IMPERIAL

Scientific Machine Learning for Modelling and Optimisation of Nonlinear Partial Differential Equations in Fluids

Defne Ege Ozan

Imperial College London

Department of Aeronautics

December 2025

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Aeronautics of Imperial College London and the Diploma of Imperial College London.

Declaration

I hereby declare that this thesis is the result of my own work, except where explicit reference is made to the work of others, in which case they are appropriately acknowledged.

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Defne Ege Ozan

Abstract

The prediction and control of flow and acoustic systems present a fundamental challenge due to the high-dimensional, nonlinear and chaotic dynamics with multi-physics interactions. This thesis proposes scientific machine learning methods for modelling these systems, reconstruction of their full state from partial observations, and their optimisation.

First, we model thermoacoustic dynamics from synthetic sensor data. We develop Galerkin neural networks, which learn from data, whilst being constrained with prior knowledge of the acoustics by (i) employing periodic activations; (ii) a physics-informed loss in the training; and (iii) a hard-constrained architecture in a physically-motivated solution space. We test the network on a prototypical nonlinear time-delayed model, i.e., the Rijke tube, and a higher-fidelity model. We accurately reconstruct the velocity from only pressure measurements.

Second, we infer gradients (sensitivities) from data with the adjoint method. The parameter-aware echo state network learns the dynamics of nonlinear regimes with varying parameters. We derive its adjoint, and infer the climate sensitivity of the chaotic Lorenz system to the system's parameters. The time-delayed thermoacoustic echo state network improves generalisability on the Rijke tube, and accurately infers the adjoint sensitivities of the acoustic energy with respect to the flame parameters and initial conditions, whilst identifying local bifurcations. We suppress a nonlinear oscillation via gradient-based optimisation.

Third, we enable the active control of chaotic systems with partial observability. Data-assimilated model-informed reinforcement learning integrates (i) low-order models to approximate high-dimensional dynamics; (ii) sequential data assimilation to correct the model prediction when observations become available; and (iii) an off-policy actor-critic reinforcement learning algorithm to discover an optimal control strategy. We test the framework on the chaotic solutions of the Kuramoto-Sivashinsky equation. We estimate its full state with (i) a physics-based coarse-grained model; and (ii) the control-aware echo state network to stabilise its chaotic dynamics.

Acknowledgements

First of all, I would like to acknowledge my supervisor, Luca Magri, to whom I am deeply grateful for this opportunity. I thank him for his scientific guidance and rigour, and his mentorship. His relentless questioning —*"So what?"*— has pushed me to think more deeply about the meaning of my results beyond the technical details. His enthusiasm for pushing the boundaries in research has been a great source of inspiration.

I would like to gratefully acknowledge the ERC Starting Grant PhyCo (949388) and UKRI Grant AI for Net Zero (EP/Y005619/1) for funding me during my PhD. I would also like to thank the Leeds Institute of Fluid Dynamics (LIFD) for the best talk award that I received at their data-driven methods for fluid mechanics workshop.

I would like to thank Georgios Rigas for welcoming me to the AI for Net Zero team, and generously offering me his expertise with unmatched enthusiasm. I was fortunate to work closely with Andrea Nóvoa, with whom I shared insightful discussions and long derivations. I thank them both for our collaboration in the final chapter of my thesis. I extend my sincere thanks to everyone in the AI for Net Zero project. It was a privilege to be part of such an exceptional, multidisciplinary team, and I am grateful for all that I have learned. I would like to especially thank Max Weissenbacher for helping me get started with reinforcement learning and for the fruitful discussions that we had.

I would like to thank my examiners Jonas Moeck and Yongyun Hwang for their constructive comments on the thesis and for their thoughtful questions that made the viva an enjoyable discussion.

I would like to thank the Department of Aeronautics for creating a supportive environment ideal for research. I would like to express my gratitude to Lisa Kelly, Siân Haynes and Clodagh Li for their efforts in helping me at various stages.

I feel lucky and honoured to have been part of a research group with such brilliant colleagues, who have been equally caring and kind. I thank Elise Özalp, Yaxin Mo and Daniel Kelshaw, who started this PhD journey with me, and Osama Ahmed, who joined us later, for the camaraderie we have built, the unwavering support they have given me through the ups and downs of the PhD, and the many scientific discussions we had. A special thanks goes out to Yaxin and Daniel for maintaining our group machines, Yael and Maya. I thank Alberto Racca, Georgios Margazoglou, Tullio Traverso, Nikos Bempedelis, Antonio Colanera and Lloyd Fung for sharing their knowledge with me through valuable discussions and encouraging me to keep going. I appreciated the opportunity to interact with many colleagues in the Department, and I am particularly thankful to Yacine Bengana, Sidhartha Sahu and Priyam Gupta for their support and insights.

I would like to thank the open-source community for the software and the shared knowledge that enabled my work, and the broader scientific community for the open exchange of ideas and discourse. I thank the anonymous reviewers of my publications, whose feedback strengthened my work.

I am grateful to my parents, Ahu and Hüsnü, and my grandparents, Çiğdem and Alpaslan, for their encouragement and continuous support of my pursuit in science. I give my heartfelt thanks to my friends, Doğa, Ecem, Ekin, Michaël and Zoe, who have been there for me no matter the

distance. Otto deserves the most special thanks because he always manages to bring me joy. And to my partner in this shared path, thank you for everything.

Contents

1 Introduction **17**

1.1 Modelling and reconstruction 18

1.1.1 Modelling the temporal evolution of dynamics 18

1.1.2 Reconstruction from partial state information 20

1.1.3 Physics-constrained machine learning 21

1.2 Optimisation and control 21

1.2.1 Equation-based optimisation and control 21

1.2.2 Data-driven control strategies 23

1.3 Thesis objectives 24

1.4 Thesis structure 25

2 Modelling and optimisation of dynamical systems **27**

2.1 Definition of a dynamical system 27

2.2 Partial differential equations and Galerkin projection 28

2.3 Observations 29

2.4 Adjoint sensitivity 30

3 Nonlinear dynamical systems **32**

3.1 Nonlinearity, bifurcations and the route to chaos 32

3.2 Sensitivities in chaotic systems 33

3.3 Chaos: the Lorenz system 34

3.4 Multi-physics: Thermoacoustics and the Rijke tube 35

3.4.1 The Rijke tube 36

3.4.2 Eigenmodes of the Rijke tube 36

3.4.3 A low-order model using Galerkin projection 37

3.4.4 Bifurcation analysis using the low-order model 39

3.4.5 A higher-fidelity model using travelling waves 40

3.5 Spatiotemporal chaos: the Kuramoto-Sivashinsky equation 41

4 Hard-constrained neural networks for modelling nonlinear thermoacoustics **43**

4.1 Introduction 43

4.2 Standard feedforward neural networks 45

4.3 Galerkin neural networks 47

4.4 Acoustic neural networks 47

4.4.1 Periodically activated feedforward neural networks *(P-FNNs)* 49
4.4.2 Soft constraints with physics-informed losses 49
4.4.3 Hard-constraints with Galerkin neural networks *(GalNNs)* 50
4.4.4 Periodic Galerkin neural networks *(P-GalNNs)* 52
4.5 Extrapolation and state reconstruction: Twin experiments 53
4.5.1 Computation of the physics-informed loss 54
4.5.2 Extrapolation in time 55
4.5.3 Learning from partial state measurements 59
4.6 Results on higher-fidelity data 60
4.6.1 Comparison of networks performance for reconstruction 62
4.6.2 Robustness to sparsity and noise 64
4.6.3 Inferring heat release 68
4.6.4 Nonideal boundary conditions 70
4.7 Conclusion 71

5 Data-driven computation of adjoint sensitivities without adjoint solvers 73
5.1 Introduction 73
5.2 Parameter-aware echo state networks *(PA-ESNs)* 75
5.2.1 Architecture and training of the PA-ESN 76
5.2.2 Performance metrics 79
5.3 Adjoint of the PA-ESN 79
5.3.1 Sensitivity to the system's parameters 80
5.3.2 Sensitivity to the system's initial conditions 81
5.3.3 Jacobian of the PA-ESN 82
5.3.4 Sensitivity to the network weights 83
5.4 Learning the Lorenz system and parameter sensitivities 83
5.4.1 Updating weights with adjoint sensitivities 88
5.5 Thermoacoustic echo state network *(T-ESN)* 89
5.5.1 Design of the T-ESN 89
5.5.2 Adjoint of the T-ESN 91
5.5.3 Jacobian of the T-ESN 92
5.6 Learning thermoacoustics and parameter sensitivities 92
5.6.1 Training and validation 92
5.6.2 Prediction performance 95
5.6.3 Computation of adjoint sensitivities 97
5.7 Gradient-based optimisation of thermoacoustics 98
5.8 Robustness to noise 100
5.9 Learning bifurcations of thermoacoustics 102
5.10 Inferring sensitivities to initial conditions 107
5.11 Discussion 108
5.12 Conclusion 109

6 Data-assimilated model-informed reinforcement learning 111
6.1 Introduction 111

6.2 Methodology . . . 114
6.2.1 Reinforcement learning . . . 114
6.2.2 Partially Observed Markov Decision Processes (POMDPs) . . . 116
6.2.3 State estimation via data assimilation . . . 118
6.2.4 Data-assimilated model-informed reinforcement learning *(DA-MIRL)* . . . 119
6.3 Application to spatiotemporal chaos . . . 121
6.3.1 The environment . . . 122
6.3.2 The environment model . . . 123
6.3.3 The agent . . . 127
6.4 Training of DA-MIRL . . . 128
6.4.1 Performance metrics . . . 129
6.5 Baseline analysis of model-free reinforcement learning . . . 130
6.6 Controlling spatiotemporal chaos from partial observations with DA-MIRL . . . 133
6.6.1 Results of DA-MIRL with truncated Fourier model . . . 133
6.6.2 Results of DA-MIRL with control-aware echo state network *(CA-ESN)* . . . 137
6.6.3 Generalisation to more chaotic regimes . . . 141
6.7 Conclusion . . . 142

7 Conclusion and Outlook **143**
7.1 Conclusion . . . 143
7.2 Outlook . . . 145

A Thermoacoustics **165**
A.1 Derivation of the thermoacoustic eigenmodes . . . 165

B Adjoint equations **167**
B.1 Derivation of the adjoint of a discrete-time system . . . 167
B.2 Derivation of the adjoint of a time-delayed discrete-time system . . . 168

C Ensemble Kalman Filter **170**
C.1 Derivation of the posterior update . . . 170

D Additional results **172**
D.1 Results of the DA-MIRL using the CA-ESN model with Lu readout . . . 172

E Computational costs **174**
E.1 Computational costs in Chapter 4 . . . 174
E.2 Computational costs in Chapter 5 . . . 175
E.3 Computational costs in Chapter 6 . . . 176

List of Figures

3.1 Schematic of the Rijke tube. 35
3.2 Bifurcation diagram of the Rijke tube with time delay $\tau = 0.2$ and varying heat source strength β. 39
3.3 Different dynamical regimes of the Kuramoto-Sivashinsky (KS) equation with varying viscosity parameter ν. The solutions become increasingly more intricate as ν decreases. 41

4.1 A standard feedforward neural network (FNN). 45
4.2 Galerkin neural network (GalNN). This network is composed of two branches; (a) spatial branch (hard constraint), and (b) a temporal branch, which learns the temporal behaviour. 48
4.3 The Galerkin neural network (GalNN) tailored for thermoacoustics. 51
4.4 Extrapolation in time of the acoustic pressure with periodically activated neural networks in comparison to standard activations. 56
4.5 Effect of hyperparameter ξ in the sine activation $1/\xi \cdot \sin(\xi z)$. 56
4.6 Long-term extrapolation in time of periodic acoustic pressure with acoustic neural networks; periodically activated feedforward neural network (P-FNN), Galerkin neural network (GalNN), and periodic Galerkin neural network (P-GalNN). 58
4.7 Trained weights of the first hidden layers of acoustic neural networks. 58
4.8 Extrapolation in time of non-periodic acoustic pressure with GalNN. 60
4.9 Reconstruction of acoustic velocity from pressure measurements only using P-FNN and GalNN and their physics-informed counterparts. 61
4.10 Acoustic eigenfunctions, which span the solution space of the GalNN. 62
4.11 Training feedforward neural networks on data from higher-fidelity model for a periodic acoustic solution. 63
4.12 Comparison of acoustic neural networks in terms of reconstruction performance from higher-fidelity data. 64
4.13 Effect of the regularisation on P-GalNNs in case of sparse and noisy measurements. 65
4.14 Effect of number of pressure sensors and Galerkin modes in the GalNN with and without physics-based loss. 66
4.15 Robustness to noise with GalNNs. 67
4.16 Reconstruction of acoustic velocity from only noisy pressure measurements using physics-informed periodically activated feedforward neural network (PI-P-FNN) an physics-informed periodic Galerkin neural network (PI-P-GalNN) 68

4.17 Reconstruction of acoustic pressure and velocity from only pressure measurements using PI-P-GalNN with different number of sensors and Galerkin modes. 69
4.18 Inference of heat release rate using P-GalNNs trained on pressure and velocity measurements. 70
4.19 Comparison of prediction of acoustic neural networks in the case of nonideal boundary conditions. 71

5.1 Schematic of the parameter-aware echo state network (PA-ESN). 76
5.2 Prediction of the Lorenz attractor with varying system's parameters with PA-ESN. 84
5.3 Short-term prediction and long-term inference of statistics of different chaotic regimes of the Lorenz system with PA-ESN. 84
5.4 Convergence of the ensemble adjoint estimate for the Lorenz system with different integration times. 87
5.5 Inference of adjoint sensitivities of the Lorenz system to the system's parameters. . 87
5.6 Adjoint-accelerated gradient updates of the trainable weights of the ESN to minimise the prediction error on a target trajectory. 88
5.7 Schematic of the proposed thermoacoustic echo state network (T-ESN) in closed-loop configuration. 90
5.8 Short-term closed-loop prediction performance of (a) standard PA-ESN (5.5), (b) advection-augmented PA-ESN (5.5), (c) T-ESN (5.27) on the Rijke tube data. . . . 96
5.9 Prediction of short-term time series, long-term statistics and amplitude spectrum of the acoustic energy of seen and unseen limit-cycle regime with T-ESN. 96
5.10 T-ESN infers sensitivity of the time-averaged acoustic energy, i.e., objective $\mathcal{J}$, to heat release strength β. 97
5.11 T-ESN infers sensitivity of the time-averaged acoustic energy, i.e., objective $\mathcal{J}$, to time-delay τ. 97
5.12 Gradient-based optimisation of the time-averaged acoustic energy starting from different initial (β, τ) using adjoint sensitivities computed by a single instance of the T-ESN. 99
5.13 Gradient-based optimisation of time-averaged acoustic energy marked as Path 3 in Fig. 5.12. 100
5.14 Selection of the regularisation parameter λ in the ESN when the data is noisy. . . . 101
5.15 Prediction of limit-cycle attractors from noisy data with T-ESN. 102
5.16 Effect of regularisation in the inference of sensitivities to the heat release strength β with the T-ESN. 102
5.17 Effect of regularisation in the inference of sensitivities to the time delay τ with the T-ESN. 103
5.18 Prediction of the Hopf bifurcation point with T-ESN trained on only limit-cycle data. 103
5.19 Thermoacoustic echo state network captures bifurcations to period-doubling and quasi-periodic regimes; prediction of short-term time series, long-term statistics and the amplitude spectra in these regimes. 104
5.20 T-ESN captures bifurcations to chaotic regimes; prediction of short-term time series, long-term statistics and the amplitude spectra in these regimes. 105

5.21 Bifurcation diagrams obtained from the true system, T-ESN trained on $\beta_{\text{train}} = [1, 5]$, and trained on $\beta_{\text{train}} = [6, 8]$. 106
5.22 Computation of chaotic adjoint sensitivities using T-ESN. 106
5.23 Computation of adjoint sensitivities to initial conditions using T-ESN. 107

6.1 Three-block schematic of reinforcement learning methods under partial observability. Comparison between (a) model-free RL with belief state and (b) the proposed data-assimilated model-informed RL. 112
6.2 Schematic of the data-assimilated model-informed reinforcement learning (DA-MIRL), which aims to control a partially observed environment. 120
6.3 Schematic of the control-aware echo state network (CA-ESN). 124
6.4 Training performance of model-free RL on the KS equation for (a) different number of sensors; (b) different placements of actuators, (c) varying levels of observation error level; and (d) varying frequency of observations. 132
6.5 Numerical simulations and energy spectra of the KS equation with varying number of Fourier modes using random actuations. 134
6.6 Truncated Fourier model: Training performance for varying number of sensors n_o. . 135
6.7 Truncated Fourier model: control performance of the DA-MIRL with $n_f = 16$. . . . 135
6.8 State estimation performance of the truncated Fourier model with $n_f = 16$ with random actuations. 136
6.9 Truncated Fourier model: DA-MIRL performance 20 evaluation episodes for varying number of sensors n_o and number of Fourier modes n_f. 136
6.10 CA-ESN: DA-MIRL training for varying number of sensors n_o. 137
6.11 CA-ESN: control performance of DA-MIRL with $n_h = 1000$. 138
6.12 State estimation performance of the CA-ESN with $n_h = 1000$ in an episode with random actuations. 139
6.13 CA-ESN: DA-MIRL performance 20 evaluation episodes with varying number of sensors n_o and reservoir size n_h. 139
6.14 CA-ESN prediction in closed loop (without data assimilation) throughout RL training. 140
6.15 CA-ESN: DA-MIRL performance over increasingly chaotic regimes with decreasing viscosity parameter ν. 141

D.1 CA-ESN: Training performance of DA-MIRL when using Lu readout (6.28) with varying number of sensors. 172
D.2 CA-ESN: control performance of DA-MIRL with the nonlinear observation operator defined by the Lu readout (6.28) 173
D.3 State estimation of the CA-ESN using the nonlinear observation operator defined by the Lu readout (6.28) in an episode with random actuations. 173

List of Tables

4.1 Summary of the proposed hard-constrained feedforward neural networks (FNNs). . 53
4.2 The Rijke tube's parameters for the dataset used in the training of the proposed FNNs. 55
4.3 Summary of FNN model properties trained on the Rijke tube data. 55
4.4 Summary of FNN and P-GalNN model properties trained on higher-fidelity data of the Rijke tube. 62

5.1 The Lorenz system's parameters and the parameter-aware echo state network's (PA-ESN) configurations for this dataset. 85
5.2 Optimal hyperparameters of the PA-ESN with a reservoir size of $n_h = 1200$ and connectivity $n_{\text{conn}} = 3$ that infers the parameterised dynamics of the Lorenz system. 86
5.3 The Rijke tube's parameters and the PA-ESN's configurations for this dataset. . . 94
5.4 Summary of the PA-ESN architectures for thermoacoustics data. 95
5.5 Optimal hyperparameters of (i) standard PA-ESN, (ii) advection-augmented PA-ESN, (iii) thermoacoustic ESN (T-ESN) trained on limit-cycle regimes of the Rijke tube. 95
5.6 Summary of the analysed dynamical regimes of the Rijke tube. 104
5.7 Optimal hyperparameters of the T-ESN (5.27) trained on data from $\beta_{\text{train}} = [1, 5]$ and $\beta_{\text{train}} = [6, 8]$ regimes of the Rijke tube. 105
5.8 Angle between the sensitivity gradients of the original system to initial conditions and the estimates from T-ESN. 107

6.1 Training parameters for data-assimilated model-informed reinforcement learning (DA-MIRL) agent and control-aware ESN (CA-ESN). 131
6.2 Summary of the relation between the environment quantities and their model approximations in DA-MIRL. 133
6.3 System's configurations of the controlled Kuramoto-Sivashinsky equation for Figs. 6.6-6.15 unless stated otherwise. 134

E.1 Computation time of twin experiments with $n_o^p = 13$ pressure sensors using physics-informed Galerkin and feedforward neural networks. (Ref. § 4.5.3). 174
E.2 Computation time of nonideal boundary condition higher-fidelity experiments with $n_o^p = 5$ pressure sensors using physics-informed Galerkin and feedforward neural networks. (Ref. § 4.6.4). 174

E.3 Computational performance comparison between original solver and thermoacoustic echo state network (T-ESN). (Ref. § 5.6). 175
E.4 Training times for model-free reinforcement learning and data-assimilated model-informed reinforcement learning. 176

List of Algorithms

1 Training of parameter-aware echo state network (PA-ESN) 78
2 Direct-adjoint looping for the computation of parameter sensitivities with PA-ESN 81
3 Direct-adjoint looping for the computation of the sensitivity of the acoustic energy to the flame parameters with thermoacoustic echo state network (T-ESN) 93
4 Gradient-based parameter optimisation of the Rijke tube with direct-adjoint looping of the trained T-ESN 101
5 Initialisation of deep deterministic policy gradient (DDPG). 116
6 Update of actor and critic using DDPG. 117
7 DA-MIRL: Data-assimilated model-informed reinforcement learning 121
8 The training procedure of model-free RL and DA-MIRL 129

Nomenclature

Acronyms/Abbreviations

CA-ESN	Control-aware echo state network
DA	Data assimilation
DA-MIRL	Data-assimilated model-informed reinforcement learning
DDPG	Deep deterministic policy gradient
EnKF	Ensemble Kalman filter
ESN	Echo state network
FNN	Feedforward neural network
GalNN	Galerkin neural network
KS	Kuramoto-Sivashinsky
LT	Lyapunov time
MDP	Markov Decision Process
ML	Machine learning
MSE	Mean squared error
ODE	Ordinary differential equation
P-FNN	Periodically activated feedforward neural network
PA-ESN	Parameter-aware echo state network
PDE	Partial differential equation
PDF	Probability density function
POMDP	Partially observed markov decision process
RL	Reinforcement learning
RNN	Recurrent neural network

SciML	Scientific machine learning
T-ESN	Thermoacoustic echo state network

Roman symbols

$\dot{q}_f$	Heat release rate in the Rijke tube
$\boldsymbol{C}$	Covariance matrix
$\boldsymbol{D}_c$	Chebyshev differential matrix
$\boldsymbol{F}$	ODE operator
$\boldsymbol{H}$	Concatenated matrix of reservoir states over time and regimes
$\boldsymbol{K}$	Kalman gain in EnKF
$\boldsymbol{M}$	Observation operator acting on the state
$\boldsymbol{W}$	Weights
$\boldsymbol{W}_{\text{in}}$	ESN input weight matrix
$\boldsymbol{W}_{\text{in}}^{\theta}$	ESN input weight matrix block corresponding to system's parameters
$\boldsymbol{W}_{\text{in}}^{y}$	ESN input weight matrix block corresponding to system's observables
$\boldsymbol{W}_{\text{out}}$	ESN output weight matrix
$\boldsymbol{W}_h$	ESN reservoir weight matrix
$\boldsymbol{Y}$	Concatenated matrix of output vectors over time and regimes
$\tilde{\boldsymbol{F}}$	ODE constraint
$\tilde{\boldsymbol{y}}_{\text{in}}$	Augmented input vector to the ESN
$\boldsymbol{a}$	Action/control input vector
$\hat{\boldsymbol{y}}$	Predicted output vector
$\boldsymbol{y}$	Physical state vector
$\boldsymbol{s}$	State vector
$\boldsymbol{b}$	Biases
$\boldsymbol{c}$	Fourier coefficient vector in the KS system
$\boldsymbol{g}_\theta$	Offset shift applied to parameters in ESN
$\boldsymbol{h}$	Reservoir state vector
$\boldsymbol{o}$	Observation vector
$\boldsymbol{q}^+$	Adjoint state vector

$\boldsymbol{y}_{\text{in}}$	Input vector to the ESN
d_1, d_2	Damping coefficients in the Rijke tube
E_{ac}	Acoustic energy
k	Time step
L	Length of one-dimensional domain
n_a	Number of actions/control inputs
n_{conn}	Number of non-zero connections per reservoir state variable (connectivity)
n_{en}	Ensemble size
N_{ep}	Number of time steps in an RL episode
N_{start}	Number of time steps before the observations and control start in DA-MIRL
n_θ	Number of system's parameters
n_s	Number of state variables
N_{train}	Number of training time steps
N_{val}	Number of validation time steps
N_{wash}	Number of time steps in the washout window
N_b	Batch size
n_c	Number of Chebyshev discretisation points
n_f	Number of Fourier modes
n_g	Number of Galerkin modes
n_h	Number of reservoir state variables
n_l	Number of hidden layers
n_y	Number of physical quantities
n_{neuron}	Number of neurons in each hidden layer
N_{reg}	Number of parameter regimes
p	Pressure
Q^π	State-action value or Q-function associated with policy π
r	Reward
T	Integration time
t	Time

u	Velocity
v	Advection dummy variable
x	Spatial coordinate
x_{L}	Variable of the Lorenz system
x_f	Flame location in the Rijke tube
y_{L}	Variable of the Lorenz system
z_{L}	Variable of the Lorenz system
A	Cross-sectional area

Greek symbols

α	Galerkin coefficient
α_h	Leak rate in the ESN
α_{learn}	Learning rate
α_{soft}	Soft update rate in DDPG
β	Heat release strength in the Rijke tube
β_{L}	Parameter of the Lorenz system
ϵ	Noise variable
η	Amplitude of Galerkin velocity mode
γ	Discount factor
Γ_{in}, Γ_{out}	Reflection coefficients at the inlet and outlet of the Rijke tube
κ	Wavenumber
λ	Tikhonov regularisation coefficient for Ridge regression
$\mathbf{\Pi}$	Acoustic pressure modes
$\mathbf{\Upsilon}$	Acoustic velocity modes
μ	Amplitude of Galerkin pressure mode
Ω	Spatial domain
ω	Angular frequency
$\phi^{(l)}$	Activation function of the l^{th} layer
π	Control policy
Ψ	Galerkin mode

ρ	Density
ρ_{L}	Parameter of the Lorenz system
ρ_h	Spectral radius of reservoir matrix
ρ_I	Inflation factor in EnKF
σ	Standard deviation
σ_{L}	Parameter of the Lorenz system
τ	Time delay in the Rijke tube
$\boldsymbol{\theta}$	System's parameter vector
$\boldsymbol{\xi}_\theta$	Scaling factor vector for parameters in the PA-ESN
ξ	Scaling factor for the sine activation
ξ_{in}	Scaling factor for input weight matrix in the ESN
ζ	Damping mode in the Rijke tube

Other functions and operators

$[\cdot\,;\,\cdot]$	Vertical concatenation
$\dot{(\cdot)}$	Time derivative
i	Imaginary unit
$\mathcal{J}$	Objective functional
$\mathcal{L}$	Loss function
$\mathcal{N}$	Normal distribution
$\mathcal{U}$	Uniform distribution
$\mathscr{F}$	Fourier transform
$\boldsymbol{\mathcal{F}}$	PDE operator
$\boldsymbol{\mathcal{M}}$	Observation operator acting in the space domain
$\tilde{\mathcal{J}}$	Instantenous objective functional
$\tilde{\boldsymbol{\mathcal{F}}}$	PDE constraint
$(\cdot)'$	Fluctuation
$(\cdot)^\top$	Transpose
$\bar{(\cdot)}$	Mean

Chapter 1

Introduction

Predicting and controlling flow and acoustic systems can effectively improve the efficiency of transport and energy applications, mitigate their environmental impact, and unlock sustainable energy technologies (Gad-el-Hak, 2000; Bewley, 2001; Dowling and Stow, 2003; Dowling and Morgans, 2005; Lieuwen and Yang, 2006; Joslin et al., 2009; Brunton and Noack, 2015; Rowley and Dawson, 2017; Magri, 2019; Brunton et al., 2020; Magri et al., 2023). However, high-dimensionality, nonlinearity and chaotic dynamics render these tasks fundamentally challenging. The goals for controlling flow and acoustic systems are: (i) given a system in a steady state, improve its time-averaged performance, e.g., its energy efficiency; (ii) given a system in some state, drive it to a desired state, e.g., stabilise an unstable configuration; and (iii) given a system, prevent it from evolving into a catastrophic regime, e.g., structural failure. In general, control can be achieved in two primary approaches; passive and active control. Passive control refers to static changes, often structural or topological, applied to the system, while active control is applied dynamically, where the control inputs are often determined from the feedback of observations. These observations are usually sensor measurements, which are noisy and may contain only partial information about the state.

Depending on the application, either passive or active control mechanisms might be preferred; but at the heart of both approaches lies an optimisation problem, as the control goals can be mathematically formulated as optimisation objectives constrained by the governing dynamics of the system. This formulation traditionally requires a model. On the one hand, models of fluid systems based on first principles contain valuable physical knowledge, are interpretable and generalise across domains. On the other hand, (i) accurate models capturing multiple scales are high dimensional, and are, therefore, computationally expensive for real-time prediction and control; (ii) the governing equations, initial and boundary conditions may be partially known or unknown; and (iii) the dynamics can be unpredictable beyond short time horizons (this is the case with chaotic systems). A separate challenge arises in multi-physics systems, where the interactions of multiple physical phenomena need to be modelled. Therefore, control is tightly knit with the tasks of reduced-order modelling, i.e., building approximate yet accurate models with lower dimensionality than the original system, system identification, i.e., learning a model of the system's dynamics from data, and state estimation, i.e., the reconstruction of the full state from observations (e.g., sensor

measurements).

Recent advances in machine learning (ML) have enabled the modelling, reconstruction, and control of fluid systems directly from data (Brunton et al., 2020). However, several critical questions remain: (1) Can ML algorithms scale to real-world engineering systems, demonstrating *robustness* to uncertainties in the data and in the model? (2) Do they offer meaningful physical insight into complex problems, giving *interpretability* to their decisions? (3) Do they extrapolate to unseen settings, thereby ensuring *generalisability* beyond the training data? Scientific machine learning (SciML), sitting at the intersection of physics-based and data-driven modelling, aims to answer these questions and develop ML models that are both accurate and physically grounded as well as computationally efficient (Magri, 2020).

1.1 Modelling and reconstruction

1.1.1 Modelling the temporal evolution of dynamics

Governed by partial differential equations (PDEs), models of fluid systems are traditionally based on first principles and are typically numerically integrated following a spatial discretisation scheme, which casts them as ordinary differential equations (ODEs). Whilst direct numerical simulations offer high accuracy, their computational cost can be prohibitively expensive due to the requirements of fine spatiotemporal resolution. A common approach to reduced-order modelling is to project the infinite dimensional dynamics onto a lower-dimensional subspace using a truncated basis. The truncated basis can be analytically derived such as Fourier, or data driven such as proper orthogonal decomposition (POD) modes (Sirovich, 1987; Holmes et al., 1996), among many others. If the PDEs are known, then a reduced-order model for the time evolution of the modes can be obtained by Galerkin projection (Holmes et al., 1996; Rowley and Dawson, 2017). A nonlinear reduced basis that is not necessarily orthogonal can be identified using autoencoders, which are neural network architectures that learn an identity mapping following a compression step (e.g., Fukami et al., 2020, 2021b; Racca et al., 2023; Mo et al., 2024; Özalp and Magri, 2025).

Alternatively, data-driven models that propagate the system's dynamics forward in time can be directly identified from data. The learning task is to obtain a mapping of the system's state from one time step to the next, where the input can be the full flow state, observations or a reduced representation. In nonlinear system identification, these methods include symbolic regression techniques such as SINDy (Brunton et al., 2016) which identifies interpretable sparse expressions for the governing equations based on a predefined library of candidate terms. Neural networks offer greater flexibility to infer unknown nonlinearities in the data, though the mapping function itself is less interpretable. In particular, autoregressive deep feedforward neural networks (FNNs) have been trained to act as surrogate models of otherwise computationally expensive simulations, which is for example useful in the solution of inverse problems (e.g. Mo et al., 2019; Xiao et al., 2021a).

Recurrent neural networks (RNNs) are specialised architectures for processing sequential data via a hidden state that retains a memory of past information. The recurrent connections in the architecture enable the learning of temporal relationships from data. For this reason, they are well-suited for the prediction of time series data from dynamical systems. RNN architectures can be broadly

categorised in two groups (Vlachas et al., 2020): those trained via backpropagation through time, such as the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the gated recurrent unit (GRU) (Cho et al., 2014); and those trained via reservoir computing, such as the echo state network (ESN) (Jaeger and Haas, 2004). The LSTM has been designed with gates that decide which information to remember and which to forget, enabling it to learn long-term temporal dependencies without running into the vanishing gradient problem. The GRU has been developed as a more computationally efficient alternative to the LSTM by requiring fewer gates and parameters. The ESN operates via a high-dimensional nonlinear reservoir state, in which the connections (weights) between the different state variables are sparse, randomly generated and not trained. The training involves only the ℓ_2-norm regularised regression of a linear readout layer from the reservoir state to the output, which has a closed-form solution. Compared to other architectures, the training of the ESN is computationally more efficient because it does not require backpropagation.

RNNs have been widely applied for a variety of tasks in the prediction and analysis of dynamical systems from data. LSTMs (Vlachas et al., 2018, 2020; Özalp et al., 2023), GRUs (Vlachas et al., 2020), and ESNs (Pathak et al., 2018; Doan et al., 2020a; Racca and Magri, 2021) have been shown to successfully make time-accurate short-term forecasts of chaotic systems. In RNNs, the hidden state (or the reservoir state in ESNs) evolves dynamically according to an update rule that depends on the current hidden state and the input (the data or the observables). (Once trained, RNNs can be run autonomously, depending only on the hidden state.) This state-space formulation allows RNNs to be framed as discrete-time dynamical systems and analysed using dynamical systems theory (e.g., Margazoglou and Magri, 2023b). Consequently, beyond short-term accuracy, RNNs have also been shown to infer ergodic properties, such as Lyapunov exponents, and long-term statistics of chaotic (e.g., the Lorenz 63, the Lorenz 96 systems, and the Kuramoto-Sivashinsky equation) and turbulent systems (Pathak et al., 2017; Srinivasan et al., 2019; Vlachas et al., 2020; Heyder and Schumacher, 2021; Huhn and Magri, 2022; Özalp et al., 2023; Margazoglou and Magri, 2023b; Özalp and Magri, 2025). For the prediction of high dimensional systems in fluid dynamics, LSTMs (Wang et al., 2018; Reddy et al., 2019; Hasegawa et al., 2020) and ESNs (Heyder and Schumacher, 2021; Racca et al., 2023; Özalp and Magri, 2025) have been coupled with dimensionality reduction techniques such as POD (e.g., Heyder and Schumacher, 2021) and autoencoders (e.g., Racca et al., 2023), in which they propagate the dynamics in the resulting low-dimensional space. The ESNs have been extended to infer parameterised dynamics including bifurcations to fixed points known as amplitude death (Xiao et al., 2021b) and multi-stable regimes including chaos (Roy et al., 2022).

Recently, transformers based on the self-attention mechanism (Vaswani et al., 2017) have emerged as a successful alternative to recurrent architectures in sequence modelling and computer vision tasks. Whilst RNNs process sequences step by step, transformers operate on the entire sequence in parallel, which allows them to be trained faster and can help them capture longer-term dependencies and multi-scale behaviour in physical data (Geneva and Zabaras, 2022; Heidenreich et al., 2024). However, they require explicit encoding of the sequence order, can be computationally heavy for long sequences at inference time, and may not yield to dynamical systems analysis as naturally as RNNs. Temporal prediction capability of transformers have been validated for benchmark systems such as the Lorenz system and the Kuramoto-Sivashinsky equation (Geneva and Zabaras, 2022; Heidenreich et al., 2024), and they have been utilised for the reduced order modelling of

higher-dimensional fluid flows (Solera-Rico et al., 2024; Shokar et al., 2024). Additionally, cross-attention has been exploited for the inference of parameterised dynamics in transient and periodic regimes (Zighed et al., 2025).

1.1.2 Reconstruction from partial state information

The measurements of the full flow state is often not available in experiments, in which case we have observations that contain the partial state information. The flow reconstruction task is then to estimate the full state from such observations. For computational applications, this can encompass the super-resolution task, where high-resolution fields are determined from low-resolution data in a cheap way. Depending on the available data and the assumptions on the system's model, a variety of methods have been developed. A regression method relying on a modal decomposition can infer the contribution of the individual modes, i.e., the temporal coefficients, which can be performed with a least squares or a sparsity promoting Lasso (Callaham et al., 2019). Neural networks that map from low-resolution to high-resolution flow fields have also been trained for flow reconstruction in a fully data-driven approach (e.g., Fukami et al., 2021a).

When a dynamical model of the system is available, data assimilation (DA) methods optimally combine the observations with the prior prediction of the model based on their respective uncertainties to estimate the full state. On the one hand, variational methods based on the adjoint method processes an entire assimilation window with all the observations in it, and provide valuable sensitivity information (e.g., Zaki and Wang, 2021). On the other hand, sequential methods incorporate observations into the model as they become available, updating its predictions. Bypassing the need to store and post-process batches of data makes sequential DA desirable for real-time state estimation and control. Extensions to the Kalman filter allow handling nonlinear system dynamics and non-Gaussian uncertainties, including the extended Kalman filter (EKF), particle filter (PF), and ensemble-based methods such as the ensemble Kalman filter (EnKF) (Evensen, 2009). EnKF approximates the state covariance with an ensemble typically with a smaller number of members than the state dimension which makes it scale well to high dimensional systems whilst avoiding computation of the system's Jacobian. Because it assumes the underlying probability distributions are Gaussian, which is valid when the updates are frequent enough, it is more computationally efficient compared to the PF.

The EnKF assumes that the system's model is unbiased, which for example in the case of low-order models may not be realistic due to modelling uncertainties, and the bias that has not been accounted for can lead to suboptimal updates (e.g. Dee, 2005). ML approaches have been developed to learn surrogate models that close the low-order model and correct the biased state estimates. A bias-aware EnKF framework that infers the model bias adaptively with an ESN was proposed by Nóvoa and Magri (2022), where the bias estimate has been regularised to have small norm in Nóvoa et al. (2024a). A review on how ML and DA approaches can be unified can be found in Cheng et al. (2023).

1.1.3 Physics-constrained machine learning

Similarly to DA, when modelling and reconstructing dynamics from data, constraining prior knowledge into ML methods can significantly improve prediction, robustness and generalisability (e.g., Duraisamy et al., 2019; Brunton et al., 2020; Karniadakis et al., 2021; Doan et al., 2021). On the one hand, constraints that are imposed in the loss function as penalty terms, which act during training, are referred to as "soft constraints". Soft constraints can include the governing PDEs derived from conservation laws, initial and boundary conditions (Lagaris et al., 1998). Deep feedforward neural networks (FNNs) with soft-constrained physics information, coined as physics-informed neural networks (PINNs) (Raissi et al., 2019a), have been employed to infer flow fields from synthetic data from prototypical flows (Raissi et al., 2019b, 2020), from puffing pool fires (Sitte and Doan, 2022), experimental data from a flow over an espresso cup (Cai et al., 2021), and clinical MRI data (Kissas et al., 2020), to name only a few. Beyond PINNs, physics information has enabled super-resolution tasks without high-resolution labels in deep FNNs (Fathi et al., 2020; Wang et al., 2022; Eivazi et al., 2024) and in convolutional neural networks (Gao et al., 2021; Kelshaw and Magri, 2024; Mo and Magri, 2025; Page, 2025). Because experimental data is noisy and sparse, a probabilistic formulation was proposed in the form of physics-constrained Bayesian neural networks, in which the governing equations are imposed via the likelihood term (Sun and Wang, 2020). In ESNs, constraining the governing equations in the loss term has enabled reconstruction of unmeasured state variables and accurate forecasting of the dynamics of chaotic systems (Doan et al., 2020b,a; Özalp et al., 2023).

On the other hand, constraints that are imposed in the architecture (as opposed to the training) are referred to as "hard constraints". Hard constraints span the function space in which physical solutions live, i.e., they create an *inductive bias* (Goyal and Bengio, 2022). Known PDEs (Chen et al., 2021; Xu and Darve, 2022; Mohan et al., 2023), invariances (Ling et al., 2016), Dirichlet boundary conditions (Lagaris et al., 1998), and periodic boundary conditions (Zhang et al., 2020; Dong and Ni, 2021; Kelshaw and Magri, 2024) have been incorporated in the architecture of neural networks as hard constraints. Raynaud et al. (2022) developed modal PINNs for reconstruction of periodic flows, in which the architecture fixes periodicity with a Fourier decomposition and modal shapes are discovered from sparse data with the aid of physics-informed loss.

1.2 Optimisation and control

1.2.1 Equation-based optimisation and control

Optimal control provides a framework for designing control strategies that maximise or minimise certain objectives constrained by the governing dynamics of the system and other constraints such as safety criteria. For discrete-time systems, the optimal control problem can be solved by the Bellman equation, which provides the optimal policy and value function for all time and for all possible states, and is the basis of Dynamic Programming (DP) (Bertsekas, 2007). In continuous time, the analogous formulation is the Hamilton-Jacobi-Bellman (HJB) equation. For linear time invariant (LTI) systems, or in fluid dynamics the linearised flow equations (Kim and Bewley, 2007), with quadratic objective functions, the optimal control law is the state-feedback determined by the Linear Quadratic Regulator (LQR), which is the global optimum for this convex problem.

However, finding the global optimum is challenging because the temporal dynamics can be nonlinear, time-delayed, and chaotic, which makes the objective function nonconvex. Furthermore, the solution to the Bellman and HJB equations become computationally intractable for high-dimensional systems. A practical approach is to solve the optimisation problem by gradient descent, in which the objective function is optimised along the gradient direction until a local optimum is found. The gradient (also known as sensitivity) is the first-order information about how the quantity of interest changes with the system's parameters (including control inputs or design parameters), or the system's initial conditions. Central to gradient-based optimisation methods are adjoint methods, which are obtained by reformulating the optimal control problem via the Lagrangian formalism (this is also known as the Pontryagin's Maximum Principle). Adjoint methods enable the accurate computation of the gradient with a computational cost that is not affected by the number of parameters. For example, if the system has N parameters to be optimised, adjoint methods compute the gradient of an objective functional with respect to all N parameters with only one calculation. This is in contrast with tangent linear or finite difference methods, which would require N computations (e.g., Giles and Pierce, 2000; Gunzburger, 2002; Giannetti and Luchini, 2007; Sipp et al., 2010; Luchini and Bottaro, 2014; Magri, 2019).

In the implementation of adjoint equations, there is a distinction depending on the order of discretisation and differentiation. In the continuous adjoint approach, the adjoint of the governing PDEs is first obtained via variations, and then numerically discretised. The reverse approach is the discrete adjoint, in which the PDE is first discretised, and then the adjoint of the discretised system of equations is derived (Giles and Pierce, 2000). Whilst the continuous adjoint provides more physical insight and can be discretised later with any scheme, the discrete adjoint produces more accurate gradients that are compatible with the direct solution (Giles and Pierce, 2000; Magri, 2019). The optimisation of a quantity of interest of a nonlinear system with adjoint methods requires, in general, six ingredients:

(i) the system's nonlinear governing equations are integrated forward in time to obtain the baseline (direct) solution;

(ii) the nonlinear equations are linearised around the direct solution to obtain the Jacobian of the system;

(iii) the adjoint equations are derived from the Jacobian;

(iv) the adjoint equations are initialised at the end of the direct solution, and are integrated backwards in time (because in nonlinear systems the Jacobian depends on the direct solution, strategies such as checkpointing become often necessary to decrease the memory requirements (Griewank and Walther, 2000));

(v) at the end of the adjoint integration, the gradient of the objective function to all the design parameters of interest can be evaluated by an inner product;

(vi) the gradient is embedded in a gradient-based optimisation scheme to find a local optimum of the objective function.

This process is also known as the direct-adjoint looping.

On the one hand, in finite difference or tangent linear approaches, the computational cost increases linearly with the number of parameters because the sensitivity calculation needs to be performed for each parameter. On the other hand, the adjoint method computes the gradient of the objective with respect to all parameters with a single operation, which makes it computationally cheaper when the number of parameters is larger than the number of objective functionals (e.g., Giles and Pierce, 2000). For this reason, adjoint methods have found extensive application in design optimisation, and both linear and nonlinear sensitivity analysis of fluid dynamics applications as reviewed by Luchini and Bottaro (2014). This was pioneered by the works of Pironneau (1974) for drag minimisation, and Jameson (1988); Jameson et al. (1998); Jameson (1999) for aerodynamic shape optimisation. In flow control, the direct-adjoint looping was embedded in the active feedback control (model predictive control) of turbulence for relaminarisation and drag reduction (Bewley et al., 2001); in the mixing optimisation of a passive scalar in the two-dimensional Poiseuille flow (Foures et al., 2014); and in the suppression of bypass transition in a zero-pressure-gradient flat-plate boundary layer (Xiao and Papadakis, 2017). In reacting flows, the direct-adjoint looping enabled the computation of the optimal location for the ignition of a laminar non-premixed flame that maximises the heat release (Qadri et al., 2021); and the sensitivity of ignition in turbulent combustion (Capecelatro et al., 2019).

Adjoint-based methods require the system's Jacobian. The difficulty of deriving the Jacobian from the system's equations, which may be high dimensional and nonlinear, has motivated the development of algorithms that rely on automatic differentiation (e.g., Mohammadi and Mohammadi, 1997). In aerodynamic shape optimisation, deep neural networks (e.g., Zhang et al., 2021; Renganathan et al., 2021; Wang et al., 2024), convolutional neural networks (e.g., Liao et al., 2021) and deep neural operators (Shukla et al., 2024) have been employed as surrogate models to exploit their computational efficiency and automatic differentiation capability. In inverse modelling, Xiao et al. (2021a) uses an autoregressive deep FNN as a surrogate model of two-dimensional subsurface flow to estimate space-dependent parameters.

1.2.2 Data-driven control strategies

Difficulty of modelling in general has led to data-driven control approaches, which include sequential identification of a model from data and model-based control, or model-free paradigms such as reinforcement learning (RL), which discovers optimal policies through direct interaction with the true system, i.e., the environment in RL terminology. In the RL paradigm, an agent learns an optimal policy by interacting with an environment with the goal of maximising cumulative rewards it receives over time (Sutton and Barto, 2018). In fact, RL can be regarded as a data-driven solution to the DP problem based on the Monte-Carlo sampling of trajectories when the model of the environment is unknown or intractable (Sutton and Barto, 2018; Semeraro, 2024). Owing to these advantages, RL has gained traction in flow control problems (e.g., Vignon et al., 2023). In chaotic dynamics that are relevant to turbulence control, Bucci et al. (2019) showed that RL can discover control policies that stabilise the Kuramoto–Sivashinsky (KS) equation around its unstable equilibria. In the context of drag reduction, Rabault et al. (2019) applied RL-based control to the flow behind a two-dimensional bluff body in a laminar regime, where the RL agent has access to pressure data in the wake and near the body from direct interactions with simulations. Li and

Zhang (2022) studied the global linear stability and adjoint sensitivity of the confined cylinder wake and guided the RL agent with this information, in which they show that the RL-based control is most effective when the sensor probes are placed in the wavemaker region, which is the region most sensitive to perturbations.

In experimental settings, full state measurements are rarely available and the RL agent can only access noisy and partial observations. If partial observability is not properly accounted for, model-free RL methods can struggle, often leading to unstable or suboptimal policies (Kaelbling et al., 1998). Partial observability has been addressed by the addition of memory (Bertsekas, 2007; Sutton and Barto, 2018; Bucci et al., 2019). Xia et al. (2024) achieved drag reduction of the two-dimensional bluff body from sensor measurements only on the body by the augmentation of the observation history. Memory can also be captured in the architecture with RNNs such as LSTMs (Verma et al., 2018), ESNs (Hart et al., 2021), or using the attention mechanism (Weissenbacher et al., 2025).

Model-based RL approaches have been initially developed to counteract the issues of poor sample efficiency of model-free RL methods. Here, a model of the environment is learned through the interactions of the agent, which is then employed in the background to generate more samples of the environment. The Dyna algorithm uses experiences from both the environment and the model for training (Sutton, 1991). In a similar vein to model predictive control, the model can also be exploited for planning over finite look-ahead horizons, updating the agent according to the predicted future dynamics (e.g., Nagabandi et al., 2018; Racanière et al., 2017; Hafner et al., 2019). Building on this synergy, hybrid approaches have emerged that combine the strengths of model-based and model-free RL. Schena et al. (2024) proposed a hybrid framework that switches between a model-free RL policy and model-based policy obtained via adjoint-based optimisation, where the model's parameters are identified through DA. To reduce the computational cost of RL training in high-dimensional environments, Zeng et al. (2022) developed a reduced-order model-based strategy in which an autoencoder coupled with a neural ODE (Chen et al., 2018) learns a lower-dimensional representation of the dynamics, which was showcased for the control of the spatiotemporally chaotic KS equation (Zeng et al., 2022) and the turbulent Couette flow (Linot et al., 2023).

To improve data-driven models, physics knowledge has also been incorporated into RL. In model-free RL, Zeng and Graham (2021) developed a symmetry-aware architecture for the control of KS equation, which enhanced the sample efficiency and control performance of the RL agent. In model-based RL, an agent was trained with a physics-informed LSTM in Dyna style, which outperforms the non-informed and model-free agents (Liu and Wang, 2021).

1.3 Thesis objectives

The overarching goal of this thesis is to develop SciML methods for the prediction and control of nonlinear PDEs encountered in flow and acoustics applications. The objectives of this thesis are:

1. To design hard-constrained neural network architectures that embed the physical structure to improve generalisability and computational efficiency in modelling thermoacoustic systems. Specifically:

   (a) To predict thermoacoustic oscillations in time.

(b) To reconstruct full-state pressure and velocity fields from limited sensor data.

(c) To infer boundary conditions when they are not ideal.

2. To infer the parameterised dynamics and bifurcations of nonlinear systems from data with a parameter-aware model. In combination with Objective 1, to extend it to time-delayed thermoacoustic systems with a hard-constrained architecture.

3. To develop a data-driven framework for adjoint sensivity analysis to infer sensitivities to the system's parameters and initial conditions from data.

4. To perform gradient-based optimisation based on the data-driven gradients to suppress a thermoacoustic instability.

5. To develop a predictive data-driven model for systems with control inputs.

6. To enable RL-based control of chaotic systems under partial and noisy observability, focusing on spatiotemporally chaotic systems such as the KS equation.

1.4 Thesis structure

The work of this thesis can be found in three journal publications and three peer-reviewed conference proceedings, listed in chronological order:

(i) Ozan, D. E. and Magri, L. (2023b). Physics-aware learning of nonlinear limit cycles and adjoint limit cycles. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(6):1191–1199,

(ii) Ozan, D. E. and Magri, L. (2023a). Hard-constrained neural networks for modeling nonlinear acoustics. *Physical Review Fluids*, 8(10):103201,

(iii) Ozan, D. E. and Magri, L. (2024a). Adjoint Sensitivities of Chaotic Flows Without Adjoint Solvers: A Data-Driven Approach. In *Lecture Notes in Computer Science*, pages 345–352. Springer Nature Switzerland, Cham,

(iv) Ozan, D. E. and Magri, L. (2024b). Data-driven computation of adjoint sensitivities without adjoint solvers: An application to thermoacoustics. *Physical Review Fluids*, 9(10):103902,

(v) Ozan, D. E., Nóvoa, A., and Magri, L. (2025a). Data-Assimilated Model-Based Reinforcement Learning for Partially Observed Chaotic Flows. In *Lecture Notes in Computer Science*, pages 65–72. Springer Nature Switzerland, Cham,

(vi) Ozan, D. E., Nóvoa, A., Rigas, G., and Magri, L. (2025b). Data-assimilated model-informed reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 481(2327):20250476.

This thesis is structured as follows:

- In Chapter 2, dynamical systems, their modelling and optimisation are laid out.

- In Chapter 3, the characteristics of nonlinear systems are introduced, and the prototypical nonlinear systems that are studied in this thesis are presented. These systems are the Lorenz system, the Rijke tube, and the KS equation. A special focus is given to thermoacoustic systems captured by the Rijke tube, which are multi-physics systems and show rich nonlinear phenomena. Thermoacoustic instabilities are of interest to power generation and propulsion applications.
- In Chapter 4, deep FNNs that constrain the acoustics physics in both the architecture and the loss are developed. Inspired by the Galerkin decomposition, Galerkin neural networks that constrain the separation of time and space in its architecture are designed. The networks are validated in twin experiments and against higher-fidelity data from the Rijke tube.
- In Chapter 5, it is shown that a parameter-aware ESN infers the parameterised dynamics of the Lorenz system and the Rijke tube. Based on this architecture, a data-driven framework for the computation of adjoint sentivities is proposed. Using the data-driven direct-adjoint looping, gradient-based optimisation to suppress a thermoacoustic instability is performed.
- In Chapter 6, a framework, data-assimilated model-informed reinforcement learning (DA-MIRL), is developed to tackle control of partially observable chaotic flows. The framework is demonstrated for the stabilisation of the KS equation for varying conditions reflecting aspects of experimental settings.

Chapter 2

Modelling and optimisation of dynamical systems

In this chapter, the background to study and optimise nonlinear partial differential equations (PDEs) is provided. In § 2.1, a dynamical system is defined with an ordinary differential equation (ODE) in continuous time (physical systems), and with a difference equation in discrete time (e.g., echo state networks), which are parameterised with the system's parameters and can be forced by external inputs, i.e., actuations. In § 2.2, PDEs are cast as ODEs following a Galerkin projection onto a finite-dimensional set of basis. Throughout the thesis, synthetic sensor data is generated from simulations of different PDEs. In § 2.3, an observation operator that maps the system's state to observables is defined. For optimisation problems, the next ingredient after the system is the objective functional. In § 2.4, adjoint sensitivity analysis is introduced for continuous-time systems, which is central to gradient-based optimisation.

2.1 Definition of a dynamical system

We consider continuous-time dynamical systems in the form of

$$\dot{\boldsymbol{s}} = \boldsymbol{F}(\boldsymbol{s}(t), \boldsymbol{\theta}, \boldsymbol{a}(t)), \tag{2.1}$$

where $t \in [0, \infty)$ denotes time, $\boldsymbol{s} \in \mathbb{R}^{n_s}$ is the state vector, $\dot{\boldsymbol{s}}$ is its time derivative $d\boldsymbol{s}/dt$, $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ contains the system's parameters, $\boldsymbol{a} \in \mathbb{R}^{n_a}$ contains control inputs, or actions, that are dynamically varied, and $\boldsymbol{F} : \mathbb{R}^{n_s} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_s}$ is the deterministic operator that describes the system's evolution. We assume that $\boldsymbol{F}$ is smooth so that the Jacobian of the system $\partial \boldsymbol{F}/\partial \boldsymbol{s}$ exists and can be used for gradient-based sensitivity analysis (§ 2.4). In control problems with a continuous control input space, the control input can be parameterised using a state-feedback policy (Åström and Murray, 2008)

$$\boldsymbol{a}(t) = \boldsymbol{\pi}(\boldsymbol{s}(t), \boldsymbol{\theta}^\pi), \tag{2.2}$$

where $\boldsymbol{\pi} : \mathbb{R}^{n_s} \times \mathbb{R}^{n_{\theta^\pi}} \to \mathbb{R}^{n_a}$ is the control policy and $\boldsymbol{\theta}^\pi$ are its parameters. Optimising over the parameters $\boldsymbol{\theta}^\pi$ makes the solution of the optimal control problem more tractable (e.g., Sutton and

Barto, 2018). We can therefore generalise (2.1) as

$$\dot{\boldsymbol{s}} = \boldsymbol{F}(\boldsymbol{s}(t), \boldsymbol{\theta}), \tag{2.3}$$

where $\boldsymbol{\theta}$ contains any parameters of interest, which allows us to derive a framework suitable for both passive and active control problems.

In discrete time, the dynamics are expressed as a map from the current state to the next

$$\boldsymbol{s}(t_{k+1}) = \boldsymbol{F}(\boldsymbol{s}(t_k), \boldsymbol{\theta}, \boldsymbol{a}(t_k)), \tag{2.4}$$

where $k \in \mathbb{Z}^{\geq 0}$ denotes the time step $t_k = k\Delta t$ with Δt being the time step size. For readability, we will denote $\boldsymbol{s}(k) \coloneqq \boldsymbol{s}(t_k)$ and $\boldsymbol{a}(k) \coloneqq \boldsymbol{a}(t_k)$.

2.2 Partial differential equations and Galerkin projection

Fluid systems are typically governed by partial differential equations (PDEs) of the form

$$\frac{\partial \boldsymbol{y}}{\partial t} = \boldsymbol{\mathcal{F}}(\boldsymbol{y}(\boldsymbol{x}, t), \boldsymbol{\theta}), \tag{2.5}$$

where $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^{n_d}$ is the spatial location, n_d is the space dimension, and Ω is the spatial domain. The vector $\boldsymbol{y}$ contains physical quantities such as pressure and velocity components that are defined over the space-time domain, i.e., $\boldsymbol{y}(x, t) : \Omega \times [0, \infty) \to \mathbb{R}^{n_y}$. The nonlinear operator is denoted by $\boldsymbol{\mathcal{F}}$. We can further define a residual,

$$\widetilde{\boldsymbol{\mathcal{F}}}\left(\boldsymbol{y}(\boldsymbol{x}, t), \boldsymbol{\theta}\right) \coloneqq \frac{\partial \boldsymbol{y}}{\partial t} - \boldsymbol{\mathcal{F}}(\boldsymbol{y}(\boldsymbol{x}, t), \boldsymbol{\theta}), \tag{2.6}$$

which quantifies how well a candidate solution $\boldsymbol{y}(\boldsymbol{x}, t)$ satisfies the governing PDE, i.e.,

$$\widetilde{\boldsymbol{\mathcal{F}}}(\boldsymbol{y}^\star(\boldsymbol{x}, t), \boldsymbol{\theta}) = \boldsymbol{0} \tag{2.7}$$

if $\boldsymbol{y}^\star(\boldsymbol{x}, t)$ is a solution to the PDE.

