, 21

Lu, L., Pestourie, R., Johnson, S.G. & Romano, G. (2022b). Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *arXiv preprint arXiv:2204.06684*. 18, 44

Luo, D., Chen, Z., Hu, K., Zhao, Z., Hur, V.M. & Clark, B.K. (2023). Gauge-invariant and anyonic-symmetric autoregressive neural network for quantum lattice models. *Physical Review Research*, 013216. 12

Manning, C. & Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press. 2, 12

Mardt, A., Pasquali, L., Wu, H. & Noé, F. (2018). Vampnets for deep learning of molecular kinetics. *Nature communications*, **9**, 5. 12

McCabe, M., Harrington, P., Subramanian, S. & Brown, J. (2023). Towards stability of autoregressive neural operators. *arXiv preprint arXiv:2306.10619*. 2, 45, 99, 102

McCabe, M., Mukhopadhyay, P., Marwah, T., Régaldo-Saint Blancard, B., Rozet, F., Diaconu, C., Meyer, L., Wong, K.W.K., Sotoudeh, H., Bietti, A. *et al.* (2025). Walrus: A cross-domain foundation model for continuum dynamics. *arXiv preprint arXiv:2511.15684*. 103

McGreivy, N. & Hakim, A. (2024). Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, **6**, 1256–1269. 2

Medsker, L.R. & Jain, L. (2001). Recurrent neural networks. *Design and Applications*, **5**, 64–67. 15

Michałowska, K., Goswami, S., Karniadakis, G.E. & Riemer-Sørensen, S. (2024). Neural operator learning for long-time integration in dynamical systems with recurrent neural networks. In *2024 International Joint Conference on Neural Networks (IJCNN)*, 1–8, IEEE. 49

Migus, L., Salomon, J. & Gallinari, P. (2023). Stability of implicit neural networks for long-term forecasting in dynamical systems. *arXiv preprint arXiv:2305.17155*. 49

Mikhaeil, J., Monfared, Z. & Durstewitz, D. (2022). On the difficulty of learning chaotic dynamics with rnns. *Advances in neural information processing systems*, **35**, 11297–11312. 53

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*. 12

Myles, A.J., Feudale, R.N., Liu, Y., Woody, N.A. & Brown, S.D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, **18**, 275–285. 15

Nelson, B.K. (1998). Time series analysis using autoregressive integrated moving average (arima) models. *Academic emergency medicine*, **5**, 739–744. 24

Pascanu, R., Mikolov, T. & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, 1310–1318, Pmlr. 45

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. (2017). Automatic differentiation in pytorch. 16

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K. *et al.* (2022). Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*. 19

Peng, W., Yuan, Z. & Wang, J. (2022). Attention-enhanced neural network models for turbulence simulation. *Physics of Fluids*, **34**. 25, 49, 102, 108

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A. & Battaglia, P.W. (2020). Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*. 23

Pfau, D., Spencer, J.S., Matthews, A.G. & Foulkes, W.M.C. (2020). Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, **2**, 033429. 13

Pinkus, A. (2012). *N-widths in Approximation Theory*, vol. 7. Springer Science & Business Media. 20

Prantl, L., Ummenhofer, B., Koltun, V. & Thuerey, N. (2022). Guaranteed conservation of momentum for learning particle-based fluid dynamics. *Advances in Neural Information Processing Systems*, **35**, 6901–6913. 16

Qu, J., Cai, W. & Zhao, Y. (2022). Learning time-dependent pdes with a linear and nonlinear separate convolutional neural network. *Journal of Computational Physics*, **453**, 110928. 24

Rahman, M.A., Ross, Z.E. & Azizzadenesheli, K. (2023). U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*. 34, 35, 46, 49, 102

Raimi, D., Zhu, Y., Newell, R.G., Prest, B.C. & Bergman, A. (2023). Global energy outlook 2023: Sowing the seeds of an energy transition. *Tehnical Report, Report 23-02*. 12

Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, **19**, 932–955. 8, 13

Raissi, M., Perdikaris, P. & Karniadakis, G.E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, **378**, 686–707. 13, 16

Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S. *et al.* (2021). Skilful precipitation nowcasting using deep generative models of radar. *Nature*, **597**, 672–677. 13

Risken, H. & Risken, H. (1996). *Fokker-planck equation*. Springer. 9

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N. & Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **371**, 20110550. 24