Separation of variables is a method for solving PDEs, most notably wave and heat equations (Olver, 2020). It seeks to find solutions to PDEs in a special form as the product of functions of the individual independent variables, e.g., time and single space coordinates (Olver, 2020). A weak solution to the nonlinear problem is provided by the Galerkin method, which approximates the solution to the PDE by projecting the equations onto a finite-dimensional subspace (Holmes et al., 1996). The solution is written as a linear combination of the basis functions that span this subspace,

$$y_i(\boldsymbol{x}, t) = \sum_{j=1}^{n_g} \alpha_j^i(t) \Psi_j^i(\boldsymbol{x}), \tag{2.8}$$

where $i = 1, 2, \ldots, n_y$ denotes the i^{th} element of $\boldsymbol{y}$, $\Psi_j^i(\boldsymbol{x})$ are the basis functions, or Galerkin modes, n_g is the number of Galerkin modes, and $\alpha_j^i(t)$ are the Galerkin coefficients. Projecting the PDEs (2.5) onto the Galerkin modes transforms the problem to a set of ODEs (2.3) that describe

the evolution of the Galerkin coefficients $\boldsymbol{\alpha}^i = [\alpha_1^i, \alpha_2^i, \ldots, \alpha_{n_g}^i]^\top$.

Orthogonal Galerkin modes, such as Fourier modes, guarantee that the approximation error is orthogonal to the space spanned by the finite number of modes retained. When the low-wavenumber modes contain most of the energy, the truncation error can be minimal, which makes Galerkin methods using a Fourier basis a suitable choice for the modelling of such problems.

2.3 Observations

In experimental settings, we often do not have access to the full system state $\boldsymbol{s}$. Instead, we acquire measurements from sensors that provide partial and potentially noisy information about the underlying state, i.e., observations. In the state space formulation with the state vector $\boldsymbol{s} \in \mathbb{R}^{n_s}$ (2.1), we define observations as

$$\boldsymbol{o}(t) = \boldsymbol{M}(\boldsymbol{s}(t)) + \boldsymbol{\epsilon}(t), \tag{2.9}$$

where $\boldsymbol{o} \in \mathbb{R}^{n_o}$ is the observation vector, $\boldsymbol{\epsilon} \in \mathbb{R}^{n_o}$ is the zero-mean additive measurement noise (unbiased), which we assume is sampled from a Gaussian distribution, and $\boldsymbol{M} : \mathbb{R}^{n_s} \to \mathbb{R}^{n_o}$ is the observation operator. For observations of a subset of the state, $\boldsymbol{M} \in \mathbb{R}^{n_o \times n_s}$ is the linear operator

$$\boldsymbol{M}_{ij} = \begin{cases} 1 & \text{if } s_j \text{ is observed in the } i^{\text{th}} \text{ measurement,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

If the state is fully observed, then $\boldsymbol{M}$ is the identity matrix $\boldsymbol{I}_{n_s \times n_s}$. When considering PDEs with the state $\boldsymbol{y}(\boldsymbol{x}, t) \in \mathbb{R}^{n_y}$ (2.5), we define observations as

$$\boldsymbol{o}(t) = \boldsymbol{\mathcal{M}}(\boldsymbol{y}(\boldsymbol{x}, t)) + \boldsymbol{\epsilon}(t), \tag{2.11}$$

where $\mathcal{M}$ is the observation operator. Given independent observations of $y_i \in \boldsymbol{y}$, we have

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \boldsymbol{\mathcal{M}}_1 & 0 & \ldots & 0 \\ 0 & \boldsymbol{\mathcal{M}}_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \boldsymbol{\mathcal{M}}_{n_y} \end{bmatrix}, \tag{2.12}$$

where $\boldsymbol{\mathcal{M}}_i$ is the block operator acting on y_i. We further consider pointwise measurements of variable y_i from n_o^i sensors at locations $\boldsymbol{X}_o^i \in \mathbb{R}^{n_o^i \times d}$

$$\boldsymbol{X}_o^i = \begin{bmatrix} {\boldsymbol{x}_{o,1}^i}^\top \\ {\boldsymbol{x}_{o,2}^i}^\top \\ \vdots \\ {\boldsymbol{x}_{o,n_o^i}^i}^\top \end{bmatrix}.$$

The observation operator becomes

$$\boldsymbol{\mathcal{M}}_i(y_i(\boldsymbol{x},t)) = \boldsymbol{y}_i(\boldsymbol{X}_o^i,t) = \begin{bmatrix} y_i(\boldsymbol{x}_{o,1}^i,t) \\ y_i(\boldsymbol{x}_{o,2}^i,t) \\ \vdots \\ y_i(\boldsymbol{x}_{o,n_o^i}^i,t), \end{bmatrix} \tag{2.13}$$

where $\sum_i n_o^i = n_o$. Typically, solutions of PDEs are spatially discretised, e.g., finite difference or Galerkin methods, such that the observation operator $\mathcal{M}$ is transformed to the state-space formulation (2.9).

Although observations (2.9) and (2.11) are defined in continuous time, in reality measurements are sampled at discrete times $t_k = k\Delta t_o$ with Δt_o being the sampling time.

2.4 Adjoint sensitivity

We wish to optimise an objective functional $\mathcal{J}$, which is a time-average of a functional $\tilde{\mathcal{J}}$

$$\mathcal{J} = \langle \tilde{\mathcal{J}} \rangle = \frac{1}{T}\int_0^T \tilde{\mathcal{J}}(\boldsymbol{s}(t),\boldsymbol{\theta})\, dt, \tag{2.14}$$

subjected to the system dynamics (2.3)

$$\widetilde{\boldsymbol{F}}\left(\dot{\boldsymbol{s}}(t),\boldsymbol{s}(t),\boldsymbol{\theta}\right) \coloneqq \dot{\boldsymbol{s}}(t) - \boldsymbol{F}(\boldsymbol{s}(t),\boldsymbol{\theta}) = 0, \tag{2.15}$$

where $\boldsymbol{\theta}$ are the design or control parameters. Gradient-based optimisation requires the computation of the first-order derivative of $\mathcal{J}$ with respect to $\boldsymbol{\theta}$

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{d}{d\boldsymbol{\theta}}\left(\frac{1}{T}\int_0^T \tilde{\mathcal{J}}(\boldsymbol{s}(t),\boldsymbol{\theta})\, dt\right), \tag{2.16}$$

which is also known as the *sensitivity*. A straightforward approach to estimate sensitivity information is to apply a small perturbation ϵ to the parameters of interest $\theta_i \in \boldsymbol{\theta}$ and numerically compute the sensitivity using finite differences

$$\left.\frac{d\mathcal{J}}{d\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}} = \left.\frac{d\mathcal{J}}{d\theta_i}\right|_{\theta_i=\bar{\theta}_i} \approx \frac{\mathcal{J}|_{\theta_i=\bar{\theta}_i+\epsilon} - \mathcal{J}|_{\theta_i=\bar{\theta}_i}}{\epsilon} + O(\epsilon), \quad i = 1,2,\ldots,n_\theta, \tag{2.17}$$

which needs to be repeated for each $\theta_i \in \boldsymbol{\theta}$ to determine $d\mathcal{J}/d\boldsymbol{\theta}$. This means that the computational cost of the sensitivity increases linearly with the number of parameters. A more efficient approach is to use the adjoint equation. Analytically, we can express the gradient $d\mathcal{J}/d\boldsymbol{\theta}$ as

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{1}{T}\int_0^T \left(\frac{\partial\tilde{\mathcal{J}}}{\partial\boldsymbol{\theta}} + \frac{\partial\tilde{\mathcal{J}}}{\partial\boldsymbol{s}}\boldsymbol{Q}\right) dt, \tag{2.18a}$$

$$\dot{\boldsymbol{Q}} = \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{s}}\boldsymbol{Q}, \tag{2.18b}$$

$$\boldsymbol{Q}(0) = \boldsymbol{0}, \tag{2.18c}$$

where we define $\boldsymbol{Q} \coloneqq d\boldsymbol{s}/d\boldsymbol{\theta},\ \boldsymbol{Q} \in \mathbb{R}^{n_s \times n_\theta}$. We refer to solving the system of equations (2.18) as the tangent linear problem. In this case, the sensitivity can be computed by formulating the tangent linear system around a base trajectory and solving for $\boldsymbol{Q}$ in time. Whilst this approach eliminates the finite difference numerical error, it still suffers from a computational cost that is linear with the number of parameters as the dimension of $\boldsymbol{Q}$ grows with the dimension of $\boldsymbol{\theta}$.

Adjoint methods circumvent this issue. By introducing the Lagrangian of the objective (2.14) and the adjoint variables $\boldsymbol{q}^+ \in \mathbb{R}^{n_s}$ as the Lagrange multipliers of the constraint (2.15), the gradient $d\mathcal{J}/d\boldsymbol{\theta}$ can be rewritten as (e.g., Magri, 2019)

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{1}{T}\int_0^T \left(\frac{\partial \tilde{\mathcal{J}}}{\partial \boldsymbol{\theta}} + {\boldsymbol{q}^+}^\top \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}} \right) dt, \tag{2.19a}$$

$$\dot{\boldsymbol{q}}^+ = -\frac{\partial \tilde{\mathcal{J}}}{\partial \boldsymbol{s}}^\top - \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{s}}^\top \boldsymbol{q}^+, \tag{2.19b}$$

$$\boldsymbol{q}^+(T) = 0, \tag{2.19c}$$

where the adjoint variables $\boldsymbol{q}^+$ need to solved by integrating the adjoint system backwards in time. (The tangent linear approach is efficient when the number of objective functionals is greater than the number of parameters, whereas the adjoint approach is efficient when the number of parameters is greater the number of objective functionals. In most applications, the latter is a common case.)

The derivation of the adjoint equations (2.19) can be found in Magri (2019) among others. In this thesis, we derive the adjoint equations for echo state networks in § 5.3.1 (See Appendix § B.1 for the derivation of the adjoint of a general discrete-time system). Sensitivity to the initial conditions can be similarly formulated as shown in § 5.3.2. We further extend the sensitivity computations to time-delayed echo state networks in § 5.5.2 (See Appendix § B.2 for the derivation of the adjoint of a general discrete-time time-delayed system).

Chapter 3

Nonlinear dynamical systems

Nonlinear dynamical systems are ubiquitous in nature and engineering (e.g., Strogatz, 2019). In this chapter, the characteristics of nonlinear systems are introduced, and those systems which are studied in this thesis are presented. These systems are:

(i) The Lorenz 63 system (§ 3.3).
Modelling atmospheric phenomena, the Lorenz system is the prototypical example of chaos known for the so-called "butterfly effect". In § 5.4 we infer the parametric sensitivities of the Lorenz system from data.

(ii) The Rijke tube (§ 3.4).
Governed by multi-physics effects, the Rijke tube is a prototypical thermoacoustic system, where the acoustics interact with heat release in a positive feedback loop. In § 4.4-4.6, we reconstruct the state of the Rijke tube from partial and noisy observation. In § 5.5- 5.10, we infer parametric sensitivities of the Rijke tube from data and employ gradient-based optimization to suppress thermoacoustic instabilities.

(iii) The Kuramoto-Sivashinsky (KS) equation (§ 3.5).
Exhibiting spatiotemporal chaos, the KS equation shares similarities with the Navier-Stokes equations, and is, therefore, typically used as a testbed for studying spatiotemporal chaos that arises in turbulent flows. In § 6.3-6.6, we perform combined state estimation and active control of the KS equation from partial observations to stabilise its chaotic dynamics.

3.1 Nonlinearity, bifurcations and the route to chaos

Whilst autonomous linear systems are limited to exponentially decaying or growing trajectories (stable or unstable), or simple harmonic oscillations (neutrally stable), nonlinear systems exhibit a rich variety of phenomena, including multistability, bifurcations and aperiodic solutions. Typical dynamical regimes observed in nonlinear systems are:

(a) A *fixed point* is an equilibrium point such that $d\boldsymbol{s}/dt = 0$.

(b) A *limit cycle* is a closed trajectory in the phase space, i.e., a periodic orbit $\boldsymbol{s}(t) = \boldsymbol{s}(t+T)$ with T being the period. The frequency spectrum is composed of the harmonics of the fundamental

angular frequency $2\pi/T$. Unlike linear periodic oscillations, nonlinear limit cycles are either attracting (stable) or repelling (unstable) for nearby trajectories. The practical importance of limit cycles is that they are self-excited oscillations (without periodic forcing), which arise in many physical and engineering applications.

(c) A *quasiperiodic* solution is composed of two or more incommensurate frequencies, i.e., the ratio of the frequencies is irrational, and the attractor is topologically equivalent to a torus in the phase space.

(d) A *chaotic* solution exhibits aperiodic, seemingly random behaviour and exponential sensitivity to initial conditions. The sensitivity to the initial conditions makes chaotic systems unpredictable for long time intervals, as infinitesimal errors grow exponentially. In dissipative chaotic systems, the solution converges to an attractor with a fractal structure, which typically has a lower dimension than the original system. In the frequency domain, chaotic solutions are characterised by a broadband spectrum.

Lyapunov exponents quantify the rate of exponential divergence or convergence of infinitesimally close trajectories. The regimes (a-d) can also be classified according to their Lyapunov exponents (e.g., Huhn and Magri, 2020)

$$\left.\begin{array}{ll} \text{Chaotic} & \text{if } \lambda_1 > 0, \\ \text{Quasi-periodic} & \text{if } \lambda_1 = 0 \text{ and } \lambda_2 = 0, \\ \text{Limit cycle} & \text{if } \lambda_1 = 0 \text{ and } \lambda_2 < 0, \\ \text{Fixed point} & \text{if } \lambda_1 < 0 \end{array}\right\} \tag{3.1}$$

where $\lambda_1 \geq \lambda_2 \geq \dots$ are the Lyapunov exponents in descending order. The Lyapunov time (LT) is the time scale associated with the leading Lyapunov exponent, $LT = 1/\lambda_1$, and is an indicator of the predictability horizon of a chaotic system.

A *bifurcation* occurs when a small perturbation of a system's parameter causes a qualitative change in the system's long-term behaviour. This can involve the change of stability of an existing solution, or the birth or death of an invariant set. For example, the Hopf bifurcation refers to the birth or death of a limit cycle from a fixed point. The Hopf bifurcation point manifests itself in the linearised dynamics as the complex conjugate pair of eigenvalues with negative real part becoming purely imaginary. As the system's parameters are varied, nonlinear systems can undergo successive sequences of bifurcations that result in more geometrically complicated attractors and eventually lead to chaos (Eckmann and Ruelle, 1985). Examples of bifurcations and the routes to chaos will be discussed in the following sections for the different dynamical systems analysed in this thesis.

3.2 Sensitivities in chaotic systems

In many applications, we are interested in the sensitivity of the asymptotic time average (e.g., Huhn and Magri, 2020),

$$\mathcal{J}_\infty = \lim_{T\to\infty} \frac{1}{T} \int_0^T \tilde{\mathcal{J}}(\boldsymbol{s}(t), \boldsymbol{\theta})\, dt, \tag{3.2}$$

which is also known as the "climate sensitivity" (Lea et al., 2000; Thuburn, 2005). However, because chaotic systems are characterised by at least one positive Lyapunov exponent, the tangent linear (2.18) and the adjoint (2.19) systems are unstable and the solutions grow unboundedly as the integration time T increases (Lea et al., 2000). Consequently, for a chaotic attractor the operation of time averaging over $T \to \infty$ does not commute with the operation of differentiation (Thuburn, 2005)

$$\frac{d}{d\boldsymbol{\theta}}\left(\lim_{T\to\infty}\frac{1}{T}\int_0^T \tilde{\mathcal{J}}(\boldsymbol{s}(t),\boldsymbol{\theta})dt\right) \neq \lim_{T\to\infty}\frac{1}{T}\int_0^T \frac{d\tilde{\mathcal{J}}(\boldsymbol{s}(t),\boldsymbol{\theta})}{d\boldsymbol{\theta}}dt, \tag{3.3}$$

where the left-hand side corresponds to the finite-difference approach (2.17) and right-hand side corresponds to the tangent linear method (2.18). The operations do not commute because the finite difference does not uniformly converge to the first-order derivative for all integration times T.

If using a finite difference approach, the computation of the sensitivity requires very large T to obtain accurate statistics of the objective functional. On the other hand, the adjoint equations can be integrated over short-time intervals. A reasonable estimate of the climate sensitivity can be then obtained by averaging this computation over an ensemble of initial conditions (Lea et al., 2000)

$$\mathcal{J}_\infty \approx \frac{1}{n_{\text{en}}}\sum_{m=1}^{n_{\text{en}}}\frac{1}{T}\int_{t=0}^{T}\frac{d\tilde{\mathcal{J}}(\boldsymbol{s}_m(t),\boldsymbol{\theta})}{d\boldsymbol{\theta}}dt, \tag{3.4}$$

where $\boldsymbol{s}_m(t)$ is a trajectory in the ensemble and T is often in the order of one Lyapunov time of the system. The integration time affects the variance and bias of the sensitivity estimate (Chandramoorthy et al., 2017). In § 5.4 and § 5.9, we compute the climate sensitivities of chaotic systems by applying the ensemble adjoint method to echo state networks.

3.3 Chaos: the Lorenz system

The Lorenz system is a reduced-order model of atmospheric convection (Lorenz, 1963). Specifically, it models the two-dimensional fluid layer between two horizontal plates at two different temperatures,

$$\frac{dx_{\text{L}}}{dt} = \sigma_{\text{L}}(y_{\text{L}} - x_{\text{L}}), \tag{3.5a}$$

$$\frac{dy_{\text{L}}}{dt} = x_{\text{L}}(\rho_{\text{L}} - z_{\text{L}}) - y_{\text{L}}, \tag{3.5b}$$

$$\frac{dz_{\text{L}}}{dt} = x_{\text{L}}y_{\text{L}} - \beta_{\text{L}}z_{\text{L}}, \tag{3.5c}$$

where x_{L}, y_{L}, and z_{L} are proportional to the rate of convective motion, the temperature difference between the ascending and descending currents, and distortion of the vertical temperature profile from linearity, respectively. The system parameters, σ_{L} and ρ_{L} are proportional to the Prandtl number and Rayleigh number, respectively, and β_{L} is a geometric scaling. At $\sigma_{\text{L}} = 10$, $\rho_{\text{L}} = 28$, and $\beta_{\text{L}} = 8/3$, the system exhibits chaotic solutions (Lorenz, 1963) (We take these values to be the default parameters unless stated otherwise).

For $\rho_{\text{L}} < 1$, the system has one stable fixed point at the origin, which is the global attractor. A pitchfork bifurcation occurs at $\rho_{\text{L}} = 1$, where the origin becomes unstable, and two new stable

equilibrium points appear at $x_\mathrm{L} = y_\mathrm{L} = \pm\sqrt{\beta_\mathrm{L}(\rho_\mathrm{L} - 1)}$, $z_\mathrm{L} = \rho_\mathrm{L} - 1$. Denoting the origin as E_0 and the other two equilibria as E_1 and E_2, we observe that the trajectories that started in one half space are attracted to E_1 and the trajectories that started in the other half space are attracted to E_2. Trajectories that started on the stable manifold of the origin are attracted to the origin. At approximately $\rho_\mathrm{L} > 13.96$, the trajectories start crossing over to the other equilibrium point, which is a distinctive feature of the Lorenz system in the chaotic regimes as well. (This implies a homoclinic orbit at this parameter value.) The pair of equilibrium points become unstable if

$$\rho_\mathrm{L} > \sigma_\mathrm{L} \frac{\sigma_\mathrm{L} + \beta_\mathrm{L} + 3}{\sigma_\mathrm{L} - \beta_\mathrm{L} - 1},$$

in a subcritical Hopf bifurcation, which occurs at $\rho_\mathrm{L} \approx 24.73$ for the default set of system's parameters. A study of the bifurcations in the Lorenz system can be found in Sparrow (1982).

3.4 Multi-physics: Thermoacoustics and the Rijke tube

Thermoacoustic systems contain nonlinearities in the heat release model, which, when coupled with the acoustics in a positive feedback loop, can generate self-excited oscillations and rich nonlinear behaviours via bifurcations (e.g., Rayleigh, 1878; Dowling and Morgans, 2005; Juniper and Sujith, 2018; Magri, 2019; Magri et al., 2023). Because these oscillations can have detrimental effects on the system's structure and performance, their prediction and control are active areas of research, for example, in gas turbines (Lieuwen and Yang, 2006; Poinsot, 2017), and rocket engines (Culick and Kuentzmann, 2006).

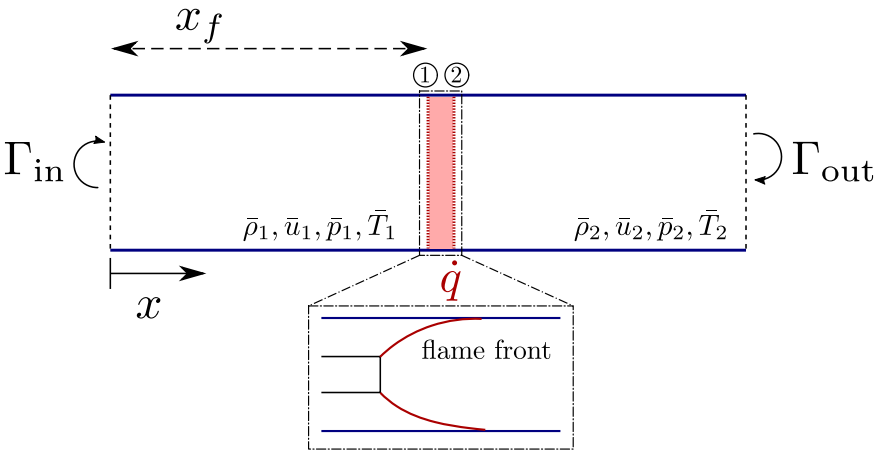


Figure 3.1: Schematic of the Rijke tube. A straight duct with open ends and a heat source, $\dot{q}_f$, located at x_f. The acoustic reflection coefficients at the inlet and outlet are Γ_in and Γ_out. The Rijke tube is simplified in § 3.4.3 with the assumptions; (i) zero Mach number, i.e., $\bar{u}_1 = \bar{u}_2 = 0$, (ii) no change in the mean flow variables before and after the flame e.g., $\bar{\rho}_1 = \bar{\rho}_2 = 1$, and (iii) ideal boundary conditions, i.e., $\Gamma_\mathrm{in} = \Gamma_\mathrm{out} = -1$. In this model, we do not solve for the flame, but model $\dot{q}_f$. The higher-fidelity model with a mean flow and a kinematic model for the flame is introduced in § 3.4.5.

3.4.1 The Rijke tube

We consider a thermoacoustic system composed of a straight duct of length $\tilde{L}$ with a compact heat source located at $\tilde{x} = \tilde{x}_f$, where $\tilde{(\cdot)}$ denotes a dimensional variable (Fig. 3.1). We make the following assumptions about the system: (i) the acoustics are one-dimensional, i.e., the tube is sufficiently longer than its diameter for the cut-on frequency to be high enough for longitudinal acoustics only to propagate; (ii) the heat release from the flame acts as a pointwise monopole source of sound (compact assumption); (iii) the mean flow is low Mach number and there is no entropy wave convection; and (iv) effects of viscosity and heat conduction are negligible (e.g., Magri, 2019).

The dynamics are governed by the dimensional equations derived from mass, momentum, and energy conservation. Modelling the heat source as a compact source results in two duct segments related by jump conditions that are enforced at the heat source location. For brevity, the suffices 1 and 2 will denote conditions before and after the flame, i.e, $\tilde{x} = \tilde{x}_{f,1}$ and $\tilde{x} = \tilde{x}_{f,2}$, respectively. The jump conditions are found from mass, momentum, and energy fluxes across the flame with the ideal gas law (e.g., Aguilar Perez, 2018). The governing equations and the jump conditions are linearised by assuming that the flow variables can be expressed as infinitesimal perturbations on top of a mean flow, i.e., $\tilde{(\cdot)} = \bar{\tilde{(\cdot)}} + \tilde{(\cdot)}'$, where $\bar{(\cdot)}$ denotes the steady mean-flow variable and $(\cdot)'$ denotes the unsteady infinitesimal perturbation. Under the low-Mach-number assumption, the acoustics are governed by conservation of mass, momentum and energy equations (e.g., Magri, 2019)

$$\frac{\partial \tilde{\rho}'}{\partial \tilde{t}} + \bar{\tilde{u}}\frac{\partial \tilde{\rho}'}{\partial \tilde{x}} + \bar{\tilde{\rho}}\frac{\partial \tilde{u}'}{\partial \tilde{x}} = 0, \tag{3.6a}$$

$$\bar{\tilde{\rho}}\frac{\partial \tilde{u}'}{\partial \tilde{t}} + \bar{\tilde{\rho}}\bar{\tilde{u}}\frac{\partial \tilde{u}'}{\partial \tilde{x}} + \frac{\partial \tilde{p}'}{\partial \tilde{x}} = 0, \tag{3.6b}$$

$$\frac{\partial \tilde{p}'}{\partial \tilde{t}} + \bar{\tilde{u}}\frac{\partial \tilde{p}'}{\partial \tilde{x}} + \gamma\bar{\tilde{p}}\frac{\partial \tilde{u}'}{\partial \tilde{x}} - (\gamma - 1)\frac{\tilde{\dot{q}}_f'}{\bar{\tilde{A}}}\delta(\tilde{x} - \tilde{x}_f) = 0, \tag{3.6c}$$

where $\tilde{\rho}$ is the density, $\tilde{u}$ is the velocity, $\tilde{p}$ is the pressure, $\tilde{\dot{q}}_f$ is the heat release rate, $\tilde{A}$ is the cross-sectional area of the duct, γ is the heat capacity ratio, and δ is the Dirac delta distribution. The jump conditions across the flame are (Dowling, 1999; Aguilar Perez, 2018)

$$\left[\tilde{A}\bar{\tilde{\rho}}\bar{\tilde{u}}\right]_1^2 = 0, \tag{3.7a}$$

$$\left[\tilde{A}\bar{\tilde{p}}'\right]_1^2 + \tilde{A}_1\bar{\tilde{u}}_1[\bar{\tilde{u}}]_1^2\tilde{\rho}_1' + \tilde{A}_1\bar{\tilde{\rho}}_1\bar{\tilde{u}}_1[\tilde{u}']_1^2 + \tilde{A}_1\bar{\tilde{\rho}}_1\tilde{u}_1'[\bar{\tilde{u}}]_1^2 - [\tilde{A}]_1^2\tilde{p}_1' = 0, \tag{3.7b}$$

$$\frac{\gamma}{(\gamma-1)}\left[\tilde{A}\bar{\tilde{p}}'\bar{\tilde{u}} + \tilde{A}\bar{\tilde{p}}\tilde{u}'\right]_1^2 + \frac{1}{2}\tilde{A}_1\bar{\tilde{u}}_1[\bar{\tilde{u}}^2]_1^2\tilde{\rho}_1' + \tilde{A}_1\bar{\tilde{u}}_1\left(\left(\frac{1}{2}\bar{\tilde{u}}_2^2 - \frac{3}{2}\bar{\tilde{u}}_1^2\right)\tilde{u}_1' + \bar{\tilde{u}}_1\bar{\tilde{u}}_2\tilde{u}_2'\right) = \tilde{\dot{q}}_f. \tag{3.7c}$$

The boundary conditions are determined by the acoustic reflection at the boundaries denoted by Γ_{in} at the inlet and Γ_{out} at the outlet.

3.4.2 Eigenmodes of the Rijke tube

A suitable eigenbasis for the thermoacoustic problem is provided by the natural acoustic eigenfunctions, because most of the acoustic energy is dominated by the first modes, and effects of mean flow or nonlinearity only slightly change the frequency and the eigenshapes. The core eigenfunctions of the thermoacoustic system with zero Mach number case are similar to a low Mach number case,

$Ma \lesssim 0.1$ (Magri and Juniper, 2014). Hence, to derive them, we assume a zero Mach number for simplification, whose dynamics are governed by the nondimensional equations (e.g., Magri and Juniper, 2014)

$$\bar{\rho}\frac{\partial u'}{\partial t} + \frac{\partial p'}{\partial x} = 0, \tag{3.8a}$$

$$\frac{\partial p'}{\partial t} + \frac{\partial u'}{\partial x} - \dot{q}'_f \delta(x - x_f) = 0, \tag{3.8b}$$

where the density is modelled as

$$\bar{\rho} = \begin{cases} \bar{\rho}_1 & \text{when } 0 \le x < x_f, \\ \bar{\rho}_2 & \text{when } x_f < x \le 1. \end{cases} \tag{3.9}$$

The dimensional variables have been scaled as, $x = \tilde{x}/\tilde{L}$, $t = \tilde{t}\tilde{\bar{c}}_{ref}/\tilde{L}$, where $\tilde{\bar{c}}_{ref}$ is the mean speed of sound; $u = \tilde{u}/\tilde{\bar{c}}_{ref}$; $\rho = \tilde{\rho}/\tilde{\bar{\rho}}_{ref}$; $p = \tilde{p}/(\tilde{\bar{\rho}}_{ref}\tilde{\bar{c}}^2_{ref})$; and $\dot{q}_f = \tilde{\dot{q}}_f(\gamma - 1)/(\tilde{\bar{\rho}}_{ref}\tilde{\bar{c}}^3_{ref})$. A physically-justified method to solve the set of PDEs (3.8) is to decompose pressure and velocity as

$$p'(x,t) = \sum_{j=1}^{n_g} \begin{cases} \mu_j(t)\Pi_j^{(1)}(x), & 0 \le x < x_f, \\ \mu_j(t)\Pi_j^{(2)}(x), & x_f < x \le 1, \end{cases} \tag{3.10a}$$

$$u'(x,t) = \sum_{j=1}^{n_g} \begin{cases} \eta_j(t)\Upsilon_j^{(1)}(x), & 0 \le x < x_f, \\ \eta_j(t)\Upsilon_j^{(2)}(x), & x_f < x \le 1, \end{cases} \tag{3.10b}$$

and project the equations onto the acoustic eigenfunctions found as below in the case of ideal boundary conditions, i.e., $p'(x=0) = p'(x=1) = 0$, (Magri and Juniper, 2014)

$$\Pi_j^{(1)}(x) = -\sin(\kappa_j\sqrt{\bar{\rho}_1}x), \quad \Pi_j^{(2)}(x) = -\left(\frac{\sin\gamma_j}{\sin\beta_j}\right)\sin(\kappa_j\sqrt{\bar{\rho}_2}(1-x)), \tag{3.11a}$$

$$\Upsilon_j^{(1)}(x) = \frac{1}{\sqrt{\bar{\rho}_1}}\cos(\kappa_j\sqrt{\bar{\rho}_1}x), \quad \Upsilon_j^{(2)}(x) = -\frac{1}{\sqrt{\bar{\rho}_2}}\left(\frac{\sin\gamma_j}{\sin\beta_j}\right)\cos(\kappa_j\sqrt{\bar{\rho}_2}(1-x)), \tag{3.11b}$$

where n_g is the number of Galerkin modes retained in the approximation, $\gamma_j = \kappa_j\sqrt{\bar{\rho}_1}x_f$, and $\beta_j = \kappa_j\sqrt{\bar{\rho}_2}(1-x_f)$. The acoustic wavenumbers κ_j are the solutions of the dispersion relationship

$$\sin\beta_j\cos\gamma_j + \cos\beta_j\sin\gamma_j\sqrt{\frac{\bar{\rho}_1}{\bar{\rho}_2}} = 0. \tag{3.12}$$

In the limit of uniform mean-flow density, i.e., $\bar{\rho}_1 = \bar{\rho}_2$, the wavenumbers are $\kappa_j = j\pi$ (Magri and Juniper, 2014). The derivation of the eigenmodes of the Rijke tube is provided in the Appendix § A.1.

3.4.3 A low-order model using Galerkin projection

The Rijke tube (Fig. 3.1) captures the qualitative nonlinear dynamics and bifurcations observed in real-life applications (Balasubramanian and Sujith, 2008). In this model, we further assume that (i) the mean flow has a zero Mach number with uniform density, and (ii) the boundary conditions are ideal, i.e. $p'(x=0,t) = p'(x=1,t) = 0$, hence $\Gamma_{\text{in}} = \Gamma_{\text{out}} = -1$. We solve the associated PDEs (3.8) using the Galerkin decomposition (3.10) and (3.11), with $\bar{\rho}_1 = \bar{\rho}_2 = 1$, which implies

that $\kappa_j = j\pi$ (Magri and Juniper, 2014), i.e.,

$$u'(x,t) = \sum_{j=1}^{n_g} \eta_j(t)\cos(j\pi x), \tag{3.13a}$$

$$p'(x,t) = -\sum_{j=1}^{n_g} \mu_j(t)\sin(j\pi x). \tag{3.13b}$$

By substituting the pressure and velocity variables in (3.8) with their Galerkin decompositions in (3.10) and projecting the dynamics onto the Galerkin modes, the dynamics of the Galerkin variables η_j and μ_j are described by a $2n_g$-dimensional system of ODEs (while most of the energy is captured by the first $n_g = 4$ modes, typically $n_g = 10$ offers a good trade off between accuracy and computational cost) (Huhn and Magri, 2020)

$$\dot{\eta}_j - \mu_j j\pi = 0, \tag{3.14a}$$

$$\dot{\mu}_j + \eta_j j\pi + \zeta_j \mu_j + 2\dot{q}'_f \sin(j\pi x_f) = 0, \tag{3.14b}$$

where the system's state is composed of the Galerkin coefficients for velocity $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots, \eta_{n_g}]^\top$ and for pressure $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_{n_g}]^\top$.

In the models of Rijke tube in literature (e.g., Landau et al., 1987; Balasubramanian and Sujith, 2008), a modal damping term acting on the pressure $\zeta p'(x,t)$ is added to the energy equation (3.8b) to account for the unmodelled effects of dissipation at the boundaries, and at viscous and thermal boundary layers at the tube walls. This term lets the higher modes to be damped out. In the projected dynamics, the damping term is $\zeta_j \mu_j$, where $\zeta_j = d_1 j^2 + d_2 j^{1/2}$ (Landau et al., 1987). The heat release rate is modelled by a modified King's law (Heckl, 1990; Huhn and Magri, 2020)

$$\dot{q}'_f = \beta\left(\sqrt{|1 + u'(x_f, t-\tau)|} - 1\right) \tag{3.15}$$

where β and τ are the heat release strength and the flame time delay, respectively.

The time-delayed problem is transformed into an initial value problem via an advection function with the dummy variable v so that it can be solved with a standard time-marching scheme (Huhn and Magri, 2020)

$$\frac{\partial v}{\partial t} + \frac{1}{\tau}\frac{\partial v}{\partial X} = 0, \quad 0 \le X \le 1, \quad v(X=0,t) = u'(x_f,t), \tag{3.16}$$

where the time-delayed velocity at the heat source is then given by $v(X=1,t) = u'(x_f, t-\tau)$. The PDE (3.16) is discretised using a Chebyshev spectral method with $n_c + 1$ points (Trefethen, 2000). Hence, $\boldsymbol{v}(t) = [v_1(t), v_2(t), \ldots, v_{n_c}(t)]^\top$ adds n_c state variables to the final system of ODEs. Let $\tilde{\boldsymbol{v}}(t) = [u'(x_f,t); \boldsymbol{v}(t)]$, then

$$\frac{d\tilde{\boldsymbol{v}}}{dt} = -\frac{2}{\tau}\boldsymbol{D}_c\tilde{\boldsymbol{v}}, \tag{3.17}$$

where $\boldsymbol{D}_c \in \mathbb{R}^{(n_c+1)\times(n_c+1)}$ is the Chebyshev differential matrix (Trefethen, 2000).

The derivative of the nonlinear heat release model (3.15) with respect to $u'(x_f, t-\tau)$ is not defined

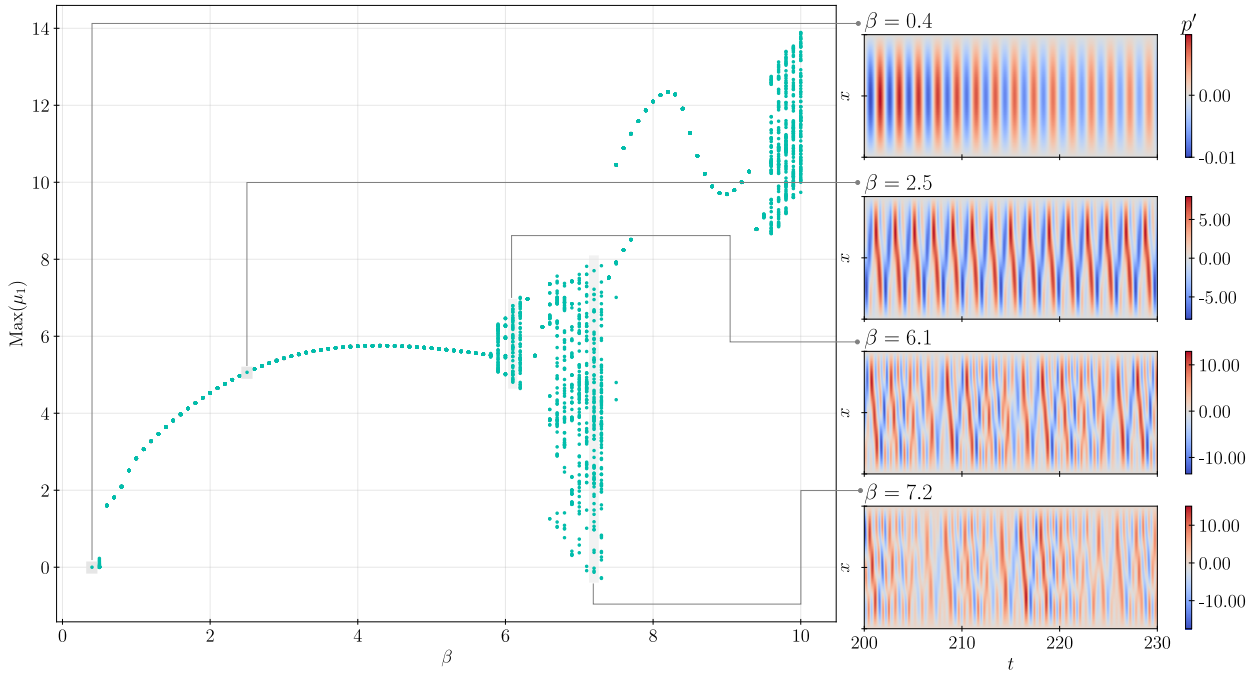


Figure 3.2: Bifurcation diagram of the Rijke tube with time delay $\tau = 0.2$ and varying heat source strength β. The plot shows the maxima of the first Galerkin coefficient associated with the pressure amplitude μ_1. The spatiotemporal evolution of pressure for fixed point, limit cycle, quasiperiodic, and chaotic regimes are visualised on the right.

for $u'(x_f, t-\tau) = -1$, which means that the Jacobian and, consequently, the sensitivity of the system can not be calculated at this value. Therefore, the heat-release model (3.15) has been modified to have smooth dynamics by approximating a small-neighbourhood of $u'(x_f, t-\tau) = -1$ with a fourth-degree polynomial (Huhn and Magri, 2020). We use this modification to compute the Jacobian of the Rijke tube and the adjoint sensitivities in § 5.6.3.

For the thermoacoustic system, the objective functional $\mathcal{J}$ is the time-averaged acoustic energy, which relates to the average energy of the undesired oscillations, given by (e.g. Huhn and Magri, 2020)

$$E_{\text{ac}}(t) = \int_0^1 \frac{1}{2}(u'^2(x,t) + p'^2(x,t))dx = \frac{1}{4}\sum_{j=1}^{n_g}(\eta_j^2(t) + \mu_j^2(t)). \tag{3.18}$$

We are interested in minimising the time-averaged acoustic energy

$$\langle E_{\text{ac}} \rangle = \frac{1}{T}\int_0^T E_{\text{ac}}(t)dt, \tag{3.19}$$

specifically in the limit $T \to \infty$ for long-term behaviour.

3.4.4 Bifurcation analysis using the low-order model

The low-order model (3.4.3) has been extensively employed to analyse the bifurcation behaviour of the Rijke tube (e.g., Balasubramanian and Sujith, 2008; Subramanian et al., 2010; Juniper, 2011; Huhn and Magri, 2020; Nóvoa and Magri, 2022). Huhn and Magri (2020) performed Covariant Lyapunov Vector analysis of this model and classified the existing nonlinear regimes according to their Lyapunov exponents (3.1). They performed their study in the parameter space given by

$\beta = [0.1, 10.0]$ and $\tau = [0.04, 0.4]$. The system displays fixed-point behaviour for low values of β or τ, whilst most of the parameter space is dominated by limit cycle regimes. Quasiperiodic and chaotic regimes are observed for values of $\beta \gtrsim 4.5$ and $0.08 \lesssim \tau \lesssim 0.36$. (Refer to Fig. 5 in Huhn and Magri (2020) for the visualisation.)

Fixing $\tau = 0.2$, the effect of varying the heat source strength is investigated in the bifurcation diagram in Fig. 3.2, where the different nonlinear regimes that arise are shown. (The other parameters are fixed to $x_f = 0.2$, $d_1 = 0.1$, $d_2 = 0.06$, $n_g = 4$, $n_c = 10$.) Around this τ value, the system follows the Ruelle-Takens-Newhouse route to the chaos (Huhn and Magri, 2020); at $\beta = 0.5$, it bifurcates from a fixed point solution to a limit cycle, at $\beta \approx 5.8$ it bifurcates from a limit cycle to a quasiperiodic attractor, between $5.8 \lesssim \beta \lesssim 6.5$ it switches between quasiperiodic and limit cycle solutions due to frequency locking, finally at $\beta \approx 6.5$ bifurcating to chaos. At $\beta \approx 7.4$, the system bifurcates to a limit cycle until $\beta = 9.5$, after which it starts showing intermittent behaviour.

A critical feature of thermoacoustic systems is that they can exhibit subcritical Hopf bifurcations (Subramanian et al., 2010). In a bistable region, a linearly stable fixed point and a stable limit cycle co-exist, separated by an unstable limit cycle. If the system is perturbed enough beyond a certain threshold, known as the fold point, it becomes attracted to the stable limit cycle and exhibits finite amplitude oscillations, i.e., triggering occurs (Juniper, 2011). This means that linear analysis alone is not sufficient to ensure stability as the system's behaviour also depends on the initial conditions. This bistable regime corresponds to both low β and τ points in the given model (Huhn and Magri, 2020). In § 5.6.3 and § 5.9, we choose the initial conditions such that the solutions both from the ground truth model and the echo state network converge to the stable limit cycle attractor for the consistency of the results.

3.4.5 A higher-fidelity model using travelling waves

In § 4.6, we will employ a higher-fidelity model of the Rijke tube to validate the proposed methodology. The solution of the original governing PDEs (3.6) can be obtained by a travelling wave approach with the jump conditions at the flame (Dowling, 1999). In this model, the pressure and velocity are expressed as functions of two acoustic travelling waves, which propagate up- and downstream of the tube. These waves are derived by applying the method of characteristics to the acoustic wave equation and defined as f and g, with propagation velocities $\tilde{\bar{c}}_1 \pm \tilde{\bar{u}}_1$ in the upstream region $\tilde{x} < \tilde{x}_f$; and h and j, with propagation velocities $\tilde{\bar{c}}_2 \pm \tilde{\bar{u}}_2$ in the downstream region $\tilde{x} > \tilde{x}_f$. This model can also account for nonideal open boundary conditions via reflection coefficients Γ_{in} and Γ_{out} at the inlet and outlet, $f(\tilde{t}) = \Gamma_{\text{in}} g(\tilde{t} - \tilde{\tau}_u)$ and $j(\tilde{t}) = \Gamma_{\text{out}} h(\tilde{t} - \tilde{\tau}_d)$, where $\tilde{\tau}_u$ and $\tilde{\tau}_d$ are the travelling times of the waves from the flame to the up- and downstream boundaries, respectively. If the boundary conditions are ideal, i.e., fully reflective, then $\Gamma_{\text{in}} = \Gamma_{\text{out}} = -1$. The full set of equations that describe the dynamics of the waves are given by (Magri et al., 2016a; Aguilar Perez, 2018)

$$\boldsymbol{X} \begin{bmatrix} g(\tilde{t}) \\ h(\tilde{t}) \end{bmatrix} = \boldsymbol{Y} \begin{bmatrix} g(\tilde{t} - \tilde{\tau}_u) \\ h(\tilde{t} - \tilde{\tau}_d) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\tilde{\dot{q}}_f - \tilde{\bar{\dot{q}}}_f}{\tilde{A}_1 \tilde{\bar{c}}_1} \end{bmatrix}, \tag{3.20}$$

where the matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are functions of the mean-flow variables and are obtained from the jump conditions. We generate the data with the code implementation of the kinematic flame model

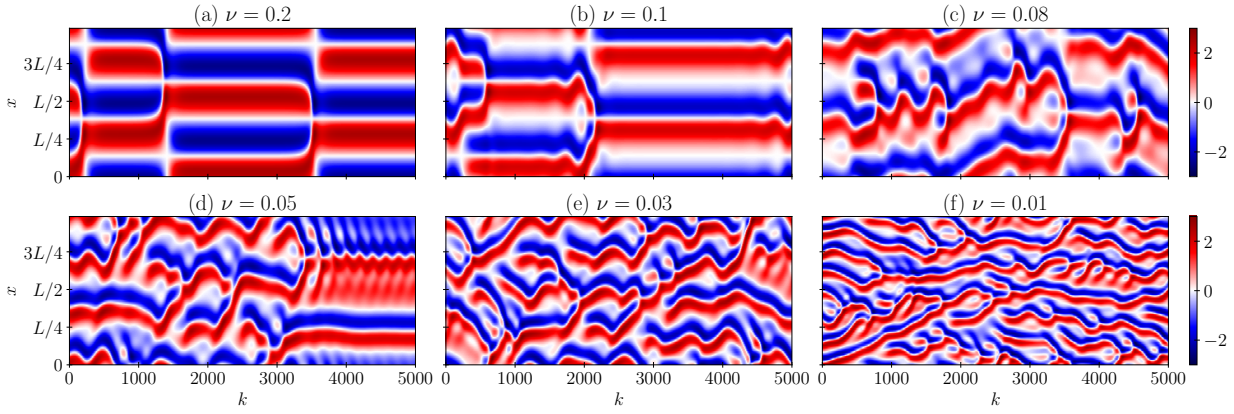


Figure 3.3: Different dynamical regimes of the Kuramoto-Sivashinsky (KS) equation with varying viscosity parameter ν. The solutions become increasingly more intricate as ν decreases.

available at Aguilar (2021).

3.5 Spatiotemporal chaos: the Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky (KS) equation is a prototypical model of spatiotemporal chaos, which arises in several physical phenomena such as chemical reaction-diffusion systems (Kuramoto, 1978), flame front instabilities (Sivashinsky, 1980) and liquid film flows (Sivashinsky and Michelson, 1980). Despite its one-dimensional form, the KS equation exhibits rich nonlinear dynamics and spatiotemporal chaos, making it a benchmark system for studying control of chaotic flows (e.g., Gomes et al., 2017; Shawki and Papadakis, 2020; Bucci et al., 2019). Including a forcing term f to implement controls, the equation is described by

$$\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} + u\frac{\partial u}{\partial x} = f(x,t), \tag{3.21}$$

where $u(x,t) : [0, L) \times [0, \infty) \to \mathbb{R}$ is the velocity on a periodic domain of length L, i.e., $u(x,t) = u(x+L,t)$, and f is a spatial function of the actuations $\boldsymbol{a}(t)$ (§ 6.3.3). Equivalently, Eq. (3.21) can be scaled to 2π-periodic domains, which introduces the viscosity parameter $\nu = (2\pi/L)^2$ multiplying the fourth order term (Gomes et al., 2017).

We discretise $u(x,t)$ using the Fourier expansion for even number of n_f modes

$$u(x,t) \approx \frac{1}{n_f}\left(c_0(t) + c_{n_f/2}(t)\exp\left(\mathrm{i}\frac{\pi n_f}{L}x\right) + \sum_{j=1}^{n_f/2-1} c_j(t)\exp\left(\mathrm{i}\frac{2\pi j}{L}x\right) + c.c.\right), \tag{3.22}$$

in which we retain the Fourier coefficients corresponding to the non-redundant positive wavenumbers $\boldsymbol{c} = [c_0, \ldots, c_{n_f/2}]^\top$ and ensure real-valuedness via the complex conjugate. We propagate the dynamics of the Fourier coefficients,

$$\frac{dc_j}{dt} = (\kappa_j^2 - \kappa_j^4)c_j(t) - \frac{1}{2}\mathrm{i}\kappa_j \sum_{p+q=j} c_p(t)c_q(t) + f_j(t), \tag{3.23}$$

where $\kappa_j = (2\pi j)/L$ is the wavenumber and $f_j(t)$ is the Fourier coefficient of the forcing term

at the wavenumber κ_j. The numerical solver follows the implementations of Bucci et al. (2019); Weissenbacher et al. (2025) for comparability of the results for RL-based control of the KS equation. The set of equations (3.23) is integrated using a semi-implicit third-order Runge-Kutta scheme (Kar, 2006). The nonlinear advection term is responsible for the transfer of energy between different scales, i.e., wavenumbers, as it is a convolution in the frequency domain. Numerically, this term is computed in the real domain as a multiplication. No explicit dealiasing (e.g., the 2/3 rule) is applied, as the energy spectrum decays rapidly with increasing wavenumbers (see Fig. 6.14) due to the fourth-order diffusion term $\partial^4 u/\partial x^4$. For the regimes that we study, a grid size of $n_f = 64$ sufficiently resolves the dynamics. For more chaotic regimes, i.e., $\nu < 0.01$, increasing the grid size or applying spectral dealiasing might become necessary.

As the viscosity parameter ν is varied, the uncontrolled KS equation displays different dynamical regimes. As the viscosity parameter decreases $\nu < 1$ (i.e., as $L > 2\pi$), the stable zero solution becomes unstable, and the system bifurcates to periodic, traveling wave, and chaotic solutions with increasing dynamical complexity (Kalogirou et al., 2015). The solutions are considered fully chaotic for $\nu \leq 0.01$. These regimes are visualised in Fig. 3.3. In § 6.3, we will focus on the regimes $\nu = \{0.08, 0.05, 0.03, 0.01\}$ (Fig. 3.3(c – f)), which range from weakly chaotic to transient and fully chaotic.

Chapter 4

Hard-constrained neural networks for modelling nonlinear thermoacoustics

The work in this chapter can be found in Ozan and Magri (2023b) and Ozan and Magri (2023a). The code is available as a GitHub repository at MagriLab/HCTA. A step-by-step tutorial of the code is available at MagriLab/NewtonWorkshop2023. The simulation dataset for thermoacoustics can be downloaded at Ozan (2023).

4.1 Introduction

In this chapter, we design Galerkin neural networks, which hard-encode the solution structure into the architecture. These networks are inspired by Galerkin projection, which is a common technique for solving PDEs by projecting the equations onto a finite set of modes, which transforms the problem into a set of ODEs (§ 2.2). The basis can be analytically derived such as Fourier modes, or data driven such as proper orthogonal decomposition (POD) modes (Sirovich, 1987; Holmes et al., 1996). In this chapter, we take advantage of a physical basis.

We focus on solutions from acoustics and thermoacoustics (§ 3.4), which originate from nonlinear wave equations. Traditionally, the prediction of thermoacoustics has been achieved with first principles. In the time domain, a direct approach is the brute-force time integration of the governing equations following a discretisation scheme. High-fidelity models such as large-eddy simulations that model the acoustics and the flame simultaneously on fine grids provide highly accurate solutions, but they are computationally expensive (Poinsot, 2017). To reduce the computational cost of brute-force time integration, continuation and shooting methods find limit cycles as exact periodic solutions of the equations and great accuracy, but they may be computationally expensive and require iterative methods (Waugh et al., 2014). Low-fidelity models reduce the computational effort at the expense of accuracy while obtaining models that can be used for stability, bifurcation analysis, and parametric studies. Generally, these approaches combine a linear acoustic solver with a heat release model. In this direction, nonlinear behaviour of longitudinal and annular combustors has been investigated by employing network models based on the travelling-wave (e.g., Dowling, 1999; Dowling and Stow, 2003; Li and Morgans, 2015; Bauerheim et al., 2014; Orchini et al., 2019),

Galerkin decomposition of pressure and velocity using only a finite number of acoustic eigenmodes and projection of the PDEs onto those modes (e.g., Zinn and Lores, 1971; Balasubramanian and Sujith, 2008), and numerical discretisations of the PDEs (Sayadi et al., 2014; Huhn and Magri, 2020). In the frequency domain, flame models are defined via a flame transfer function (FTF), or a flame describing function (FDF). FTF cannot predict the limit-cycle amplitude since it is a linear function of only the input frequency and not the amplitude. In contrast, FDFs are defined as transfer functions parameterised with the input amplitude, thus, the zero growth rate mode can be identified and consequently the existence and the limit-cycle amplitude can be predicted (Noiray et al., 2008). This approach provides computationally cheap estimates, however, the FDF operates only with single input and single output frequencies, which has been extended to multiple frequencies (Orchini and Juniper, 2016; Haeringer et al., 2019). Predictions with first principles only are either computationally expensive (e.g., large-eddy simulation), or as good as the model assumptions. Because thermoacoustics is a multi-physics phenomenon, the model assumptions made in low-order models are inevitably substantial. This motivates adding data into the first-principles modelling of thermoacoustics, for which machine learning methods excel. The data typically comes from laboratory experiments that are conducted with setups consisting of a duct with a heat source, which can be a flame or an electrically heated wire mesh (e.g., Matveev, 2003; Kabiraj et al., 2012), also in annular configuration (e.g., Moeck et al., 2010). In the experiments, the collected data is usually the acoustic pressure measured by microphones at high sampling rates.

In this direction, data assimilation techniques have been applied to improve physics-based qualitatively accurate models of a prototypical thermoacoustic system, whilst estimating model parameters. Nóvoa and Magri (2022) combined a thermoacoustic low-order model with an ensemble Kalman filter, and an echo state network for real-time bias-aware data assimilation of the acoustic state (Nóvoa et al., 2023, 2024a). In other applications, neural networks have been developed for heat release model or flame response inference in thermoacoustics (Selimefendigil and Polifke, 2011; Jaensch and Polifke, 2017; Tathawadekar et al., 2021). Physics-informed neural networks were also shown to be successful for the reconstruction of acoustic fields with nonideal boundary conditions (Silva Garzon et al., 2023).

The overarching goal of this chapter is to generalise and develop acoustic neural networks to embed the structure of the nonlinear wave solution into the architecture. The specific objective of this chapter is three-fold: to (i) predict and extrapolate in time thermoacoustic oscillations, (ii) reconstruct pressure and velocity over the entire domain from pressure sensors only, and (iii) obtain a model that is robust to noise and generalisable to unseen scenarios.

This chapter is organised as follows. In § 4.2, we provide the mathematical background for feedforward neural networks. Building on Galerkin decomposition (§ 2.2), in § 4.3, we introduce Galerkin neural networks. In § 4.4, we discuss the application of Galerkin neural networks to acoustics and thermoacoustics developing acoustic neural networks. In § 4.5 and § 4.6, we show results for twin experiments on synthetic data from a Rijke tube (§ 3.4.3) and on synthetic data from a higher-fidelity model (§ 3.4.5). The chapter ends with § 4.7 laying out the conclusions.

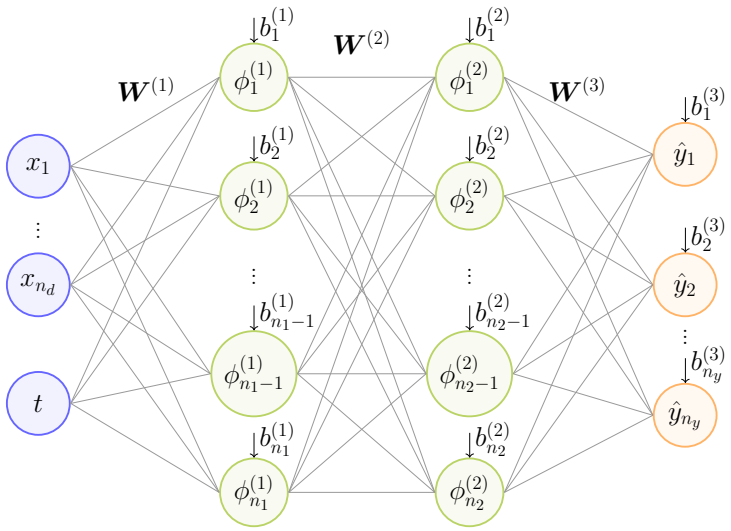


Figure 4.1: Example of a standard feedforward neural network (FNN) with two hidden layers. This is the data-driven only model, which has no prior knowledge embedded in the architecture. The trainable parameters are the weights, $\boldsymbol{W}^{(\cdot)}$, and the biases, $\boldsymbol{b}^{(\cdot)}$.

4.2 Standard feedforward neural networks

Let $\boldsymbol{y} \in \mathbb{R}^{n_y}$ represent some vector of physical quantities pertaining to a system that depends on space, $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^{n_d}$, where n_d is the dimension, and time, $t \in \mathbb{R}$, e.g., $\boldsymbol{y} = [p'(x,t); u'(x,t)]$ contains the one-dimensional acoustic pressure and velocity in the Rijke tube (§ 3.4.1). Then, given full or partial observations of $\boldsymbol{y}$, our goal is to learn a model $\boldsymbol{f}$ that predicts an output vector $\hat{\boldsymbol{y}} \in \mathbb{R}^{n_y}$ from an input vector $(\boldsymbol{x}, t)$ while minimising an error metric $\mathcal{L}$. A feedforward neural network (FNN) is defined by a composition of functions, $\boldsymbol{f}$, which, when appropriately designed, can approximate any continuous function in a specified range (Hornik et al., 1989)

$$\boldsymbol{f}(\boldsymbol{x}, t) := \boldsymbol{f}^{(n_L)}\left(\boldsymbol{f}^{(n_L-1)}\left(\dots \boldsymbol{f}^{(1)}\left(\boldsymbol{x}, t\right)\right)\right), \tag{4.1a}$$

$$\boldsymbol{f}^{(l)}(\boldsymbol{z}) := \boldsymbol{\phi}^{(l)}\left(\boldsymbol{W}^{(l)}\boldsymbol{z} + \boldsymbol{b}^{(l)}\right), \tag{4.1b}$$

where $\boldsymbol{f}^{(l)}$ is the function that maps the layer $l-1$ to the layer l, where $l = 1, \dots, n_L$ and n_L is the number of layers; $\boldsymbol{W}$ are the weights matrices and $\boldsymbol{b}$ are the biases, which are the trainable parameters; and $\boldsymbol{\phi}^{(l)}$ are the activation functions from the layer $l-1$ to the layer l, which are applied to each component of the argument (Fig. 4.1). For regression, in the last layer, the activation $\boldsymbol{\phi}^{(n_L)}$ is linear. The neural network offers an ansatz for a continuous function through linear operations and simple nonlinear activations (refer to Magri (2023) for a pedagogical and geometric explanation).

The FNN is a standard architecture that is fully data driven, i.e., no prior knowledge is embedded in the network (Fig. 4.1). The network's weights and biases are optimised via gradient descent to minimise an error, which is provided by a loss function $\mathcal{L}$

$$\{\boldsymbol{W}^*, \boldsymbol{b}^*\} = \underset{\{\boldsymbol{W}, \boldsymbol{b}\}}{\arg\min}\, \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}). \tag{4.2}$$

When no physics-constraint is imposed as in standard neural networks, the data-driven loss is quantified by the mean-squared error (MSE) between the measured data, i.e., observations (2.11), and predictions of the network (4.1a). We account for the cases of partial state observations with an observation operator $\mathcal{M}$ (2.12) that indicates which state variables are measured. This includes both the case of sparse sensor placement, i.e., a coarse grid of measurements, and the case where some physical variables are entirely unobserved. Our dataset $\mathcal{D}$ consists of pointwise measurements of the physical variables $y_i \in \boldsymbol{y},\ i = 1, 2, \ldots, n_y$ sampled at discrete space and time points, that is

$$\mathcal{D}_i = \left\{ \left(\boldsymbol{X}_o^i, t_k, \boldsymbol{o}_i(t_k) \right) \right\}_{k=1}^{N_{\text{train}}}, \quad \mathcal{D} = \{\mathcal{D}_i\}_{i=1}^{n_y} \tag{4.3}$$

where

$$\boldsymbol{o}_i(t_k) = \boldsymbol{y}_i\left(\boldsymbol{X}_o^i, t_k\right) + \epsilon_i(t_k),$$

$$\boldsymbol{X}_o^i = \begin{bmatrix} \boldsymbol{x}_{o,1}^{i\ \top} \\ \boldsymbol{x}_{o,2}^{i\ \top} \\ \vdots \\ \boldsymbol{x}_{o,n_o^i}^{i\ \top} \end{bmatrix}, \quad \boldsymbol{y}_i\left(\boldsymbol{X}_o^i, t_k\right) = \begin{bmatrix} y_i(\boldsymbol{x}_{o,1}^i, t_k) \\ y_i(\boldsymbol{x}_{o,2}^i, t_k) \\ \vdots \\ y_i(\boldsymbol{x}_{o,n_o^i}^i, t_k) \end{bmatrix}, \quad \boldsymbol{o}_i(t_k) = \begin{bmatrix} o_{i,1}(t_k) \\ o_{i,2}(t_k) \\ \vdots \\ o_{i,n_o^i}(t_k) \end{bmatrix}$$

and $\boldsymbol{o}_i(t_k) \in \mathbb{R}^{n_o^i}$ are the observations, $\boldsymbol{X}_o^i \in \mathbb{R}^{n_o^i \times n_d}$ with $n_o = \sum_i n_o^i$ are the observed spatial locations, i.e., sensor locations in an experiment, and $\boldsymbol{\epsilon}_i(t_k) \in \mathbb{R}^{n_o^i}$ is the measurement noise. In this work we sample measurement noise from the Gaussian distribution,

$$\boldsymbol{\epsilon}_i(t_k) \sim \mathcal{N}(0, \sigma_o^2 \sigma_i^2), \tag{4.4}$$

where σ_i is the standard deviation of the physical quantity y_i computed over the training domain and σ_o is the scaling that determines the noise level. We denote the length of the time series in time units as $t_{\text{train}} = \Delta t_o N_{\text{train}}$, where $1/\Delta t_o$ is the observation frequency. (In this chapter, we do not test against varying observation frequency, which will be considered in § 6).

During training, we sample batches of input-output data from the datasets $\mathcal{D}_i$. For this purpose, we first rewrite the dataset as a list which we can sample input-output pairs from

$$\mathcal{D}_i = \left\{ \left(\boldsymbol{x}_o^{i\,(n)}, t^{(n)}, o_i^{(n)} \right) \right\}_{n=1}^{N_{\text{train}} n_o^i}. \tag{4.5}$$

We define a batch $\mathcal{B}_i \subseteq \mathcal{D}_i$ of size N_b, where each sample is a tuple $\left(\boldsymbol{x}_o^i, t, o_i\right)$. The data-driven loss over one batch is then given as

$$\mathcal{L}_{\mathcal{D}_i} = \frac{1}{N_b} \sum_{n=1}^{N_b} \left(o_i^{(n)} - \hat{y}_i \left(\boldsymbol{x}_o^{i\,(n)}, t^{(n)} \right) \right)^2, \tag{4.6}$$

where $\left(\boldsymbol{x}_o^{i\,(n)}, t^{(n)}, o_i^{(n)} \right) \in \mathcal{B}$.

We compute the total data-driven loss as

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{|\mathcal{I}_o|} \sum_{i \in \mathcal{I}_o} \mathcal{L}_{\mathcal{D}_i}, \tag{4.7}$$

where $\mathcal{I}_o \subseteq \{1, \ldots, n_y\}$ is the set of indices corresponding to the physical variables that we have some observations from, and $|\mathcal{I}_o|$ is its cardinality. For example, in this work, we consider cases where we reconstruct both pressure and velocity fields from only sparse measurements of pressure, e.g., § 4.6.2.

4.3 Galerkin neural networks

FNNs are flexible and useful tools for function approximation, however, they can suffer from under- or overfitting, especially in the case of scarce or noisy data. This requires a careful hyperparameter tuning, which becomes computationally expensive in proportion with the size of network. Even then, the network may not be able to capture some features of the solution because of missing data. In order to counteract these shortcomings, we exploit the physical knowledge about the spatiotemporal basis of the system in question, and propose a network structure inspired by the Galerkin decomposition of the system as motivated in § 2.2. The chosen Galerkin modes are a known nonlinear transformation of the spatial coordinates, which is introduced in the network. Therefore, by design, this network can be configured to automatically satisfy the boundary conditions. We will refer to this network architecture as the *Galerkin neural network* (*GalNN*). The Galerkin network is composed of a spatial branch that transforms $\boldsymbol{x}$ into the *a priori* Galerkin modes, $\boldsymbol{\Psi}^i(\boldsymbol{x})$ (2.8), and a temporal branch that is an FNN (4.1) that takes only t as an input and predicts the unknown Galerkin amplitudes, $\boldsymbol{\alpha}^i(t)$ (2.8), as outputs. Although here we fix the Galerkin modes a priori with a suitable basis for acoustics applications, the neural network ansatz naturally allows for adaptability and future extensions, where the spatial modes are optimised through trainable weights (§ 7.2). The final outputs, $\hat{\boldsymbol{y}}$, are computed by (2.8). Formally, the GalNN is defined as

$$f_i(\boldsymbol{x}, t) = \sum_{j=(i-1)n_g+1}^{in_g} g_j(\boldsymbol{x}) h_j(t), \tag{4.8}$$

where $\boldsymbol{f}$ is the map from $(\boldsymbol{x}, t)$ to $\hat{\boldsymbol{y}}$, $i = 1, 2, \ldots, n_y$ denotes the i^{th} element of the output vector, $\boldsymbol{g}(\boldsymbol{x})$ is the spatial branch given by $\boldsymbol{g}(\boldsymbol{x}) = [\boldsymbol{\Psi}^1(\boldsymbol{x}); \boldsymbol{\Psi}^2(\boldsymbol{x}); \ldots; \boldsymbol{\Psi}^{n_y}(\boldsymbol{x})]$ and $\boldsymbol{h}(t)$ is the temporal branch modelled as an FNN, the outputs of which are denoted as $[\boldsymbol{\alpha}^1(t); \boldsymbol{\alpha}^2(t); \ldots; \boldsymbol{\alpha}^{n_y}(t)]$. The data-driven loss is defined the same as the standard FNN case (4.7) over the prediction error on $\boldsymbol{y}$. The time evolution of the Galerkin amplitudes are thus obtained as an intermediary step. This architecture is shown in Fig. 4.2.

4.4 Acoustic neural networks

The dynamics of thermoacoustic oscillations are dominated by unstable eigenfunctions (e.g., Magri, 2019). We propose neural networks that can naturally infer acoustic dynamics, which are the basis functions of nonlinear thermoacoustic behaviours. The proposed networks embed the prior knowledge through the activation functions (§ 4.4.1), through a penalisation loss function in the training (soft constraint, § 4.4.2), and through the architecture (hard constraint, § 4.4.3). For the thermoacoustic system described in § 3.4, the input vector consists of the one-dimensional spatial coordinate, x, and time, t, and the output vector consists of the acoustic pressure and

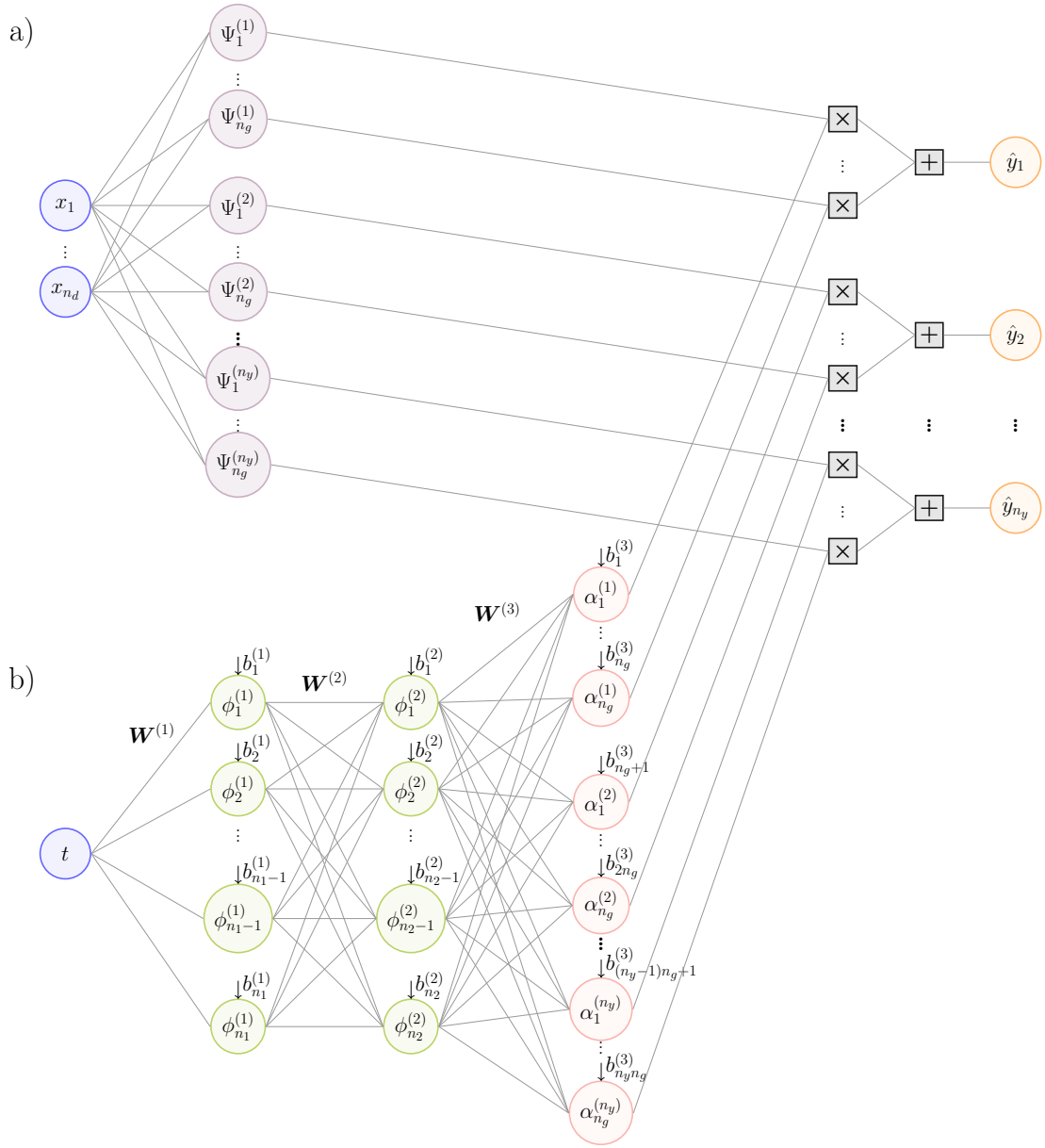


Figure 4.2: Galerkin neural network (GalNN). This network is composed of two branches; (a) spatial branch (hard constraint), and (b) a temporal branch, which learns the temporal behaviour. The trainable parameters are the weights, $\boldsymbol{W}^{(.)}$, and the biases, $\boldsymbol{b}^{(.)}$.

velocity fluctuations, $\boldsymbol{y} = [p', u']$. Subsequently, we denote the sensor locations for the pressure and velocity measurements as $\boldsymbol{x}_o^p \in \mathbb{R}^{n_o^p}$ and $\boldsymbol{x}_o^u \in \mathbb{R}^{n_o^u}$, respectively. In § 4.4.1, we motivate employing periodic, i.e., sinusoidal, activation functions in the standard FNN for acoustic problems, as solutions to the linear, undamped wave equation are characterised by sinusoids (§ A.1). The nonlinear, time-delayed and damped equations in thermoacoustic equations extend this behaviour with richer, multi-frequency temporal dynamics (§ 3.4.4). Further, in § 4.4.2, we include a physics-based regularisation term that penalises solutions that violate the conservation laws governing the acoustic dynamics. Finally, in § 4.4.3, we promote a hard-constrained architecture in the form of a GalNN (§ 4.3). Inspired by separation of variables (§ 2.2), we design a neural network architecture that spans the Hilbert space with the acoustic eigenfunctions as the Galerkin modes, whilst having trainable parameters for inference and closure of the unknown physical terms. The acoustic neural networks are summarised at the end of this section in Tab. 4.1.

4.4.1 Periodically activated feedforward neural networks *(P-FNNs)*

FNNs with conventional activation functions fail at extrapolating periodic functions in time, where the extrapolation behaviour is determined by the analytical form of the activation function (Ziyin et al., 2020), i.e., the rectified linear unit (ReLU) acts linearly, and the hyperbolic tangent (tanh) saturates to a constant. In order to augment the extrapolation capability, we observe that nonlinear thermoacoustic dynamics originate from the nonlinear interactions of the acoustic modes (e.g., Magri, 2019), which reduce to Fourier modes in uniform ducts with fully reflecting boundary conditions. From a data-driven perspective, the sinusoidal patterns of the data in both time and space motivate the use of weights and biases that are periodically activated. A straightforward strategy is to employ periodic activations in the acoustic neural networks, i.e., $\boldsymbol{\phi}^{(l)} = \sin(\boldsymbol{z})$ in Eq. (4.1b) (Sitzmann et al., 2020; Ziyin et al., 2020). The physics of the system, namely the periodic nature of the solutions, is embedded in the network itself via the choice of activation function, which provides an inductive bias on the function space of the network and improves extrapolation (Ziyin et al., 2020). While a single sine layer with one neuron would result in periodic behaviour, composition of multiple layers with multiple neurons enable capturing the more complex patterns in the data that include multiple frequencies. The physical meaning of the trained weights of sine activated neural networks will be demonstrated in § 4.5.2, for example, in Fig. 4.7. Furthermore, this choice of activation function can also mitigate the known spectral bias of neural networks toward lower frequencies (Rahaman et al., 2019). The weights of a layer l with the sine activation are initialised from a uniform distribution in the range $\left[-\sqrt{3/\text{fan}_\text{in}}, \sqrt{3/\text{fan}_\text{in}}\right]$, where fan_in is the number of neurons in the layer $l-1$ (Ziyin et al., 2020), and they are updated during training using gradient descent to minimise the loss $\mathcal{L}$.

4.4.2 Soft constraints with physics-informed losses

Prior knowledge can be embedded as a penalisation term in the loss function in Eq. (4.2) to minimise during training (e.g., Lagaris et al., 1998; Raissi et al., 2019a). This approach improves the generalisability of the network, while promoting physical outputs (Karniadakis et al., 2021). In thermoacoustics, neural network predictions should fulfill the acoustic conservation laws, therefore,

we penalise solutions that violate momentum and energy equations in the loss function

$$\mathcal{L} = \lambda_{\mathcal{D}}\mathcal{L}_{\mathcal{D}} + \lambda_M\mathcal{L}_M + \lambda_E\mathcal{L}_E, \tag{4.9}$$

where $\mathcal{L}_{\mathcal{D}}$ is the data-driven loss (4.7) (§ 4.2), and $\mathcal{L}_M$ and $\mathcal{L}_E$ are the residual losses from the conservation of momentum and energy equations, respectively. In the physics-informed loss (4.9), we do not impose the conservation of mass equation (3.6a), as the linearised equations for momentum (3.6b) and energy (3.6c) are independent of the density fluctuations ρ', and are sufficient to determine the pressure and velocity fluctuations, p' and u'. This formulation assumes that the isentropic relation for an ideal gas holds, where the density fluctuations can be computed a posteriori from the pressure. The non-negative scalars $\lambda_{\mathcal{D}}$, λ_M, and λ_E are regularisation hyperparameters. Obtaining momentum and energy losses, $\mathcal{L}_M$ and $\mathcal{L}_E$, requires the evaluation of the physical residuals of the network predictions. We denote the partial differential operators that define the residual from momentum and energy equations as $\tilde{\mathcal{F}}_M(p', u')$ and $\tilde{\mathcal{F}}_E(p', u')$, respectively. A given (p', u') is a solution of the system if the residuals are zero, i.e., $\tilde{\mathcal{F}}_M(p', u') = 0$ and $\tilde{\mathcal{F}}_E(p', u') = 0$. We will exactly define these residuals separately for the low-order model of the Rijke tube (§ 3.4.3) and for the higher-fidelity model (§ 3.4.5). As we can evaluate the physical loss at any location in time and space, we let $\mathcal{X}_s = \{(x^{(n)}, t^{(n)})\}_{n=1}^{N_s}$ denote the set of collocation points uniformly sampled over the entire spatiotemporal domain, i.e., $x^{(n)} \sim \mathcal{U}(0, 1)$ and $t^{(n)} \sim \mathcal{U}(0, t_{\text{train}})$. Recalling our definition of the batch of observed space and time locations (4.6), we denote $\mathcal{X}_b = \{(x^{(n)}, t^{(n)})\}_{n=1}^{N_b}$, where $x^{(n)} \in \left\{\boldsymbol{x}^p_{o,i}\right\}_{i=1}^{n_o^p} \cup \left\{\boldsymbol{x}^u_{o,i}\right\}_{i=1}^{n_o^u}$ and $t^{(n)} \in \{t_k\}_{k=1}^{N_{\text{train}}}$. The physics-informed losses are then computed as

$$\mathcal{L}_M = \frac{1}{|\mathcal{X}|}\sum_{(x,t)\in\mathcal{X}} \tilde{\mathcal{F}}_M\big(\hat{p}'(x,t), \hat{u}'(x,t)\big)^2, \tag{4.10a}$$

$$\mathcal{L}_E = \frac{1}{|\mathcal{X}|}\sum_{(x,t)\in\mathcal{X}} \tilde{\mathcal{F}}_E\big(\hat{p}'(x,t), \hat{u}'(x,t)\big)^2, \tag{4.10b}$$

where $\mathcal{X} = \mathcal{X}_b \cup \mathcal{X}_s$ and $|\mathcal{X}| = N_b + N_s$ is its cardinality. To evaluate the losses, we employ automatic differentiation using `tf.GradientTape` functionality from TensorFlow (Abadi et al., 2015). The automatic differentiation of the output variables with respect to the input variables yields the Jacobian that contains the partial derivatives $\partial\hat{p}'/\partial x$, $\partial\hat{p}'/\partial t$, $\partial\hat{u}'/\partial x$, and $\partial\hat{u}'/\partial t$. These partial derivatives are plugged in the operators, $\tilde{\mathcal{F}}_M$ and $\tilde{\mathcal{F}}_E$, to calculate the residual of the network.

4.4.3 Hard-constraints with Galerkin neural networks *(GalNNs)*

Linear wave equations and wave solutions can be represented in an acoustic eigenbasis, namely Fourier modes, which is complete. We note that only in the one-dimensional setting considered here, the acoustic eigenfunctions are simple harmonics. In higher dimensions, the eigenfunctions can be determined using a Helmholtz equation solver. On the other hand, the thermoacoustic problem is governed by a wave equation with a nonlinearity arising from the heat release. However, the nonlinearity will only slightly change the frequency and mode coupling, which motivates the use of the acoustic eigenspace as an expressive basis to represent the nonlinear solutions as well (Zinn and Lores, 1971). Motivated by the Galerkin decomposition (3.10), we express the acoustic and thermoacoustic solutions as a GalNN (4.8). We embed the eigenmodes of the Rijke tube described

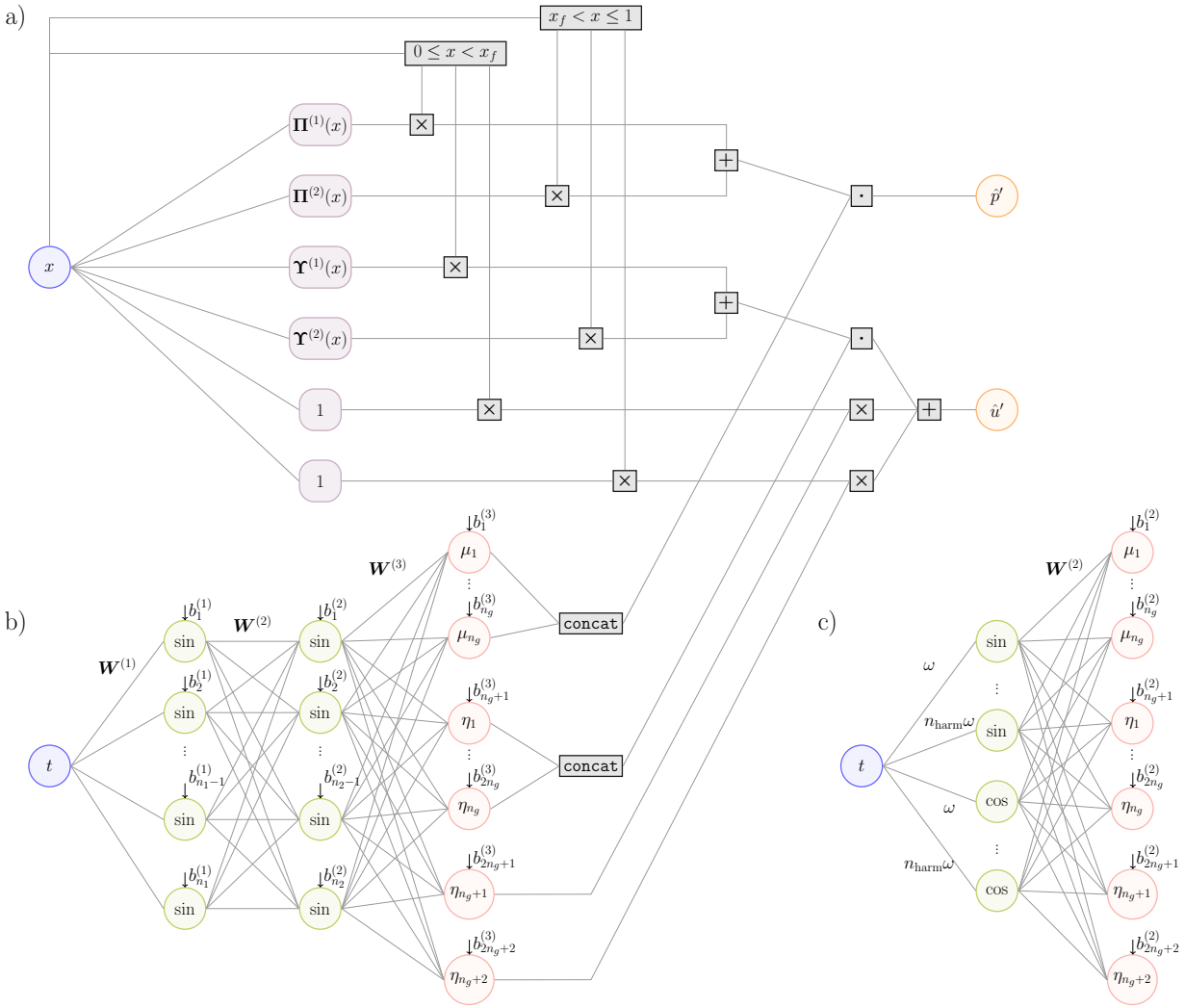


Figure 4.3: The Galerkin neural network (GalNN) tailored for thermoacoustics. (a) Spatial branch, in which the solution space is spanned by the acoustic eigenfunctions (3.11) (hard constraint). The spatial modes $\Pi_j^{(1)}$, $\Pi_j^{(2)}$, $\Upsilon_j^{(1)}$, $\Upsilon_j^{(2)}$, where $j = 1, 2, .., n_g$, are collected in the vectors $\mathbf{\Pi}^{(1)}$, $\mathbf{\Pi}^{(2)}$, $\mathbf{\Upsilon}^{(1)}$ and $\mathbf{\Upsilon}^{(2)}$, respectively. Step functions that model the discontinuity through the compact heat source (4.11) are shown at the bottom of the spatial branch. (b) Temporal branch, which is a periodically activated feedforward neural network. The outputs μ_j and η_j are stacked as vectors, which are projected onto the modes $\mathbf{\Pi}^{(1)}$, $\mathbf{\Pi}^{(2)}$, $\mathbf{\Upsilon}^{(1)}$ and $\mathbf{\Upsilon}^{(2)}$. The outputs η_{n_g+1} and η_{n_g+2} are associated with the discontinuity modes (4.11). The trainable parameters are the weights, $\boldsymbol{W}^{(\cdot)}$, and the biases, $\boldsymbol{b}^{(\cdot)}$ (c) Temporal branch of the periodic Galerkin neural network further constrained for limit-cycle solutions. It is reconstructed such that the weights are integer multiples of only one trainable fundamental angular frequency, ω. The trainable parameters are ω, the weights, $\boldsymbol{W}^{(2)}$, and the biases, $\boldsymbol{b}^{(2)}$.

in § 3.4.2 in the GalNN, where $\Pi_j^{(1)}$, $\Pi_j^{(2)}$ (3.11a), and $\Upsilon_j^{(1)}$, $\Upsilon_j^{(2)}$ (3.11b) are the Galerkin modes, and μ_j and η_j are the Galerkin amplitudes, corresponding to pressure and velocity, respectively. (The superscripts $(\cdot)^{(1)}$ and $(\cdot)^{(2)}$ denote the up- and downstream conditions across the flame.) The pressure and velocity are determined from the predicted Galerkin amplitudes using (3.10). This architecture is shown in Fig. 4.3 (a, b).

Whereas the acoustic pressure is continuous across the flame, the acoustic velocity undergoes a discontinuity due to the dilation from the heat-release rate (Magri, 2019). When such a discontinuity is approximated with a finite Fourier series, such as the Galerkin modes we are using, the Gibbs phenomenon manifests itself as high-frequency oscillations around the discontinuity (Magri and Juniper, 2013; Sayadi et al., 2014), which causes an unphysical behaviour. To physically capture the discontinuity and eliminate unphysical oscillations in the predictions, we add two step functions in the velocity modes

$$\Upsilon_{n_g+1}^{(1)}(x) = 1, \quad \Upsilon_{n_g+1}^{(2)}(x) = 0, \tag{4.11a}$$

$$\Upsilon_{n_g+2}^{(1)}(x) = 0, \quad \Upsilon_{n_g+2}^{(2)}(x) = 1, \tag{4.11b}$$

such that the summation of the velocity modes (3.10b) runs from 1 to $n_g + 2$. The modes $\Upsilon_{n_g+1}^{(.)}$ and $\Upsilon_{n_g+2}^{(.)}$ are weighted by independent coefficients, η_{n_g+1} and η_{n_g+2}, which allows them to capture a jump discontinuity at the flame location.

The temporal branch of the GalNN should be capable of extrapolation in time of thermoacoustic oscillations. These solutions bifurcate from limit-cycle oscillations, which are periodic, to quasiperiodic and chaotic oscillations. As they originate from periodic solutions, even the quasiperiodic and chaotic solutions are in fact dominated by periodicity, which can also be observed in the frequency spectrum of the oscillations (later in § 4.5.2). Therefore, we deploy a periodically activated FNN (§ 4.4.1) as the temporal branch of the GalNN.

4.4.4 Periodic Galerkin neural networks *(P-GalNNs)*

We enforce periodic activations (§ 4.4.1) in the Galerkin neural network in order to achieve extrapolation for limit-cycle, quasiperiodic, and chaotic solutions. For simplicity, if we consider a single-layer periodically activated GalNN, then the outputs of the temporal branch of the network are given as a sum of sinusoids, where the hidden layer weights represent the angular frequencies and the biases represent the phases, i.e., $\sin(\boldsymbol{W}^{(1)}t + \boldsymbol{b}^{(1)})$. For a signal given as a sum of sinusoids, its angular frequency is the greatest common divisor of the angular frequencies of its components. For a limit cycle, the greatest common divisor is its fundamental angular frequency, and the other frequencies are its harmonics. This applies to a periodically activated neural network as well. However, the weights of the neural network are initialised randomly and then trained on a finite amount of data, so it is numerically unlikely for the learned weights to be exact integer multiples of a fundamental frequency. To overcome this numerical challenge, we add a further constraint on the GalNN to guarantee periodic behaviour with the fundamental frequency of the limit cycle. We create a hidden layer that takes time as input and outputs

$$\boldsymbol{\phi}^{(1)}\left(\boldsymbol{W}^{(1)}t\right) = \left[\sin(\omega t), \sin(2\omega t), \ldots, \sin(n_{\text{harm}}\omega t), \cos(\omega t), \cos(2\omega t), \ldots, \cos(n_{\text{harm}}\omega t)\right]^\top$$

Table 4.1: Summary of the proposed hard-constrained feedforward neural networks (FNNs). All the networks can be equipped with a soft constraint, which penalises non-physical solutions in the loss function (see § 4.4.2). When the soft constraint is present, the abbreviation will be prefixed with PI (physics-informed). For example, a Galerkin network with the soft constraint will be abbreviated as PI-GalNN.

Network	Abbreviation	Constraint	Separation of time and space	Can extrapolate in time	Section
Feedforward neural network	FNN	No	No	No	§ 4.2
Periodically activated feedforward neural network	P-FNN	No	No	Yes, via activation function	§ 4.4.1
Galerkin neural network	GalNN	Hard	Yes	Yes	§ 4.4.3
Periodic Galerkin neural network	P-GalNN	Hard	Yes	Yes, for limit-cycles it guarantees fundamental frequency	§ 4.4.4

such that the weights in the periodic activations are integer multiples of one trainable variable, ω. The variable ω is initialised to the nondimensional angular frequency of the acoustic system, π, which is an educated guess for the actual thermoacoustic frequency as from our physical knowledge we know that the nonlinearity will only slightly change this. Hence, upon training, ω corresponds to the angular frequency of the limit cycle, and the number of harmonics n_{harm} can be regarded as a hyperparameter. Because we have both sine and cosine activations for the same angular frequency, the phase information is automatically captured, hence we do not require a bias term. The temporal branch of the GalNN tailored for limit cycles is shown in Fig. 4.3 (c).

The periodic Galerkin neural network shares structural similarities with the spectral POD-Galerkin method (Towne et al., 2018) in that they both represent the solution as a modal expansion at fixed temporal frequencies (which are harmonics of a fundamental frequency if the underlying data is periodic). Here, we address the problem of reconstruction of thermoacoustic oscillations from sparse sensor measurements using a priori spatial modes, whereas spectral POD is in general employed to extract coherent structures from uniformly sampled data. Therefore, we leave the direct comparison of the two methods as scope for future work.

4.5 Extrapolation and state reconstruction: Twin experiments

We perform twin experiments on synthetic data generated from a prototypical thermoacoustic system, the Rijke tube 3.4.3, to demonstrate how the acoustic neural networks developed in 4.4 tackle extrapolation and state reconstruction.

4.5.1 Computation of the physics-informed loss

The physics-informed loss of the Rijke tube system is calculated by evaluating the left-hand side of the PDEs (3.8), i.e., the momentum residual, $\tilde{\mathcal{F}}_M$, is defined as the left-hand side of (3.8a), and the energy residual, $\tilde{\mathcal{F}}_E$, is defined as the left-hand side of (3.8b). When calculating the heat release term (3.15) in the physics-informed loss, the time-delayed velocity at the flame can be predicted from the network as $\hat{u}'(x_f, t_k - \tau)$, i.e., by evaluating the network at $(x = x_f, t = t_k - \tau)$. For the GalNN, this is

$$\hat{u}'(x_f, t_k - \tau) = \sum_{j=1}^{n_g} \hat{\eta}_j(t_k - \tau)\cos(j\pi x_f).$$

Therefore, since the model takes time as an input for both FNN and GalNN architectures, there is no need to account for the time delay with a separate dummy variable. The evaluation of the heat-release model requires the system's parameters x_f, β, and τ to be known.

For the GalNN, as Galerkin amplitudes are available, the physical residuals of the Rijke tube can be directly computed from the ODEs (3.14) in these twin experiments. For the FNN, the calculation of heat release and modal damping in the physics-informed loss requires transformation of the predictions to the frequency domain and the following procedure.

The modal damping is defined as a multiplication in the frequency domain and a convolution in the spatial domain with the pressure. Using an FNN, we predict the pressure at a given time in the spatial domain. In order to compute the contribution of the damping in the physics-informed loss, at each training step, we transform the pressure predicted in the spatial domain to the frequency domain via Fast Fourier Transformation (FFT), compute the damping in the frequency domain, and then transform the result back to the spatial domain. When sampling from signals, the distinct Fourier frequencies (wavenumbers) are given as $\kappa_m = m \cdot 2\pi/(M\Delta x), m = 0, 1, \ldots, M/2$, where M is total number of samples, Δx is the length between samples, and $M\Delta x$ gives the length of the sampled signal. During simulation, we considered n_g Galerkin modes. So, in our case, we set $M \geq 2n_g$. Notice that the spatial domain is $[0, 1]$, while the wavenumbers of the Galerkin modes have a resolution of π, which is only half-period in this domain. Consequently, the resolution of the Fourier frequencies must be set to $2\pi/(M\Delta x) = \pi$ and hence, $M\Delta x = 2$, which means that the FFT must be taken over a domain of $[0, 2]$, such that the samples are at $x_m = m\frac{2}{M}, m = 0, 1, \ldots, M-1$ and we have one period of discrete samples. Predicting the pressure over the $x = [0, 2]$ domain could lead to errors as there is no training data in this region. Instead, we first predict over the original $x = [0, 1]$ domain and then stack this prediction with its negative symmetric with respect to $x = 1$. We have

$$\hat{p}' = \left\{ \hat{p}'(x_m, t) \,\middle|\, x_m = m\frac{2}{M},\ m = 0, 1, .., M-1,\ M \geq 2n_g \right\}, \tag{4.12}$$

and its Fourier transform $\mathscr{F}(\hat{p}')$. Now, we take only the part of $\mathscr{F}(\hat{p})$ that corresponds to the positive frequencies and element-wise multiply it with the damping modes ζ,

$$\chi_j = \mathscr{F}_j^+(\hat{p}')\zeta_j, \quad j = 1, 2, \ldots, n_g. \tag{4.13}$$

The training is done in batches, so the x locations we are interested in calculating the effect of

Table 4.2: The Rijke tube's parameters for the dataset used in the training of the proposed FNNs.

System's constants			
Galerkin modes (n_g)		10	
Chebyshev discretisation (n_c)		10	
Flame location (x_f)		0.2	
Damping coefficients		$d_1 = 0.1$, $d_2 = 0.06$	
Time delay (τ)		0.2	
Solver time step (Δt)		0.001	
Dynamical regime	Limit cycle	Quasiperiodic	Chaotic
Heat release strength (β)	5.7	6.1	7.0

Table 4.3: Summary of FNN model properties trained on the Rijke tube data. The configured hyperparameters are the number of hidden layers n_l, the activation function $\phi^{(l)}$, the number of neurons in each hidden layer n_{neuron}, the learning rate α_{learn}, and the batch size N_b.

Hyperparameter	ReLU FNN	tanh FNN	sin FNN	sin-ReLU FNN	GalNN	P-GalNN
n_l	5	5	3	2	2	1
$\phi^{(l)}$	ReLU	tanh	sin	sin-ReLU	sin	harmonics
n_{neuron}	96	96	32	64	16	40
α_{learn}	0.001	0.001	0.001	0.001	0.001	0.001
N_b			64			
Optimiser			Adam			

damping at may not necessarily collide with the spatial grid that we have previously constructed. So, we compute the inverse Fourier transform for these locations in the training batch

$$\zeta p'(x) = \frac{1}{2M} \sum_{j=1}^{n_g} \chi_j \exp(\mathrm{i}j\pi x) + \chi_j^* \exp(-\mathrm{i}j\pi x), \tag{4.14}$$

where χ^* denotes the complex conjugate of χ. We use the contribution of damping in the residual of the energy equation (3.8b) when computing the physics-informed loss for the energy equation (4.10b).

Although the heat-release term acts as a Dirac delta in the spatial domain, the solver implements Galerkin decomposition, which projects this term onto a truncated set of modes. Plugging back the decomposition (3.10) in the left-hand side of the energy equation (3.8b), the remaining heat release term is equal to $\sum_{j=1}^{n_g} 2\dot{q}'_f \sin(j\pi x_f) \sin(j\pi x)$, which is the effective term in the simulations. As the number of Galerkin modes approaches to infinity, this approaches the Dirac delta. However, from a practical point of view, the number of modes in the simulation is finite, thus we use this expression to find the contribution of heat release in the residual of the energy equation (3.8b) when computing the physics-informed loss for the energy equation (4.10b).

4.5.2 Extrapolation in time

We simulate the Rijke tube system (3.8) with the parameters; $n_g = 10$, $n_c = 10$, $x_f = 0.2$, $\beta = 5.7$, $\tau = 0.2$, $d_1 = 0.1$, $d_2 = 0.06$ (Tab. 4.2). (These numbers of Galerkin modes and Chebyshev

points provide a trade-off between accuracy and computational effort (Huhn and Magri, 2020).) After a transient period, this system settles onto a limit cycle with a period of ≈ 1.927 time units. The dataset is composed of $t_{\text{train}} = 4$ time units long (approx. 2 periods) pressure and velocity measurements from $n_o^p = n_o^u = 13$ sensor locations, which are uniformly distributed along the tube including the boundaries. Before training, the inputs are standardised to have a zero mean and unit variance.

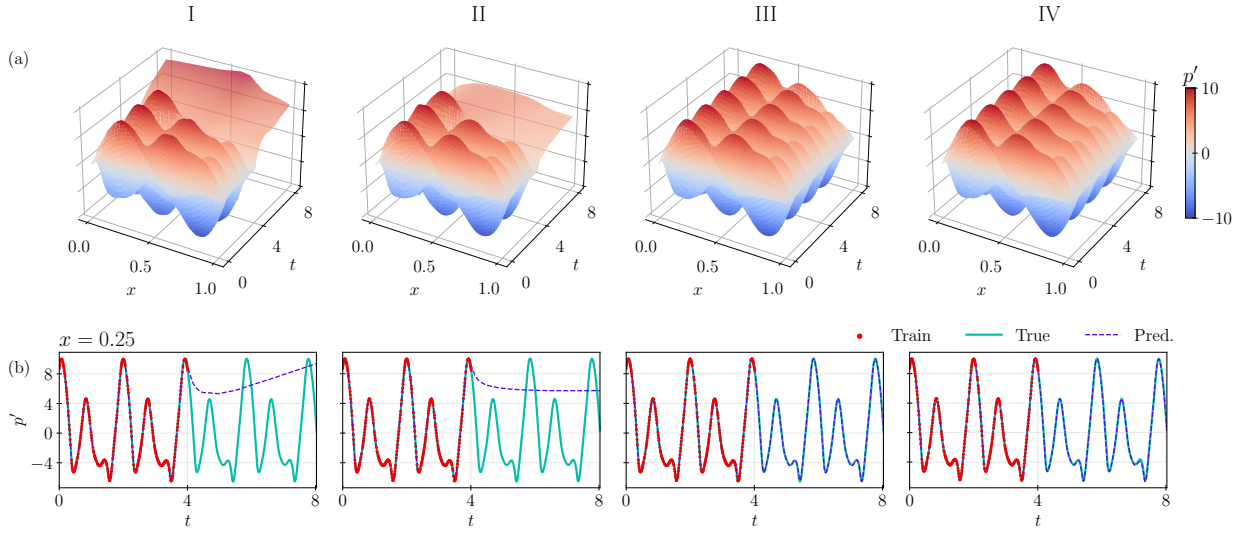


Figure 4.4: Extrapolation in time of the acoustic pressure with standard feedforward neural networks. Effect of activation functions: (I) ReLU, (II) tanh, (III) sine in all hidden layers, and (IV) sine-ReLU (sine in the first layer, ReLU for the rest of the hidden layers). Top row (a): predictions in both time and space. Bottom row (b): predictions in time at $x = 0.25$, training data, and ground truth. Typical activation functions, such as ReLU (I) and tanh (II), fail to extrapolate in time.

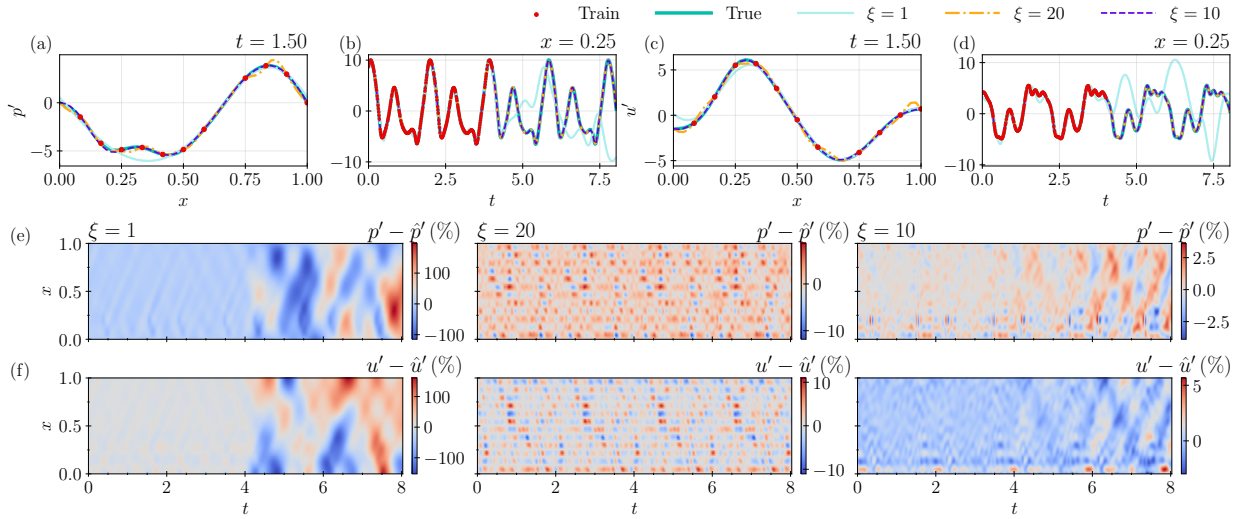


Figure 4.5: Effect of hyperparameter ξ in the sine activation $1/\xi \sin(\xi z)$. Shown, (a, c) spatial shapes at $t = 1.5$; (b, d) time series at $x = 0.25$; percentage prediction error of (e) pressure and (f) velocity, relative to their maxima over the spatiotemporal domain covering two periods beyond training. This hyperparameter imposes an inductive bias on the frequency content of predictions.

Standard feedforward neural networks

Figs. 4.4 (I) and (II) show the results of FNNs with ReLU and tanh activation functions when trained on this dataset. The network hyperparameters (number of layers, number of neurons, learning rate) have been tuned via a grid search, in which we choose the architectures that resulted in the smallest validation losses. The network properties are provided in Tab. 4.3. The ReLU network is initialised with the method of He et al. (2015), and the tanh network with the method of Glorot and Bengio (2010). As illustrated in Figs. 4.4 (I) and (II), FNNs with conventional activation functions fail at extrapolating periodic functions in time.

Fig. 4.4 (III) shows the extrapolation capability of an FNN equipped with sine activation, i.e., a periodically activated FNN (P-FNN) as introduced in § 4.4.1. (Note that when using the sine activation, the inputs are not normalised before training.) The activation function is modified to include a hyperparameter ξ such that it becomes

$$\phi(z) = \frac{1}{\xi}\sin(\xi z). \tag{4.15}$$

This hyperparameter is used to fine tune the frequency content of the learned functions. For this dataset, it is determined as $\xi = 10$ with a grid search. The effect of varying the hyperparameter ξ is shown in more detail in Fig. 4.5 for $\xi = 1, 10, 20$. Low ξ, i.e., $\xi = 1$, results in a low frequency model, whereas high ξ, i.e., $\xi = 20$, results in a high frequency model, which can especially be observed in the pressure and velocity shapes in the spatial domain. Physically, since the weights are initialised as $\sim \mathcal{U}(-1.22, 1.22)$ (§ 4.4.1), multiplying this by $\xi = 10$, provides a good initial guess on the order of magnitude of the angular frequency for the training and is thus chosen as the optimum value for this dataset. Since the sampling frequency of the training data is high enough, we do not observe the effect of high ξ in the time domain. Ziyin et al. (2020) reported similar findings in their studies when using the $1 + \sin^2(z)$ activation.

Fig. 4.4 (IV) shows the case for the sine-ReLU configuration, which can also extrapolate in time. Both sine and sine-ReLU networks can extrapolate also at locations where no sensor measurements are available. Deeper architectures are required when training with purely ReLU or tanh networks compared to sine or sine-ReLU networks. For example, in this case we use two layers of sine and sine-ReLU vs. five layers of ReLU and tanh, and even then the ultimate training and validation losses of sine and sine-ReLU networks were one order of magnitude smaller. To conclude, using periodic activations produces more physical and expressive neural network models for acoustics than the conventional activations, which can also extrapolate in time, whilst speeding up training and hyperparameter search as a consequence of the smaller network sizes required.

Physics-constrained networks and long-term extrapolation

We showed that networks equipped with periodic activations make accurate predictions on data points chosen in a time range right after the training (Fig. 4.4 (III-IV) and Fig. 4.6 (a)). Here, we analyse how the learned models extrapolate in the long term. The ideal model of a periodic solution should capture the periodicity that the system exhibits for all times. Network predictions over a long time period are plotted in Fig. 4.6 (b) and (d), for P-FNN and GalNN, respectively, when trained on data from approximately two periods. Over a long time range, we observe a beating-like

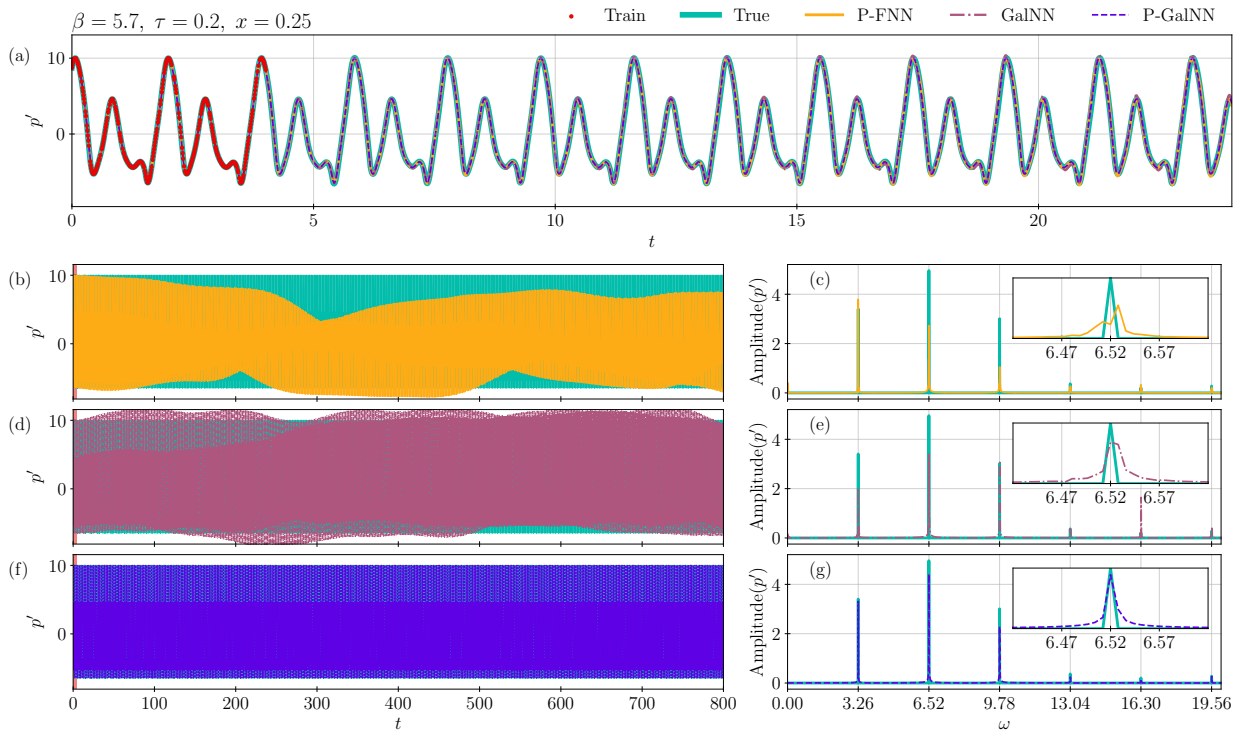


Figure 4.6: Long-term extrapolation in time of periodic acoustic pressure time series with acoustic neural networks; (b, c) periodically activated feedforward neural network (P-FNN), (d, e) Galerkin neural network (GalNN), and (f, g) periodic Galerkin neural network (P-GalNN). Shown $p'(x = 0.25, t)$ (a) Short-term predictions, (b, d, f) long-term extrapolations (the training data is marked by the shaded red area), (c, e, g) amplitude spectra compared to the truth. Constraining prior knowledge on periodicity is key to an accurate long-term prediction.

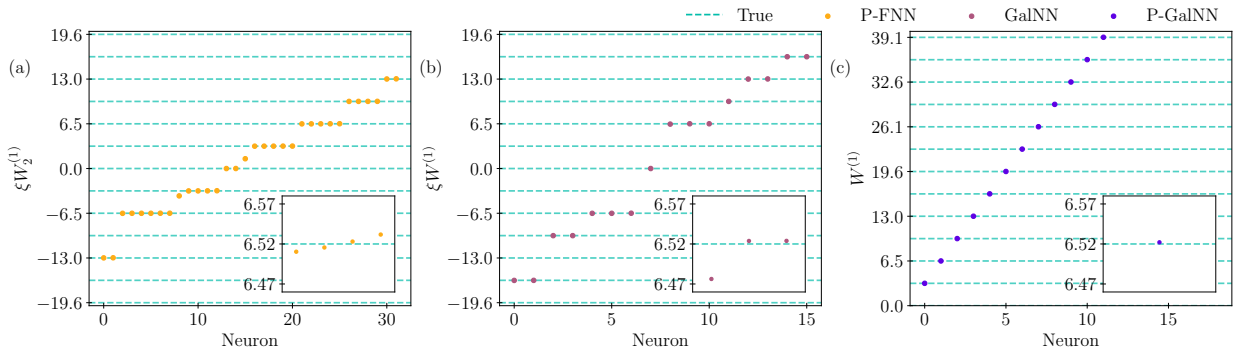


Figure 4.7: In reference to Fig. 4.6, trained weights of the first hidden layers of (a) P-FNN, (b) GalNN, and (c) P-GalNN. These weights are multiplied with the time input before periodic activation, i.e., $\sin(\xi \boldsymbol{W}_1^{(1)} x + \xi \boldsymbol{W}_2^{(1)} t + \xi \boldsymbol{b}^{(1)})$ for P-FNN, $\sin(\xi \boldsymbol{W}^{(1)} t + \xi \boldsymbol{b}^{(1)})$ for GalNN, where ξ is a hyperparameter, and $\sin(\boldsymbol{W}^{(1)} t)$ for periodic GalNN.

phenomenon. Beating happens when sinusoids with frequencies that are very close to each other but not exactly the same are summed.

In the frequency spectra of the acoustic variables, the peaks are at the harmonics of the angular frequency of the limit cycle, which we determined as $\omega^* = 3.2605$ by inspection of the frequency spectrum and autocorrelation of the time series. After the training, this is captured by the weights in the first hidden layer as discussed in § 4.4.4. These weights are multiplied with the time input

before periodic activation, i.e., $\sin(\xi \boldsymbol{W}_1^{(1)} x + \xi \boldsymbol{W}_2^{(1)} t + \xi \boldsymbol{b}^{(1)})$ for P-FNN, and $\sin(\xi \boldsymbol{W}^{(1)} t + \xi \boldsymbol{b}^{(1)})$ for GalNN, where ξ is the hyperparameter of the sine activation (4.15). The optimised weights are placed close to the harmonics of the angular frequency of the limit cycle. Fig. 4.7 shows the (sorted) weights of the P-FNN and GalNN. The harmonics of the angular frequency of the limit cycle are shown in dashed horizontal lines. Although these weights are close to the harmonics of the frequency of the limit cycle, they are not exact integer multiples of each other. Even the weights clustered around one frequency slightly differ from each other. This can be observed in the amplitude spectrum of the predicted time series as well, shown in Fig. 4.6 (c) and (e). These observations align with the beating behaviour.

We overcome this issue by using the GalNN tailored for this specific purpose as described in § 4.4.4. The long term predictions are shown in Fig. 4.6 (f) and the amplitude spectrum of the predicted signal in Fig. 4.6 (g). (For the periodic Galerkin neural network, one can also directly compute the amplitude spectrum of the Galerkin variables from the weights of the output layer. The ℓ_2-norm of the weights that multiply the sine and cosine activations at the same angular frequency gives the contribution of that frequency in the spectrum.) The estimated ω^* by the periodic GalNN is 3.2615, which aligns with the true signal. The trained weights, which are activated as $\left[\sin(\boldsymbol{W}^{(1)} t); \cos(\boldsymbol{W}^{(1)} t)\right]$ in the periodic GalNN, are shown in Fig. 4.7 (c). Increasing the training time improves the extrapolation ability for both P-FNN and GalNN. However, with the constrained structure of the GalNN, we eliminate the beating behaviour and achieve accurate long-term extrapolation without using more training data.

As discussed in § 3.4.4, in addition to limit cycles, thermoacoustic systems can also exhibit quasiperiodic and chaotic oscillations through bifurcations (e.g., Kabiraj et al., 2012). In the low-order model, the bifurcation parameters of the Rijke tube are the heat source location, x_f, heat-release strength, β, and time-delay, τ. With $x_f = 0.2$ and $\tau = 0.2$ and increasing β, the system takes a Ruelle-Takens-Newhouse route to chaos (Huhn and Magri, 2020); at $\beta = 0.5$, it bifurcates from a fixed point solution to a limit cycle, at $\beta = 5.8$ from a limit cycle to a quasiperiodic attractor, and at $\beta = 6.5$ from a quasiperiodic attractor to chaos (§ 3.4.4). We collect quasiperiodic data from $\beta = 6.1$ regime and chaotic data from $\beta = 7.0$ regime. For the quasiperiodic system, we train a GalNN with 2 sine layers of 96 neurons on data from the 80 time units long time series following a hyperparameter tuning with a grid search. The predictions are shown in Fig. 4.8 (a – c) with the amplitude spectrum. The Galerkin network can capture the quasiperiodic behaviour and the frequency spectrum correctly. For the chaotic system, we train a GalNN with 4 sine layers of 256 neurons on data from the 200 time units long time series. The predictions are shown in Fig. 4.8 (d – f) along with the amplitude spectrum. To train on chaotic data, we required a longer time series and a deeper and wider network because the nonlinear dynamics is richer than the quasiperiodic solution. The network's prediction correctly captures the dominant frequencies and the amplitude of the true signal.