Rohrhofer, F.M., Posch, S., Gössnitzer, C. & Geiger, B.C. (2022). Understanding the difficulty of training physics-informed neural networks on dynamical systems. *arXiv preprint arXiv:2203.13648*. 16

Roska, T., Chua, L.O., Wolf, D., Kozek, T., Tetzlaff, R. & Puffer, F. (1995). Simulating nonlinear waves and partial differential equations via cnn. i. basic techniques. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **42**, 807–815. 14

Ruff, K.M. & Pappu, R.V. (2021). Alphafold and implications for intrinsically disordered proteins. *Journal of Molecular Biology*, **433**, 167208. 12

Sanchez-Gonzalez, A., Stachenfeld, K., Fielding, D., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S. & Battaglia, P. (2021). Learning general-purpose cnn-based simulators for astrophysical turbulence. In *ICLR 2021 SimDL Workshop*. 45, 49

Söderlind, G. (2002). Automatic control and adaptive time-stepping. *Numerical Algorithms*, **31**, 281–310. 102

Strauss, W.A. (2007). *Partial differential equations: An introduction*. John Wiley & Sons. 1

Szabó, A. & Castelnovo, C. (2020). Neural network wave functions and the sign problem. *Physical Review Research*, **2**, 033075. 12

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D. & Niepert, M. (2022). Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, **35**, 1596–1611. 36, 40, 42, 43, 76, 102

Takamoto, M., Alesiani, F. & Niepert, M. (2023). Learning neural pde solvers with parameter-guided channel attention. In *International Conference on Machine Learning*, 33448–33467, PMLR. 50, 95, 102

Tan, L. & Chen, L. (2022). Enhanced deeponet for modeling partial differential operators considering multiple input functions. *arXiv preprint arXiv:2202.08942*. 18

Tan, W.Y. (1992). *Shallow water hydrodynamics: Mathematical theory and numerical solution for a two-dimensional system of shallow-water equations*. Elsevier. 9

Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T. & Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 7537–7547. 106

Teutsch, P. & Mäder, P. (2022). Flipped classroom: effective teaching for time series forecasting. *arXiv preprint arXiv:2210.08959*. 50, 95

Thompson, S.E. & Parthasarathy, S. (2006). Moore's law: the future of si microelectronics. *Materials today*, **9**, 20–25. 12

Tran, A., Mathews, A., Xie, L. & Ong, C.S. (2021). Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*. 23, 33

Tripura, T. & Chakraborty, S. (2022). Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*. 23

Vashchenko, V.A. & Sinkevitch, V.F. (2008). *Physical limitations of semiconductor devices*, vol. 340. Springer. 12

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, **30**. 25, 102, 106, 110

Vital, J.A., Gaurut, M., Lardy, R., Viovy, N., Soussana, J.F., Bellocchi, G. & Martin, R. (2013). High-performance computing for climate change impact studies with the pasture simulation model. *Computers and electronics in agriculture*, **98**, 131–135. 12

Vlachas, P.R. & Koumoutsakos, P. (2024). Learning on predictions: Fusing training and autoregressive inference for long-term spatiotemporal forecasts. *Physica D: Nonlinear Phenomena*, **470**, 134371. 45, 50, 95

Vlachas, P.R., Arampatzis, G., Uhler, C. & Koumoutsakos, P. (2022). Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, **4**, 359–366. 14, 25

Wang, R., Kashinath, K., Mustafa, M., Albert, A. & Yu, R. (2020). Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1457–1466. 14

Wang, S. & Perdikaris, P. (2023). Long-time integration of parametric evolution equations with physics-informed deeponets. *Journal of Computational Physics*, **475**, 111855. 49

Wang, S., Wang, H. & Perdikaris, P. (2021). On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, **384**, 113938. 16

Wang, S., Seidman, J.H., Sankaran, S., Wang, H., Pappas, G.J. & Perdikaris, P. (2024). CViT: Continuous vision transformer for operator learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, arXiv:2405.13998. 103

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A. & Benson, S.M. (2022). U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, **163**, 104180. 12, 19, 23

Wu, H., Luo, H., Wang, H., Wang, J. & Long, M. (2024). Transolver: A fast transformer solver for PDEs on general geometries. *arXiv preprint arXiv:2402.02366*. 103