4.5.3 Learning from partial state measurements

Full state measurements are difficult to perform in thermoacoustics, i.e., only pressure measurements are taken with microphones. To emulate such a scenario, we perform the training using only pressure data, discarding the velocity data. Therefore, in the formulation of the data-driven

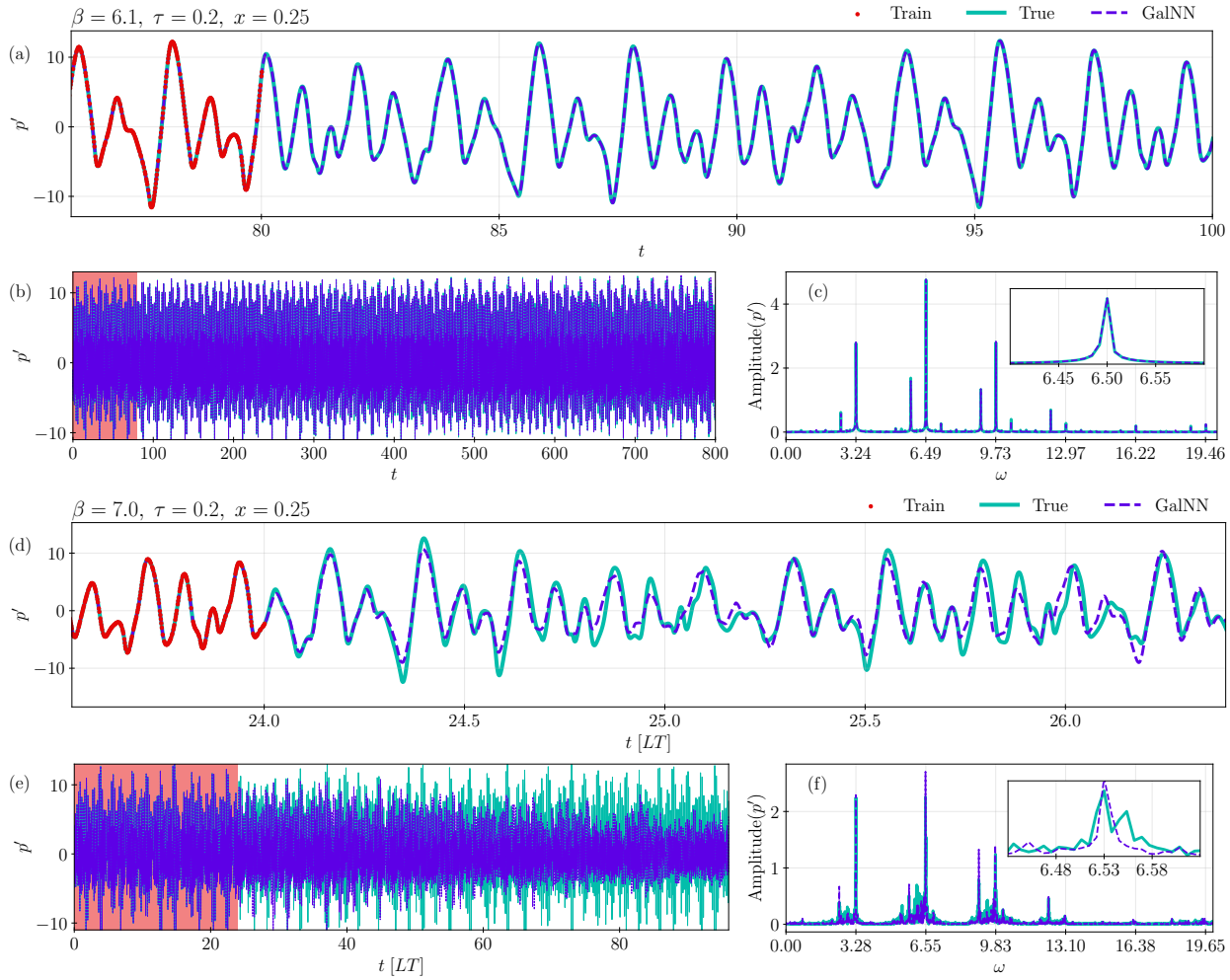


Figure 4.8: Extrapolation in time for non-periodic solutions with GalNN. (a-c) Quasiperiodic solution; and (d-f) chaotic solution for the acoustic pressure $p'(x = 0.25, t)$. For the chaotic time series, the time axis is normalised by the Lyapunov time (LT) of the system, where LT $\approx$ 8.33 time units in this case. Training data shown within the red vertical line (80 time units for the quasiperiodic case, 200 time units (24 LT) for the chaotic case).

loss (4.7), we have $\mathcal{I}_o = 1$ or $n_o^u = 0$. Our aim is to learn a model that can also output velocity information without ever being exposed to measurements from it during training. Fig. 4.9 shows the predictions of the P-FNN and the GalNN trained with and without physics information. When the loss is computed solely from the data, only the pressure can be learned. On the other hand, velocity predictions are made possible by enforcing the physics via the physics-informed loss term. For the purely data-driven networks, the output weights related to velocity are not updated, hence the velocity predictions are made with the initial weight values which are small. When both momentum and energy equations are imposed, both physics-informed networks can accurately reconstruct the velocity.

4.6 Results on higher-fidelity data

In order to assess how well the GalNN approach generalises to higher-fidelity data, which is not generated by the Galerkin method, we employ a higher-fidelity model (§ 3.4.5) that includes the effects of the mean flow and has a kinematic flame model for the flame (Dowling, 1999). We

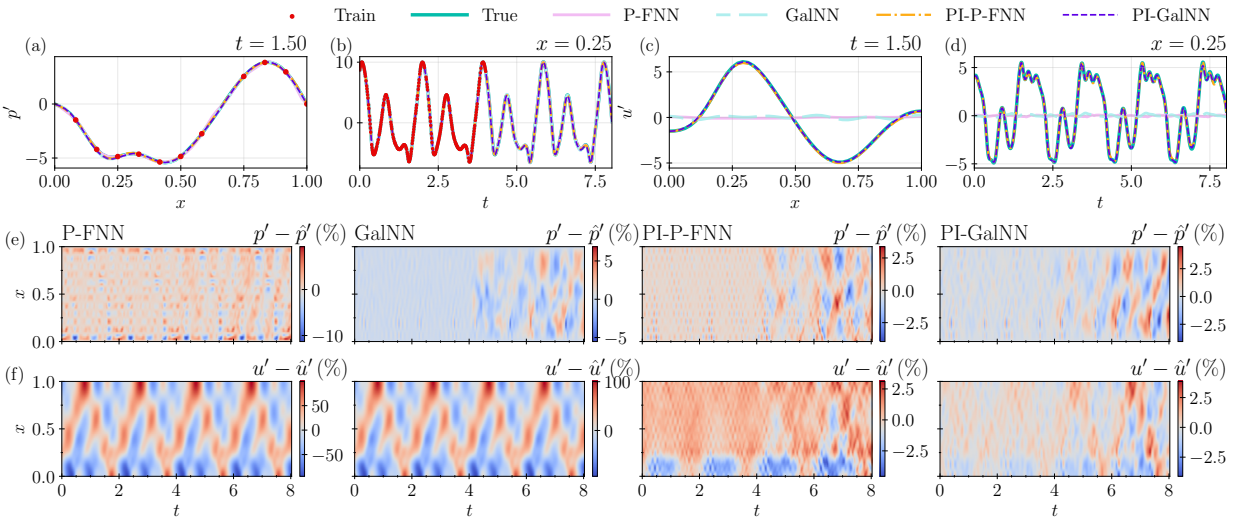


Figure 4.9: Reconstruction of acoustic velocity from pressure measurements only. Predictions on acoustic (a, b) pressure and (c, d) velocity from only $n_o^p = 13$ pressure observations with different physics-constrained networks. Shown, (a, c) spatial shapes at $t = 1.5$; (b, d) time series at $x = 0.25$; percentage prediction error of (e) pressure and (f) velocity, relative to their maxima over the spatiotemporal domain covering two periods beyond training. The acoustic velocity is accurately reconstructed from pressure measurements with the physics-informed periodically activated feedforward neural network (PI-P-FNN) and Galerkin neural network (PI-GalNN).

simulate a system with ideal boundary conditions, i.e., $\Gamma_{\text{in}} = \Gamma_{\text{out}} = -1$, that results in limit-cycle oscillations using the mean-flow parameters for the inlet velocity $\tilde{\bar{u}}_1 = 4\, m/s$; for the inlet temperature $\tilde{\bar{T}}_1 = 293\, K$; for the inlet and outlet pressure $\tilde{\bar{P}}_1 = \tilde{\bar{P}}_2 = 101300\, Pa$; and for the heat release rate $\tilde{\bar{Q}} = 2000\, W$. This is a low Mach number configuration with $\overline{Ma}_1 = 0.0117$ and $\overline{Ma}_2 = 0.0107$. The cross-sectional area changes at the flame location with a ratio of $A_1/A_2 = 0.75$. The dimensional sampling frequency of the system, $\tilde{f}_s$, is 10 kHz.

In the preprocessing of the data, we nondimensionalise the variables as, $x = \tilde{x}/\tilde{L}$; $t = \tilde{t}\tilde{\bar{c}}_{ref}/\tilde{L}$; $u = \tilde{u}/\tilde{\bar{c}}_{ref}$; $\rho = \tilde{\rho}/\tilde{\bar{\rho}}_{ref}$; $p = \tilde{p}/(\tilde{\bar{\rho}}_{ref}\tilde{\bar{c}}^2_{ref})$; $\dot{q}_f = \tilde{\dot{q}}_f(\gamma - 1)/(\tilde{\bar{\rho}}_{ref}\tilde{\bar{c}}^3_{ref})$ where we choose $\tilde{(\cdot)}_{ref} = (\tilde{L}_1\tilde{(\cdot)}_1 + \tilde{L}_2\tilde{(\cdot)}_2)/\tilde{L}$ as the weighted average of the mean values of variable $(\cdot)$ across the flame. The nondimensional flame location is at $x_f = 0.61$, and the nondimensional densities in the duct segments before and after the flame are $\bar{\rho}_1 = 1.145$ and $\bar{\rho}_2 = 0.769$. These quantities are used to determine the acoustic mode shapes for the GalNNs (3.11) (Fig. 4.10 (a) and (b)).

While creating the training and validation sets, we consider two types of validation; (i) validation of the interpolation capability, and (ii) validation of the extrapolation capability, i.e., when the data is sampled from outside the time range of training. A good validation of the interpolation loss prevents overfitting. Validation of extrapolation loss indicates how well the model generalises beyond the time range of training. After discarding a transient, we split the first 4 nondimensional time units long time series (approximately two periods) of the limit cycle into training and validation of the interpolation sets, assigning a 80% to 20% ratio between them. The validation data for the extrapolation is then taken as the next 4 nondimensional time units.

The physics-informed loss of the higher-fidelity system is calculated by evaluating the left-hand side of the PDEs (3.6) after being nondimensionalised as described above. The momentum residual,

$\tilde{\mathcal{F}}_M$, is defined as the left-hand side of (3.6b). In contrast to the twin experiments with Rijke tube, for the higher-fidelity data, we assume that we do not have access to the heat release model. Hence, the energy residual, $\tilde{\mathcal{F}}_E$, is defined as the left-hand side of (3.6c) excluding the heat release term, $\dot{q}_f$. This is acceptable because the heat release term only acts at the flame location due to the compact assumption and the Dirac delta.

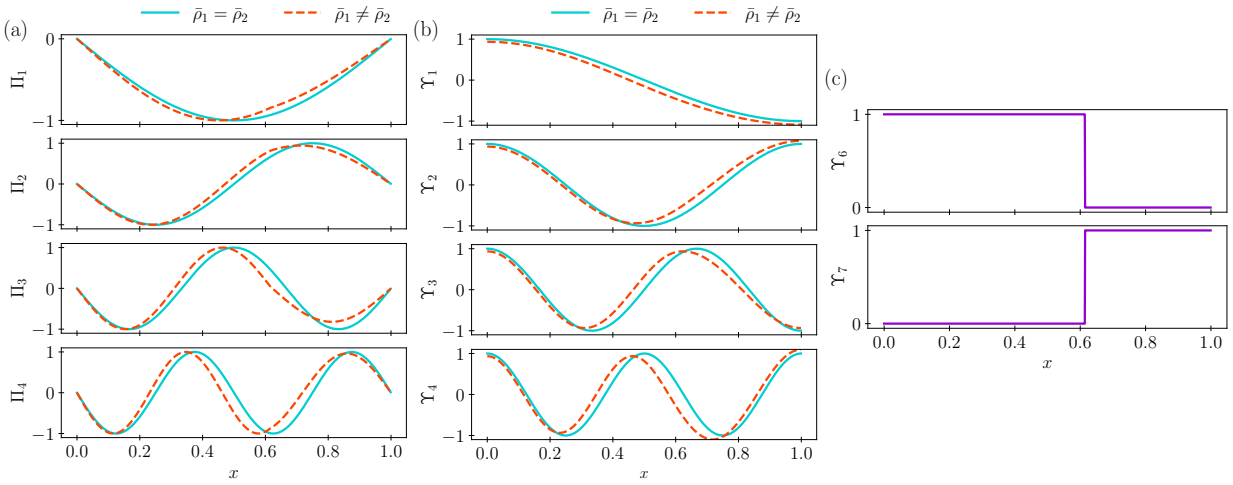


Figure 4.10: Acoustic eigenfunctions, which span the solution space of the GalNN. (a) Pressure 3.11a, (b) velocity 3.11b, and (c) step function that model the jump discontinuity 4.11. Acoustic eigenfunctions with $\kappa_j = j\pi$, i.e., when $\bar{\rho}_1 = \bar{\rho}_2$, are shown in straight lines; acoustic eigenfunctions when $\bar{\rho}_1 = 1.145, \bar{\rho}_2 = 0.769$, i.e., $\bar{\rho}_1 \neq \bar{\rho}_2$ are shown in dashed lines.

4.6.1 Comparison of networks performance for reconstruction

Table 4.4: Summary of FNN and P-GalNN model properties trained on higher-fidelity data. FNNs with different activations; ReLU, sine, sine-ReLU (sine in the first layer, ReLU in the rest). P-GalNNs with different choices of spatial bases; (i) natural acoustic modes (3.11) for $\bar{\rho}_1 = \bar{\rho}_2 = 1$, (ii) natural acoustic modes for $\bar{\rho}_1 = 1.145, \bar{\rho}_2 = 0.769$; and (iii) case (ii) with the addition of step functions in the velocity modes to handle the jump discontinuity (4.11).

Hyperparameter	ReLU FNN	sine FNN	sine-ReLU FNN	P-GalNN (i)	P-GalNN (ii)	P-GalNN (iii)
n_l	6	3	4	1	1	1
n_g	—	—	—	20	20	20
$\phi^{(l)}$	ReLU	sin	sin-ReLU	harmonics	harmonics	harmonics
n_{neuron}	64	64	32	20	20	20
α_{learn}	0.01	0.01	0.01	0.0004	0.0004	0.0004
N_b				32		
Optimiser				Adam		

We analyse the performance of the different networks that we have built in § 4.4 on the reconstruction of the higher-fidelity data. We compare three configurations of the feedforward neural network with; (i) ReLU activations in all hidden layers, (ii) sine activation in all hidden layers, (iii) sine activation in the first layer, ReLU activation in the rest of the hidden layers, and three configurations of the periodic Galerkin neural network with one hidden trainable frequency layer; (i) with $\kappa_j = j\pi$ as the acoustic angular frequencies as before, i.e., when $\bar{\rho}_1 = \bar{\rho}_2 = 1$, (ii) with κ_j

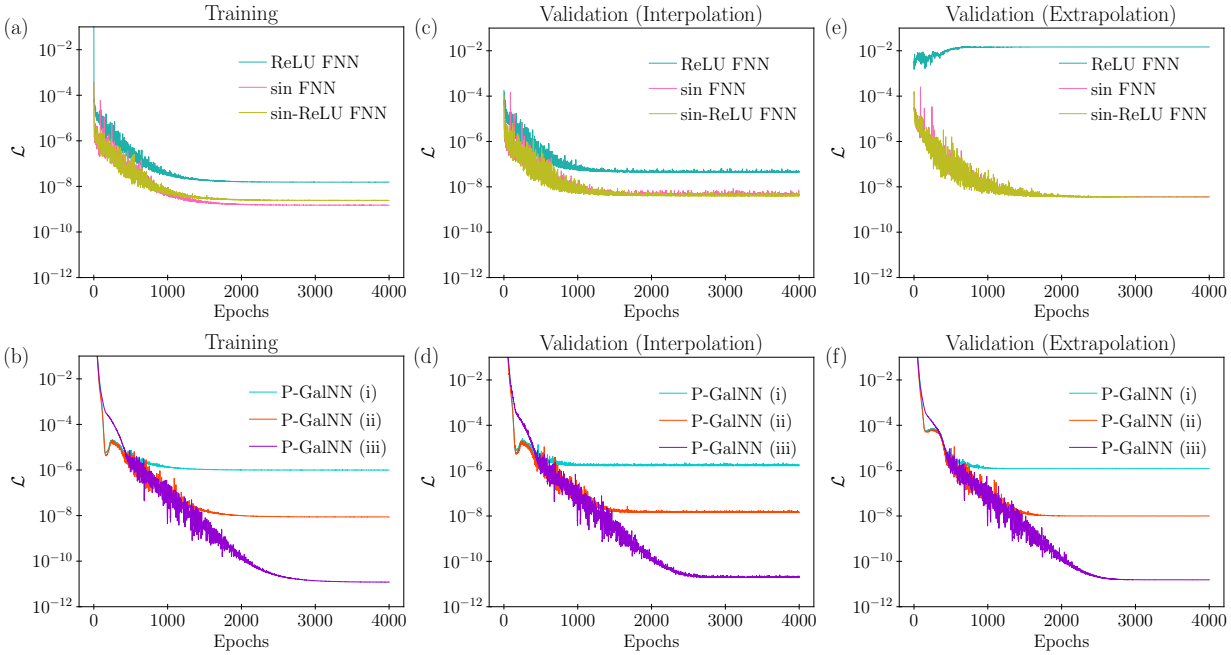


Figure 4.11: Training on data from higher-fidelity model for a periodic acoustic solution. Loss-function histories. (a, b) Training, (c, d) validation of interpolation, and (e, f) validation of extrapolation. Feedforward neural networks (FNNs) with different activations; ReLU, sine, sine-ReLU (sine in the first layer, ReLU in the rest). Period Galerkin neural networks (P-GalNNs) with different choices of spatial bases; (i) natural acoustic modes (3.11) for $\bar{\rho}_1 = \bar{\rho}_2 = 1$, (ii) natural acoustic modes for $\bar{\rho}_1 = 1.145, \bar{\rho}_2 = 0.769$; and (iii) case (ii) with the addition of step functions in the velocity modes to handle the jump discontinuity (4.11).

determined from the dispersion relationship using the real mean-flow density values of the system, i.e., when $\bar{\rho}_1 = 1.145, \bar{\rho}_2 = 0.769$, (iii) the Galerkin modes chosen as (ii) with the addition of step functions in the velocity modes to handle the jump discontinuity (4.11). These modes are illustrated in Fig. 4.10

For each configuration of activations, the hyperparameters of the architecture and training (number of layers, number of neurons, and learning rate) have been tuned with a grid search, in which we choose the set of hyperparameters that resulted in the smallest validation losses. These properties are provided in Tab. 4.4 for each network.

The training and validation loss histories of all models are shown in Fig. 4.11. The ReLU network has higher training and interpolation losses, cannot fit the data as well as the sine and sine-ReLU networks, and cannot extrapolate. For all the other networks (sine FNN, sine-ReLU FNN, P-GalNN (i), P-GalNN (ii), P-GalNN (iii)), the validation losses are close to the training loss. The P-GalNN (iii) scores the lowest losses. The periodic Galerkin networks can have similar or better prediction performance as the other deeper feedforward neural networks, even with one hidden layer following a physical choice of spatial basis.

Fig. 4.12 visualises the velocity predictions and their errors between the predictions and the true data obtained with the networks. The P-GalNN (i) exhibits an error pattern that resembles an offset from the true solution, which indicates a biased model. The prediction is also oscillatory and the error is especially high at the boundaries. The more physically motivated choice of P-GalNN

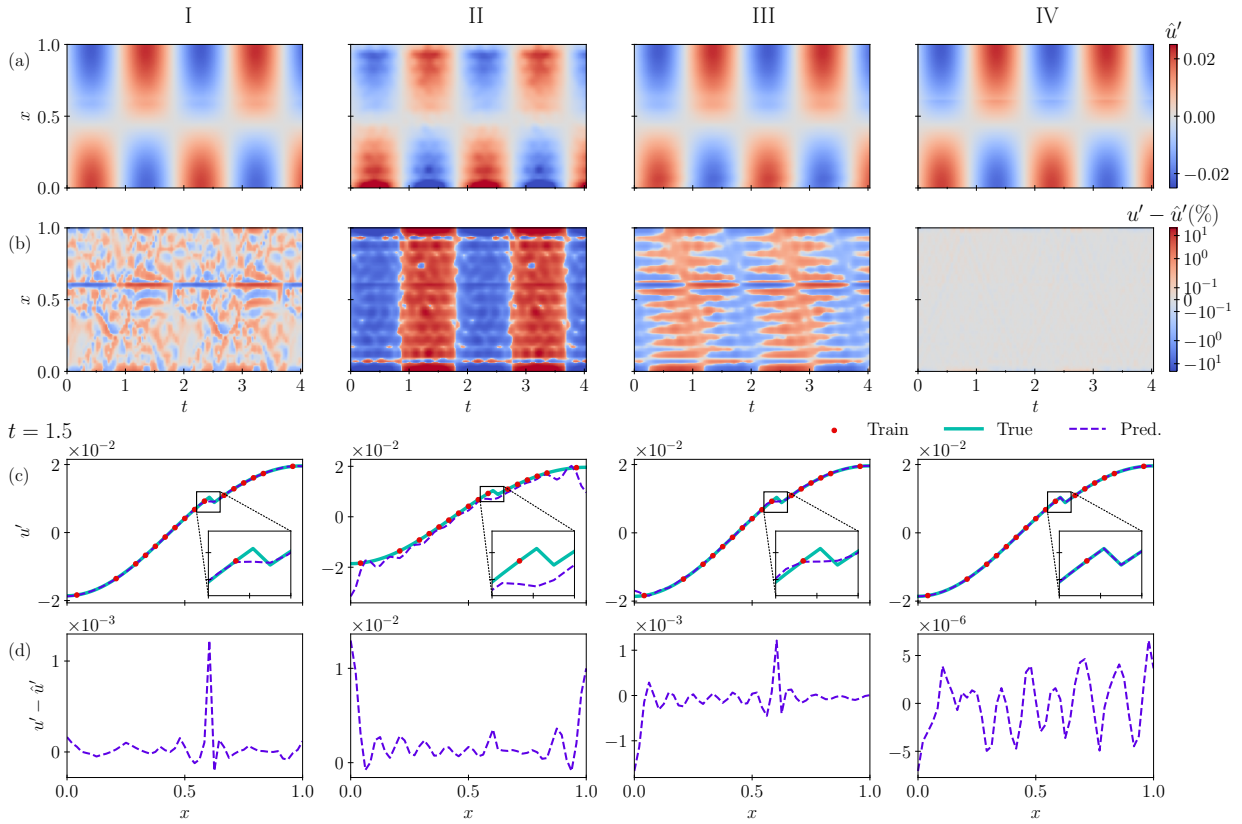


Figure 4.12: Comparison of acoustic neural networks in terms of reconstruction performance from higher-fidelity data. From left to right, (I) sine-ReLU FNN, i.e., sine in the first layer, ReLU in the rest; (II) P-GalNN (i) with natural acoustic modes (3.11) for $\bar{\rho}_1 = \bar{\rho}_2 = 1$; (III) P-GalNN (ii) with natural acoustic modes for $\bar{\rho}_1 = 1.145, \bar{\rho}_2 = 0.769$; (IV) P-GalNN (ii) with the addition of step functions in the velocity modes to handle the jump discontinuity (4.11). From top to bottom, (a) velocity prediction in time and space, (b) percentage error between ground truth and prediction in time and space relative to the maximum, (c) velocity prediction and (d) error at a fixed time instance, $t = 1.5$.

(ii), eliminates these errors, however the discontinuity jump still cannot be captured, which leads to high frequency oscillations in the prediction around the flame location. A similar phenomenon is also observed for the sine-ReLU network, where the error is the highest at the flame location. This problem is overcome by P-GalNN (iii), which results in the smallest training and validation errors overall. This network finds the nondimensional dominant angular frequency as $\omega^* = 3.3164$ and thus, the period as $T^* = 1.8946$, which match closely with the angular frequency $\omega^* = 3.3166$ and period $T^* = 1.8945$ of the true pressure signal. The periodic Galerkin network successfully learns the correct period.

4.6.2 Robustness to sparsity and noise

In this section, we demonstrate the robustness of the acoustic neural networks to sparsity and noise in the training data. We select P-GalNN (iii) from the previous section because of its more generalisable performance in comparison to the other architectures and investigate the regularisation methods that can be used in order to prevent overfitting in the case of corrupted or scarce data. We compare a physics-based regularisation, i.e., physics-informed loss, with ℓ_1- and ℓ_2-norm regu-

larisations, which act on the network weights as $||\boldsymbol{W}||_1$ and $||\boldsymbol{W}||_2$ respectively in the loss term. Whilst ℓ_2-norm prevents overfitting by reducing the magnitudes of the weights, ℓ_1-norm induces sparsity.

We add zero mean Gaussian noise on the nondimensional training data with a standard deviation equal to $\sigma_o = 10\%$ of the standard deviation of the true solution function over the whole domain (20 dB signal-to-noise ratio). Using the described regularisation methods, we train networks with 10 Galerkin modes on this new dataset this time given only on a coarse grid. The comparison of the obtained predictions when using (i) no regularisation, (ii) ℓ_2-norm, (iii) ℓ_1-norm, and (iv) physics-based regularisation, which imposes a penalty on the training based on the residuals of the momentum and energy equations, is shown in Fig. 4.13. We train multiple networks with

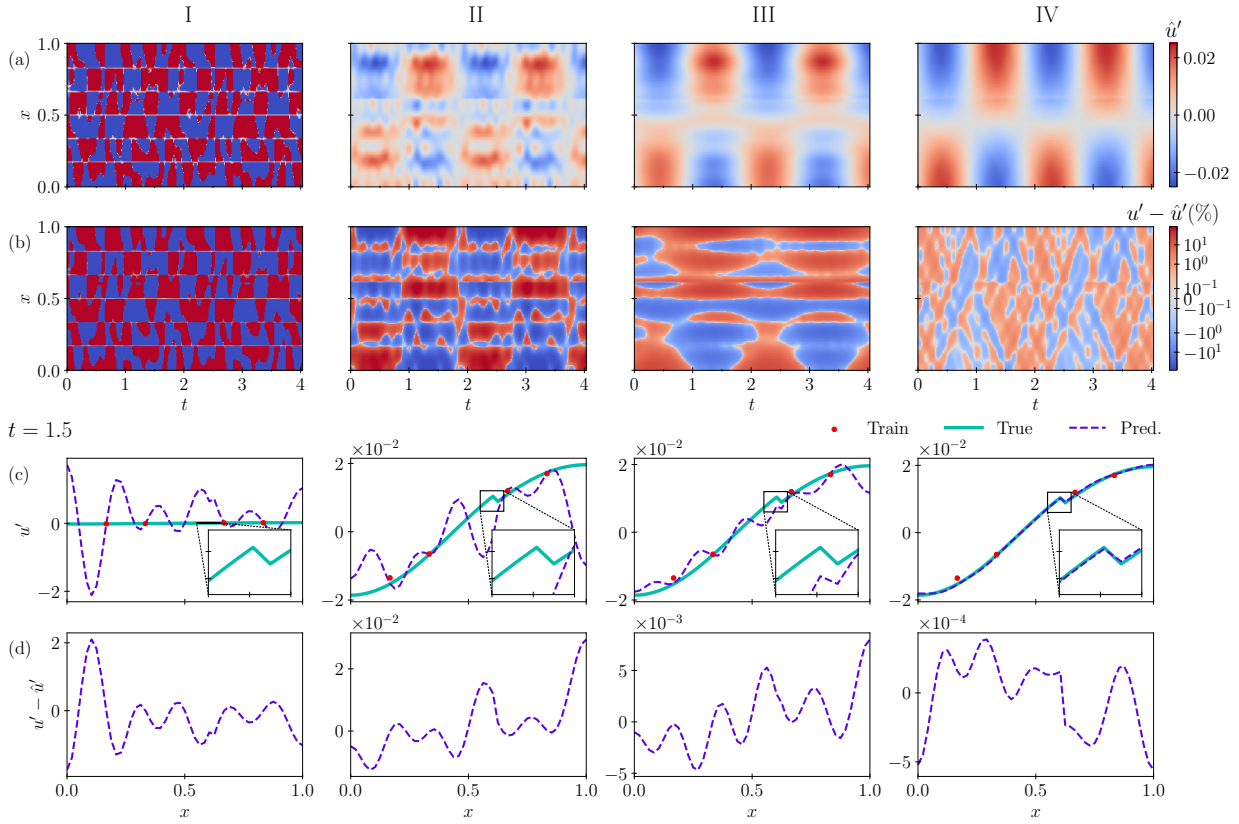


Figure 4.13: In reference to Fig. 4.12, effect of the regularisation on P-GalNNs in case of sparse and noisy measurements. From left to right, (I) no regularisation, (II) ℓ_2-norm regularisation, (III) ℓ_1-norm regularisation, and (IV) physics-based regularisation. From top to bottom: (a) acoustic velocity, (b) percentage error between ground truth and prediction in time and space relative to the maximum, (c) velocity prediction at a fixed time instance, and (d) error at a fixed time instance, $t = 1.5$.

varying values of regularisation coefficients and show only the results for those with the smallest validation losses. Without regularisation, the model can fit the training data perfectly, however the interpolation points show spikes, which is a clear demonstration of overfitting. This case of overfitting emerges when there is a discrepancy between the number of sensors and the number of Galerkin modes dictated by Nyquist-Shannon sampling theorem. The highest number of Galerkin

modes is given as

$$n_g^\star = \arg\max_j \{j \mid \kappa_j\sqrt{\bar{\rho}_1} \leq \kappa_{Nyquist},\ \kappa_j\sqrt{\bar{\rho}_2} \leq \kappa_{Nyquist}\} \tag{4.16}$$

where $\kappa_{Nyquist} = \pi/\Delta x$ with Δx being the spacing between sensor locations. For the given data, the nondimensional spacing in the x-domain is $\Delta x = 0.1667$. Nyquist-Shannon sampling theorem dictates the highest wavenumber that we can sample with this spacing, $m_{Nyquist} = \pi/\Delta x = 18.8496$. The wavenumbers of the Galerkin modes are determined a priori from the dispersion relationship. For this system, we find $\{\kappa_5\sqrt{\bar{\rho}_1}, \kappa_5\sqrt{\bar{\rho}_2}\} = \{16.7795, 13.7531\}$ and $\{\kappa_6\sqrt{\bar{\rho}_1}, \kappa_6\sqrt{\bar{\rho}_2}\} = \{20.2816, 16.6235\}$, and thus the highest number of Galerkin modes that can be reconstructed in both up- and downstream regions $n_g^\star = 5$.

The physics information significantly eliminates overfitting and outperforms ℓ_1- and ℓ_2-norm regularisations. The ℓ_2-norm regularisation does not work as well because the weights are not selectively regularised, i.e., dominant modes/wavenumbers will be regularised in the same way as non-dominant ones. The ℓ_1-norm regularisation performs better since modes corresponding to higher wavenumbers are discarded as ℓ_1-norm promotes sparsity and in the physical basis, the energy is contained in the low-wavenumber modes.

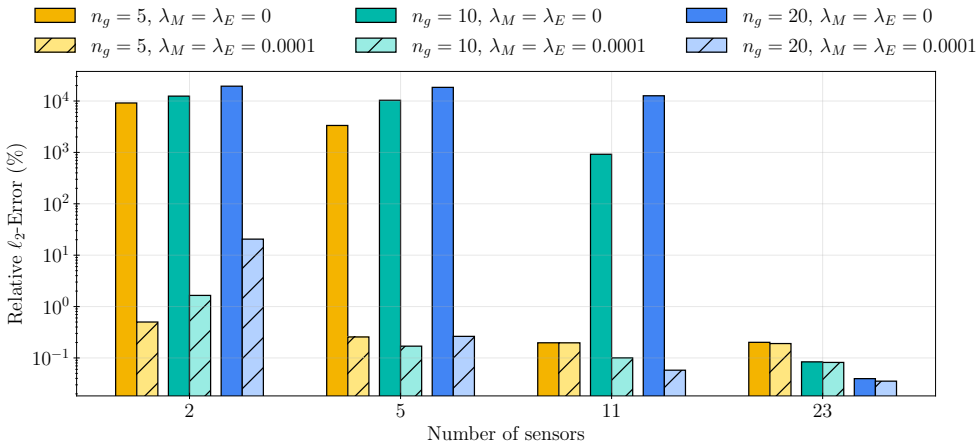


Figure 4.14: Effect of number of pressure sensors and Galerkin modes in the GalNN with and without physics-based loss. (The regularisation hyperparameter of the data-driven loss is $\lambda_D = 1$.)

We perform a quantitative study to investigate the robustness of the prediction performance to sparsity and noise. First, we vary the number of sensors along the tube. Fig. 4.14 shows the relative ℓ_2-error achieved by Galerkin networks with different number of modes when trained on noise-free data given on spatial grids of $n_o^p = n_o^u = \{2, 5, 11, 23\}$. (Note that the number of Galerkin modes in the figure does not include the additional two jump modes for velocity.) The relative ℓ_2-error is calculated from the prediction on the fine grid of 49 points over the training range by dividing the ℓ_2-norm of the error by the ℓ_2-norm of the ground truth. The networks are trained with and without physics-information in the loss function. We fix the weighting of the data-driven loss to $\lambda_D = 1$, and the physics loss to $\lambda_M = \lambda_E = 0.0001$ for physics-informed and to $\lambda_M = \lambda_E = 0$ otherwise. As discussed above, high number of Galerkin modes leads to overfitting for coarse grids, when the Nyquist wavenumber is not sufficient to resolve the high wavenumbers,

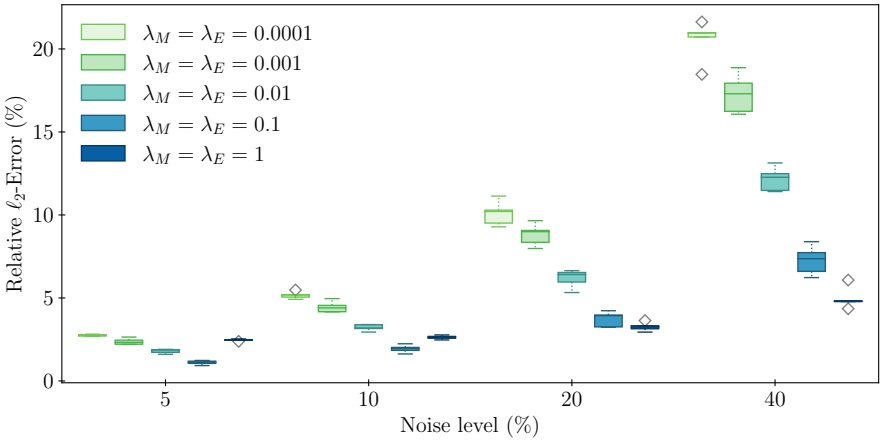


Figure 4.15: Robustness to noise. GalNNs with different regularisation hyperparameters for physics-based losses with data corrupted with noise. (The regularisation hyperparameter of the data-driven loss is $\lambda_D = 1$.)

e.g., $n_o^p = n_o^u = 5$ sensors and $n_g = 10$ Galerkin modes. Furthermore, we observe large prediction error even when the Nyquist condition is satisfied, e.g. $n_o^p = n_o^u = 5$ sensors and $n_g = 5$ Galerkin modes. We find that this type of error is mostly concentrated at the boundaries of the velocity, as we have not provided any boundary data and therefore, the model overfits to the rest of the data, not generalizing well over the boundaries. For pressure, this is not an issue, because the boundary conditions are Dirichlet and already encapsulated within the provided Galerkin basis. On the other hand, low number of Galerkin modes may not be enough to approximate the pressure and velocity shapes in fine detail, e.g., $n_o^p = n_o^u = 11$ sensors and $n_g = 5$ Galerkin modes. The addition of the physics-information shows a marked difference and helps overcoming these limitations. Fig. 4.17 shows the reconstruction errors associated with physics-informed training with varying numbers of pressure sensors.

Next, we choose the network with $n_g = 10$ modes and add increasing levels of noise to the data from $n_o^p = n_o^u = 5$ sensor measurements. We take 5 different realisations of noise and show the prediction relative ℓ_2 error as a box plot in Fig. 4.15 for noise levels of $\sigma_o = \{5\%, 10\%, 20\%, 40\%\}$ (26, 20, 14, 8 dB signal-to-noise ratios). Noting that even in the noise-free case, the non-physics-informed network performs poorly, we focus on physics-informed training with varying weightings of the physics loss. As the noise level increases, the optimum weighting of the physics-information increases as well, since the data becomes less reliable. We obtain good predictions of pressure and velocity over the entire domain, even though the data is sparse and noisy. The network is also robust to different realisations of noise.

We conclude by demonstrating how the physics-informed training performs in the absence of velocity data. For this purpose, we examine the case of the PI-P-GalNN with $n_g = 10$ Galerkin modes trained on data from $n_o^u = 5$ sensor measurements and $\sigma_o = 10\%$ noise. The prediction results of a physics-informed periodic Galerkin neural network (PI-P-GalNN) are compared with a sine-ReLU physics-informed feedforward neural network (PI-P-FNN) in Fig. 4.16. For the PI-P-GalNN, we report the following relative ℓ_2-errors: 2.63 % for pressure and 2.75 % for velocity in the training range ($t = 0 - 4$ time units), and 2.84 % for pressure and 2.80 % for velocity in the extrapolation range ($t = 4 - 8$ time units). In comparison, for the sine-ReLU feedforward neural network, we

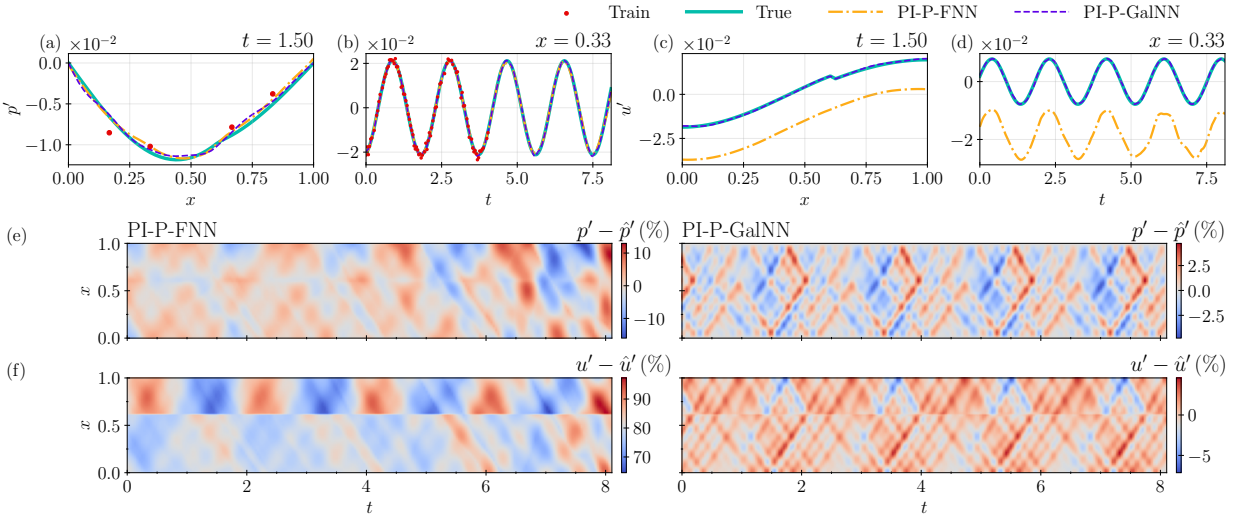


Figure 4.16: Reconstruction of acoustic velocity from only noisy pressure measurements. Predictions on acoustic (a, b) pressure and (c, d) velocity from only $n_o^p = 5$ pressure observations with physics-based loss using the periodically activated feedforward neural network (PI-P-FNN) and periodic Galerkin neural network (PI-P-GalNN). Shown, (a,c) pressure and velocity shapes along the tube at a fixed time instance, $t = 1.5$; and (b,d) pressure and velocity time series at a fixed point in the tube, $x = 0.33$; percentage prediction errors of (e) pressure and (f) velocity, relative to their respective maxima over the spatiotemporal domain covering two periods beyond training. PI-P-GalNN successfully reconstructs the velocity field.

report the following relative ℓ_2-errors: 4.81 % for pressure and 164.22 % for velocity in the training range ($t = 0 - 4$ time units), and 9.77 % for pressure and 160.57 % for velocity in the extrapolation range ($t = 4 - 8$ time units). With the PI-P-GalNN, we can reconstruct the velocity as well as pressure. The PI-P-FNN fits to the noisy data more compared to the PI-P-GalNN, and can only recover the velocity with an offset, completely discarding the velocity jump. A constant offset of the velocity, $\hat{u}'(x,t) = u'(x,t) + c$, satisfies the governing equations (3.6). Because the partial derivatives of the constant offset will be zero, the residual of the governing equations will not be affected. If the velocity fluctuations are known to have a zero mean, one strategy could be to remove the mean of the predictions in the postprocessing, and a second strategy could be to remove the biases from the output layer of the neural network. However, even when the offset is removed, the PI-P-FNN cannot capture the jump of the velocity at the flame location.

4.6.3 Inferring heat release

Once the model is trained on pressure and velocity data, the heat release rate can be inferred from the energy jump condition (3.7c) as a consequence via an algebraic calculation. For this purpose, we evaluate the pressure and velocity on either side of the flame using the trained model, i.e., $(\hat{p}'_1, \hat{u}'_1) = (\hat{p}'(x_f - \epsilon, t), \hat{u}'(x_f - \epsilon, t))$ and $(\hat{p}'_2, \hat{u}'_2) = (\hat{p}'(x_f + \epsilon, t), \hat{u}'(x_f + \epsilon, t))$ for small ϵ. (The up-stream density fluctuation is determined from the pressure as $\hat{\rho}'_1 = \hat{p}'_1/\bar{c}_1^2$.) The inferred heat release using the P-GalNNs from the previous sections is shown in Fig. 4.18. In the noise-free case, in which the data is given on a fine grid of sensor measurements, there is a good match between the predicted time series and the time series from the simulation data. In the noisy case, in which the data is contaminated with 10 % noise and given on a coarse grid of pressure and velocity

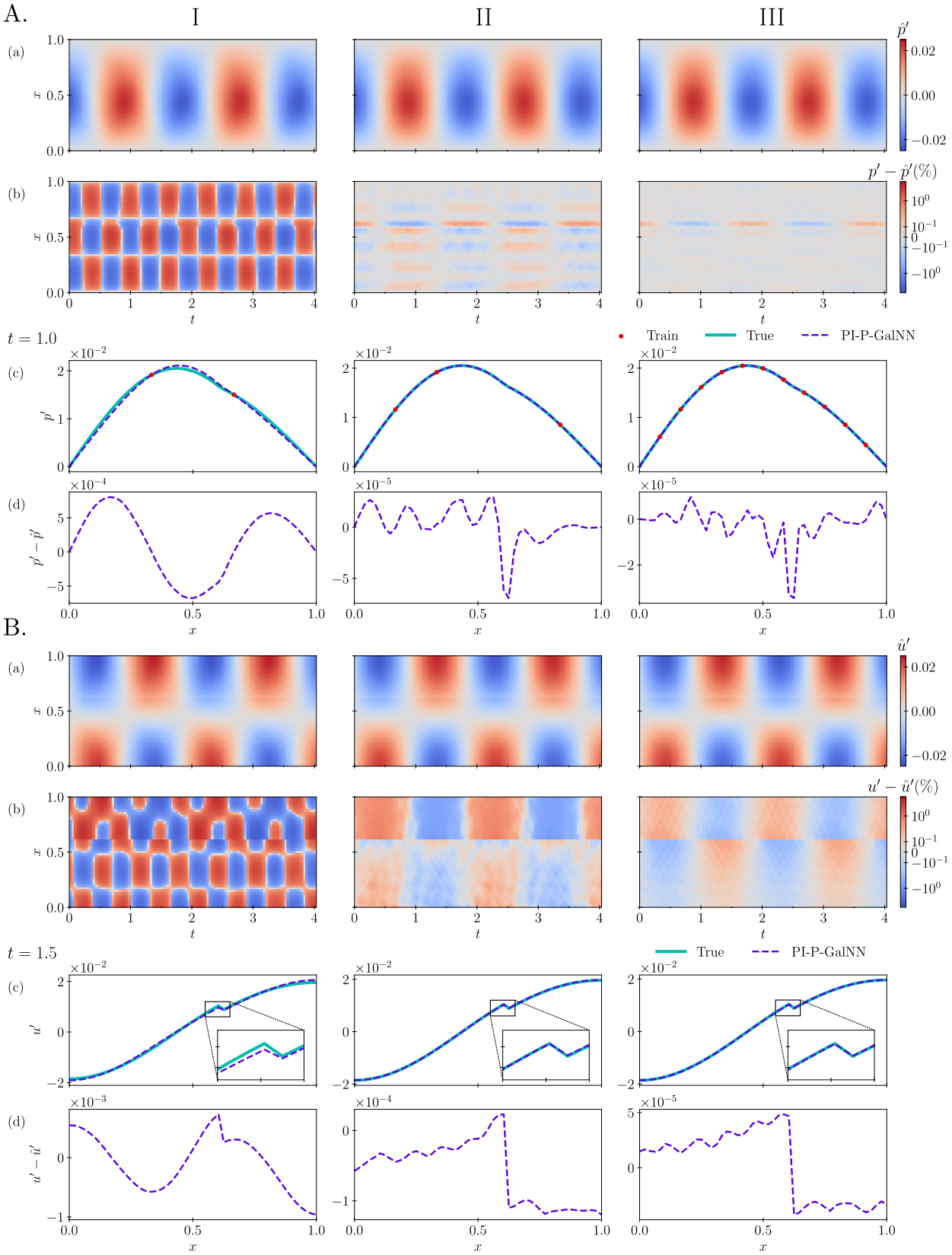


Figure 4.17: Reconstruction of acoustic pressure and velocity from only pressure measurements using PI-P-GalNN with different number of sensors and Galerkin modes. From left to right, (I) 2 sensors, 5 Galerkin modes, (II) 5 sensors, 10 Galerkin modes, (III) 11 sensors, 20 Galerkin modes. From top to bottom: (a) acoustic A.) pressure and B.) velocity, (b) error between ground truth and prediction, (c) prediction of A.) pressure and B.) velocity at a fixed time instance, and (d) error at a fixed time instance, A.) $t = 1.0$ and B.) $t = 1.5$.

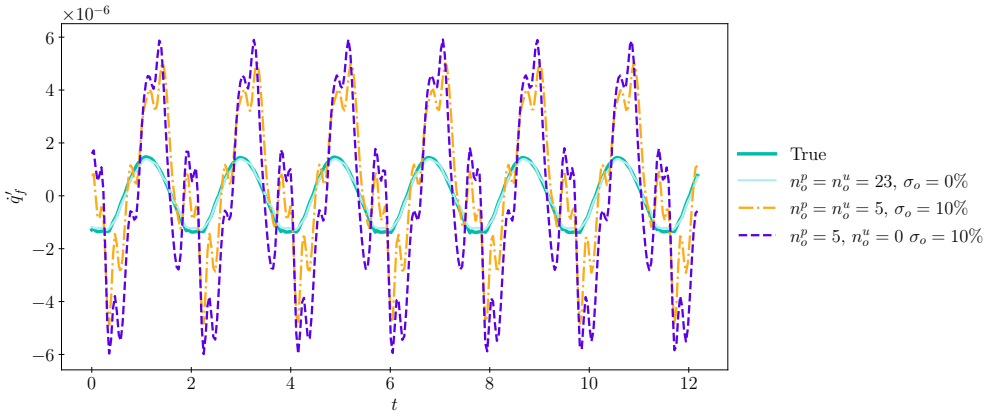


Figure 4.18: Inference of heat release rate using P-GalNNs trained on pressure and velocity measurements. Datasets are (i) fine grid of noise free measurements $n_o^p = n_o^u = 23$, (ii) coarse grid of noisy measurements $n_o^p = n_o^u = 5, \sigma_o = 10\%$, and (iii) coarse grid of only noisy pressure measurements $n_o^p = 5, n_o^u = 0, \sigma_o = 10\%$.

measurements, the prediction accuracy of the network drops. Note that the order of magnitude of the nondimensionalised heat release is much smaller than that of velocity or pressure, hence any errors in pressure or velocity prediction will have a larger impact on the prediction of the heat release.

4.6.4 Nonideal boundary conditions

So far, we have restricted ourselves to ideal boundary conditions. However, in real experiments, we deal with nonideal conditions, in which the acoustic waves are not fully reflected at the tube ends (e.g., Levine and Schwinger, 1948). As a result, in contrast to the ideal case and the acoustic modes we have obtained under that assumption, the pressure fluctuations at the tube ends are not zero, i.e., $p'(x=0,t) \neq 0,\ p'(x=1,t) \neq 0$. In order to tackle nonideal boundary conditions, we add a linear term in the pressure modes (3.11a),

$$\Pi^{(1)}_{N_g+1}(x) = \Pi^{(2)}_{n_g+1}(x) = x, \tag{4.17a}$$

$$\Pi^{(1)}_{N_g+2}(x) = \Pi^{(2)}_{n_g+2}(x) = 1, \tag{4.17b}$$

such that the summation (3.10a) runs from 1 to $n_g + 2$. The modes Π_{n_g+1} and Π_{n_g+2} are weighted by independent coefficients, μ_{n_g+1} and μ_{n_g+2}. This way, the linear term provides the offset to the predicted pressure fluctuations at the tube ends, i.e., $\mu_{n_g+1}(t)x + \mu_{n_g+2}(t)$. Once trained, we expect to estimate $\mu_{n_g+2}(t) = p'(x=0,t)$ and $\mu_{n_g+1}(t) = p'(x=1,t) - p'(x=0,t)$.

We consider three cases and generate data from the higher-fidelity model using the following parameters: (I) Close to ideal: $\tilde{\bar{u}}_1 = 8 m/s$, $\tilde{\bar{Q}} = 10000 W$, $\Gamma_{\text{in}} = \Gamma_{\text{out}} = -0.985$; (II) less close to ideal: $\tilde{\bar{u}}_1 = 16\, m/s$, $\tilde{\bar{Q}} = 60000\, W$, $\Gamma_{\text{in}} = \Gamma_{\text{out}} = -0.85$; and (III) different at both ends: $\tilde{\bar{u}}_1 = 16\, m/s$, $\tilde{\bar{Q}} = 40000\, W$, $\Gamma_{\text{in}} = -0.985, \Gamma_{\text{out}} = -0.85$. The nondimensional densities in the duct segments before and after the flame are; for case (I) $\bar{\rho}_1 = 1.269, \bar{\rho}_2 = 0.571$, for case (II) $\bar{\rho}_1 = 1.434, \bar{\rho}_2 = 0.307$, and for case (III) $\bar{\rho}_1 = 1.376, \bar{\rho}_2 = 0.400$.

Fig. 4.19 compares the predictions of a PI-P-GalNN (15 Galerkin modes) with a linear term against

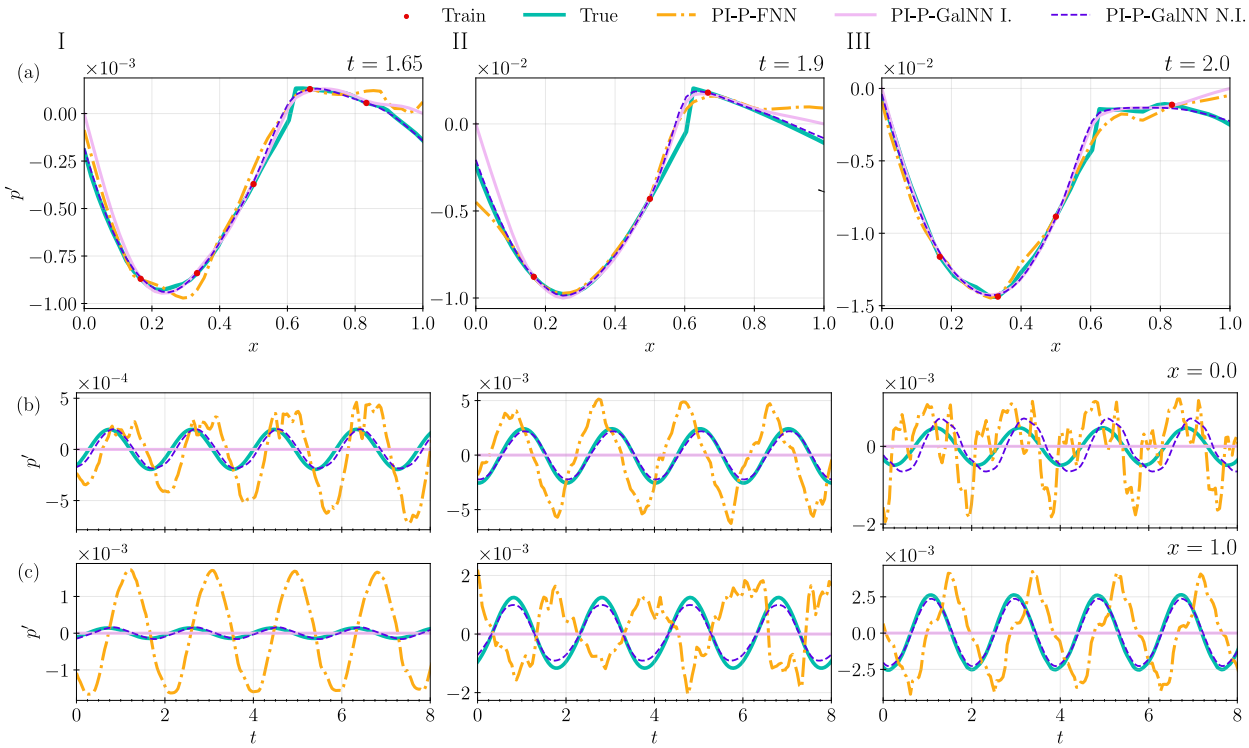


Figure 4.19: Prediction in the case of nonideal boundary conditions for a training case of only 5 noise-free pressure sensors. Comparison of a standard physics-informed periodic Galerkin neural (PI-P-GalNN I.), physics-informed periodic feedforward network (PI-P-FNN) with a PI-P-GalNN for nonideal boundary conditions (PI-P-GalNN N.I.). Training performed on three test cases; (I) $\Gamma_{\text{in}} = \Gamma_{\text{out}} = -0.985$, (II) $\Gamma_{\text{in}} = \Gamma_{\text{out}} = -0.85$ and (III) $\Gamma_{\text{in}} = -0.985, \Gamma_{\text{out}} = -0.85$. Shown in (a) pressure shape at a fixed time instance, (I) $t = 1.65$, (II) $t = 1.90$, and (III) $t = 2.00$; (b,c) pressure fluctuations at $x = 0$ and $x = 1$ respectively. The linear term can capture the non-zero pressure fluctuations at the boundaries, when they are close to ideal.

the standard PI-P-GalNN and the PI-P-FNN for the above cases, where the training data consists of only five pressure sensors, $n_o^p = 5,\ n_o^u = 0$. With the addition of the linear term, the PI-P-GalNN is capable of capturing the non-zero pressure fluctuations at the boundaries for near ideal conditions even from sparse measurements given at other locations. While PI-P-FNN is a flexible tool that can handle various boundary conditions, it requires the boundary data (Silva Garzon et al., 2023), whereas the PI-P-GalNN does not.

4.7 Conclusion

In this work, we model acoustic and thermoacoustic pressure and velocity oscillations from synthetic data. The synthetic data captures the rich nonlinear behaviour of thermoacoustic oscillations observed in propulsion and power generation. We develop acoustic neural networks to tackle the tasks of (i) extrapolation in space and time, and (ii) reconstruction of full acoustic state from partial observations by exploiting prior knowledge on the physics of acoustics. The prior knowledge is embedded in the network as both soft and hard constraints.

First, as acoustic and thermoacoustic systems are dominated by Fourier eigenfunctions, we promote a physically-motivated choice of sinusoidal activation functions. Unlike standard feedforward neural network architectures that employ ReLU or tanh activations, periodically activated networks

can extrapolate in time, and the trained weights hold frequency information. This means that spatiotemporal patterns can be learned more efficiently from less data because the network is more expressive as well as robust.

Second, we inform the training with the acoustic conservation laws with penalty terms in the loss function. This term regularises the predictions in the case of noisy and scarce data, and enables reconstruction of unobserved states. Typically, in thermoacoustics experiments, only pressure measurements are available.

Third, inspired by Galerkin decomposition, we design a neural network with temporal and spatial branches (Galerkin neural network), which spans a physical function space via the choice of acoustic eigenmodes as the spatial basis. In order to account for nonideal boundary conditions that occur when the acoustic waves are not fully reflected at the boundaries, we add a linear mode in the pressure modes, which captures the non-zero pressure fluctuations at the inlet and outlet for near ideal conditions without requiring data from the boundaries.

We consider two test cases: (i) twin experiments on synthetic data from a Rijke tube with a nonlinear, time-delayed heat release model with Galerkin discretisation of the PDEs, and (ii) higher-fidelity data generated by a thermoacoustic network model with a kinematic flame model that takes into account mean-flow effects. The first case shows that the standard feedforward neural networks fail at extrapolation, while periodically activated networks can extrapolate and reconstruct accurately from partial measurements. On this test case, we also demonstrate the long term extrapolation capability of Galerkin neural networks on periodic and quasiperiodic solutions, while for chaotic solutions, we recover the dominant components in the frequency spectrum. In the second test case, we show the generalisability and robustness of the Galerkin neural networks to higher-fidelity data, which can be noisy and contain only pressure measurements. Hard-constraining neural networks with prior knowledge reduces the required network size, search space of hyperparameters as well as the amount of data required for training. The computation times for some characteristic cases are provided in the Appendix § E.1.

This work opens up possibilities to learn the nonlinear dynamics of thermoacoustics using physics-aware data-driven methods. The developed GalNNs can be transferred to other problems that are solved by Galerkin methods to enhance robustness and expressivity.

Chapter 5

Data-driven computation of adjoint sensitivities without adjoint solvers

The work in this chapter can be found in Ozan and Magri (2024a) and Ozan and Magri (2024b). The code is available as a GitHub repository at MagriLab/Adjoint-ESN.

5.1 Introduction

Many computational applications including optimisation, data assimilation, and uncertainty quantification require the information about how a quantity of interest is influenced by the system's parameters and initial conditions. Sensitivity analysis provides the gradient of an objective functional with respect to all system's parameters. A direct method for estimating the sensitivity is to perturb the parameter and approximate the gradient with a finite difference between the base and the perturbed solutions. This procedure suffers from numerical errors and needs to be repeated for each parameter. Therefore, for systems with many parameters, the computational cost increases linearly with the number of parameters. Adjoint methods turn this around. With an adjoint formulation, the sensitivity to all parameters is computed with a single operation, which requires the adjoint system. For this reason, adjoint methods have become central to optimisation in flow and acoustics applications (e.g., Peter and Dwight, 2010; Sipp et al., 2010; Luchini and Bottaro, 2014; Camarri, 2015; Magri, 2019) and uncertainty quantification (e.g., Magri et al., 2016b; Mensah et al., 2018; Magri, 2019).

In this chapter, we propose a SciML framework for the computation of adjoint sensitivities from data. Recapping from § 1.2, the optimisation of a quantity of interest via direct-adjoint looping follows these steps:

(i) the system's nonlinear governing equations are integrated forward in time to obtain the baseline (direct) solution;

(ii) the nonlinear equations are linearised around the direct solution to obtain the Jacobian of the system;

(iii) the adjoint equations are derived from the Jacobian;

(iv) the adjoint equations are initialised at the end of the direct solution, and are integrated backwards in time;

(v) at the end of the adjoint integration, the gradient of the objective function to all the design parameters of interest can be evaluated by an inner product;

(vi) the gradient is embedded in a gradient-based optimisation scheme to find a local optimum of the objective function.

To bypass the need of the system's equations, a data-driven model can be employed. The key idea of this chapter is to learn an approximation of the dynamical systems from observables, and then compute the adjoint of it. This approach is useful when working with experimental data, or data whose equations are not fully known. To infer the governing dynamics from observables (data), we employ a type of reservoir computer, the echo state network (ESN) (Jaeger and Haas, 2004), which is a universal approximator (Grigoryeva and Ortega, 2018) and a type of recurrent neural network (RNN). The ESN's training is straightforward because it does not require backpropagation. Its Jacobian can also be computed in a straightforward way due to the shallow architecture (Margazoglou and Magri, 2023b). In this chapter, we take an additional step to the classical ESN. Because in order to perform design optimisation of system's parameters, we require a parameter-aware extension of the echo state network (Xiao et al., 2021b), which learns the parameterised dynamics and can operate across parameter regimes.

In chaotic systems, in which nearby trajectories diverge at an exponential rate, the adjoint system becomes unstable and the sensitivities diverge after some integration time. This poses a challenge as the quantities of interest are usually long-term time averages. A possible approach that we also exploit in this chapter is to take an ensemble of short-term trajectories (3.4) (Lea et al., 2000). Other approaches include probability-density-based methods (Thuburn, 2005), and shadowing methods (Wang, 2013), which can provide more accurate estimates however can be computationally expensive and algorithmically challenging. Therefore, we opt for the ensemble method to keep the focus on the SciML methodology.

In this chapter, we focus on the suppression of thermoacoustic instabilities. Both passive (e.g., Noiray and Schuermans, 2012; Magri and Juniper, 2013) and active feedback (e.g., Dowling and Morgans, 2005; Gelbert et al., 2012) control approaches have been developed for this purpose. In thermoacoustic systems, optimisation is especially challenging because as shown in § 3.4.3 thermoacoustics exhibit nonlinear behaviour and bifurcations to periodic, quasiperiodic, and chaotic regimes (Kabiraj et al., 2012), which motivates the development of methods for nonlinear analysis. Preliminary design of thermoacoustic systems is performed by linear stability (Lieuwen and Yang, 2006; Juniper and Sujith, 2018). Linear stability analysis determines the growth rates of infinitesimal perturbations to the mean flow, i.e., the eigenvalues of the linearised system of equations around the mean flow (Magri et al., 2023). To name a few examples, adjoint-based linear stability analysis enabled calculation of sensitivity to parameters and passive control mechanisms (Magri and Juniper, 2013), optimal placement and tuning of acoustic dampers (Mensah and Moeck, 2017) and stabilisation of thermoacoustic oscillations by geometry changes (Aguilar and Juniper, 2018). An in-depth review of development of adjoint methods for thermoacoustics can be found in (Magri, 2019; Magri et al., 2023). Beyond linear stability analysis, adjoint methods were used for finding

the lowest energy initial state that leads to triggering (Juniper, 2011), weakly nonlinear analysis to predict subcritical Hopf bifurcations (Orchini et al., 2016), and Floquet analysis to calculate the stability and the period of limit-cycle oscillations (Magri, 2019). The stability of chaotic thermoacoustics was studied by Huhn and Magri (2020) using covariant Lyapunov analysis, who analytically showed that this is equivalent to eigenvalue and Floquet analysis for fixed point and limit-cycle solutions, respectively. As a result, Huhn and Magri (2020) achieved gradient-based minimisation of acoustic energy in chaotic thermoacoustics using shadowing methods. The application of this method was challenged by the discontinuities observed in the acoustic energy when bifurcations occur, computational cost of shadowing methods, and non-hyperbolicity of thermoacoustics for some regimes. To circumvent these issues, a gradient-free optimisation method based on Bayesian optimisation was proposed (Huhn and Magri, 2022).

The overarching objective of this chapter is to address the challenge of developing code-specific adjoint solvers, which requires the system's equations and the Jacobian to be known. We propose data-driven nonlinear direct-adjoint looping to compute sensitivities to the system's parameters and initial conditions, and apply this to (i) the Lorenz system (§ 3.3), and (ii) the Rijke tube (§ 3.4). The specific objectives are to introduce a robust parameter-aware ESN architecture for this purpose, derive its adjoint, and demonstrate the data-driven inference of parameterised dynamics and associated adjoint sensitivities. For the Lorenz system, we show that the standard parameter-aware architecture (i) captures the parameterised dynamics and (ii) infers adjoint sensitivities to the parameter changes. For the thermoacoustic problem, we develop a physics-aware architecture and refer to it as the thermoacoustic ESN (T-ESN) and show that (i) it learns the parameterised dynamics of nonlinear, time-delayed thermoacoustics, (ii) predicts bifurcations, (iii) accurately obtains the adjoint sensitivities, (iv) the acoustic energy is minimised in a gradient-based optimisation framework, and (v) demanding scenarios such as training with noisy or chaotic data can be handled.

This chapter is organised as follows. In §. 5.2, we introduce the parameter-aware ESN. In §. 5.3, we derive the adjoint of the ESN. In § 5.4, we apply the methodology to the Lorenz system. In § 5.5, we introduce the T-ESN designed for nonlinear, time-delayed thermoacoustics and derive its adjoint. In § 5.6, we show the performance of the T-ESN in terms of prediction and accuracy of the learned adjoint sensitivities, and employ it for the gradient-based design optimisation of thermoacoustics in § 5.7. We address training with noisy data in § 5.8, and learning chaotic regimes in § 5.9. We show computation of sensitivity to initial conditions in § 5.10. We discuss advantages and limitations of the proposed approach in § 5.11. § 5.12 concludes the chapter.

5.2 Parameter-aware echo state networks *(PA-ESNs)*

Given the evolution equations of a dynamical system (2.15), we can compute its adjoint (2.19). Developing the adjoint solver however requires the system Jacobian, $\partial \boldsymbol{F}/\partial \boldsymbol{x}$. This Jacobian is code specific. The key idea of this chapter is to eliminate code-specific adjoint solvers by having a machine that learns the dynamics from data, and is itself a dynamical system, the Jacobian of which is readily available independent of the system in question. Recurrent neural networks (RNNs) are well-suited for this task, as they are neural network architectures that can process sequential data

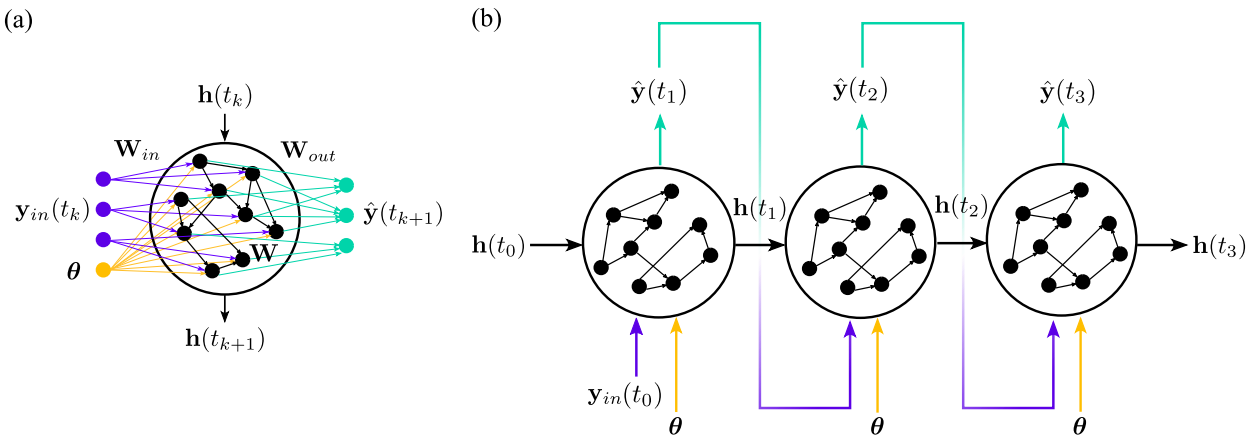


Figure 5.1: Schematic of the parameter-aware echo state network (PA-ESN); (a) the architecture, and (b) closed-loop configuration. The ESN learns a mapping from the inputs $\boldsymbol{y}_{\text{in}}$ at time step k to the outputs $\boldsymbol{y}$ at time step $k+1$. (The network prediction is denoted by $\hat{\boldsymbol{y}}$.) This achieved via the hidden state of the ESN $\boldsymbol{h}(t_k)$, which models a reservoir that contains a time-delayed embedding of the inputs, i.e., the reservoir state. In the PA-ESN, the input channel $\boldsymbol{y}_{\text{in}}$ is augmented with the parameter vector $\boldsymbol{\theta}$, which allows the network to capture variations of the dynamics to the system's parameters. In closed-loop configuration, the network's prediction $\hat{\boldsymbol{y}}(t_k)$ is fed back as the input for the next time step $\boldsymbol{y}_{\text{in}}(t_{k+1})$ so that the network can forecast the dynamics autonomously.

and make forecasts of time series from dynamical systems. Although the dynamical systems we consider in Eq. (2.15) are continuous in time, RNNs accept data as a sequence in discrete time. RNNs possess memory through a hidden state and thus they can learn temporal relationships. The hidden state enables the RNNs to be framed as discrete-time dynamical systems (2.4) (Margazoglou and Magri, 2023b). In this work, we utilise the echo state network (ESN) (Jaeger and Haas, 2004), which is a reservoir computer, and a type of RNN that is straightforward to train because it does not require backpropagation during training.

5.2.1 Architecture and training of the PA-ESN

The hidden state of the ESN is called the reservoir state, where the reservoir is a high-dimensional embedding of the input (Hart et al., 2020; Duan et al., 2023). The evolution of the reservoir state is a nonlinear discrete-time map from reservoir state at time step k to reservoir state at time step $k+1$,

$$\boldsymbol{h}(k+1) = (1-\alpha_h)\boldsymbol{h}(k) + \alpha_h \tanh(\boldsymbol{W}_{\text{in}}\boldsymbol{y}_{\text{in}}(k) + \boldsymbol{W}_h\boldsymbol{h}(k)), \tag{5.1}$$

where α_h is the leak rate which acts as an exponential moving average filter, $\boldsymbol{y}_{\text{in}} \in \mathbb{R}^{n_y}$ is the input vector, $\boldsymbol{h}(k) \in \mathbb{R}^{n_h}$ is the reservoir state, $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{n_h \times n_y}$ is the input matrix, and $\boldsymbol{W}_h \in \mathbb{R}^{n_h \times n_h}$ is the state matrix. The prediction of the output is a linear combination of the reservoir state

$$\hat{\boldsymbol{y}}(k+1) = \boldsymbol{W}_{\text{out}}\boldsymbol{h}(k+1), \tag{5.2}$$

where $\boldsymbol{y} \in \mathbb{R}^{n_y}$ is the target output vector, the dynamics of which we aim to model, $\hat{\boldsymbol{y}} \in \mathbb{R}^{n_y}$ is the predicted output vector, and $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{n_y \times n_h}$ is the output matrix that maps the reservoir state to the output vector. The matrices $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}_h$ are sparse, randomly generated, and not trained, whilst $\boldsymbol{W}_{\text{out}}$ is trained via Ridge regression (Lukoševičius, 2012). The input matrix is scaled as $\boldsymbol{W}_{\text{in}} = \xi_{\text{in}}\widetilde{\boldsymbol{W}}_{\text{in}}$, where $\widetilde{\boldsymbol{W}}_{\text{in}}$ is constructed such that each row contains only one nonzero element

drawn from a uniform distribution $\sim \mathcal{U}(-1,1)$, i.e., the inputs are activated individually. The state matrix $\boldsymbol{W}_h$ describes the reservoir connections, and is scaled as $\boldsymbol{W}_h = \rho_h \widetilde{\boldsymbol{W}}_h$, where $\widetilde{\boldsymbol{W}}_h$ is an Erdős-Renyi matrix with a given average number of nonzero elements per row (connectivity, n_{conn}) drawn from a uniform distribution $\sim \mathcal{U}(-1,1)$ that is scaled to have a unity spectral radius. So, the scalar ρ_h is set as the spectral radius of $\boldsymbol{W}_h$. In this architecture, the reservoir state variables "echo" the input, and so the reservoir holds memory.

During training $\boldsymbol{y}_{\text{in}}$ is fed into the network in open loop and the evolution of the reservoir state is saved. The reservoir state is initialised to a zero vector. An initial time window of N_{wash} time steps is used for the washout stage, which eliminates the effects of the initial condition and is discarded in training. The training is performed by solving for $\boldsymbol{W}_{\text{out}}$ (Lukoševičius, 2012)

$$\boldsymbol{W}^*_{\text{out}} = \underset{\boldsymbol{W}_{\text{out}}}{\arg\min} \frac{1}{n_y} \sum_{i=1}^{n_y} \left(\sum_{k=1}^{N_{\text{train}}} \left(y_i(k) - \hat{y}_i(k)\right)^2 + \lambda ||\boldsymbol{w}_{out,i}||_2^2 \right), \tag{5.3}$$

where N_{train} is the number of training steps, $\boldsymbol{w}_{out,i}$ is the i^{th} row of $\boldsymbol{W}_{\text{out}}$, and λ is the Tikhonov coefficient. This is a Ridge regression problem, which minimises both the error between network predictions for the next step and the norm of the output weights for regularisation. The solution is (Lukoševičius, 2012)

$$\boldsymbol{W}^*_{\text{out}} = \boldsymbol{Y}\boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{H}^\top + \lambda \boldsymbol{I}_{n_h \times n_h})^{-1}, \tag{5.4}$$

where $\boldsymbol{H} = [\boldsymbol{h}(1)\ \boldsymbol{h}(2)\ \dots\ \boldsymbol{h}(N_{\text{train}})]$, $\boldsymbol{H} \in \mathbb{R}^{n_h \times N_{\text{train}}}$, $\boldsymbol{Y} = [\boldsymbol{y}(1)\ \boldsymbol{y}(2)\ \dots\ \boldsymbol{y}(N_{\text{train}})]$, $\boldsymbol{Y} \in \mathbb{R}^{n_y \times N_{\text{train}}}$, and $\boldsymbol{I}$ is the identity matrix. Tuning an ESN involves searching for optimal hyperparameters. Whilst the input and state matrices are randomly generated, the hyperparameters determine the proper timescales and nonlinearity (Manneschi et al., 2021; Storm et al., 2022). These hyperparameters are input matrix scaling ξ_{in}, the spectral radius of the state matrix ρ_h (typically ≤ 1 to satisfy the echo state property), the leak rate α_h, and the Tikhonov coefficient λ. These hyperparameters are determined by validation. A detailed explanation of training and robust validation of ESNs can be found in Racca and Magri (2021). After training, the ESN can run autonomously in closed-loop by setting $\boldsymbol{y}_{\text{in}}(k) = \hat{\boldsymbol{y}}(k)$.

In the formulation (5.1), the ESN is unaware of the system's parameters, i.e., it can only operate in one regime. For our purposes, we require an ESN to learn the parameter-dependent dynamics of a system such that we can capture the variations in different regimes. A parameter-aware extension of the ESN has been employed to predict when the system bifurcates to fixed point solutions, a phenomenon known as amplitude death (Xiao et al., 2021b), and bifurcations in multi-stable regimes including chaos (Roy et al., 2022). It can be formulated as (Xiao et al., 2021b)

$$\boldsymbol{h}(k+1) = (1-\alpha_h)\boldsymbol{h}(k) + \alpha_h \tanh(\boldsymbol{W}_{\text{in}}\tilde{\boldsymbol{y}}_{\text{in}}(k) + \boldsymbol{W}_h \boldsymbol{h}(k)), \tag{5.5}$$

where $\tilde{\boldsymbol{y}}_{\text{in}}(k)$ is the augmented input, defined as

$$\tilde{\boldsymbol{y}}_{\text{in}}(k) = [\boldsymbol{y}_{\text{in}}(k);\ \boldsymbol{\xi}_{\boldsymbol{\theta}} \odot (\boldsymbol{\theta} - \boldsymbol{g}_\theta)] \tag{5.6}$$

where [;] denotes vertical concatenation; $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ denotes the system's parameters and are fed to the network through the input channel, which we assume to be constant in time for each

system; and $\odot$ denotes the Hadamard product. The input matrix is composed of two blocks $\boldsymbol{W}_{\text{in}} = \left[\boldsymbol{W}_{\text{in}}^{y}\ \boldsymbol{W}_{\text{in}}^{\theta}\right] = [\xi_{\text{in}}\widetilde{\boldsymbol{W}}_{\text{in}}^{y}\ \widetilde{\boldsymbol{W}}_{\text{in}}^{\theta}]$, $\boldsymbol{W}_{\text{in}}^{y} \in \mathbb{R}^{n_h \times n_y}$, $\boldsymbol{W}_{\text{in}}^{\theta} \in \mathbb{R}^{n_h \times n_\theta}$. It is constructed such that the parameters are fully connected with all the reservoir nodes, i.e., the weights associated with the parameters $\boldsymbol{W}_{\text{in}}^{\theta}$ are nonzero for all rows and drawn from the uniform distribution $\sim \mathcal{U}(-1, 1)$. The parameter-aware ESN adds two additional hyperparameters per physical parameter θ_i (Xiao et al., 2021b): A scalar g_{θ_i} shifts (offsets) the parameter θ_i as $\theta_i - g_{\theta_i}$ and a scalar ξ_{θ_i} scales it as $\xi_{\theta_i}(\theta_i - g_{\theta_i})$. We separately apply the input scaling ξ_{in} on $\widetilde{\boldsymbol{W}}_{\text{in}}^{y}$ only. Similarly to the scaling of the input state, the scaling and shifting of the system's parameters are necessary to capture the correct nonlinear behaviour with the tanh function. The training of the parameter-aware ESN is performed as the standard ESN (5.4). The dataset contains data from different parameter regimes, which means that we need to save the evolution of the reservoir state for each training regime, and concatenate as $\boldsymbol{H} = \left[\boldsymbol{H}^1\ \boldsymbol{H}^2\ \ldots \boldsymbol{H}^{N_{\text{reg}}}\right]$ (likewise, the outputs are horizontally concatenated as $\boldsymbol{Y} = \left[\boldsymbol{Y}^1\ \boldsymbol{Y}^2\ \ldots \boldsymbol{Y}^{N_{\text{reg}}}\right]$), where N_{reg} is the number of training regimes. The algorithm for training a PA-ESN is given in Alg. 1. The implementation of the PA-ESN and its validation based on Bayesian optimisation has been adapted from the standard ESN implementation of Racca (2022).

In § 5.4.1 and § 5.8, we consider noisy input and output data

$$\boldsymbol{y}_{\text{in}}(k) = \boldsymbol{y}(k) = \boldsymbol{y}^{\text{true}}(k) + \boldsymbol{\epsilon}(k), \quad \boldsymbol{\epsilon}_i(k)\ \sim \mathcal{N}(0, \sigma_o^2 \sigma_i^2), \tag{5.7}$$

where $\boldsymbol{y}^{\text{true}}$ denotes the true values before the addition of noise, σ_i is the standard deviation of y_i^{true} calculated individually in each considered parameter regime, and σ_o is the scaling that determines the noise level.

Algorithm 1 Training of parameter-aware echo state network (PA-ESN)

for each parameter regime $\boldsymbol{\theta}^i$, $i = 1$ to N_{reg} **do**
  Initialise $\boldsymbol{h}(0) \leftarrow \boldsymbol{0}_{n_h}$
  for $k = 1$ to $N_{\text{wash}} + N_{\text{train}}$ **do** ▷ Run in open loop
    $$\boldsymbol{h}(k) \leftarrow (1 - \alpha_h)\boldsymbol{h}(k-1) + \alpha_h \tanh\left(\boldsymbol{W}_{\text{in}}\begin{bmatrix}\boldsymbol{y}(k-1)\\ \boldsymbol{\xi_p} \odot (\boldsymbol{\theta}^i - \boldsymbol{g}_\theta)\end{bmatrix} + \boldsymbol{W}_h \boldsymbol{h}(k-1)\right)$$
    if $k > N_{\text{wash}}$ **then**
      $\boldsymbol{H}^i \leftarrow [\boldsymbol{H}^i\ \boldsymbol{h}(k)]$ ▷ Append reservoir state vector
      $\boldsymbol{Y}^i \leftarrow [\boldsymbol{Y}^i\ \boldsymbol{y}(k)]$ ▷ Append output vector
  $\boldsymbol{H} \leftarrow [\boldsymbol{H}\ \boldsymbol{H}^i]$
  $\boldsymbol{Y} \leftarrow [\boldsymbol{Y}\ \boldsymbol{Y}^i]$
Solve Ridge regression:
$\boldsymbol{W}_{\text{out}} \leftarrow \boldsymbol{Y}\boldsymbol{H}^\top \left(\boldsymbol{H}\boldsymbol{H}^\top + \lambda \boldsymbol{I}_{n_h \times n_h}\right)^{-1}$

5.2.2 Performance metrics

Validation error As error metric for the validation, we compute relative ℓ_2-error between the prediction, $\hat{\boldsymbol{y}}$ and the ground truth $\boldsymbol{y}$

$$e_{\text{val}} = \frac{\sqrt{\sum_{k=1}^{N_{\text{val}}} \sum_{i=1}^{n_y} \left(y_i(k) - \hat{y}_i(k)\right)^2}}{\sqrt{\sum_{k=1}^{N_{\text{val}}} \sum_{i=1}^{n_y} y_i^2(k)}}, \tag{5.8}$$

where N_{val} is the number of validation steps. For selection of the hyperparameters, we determine the average validation error over a number of short trajectories or folds, validation regimes, and realisations of the ESN. For the parameter-aware ESN, we emphasise the importance of validating on unseen parameter regimes for generalisability.

Predictability horizon We compute a predictability horizon (PH) for chaotic dynamics (e.g., Doan et al., 2020a; Pathak et al., 2018; Racca and Magri, 2021). Given a time series of the ground truth $\{\boldsymbol{y}(k)\}_{k=1}^N$ and the prediction $\{\hat{\boldsymbol{y}}(k)\}_{k=1}^N$, we define the PH as the first time the normalised error between the truth and threshold exceeds a certain threshold,

$$PH_i = \min \left\{ \frac{t_k}{LT} \middle| \; e_i(k) > 0.5 \right\}, \quad PH = \min_i PH_i \tag{5.9}$$

where the time is scaled by the Lyapunov time of the system and

$$e_i(k) = \sqrt{\frac{\left(y_i(k) - \hat{y}_i(k)\right)^2}{\sigma_i^2}}, \tag{5.10}$$

is the prediction error normalised by the variance of the ground truth $\sigma_i^2 = 1/N \sum_{k=1}^N \left(y_i(k) - \bar{y}_i\right)^2$ with $\bar{y}_i = 1/N \sum_{k=1}^N y_i(k)$ denoting the mean.

5.3 Adjoint of the PA-ESN

In this section, we derive the adjoint of the parameter-aware ESN. In closed-loop, the ESN is governed by,

$$\boldsymbol{h}(k+1) = (1-\alpha_h)\boldsymbol{h}(k) + \alpha_h \tanh(\boldsymbol{W}_{\text{in}}[\boldsymbol{W}_{\text{out}}\boldsymbol{h}(k); \boldsymbol{\xi}_\theta \odot (\boldsymbol{\theta} - \boldsymbol{g}_\theta)] + \boldsymbol{W}_h \boldsymbol{h}(k)), \tag{5.11}$$

which is an autonomous dynamical system. This means that the evolution of the system solely depends on the reservoir state $\boldsymbol{h}$. This is the configuration that we use for the computation of the adjoint. We first tackle sensitivity to the system's parameters, and then to the system's initial conditions.

5.3.1 Sensitivity to the system's parameters

We consider the time-averaged objective functional (2.14), which we wish to optimise. In discrete time, we reformulate it as a sum over a discrete number of time steps, denoted by N

$$\mathcal{J} = \frac{1}{N}\sum_{k=1}^{N} \tilde{\mathcal{J}}(\boldsymbol{h}(k)). \tag{5.12}$$

Without loss of generality, we assume that the objective functional does not explicitly depend on the parameters, i.e., $\partial\tilde{\mathcal{J}}/\partial\boldsymbol{\theta} = 0$. For the parameter-aware ESN, the tangent linear problem is

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{1}{N}\sum_{k=1}^{N} \frac{d\tilde{\mathcal{J}}(\boldsymbol{h}(k))}{d\boldsymbol{h}(k)}\boldsymbol{Q}(k), \tag{5.13a}$$

$$\boldsymbol{Q}(k+1) = \frac{\partial \boldsymbol{h}(k+1)}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{h}(k+1)}{\partial \boldsymbol{h}(k)}\boldsymbol{Q}(k), \tag{5.13b}$$

$$\boldsymbol{Q}(0) = \boldsymbol{0}, \tag{5.13c}$$

where $\boldsymbol{Q}(k) \equiv d\boldsymbol{h}(k)/d\boldsymbol{\theta}$, $\boldsymbol{Q}(k) \in \mathbb{R}^{n_h \times n_\theta}$. Physically, Eq. (5.13c) means that the initial condition is independent of the parameter. Because the dimension of the tangent linear system, $\boldsymbol{Q}(k)$, grows with the number of parameters, adjoint methods enable a computationally cheap and accurate way to obtain sensitivities (Magri, 2019). Using the Lagrangian formulation of the objective and constraints given by the evolution equations (Appendix § B.1), the adjoint problem is formulated as[1]

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \sum_{k=1}^{N} \boldsymbol{q}^{+\top}(k)\frac{\partial \boldsymbol{h}(k)}{\partial \boldsymbol{\theta}}, \tag{5.14a}$$

$$\boldsymbol{q}^{+}(k) = \frac{1}{N}\frac{d\tilde{\mathcal{J}}(\boldsymbol{h}(k))}{d\boldsymbol{h}(k)}^{\top} + \frac{\partial \boldsymbol{h}(k+1)}{\partial \boldsymbol{h}(k)}^{\top}\boldsymbol{q}^{+}(k+1), \tag{5.14b}$$

$$\boldsymbol{q}^{+}(N) = \frac{1}{N}\frac{d\tilde{\mathcal{J}}(\boldsymbol{h}(N))}{d\boldsymbol{h}(N)}^{\top}. \tag{5.14c}$$

Eq. (5.14) is in a form similar to the continuous time equations (2.19), however, we have replaced the system's Jacobian with the ESN's Jacobian, thus, the adjoint variables $\boldsymbol{q}^{+}$ are the adjoint of the ESN's reservoir state. The full derivation of the adjoint equations can be found in the Appendix § B.1

Algorithmically (shown in Alg. 2), we first obtain the direct solution, $\boldsymbol{h}(k)$, as a baseline trajectory by running the ESN in closed-loop for a time window. Then, the adjoint equations (5.14b) are solved backwards in time starting from the terminal condition $\boldsymbol{q}^{+}(N)$ (5.14c). This procedure requires the computation of the Jacobian with respect to the reservoir state $\partial\boldsymbol{h}(k+1)/\partial\boldsymbol{h}(k)$ and with respect to the parameters $\partial\boldsymbol{h}(k)/\partial\boldsymbol{\theta}$, associated with the ESN, which are provided in § 5.3.3.

[1]With a slight abuse of notation, we use $\boldsymbol{q}^{+}$ to denote the adjoint of the reservoir state $\boldsymbol{h}$, in addition to the adjoint of the generic state vector $\boldsymbol{s}$ in Eq. (2.19).

The objective functional we are interested in depends on the output vector, e.g.,

$$\mathcal{J} = \frac{1}{N}\sum_{k=1}^{N} ||\hat{\boldsymbol{y}}(k)||_2^2 = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{W}_{\text{out}}\boldsymbol{h}(k)||_2^2. \tag{5.15}$$

Thus, the gradient of the objective with respect to the reservoir state is

$$\frac{d\tilde{\mathcal{J}}(\boldsymbol{h}(k))}{d\boldsymbol{h}(k)} = 2\boldsymbol{W}_{\text{out}}^{\top}\boldsymbol{W}_{\text{out}}\boldsymbol{h}(k). \tag{5.16}$$

Minimisation of equation (5.15) defines the optimisation problem where the ESN models the physical dynamical system.

Algorithm 2 Direct-adjoint looping for the computation of parameter sensitivities with PA-ESN

Initialise $\boldsymbol{h}_0(0) \leftarrow \boldsymbol{0}_{n_h}, \boldsymbol{y}(0)$ ▷ Washout initialisation

for $k = 1$ to N_{wash} **do**

$$\boldsymbol{h}_0(k) \leftarrow (1-\alpha_h)\boldsymbol{h}_0(k-1) + \alpha_h \tanh\left(\boldsymbol{W}_{\text{in}}\begin{bmatrix}\boldsymbol{y}(0)\\ \boldsymbol{\xi_p}\odot(\boldsymbol{\theta}-\boldsymbol{g}_\theta)\end{bmatrix} + \boldsymbol{W}_h\boldsymbol{h}_0(k-1)\right)$$

Initialise $\boldsymbol{h}(0) \leftarrow \boldsymbol{h}_0(N_{\text{wash}})$ ▷ Initialise reservoir state

for $k = 1$ to $N_{\text{transient}} + N$ **do** ▷ Run direct solution forward in time (closed-loop)

$$\boldsymbol{h}(k) \leftarrow (1-\alpha_h)\boldsymbol{h}(k-1) + \alpha_h \tanh\left(\boldsymbol{W}_{\text{in}}\begin{bmatrix}\boldsymbol{W}_{\text{out}}\boldsymbol{h}(k-1)\\ \boldsymbol{\xi_p}\odot(\boldsymbol{\theta}-\boldsymbol{g}_\theta)\end{bmatrix} + \boldsymbol{W}_h\boldsymbol{h}(k-1)\right)$$

Remove transient $\boldsymbol{h}[0:N] \leftarrow \boldsymbol{h}[N_{\text{transient}} : N_{\text{transient}} + N]$

Initialise $\boldsymbol{q}^+(N) \leftarrow 1/N\Big(d\tilde{\mathcal{J}}(\boldsymbol{h}(N))/d\boldsymbol{h}(N)\Big)^{\top}$ ▷ Initialise adjoint state

for $k = N-1$ to 1 **do** ▷ Run adjoint solution backward in time

$\partial\boldsymbol{h}(k+1)/\partial\boldsymbol{h}(k) \leftarrow$ Eq. (5.22) ▷ Jacobian wrt. reservoir state vector

$$\boldsymbol{q}^+(k) \leftarrow 1/N\Big(d\tilde{\mathcal{J}}(\boldsymbol{h}(k))/d\boldsymbol{h}(k)\Big)^{\top} + \left(\partial\boldsymbol{h}(k+1)/\partial\boldsymbol{h}(k)\right)^{\top}\boldsymbol{q}^+(k+1)$$

$\partial\boldsymbol{h}(k)/\partial\boldsymbol{\theta} \leftarrow$ Eq. (5.23) ▷ Jacobian wrt. parameters

Accumulate sensitivity $d\mathcal{J}/d\boldsymbol{\theta} \leftarrow d\mathcal{J}/d\boldsymbol{\theta} + \boldsymbol{q}^{+\top}(k)\partial\boldsymbol{h}(k)/\partial\boldsymbol{\theta}$

5.3.2 Sensitivity to the system's initial conditions

We consider the sensitivity of an objective functional of the final reservoir state after N time steps, i.e., $\mathcal{J} = \mathcal{J}(\boldsymbol{h}(N))$ to the initial conditions. In the ESN's equations (5.5), the system's initial conditions $\boldsymbol{y}(0)$ appear in the input as $\boldsymbol{y}_{\text{in}}(0)$. When computing the sensitivity to the initial input vector $\boldsymbol{y}_{\text{in}}(0)$, we need to make a distinction between whether the first step of the propagation of the ESN's equations is performed in open loop or closed loop. Because of the recurrence in the ESN, the full state of the ESN is described by the reservoir state. Therefore, for the ESN's estimate of the sensitivity to match the original system, we first run a washout so that the output of the reservoir synchronises with the original system and then perform the first step in closed loop.

Otherwise, in the open-loop case, $\boldsymbol{y}_{\text{in}}(0)$ only acts as an external input, and the ESN might not at all be sensitive to such perturbations, for example if the leak rate α_h is small so that the next reservoir state heavily relies on the previous reservoir state and not so much on the input vector. (For a use case of the resulting open-loop Jacobian, see an application in data assimilation in Nóvoa et al. (2024b).) The sensitivity to the initial condition is

$$\frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{y}_{\text{in}}(0)} = \frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{h}(N)} \frac{d\boldsymbol{h}(N)}{d\boldsymbol{h}(1)} \left(\frac{\partial \boldsymbol{h}(1)}{\partial \boldsymbol{y}_{\text{in}}(0)} + \frac{\partial \boldsymbol{h}(1)}{\partial \boldsymbol{h}(0)} \frac{d\boldsymbol{h}(0)}{d\boldsymbol{y}_{\text{in}}(0)} \right), \tag{5.17}$$

where the gradients $\partial \boldsymbol{h}(1)/\partial \boldsymbol{y}_{\text{in}}(0)$ and $\partial \boldsymbol{h}(1)/\partial \boldsymbol{h}(0)$ are found by differentiating the open-loop equation (5.5) and the gradients $\partial \boldsymbol{h}(k+1)/\partial \boldsymbol{h}(k),\ i = 1, 2, \ldots, N-1$ are found by differentiating the closed-loop equation (5.11). From the output of the ESN (5.2), we find the gradient $d\boldsymbol{h}(0)/d\boldsymbol{y}_{\text{in}}(0) = \boldsymbol{W}_{\text{out}}^{\dagger}$, where $(.)^{\dagger}$ indicates pseudo-inverse. Alternatively, we compute $d\mathcal{J}(\boldsymbol{h}(N))/d\boldsymbol{h}(0)$ in closed-loop configuration for all time steps and then determine by the chain rule

$$\frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{y}_{\text{in}}(0)} = \frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{h}(0)} \boldsymbol{W}_{\text{out}}^{\dagger}. \tag{5.18}$$

The tangent linear equations are given by

$$\frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{h}(0)} = \frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{h}(N)} \boldsymbol{Q}(N), \tag{5.19a}$$

$$\boldsymbol{Q}(k+1) = \frac{\partial \boldsymbol{h}(k+1)}{\partial \boldsymbol{h}(k)} \boldsymbol{Q}(k), \tag{5.19b}$$

$$\boldsymbol{Q}(0) = \boldsymbol{I}_{n_h \times n_h}, \tag{5.19c}$$

where $\boldsymbol{Q}(k) \equiv d\boldsymbol{h}(k)/d\boldsymbol{h}(0),\ \boldsymbol{Q}(k) \in \mathbb{R}^{n_h \times n_h}$ and $\boldsymbol{I}$ is the identity matrix. The adjoint equations are

$$\frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{h}(0)} = \boldsymbol{q}^{+\top}(0), \tag{5.20a}$$

$$\boldsymbol{q}^{+}(k) = \frac{\partial \boldsymbol{h}(k+1)}{\partial \boldsymbol{h}(k)}^{\top} \boldsymbol{q}^{+}(k+1), \tag{5.20b}$$

$$\boldsymbol{q}^{+}(N) = \frac{d\mathcal{J}(\boldsymbol{h}(N))}{d\boldsymbol{h}(N)}^{\top}. \tag{5.20c}$$

In this case, the number of "parameters" is equal to the dimension of the reservoir state vector, which is typically large, and therefore the adjoint approach significantly reduces the computational cost as compared to finite differences or the tangent linear approach.

5.3.3 Jacobian of the PA-ESN

Computation of the adjoint requires the system's Jacobian and partial derivative with respect to the system's parameters. We first define,

$$\tilde{\boldsymbol{h}}(k) = \frac{\boldsymbol{h}(k+1) - (1-\alpha_h)\boldsymbol{h}(k)}{\alpha_h}, \tag{5.21}$$

which is used to compute the derivative of the $\tanh(\cdot)$ term in the ESN equations, i.e., (5.5). The Jacobian with respect to the reservoir state and the Jacobian with respect to the system's parameters are then given as

$$\frac{\partial \boldsymbol{h}(k+1)}{\partial \boldsymbol{h}(k)} = (1-\alpha_h)\boldsymbol{I}_{n_h \times n_h} + \alpha_h \text{diag}\left(\boldsymbol{1} - \tilde{\boldsymbol{h}}(k) \odot \tilde{\boldsymbol{h}}(k)\right)(\boldsymbol{W}_{\text{in}}^{y}\boldsymbol{W}_{\text{out}} + \boldsymbol{W}_h), \tag{5.22}$$

$$\frac{\partial \boldsymbol{h}(k)}{\partial \boldsymbol{\theta}} = \alpha_h \text{diag}\left(\boldsymbol{1}_{n_h} - \tilde{\boldsymbol{h}}(k-1) \odot \tilde{\boldsymbol{h}}(k-1)\right)\boldsymbol{W}_{\text{in}}^{\theta}\text{diag}(\boldsymbol{\xi}_\theta), \tag{5.23}$$

where $\boldsymbol{I}$ denotes the identity matrix, $\boldsymbol{1}$ is a vector of ones, and $\text{diag}(\cdot)$ denotes a diagonal matrix that has $(\cdot)$ as its diagonal.

5.3.4 Sensitivity to the network weights

The adjoint method also enables the efficient computation of the sensitivities to the network's weights, specifically the output weights $\boldsymbol{W}_{\text{out}}$, which are the trainable weights of the ESN. For training purposes, the objective functional is not a physical quantity but rather the training loss (5.3), i.e.,

$$\mathcal{J} = 1/N \sum_k ||\boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)||_2^2 + \lambda \sum_i ||w_{\text{out},i}||_2^2. \tag{5.24}$$

In this case, we require the gradient of the training loss with respect to the trainable weights. This is similar to the concept of backpropagation through time in the training of RNNs. Adjoint method has also been utilised to train the weights of neural ODEs (Chen et al., 2018). For the ESN, the adjoint equations (5.14) to compute the sensitivities to the weights are the same except we replace the parameter vector with the output weights in the final dot product, i.e., Eq. (5.14a) becomes

$$\frac{d\mathcal{J}}{d\boldsymbol{W}_{\text{out}}} = \sum_{k=1}^{N} \frac{1}{N} \frac{\partial \tilde{\mathcal{J}}(\boldsymbol{h}(k), \boldsymbol{W}_{\text{out}})}{\partial \boldsymbol{W}_{\text{out}}} + \boldsymbol{q}^{+\top}(k) \frac{\partial \boldsymbol{h}(k)}{\partial \boldsymbol{W}_{\text{out}}} + 2\lambda \boldsymbol{W}_{\text{out}}. \tag{5.25}$$

In contrast to the previous cases where we assumed that the objective functional did not explicitly depend on the system's parameters, in this case, we have to account for the partial derivative $\partial \mathcal{J}/\partial \boldsymbol{W}_{\text{out}}$ as the training error explicity depends on the output weights both in the prediction and regularisation terms. Since $\boldsymbol{W}_{\text{out}}$ is a matrix, the partial derivative $\partial \boldsymbol{h}(k)/\partial \boldsymbol{W}_{\text{out}}$ is a 3^{rd} order tensor, which we express using the index notation

$$\left(\frac{\partial \boldsymbol{h}(k)}{\partial \boldsymbol{W}_{\text{out}}}\right)_{lmn} = \frac{\partial h_l(k)}{\partial w_{\text{out},mn}} = \alpha_h \left(1 - \tilde{h}_l^2(k-1)\right) w_{\text{in},lm} h_n(k-1) \tag{5.26}$$

and $l = 1, 2, \ldots, n_h,\ m = 1, 2, \ldots, n_y,\ n = 1, 2, \ldots, n_h$.

5.4 Learning the Lorenz system and parameter sensitivities

We demonstrate the data-driven computation of adjoint sensitivities on the Lorenz system (§ 3.3). We generate the dataset for the training and validation of the ESN by time-marching the ODEs (3.5) for different sets of parameters. The parameters ($\sigma_{\text{L}} = 10, \rho_{\text{L}} = 28, \beta_{\text{L}} = 8/3 \approx 2.667$) lead to a chaotic solution (Lorenz, 1963). We randomly choose 20 regimes for training, and 5 regimes for validation from a grid of parameters, $\sigma_{\text{L}} = \{8, 10, 12, 14, 16\}$, $\rho_{\text{L}} = \{30, 35, 40, 45, 50\}$, and $\beta_{\text{L}} =$

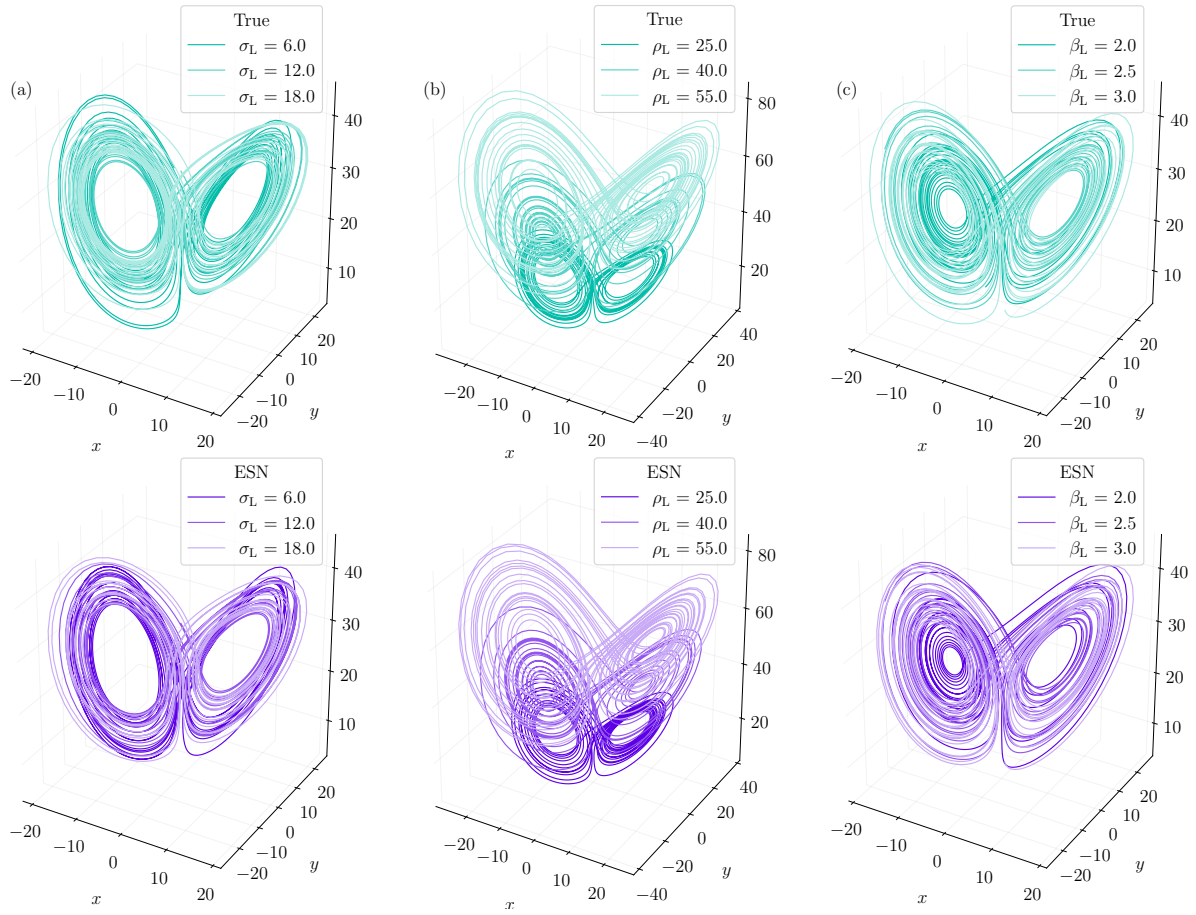


Figure 5.2: Prediction of the Lorenz attractor with varying system's parameters with PA-ESN. Varying (a) σ_L, (b) ρ_L, and (c) β_L, while the rest of the parameters are fixed to ($\sigma_L = 10, \rho_L = 28, \beta_L = 8/3 \approx 2.667$). The PA-ESN captures the variations of the attractor to the parameters.

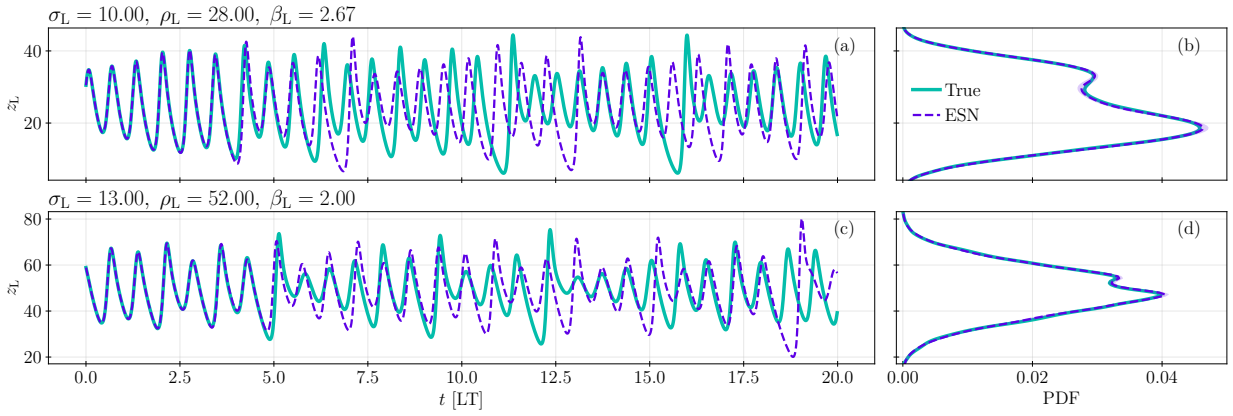


Figure 5.3: Short-term prediction (a, c) and long-term inference of the statistics (b, d) of z_L with PA-ESN. Shown two regimes with different Lyapunov times (LTs); (a, b) ($\sigma_L = 10, \rho_L = 28, \beta_L = 8/3 \approx 2.667$), and (c, d) ($\sigma_L = 13, \rho_L = 52, \beta_L = 1.75$). The statistics are calculated over 1000 LTs for 10 realisations of the ESN (shown mean and one standard deviation in lighter colour), after a washout stage in which we repeatedly feed the same initial condition, and a transient time that is discarded. The PA-ESN can successfully infer the dynamics and long-term statistics of different chaotic regimes even when they were not seen during training.

$\{1, 1.5, 2, 2.5, 3\}$. We observe that it is important to validate the ESN on regimes unseen during the training in order to choose a model that generalises well over a range of different parameters. The

Table 5.1: The Lorenz system's parameters and the parameter-aware echo state network's (PA-ESN) configurations for this dataset.

Parameter	**Value/Configuration**
Default parameters	
σ_{L}	10
ρ_{L}	28
β_{L}	8/3
Training and validation data	
σ_{L}	$\{8, 10, 12, 14, 16\}$
ρ_{L}	$\{30, 35, 40, 45, 50\}$
β_{L}	$\{1, 1.5, 2, 2.5, 3\}$
Time step (Δt)	0.01
Total number of regimes	125
Training/validation split	20/5 regimes (randomly selected)
Echo state network	
Reservoir size (n_h)	1200
Connectivity (n_{conn})	3
Training	
Washout window	4 time units ($N_{\mathrm{wash}} = 400$ steps)
Training data	10 time units ($N_{\mathrm{train}} = 1000$ steps/regime)
Validation	
Number of ESN realisations	5
Number of folds	10
Length of folds	2 time units, ($N_{\mathrm{val}} = 200$ steps each)
Error metric	Average relative ℓ_2-error (5.8)

regimes in the training and validation datasets display chaotic behaviour with varying Lyapunov times (LTs), i.e., the time-scale of divergence of two nearby trajectories in a chaotic system, (between 0.77 and 4.80 time units). The numerical integration is performed with a fourth-order Runge-Kutta scheme with a time-step of $\Delta t = 0.01$. After a transient, we select the first 4 time units for washout ($N_{\mathrm{wash}} = 400$) and 10 time units as training data ($N_{\mathrm{train}} = 1000$). The system's and the ESN's default configuration are provided in Tab. 5.1. The optimal hyperparameters for an ESN of reservoir size, $n_h = 1200$ and a connectivity, $n_{\mathrm{conn}} = 3$ between the reservoir state variables are provided in Tab. 5.2.

In the ESN, the input state is lifted to a high-dimensional reservoir state, so that the output state can be predicted using only a linear readout following a single nonlinear activation layer. For the Lorenz 63 system, the performance of the classical, i.e., parameter-unaware, ESN typically saturates at a reservoir size of $n_h = 300$, when predicting trajectories within a single regime using the default parameters (Tab. 5.1) (Racca and Magri, 2021). Here, we select the reservoir size by comparing the validation losses obtained with increasing reservoir sizes, $n_h = \{300, 600, 1200, 1800\}$. When predicting trajectories across different regimes in the three-dimensional parameter space with a parameter-aware ESN, we found that a larger reservoir compared to the single parameter regime case is required, with the performance saturating around $n_h = 1200$. (The validation error decreases by approximately 77% from $n_h = 300$ to $n_h = 600$, 31% from $n_h = 600$ to $n_h = 1200$ and 13%

Table 5.2: Optimal hyperparameters of the PA-ESN with a reservoir size of $n_h = 1200$ and connectivity $n_{\text{conn}} = 3$ that infers the parameterised dynamics of the Lorenz system. Refer to Eq. (5.5) for the hyperparameters.

	Description	Range	Optimal Value
ρ_h	Spectral radius	0.01–1	2.20×10^{-1}
ξ_{in}	Input scaling	0.001–1	6.79×10^{-2}
$\xi_{\sigma_{\text{L}}}$	σ_{L} input scaling	0.001–0.01	2.80×10^{-3}
$g_{\sigma_{\text{L}}}$	σ_{L} offset	40–100	68.7
$\xi_{\rho_{\text{L}}}$	ρ_{L} input scaling	0.001–0.01	1.50×10^{-3}
$g_{\rho_{\text{L}}}$	ρ_{L} offset	40–100	84.8
$\xi_{\beta_{\text{L}}}$	β_{L} scaling	0.01–0.1	3.93×10^{-2}
$g_{\beta_{\text{L}}}$	β_{L} offset	40–100	74.5
α_h	Leak rate	0.1–1	8.85×10^{-1}
λ	Tikhonov coefficient	10^{-10}–10^{-3}	10^{-10}

from $n_h = 1200$ to $n_h = 1800$.) However, the increased model capacity replaces the need to train separate parameter-unaware models for each parameter regime, as well as enabling the inference of parameter sensitivities. In this work, we do not explicitly perform an optimisation over the reservoir connectivity n_{conn}. When the reservoir size n_h is sufficiently large, the effects of the reservoir size, connectivity, and input scaling can be studied through a single combined hyperparameter (Storm et al., 2022), which makes the choice of connectivity less critical. We have selected connectivity values from a reasonable range found in literature (e.g., Racca and Magri, 2021; Roy et al., 2022). In general, we prefer lower connectivity values, because they result in sparser input and state matrices, which are more computationally efficient for forward-in-time propagation of the dynamics.

The prediction of the Lorenz attractor in the phase space with the trained PA-ESN is shown in Fig. 5.2 with varying system's parameters. The parameters change the geometry of the attractor, which the PA-ESN captures. For example, z_{L} values in the attractor set increase with increasing ρ_{L}. Previous works have applied and analysed the ensemble adjoint method to compute the sensitivity of the time-averaged z_{L}, $\bar{z}_{\text{L}}$, to the parameter ρ_{L} in the Lorenz system due to their nearly linear relationship (e.g., Lea et al., 2000; Eyink et al., 2004). Therefore, we choose $\bar{z}_{\text{L}}$ as the quantity of interest. The short-term prediction and the long-term inference of statistics for z_{L} are illustrated in Fig. 5.3 for two regimes; (i) the reference configuration, ($\sigma_{\text{L}} = 10, \rho_{\text{L}} = 28, \beta_{\text{L}} = 8/3 \approx 2.667$) with a Lyapunov time of $LT = 1.1$, and (ii) a configuration with a shorter Lyapunov time $LT = 0.8$ ($\sigma_{\text{L}} = 13, \rho_{\text{L}} = 52, \beta_{\text{L}} = 2.0$). Neither of these regimes were seen during training or validation. Nonetheless, time-accurate short-term forecasting can be performed with a predictability horizon (5.9) of 4.8 LT for the regime (i), and 5.8 LT for regime (ii) (average over 100 initial conditions on the attractor), and the long-term statistics are captured for a wide range of seen and unseen regimes with a single instance of the parameter-aware ESN. We compute the climate sensitivity of $\bar{z}_{\text{L}}$ to all parameters using an ensemble adjoint method (Lea et al., 2000). Climate sensitivity (3.2) refers to the long-term behaviour of the system, where the sensitivity is independent of initial condition. In the ensemble adjoint method, the sensitivities over an ensemble of short time-series with different initial conditions on the attractor are computed and the climate sensitivity is estimated by the mean of the ensemble (3.4). Although $\bar{z}_{\text{L}}$ varies smoothly with the parameters (Fig. 5.5), in our numerical calculations, the finite difference method implemented by adding a small perturbation

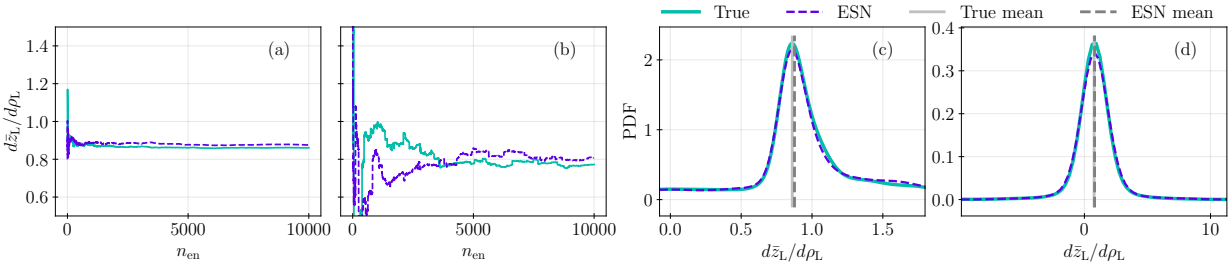


Figure 5.4: Convergence of the ensemble adjoint estimate for the Lorenz system with (a) 0.5 LT and (b) 1 LT. The statistics of the ensemble (two standard deviations from the mean) with $n_{en} = 10^4$ members are shown in (c) for 0.5 LT and (d) 1 LT integration time. There is a trade-off between the bias and variance of the ensemble computation with respect to the integration time.