Wu, T., Maruyama, T. & Leskovec, J. (2022). Learning to accelerate partial differential equations via latent global evolution. *Advances in Neural Information Processing Systems*, **35**, 2240–2253. 13, 49, 104

Xu, W., Lu, Y. & Wang, L. (2023). Transfer learning enhanced deeponet for long-time prediction of evolution equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 10629–10636. 49

Xu, Z.J. (2018). Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*. 22

Yang, Y., Zhang, X., Guan, Q. & Lin, Y. (2022). Making invisible visible: Data-driven seismic inversion with spatio-temporally constrained data augmentation. *IEEE Transactions on Geoscience and Remote Sensing*, **60**, 1–16. 15

You, H., Yu, Y., D'Elia, M., Gao, T. & Silling, S. (2022). Nonlocal kernel network (nkn): a stable and resolution-independent deep neural network. *arXiv preprint arXiv:2201.02217*. 19

Yu, Y., Si, X., Hu, C. & Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, **31**, 1235–1270. 25

Yun, S., Jeong, M., Kim, R., Kang, J. & Kim, H.J. (2019). Graph transformer networks. *Advances in neural information processing systems*, **32**. 24

Zhang, P., Hu, Y., Jin, Y., Deng, S., Wu, X. & Chen, J. (2021). A maxwell's equations based deep learning method for time domain electromagnetic simulations. *IEEE Journal on Multiscale and Multiphysics Computational Techniques*, **6**, 35–40. 13

Zhang, X., Wang, L., Helwig, J., Luo, Y., Fu, C., Xie, Y., Liu, M., Lin, Y., Xu, Z., Yan, K. *et al.* (2023). Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*. 99

Appendix A

Fourier Analysis

Fourier Analysis provides a way to represent general functions as a sum of simple trigonometric functions (sine and cosine functions) of increasing frequency. This method was introduced by Fourier (1878) to study the heat equation, showing that the eigenfunctions of the heat equation are a Fourier transform *basis* of sines and cosines and the eigenvalues are the specific frequencies determined by the boundary conditions. Hence, sine and cosine functions of increasing frequency form an orthogonal basis in function space. The Fast Fourier Transform (FFT) provides an efficient way to compute Fourier transforms numerically (Cooley & Tukey, 1965). Fourier analysis has been instrumental in applications such as image & signal processing and solving Partial Differential Equations (PDEs).

A.1 Fourier Series and Transform

Consider a continuous top-hat function as shown in Fig. A.1. The Fourier series for an L-periodic function on $[-L, L]$ is given by:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos\left(\frac{\pi k x}{L}\right) + b_k \sin\left(\frac{\pi k x}{L}\right) \right), \tag{A.1}$$

with coefficients a_k and b_k given by

$$a_k = \frac{1}{2L} \int_{-L}^{L} f(x) \cos\left(\frac{\pi k x}{L}\right) dx, \tag{A.2}$$

$$b_k = \frac{1}{2L} \int_{-L}^{L} f(x) \sin\left(\frac{\pi k x}{L}\right) dx. \tag{A.3}$$

The coefficients a_k and b_k can be understood as the projection of the function $f(x)$ onto the orthogonal cosine and sine basis $\{\cos(\pi kx/L), \sin(\pi kx/L)_{k=0}^{\infty}\}$.

It is more convenient to work in complex form, and express the sine and cosine functions using Euler's formula $e^{ikx} = \cos(kx) + i\sin(kx)$ with complex coefficient $c_k = \alpha_k + i\beta_k$.

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}. \tag{A.4}$$

The functions $\psi_k = e^{ikx}$ provide a basis for periodic, complex-valued functions.

A Fourier series represents periodic functions so that the function repeats itself outside the domain, whereas the Fourier transform is a limit of the Fourier series as the length of the domain goes to infinity. On a domain $x \in [-L, L]$, the Fourier series is

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos\left(\frac{\pi kx}{L}\right) + b_k \sin\left(\frac{\pi kx}{L}\right)\right) = \sum_{k=-\infty}^{\infty} c_k e^{i\pi kx/L} \tag{A.5}$$

where c_k is

$$c_k = \frac{1}{2L} \int_{-L}^{L} f(x) e^{ik\pi x/L}\, dx. \tag{A.6}$$

The function $f(x)$ is represented by a sum of sines and cosines with a discrete set of frequencies given by $\omega_k = k\pi/L$. In the Fourier transform, the discrete frequencies become continuous as $L \to \infty$.