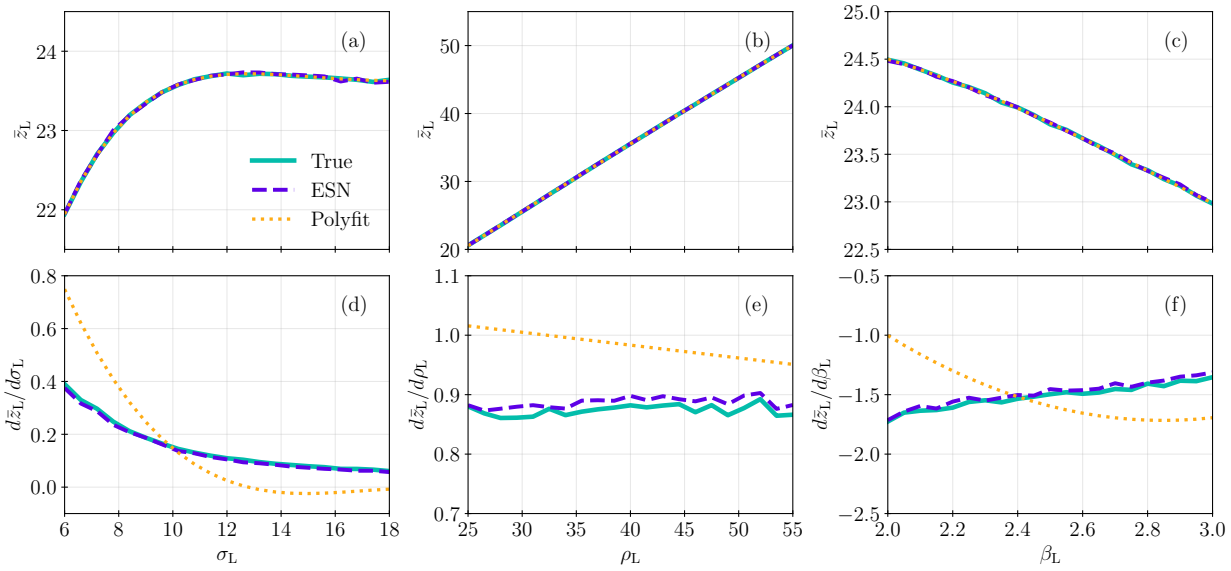


Figure 5.5: Inference of sensitivities of $\bar{z}_L$ to parameters, σ_L, ρ_L, and β_L. Top row (a – c) shows the change of $\bar{z}_L$ for varying σ_L, ρ_L, and β_L, for the true system, PA-ESN, and a polynomial fit on the true system's values. Bottom row (d – f) shows the derivative of $\bar{z}$ with respect to the respective parameter obtained by the ensemble adjoint method using the true system's adjoint, PA-ESN's adjoint, and by differentiating the polynomial fit.

to the parameters and integrating the nonlinear equations for a long time (5000 LT) yielded noisy and inaccurate estimates for the climate sensitivity. In Lea et al. (2000), the climate sensitivity for ρ_L is computed by a linear fit to the $\bar{z}_L$ vs. ρ_L data with other parameters being fixed to their default values. In Fig. 5.5, as a baseline, we obtain direct estimates of the climate sensitivities with a polynomial fit for each of the parameters. (The degree of the polynomial is determined by leave-one-out cross-validation. In this validation method, the polynomial is fitted on all but one data point, and the prediction accuracy of the fit is tested on this point. The process is then repeated for all data points one at a time, with the average error providing the validation score.)

In contrast, the adjoint method has the advantage that the sensitivity to all parameters in any given parameter regime can be calculated with a single operation. However, we also note that for chaotic systems, the ensemble adjoint based sensitivity estimator is associated with a bias and a variance depending on the integration time of each trajectory (Chandramoorthy et al., 2017).

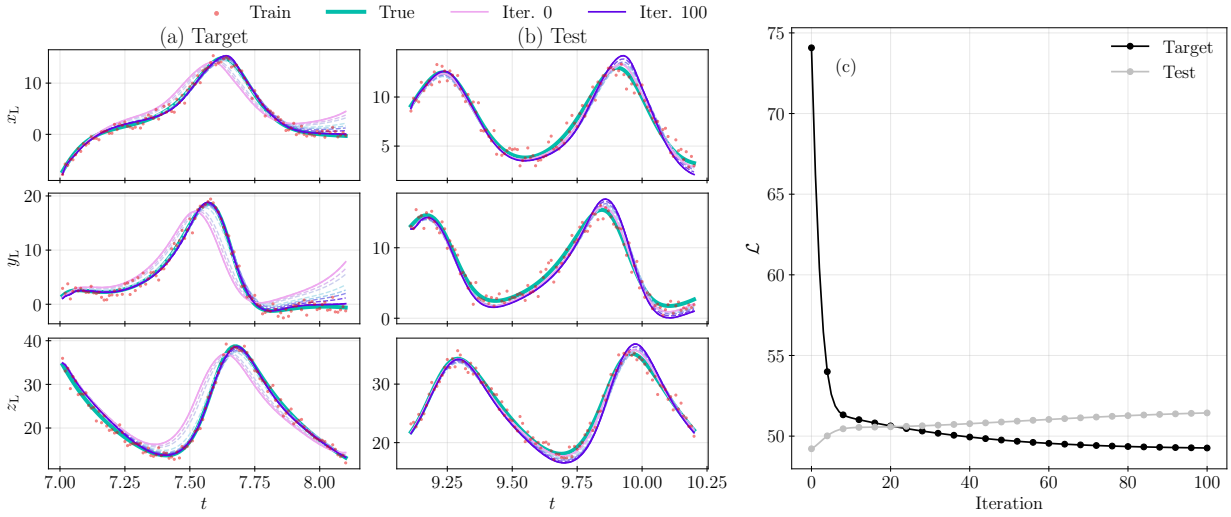


Figure 5.6: Adjoint methods enable efficient gradient updates of the trainable weights of the ESN to minimise the prediction error (plus a regularisation term) on a target trajectory. The closed-loop prediction performance throughout the optimisation process (a) on the target dataset which we are optimising the prediction error for, and (b) on the test dataset. The closed-loop prediction before the gradient descent (Iteration 0) is shown in pink, whereas the final prediction (Iteration 100) is shown in purple. The intermediate stages are shown in dashed lines. In (c) we show the evolution of the loss function (5.24).

While the bias decreases with increasing integration time, the variance increases and the so-called Lévy flights, i.e., long jumps in the mean estimation, appear, requiring more ensemble members for convergence (Eyink et al., 2004). This process is illustrated in Fig. 5.4 for integration times of 0.5 LT and 1.0 LT. We opt for an integration time of half a Lyapunov time and an ensemble of 10000 trajectories, which provide an acceptable trade-off between accuracy and computational cost. Fig. 5.5 compares the objective and the adjoint sensitivity estimates of the ESN with the original system as well as a direct estimate using a polynomial fit of the objective values. Sensitivities pertaining to the parameters σ_L, ρ_L, and β_L, are shown while each of them is varied and the other two are fixed at the given values for the regime (i). The ESN estimates closely match with the true estimates obtained via ensemble adjoint method. The difference between the direct estimate and the adjoint estimates is expected due to the above mentioned bias. We only show the component of the sensitivity associated with the parameter we vary, but in fact the adjoint method has output the sensitivities to all parameters, which similarly match with the true estimates.

5.4.1 Updating weights with adjoint sensitivities

Whilst the focus of this chapter is to infer sensitivities of the true system to the system's parameters and initial conditions, the adjoint methods enable us to efficiently compute the gradient of the training loss with respect to the network's trainable weights, $\boldsymbol{W}_\mathrm{out} \in \mathbb{R}^{n_y \times n_h}$ where n_h is typically high-dimensional (§ 5.3.4). Here, we provide a short demonstration, further investigation of training with adjoint methods is beyond the scope of this thesis. We train a standard ESN of reservoir size $n_h = 300$ and connectivity $n_\mathrm{conn} = 3$ on data from the default Lorenz system (Tab. 5.1). We choose a short time series (5 time units, $N_\mathrm{train} = 500$) and noisy dataset ($\sigma_o = 10\%$), such that the closed-loop prediction performance does not generalise sufficiently well. (The hyperparameters are set to

$\rho_h = 0.4$, $\sigma_{\text{in}} = 0.1$, $\alpha_h = 0.8$, and most importantly the Tikhonov coefficient $\lambda = 10^{-5}$ due to the noise, as it enforces the regularisation, without any optimisation.) We compute the sensitivities to the output weights (5.25) and update the weights using gradient descent with a learning rate of $\lambda_{\text{learn}} = 2 \times 10^{-6}$. The target trajectory, based on which we calculate the prediction error or the data-driven loss, is chosen as a separate time series of 1 LT after the initial training trajectory (also noisy). This process is visualised in Fig. 5.6. The weights can be updated in the closed-loop configuration to minimise the training loss in short time horizons such that the accuracy of the ESN's closed-loop prediction increases. On the test dataset, the loss increases slightly since the weights fit to the target dataset, and then saturates.

5.5 Thermoacoustic echo state network *(T-ESN)*

We propose a parameter-aware ESN with hard-constrained physics for thermoacoustics. As explored in Chapter 4, in the context of machine learning, hard constrainining means embedding prior knowledge into the architecture of the neural network, as opposed to soft constraining, which refers to imposing prior knowledge as a penalty during training (Doan et al., 2020a). This approach makes neural networks more robust and generalisable on unseen data.

Recapping from § 3.4, thermoacoustics are described by a nonlinear wave equation. We use the low-order model of the Rijke tube (§ 3.4.3) as the testbed. The system is described by the evolution of the Galerkin amplitudes for pressure and velocity. We fix the heat source location $x_f = 0.2$, which causes thermoacoustic instabilities to occur. The nonlinearity is contained in the heat release term (3.15), which is coupled with the time-delayed velocity at the heat source location, $u_f(t-\tau) = u(x_f, t-\tau)$. We analyse the dynamics as a function of the heat release strength β and the time-delay τ at the heat source location, which are the parameters of interest. By varying these parameters, the system displays fixed-point, limit-cycle, quasiperiodic, and chaotic solutions. The cost functional is the time-averaged acoustic energy E_{ac}, which we wish to minimise to suppress an instability. We hard-constrain the physical knowledge about the thermoacoustics in the network architecture in two ways: (i) by making the network explicitly time-delayed, and (ii) by designing the input weights matrix.

5.5.1 Design of the T-ESN

First of all, our aim is to develop a parameter-aware ESN that learns the dynamics of the Galerkin amplitudes $\eta_j(t)$ and $\mu_j(t)$ for $j = 1, 2, \ldots, n_g$ in Eqs. (3.13) and (3.14) from the low-order thermoacoustic system introduced in § 3.4.3. Initially, we consider two cases without any hard constraining: (i) the Galerkin amplitudes as the input and output vectors $\boldsymbol{y}_{\text{in}}(k) = \hat{\boldsymbol{y}}(k) = [\hat{\boldsymbol{\eta}}(k);\ \hat{\boldsymbol{\mu}}(k)]$, and (ii) the Galerkin amplitudes and the advection variables that capture the time-delay history (3.16) as the input and output vectors $\boldsymbol{y}_{\text{in}}(k) = \hat{\boldsymbol{y}}(k) = [\hat{\boldsymbol{\eta}}(k);\ \hat{\boldsymbol{\mu}}(k);\ \hat{\boldsymbol{v}}(k)]$. In these cases, we feed both β and τ as parameters through the input channel, $\boldsymbol{\theta} = [\beta;\ \tau]$. However, neither of these options is inherently time delayed. Therefore, we design an ESN for time-delayed systems and refer to it as thermoacoustic ESN (T-ESN). During closed-loop prediction, we compute the delayed velocity at the flame location from past predictions of the velocity Galerkin amplitudes. We then feed the delayed velocity into the network as an input for the prediction of the next time step. One step of

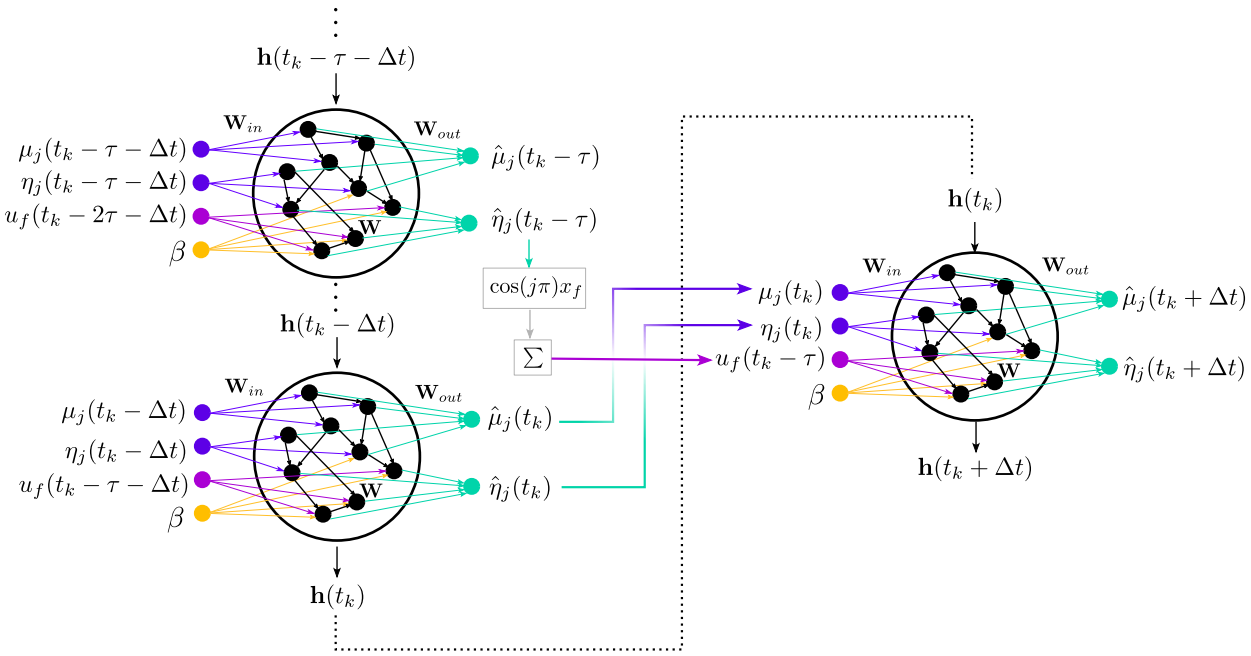


Figure 5.7: Schematic of the proposed thermoacoustic echo state network (T-ESN) in closed-loop configuration. T-ESN hard constrains the physical knowledge about the nonlinear and time-delayed thermoacoustics. The time-delayed velocity at the flame $u_f(t_k - \tau)$ is computed using delayed predictions of the network (5.27a), and then fed to the network as input at time t_k, which embeds the time delay parameter τ in the architecture. The heat release strength parameter β is fed through the input channel similarly to Fig. 5.1.

T-ESN is

$$u_f(k - N_\tau) = \sum_{j=1}^{n_g} \cos(j\pi x_f)\hat{\eta}_j(k - N_\tau), \tag{5.27a}$$

$$\boldsymbol{h}(k+1) = (1 - \alpha_h)\boldsymbol{h}(k) + \alpha_h \tanh(\boldsymbol{W}_{\text{in}}[\boldsymbol{y}_{\text{in}}(k); u_f(k - N_\tau); \xi_\beta(\beta - g_\beta)] + \boldsymbol{W}_h\boldsymbol{h}(k)), \tag{5.27b}$$

$$\hat{\boldsymbol{y}}(k+1) = [\hat{\boldsymbol{\eta}}(k+1);\ \hat{\boldsymbol{\mu}}(k+1)] = \boldsymbol{W}_{\text{out}}\boldsymbol{h}(k+1), \tag{5.27c}$$

where $N_\tau = \text{int}(\tau/\Delta t)$, which implies that the ESN handles delays that are integer multiples of the time step, as it is a discrete-time map. To extend this, one can use interpolation to approximate the velocity at the desired time delay, however this is beyond the scope of this thesis. A schematic of the T-ESN in closed-loop configuration is shown in Fig. 5.7.

Motivated by thermoacoustic physics, we design the input weights matrix based on the following observations; (i) the acoustics are linear and the modes are sparsely connected, (ii) the heat release is nonlinear, (iii) the heat release strength β is only coupled with $u_f(t - \tau)$

$$\boldsymbol{W}_{\text{in}} = \begin{bmatrix} \boldsymbol{w}_{1,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{w}_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{w}_{2n_g,2n_g} & 0 & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{w}_f & \boldsymbol{w}_\beta \end{bmatrix}, \tag{5.28}$$

where $\boldsymbol{w}_{j,j} \in \mathbb{R}^{\text{int}(n_h/(2n_g+1))}$, $j = 1, \ldots, 2n_g, \boldsymbol{w}_f \in \mathbb{R}^{\text{int}(n_h/(2n_g+1))}, \boldsymbol{w}_\beta \in \mathbb{R}^{\text{int}(n_h/(2n_g+1))}$ The

tanh activation acts approximately linearly near zero, so the terms that linearly contribute to the dynamics have a smaller scaling, whereas the nonlinear terms have a larger scaling. Therefore, we scale the input weights that multiply the linear acoustics with one scaling i.e., $\boldsymbol{w}_{j,j} = \xi_{\text{in}}\tilde{\boldsymbol{w}}_{j,j}$, and the input weights that multiply the delayed velocity that contributes to the nonlinear heat release term with another scaling, i.e., $\boldsymbol{w}_f = \xi_f\tilde{\boldsymbol{w}}_f$, because we expect that $\xi_f > \xi_{\text{in}}$ ($\tilde{\boldsymbol{w}}_{j,j}$, $\tilde{\boldsymbol{w}}_f$ and $\boldsymbol{w}_\beta$ are drawn from the uniform distribution $\sim \mathcal{U}(-1,1)$ as before). While doing so we do not impose the linearity, but rather allow the machine to discover this via the optimisation of the scaling terms. Separately, the parameter β is shifted and scaled by $\xi_\beta(\beta - g_\beta)$. The scalings are determined by the hyperparameter search.

5.5.2 Adjoint of the T-ESN

The time-delayed architecture requires the derivation of suitable adjoint equations. First, the tangent linear problem for computing the gradient with respect to the parameters becomes

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^{N}\frac{d\tilde{\mathcal{J}}(\boldsymbol{h}(k))}{d\boldsymbol{h}(k)}\boldsymbol{Q}(k), \tag{5.29a}$$

$$\boldsymbol{Q}(k+1) = \frac{\partial\boldsymbol{h}(k+1)}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{h}(k+1)}{\partial\boldsymbol{h}(k)}\boldsymbol{Q}(k) + \frac{\partial\boldsymbol{h}(k+1)}{\partial\boldsymbol{h}(k-N_\tau)}\boldsymbol{Q}(k-N_\tau), \tag{5.29b}$$

$$\boldsymbol{Q}(\leq 0) = \boldsymbol{0}. \tag{5.29c}$$

The adjoint of the time-delayed system is time-advanced

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \sum_{i=1}^{N}\boldsymbol{q}^{+\top}(k)\frac{\partial\boldsymbol{h}(k)}{\partial\boldsymbol{\theta}}, \tag{5.30a}$$

$$\boldsymbol{q}^{+}(k) = \frac{1}{N}\frac{d\tilde{\mathcal{J}}(\boldsymbol{h}(k))}{d\boldsymbol{h}(k)}^{\top} + \frac{\partial\boldsymbol{h}(k+1)}{\partial\boldsymbol{h}(k)}^{\top}\boldsymbol{q}^{+}(k+1) + \frac{\partial\boldsymbol{h}(k+1+N_\tau)}{\partial\boldsymbol{h}(k)}^{\top}\boldsymbol{q}^{+}(k+1+N_\tau), \tag{5.30b}$$

$$\boldsymbol{q}^{+}(N) = \frac{1}{N}\frac{d\tilde{\mathcal{J}}(\boldsymbol{h}(N))}{d\boldsymbol{h}(N)}^{\top}, \tag{5.30c}$$

$$\boldsymbol{q}^{+}(\geq N) = \boldsymbol{0}. \tag{5.30d}$$

The full derivation of the adjoint of time-delayed systems can be found in Appendix § B.2. The tangent linear and adjoint equations for the initial condition sensitivity are found similarly by including the time-delayed or time-advanced Jacobian to the Eqs. (5.19) and (5.20) in § 5.3, respectively.

To obtain the sensitivity to time delay, we need

$$\frac{\partial\boldsymbol{h}(k+1)}{\partial\tau} = \frac{\partial\boldsymbol{h}(k+1)}{\partial u_f(t_k-\tau)}\frac{\partial u_f(t_k-\tau)}{\partial\tau} \tag{5.31}$$

In the delayed formulation of the T-ESN, the delayed velocity $u_f(t_k-\tau)$ (5.27a) does not continuously depend on τ, i.e., it is given in discrete time $u_f(k-N_\tau)$. We approximate the gradient $\partial u_f(t_k-\tau)/\partial\tau$ with a central finite difference

$$\frac{\partial u_f(t_k-\tau)}{\partial\tau} \approx \frac{u_f(k-(N_\tau+1)) - u_f(k-(N_\tau-1))}{2\Delta t}. \tag{5.32}$$

The direct-adjoint looping using T-ESN is provided in Alg. 3.

5.5.3 Jacobian of the T-ESN

For the thermoacoustic ESN, we have the following gradients

$$\frac{\partial \boldsymbol{h}(k+1)}{\partial u_f(k-N_\tau)} = \alpha_h \mathrm{diag}(1-\tilde{\boldsymbol{h}}(k)\odot\tilde{\boldsymbol{h}}(k))\boldsymbol{W}_{\mathrm{in}}^{u_f}, \tag{5.33}$$

$$\frac{\partial \boldsymbol{h}(k+1)}{\partial \boldsymbol{h}(k-N_\tau)} = \frac{\partial \boldsymbol{h}(k+1)}{\partial u_f(k-N_\tau)}\boldsymbol{\Psi}\boldsymbol{W}_{\mathrm{out}}, \tag{5.34}$$

where $\boldsymbol{\Psi} = [\cos(\pi x_f)\ \cos(2\pi x_f)\ \ldots\ \cos(n_g\pi x_f)\ \boldsymbol{0}_{n_g}^\top]$ contains the Galerkin modes associated with velocity and masks the Galerkin modes associated with pressure.

5.6 Learning thermoacoustics and parameter sensitivities

In this section, we learn the dynamics of the Rijke tube parameterised by the heat release strength, β, and the time delay, τ. First, we focus on limit-cycle regimes and find that the thermoacoustic ESN (T-ESN) outperforms other parameter-aware ESN architectures in terms of generalisability over unseen regimes. Then, using the trained T-ESN, we infer the adjoint sensitivities of the acoustic energy to these parameters. We compare the time-series predictions and inferred adjoint sensitivities to the ground truth. The ground truth is obtained by integrating the original system described by the ODEs (3.14), and its adjoint equations (2.19), which requires the computation of the Jacobian of the ODEs (3.14).

5.6.1 Training and validation

For training and validation, we create a dataset from time-delays, $\tau = \{0.1, 0.15, 0.2, 0.25, 0.3\}$ and heat release strengths, $\beta = \{1.0, 2.0, 3.0, 4.0, 5.0\}$, giving a total of 25 thermoacoustic regimes. These parameters generate limit-cycle solutions except for $(\beta = 5.0, \tau = 0.15)$, which produces a quasiperiodic solution. We fix the flame location $x_f = 0.2$, modal damping coefficients $d_1 = 0.1$ and $d_2 = 0.06$, the number of Galerkin modes $n_g = 4$, and the number of Chebyshev discretisation $n_c = 10$ (Huhn and Magri, 2020). The resulting system of ODEs has a dimension of $2n_g + n_c = 18$. Previous studies suggest that $n_g = 10$ offers a good trade off between accuracy and computational cost (Huhn and Magri, 2020). We have opted for generating a dataset with $n_g = 4$ to make the analysis of the results simpler, whilst still retaining the dominant thermoacoustic dynamics, as most of the energy is captured by the first four modes. We further note that for the chosen flame location $x_f = 0.2$, the fifth and tenth modes have zero contribution due to the heat release term vanishing in the energy equation (3.14b) as $\sin(j\pi x_f) = 0$ for $j \in \{5, 10\}$. We have separately trained the T-ESN on a dataset with $n_g = 10$ following hyperparameter optimisation (but with the same reservoir size and connectivity as the $n_g = 4$ case), and confirmed that the predictive performance remains comparable (§ 5.6.3).

We generate the data by time-marching the ODEs (3.10) using the `scipy.odeint` tool with a time step of 0.001, which we downsample to 0.01 for the ESN. We remove a transient of 200 time units, select the next 4 time units as washout ($N_{\mathrm{wash}} = 400$) and 8 time units as training data

Algorithm 3 Direct-adjoint looping for the computation of the sensitivity of the acoustic energy to the flame parameters with thermoacoustic echo state network (T-ESN)

Initialise $\boldsymbol{h}_0(0) \leftarrow \boldsymbol{0}_{n_h},\ \boldsymbol{\eta}(0)\ \boldsymbol{\mu}(0)\ u_f(-N_\tau)$ ▷ Washout initialisation

for $k = 1$ to N_{wash} **do**

$$\boldsymbol{h}_0(k) \leftarrow (1-\alpha_h)\boldsymbol{h}_0(k-1) + \alpha_h \tanh\left(\boldsymbol{W}_{\text{in}}\begin{bmatrix}\boldsymbol{\eta}(0)\\ \boldsymbol{\mu}(0)\\ u_f(-N_\tau)\\ \xi_\beta(\beta-g_\beta)\end{bmatrix} + \boldsymbol{W}_h\boldsymbol{h}_0(k-1)\right)$$

Initialise $\boldsymbol{h}(-N_\tau:0) \leftarrow \boldsymbol{h}_0(N_{\text{wash}}-N_\tau:N_{\text{wash}})$ ▷ Initialize reservoir state

for $k = 1$ to $N_{\text{transient}}+N$ **do** ▷ Run direct solution forward in time (closed loop)

$\hat{\boldsymbol{y}}(k-1) \leftarrow \boldsymbol{W}_{\text{out}}\boldsymbol{h}(k-1)$ ▷ Predict Galerkin coefficients

$(\hat{\boldsymbol{\eta}}(k-1),\ \hat{\boldsymbol{\mu}}(k-1)) \leftarrow \hat{\boldsymbol{y}}(k-1)$

$\hat{\boldsymbol{y}}(k-N_\tau-1) \leftarrow \boldsymbol{W}_{\text{out}}\boldsymbol{h}(k-N_\tau-1)$

$(\hat{\boldsymbol{\eta}}(k-N_\tau-1),\ \hat{\boldsymbol{\mu}}(k-N_\tau-1)) \leftarrow \hat{\boldsymbol{y}}(k-N_\tau-1)$

$\hat{u}_f(k-N_\tau-1) \leftarrow \sum_j \cos(j\pi x_f)\hat{\eta}_j(k-N_\tau-1)$ ▷ Predict time-delayed velocity

$$\boldsymbol{h}(k) \leftarrow (1-\alpha_h)\boldsymbol{h}(k-1) + \alpha_h \tanh\left(\boldsymbol{W}_{\text{in}}\begin{bmatrix}\hat{\boldsymbol{\eta}}(k-1)\\ \hat{\boldsymbol{\mu}}(k-1)\\ u_f(k-N_\tau-1)\\ \xi_\beta(\beta-g_\beta)\end{bmatrix} + \boldsymbol{W}_h\boldsymbol{h}(k-1)\right)$$

Remove transient $\boldsymbol{h}(-N_\tau:N_\tau+N) \leftarrow \boldsymbol{h}(N_{\text{transient}}-N_\tau:N_{\text{transient}}+N)$

$\mathcal{J} \leftarrow \langle \hat{E}_{\text{ac}} \rangle$ (3.19) ▷ Predicted time-averaged acoustic energy

$d\tilde{\mathcal{J}}(\boldsymbol{h}(N))/d\boldsymbol{h}(N) \leftarrow 1/2\boldsymbol{W}_{\text{out}}^\top\hat{\boldsymbol{y}}(N)$ ▷ Gradient wrt. $\hat{E}_{\text{ac}}$ (3.18)

Initialise $\boldsymbol{q}^+(N) \leftarrow 1/N\Big(d\tilde{\mathcal{J}}(\boldsymbol{h}(N))/d\boldsymbol{h}(N)\Big)^\top$ ▷ Initialise adjoint state

Initialise $\boldsymbol{q}^+(N+1:N+N_\tau) \leftarrow \boldsymbol{0}_{n_h\times N_\tau}$

Initialise $d\mathcal{J}/d\boldsymbol{\theta} \leftarrow \boldsymbol{0}_{n_\theta}^\top$

for $k = N-1$ to 1 **do** ▷ Run adjoint solution backward in time

$\partial\boldsymbol{h}(k+1)/\partial\boldsymbol{h}(k) \leftarrow$ Eq. (5.22) ▷ Jacobian wrt. reservoir state vector

$\partial\boldsymbol{h}(k+1+N_\tau)/\partial\boldsymbol{h}(k) \leftarrow$ Eq. (5.34) ▷ Time-advanced Jacobian

$d\tilde{\mathcal{J}}(\boldsymbol{h}(k))/d\boldsymbol{h}(k) \leftarrow 1/2\boldsymbol{W}_{\text{out}}^\top\hat{\boldsymbol{y}}(k)$

$$\boldsymbol{q}^+(k) \leftarrow 1/N\Big(d\tilde{\mathcal{J}}(\boldsymbol{h}(k))/d\boldsymbol{h}(k)\Big)^\top + \big(\partial\boldsymbol{h}(k+1)/\partial\boldsymbol{h}(k)\big)^\top\boldsymbol{q}^+(k+1) + \big(\partial\boldsymbol{h}(k+1+N_\tau)/\partial\boldsymbol{h}(k)\big)^\top\boldsymbol{q}^+(k+1+N_\tau)$$

$\partial\boldsymbol{h}(k)/\partial\beta \leftarrow$ Eq. (5.23) ▷ Jacobian wrt. β

$\partial\boldsymbol{h}(k)/\partial\tau \leftarrow$ Eq. (5.31) ▷ Jacobian wrt. τ

$\partial\boldsymbol{h}(k)/\partial\boldsymbol{\theta} \leftarrow \Big[\partial\boldsymbol{h}(k)/\partial\beta \quad \partial\boldsymbol{h}(k)/\partial\tau\Big]$ ▷ Append

Accumulate sensitivity $d\mathcal{J}/d\boldsymbol{\theta} \leftarrow d\mathcal{J}/d\boldsymbol{\theta} + \boldsymbol{q}^{+\top}(k)\partial\boldsymbol{h}(k)/\partial\boldsymbol{\theta}$

Table 5.3: The Rijke tube's parameters and the PA-ESN's configurations for this dataset.

Parameter	**Value/Configuration**
System constants	
Flame location (x_f)	0.2
Damping coefficients	$d_1 = 0.1$, $d_2 = 0.06$
Galerkin modes (n_g)	4
Chebyshev discretisation (n_c)	10
Solver time step (Δt)	0.001
Training & validation data	
Heat release strengths (β)	$\{1.0, 2.0, 3.0, 4.0, 5.0\}$
Time delays (τ)	$\{0.1, 0.15, 0.2, 0.25, 0.3\}$
Total number of regimes	25
Train/validation split	20/5 regimes
Echo state network	
Reservoir size (n_h)	1200
Connectivity (n_{conn})	20
Downsampled time step (Δt)	0.01
Training	
Transient	200 time units
Washout window	4 time units ($N_{\text{wash}} = 400$ steps)
Training data	8 time units ($N_{\text{train}} = 800$ steps/regime)
Validation	
Number of ESN realisations	5
Number of folds	5
Length of folds	4 time units, $N_{\text{val}} = 400$ steps each
Error metric	Average relative ℓ_2-error (5.8)

($N_{\text{train}} = 800$ per regime). We found that increasing the training length further does not improve the performance whereas it increases the computational effort. We randomly split this dataset into 20 regimes for training and 5 regimes for validation of the hyperparameters. During validation, we evaluate the closed-loop performance of 5 random realisations of the ESN, i.e., realisations of the randomly generated input and state matrices, over 5 trajectories, or folds, of length 4 time units (approximately 2 periods) starting from random time steps. We compute the validation error as the relative ℓ_2-error (5.8), which we average over the number of folds, number of validation regimes, and number of realisations. We employ a Bayesian optimisation scheme using the `skopt.gp_minimise` tool to determine the set of hyperparameters that have the minimal validation error (Racca and Magri, 2021). The configurations are summarised in Tab. 5.3

We consider three networks: (i) the standard parameter-aware ESN (5.5) with the Galerkin amplitudes as the input and output vectors $\boldsymbol{y}_{\text{in}}(k) = \hat{\boldsymbol{y}}(k) = [\hat{\boldsymbol{\eta}}(k);\ \hat{\boldsymbol{\mu}}(k)]$, (ii) we augment the input and output vectors with the advection variable $\boldsymbol{y}_{\text{in}}(k) = \hat{\boldsymbol{y}}(k) = [\hat{\boldsymbol{\eta}}(k);\ \hat{\boldsymbol{\mu}}(k);\ \hat{\boldsymbol{v}}(k)]$, and (iii) thermoacoustic ESN (T-ESN) (5.27). The standard and the advection-augmented parameter-aware ESNs have both β and τ as explicit parameters in the input channel, i.e., in Eq. (5.5) $\boldsymbol{\theta} = [\beta;\ \tau]$. The T-ESN has β as a parameter in the input channel and τ is modelled as a time-delay in the delayed velocity term in the input channel $u_f(k - N_\tau) \approx u_f(t_k - \tau)$ (5.27a). These networks are

Table 5.4: Summary of the PA-ESN architectures for thermoacoustics data.

Network Type	Input Vector $\boldsymbol{y}_{\text{in}}(k)$	Output Vector $\hat{\boldsymbol{y}}(k)$	Parameters
ESN (i): Standard	$\begin{bmatrix}\hat{\boldsymbol{\eta}}(k)\\ \hat{\boldsymbol{\mu}}(k)\end{bmatrix}$	$\begin{bmatrix}\hat{\boldsymbol{\eta}}(k)\\ \hat{\boldsymbol{\mu}}(k)\end{bmatrix}$	$\boldsymbol{\theta} = \begin{bmatrix}\beta\\ \tau\end{bmatrix}$
ESN (ii): Advection-augmented	$\begin{bmatrix}\hat{\boldsymbol{\eta}}(k)\\ \hat{\boldsymbol{\mu}}(k)\\ \hat{\boldsymbol{v}}(k)\end{bmatrix}$	$\begin{bmatrix}\hat{\boldsymbol{\eta}}(k)\\ \hat{\boldsymbol{\mu}}(k)\\ \hat{\boldsymbol{v}}(k)\end{bmatrix}$	$\boldsymbol{\theta} = \begin{bmatrix}\beta\\ \tau\end{bmatrix}$
ESN (iii): Thermoacoustic	$\begin{bmatrix}\hat{\boldsymbol{\eta}}(k)\\ \hat{\boldsymbol{\mu}}(k)\\ u_f(k-N_\tau)\end{bmatrix}$	$\begin{bmatrix}\hat{\boldsymbol{\eta}}(k)\\ \hat{\boldsymbol{\mu}}(k)\end{bmatrix}$	$\boldsymbol{\theta} = [\beta]$

Table 5.5: Optimal hyperparameters of (i) standard PA-ESN (5.5), (ii) advection-augmented PA-ESN, (iii) thermoacoustic ESN (T-ESN) (5.27) trained on data from $\beta = \{1.0, 2.0, 3.0, 4.0, 5.0\}$ and $\tau = \{0.1, 0.15, 0.2, 0.25, 0.3\}$ regimes (limit-cycle regimes) of the Rijke tube.

	Description	Range	ESN (i)	ESN (ii)	ESN (iii)
ρ_h	Spectral radius	0.01–1.0	3.63×10^{-2}	3.67×10^{-2}	1.25×10^{-2}
ξ_{in}	Input scaling	0.01–2.0	3.48×10^{-1}	3.29×10^{-1}	7.90×10^{-2}
ξ_f	u_f scaling	0.01–2.0	—	—	1.05
ξ_β	β scaling	0.01–2.0	1.52×10^{-2}	1.62×10^{-2}	1.09×10^{-1}
g_β	β offset	-10.0–10.0	-5.38	-5.43	-10.0
ξ_τ	τ scaling	0.01–2.0	7.74×10^{-1}	7.32×10^{-1}	—
g_τ	τ offset	-1.0–1.0	-2.06×10^{-1}	7.27×10^{-1}	—
α_h	Leak rate	0.01–1.0	5.97×10^{-2}	6.27×10^{-2}	3.63×10^{-2}
λ	Tikhonov coefficient	10^{-6}–10^{-1}	10^{-5}	10^{-5}	10^{-6}

summarised in Tab. 5.4. For comparison, we fix the reservoir size, i.e., number of reservoir state variables, $n_h = 1200$ and connectivity of the reservoir state variables $n_{\text{conn}} = 20$, i.e., average number of nonzero elements per row in the state matrix $\boldsymbol{W}_h$. The optimal hyperparameters are provided in Tab. 5.5. The third column of Tab. 5.5 shows the ranges from which the hyperparameters are sampled during the validation; ρ_h, ξ_{in}, ξ_f, ξ_β, ξ_τ, α_h and λ are sampled uniformly on a $\log_{10}$ scale, while g_β and g_τ are sampled uniformly.

5.6.2 Prediction performance

The computed adjoint sensitivities are only as accurate as the model we have. Therefore, the first task of the ESN is to accurately emulate the dynamics of regimes both seen and unseen during training. This includes two subtasks: (i) time-accurate prediction from a given initial condition, and (ii) convergence to the correct attractor. The latter is necessary for long-term performance because we are interested in the sensitivity of long-time averaged quantities. Fig. 5.8 shows the short-term prediction performance of the standard, advection-augmented and thermoacoustic ESNs over a grid of β and τ values. For a given regime, after a washout stage, the ESN is run in closed-loop for prediction. This means that the network is forecasting the dynamics. (Since we are interested in quantifying the short-term accuracy of the networks' forecast, we assume that we have washout data for all regimes, which serves as initial condition). The relative ℓ_2 error (5.8) between prediction and ground truth of $[\boldsymbol{\eta};\ \boldsymbol{\mu}]$ over 20 time units is averaged over 5 folds starting

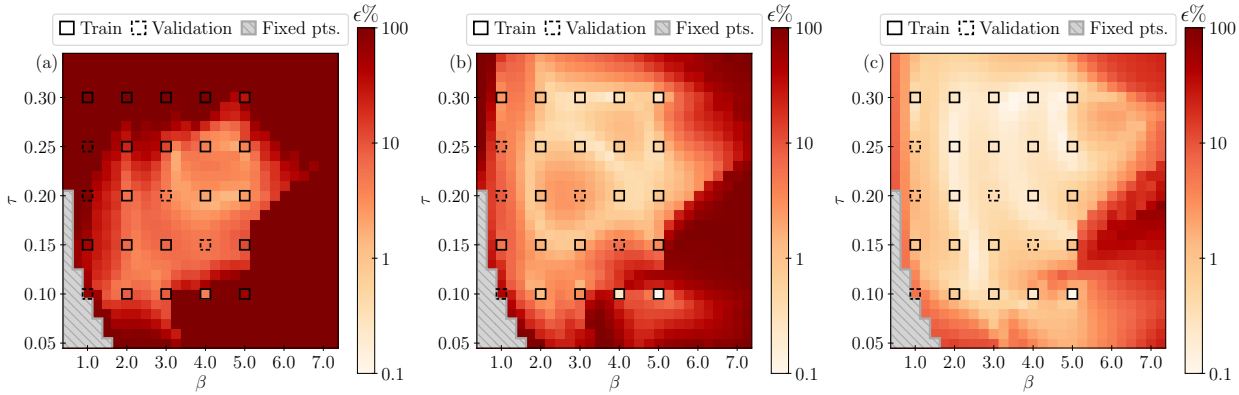


Figure 5.8: Short-term closed-loop prediction performance of (a) standard PA-ESN (5.5), (b) advection-augmented PA-ESN (5.5), (c) T-ESN (5.27) on the Rijke tube data. Relative ℓ_2 error between prediction and ground truth of $[\boldsymbol{\eta};\ \boldsymbol{\mu}]$ over 20 time units on a $\log_{10}$ scale.

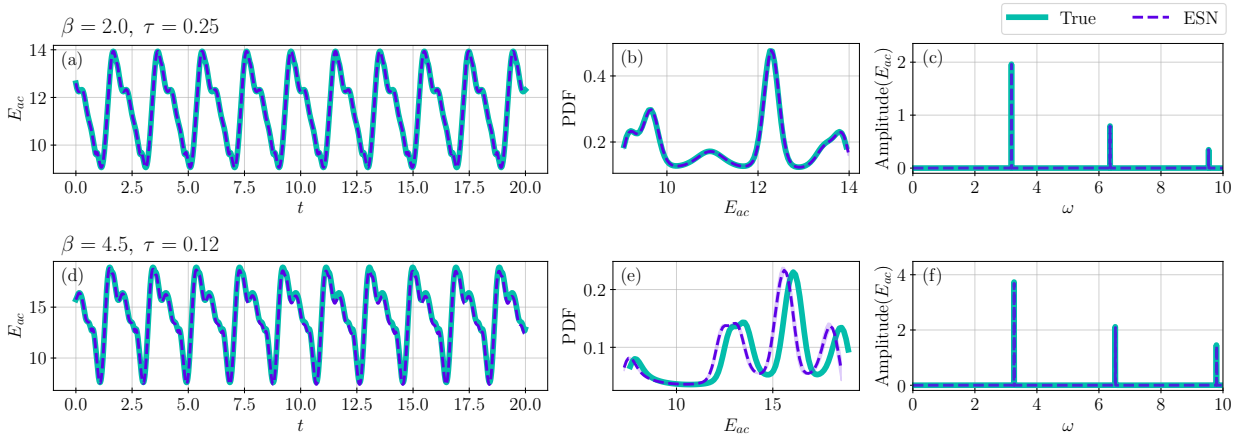


Figure 5.9: T-ESN learns limit-cycle regimes both seen and unseen during training. Short-term prediction, probability distribution function (PDF), and amplitude spectrum of the acoustic energy of (a – c) a limit-cycle regime seen during training ($\beta = 2.0,\ \tau = 0.25$) and (d – f) an unseen limit-cycle regime ($\beta = 4.5,\ \tau = 0.12$). In the PDFs, the mean of an ensemble of 5 realisations of the ESN is shown, one standard deviation from the mean is plotted in lighter colour.

from random time steps, and 5 realisations of the ESN. Standard parameter-aware ESN is not capable of sustaining the limit-cycle solutions for approximately 10 periods even for the training regimes, whereas the augmentation of the advection variables $\boldsymbol{v}$ in the input and output significantly improves the performance. The T-ESN generalises to unseen regimes. Owing to the nonlinearity in the system, the prediction performance decreases as we move away from the training regimes.

In Fig. 5.9, we show the performance of the T-ESN on (i) a limit-cycle regime seen during training ($\beta = 2.0,\ \tau = 0.25$), and (ii) an unseen limit-cycle regime ($\beta = 4.5,\ \tau = 0.12$). Based on Fig. 5.8, we choose regime (ii) as an example of an unseen regime with higher relative ℓ_2 error compared to the other limit-cycle regimes. In these figures, we assess the performance by (from left to right) short-term time-series prediction for 20 time units after washout, the probability density function (PDF), i.e., statistics, and amplitude spectrum of the acoustic energy E_{ac} (3.18). The acoustic energy is defined as a quadratic function of the pressure and velocity fluctuations that is integrated over the spatial domain, and it provides a scalar measure for the energy of the oscillations. Because

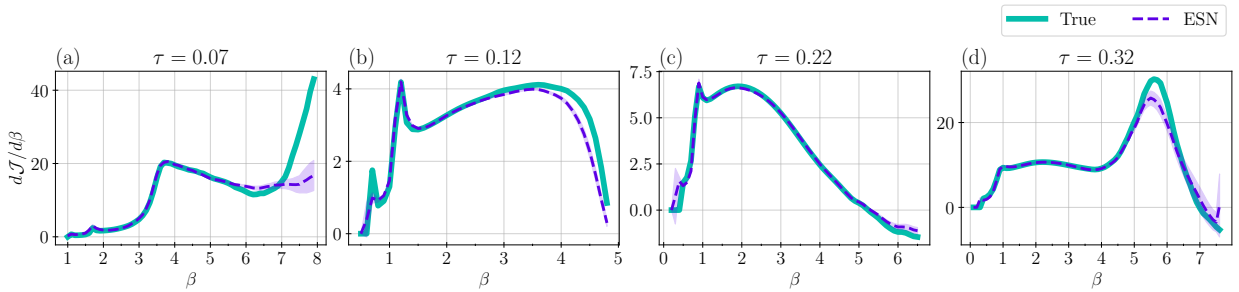


Figure 5.10: T-ESN infers sensitivity of the time-averaged acoustic energy, i.e., objective $\mathcal{J}$, to heat release strength β. Shown $d\mathcal{J}/d\beta$ at fixed (a) $\tau = 0.07$, (b) $\tau = 0.12$, (c) $\tau = 0.22$, (d) $\tau = 0.32$. The mean of the sensitivity of an ensemble of 5 realisations of the ESN is shown, one standard deviation from the mean is plotted in lighter colour.

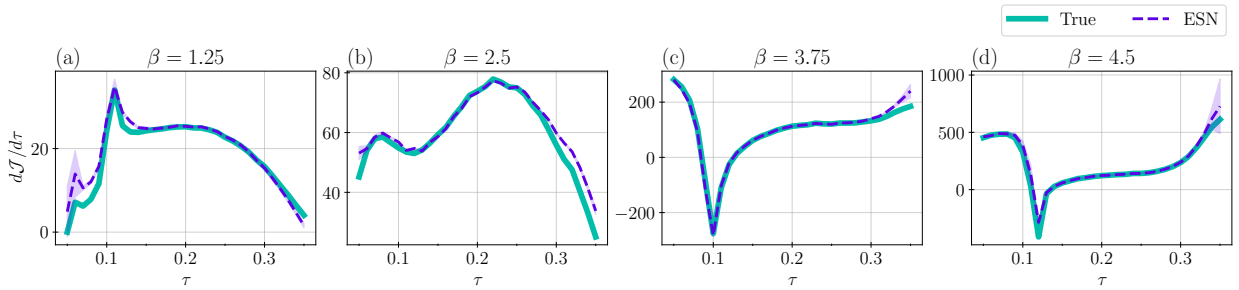


Figure 5.11: T-ESN infers sensitivity of the time-averaged acoustic energy, i.e., objective $\mathcal{J}$, to time-delay τ. Shown $d\mathcal{J}/d\tau$ at fixed (a) $\beta = 1.25$, (b) $\beta = 2.5$, (c) $\beta = 3.75$, (d) $\beta = 4.5$. The mean of the sensitivity of an ensemble of 5 realisations of the ESN is shown, one standard deviation from the mean is plotted in lighter colour.

of this quadratic dependence, its amplitude spectrum contains harmonics that do not appear in the spectra of the individual pressure and velocity signals. While we have separately verified the prediction accuracy for pressure and velocity (see Fig. 5.8), here, we focus on the acoustic energy because it is the quantity of interest for the optimisation problem we formulate for the thermoacoustic system (§ 5.7), where the objective is to suppress the oscillations by minimising the acoustic energy. For the PDF and amplitude spectrum, we feed the initial condition of zeros except $\eta_1(t_0) = 1.0$ in the washout stage, let the ESN evolve in closed loop, discard a transient of 200 time units and use the next 1000 time units.(While computing the amplitude spectrum we remove the mean of the signal.) This procedure and results verify that the ESN has learned the correct limit-cycle attractor because it converges to this set of states from an initial condition that is not in the training set.

5.6.3 Computation of adjoint sensitivities

We compare the adjoint sensitivities computed by the T-ESN following the procedure described in § 5.5.2 with the adjoint sensitivities obtained from the original system described by Eq. (3.14). We refer to the latter as the ground truth, which is computed by deriving the Jacobian of the ODEs (3.10) and integrating the adjoint equations (2.19). The sensitivities are shown for varying β and τ values in Figs. 5.10 and 5.11, respectively. For ease of results, we show the sensitivity to β when β is varied and τ is fixed, and vice versa, however recall that the adjoint computation outputs

the gradient with respect to all parameters. In the washout stage, we feed the initial condition of zeros except $\eta_1(t_0) = 1.5$, assuming that no data is available from the original system for the washout. This value of $\eta_1(t_0)$ makes sure that the trajectories converge to the stable limit cycle and not the stable fixed point in the multistable regimes. After the washout and the transient, we solve the adjoint equations around a trajectory of 100 time units, for which we find that the adjoint sensitivity converges to a steady-state value. The learned adjoint sensitivities match closely with the ground truth, while the results are also robust to different realisations of the network. Even though the training data was generated from discrete values of (β, τ) on a grid (Fig. 5.8), we are able to obtain sensitivity information from a range of (β, τ) values that are unseen during the training using a single instance of the T-ESN. The smoothness of the sensitivity curve also indicates that the dynamics of the T-ESN varies smoothly with the parameters. The accuracy decreases as we move away from the training regimes, e.g., beyond $\beta = 5.0$ in the $d\mathcal{J}/d\beta$ plot at $\tau = 0.32$ (Fig. 5.10 (d)), but the ESN captures the trend of the sensitivities.

Figs. 5.10 and 5.11 also show that the T-ESN infers the existence of bifurcations to fixed point solutions. The fixed point of the Rijke tube is the zero vector, i.e., the mean flow, which means that the sensitivity is zero for the fixed point regimes, e.g., at fixed $\tau = 0.22$ (Fig. 5.10 (c)) for $\beta < 0.45$. The prediction of this transition is shown in more detail in § 5.9.

For validation of scalability to higher dimensions, we separately train the T-ESN on a dataset with $n_g = 10$ Galerkin modes after running the hyperparameter optimisation procedure using the same reservoir size and connectivity as the $n_g = 4$ case. We compare the computed sensitivities across representative regimes. For the limit cycle regime, $(\beta = 2.0, \tau = 0.25)$ shown in Fig. 5.9 (a – c), the ground-truth parameter sensitivities are $(d\mathcal{J}/d\beta = 7.86, d\mathcal{J}/d\tau = 53.63)$ for $n_g = 4$, and $(d\mathcal{J}/d\beta = 7.65, d\mathcal{J}/d\tau = 51.24)$ for $n_g = 10$, changing only slightly. A single realisation of the T-ESN trained on the $n_g = 4$ dataset predicts these sensitivities as $(d\mathcal{J}/d\beta = 7.70, d\mathcal{J}/d\tau = 53.04)$, while the T-ESN trained on $n_g = 10$ dataset predicts them as $(d\mathcal{J}/d\beta = 7.91, d\mathcal{J}/d\tau = 51.72)$. Both predictions match closely with the ground truth. In the unseen limit cycle regime $(\beta = 4.5, \tau = 0.12)$ shown in Fig. 5.9 (d – f), where the prediction error is the highest (Fig. 5.8) and the objective functional has a saddle (§ 5.7), the ground-truth parameter sensitivities are more affected by the system size: $(d\mathcal{J}/d\beta = 3.19, d\mathcal{J}/d\tau = -407.56)$ for $n_g = 4$, and $(d\mathcal{J}/d\beta = 0.91, d\mathcal{J}/d\tau = -3.90)$ for $n_g = 10$. In this case, the T-ESN's predictions of the sensitivities remain accurate with respect to the underlying dynamics of its training data: It predicts $(d\mathcal{J}/d\beta = 2.57, d\mathcal{J}/d\tau = -316.81)$ for the $n_g = 4$ dataset, and $(d\mathcal{J}/d\beta = 0.86, d\mathcal{J}/d\tau = -6.16)$ for the $n_g = 10$ dataset.

5.7 Gradient-based optimisation of thermoacoustics

We apply the inferred adjoint sensitivities to a gradient-based optimisation framework. One realisation of the T-ESN suffices for the computation of the gradients and optimisation for all parameters. In this work, the goal of the optimisation is to determine the set of parameters (β, τ) that minimises the time-averaged acoustic energy (3.18). We restrict the domain to $\beta = [0.5, 5.5]$, $\tau = [0.05, 0.35]$, which prevents the parameters to take negative (non-physical) values, and the system to bifurcate to quasiperiodic or chaotic regimes. We wish to avoid this during gradient-based optimisation because first, the acoustic energy tends to display discontinuous jumps at the parameter values where

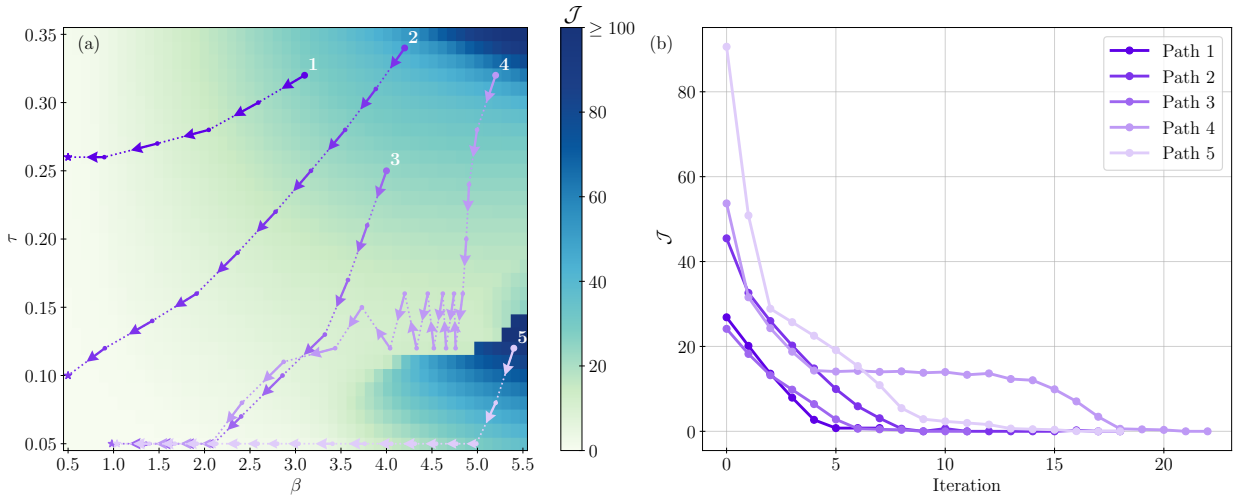


Figure 5.12: Gradient-based optimisation of the time-averaged acoustic energy starting from different initial (β, τ) using adjoint sensitivities computed by a single instance of the T-ESN. In (a), example optimisation paths starting from different initial (β, τ) are plotted on top of the landscape of the objective functional $\mathcal{J}$. The values $\mathcal{J}$ takes during the iterations of optimisation are plotted in (b).

the system bifurcates (Huhn and Magri, 2020), which can lead to undefined gradients between different nonlinear regimes. Second, while the adjoint sensitivity can be computed for limit-cycle and quasiperiodic regimes for arbitrary integration times, for chaotic regimes the integration time is limited before the adjoint system becomes unstable, and the sensitivities diverge (Lea et al., 2000; Huhn and Magri, 2020). Another consideration is the respective orders of magnitudes of the parameters and the sensitivities to them. From Figs. 5.10 and 5.11, β and τ as well as $d\mathcal{J}/d\beta$ and $d\mathcal{J}/d\tau$ have different orders of magnitudes, i.e., $\beta \sim O(1)$, $\tau \sim O(10^{-1})$. The difference in the orders of magnitudes requires the parameters and their respective gradients to be normalised, such that $\breve{\beta} = \beta/\bar{\beta}$, $\breve{\tau} = \tau/\bar{\tau}$, $d\mathcal{J}/d\breve{\beta} = \bar{\beta} d\mathcal{J}/d\beta$, $d\mathcal{J}/d\breve{\tau} = \bar{\tau} d\mathcal{J}/d\tau$, where $\bar{(\cdot)}$ denotes the mean of the parameter in the domain considered (Huhn and Magri, 2020). The optimisation is performed with the steepest-descent method using the normalised gradients with a step size of 0.2. (Since τ can only take discrete values, we round it according to the time-step size of the ESN before the next step of the optimisation.) The optimisation stops when (i) the acoustic energy is near zero, $\mathcal{J}_i < 10^{-4}$, or (ii) the relative change in acoustic energy is below a threshold, $|\mathcal{J}_{i-1} - \mathcal{J}_i|/|\mathcal{J}_{i-1}| < 10^{-4}$, where subscript i denotes the i^{th} iteration step. The optimisation algorithm using the direct-adjoint looping with the trained PA-ESN is provided in Alg. 4. Iterations of optimisation starting from different initial (β, τ) are shown in Fig. 5.12. The optimal (β, τ) values are found to be in the fixed point regimes, i.e., when the instability is suppressed. For the Rijke tube, the fixed point is the zero vector (mean flow), and as such the acoustic energy, i.e., the acoustic energy of the fluctuations from the mean flow, is equal to zero at these values. This result shows the significance of the ability of the ESN to infer fixed points. The process of optimisation is illustrated for Path 3 in Fig 5.13. The optimisation follows a path where the amplitude of the oscillations and so the acoustic energy decreases, for which the ESN accurately predicts the changing dynamics. Similar behaviour is observed for other paths. An exceptional case is Path 4, which passes through a saddle in the landscape of the objective function near $(\beta = 4.5, \tau = 0.15)$. The change in the sign of the gradient, as shown in Fig. 5.11 (d) in the sensitivity plot for $\beta = 4.5$ around $\tau = 0.13$, causes the

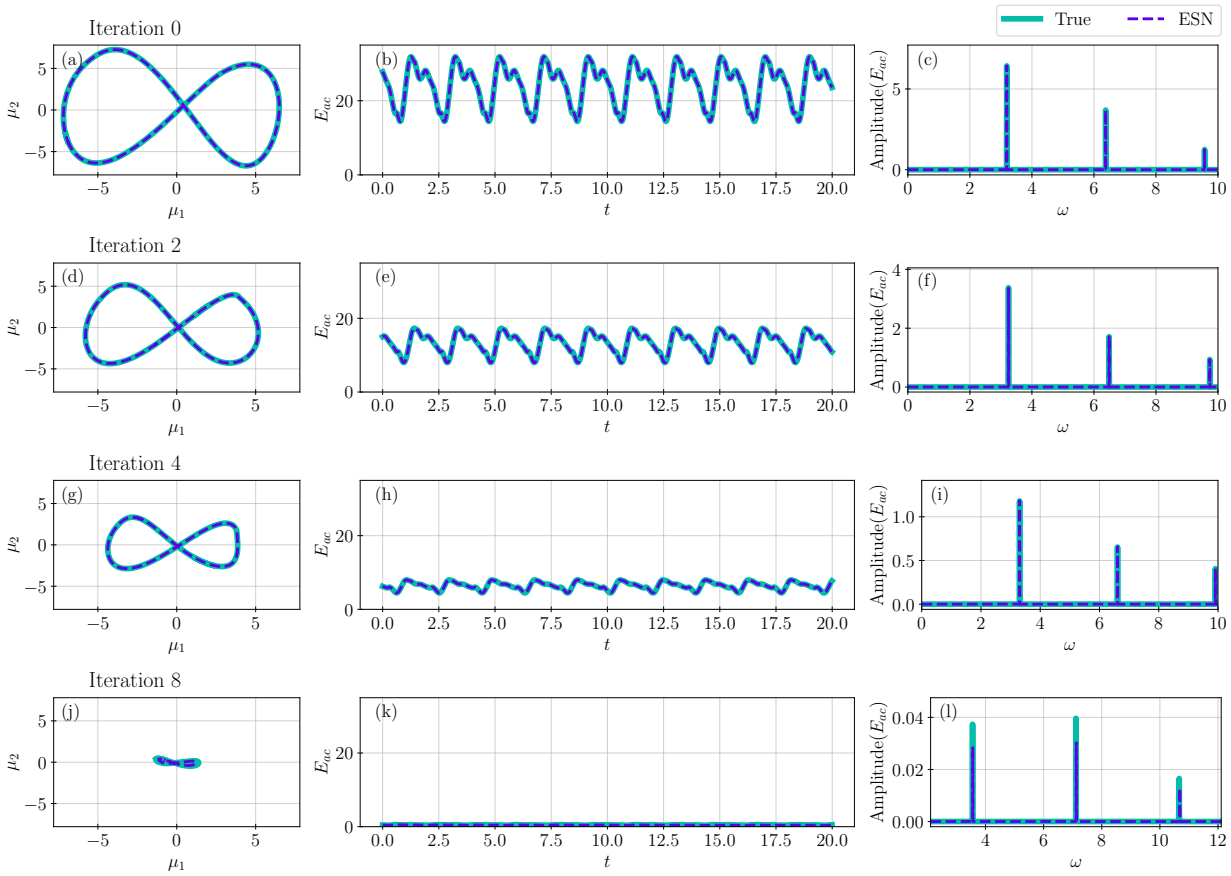


Figure 5.13: Gradient-based optimisation of time-averaged acoustic energy marked as Path 3 in Fig. 5.12. Shown (a – c) iteration 0; ($\beta = 4.0, \tau = 0.25$), (d – f) iteration 2; ($\beta = 3.57, \tau = 0.17$), (g – i) iteration 4; ($\beta = 2.85, \tau = 0.1$), and (j – l) iteration 8; ($\beta = 1.64, \tau = 0.07$). From left to right; the first two pressure modes in the phase space, short-term time series prediction of the acoustic energy, and long-term amplitude spectrum of the acoustic energy. The optimisation follows a path along which the amplitude of the oscillations and the acoustic energy decreases, while the thermoacoustic echo state network accurately predicts the changing dynamics.

"zigzagging" path in this region.

5.8 Robustness to noise

If the data is acquired from real-life experiments, the measurements may be corrupted by noise. This is the case when the system equations are unknown and we wish to learn sensitivities from experimental data. To test the performance of the T-ESN under such a scenario, we consider zero-mean additive Gaussian noise with a standard deviation that is equal to $\sigma_o = \{2\%, 5\%, 10\%\}$ of the standard deviation of the true data (these noise levels correspond to 33, 26, and 20 dB signal-to-noise ratios, respectively). We repeat the training on the noisy data using the hyperparameters chosen before (Tab. 5.5) and investigate the effect of regularisation. The regularisation parameter, λ can be determined by evaluating the validation error during hyperparameter selection. Fig. 5.14 shows the statistics (mean and one standard deviation) of the validation error of the T-ESN for different levels of noise. The validation error is given by the mean relative ℓ_2 error (5.8) as described in § 5.6.1. (In the noisy case, we increase the number of evaluated trajectories to 20 as this helps with the covergence of the error statistics.) To obtain the error statistics, we repeat this calculation for

Algorithm 4 Gradient-based parameter optimisation of the Rijke tube with direct-adjoint looping of the trained T-ESN

Set Step size $\alpha = 0.2$, thresholds $\epsilon_1 = 10^{-4}$, $\epsilon_2 = 10^{-4}$
Initialise $\beta \in [\beta_{\min}, \beta_{\max}]$, $\tau \in [\tau_{\min}, \tau_{\max}]$, $J_{\text{prev}} \leftarrow 10^5$
Get mean $\bar{\beta} \leftarrow (\beta_{\max} + \beta_{\min})/2, \bar{\tau} \leftarrow (\tau_{\max} + \tau_{\min})/2$
while not converged **do**
    $(\mathcal{J}, d\mathcal{J}/d\beta, d\mathcal{J}/d\tau) \leftarrow$ Alg. 3 ▷ Direct-adjoint looping
    if $\mathcal{J} < \epsilon_1$ **or** $|\mathcal{J}_{\text{prev}} - \mathcal{J}|/|\mathcal{J}_{\text{prev}}| < \epsilon_2$ **then** ▷ Check convergence
        converged $\leftarrow$ **True**
    $\mathcal{J}_{\text{prev}} \leftarrow \mathcal{J}$
    Normalise $\breve{\beta} \leftarrow \beta/\bar{\beta}$, $\breve{\tau} \leftarrow \tau/\bar{\tau}$
    Normalise $d\mathcal{J}/d\breve{\beta} \leftarrow \bar{\beta} \cdot d\mathcal{J}/d\beta$, $d\mathcal{J}/d\breve{\tau} \leftarrow \bar{\tau} \cdot d\mathcal{J}/d\tau$
    Update with steepest descent:
        $\breve{\beta} \leftarrow \breve{\beta} - \alpha \cdot d\mathcal{J}/d\breve{\beta}$
        $\breve{\tau} \leftarrow \breve{\tau} - \alpha \cdot d\mathcal{J}/d\breve{\tau}$
    Denormalise $\beta \leftarrow \breve{\beta} \cdot \bar{\beta}, \tau \leftarrow \breve{\tau} \cdot \bar{\tau}$
    Round $\tau \leftarrow \text{int}(\tau/\Delta t) \cdot \Delta t$ ▷ Time step precision
    Check the bounds $\beta \leftarrow \min(\max(\beta, \beta_{\min}), \beta_{\max})$, $\tau \leftarrow \min(\max(\tau, \tau_{\min}), \tau_{\max})$
$\beta^* \leftarrow \beta$
$\tau^* \leftarrow \tau$

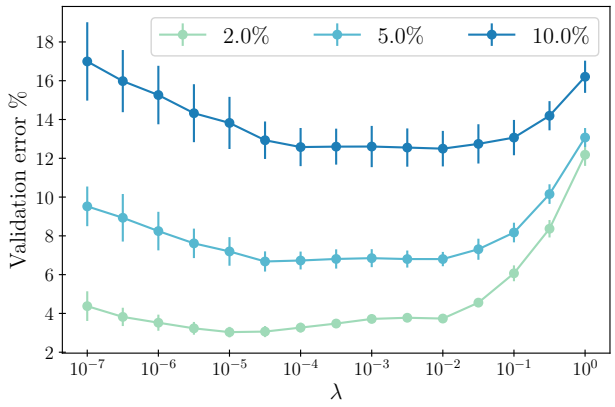


Figure 5.14: Selection of the regularisation parameter λ when the data is noisy. Shown is the validation error of T-ESN on data with different levels of noise (indicated in the legend) when the training is performed with increasing regularisation. The error statistics for 20 different realisations of noise are plotted.

20 different realisations of noise applied to the training and validation data. Because the validation data contains noise, the irreducible error is the noise level. We find that there is a range of λ values that result in a similarly low validation error, and this optimal range of regularisation increases with increasing levels of noise.

For the $\sigma_o = 5\%$ noise level case, we show the performance of closed-loop prediction on a parameter regime ($\beta = 2.0, \tau = 0.25$), which is in the training set, in Fig. 5.15. The T-ESN successfully

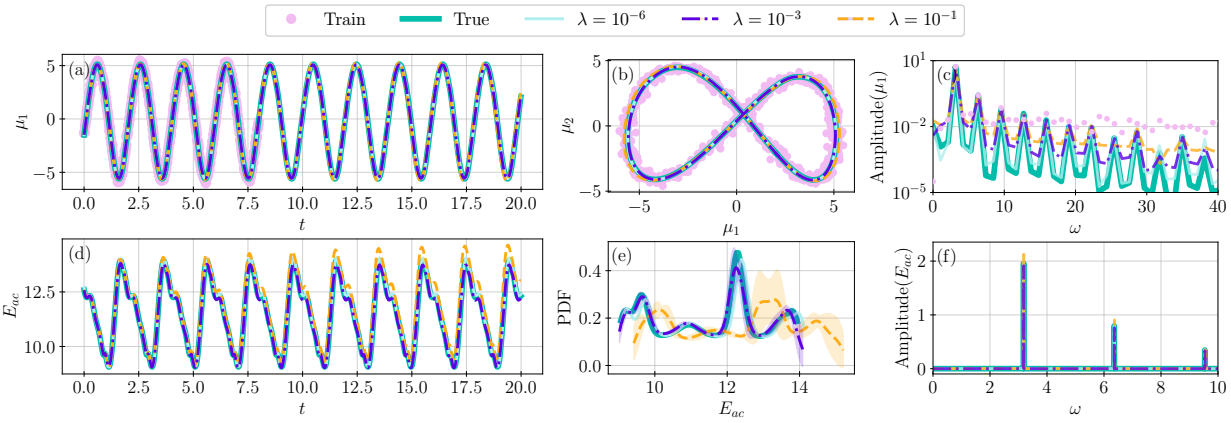


Figure 5.15: T-ESN learns the true limit-cycle attractors of the original system from noisy data. Short-term closed-loop prediction on the training data from ($\beta = 2.0, \tau = 0.25$) regime is shown for the first pressure mode as time series (a) and for the first and second pressure modes in the phase space (b). The amplitude spectrum (c) is computed for the same time window as the training data (8 time units) for consistency in frequency resolution and shown in logarithmic scale. Short-term prediction (d), probability distribution function (PDF) (e), and amplitude spectrum (f) of the acoustic energy are shown as in Fig. 5.9.

learns the limit-cycle attractor from noisy data and does not become unstable when the training data is noisy. We compute the adjoint sensitivities for the same test cases in §5.6.3, these are shown in Figs. 5.16 and 5.17 for the heat release strength β and time delay τ, respectively. The sensitivity computations are affected by the noisy training especially in small τ regimes ($\tau \leq 0.1$) that are unseen during training, e.g., Fig. 5.16 (a, b). These errors are alleviated by increasing the regularisation. However, using a very large regularisation might decrease the performance, for example in higher τ regimes ($\tau \geq 0.25$) as shown in Figs. 5.15 and 5.17 (b).

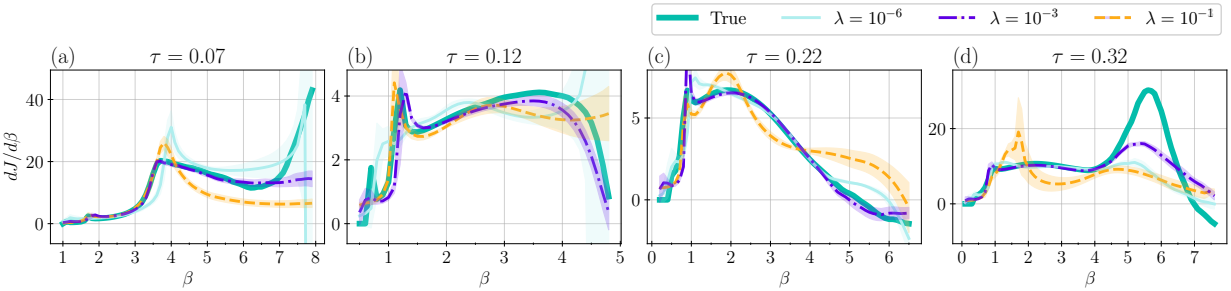


Figure 5.16: Regularisation with an optimal Tikhonov coefficient $\lambda = 10^{-3}$ improves inference of sensitivity to heat release strength in case of noise level $\sigma_o = 5\%$ (See Fig. 5.14). Shown $d\mathcal{J}/d\beta$ at fixed (a) $\tau = 0.07$, (b) $\tau = 0.12$, (c) $\tau = 0.22$, (d) $\tau = 0.32$. The mean of the sensitivity of an ensemble of 5 realisations of the ESN is shown for different regularisation coefficients, one standard deviation from the mean is plotted in lighter colour.

5.9 Learning bifurcations of thermoacoustics

In §. 5.6, we observe that the ESN predicts bifurcations from limit-cycle oscillations to fixed points, i.e., Hopf bifurcation, when the heat release strength β is decreased. This is achieved even though the ESN was trained on only limit-cycle data from $\beta = \{1.0, 2.0, 3.0, 4.0, 5.0\}$ and

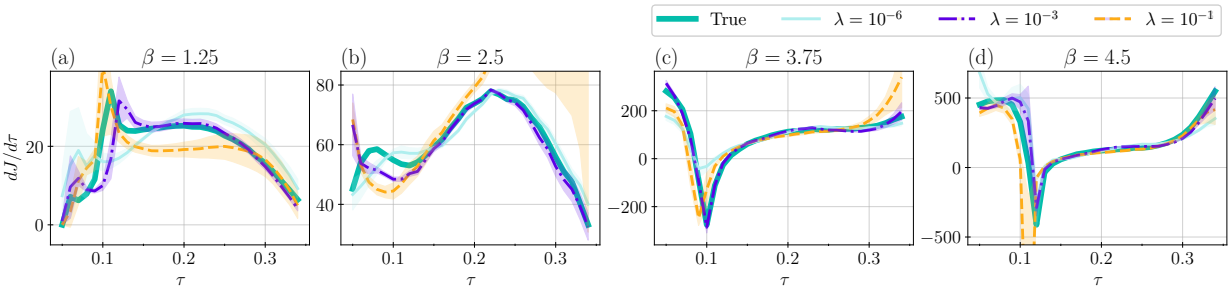


Figure 5.17: Regularisation with an optimal Tikhonov coefficient $\lambda = 10^{-3}$ improves inference of sensitivity to time delay τ in case of noise level $\sigma_o = 5\%$ (See Fig. 5.14). Shown $d\mathcal{J}/d\tau$ at fixed (a) $\beta = 1.25$, (b) $\beta = 2.5$, (c) $\beta = 3.75$, (d) $\beta = 4.5$. The mean of the sensitivity of an ensemble of 5 realisations of the ESN is shown for different regularisation coefficients, one standard deviation from the mean is plotted in lighter colour.

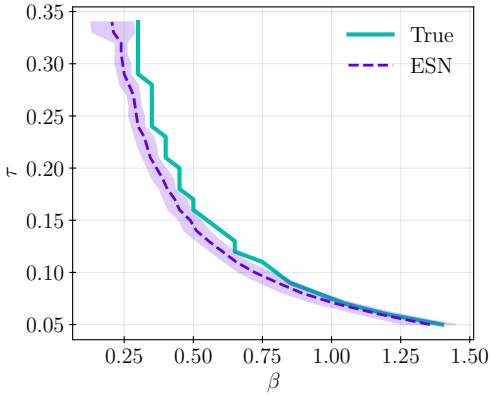


Figure 5.18: Prediction of the Hopf bifurcation point, i.e., when the system transition from fixed point to limit-cycle attractor, with T-ESN trained on only limit-cycle data, where $\beta_{\text{train}} = [1, 5]$ and $\tau_{\text{train}} = [0.1, 0.3]$. Shown the mean and standard deviation in lighter colour of the predicted parameter values over 9 realisations of the ESN. Given a τ value, the ESN infers the β value at which a Hopf bifurcation occurs with a mean absolute error of 0.067 ± 0.046.

$\tau = \{0.1, 0.15, 0.2, 0.25, 0.3\}$ regimes. In this section, we further investigate this condition. Given a τ value, we determine the approximate β value at which a Hopf bifurcation occurs. We run both the original system and the ESN prediction with a fine grid of β values increasing in 0.05 increments. We select the Hopf bifurcation point using an energy criterion as the last point where the acoustic energy monotonously decreases (the system is stable and converges to the zero vector fixed point). We initialise the system with the initial condition of zeros except $\eta_1(t_0) = 1.5$. Due to the subcritical Hopf bifurcation and triggering, the transition to the limit-cycle regime is impacted by the initial condition. The prediction of the Hopf bifurcation point is shown in Fig. 5.18 over 9 realisations of the ESN. (We remove 1 realisation out of 10 due to spurious results.) The parameter value at which the bifurcation occurs has a small error compared to the true system. For example, at fixed $\tau = 0.22$ the Hopf bifurcation occurs at $\beta = 0.40$, whereas 7 out of 9 realisations of the T-ESN predict it as $\beta = 0.35$, and the remaining 2 realisations predicts it as $\beta = 0.25$ and $\beta = 0.30$. Over all the considered τ points ($\tau = \{0.05, 0.06, \ldots, 0.034\}$) and realisations, the mean absolute error of the predictions amounts to 0.067 ± 0.046.

On the other hand, if β is increased, the thermoacoustic system displays bifurcations from limit-

Table 5.6: Summary of the analysed dynamical regimes of the Rijke tube.

Regime type	(β, τ)	Figure
Limit cycle (training)	(2.0, 0.25)	Fig. 5.9 (a – c)
Limit cycle (unseen)	(4.5, 0.12)	Fig. 5.9 (d – f)
Period-doubling	(7.5, 0.3)	Fig. 5.19 (a – c)
Quasi-periodic	(6.1, 0.2)	Fig. 5.19 (d – f)
Chaotic I	(7.6, 0.22)	Fig. 5.20 (a – c)
Chaotic II	(8.7, 0.23)	Fig. 5.20 (d – f)

cycle oscillations to chaos via period-doubling and quasiperiodic routes (Huhn and Magri, 2020). In this section, we investigate whether the ESN can also predict these bifurcations and attractors. We train and validate a separate T-ESN on data generated by larger values $\beta = \{6.0, 6.5, 7.0, 7.5, 8.0\}$ (for brevity, we refer to this dataset as $\beta_{\text{train}} = [6, 8]$). The reservoir size and connectivity are 1200 and 20, respectively as before. The optimal hyperparameters found for this case are provided in Table 5.7 along with the hyperparameters for the previous case when the training was performed with data from $\beta = \{1.0, 2.0, 3.0, 4.0, 5.0\}$ (for brevity, we refer to this dataset as $\beta_{\text{train}} = [1, 5]$). The faster changing dynamics, e.g., chaotic behaviour, in the larger β regimes might be responsible for the increase in the spectral radius and leak rate.

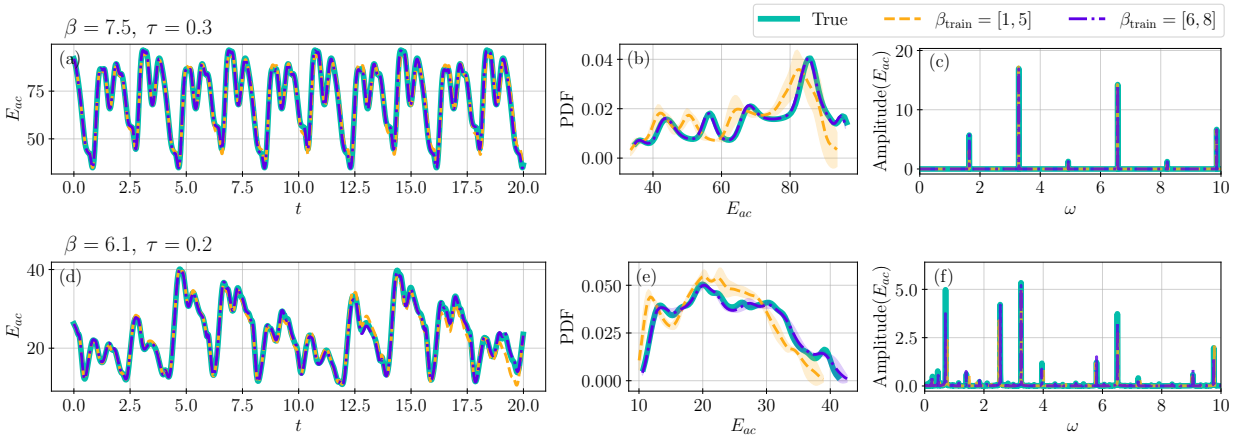


Figure 5.19: Thermoacoustic echo state network captures bifurcations to period-doubling and quasi-periodic regimes. Short-term prediction, statistics, and amplitude spectrum of (a – c) an unseen limit-cycle regime where period-doubling occurs ($\beta = 7.5,\ \tau = 0.3$), (d – f) an unseen quasi-periodic regime ($\beta = 6.1,\ \tau = 0.2$)

We examine the T-ESNs' prediction performance across different nonlinear regimes (Tab. 5.6) First, in Fig. 5.19 (a – c) we show a limit-cycle regime where period-doubling occurs ($\beta = 7.5,\ \tau = 0.3$), and in Fig. 5.19 (d – f) a quasi-periodic regime ($\beta = 6.1,\ \tau = 0.2$). Second, we show two chaotic thermoacoustic regimes with different Lyapunov times; in Fig. 5.20 (a – c) a regime ($\beta = 7.6,\ \tau = 0.22$) with a Lyapunov time of 8.5 time units and in Fig. 5.20 (d – f) a regime ($\beta = 8.7,\ \tau = 0.23$) with a Lyapunov time of 3.9 time units. The ESN trained on $\beta_{\text{train}} = [1, 5]$ can qualitatively predict the bifurcations and capture the statistics and dominant frequencies correctly, especially for the non-chaotic regimes. Training the T-ESN on $\beta_{\text{train}} = [6, 8]$, which encompasses the regimes that we are testing, increases accuracy of both short- and long-term predictions.

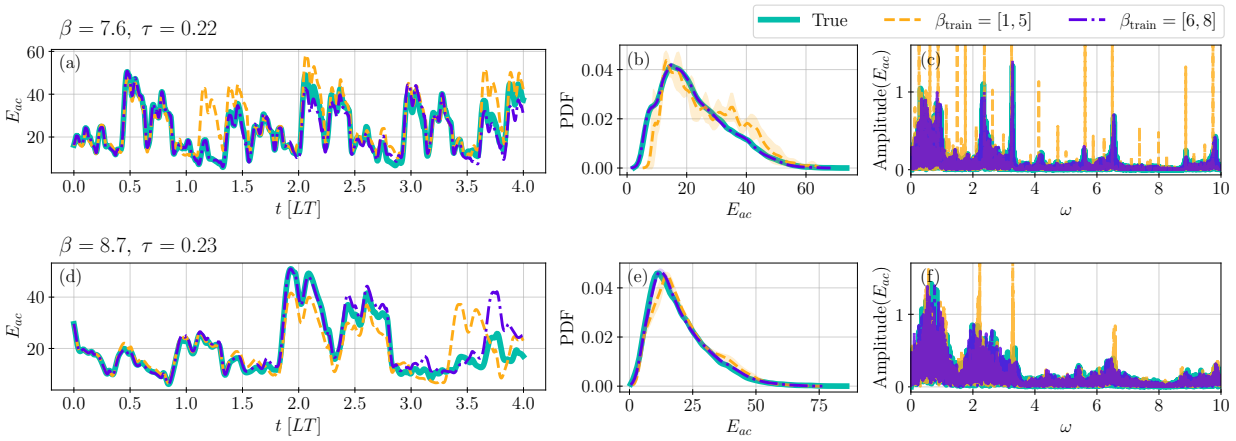


Figure 5.20: T-ESN captures bifurcations to chaotic regimes. Prediction, statistics, and amplitude spectrum of (a – c) an unseen chaotic regime ($\beta = 7.6,\ \tau = 0.22$) with a Lyapunov time of 8.5, and (d – f) an unseen chaotic regime ($\beta = 8.7,\ \tau = 0.23$) with a Lyapunov time of 3.9.

The ability of the T-ESNs trained on different β regimes to predict bifurcations is further shown in Fig. 5.21 in the bifurcation diagrams for fixed $\tau = 0.22$ (a – c) and $\beta = 7.6$ (d – f) cases. This ability of the parameter-aware ESN to predict bifurcations to quasiperiodic or chaotic regimes at unseen parameter values is in itself not an obvious result as the network only had access to limit-cycle regimes at lower β values for training and validation. When we increase β and a bifurcation occurs, the acoustic energy increases with a discontinuous jump, which indicates that the amplitude of the oscillations abruptly change. Therefore, the prediction of these regimes becomes an extrapolation problem because the range of data we are interested in is different from the range of data the network was trained on. In this case, incorporating physical knowledge via the T-ESN architecture enables extrapolation performance. The bifurcation diagrams confirm that in the higher β regimes, the T-ESN trained on $\beta_{\text{train}} = [1, 5]$ can qualitatively capture the bifurcations to quasiperiodic and chaotic regimes, however is not as quantitatively accurate as the T-ESN trained on $\beta_{\text{train}} = [6, 8]$. (The reverse is true for the lower β regimes.) When β is further increased beyond $\beta = 10.0$, the quantitatively accurate performance of the T-ESN trained on $\beta_{\text{train}} = [6, 8]$ also starts diminishing.

Table 5.7: Optimal hyperparameters of the thermoacoustic ESN (5.27) trained on data from $\beta_{\text{train}} = [1, 5]$, i.e., $\beta = \{1.0, 2.0, 3.0, 4.0, 5.0\}$ and $\beta_{\text{train}} = [6, 8]$, i.e., $\beta = \{6.0, 6.5, 7.0, 7.5, 8.0\}$ regimes of the Rijke tube. The time delay values are kept the same across the two training sets, $\tau = \{0.1, 0.15, 0.2, 0.25, 0.3\}$. Refer to (5.27) for the hyperparameters.

Symbol	Description	Range	ESN $\beta_{\text{train}} = [1, 5]$	ESN $\beta_{\text{train}} = [6, 8]$
ρ_h	Spectral radius	0.01–1.0	1.25×10^{-2}	1.60×10^{-1}
ξ_{in}	Input scaling	0.01–2.0	7.90×10^{-2}	1.22×10^{-2}
ξ_f	u_f scaling	0.01–2.0	1.05	1.30
ξ_β	β scaling	0.01–2.0	1.09×10^{-1}	3.80×10^{-1}
g_β	β offset	-10.0–10.0	-10.0	-4.01
α_h	Leak rate	0.01–1.0	3.63×10^{-2}	6.37×10^{-1}
λ	Tikhonov coefficient	10^{-6}–10^{-1}	10^{-6}	10^{-6}

In chaotic systems, due to the positive Lyapunov exponent, the adjoint system is unstable and

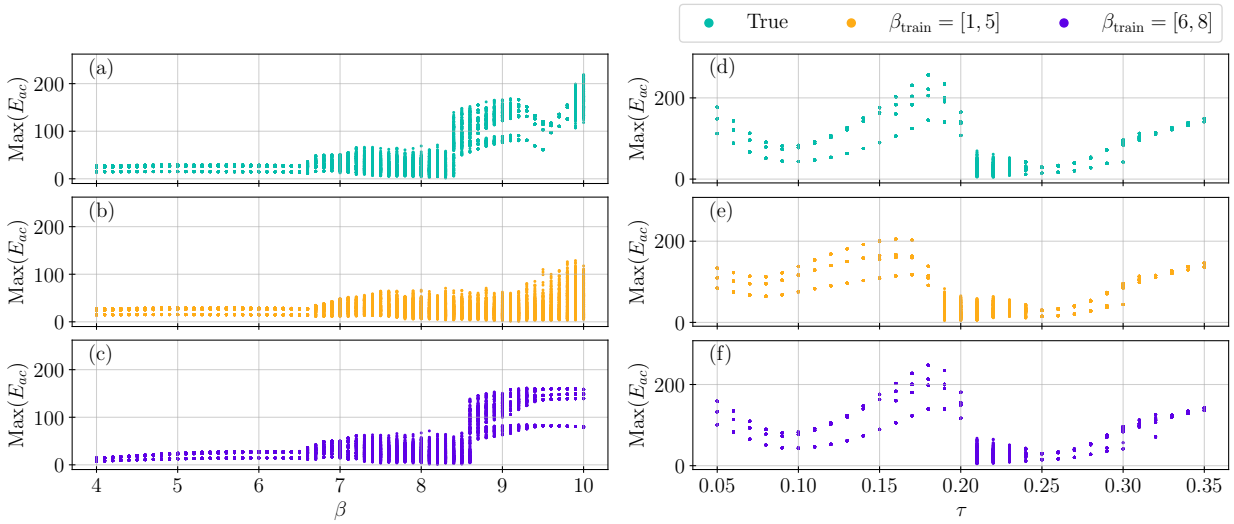


Figure 5.21: Bifurcation diagrams obtained from the true system (top row), T-ESN trained on $\beta_{\text{train}} = [1, 5]$ (middle row), and trained on $\beta_{\text{train}} = [6, 8]$ (bottom row). The peaks of the acoustic energy are plotted. Left half (a – c) shows varying β for fixed $\tau = 0.22$, and right half (d – f) shows varying τ for fixed $\beta = 7.6$.

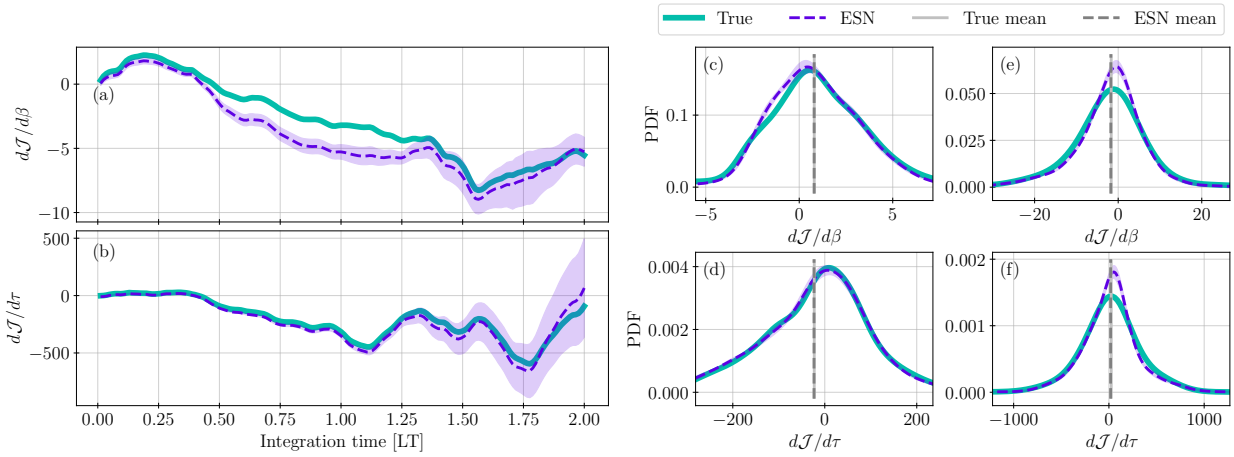


Figure 5.22: Computation of chaotic adjoint sensitivities using T-ESN. The adjoint sensitivities for a single trajectory at ($\beta = 7.6$, $\tau = 0.22$) is shown on the left (a, b). The probability distribution of the sensitivity is computed from 2000 short trajectories with integration time of 0.5 Lyapunov Time (c, d) and 1.0 Lyapunov Time (e, f).

the adjoint sensitivities diverge (Lea et al., 2000; Huhn and Magri, 2020). The ESN learns and replicates the underlying chaotic dynamics of the system in question (e.g., Margazoglou and Magri, 2023b). Because the ESN is a dynamical system (5.11), the adjoint sensitivity in chaotic regimes is unstable in the long-term (Huhn and Magri, 2020). This means that we can only integrate adjoint equations for short time windows. Using the T-ESN trained on $\beta_{\text{train}} = [6, 8]$, we compute the adjoint sensitivities of short-time trajectories for the ($\beta = 7.6$, $\tau = 0.22$) regime (shown in Fig. 5.20 (a – c)). Figs. 5.22 (a, b) show the evolution of the adjoint sensitivities with integration time for one trajectory for an ensemble of 5 T-ESNs. For short integration times, the T-ESN predicts the chaotic sensitivities close to the original system, but diverges as the integration time increases due to the chaotic nature of the system. One approach to estimate the sensitivity of chaotic systems is by taking the mean of an ensemble trajectories (Lea et al., 2000) as applied to

the Lorenz system in § 5.4. In Figs. 5.22 (c, d) and (e, f), we show the probability distributions and the mean of the adjoint sensitivities for 2000 trajectories for integration times of 0.5 LT and 1 LT, respectively. In order to obtain the PDFs, we let the T-ESN run autonomously without any washout data from the original system. For an integration time of 0.5 LT, we can recover the PDF as well as the mean, while for 1.0 LT, low statistical moments are matched.

5.10 Inferring sensitivities to initial conditions

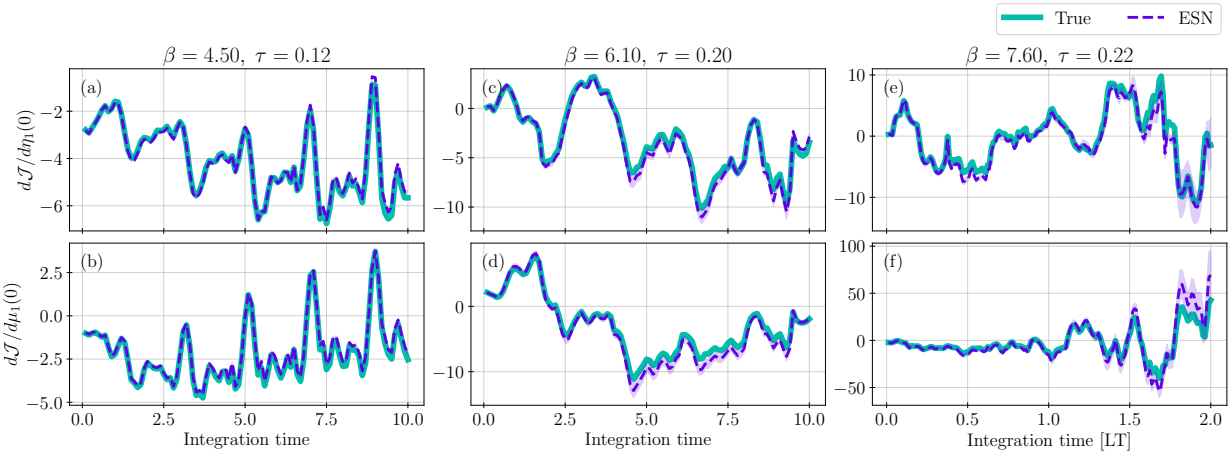


Figure 5.23: Computation of adjoint sensitivities to initial conditions using T-ESN. The adjoint sensitivity for a single trajectory at unseen (a, b) limit-cycle ($\beta = 4.5,\ \tau = 0.12$), (c, d) quasiperiodic ($\beta = 6.1,\ \tau = 0.2$), (e, f) chaotic ($\beta = 7.6,\ \tau = 0.22$) regimes. The sensitivity of the acoustic energy of the final flow state to perturbation to the first velocity mode η_1 (top row) and to the first pressure mode μ_1 (bottow row) are shown for increasing integration times.

Table 5.8: Angle between the sensitivity gradients of the original system to initial conditions and the estimates from T-ESN. The mean angle over 20 independent trajectories and 5 realisations of T-ESN is given in degrees.

Integration Time	Limit cycle ($\beta = 4.5,\ \tau = 0.12$)	Quasiperiodic ($\beta = 6.1,\ \tau = 0.2$)
2.0	2.2	6.8
4.0	2.3	12.8
10.0	3.2	16.2

Integration Time (LT)	Chaotic ($\beta = 7.6,\ \tau = 0.22$)
0.5	8.0
1.0	15.5
2.0	34.3

The adjoint method allows us to efficiently compute sensitivities to initial conditions. In this section, we infer the sensitivity to initial conditions, i.e., $\boldsymbol{y}(0) = [\boldsymbol{\eta}(0);\ \boldsymbol{\mu}(0)]$, using the T-ESNs trained in the previous sections. The objective functional is the acoustic energy of the final flow state. The sensitivities to perturbations to the first velocity and pressure modes, η_1 and μ_1, are shown in Fig. 5.23 for a trajectory from exemplary limit-cycle ($\beta = 4.5,\ \tau = 0.12$), quasiperiodic

($\beta = 6.1,\ \tau = 0.2$), and chaotic ($\beta = 7.6,\ \tau = 0.22$) regimes. To assess the accuracy of the estimates, we repeat this procedure for multiple independent trajectories. Because in gradient-based optimisation we are interested in the gradient direction, we compute the angle between the sensitivity gradients of the original system and the T-ESN estimates as an indicator of the accuracy. The mean of the computed angles over 20 trajectories and 5 realisations of T-ESN are provided in Tab. 5.8 for different integration times. We find that T-ESN achieves high accuracy for all integration times in the limit-cycle regimes and for short integration times in quasiperiodic and chaotic regimes.

5.11 Discussion

In this section, we offer a comparison of the proposed ESN-based adjoint solver with traditional, i.e., physics-based, adjoint solvers for gradient-based optimisation. With the physics-based approach, we need (i) governing equations (plus boundary conditions and initial conditions), (ii) a numerical scheme to solve them, (iii) the linearisation (tangent model), and (iv) its dual with respect to a chosen inner product (adjoint code). We need to derive/modify the adjoint code any time that any of the above is modified. On the other hand, in the ESN-based approach the adjoint code is derived once and does not change, i.e., the adjoint equations of the ESN described by Eqs. (5.14) and (5.20) are independent of the data and system under investigation. Similarly, the adjoint equations of the time-delayed ESN described by Eq. 5.30 are applicable to any system with a constant time-delay. The ESN-based approach is also particularly attractive when we have experimental data, in which the equations might not be known (or partly known) and the data contains noise, because it learns the dynamics from observables.

As a data-driven approach, the ESN-based method relies on (i) having sufficient data that covers the solution space of the problem, and (ii) training and validation of the ESN. The parameter-aware extension of the ESN addresses the more challenging task of learning the dynamics in different parameter regimes, across which the dynamics are not statistically stationary. When utilising a parameter-aware ESN for the computation of parameter sensitivities, as the number of system's parameters increase, more training regimes are required. This is in contrast with physics-based adjoint solvers, which are derived from the system's equations that generalise over the parameter space. Thus, if the ESN-based adjoint solver is sought to replace a physics-based adjoint solver, the advantage can vary from case to case depending on the cost of data acquisition and simulation time. Because the network learns a continuous parameterisation of the dynamics even if it is trained on data from a discrete set of parameters, the adjoint can be computed around any set of parameters. With a continuous parameterisation of the objective functional, a future research direction is to combine Bayesian optimisation (e.g., Huhn and Magri, 2022), which relies on sampling via the ESN, and adjoint-based ESN gradients. The physics-based adjoint solver depends on the modelling assumptions, whereas the accuracy of the ESN-based adjoint solver depends on data and the hyperparameters. In high-dimensional systems, the ESN-based adjoint solver can be combined with a data-driven model reduction techniques, e.g., autoencoding as in Racca et al. (2023).

Finally, regarding computational costs, the training of the ESN is faster compared to other neural networks that are trained with stochastic gradient descent and backpropagation, and the T-ESN

runs faster than the original solver when simulating the thermoacoustic system considered in this work. While this can be case dependent, in general the propagation of the ESN equations is fast thanks to the sparse weight matrices even though the reservoir state can be high dimensional (here, 1200). An optimisation of the code for matrix multiplications on a GPU, e.g., using JAX (Bradbury et al., 2018; Kochkov et al., 2021), is scope for future work. The computation times for the direct solutions, adjoint solutions and the training of the PA-ESN are provided in the Appendix § E.2.

Using ensemble variational methods, it is possible to circumvent the Jacobian computation in the adjoint method by approximating it with a Monte Carlo sampling of a state ensemble. The parameter-aware ESN can also be integrated into these methods. Similarly, in the next chapter (Chapter 6), we employ the ESN as a data-driven model in an ensemble Kalman filter, in which the optimisation objective is the minimisation of the state estimation error in a chaotic system from partial observations of the state. Given that the forward-in-time propagation of the trajectories with the ESN is very fast, with parallelisation (§ 6.2.4), ensemble methods can indeed be computationally more efficient than the adjoint method in scenarios with a small number of parameters. The key advantage of the adjoint method is that it provides the exact gradients with a computational cost that is independent of the number of parameters.

5.12 Conclusion

Adjoint methods are established tools for gradient-based optimisation. Although adjoint methods offer an accurate and efficient gradient computation, they are problem specific. In this chapter, we propose a strategy to infer adjoint sensitivities from data (observables). We showcase the method on (i) a prototypical low-dimensional chaotic system (Lorenz), and (ii) a prototypical nonlinear time-delayed thermoacoustic system (the Rijke tube).

First, we propose a parameter-aware echo state network (PA-ESN). This network learns the parameterised dynamics, i.e., it infers how the dynamics change with varying parameters, thereby making predictions for parameters that are not used during training. Second, we derive the adjoint parameter-aware echo state network, which is used to compute the adjoint sensitivities of the physical system. The proposed framework is general and can be applied to prototypical low-dimensional dynamical systems, e.g., the Lorenz system, without any modification of the standard architecture. For the Lorenz system, we compute the sensitivity of a time-averaged state variable, $\bar{z}_\mathrm{L}$, at varying parameter values to all parameters, σ_L, ρ_L, and β_L. We find that the estimated sensitivities match closely with the sensitivities obtained by the adjoint of the original system. We observe a known bias (Chandramoorthy et al., 2017) between the ensemble adjoint sensitivities and direct estimates calculated by fitting a polynomial to the objective function.

For the thermoacoustic system of interest we find that embedding physical knowledge improves generalizability and diminishes the need for data. Therefore, third, we design a parameter-aware ESN motivated by the physics of thermoacoustics (thermoacoustic ESN, or T-ESN in short) by (i) making the ESN explicitly time-delayed, and (ii) designing the input matrix in consideration of acoustic physics. We extend the derivation of the adjoint to the time-delayed case. We find that the T-ESN accurately predicts the dynamics of different nonlinear thermoacoustic regimes, and successfully infers the adjoint sensitivities. These results are validated against the ground

truth generated from integration of the direct and adjoint equations of the original system. We optimise the acoustic energy with steepest gradient descent, in which the gradients are computed by only one T-ESN. Once the T-ESN has been trained, the algorithm runs autonomously, i.e., we do not run any additional simulations of the original model. As the optimisation proceeds with the iterations, the acoustic energy and the amplitude of the oscillations decrease, finally reaching a fixed-point solution with zero acoustic energy. Fourth, we investigate the robustness to noisy data and chaotic regimes. The T-ESN can qualitatively predict bifurcations to nonlinear regimes beyond the training range of parameter values. Quantitative accuracy increases when the range of training parameters covers the regime in which we are interested. Finally, in addition to parameter sensitivities, we infer sensitivities to initial conditions.

This work opens up possibilities for data-driven gradient-based optimisation without developing code-specific adjoint solvers.

Chapter 6

Data-assimilated model-informed reinforcement learning

The work in this chapter can be found in Ozan et al. (2025a) and Ozan et al. (2025b). This work was a collaboration with Andrea Nóvoa and Georgios Rigas, who contributed to the methodology development (A.N. in data assimilation, and G.R. in reinforcement learning) and reviewing. A.N. additionally contributed to refining the manuscript and supported with visualisations.

The code developed in this work is available as a GitHub repository at MagriLab/DA-RL[1]. The individual experiment runs can be viewed at Weights & Biases for model-free RL (DA-RL-MF), DA-MIRL with the truncated Fourier model (DA-MIRL-Fo), and with the ESN (DA-MIRL-ESN).

6.1 Introduction

Chaos arises in many fields in science and engineering and is characterised by (i) nonlinearity, (ii) aperiodic behaviour, and (iii) extreme sensitivity to small perturbations, making its prediction and control fundamentally challenging (Ott, 2008). One example of spatiotemporally chaotic systems is turbulent flows (Holmes et al., 1996), in which the nonlinearity leads to the transfer of energy across scales. Turbulence bears practical significance for energy and transportation systems in terms of their efficiency and energy consumption. Taming turbulence in these cases can effectively lead to drag reduction of transport vehicles, lift increase of wings, efficiency increase of wind turbines, and mixing enhancement of combustors, for which closed-loop control has been deemed key (Brunton and Noack, 2015). However, predicting these systems involves high-dimensional computationally intensive models, which are not suitable in real-time control applications (e.g., Bewley et al., 2001).

Practically, closed-loop control of turbulent flows relies on sensor measurements of the flow field. In closed-loop control, the actuation signals, or control inputs, are dynamically adjusted based on this feedback from the sensors in order to drive the system towards a desired objective (Åström and Murray, 2008). Optimal control provides a framework for designing control strategies that maximise or minimise certain objectives given some constraints (Bertsekas, 2007), e.g., minimising the energy consumption of a system. Such modern control strategies for nonlinear and time-varying systems

[1] I acknowledge the use of ChatGPT-4.5 for debugging and streamlining this code.

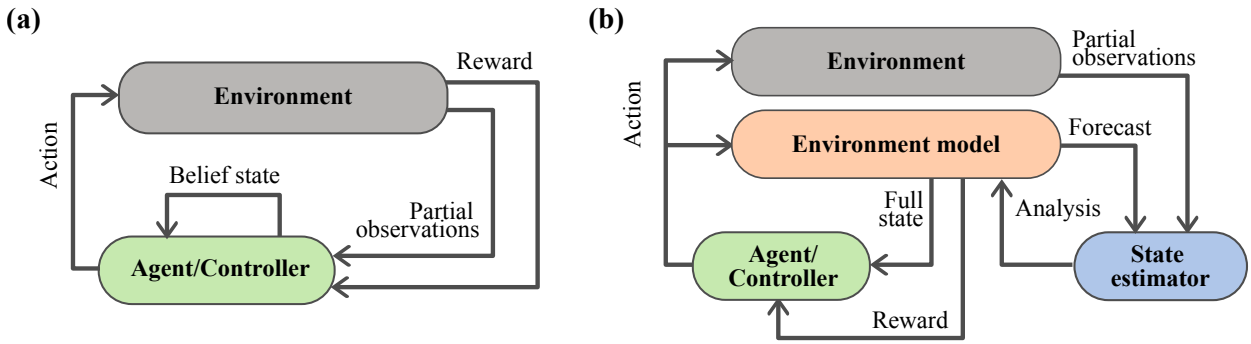


Figure 6.1: Three-block schematic of reinforcement learning methods under partial observability. Comparison between (a) model-free RL with belief state and (b) the proposed data-assimilated model-informed RL.

are most effectively expressed in the state-space formulation, which assumes direct access to the system's state. However, in practice, full state measurements are rarely available and one can only access noisy and partial observations. Thus, we need a method to estimate (i.e., approximate) the full state from the available observations, which can in turn be used as feedback (Luenberger, 1971; Kalman, 1960). This is known as observer-based control. Data assimilation (DA) methods solve the inverse problem of inferring the most likely system state by optimally combining model forecasts and observations, whilst accounting for their uncertainties (Evensen, 2009). Specifically, sequential DA assimilates the observations as they become available, bypassing the need to store and post-process the data, which makes it suitable for real-time state estimation. For linear time-invariant (LTI) systems, a well-established example of observer-based control is Linear-Quadratic-Gaussian (LQG) control, which optimally combines state estimation and control design. LQG consists of two main components (i) the Kalman filter, which provides the minimum-variance estimate of the system state given noisy measurements, and (ii) the Linear-Quadratic Regulator (LQR), which computes the optimal state-feedback control law for a quadratic cost function. Due to the separation principle, in LTI systems the design of the state estimator and the controller can be treated independently. However, LQG in general is not adequate for controlling spatiotemporal chaos as the linear and Gaussian assumptions may break down in high-dimensional chaotic regimes.

Optimisation lies at the core of flow control, where the objective functional can represent aerodynamic performance or energy efficiency, constrained by the governing flow physics. For high-dimensional systems, adjoint methods enable gradient-based optimisation by efficiently computing sensitivities of the objective functional with respect to all control parameters (e.g., Gunzburger, 2002). Adjoint methods are model based, and require the derivation of the Jacobian of the system. Whilst these approaches are powerful, for chaotic systems, due to the exponential divergence of infinitesimally close trajectories, the integration of the adjoint system becomes unstable beyond short time windows (Lea et al., 2000). We have dealt with adjoint methods extensively in Chapter 5. Since control policies based on state feedback are often parameterised, the data-driven framework derived in Chapter 5 for parameter sensitivities is also applicable to optimal control problems. However, in this chapter, we explore a directly data-driven optimal control method for chaotic systems, namely reinforcement learning.

Reinforcement learning (RL) has emerged as a model-free alternative for discovering optimal control

policies directly from data or simulations (Sutton and Barto, 2018) and, therefore, has been applied to a variety of flow control problems (e.g., Rabault et al., 2019; Guastoni et al., 2023; Vignon et al., 2023). In the RL paradigm, an agent learns an optimal policy by interacting with an environment with the goal of maximising cumulative rewards it receives over time (Sutton and Barto, 2018). Within RL, we focus on actor-critic methods, which are a class of RL algorithms suitable for the continuous state and action spaces relevant to our control problem of chaotic systems. Actor-critic methods achieve this by iteratively learning both the optimal value function and the optimal control policy (Sutton and Barto, 2018). Traditionally, RL assumes Markovian dynamics and full state observability, i.e., it is a Markov decision processes (MDPs) (Sutton and Barto, 2018). When only a subset of the state can be measured, the problem becomes a partially observed Markov decision process (POMDP) (Kaelbling et al., 1998). In POMDP problems, model-free RL methods can struggle, often leading to unstable or suboptimal policies (Kaelbling et al., 1998). Keeping the entire observation history in order to address partial observability is infeasible as the dimension of this history grows with each time step. This motivates the search for sufficient lower-dimensional statistics, i.e., the belief state (Bertsekas, 2007), which is a probability distribution over all possible system states, given the history of observations and actions. The belief state provides sufficient statistics to encapsulate the information from the observation and action history needed for optimal decision-making under partial observability (Kaelbling et al., 1998). Model-free RL approach to solving POMDPs is illustrated in Fig. 6.1 (a). In model-free RL methods utilising recurrent neural networks (RNN) to retain the memory efficiently, the belief is implicitly represented by the hidden state of the RNN, which must be trained jointly with the value and the policy. Alternatively, the belief can be captured explicitly in model-based RL methods. Here, a model of the environment is either learned or pre-defined. In contrast to model-driven control, this model is used to generate additional samples of the environment for training, such as the Dyna algorithm (Sutton, 1991), or to plan over finite horizons (similarly to model predictive control), i.e., the agent uses the model to simulate the consequences of candidate action sequences, updating its policy based on the predicted future dynamics (e.g., Racanière et al., 2017; Nagabandi et al., 2018; Hafner et al., 2019).

We propose a framework that transforms the POMDP into a MDP with data assimilation, that is, the data-assimilated model-informed RL (DA-MIRL). The framework, inspired by observer-based control, is depicted in Fig. 6.1 (b), consists of three key components: (i) a predictive model of the system's dynamics, (ii) sequential data assimilation for real-time state estimation, and (iii) reinforcement learning for discovering optimal control policies. This approach divides the problem of end-to-end training of observer and controller in RL into individual modules. We combine the ensemble Kalman filter (EnKF) as the state estimator (Evensen, 2009) with an actor-critic model-free RL method, the deep deterministic policy gradient (DDPG). Recent studies have explored integrating linear Kalman filters with RL for drone navigation (Marino and Guglieri, 2024) and learning a correction for the state estimates of the EnKF in unactuated systems using RL (Hammoud et al., 2024). Our framework focuses on spatio-temporally chaotic systems. The EnKF extends the linear Kalman filter, and employs a low-dimensional ensemble of model forecasts to approximate the state covariance, making it scalable to high-dimensional systems such as spatiotemporal chaos. For the models we consider: (i) the model is physics-based and low-order such that we can run it in real time; and (ii) the model is fully data-driven such that we assume we have no knowledge of the system's equations but have access to full state data from the system excited by random actuta-

tions, i.e., similar to open-loop system identification. For this purpose, we propose a control-aware echo state network (CA-ESN), which is designed here as an efficient architecture that predicts the system's response to control inputs.

This chapter is organised as follows. In § 6.2, we define the problem using the notation from RL literature and introduce the building blocks of the framework. In § 6.3, we introduce the application of DA-MIRL to the Kuramoto-Sivashinsky (KS) equation (§ 3.5), which is a PDE that can display spatiotemporal chaos, as the test case. In § 6.4, we give details on the training procedure and performance metrics of the model-free RL and DA-MIRL. We obtain a baseline of the performance of the model-free RL agent in § 6.5. In § 6.6, we demonstrate the effectiveness of the proposed DA-MIRL framework. § 6.7 concludes the chapter.

6.2 Methodology

We define the time evolution of a controlled system with the discrete-time state space equations

$$\begin{aligned} \boldsymbol{s}(k+1) &= \boldsymbol{F}(\boldsymbol{s}(k), \boldsymbol{a}(k)) + \boldsymbol{\epsilon}_F(k), \\ \boldsymbol{a}(k) &= \boldsymbol{\pi}(\boldsymbol{s}(k)) \end{aligned} \tag{6.1}$$

where $k \in \mathbb{Z}^{\geq 0}$ is the time step, $\boldsymbol{s}(k) \coloneqq \boldsymbol{s}(t_k) \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ is the state at time $t_k = k\Delta t$; $\boldsymbol{a}(k) \coloneqq \boldsymbol{a}(t_k) \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ is the actuation, i.e., the control input applied to the system at time t_k, $\boldsymbol{\epsilon}_F(k) \in \mathbb{R}^{n_s}$ encapsulates the aleatoric and epistemic modelling errors; and the initial condition $\boldsymbol{s}_0$ is drawn from a normal distribution $\sim \mathcal{N}(\bar{\boldsymbol{s}}_0, \boldsymbol{C}^0_{ss})$. This formulation specifically accounts for uncertainties in modelling compared to the definition (2.1). We apply the control as a deterministic state feedback $\boldsymbol{\pi} : \mathcal{S} \to \mathcal{A}$ and the dynamics of the system are encapsulated by the operator $\boldsymbol{F} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, which is Markovian, i.e.,

$$\boldsymbol{s}(k) \sim \mathcal{P}(\boldsymbol{s}(k) \mid \boldsymbol{s}(k-1), \boldsymbol{s}(k-2), \ldots, \boldsymbol{s}(0), \boldsymbol{a}(k-1), \boldsymbol{a}(k-2), \ldots, \boldsymbol{a}(0)) = \mathcal{P}(\boldsymbol{s}(k) \mid \boldsymbol{s}(k-1), \boldsymbol{a}(k-1)) \tag{6.2}$$

where $\mathcal{P}(\ \mid\ )$ is the transition probability distribution, which in (6.2) describes how likely the system is to transition to a particular state, conditioned on the current state and action.

In real-world applications, the full system state is often unobservable. Instead, we receive observations from sensors that provide partial and potentially noisy information about the underlying state

$$\boldsymbol{o}(k) = \boldsymbol{M}(\boldsymbol{s}(k)) + \boldsymbol{\epsilon}(k), \tag{6.3}$$

where $\boldsymbol{o} \in \mathcal{O} \subseteq \mathbb{R}^{n_o}$ is the observation vector, $\boldsymbol{\epsilon} \in \mathbb{R}^{n_o}$ is the additive measurement noise, and $\boldsymbol{M} : \mathcal{S} \to \mathcal{O}$ is the observation operator. We approximate the measurement noise $\boldsymbol{\epsilon}$ as Gaussian noise. Given partial and noisy observations $\boldsymbol{o}(k)$, our goal is to find the optimal control policy $\boldsymbol{\pi}^\star$ that optimises a certain objective functional, e.g., § 6.3.3, constrained by the system's dynamics (6.1).

6.2.1 Reinforcement learning

First, we define the optimal control problem in an RL setting as an MDP under the assumption of full state knowledge. In RL, the optimisation objective is defined through the reward function

$r(\boldsymbol{s}(k), \boldsymbol{a}(k)) : \mathcal{S} \times \mathcal{A} \to \mathcal{R} \subseteq \mathbb{R}$, which assigns a scalar value to each state-action pair, representing the immediate benefit of taking action $\boldsymbol{a}(k)$ in state $\boldsymbol{s}(k)$ at time k. Then, given a state $\boldsymbol{s}(k)$, the agent, i.e., the controller, selects an action $\boldsymbol{a}(k)$ according to a policy $\boldsymbol{\pi}$, aiming to maximise the expected return, which is defined as the cumulative reward.