As $L \to \infty$, let $\omega = k\pi/L = k\Delta\omega$, where $\Delta\omega = \pi/L$. Thus, the Fourier transform becomes

$$f(x) = \lim_{\Delta\omega \to 0} \sum_{k=-\infty}^{\infty} \frac{\Delta\omega}{2\pi} \int_{-\pi/\Delta\omega}^{\pi/\Delta\omega} f(\xi) e^{-ik\Delta\omega\xi}\, d\xi\ e^{ik\Delta\omega x} \tag{A.7}$$

The inner integral becomes the Fourier transform of $f(x)$, i.e. $\hat{f}(\omega) \triangleq \mathcal{F}(f(x))$. The summation with weight $\Delta\omega$ becomes a Riemann integral:

$$f(x) = \mathcal{F}^{-1}(\hat{f}(\omega)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{-i\omega x}\, d\omega \tag{A.8}$$

$$\hat{f}(\omega) = \mathcal{F}(f(x)) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x}\, dx. \tag{A.9}$$

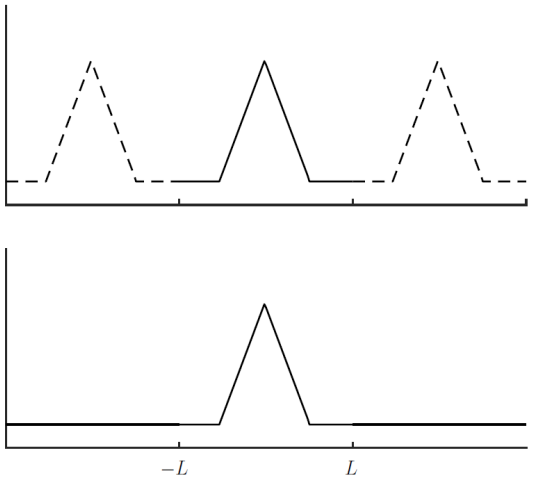


Figure A.1: (top) Periodic function valid for Fourier series. (bottom) Non-periodic function valid for the Fourier transform.

A.2 Properties of the Fourier transform

A.2.1 Derivatives of functions

The Fourier transform of the derivative of a function is defined as

$$\begin{aligned}\mathcal{F}\left(\frac{d}{dx}f(x)\right) &= \int_{-\infty}^{\infty} f'(x)e^{-i\omega x}\,dx \\ &= i\omega\int_{-\infty}^{\infty} f(x)e^{-i\omega x}\,dx \qquad \text{(Integrating by parts)} \\ &= i\omega\mathcal{F}(f(x)).\end{aligned} \tag{A.10}$$

Therefore, the differentiation of a function is a multiplication operation in the Fourier space.

A.2.2 Linearity

The Fourier transform is a linear operator

$$\mathcal{F}(\alpha f(x) + \beta g(x)) = \alpha\mathcal{F}(f) + \beta\mathcal{F}(g). \tag{A.11}$$

$$\mathcal{F}^{-1}(\alpha\hat{f}(\omega) + \beta\hat{g}(\omega)) = \alpha\mathcal{F}^{-1}(\hat{f}) + \beta\mathcal{F}^{-1}(\hat{g}). \tag{A.12}$$

A.2.3 Parseval's Theorem

The Fourier transform preserves the L_2 norm up to a constant factor.

$$\int_{-\infty}^{\infty} |\hat{f}(\omega)|^2 d\omega = 2\pi \int_{-\infty}^{\infty} |f(x)|^2 dx \tag{A.13}$$

A.2.4 Convolution

The convolution of two functions, i.e. $(f * g)$ in the Fourier space, is a product of two Fourier-transformed functions.

$$\begin{aligned}
\mathcal{F}^{-1}(\hat{f}\hat{g}) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega)\hat{g}(\omega) e^{i\omega x}\, d\omega \\
&= \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega x} \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} g(y) e^{-i\omega y}\, dy \right) d\omega \\
&= \int_{-\infty}^{\infty} g(y) \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega(x-y)}\, d\omega \right) dy \\
&= \int_{-\infty}^{\infty} g(y) f(x-y) dy = g * f = f * g
\end{aligned} \tag{A.14}$$