The value function $V^{\pi} : \mathcal{S} \to \mathbb{R}$ represents the expected cumulative reward starting from state $\boldsymbol{s}(0)$ and following policy $\boldsymbol{\pi}$:

$$V^{\pi}(\boldsymbol{s}(0)) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(\boldsymbol{s}(k), \boldsymbol{\pi}(\boldsymbol{s}(k)))\right], \tag{6.4}$$

where $\gamma \in [0, 1]$ is the discount factor that weights the importance of immediate rewards over future ones. Due to the Markov property (6.2) and using the principle of optimality, the optimisation problem can be solved recursively, leading to the Bellman equation (Bertsekas, 2007)

$$V^{\pi}(\boldsymbol{s}(k)) = \mathbb{E}\left[r(\boldsymbol{s}(k), \boldsymbol{\pi}(\boldsymbol{s}(k))) + \gamma V^{\pi}(\boldsymbol{s}(k+1))\right]. \tag{6.5}$$

In the RL setting where the model dynamics are unknown, it is more practical to introduce the action-value function. The action-value function (Sutton and Barto, 2018), or Q-function, $Q^{\pi} : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$, evaluates the expected return when taking any action $\boldsymbol{a}(0) \in \mathcal{A}$ in state $\boldsymbol{s}(0)$ at time t_0 and subsequently following policy $\boldsymbol{\pi}$

$$Q^{\pi}\left(\boldsymbol{s}(0), \boldsymbol{a}(0)\right) = \mathbb{E}\left[r\left(\boldsymbol{s}(0), \boldsymbol{a}(0)\right) + \sum_{k=1}^{\infty} \gamma^k r\left(\boldsymbol{s}(k), \boldsymbol{\pi}(\boldsymbol{s}(k))\right)\right]. \tag{6.6}$$

The Q-function enables the comparison of the effect of different actions through exploration, and the Q-learning algorithm (Watkins, 1989) estimates the optimal Q-function, $Q^{\star}(\boldsymbol{s}, \boldsymbol{a})$, without requiring knowledge of the system dynamics, by iteratively applying the Bellman optimality update. The Bellman equation for the Q-function can be formulated similarly to Eq. (6.5) (Sutton and Barto, 2018)

$$Q^{\pi}(\boldsymbol{s}(k), \boldsymbol{a}(k)) = \mathbb{E}\big[r(\boldsymbol{s}(k), \boldsymbol{a}(k)) + \gamma Q^{\pi}\left(\boldsymbol{s}(k+1), \boldsymbol{\pi}(\boldsymbol{s}(k+1))\right)\big]. \tag{6.7}$$

The optimal control policy $\boldsymbol{\pi}^{\star}$ maximises the action-value function

$$Q^{\star}(\boldsymbol{s}, \boldsymbol{a}) = \max_{\boldsymbol{\pi}} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}), \text{ for all } \boldsymbol{s} \in \mathcal{S} \text{ and } \boldsymbol{a} \in \mathcal{A}. \tag{6.8}$$

We employ deep deterministic policy gradient (DDPG) algorithm (Lillicrap et al., 2016) to solve Eq. (6.8) owing to its conceptual and implementation simplicity, as well as to its demonstrated effectiveness in controlling the KS equation (e.g., Bucci et al., 2019). DDPG is a model-free RL algorithm designed for environments with continuous state and action spaces. Its architecture consists of two neural networks: (i) an actor network $\boldsymbol{\pi}(\boldsymbol{s}; \boldsymbol{\theta}^{\pi})$, i.e., the policy, which maps the state $\boldsymbol{s}$ to a continuous deterministic action $\boldsymbol{a}$; and (ii) a critic network $Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}^{Q})$, which estimates the action-value function associated with the current policy. The actor and critic are parameterised by their respective weights $\boldsymbol{\theta}^{\pi}$ and $\boldsymbol{\theta}^{Q}$. The actor is trained to choose the optimal action for each state by maximising the action-value function approximated by the critic, i.e., the action-value network, $Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}^{Q})$. The critic is trained by the temporal difference error which is based on the Bellman

equation (6.7) (Sutton and Barto, 2018). The Bellman equation represents an expectation over the state transition dynamics and policy. Since the process model is unknown and solving the equation can be intractable in high-dimensional continuous spaces, this expectation is approximated via Monte Carlo sampling.

Whilst the actions are deterministic, DDPG introduces exploration noise on the deterministic policy to promote exploration. As the agent interacts with the environment, the tuples containing $(\boldsymbol{s}(k), \boldsymbol{a}(k), r(k), \boldsymbol{s}(k+1))$ are stored in an experience replay buffer $\mathcal{B}$, and sampled in batches for training in an off-policy fashion, which enhances sample efficiency via reuse of past experience and reduces the correlation between consecutive updates and avoids overfitting to the last seen data. The DDPG algorithm employs two pairs of actor-critic networks for stability of the training, where the target networks, $\boldsymbol{\pi}'$ and $Q^{\pi\prime}$, are updated with an exponential moving average of the learning networks. The soft update rate is denoted by α_{soft}. The initialisation of the DDPG algorithm is given in Alg. 5. The update of actor-critic networks using a random batch of $(\boldsymbol{s}(k), \boldsymbol{a}(k), r(k), \boldsymbol{s}(k+1)) \in \mathcal{B}$ of batch size N_b is given in Alg. 6.

Algorithm 5 Initialisation of deep deterministic policy gradient (DDPG).

1: **Initialise** critic network $Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}^{Q})$ with random weights $\boldsymbol{\theta}^{Q}$
2: **Initialise** actor network $\boldsymbol{\pi}(s; \boldsymbol{\theta}^{\pi})$ with random weights $\boldsymbol{\theta}^{\pi}$
3: **Initialise** target networks $Q^{\pi\prime}$ and $\boldsymbol{\pi}'$ with weights $\boldsymbol{\theta}^{Q'} \leftarrow \boldsymbol{\theta}^{Q}$, $\boldsymbol{\theta}^{\pi'} \leftarrow \boldsymbol{\theta}^{\pi}$
4: **Initialise** replay buffer $\mathcal{B}$

6.2.2 Partially Observed Markov Decision Processes (POMDPs)

Reinforcement learning methods are typically formulated as MDPs, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ (Sutton and Barto, 2018). A key assumption in this formulation is the full state observability so that the dynamics, i.e., the transitions, are Markovian, and, therefore, optimality defined through the Bellman equation (6.7) holds. However, in reality, the control problem is a POMDP, because only a subset of the full state can be measured, especially for chaotic and turbulent systems. The POMDP problem is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \mathcal{Z})$ including the observation $\mathcal{O}$ and the observations' likelihood $\mathcal{Z}(\boldsymbol{o}(k)|\boldsymbol{s}(k))$ spaces, where $\mathcal{Z}(\boldsymbol{o}(k) \mid \boldsymbol{s}(k))$ denotes the probability of making observation $\boldsymbol{o}(k)$ given that the system is in state $\boldsymbol{s}(k)$ (Kaelbling et al., 1998). Although increasing the randomness in the agent via stochastic policies might increase the probability of encountering a good action (Singh et al., 1994; Kaelbling et al., 1998), conventional RL algorithms to POMDPs yield suboptimal results because the performance of the best stationary stochastic policy can be unboundedly worse than the optimal policy in the underlying MDP (Singh et al., 1994).

Another option is to reduce the POMDP to an MDP by using the memory of previous actions and observations. In this formulation, the history of observations and actions are augmented in an information vector $\boldsymbol{I}(k) = [\boldsymbol{o}(0); \boldsymbol{o}(1); \dots \boldsymbol{o}(k); \boldsymbol{a}(0); \boldsymbol{a}(1); \dots; \boldsymbol{a}(k-1)]$ (Bertsekas, 2007). The information vector can be then fed to the controller as the state vector. This approach intuitively leverages the maximum information available to the agent. From a dynamical systems point of view, partial observations (similarly to dimension reduction in modelling) introduce non-Markovian memory effects, which is formalised by the Mori-Zwanzig framework (Gouasmi et al.,

Algorithm 6 Update of actor and critic using DDPG.

1: **Sample batch** of N_b random tuples $\left(\boldsymbol{s}(k)^{(i)}, \boldsymbol{a}(k)^{(i)}, r(k)^{(i)}, \boldsymbol{s}(k+1)^{(i)}\right),\ i = 1, 2, \ldots, N_b$ from $\mathcal{B}$

2: **Set temporal difference error for critic**:

$$\delta^{(i)} = r(k)^{(i)} + \gamma Q^{\pi\prime}\left(\boldsymbol{s}(k+1)^{(i)}, \boldsymbol{\pi}'\left(\boldsymbol{s}(k+1)^{(i)}; \boldsymbol{\theta}^{\pi'}\right); \boldsymbol{\theta}^{Q'}\right) - Q^{\pi}\left(\boldsymbol{s}(k)^{(i)}, \boldsymbol{a}(k)^{(i)}; \boldsymbol{\theta}^{Q}\right)$$

3: **Update critic** by minimizing loss:

$$L_Q = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(\delta^{(i)}\right)^2$$

Compute the gradient $\nabla_{\boldsymbol{\theta}^Q} L_Q$ by backpropagation and apply gradient descent on $\boldsymbol{\theta}^Q$.

4: **Update actor** by minimizing the loss:

$$L_\pi = -\frac{1}{N_b}\sum_{i=1}^{N_b} Q^{\pi}\left(\boldsymbol{s}(k)^{(i)}, \boldsymbol{\pi}\left(\boldsymbol{s}(k)^{(i)}; \boldsymbol{\theta}^{\pi}\right); \boldsymbol{\theta}^{Q}\right).$$

Compute the policy gradient $\nabla_{\boldsymbol{\theta}^\pi} L_\pi$ using backpropagation and the chain rule:

$$\nabla_{\boldsymbol{\theta}^\pi} Q^{\pi}\left(\boldsymbol{s}, \boldsymbol{\pi}\left(\boldsymbol{s}; \boldsymbol{\theta}^{\pi}\right); \boldsymbol{\theta}^{Q}\right) = \nabla_{\boldsymbol{a}} Q^{\pi}\left(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}^{Q}\right)\big|_{\boldsymbol{a}=\boldsymbol{\pi}(\boldsymbol{s};\boldsymbol{\theta}^\pi)} \nabla_{\boldsymbol{\theta}^\pi}\boldsymbol{\pi}\left(\boldsymbol{s}; \boldsymbol{\theta}^{\pi}\right)$$

Apply gradient descent on $\boldsymbol{\theta}^\pi$.

5: **Update target networks**:

$$\boldsymbol{\theta}^{Q'} \leftarrow \alpha_{\text{soft}}\boldsymbol{\theta}^{Q} + (1-\alpha_{\text{soft}})\boldsymbol{\theta}^{Q'}$$
$$\boldsymbol{\theta}^{\pi'} \leftarrow \alpha_{\text{soft}}\boldsymbol{\theta}^{\pi} + (1-\alpha_{\text{soft}})\boldsymbol{\theta}^{\pi'}$$

2017). For chaotic dynamical systems, the augmentation of the state history can be justified through Takens's embedding theorem (Takens, 1981) that states that a chaotic attractor can be reconstructed from delayed embeddings of a single observable. On the other hand, the dimension of the information vector $\boldsymbol{I}(k)$ increases at every new measurement time. Therefore, instead we are interested in keeping a "sufficient statistic" with a smaller dimension that encapsulates the necessary information from $\boldsymbol{I}(k)$, i.e., is Markovian, (Bertsekas, 2007). The belief state (Kaelbling et al., 1998), which expresses the posterior distribution over the state given observations, is a sufficient statistic. For systems with a finite state space, the belief state is a probability vector, where each entry corresponds to the probability of the system being in a particular state (Krishnamurthy, 2016). For continuous state spaces, the belief becomes a continuous probability density function over the state space. Given a model and observations, the belief, or the state estimate, can be updated sequentially in a Bayesian framework, which is introduced next.

6.2.3 State estimation via data assimilation

Due to uncertainties in the modelling and initial conditions as well as due to the chaotic nature of the system, the model's prediction will decorrelate from the true state after some time steps. By assimilating the observations from the environment $\boldsymbol{o}$ and the prediction of the full state from the model (the forecast), sequential DA provides an improved prediction on the full state (the analysis) which can then be fed into the actor without the need to modify the RL algorithm. In a Markovian setting (See § 6.2 Eq. (6.2)), sequential DA methods leverage Bayesian statistics to provide real-time analysis without full data history. The posterior of the state estimate is found by the Bayes' rule

$$\underbrace{\mathcal{P}(\boldsymbol{s}(k) \mid \boldsymbol{o}(1:k))}_{\text{Posterior}} \quad \propto \quad \underbrace{\mathcal{P}(\boldsymbol{o}(k) \mid \boldsymbol{s}(k))}_{\text{Observations' likelihood}} \quad \underbrace{\mathcal{P}(\boldsymbol{s}(k) \mid \boldsymbol{o}(1:k-1))}_{\text{Prior}}, \tag{6.9}$$

where $\boldsymbol{o}(1:k) = \{\boldsymbol{o}(1),\ \boldsymbol{o}(2),\ \ldots,\ ,\boldsymbol{o}(k)\}$ are the observations up until time step k. The analysis state is then chosen as the maximum a posteriori (MAP) estimate, i.e., it maximises the posterior (6.9). The observations' likelihood is determined from the measurement model (6.3). The prior evolves according to the Chapman–Kolmogorov equation, which propagates uncertainty forward in time through the process model (6.1),

$$\mathcal{P}(\boldsymbol{s}(k) \mid \boldsymbol{o}(1:k-1)) = \int \mathcal{P}(\boldsymbol{s}(k) \mid \boldsymbol{s}(k-1))\mathcal{P}(\boldsymbol{s}(k-1) \mid \boldsymbol{o}(1:k-1))d\boldsymbol{s}(k-1). \tag{6.10}$$

For linear dynamics and Gaussian statistics, the optimal solution to the Bayesian filtering problem is given by the Kalman filter. For nonlinear systems, the prior is not analytically tractable, but it can be approximated using ensemble methods based on Monte Carlo sampling. Ensemble data assimilation represents the state distribution with a finite set of n_{en} samples, where the sampling error scales as $\mathcal{O}(n_{\text{en}}^{-1/2})$ due to the central limit theorem. Under Gaussian assumptions, the model expectation and uncertainty are approximated by the ensemble mean and covariance, respectively, where the mean is the MAP estimate. Following (Nóvoa et al., 2024b), we define the augmented state

$$\tilde{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{M}(\boldsymbol{s}) \end{bmatrix}, \tag{6.11}$$

such that its covariance is

$$\boldsymbol{C}_{\tilde{s}\tilde{s}} = \begin{bmatrix} \boldsymbol{C}_{ss} & \boldsymbol{C}_{sM} \\ \boldsymbol{C}_{Ms} & \boldsymbol{C}_{MM} \end{bmatrix} \approx \frac{1}{n_{\text{en}} - 1} \sum_{j=1}^{n_{\text{en}}} (\tilde{\boldsymbol{s}}_m - \bar{\tilde{\boldsymbol{s}}}) \otimes (\tilde{\boldsymbol{s}}_m - \bar{\tilde{\boldsymbol{s}}}), \tag{6.12}$$

where $m = 1, 2, \ldots, n_{\text{en}}$ denotes each ensemble member, $\bar{\tilde{\boldsymbol{s}}}$ is the ensemble mean and $\otimes$ is the dyadic product. This formulation accounts for nonlinear observation operators, whilst being exactly equivalent to the standard formulation of the EnKF (Evensen, 2009) in the linear case. With this, the EnKF update to the state reads

$$\boldsymbol{s}_m^{\text{a}} = \boldsymbol{s}_m^{\text{f}} + \boldsymbol{K}\Big[\boldsymbol{o}_m - \boldsymbol{M}(\boldsymbol{s}_m^{\text{f}})\Big], \tag{6.13}$$

$$\boldsymbol{K} = \boldsymbol{C}_{sM}^{\text{f}}\Big(\boldsymbol{C}_{oo} + \boldsymbol{C}_{MM}^{\text{f}}\Big)^{-1}, \tag{6.14}$$

where $\boldsymbol{K}$ is the Kalman gain, and the superscripts 'f' and 'a' indicate 'forecast' (prior update) and 'analysis' (posterior update), respectively. A derivation of the posterior update from a Bayesian perspective is included in Appendix § C.1.

To avoid underestimating the uncertainty in the analysis, we perturb the observation $\boldsymbol{o}$ such that each ensemble member is updated with a different $\boldsymbol{o}_m \sim \mathcal{N}(\boldsymbol{o}, \boldsymbol{C}_{oo})$, where $\boldsymbol{C}_{oo}$ is the user-defined observation error covariance (Anderson, 2001). We define

$$\boldsymbol{C}_{oo}(k) = \sigma_o^2 \left(\max\left(\left| \boldsymbol{M}(\boldsymbol{s}(k)) \right| \right) \right)^2 \boldsymbol{I}_{n_o \times n_o}, \tag{6.15}$$

where σ_o is the noise level, $\boldsymbol{I}$ is the identity matrix, and we apply the element-wise absolute value and maximum operations on the noise-free observations. This choice of covariance matrix assumes uncorrelated observations and ensures that the noise is proportional to the observed flow state, which is not statistically stationary when applying control actions to a system before stabilisation. To further minimise the covariance underestimation, we apply a multiplicative inflation to the ensemble as

$$\boldsymbol{s}_m^{\mathrm{a}} = \bar{\boldsymbol{s}}^{\mathrm{a}} + \rho_I (\boldsymbol{s}_m^{\mathrm{a}} - \bar{\boldsymbol{s}}^{\mathrm{a}}), \tag{6.16}$$

where $\rho_I > 1$ is inflation factor (Evensen, 2009).

As illustrated in Fig. 6.2 the EnKF is an iterative process with two steps:

(i) *Forecast step (prior update)*: Forecast the ensemble of states with the model $\boldsymbol{F}$ until a time when there are observations $\boldsymbol{o}$ available from the environment. This yields the forecast ensemble $\{\boldsymbol{s}^{\mathrm{f}}\}_{m=1}^{n_{\mathrm{en}}}$, which is the prior of the states.

(ii) *Analysis step (posterior estimate)*: When observations $\boldsymbol{o}$ from the environment are available, we perturb them to create an ensemble $\{\boldsymbol{o}\}_{m=1}^{n_{\mathrm{en}}}$ and apply the EnKF (6.13). The EnKF results in the analysis ensemble $\{\boldsymbol{s}^{\mathrm{a}}\}_{m=1}^{n_{\mathrm{en}}}$, which is then inflated (6.16) and used as the new initial condition in (i).

6.2.4 Data-assimilated model-informed reinforcement learning *(DA-MIRL)*

In model-free RL, the agent acts as a compensator, simultaneously performing both state estimation and control without explicitly modelling the underlying system dynamics. This requires end-to-end training of the observer, which learns a mapping from the observations to a representation of the system's state, and the controller, which learns an optimal policy based on this state. In general, this joint learning problem is highly nonconvex and is challenging to optimise. We break this problem down into modules and propose a data-assimilated model-informed reinforcement learning framework, i.e., the DA-MIRL which integrates:

1. *Model-free agent*: an off-policy actor-critic RL algorithm for learning the control policy, specifically a DDPG (§ 6.2.1).

2. *Environment model:* a predictive model of the system's dynamics, i.e., a model of the environment. The model can be either equation-based, or data-driven if the system's equations are unknown or are too expensive for real-time simulation. We discuss two examples of imperfect

models in § 6.3.2.

3. *State estimator*: an ensemble-based data assimilation method for real-time state estimation, specifically the EnKF. The estimator updates the model predictions using observations from the environment. Then, the full state prediction from the model is fed to the agent, which bypasses the need to modify the implementation of classical RL algorithms for POMDPs (§ 6.2.3).

Here, we choose the off-the-shelf DDPG and EnKF algorithms to demonstrate the methodology without additional algorithmic complexity. Alternative RL or ensemble DA algorithms (e.g., Soft

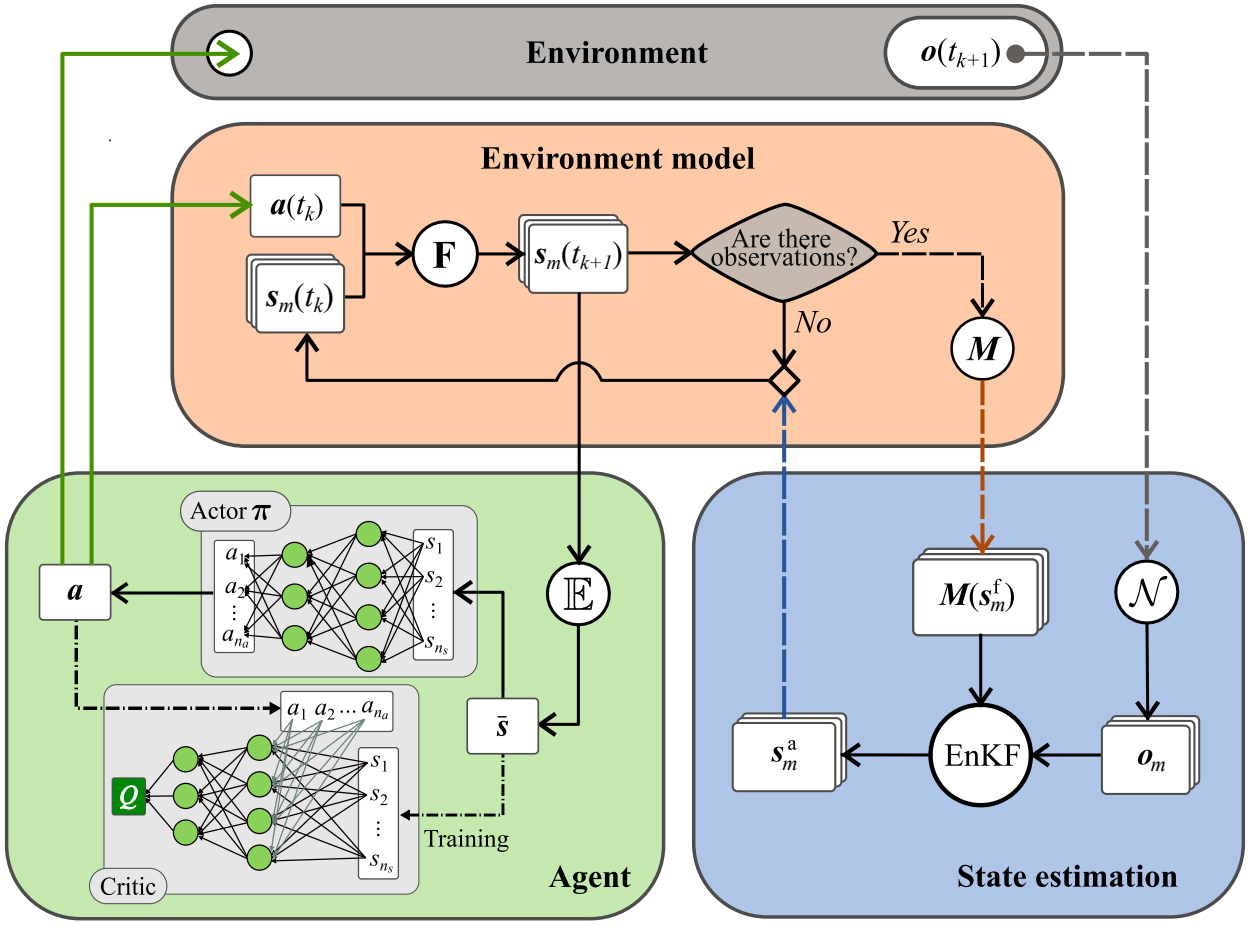


Figure 6.2: Schematic of the data-assimilated model-informed reinforcement learning (DA-MIRL), which aims to control a partially observed environment (grey). The DA-MIRL integrates three components: model environment (orange), state estimation (blue) and agent (green). The numerical model $\boldsymbol{F}$ approximates the environment by forecasting an ensemble of states $\boldsymbol{s}_m$ (stacked boxes). If there are no observations, the model runs autonomously (that is to say without intervention and not without actuation), otherwise, the model prediction is updated by the state estimator. The state estimator perturbs the observations $\boldsymbol{o}$ and assimilates them with the forecast ensemble via the ensemble Kalman filter (EnKF), which results in the analysis ensemble $\boldsymbol{s}_m^{\mathrm{a}}$. The model $\boldsymbol{F}$ is re-initialised with $\boldsymbol{s}_m^{\mathrm{a}}$. The actor-critic agent interacts with both the environment and its model to apply and determine the optimal action at any time k. The critic, which is active only during training, approximates the Q-function from the state-action pair. The actor (i.e., the policy $\boldsymbol{\pi}$) determines the action $\boldsymbol{a}$ from the expected value of the full state of the environment. We do not have access to the environment's full state, hence, we feed the expected value given by the model, i.e., the ensemble mean $\bar{\boldsymbol{s}}$.

Actor-Critic or square-root filters) can be implemented for improving the performance whilst keeping the same theoretical background. The algorithm is fully implemented in `JAX` (Bradbury et al., 2018), leveraging parallelisation for the ensemble computations and Just-In-Time (JIT) compilation for the efficiency, and runs on GPU. The computation times for the test case with the KS equation are provided in the Appendix § E.3. The implementation of the ensemble Kalman filter (EnKF) has been adapted from Nóvoa (2024), and the implementation of the deep deterministic policy gradient (DDPG) has been adapted from Huang et al. (2022).

Algorithm 7 DA-MIRL: Data-assimilated model-informed reinforcement learning

Initialise DDPG, $Q^\pi, \boldsymbol{\pi}, Q^{\pi\prime}, \boldsymbol{\pi}', \mathcal{B} \leftarrow$ Alg. 5
for episode = 1 to training episodes **do**
  Initialise ensemble $\{\boldsymbol{s}_m\}_{m=1}^{n_\text{en}}$, $\boldsymbol{s}_m \sim \mathcal{N}(\bar{\boldsymbol{s}}_0, \boldsymbol{C}_{ss}^0)$
  while $k <$ observation starts **do**
    for each ensemble member $m = 1$ to n_en **do**
      $\boldsymbol{s}_m \leftarrow F(\boldsymbol{s}_m, \boldsymbol{0})$
  for each observations $\boldsymbol{o}(k)$ in the episode **do**
    Perturb $\boldsymbol{o}_m(k) \sim \mathcal{N}(\boldsymbol{o}(k), \boldsymbol{C}_{oo}^k)$
    for each ensemble member $m = 1$ to n_en **do** ▷ Analysis step
      $\boldsymbol{s}_m(k) \leftarrow \boldsymbol{s}_m(k) + \boldsymbol{K}(\boldsymbol{o}_m(k) - \boldsymbol{M}(\boldsymbol{s}_m(k)))$ (6.13)
      $\bar{\boldsymbol{s}}(k) \leftarrow \frac{1}{n_\text{en}} \sum_m \boldsymbol{s}_m(k)$
      $\boldsymbol{s}_m(k) \leftarrow \bar{\boldsymbol{s}}(k) + \rho_I(\boldsymbol{s}_m(k) - \bar{\boldsymbol{s}}(k))$
    while There are no observations **do**
      Select action $\boldsymbol{a}(k) \leftarrow \boldsymbol{\pi}(\bar{\boldsymbol{s}}(k)) + \boldsymbol{\epsilon}_a(k)$, $\boldsymbol{\epsilon}_a(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_{aa})$
      for $m = 1$ to n_en **do** ▷ Forecast step
        $\boldsymbol{s}_m(k+1) \leftarrow \boldsymbol{F}(\boldsymbol{s}_m(k), \boldsymbol{a}(k))$
      $\bar{\boldsymbol{s}}(k+1) \leftarrow \frac{1}{n_\text{en}} \sum_m \boldsymbol{s}_m(k+1)$
      Store transition $(\bar{\boldsymbol{s}}(k), \boldsymbol{a}(k), r(k), \bar{\boldsymbol{s}}(k+1)) \rightarrow \mathcal{B}$
      Update networks $Q^\pi, \boldsymbol{\pi}, Q^{\pi\prime}, \boldsymbol{\pi}' \leftarrow$Alg. 6
      $k \leftarrow k + 1$

6.3 Application to spatiotemporal chaos

We test the DA-MIRL framework using a twin experiment approach on the KS equation (§ 3.5). Previous works have investigated the RL-based control of the KS equation as the system serves as a testbed for studying the control of spatiotemporal chaos. Bucci et al. (2019) showed that RL can discover nonlinear control policies that stabilise the KS equation around its unstable equilibria. Zeng and Graham (2021) developed a symmetry-aware architecture, which enhanced the sample efficiency and control performance of the RL agent in this problem. In this study, they further compared the RL-based control with a traditional model-based optimal control method, the linear quadratic regulator (LQR). Specifically, LQR determines the optimal state-feedback law for a linearised model around a given state. Zeng and Graham (2021) found that the linearised dynamics

around the trivial zero solution are not controllable with their actuation setup, therefore making LQR ineffective in this region. When the dynamics are instead linearised around the nontrivial steady state solution that the RL agent drives the system toward, the LQR based control also becomes capable of stabilisation around this state. While LQR provides an efficient controller when a controllable linearisation of the model is available, unlike RL, it cannot discover global control strategies or stabilising operating points. While adaptive linear control methods may partially address this limitation, we adopt a learning-based framework that can discover global nonlinear optimal control strategies directly from data, consistent with the results in recent flow control applications (e.g., Vignon et al., 2023).

In our framework, we define a numerically simulated *true* environment of the KS equation (§ 6.3.1) which is assumed unknown, and we approximate the truth with approximate models of the environment (§ 6.3.2). The same control input is applied to both the true and model environments (§ 6.3.3), and the performance of the DA-MIRL is evaluated by comparing model predictions with the true solution.

6.3.1 The environment

The state of the environment (the truth) is the solution of the KS equation in the physical domain, resolved using $n_f^{\text{true}} = 64$ Fourier modes, i.e.,

$$\boldsymbol{u}^{\text{true}}(k) \coloneqq \boldsymbol{u}(\boldsymbol{x}_s, t_k) = \mathscr{F}^{-1}(\boldsymbol{c}^{\text{true}}(k))[\boldsymbol{x}_s], \tag{6.17}$$

where $\boldsymbol{x}_s = [0, 1, \ldots, 63]^\top L/64$ are the grid points following the numerical discretisation, we denote the velocity vector $\boldsymbol{u}(\boldsymbol{x}_s, t_k) = [u(x_{s,1}, t_k), u(x_{s,2}, t_k), \ldots, u(x_{s,64}, t_k)]^\top$ with $x_{s,j}$ being j^{th} element of $\boldsymbol{x}_s$, and $\mathscr{F}^{-1}(\cdot)[\boldsymbol{x}]$ denotes the inverse Fourier transform of $(\cdot)$ evaluated at $\boldsymbol{x}$. This true state is assumed unknown. Instead, the sensors provide observations $\boldsymbol{o}(k)$, which are noisy discrete samples of (6.17)

$$\tilde{\boldsymbol{o}}(k) \coloneqq \boldsymbol{u}(\tilde{\boldsymbol{x}}_o, t_k) + \boldsymbol{\epsilon}(k), \tag{6.18}$$

where $\tilde{\boldsymbol{x}}_o = [0, 1, \ldots, (n_o - 1)]^\top L/n_o$ are the uniformly located sensors' locations, we denote the velocity vector $\boldsymbol{u}(\tilde{\boldsymbol{x}}_o, t_k) = [u(\tilde{x}_{o,1}, t_k), u(\tilde{x}_{o,2}, t_k), \ldots, u(\tilde{x}_{o,n_o}, t_k)]^\top$ with $\tilde{x}_{o,i}$ being i^{th} element of $\tilde{\boldsymbol{x}}_o$, $\boldsymbol{\epsilon}(k)$ is the observation noise (which we assume Gaussian in § 6.2.3), and the state is fully observed with $n_o = 64$. In practice, we solve for $\boldsymbol{u}^{\text{true}}$ on the grid points defined by $\boldsymbol{x}_s$ and then determine the actual observations by rounding the sensor location to the nearest grid point,

$$x_{o,i} \coloneqq x_{s,j^*}, \quad \text{where} \quad j^* = \arg\min_j |\tilde{x}_{o,i} - x_{s,j}|, \tag{6.19}$$

for $i = 1, \ldots, n_o$, which are the sensor indices, and where $x_{o,i}$ is i^{th} element of $\boldsymbol{x}_o$. So, we obtain the observations as

$$\boldsymbol{o}(k) \coloneqq \boldsymbol{M}^{\text{true}}\boldsymbol{u}^{\text{true}}(k) + \boldsymbol{\epsilon}(k), \tag{6.20}$$

where $\boldsymbol{M}^{\text{true}} \in \mathbb{R}^{n_o \times n_s}$ is the observation operator

$$M_{ij}^{\text{true}} = \begin{cases} 1, & \text{if } x_{o,i} = x_{s,j}, \\ 0, & \text{otherwise.} \end{cases} \tag{6.21}$$

For ease of notation, we will denote the vector slicing operation as $\boldsymbol{u}^{\text{true}}(k)[\boldsymbol{x}_o] \coloneqq \boldsymbol{M}^{\text{true}}\boldsymbol{u}^{\text{true}}(k)$.

The implementation of the solver and the environment have been adapted from Weissenbacher (2024).

6.3.2 The environment model

Since the true state of the system is assumed unknown, we use surrogate models to approximate the underlying dynamics of the environment. We test the DA-MIRL framework on a physics-based model (§ 6.3.2(a)) and a data-driven model (§ 6.3.2(b)). The framework is applicable across the spectrum of modelling choices, from purely physics-based to purely data-driven, where hybrid modelling approaches can also be explored. The model prediction of the true state at time k, denoted by $\hat{\boldsymbol{u}}(k)$, is corrected with observations $\boldsymbol{o}(k)$, as detailed in § 6.2.3.

(a) Physics-based model: truncated Fourier model

We first consider a physics-based low-order model of the KS equation by truncating the Fourier series to a lower number of modes than used to create the environment (§ 6.3.1). This is, $n_f < n_f^{\text{true}} = 64$. This scenario is encountered in reality when we have an accurate model, but because of computational constraints, we cannot fully resolve it and as a result we have a coarse-grained model. First, this truncation discards the effects of higher wavenumber modes, which due to the nonlinearity in the system are also coupled with the lower wavenumber modes. Second, the truncation also leads to aliasing when projecting the forcing $f(x,t)$ in the controlled KS equation (3.21), which further degrades the model performance. (We will precisely define the forcing as a mixture of Gaussians in Eq. 6.31). To circumvent this issue, we first resolve the forcing on a finer spatial grid and then truncate it to determine the forcing $f_j(t)$ at the wavenumbers κ_j in the spectral equations (3.23).

The augmented state vector of the truncated Fourier model, as defined § 6.2.3 is

$$\tilde{\boldsymbol{s}}_m = \begin{bmatrix} \boldsymbol{s}_m \\ \boldsymbol{M}(\boldsymbol{s}_m) \end{bmatrix} \coloneqq \begin{bmatrix} \boldsymbol{c}_m \\ \mathscr{F}^{-1}(\boldsymbol{c}_m)[\boldsymbol{x}_o] \end{bmatrix}, \tag{6.22}$$

for each ensemble member $m = 1, \ldots, n_{\text{en}}$. We map the Fourier coefficients to the observations using the mapping (3.22) at the sensor locations $\boldsymbol{x}_o$.

The expected reconstructed state is given by

$$\bar{\boldsymbol{s}}_{\text{RL}} = \frac{1}{n_{\text{en}}} \sum \hat{\boldsymbol{u}}_m \coloneqq \frac{1}{n_{\text{en}}} \sum \mathscr{F}^{-1}(\boldsymbol{c}_m)[\boldsymbol{x}_{n_f}], \tag{6.23}$$

which is the input to the actor-critic networks, where $\boldsymbol{x}_{n_f} = [0, 1, \ldots, (n_f - 1)]^\top L/n_f$. This mapping to the real-domain defined by the inverse Fourier transform (3.22) avoids complex-valued inputs to the actor-critic. This transformation is a linear operation, where the dimension of the state is equal to the number of Fourier modes of the model. We can obtain the velocity values at the same grid locations as the true state (6.17) using the mapping (3.22) at the locations $\boldsymbol{x}_s$, which is how we will reconstruct the true state using the truncated Fourier model in the upcoming

figures. However, we do not feed this to the RL agent, as it does not add any new information since higher wavenumbers are not captured by the low-order model.

Each ensemble member is initialised by sampling from a complex Gaussian distribution centered at $\bar{\boldsymbol{c}}_0 := \boldsymbol{c}(t_0)$. The real and imaginary components are treated independently, i.e.,

$$\mathrm{Re}(\boldsymbol{c}_m(t_0)) \sim \mathcal{N}\left(\mathrm{Re}(\bar{\boldsymbol{c}}_0), \sigma_0^2 \cdot \mathrm{diag}\left(\mathrm{Re}(\bar{\boldsymbol{c}}_0) \odot \mathrm{Re}(\bar{\boldsymbol{c}}_0)\right)\right) \tag{6.24a}$$

$$\mathrm{Im}(\bar{\boldsymbol{c}}_m(t_0)) \sim \mathcal{N}\left(\mathrm{Im}(\bar{\boldsymbol{c}}_0), \sigma_0^2 \cdot \mathrm{diag}\left(\mathrm{Im}(\bar{\boldsymbol{c}}_0) \odot \mathrm{Im}(\bar{\boldsymbol{c}}_0)\right)\right) \tag{6.24b}$$

where Re and Im denote the real and imaginary components of a vector, and σ_0 is the initial uncertainty scaling. This complex Gaussian structure respects Fourier transform symmetries, assumes independence between Fourier modes and preserves the spectral energy distribution.

(b) Data-driven model: Control-aware echo state network *(CA-ESN)*

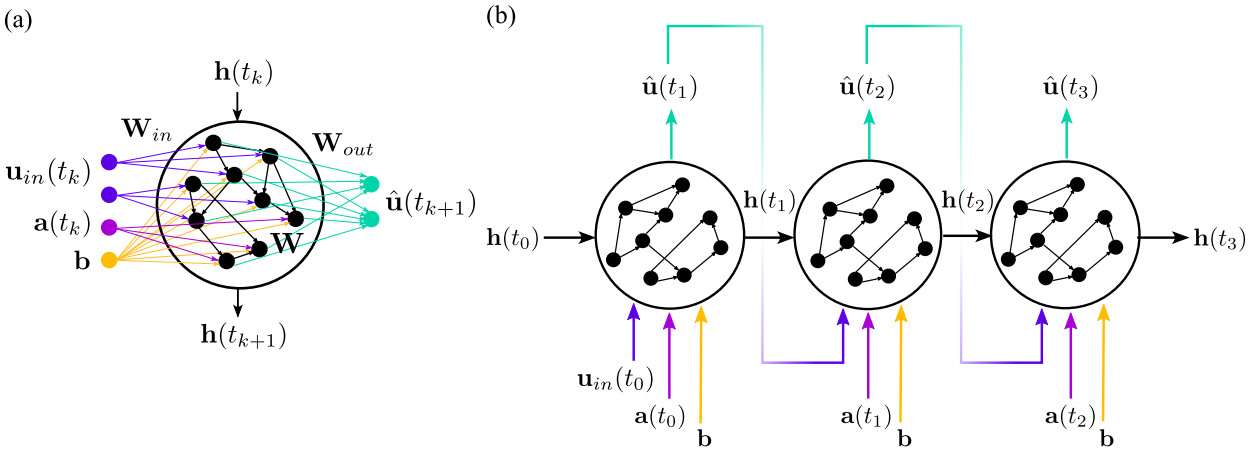


Figure 6.3: Schematic of the control-aware echo state network (CA-ESN), (a) architecture, and (b) closed-loop configuration. The actions, or control inputs, $\boldsymbol{a}$ are augmented to the input vector. The input bias $\boldsymbol{b}$ is added to each reservoir state variable. In closed-loop configuration, the CA-ESN can forecast the actuated dynamics.

Second, we adopt a data-driven approach to approximate the dynamics of the true state. Data-driven models are particularly useful when the governing physical equations are unknown or prohibitively expensive to simulate. Our objective is to develop a model that (i) captures temporal dependencies in the data, (ii) generalises across different dynamical regimes with control inputs, and (iii) admits a state space form suitable for state-feedback control from partial observations, i.e., is control-aware. To this end, we propose a control-aware extension of the echo state network (ESN) (Jaeger and Haas, 2004; Racca and Magri, 2023). A parameter-aware extension of the ESN has been introduced in § 5.2 for capturing parameterised dynamics.

To handle control inputs, we extend the standard ESN architecture by augmenting the input layer with action signals, such that the ESN can learn the actuated dynamics. This modification is illustrated in Fig. 6.3 (a). The ESNs are further appealing for real-time applications such as state estimation and control as (i) their training consists of solving a linear system, which is computationally cheaper than backpropagation-based methods and has one global minimum, (ii) the reservoir dynamics are not trained, so the reservoir state remains consistent throughout the

training, and (iii) the propagation of the dynamics is cheap due to the use of sparse matrices. The control-aware formulation is

$$\boldsymbol{h}(k+1) = (1-\alpha_h)\boldsymbol{h}(k) + \alpha_h \tanh(\boldsymbol{W}_{\text{in}}\tilde{\boldsymbol{u}}_{\text{in}}(k) + \boldsymbol{W}_h\boldsymbol{h}(k) + \boldsymbol{b}), \tag{6.25}$$

where $\boldsymbol{h} \in \mathbb{R}^{n_h}$ is the reservoir state, and $\boldsymbol{b} \in \mathbb{R}^{n_h}$ is the input bias vector, the entries of which are sampled from $\mathcal{U}(-1,1)$, and $\tilde{\boldsymbol{u}}_{\text{in}} \in \mathbb{R}^{n_u+n_a}$ is the augmented input, defined as

$$\tilde{\boldsymbol{u}}_{\text{in}}(k) = [(\boldsymbol{u}_{\text{in}}(k) - \bar{\boldsymbol{U}}) \oslash \boldsymbol{\sigma}_U; \boldsymbol{a}(k)], \tag{6.26}$$

i.e., the concatenation of the raw input state $\boldsymbol{u}_{\text{in}}$ (standardised component-wise by the temporal mean $\bar{\boldsymbol{U}}$ and standard deviation $\boldsymbol{\sigma}_U$ of the training data $\boldsymbol{U} \in \mathbb{R}^{n_u \times N_{\text{train}}}$, where $\oslash$ indicates element-wise division) and the control actions $\boldsymbol{a}$ (not normalised since $a_i \in [-1,1]$, see § 6.3.3). In the augmented input (6.26), we directly feed the full state $\boldsymbol{u}(k)$ as input, which we assume is available during the training time.

The input matrix describes the connections of the inputs to the reservoir and is generated by first constructing a matrix $\widetilde{\boldsymbol{W}}_{\text{in}}$ such that each row contains only one nonzero element drawn from a uniform distribution $\sim \mathcal{U}(-1,1)$. The weights associated with the input state and action vectors are then scaled as $\boldsymbol{W}_{\text{in}} = [\xi_{\text{in}}\widetilde{\boldsymbol{W}}^y_{\text{in}} \quad \xi_a\widetilde{\boldsymbol{W}}^a_{\text{in}}]$, where $\widetilde{\boldsymbol{W}}^u_{\text{in}} \in \mathbb{R}^{n_h \times n_u}$ and $\widetilde{\boldsymbol{W}}^a_{\text{in}} \in \mathbb{R}^{n_h \times n_a}$. The state matrix $\boldsymbol{W}_h$ describes the connections between the reservoir state variables, and is obtained by $\boldsymbol{W}_h = \rho_h\widetilde{\boldsymbol{W}}_h$, where $\widetilde{\boldsymbol{W}}_h$ is an Erdős-Renyi matrix with a given average number of nonzero elements per row (connectivity, n_{conn}) drawn from a uniform distribution $\sim \mathcal{U}(-1,1)$ scaled such that its spectral radius is unity, which lets ρ_h be the spectral radius of $\boldsymbol{W}_h$. The prediction is performed by a linear readout layer

$$\hat{\boldsymbol{u}}(k+1) = \boldsymbol{W}_{\text{out}}[\boldsymbol{h}(k+1); 1], \tag{6.27}$$

where $\boldsymbol{u} \in \mathbb{R}^{n_u}$ is the target output vector, $\hat{\boldsymbol{u}} \in \mathbb{R}^{n_u}$ is the predicted output vector, and $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{n_u \times (n_h+1)}$ is the output matrix. We find that it is necessary to break the anti-symmetry of the $\tanh(\cdot)$ for the prediction of the KS equation when using the linear readout. This is achieved by adding the input bias $\boldsymbol{b}$, where each element of $\boldsymbol{b}$ is sampled from the uniform distribution $\mathcal{U}(-1,1)$. (It is important that this bias is added to each reservoir state variable.) An alternative is to use the so-called Lu readout (Lu et al., 2017; Pathak et al., 2018)

$$\hat{\boldsymbol{u}}(k+1) = \boldsymbol{W}_{\text{out}}[\tilde{\boldsymbol{h}}(k+1); 1],\ \tilde{h}_i(k) = \begin{cases} h_i(k) & \text{if } i \text{ is odd,} \\ h_i^2(k) & \text{if } i \text{ is even,} \end{cases} \quad i = 1, 2, \ldots, n_h, \tag{6.28}$$

The squared reservoir state variables also break the antisymmetry of tanh and enable closed-loop prediction; however this results in a nonlinear observation operator in the data assimilation step. We find that the augmented state formulation (6.11) can handle this nonlinear observation operator (these results are shown in Appendix D.1); however, it is both computationally more efficient and exact to use the linear observation operator (6.27).

Similarly to the PA-ESN 5.2, whilst the input and state matrices $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}_h$ are randomly generated at the initialisation and fixed, the output weights $\boldsymbol{W}_{\text{out}}$ are trained in open-loop con-

figuration via Ridge regression. To train the ESN, first we feed $\tilde{\boldsymbol{u}}_{\text{in}}(k),\ k = 0, 1, \ldots, N_{\text{train}} - 1$ to the control-aware ESN and save the resulting reservoir state according to (6.25) in $\boldsymbol{H} = [\boldsymbol{h}(1)\ \boldsymbol{h}(2)\ \ldots\ \boldsymbol{h}(t_{N_{\text{train}}})],\ \boldsymbol{H} \in \mathbb{R}^{n_h \times N_{\text{train}}}$. We then solve for $\boldsymbol{W}_{\text{out}}$ that minimises the prediction error under regularisation of the weights to prevent overfitting (See Eqs. (5.3) and (5.4) for the solution of the Ridge regression, where we replace $\tilde{\boldsymbol{y}}$ with $\tilde{\boldsymbol{u}}_{\text{in}}$ and $\hat{\boldsymbol{y}}$ with $\hat{\boldsymbol{u}}$).

Training of an ESN includes optimal hyperparameter selection. The hyperparameters of the control-aware ESN are the leak rate α_h, the input scaling ξ_{in}, the action scaling ξ_a, the spectral radius ρ_h, and the Tikhonov coefficient λ. (Whilst in the PA-ESN, we determined a scaling for each of the parameters, in the CA-ESN, we use a single scaling for all actuations since the number of actuators is high.) We employ the Bayesian optimisation scheme from Racca and Magri (2021) for the hyperparameter search, where we choose the hyperparameters resulting in the minimum validation error (ℓ_2-error) when evaluated in closed-loop on an unseen validation dataset.

We train the control-aware ESN offline on the full state data from the system generated by random actuations. Once the output weights are trained, the ESN can operate in closed-loop configuration by setting $\boldsymbol{u}_{\text{in}}(k) = \hat{\boldsymbol{u}}(k)$, which allows the ESN to forecast the dynamics autonomously. This implies that in this configuration the model's state at time k is described by the reservoir state $\boldsymbol{h}(k)$ only. The closed-loop configuration is illustrated in Fig. 6.3 (b).

The control-aware ESN is used as the surrogate model of the environment. During the DA-MIRL deployment, the ESN runs the ensemble in closed loop between observations, and we perform a single open-loop step at with the analysis ensemble as input. We update the ESN's reservoir state via the EnKF (Nóvoa and Magri, 2025), i.e., we define

$$\tilde{\boldsymbol{s}}_m = \begin{bmatrix} \boldsymbol{s}_m \\ \boldsymbol{M}(\boldsymbol{s}_m) \end{bmatrix} \coloneqq \begin{bmatrix} \boldsymbol{h}_m \\ \hat{\boldsymbol{u}}_m[\boldsymbol{x}_o] \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_m \\ \boldsymbol{M}^{\text{true}} \boldsymbol{W}_{\text{out}}[\boldsymbol{h}_m; 1] \end{bmatrix}. \tag{6.29}$$

for each ensemble member $j = 1, \ldots, n_{\text{en}}$. We map the reservoir state vectors to observations by obtaining the estimate of the full state $\hat{\boldsymbol{u}}$ using the read-out (6.27) and selecting the observation locations.

To initialise the ensemble of reservoir state vector $\boldsymbol{h}_m(t_0),\ m = 1, 2, \ldots, n_{\text{en}}$, we first initialise an ensemble of the full state vector, i.e., $\boldsymbol{u}_m(t_0)$. We use the same sampling strategy as the Fourier model § 6.3.2(a) for comparability of the results across these models. To obtain the reservoir state $\boldsymbol{h}_m(t_0)$ we then feed the initial condition $\boldsymbol{u}_m(t_0)$ repeatedly in open-loop configuration for the duration of the washout, i.e., $N_{\text{wash}} = 100$, during which the reservoir state converges. Using a single instance of the input state for the washout instead of a sequence may introduce additional error, however, this error is mitigated by the data assimilation process.

The expected reconstructed state is

$$\bar{\boldsymbol{s}}_{\text{RL}} = \frac{1}{n_{\text{en}}} \sum \hat{\boldsymbol{u}}_m \coloneqq \frac{1}{n_{\text{en}}} \sum \boldsymbol{W}_{\text{out}}[\boldsymbol{h}_m; 1], \tag{6.30}$$

which is the input to the actor-critic networks.

In the DA-MIRL algorithm, the actor-critic receives the model state estimate $\boldsymbol{s}(k)$ that is Marko-

vian, which for the ESN is the reservoir state $\boldsymbol{h}(k)$. However, this is typically high dimensional, e.g., for the KS equation we set the reservoir state dimension to be $n_h = 1000$. Therefore, instead we feed the actor-critic the estimate of the full state $\hat{\boldsymbol{u}}(k)$ obtained via the read-out (a linear operation), which has a lower dimension, e.g., $n_u = 64$, and we know to be Markovian. During the offline training of the ESN, we assumed that the full-state information is available. This choice is similar to the concept of privileged information methods in RL (e.g., Baisero et al., 2022), which assume access to full-state information provided by simulators during offline training and only observations during online deployment. The training of the ESN (or RNNs in general) on partial observations is a considerably more difficult task than training on the full state (Gupta et al., 2024), which can be addressed for example by physics-informed approaches (e.g. Doan et al., 2020b; Özalp et al., 2023). If such methods are successfully applied, and the ESN output remains a partial, i.e., non-Markovian, observation of the underlying system, the controller can instead operate on the reservoir state or a reduced representation of it. To tackle very high-dimensional problems, strategies such as employing parallel ESNs for different parts of the state (Pathak et al., 2018) or coupling the ESN with an autoencoder for order reduction (Racca et al., 2023) can also be integrated in the DA-MIRL framework.

6.3.3 The agent

Our objective is to solve the optimal control problem to find the control actions that stabilise the KS system starting from a solution in the chaotic attractor. To this aim, we implement a DDPG actor-critic agent (§ 6.2.1). The agent interacts with both the environment (§ 6.3.1) and its model (§ 6.3.2), either the truncated Fourier model or the control-aware ESN, enabling simultaneous policy improvement and model adaptation.

Actuation strategy

The forcing is applied as a mixture of Gaussians

$$f(x,t) = \sum_{i=1}^{n_a} a_i(t) \exp\left(-\frac{(x - x_a^i)^2}{2\sigma_a^2}\right), \tag{6.31}$$

where n_a is the number actuators, $\boldsymbol{a}(t) = [a_1(t), a_2(t), \ldots, a_{n_a}(t)]^\top$, $a_i \in [-1, 1]$ are the actuators (i.e., the mixture weights), which are located at the spatial locations (i.e., the mixture means) $\boldsymbol{x}_a = \left[x_a^1, x_a^2, \ldots, x_a^{n_a}\right]^\top$, and σ_a is the width of the actuators. The agent interacts with the environment (and its models) in a zero-order hold (ZOH) approach, i.e., after each control action, the most recent value is kept unchanged until a new action becomes available. We denote the frequency at which a new action is applied to the system as ${\Delta k_a}^{-1}$, where Δk_a is the number of time steps between each new action.

Reward function

To drive the system to the trivial, i.e., zero, solution whilst spending minimal actuation power, we define the reward as a combination of the ℓ_2-norms of the state and the action

$$r(\boldsymbol{u}(k), \boldsymbol{a}(k)) = -\left(\frac{1}{\sqrt{n_f}}||\boldsymbol{u}(k+1)||_2 + \lambda_a||\boldsymbol{a}(k)||_2\right), \tag{6.32}$$

where λ_a penalizes excessive actuation and the state ℓ_2-norm is scaled by $\sqrt{n_f}$ such that the reward is comparable for models with different number of Fourier modes due to our definition of the Fourier expansion in Eq. (3.22). The reward function in this setup (6.32) depends on the full state $\boldsymbol{u}$. However, in practical applications, only partial measurements of $\boldsymbol{u}$ is available. In POMDP settings, it is common to assume that the reward is observable, even when the full state is not. In the DA-MIRL approach, we find that providing the critic with a reward computed from the true (but unobserved) state, whilst relying on the estimated state for value prediction, destabilises its training if a discrepancy between the true state and its estimate exists. Therefore, we also estimate the reward using the model, i.e., we compute $r(\hat{\boldsymbol{u}}(k), \boldsymbol{a}(k))$, for both feasibility and consistency. The optimisation objective (6.6) is infinite horizon, therefore we do not define any terminal reward.

6.4 Training of DA-MIRL

We demonstrate the DA-MIRL framework on the KS equation. The objective is threefold: (i) to construct a predictive model that accurately forecasts the dynamics, (ii) to perform real-time state estimation from partial and noisy observations via data assimilation, and (iii) to learn an optimal control policy based on the state estimates. The agent in the DA-MIRL framework is trained episodically in three stages, as summarised in Alg. 8. The training is composed of three stages of episodes: (1) randomly actuated, where actions are randomly sampled to create experiences for the replay buffer, (2) exploration-learning, where the RL agent explores the solution space and the weights of the actor-critic are updated, and (3) evaluation episodes, where the performance of the RL agent is evaluated to determine the best actor-critic weights. Random re-initialisation of the episodes enable the agent to gain experience in different portions of the state space. For each episode, the ensemble is initialised around the attractor, and then propagated for one Lyapunov time (LT) to allow for sufficient divergence of the trajectories. Prior to training, we start filling the replay buffer with samples from episodes generated by applying random actuations. We then initiate the learning with network updates, while applying the control policy given by the actor network perturbed by Gaussian noise to promote exploration. At intervals, we run evaluation episodes using the control policy given by the target actor network. We save the weights that results in the best evaluation return of the model. In Alg. 7, we show details of a learning episode. The procedure is identical for the other stages except for how the actions are determined.

We apply model-free RL to the KS equation in § 6.5 and then compare these results with the proposed DA-MIRL framework in § 6.6. The test cases in § 6.5 and § 6.6 are performed on the KS equation with $\nu = 0.08$ ($L \approx 22$) using $n_a = 8$ actuators. In the unactuated case, this is a chaotic regime, where the Lyapunov time is approximately equal to 25 time units in Eq. (3.21) or 500 time steps. To assess robustness, additional results in other chaotic regimes are shown in § 6.6.2. The

Algorithm 8 The training procedure of model-free RL and DA-MIRL

Initialisation: Run Alg. 5
Stage 1: Random Actuation ▷ Replay buffer warm-up
Apply random actuation $\boldsymbol{a}(k) \sim \mathcal{U}(-1, 1)$
Store $(\boldsymbol{s}(k), \boldsymbol{a}(k), r(k), \boldsymbol{s}(k+1))$ in buffer $\mathcal{B}$
for episode = 1 to training episodes **do**
Stage 2: Learning ▷ Policy exploration
Apply policy $\boldsymbol{a}(k) = \boldsymbol{\pi}(\boldsymbol{s}(k); \boldsymbol{\theta}^{\pi}) + \boldsymbol{\epsilon}_a(k)$
Store $(\boldsymbol{s}(k), \boldsymbol{a}(k), r(k), \boldsymbol{s}(k+1))$ in $\mathcal{B}$
Update actor and critic using Alg. 6
if mod(episode, evaluation frequency) = 0 **then**
Stage 3: Evaluation ▷ Policy evaluation
Apply target policy $\boldsymbol{a}(k) = \boldsymbol{\pi}'(\boldsymbol{s}(k); \boldsymbol{\theta}^{\pi'})$
if $R_{\text{eval}} > R^{\star}_{\text{eval}}$ **then**
$R^{\star}_{\text{eval}} \leftarrow R_{\text{eval}}$
$\boldsymbol{\theta}^{Q\star} \leftarrow \boldsymbol{\theta}^{Q'}$
$\boldsymbol{\theta}^{\pi\star} \leftarrow \boldsymbol{\theta}^{\pi'}$

training details are provided in Tab. 6.1.

6.4.1 Performance metrics

Here, we define the performance metrics relevant to the training of DA-MIRL, which we will refer to in § 6.5 and § 6.6.

Episode return We define the episode return as the sum of non-discounted rewards in an episode after the observation starts

$$R = \sum_{k=N_{\text{start}}}^{N_{\text{ep}}} r(\boldsymbol{u}(k), \boldsymbol{a}(k)), \tag{6.33}$$

which is computed using the full state vector $\boldsymbol{u}(k)$ from the true system, i.e., the environment.

Actor and critic losses The actor-critic networks are trained simultaneously. During training, we compute the associated losses over a randomly sampled batch N_b from the replay buffer at each time step. We first define the Q-loss, which is minimised to train the critic. This loss is based on the temporal difference (TD) error (Sutton and Barto, 2018; Lillicrap et al., 2016)

$$\delta^{(i)} = r(k)^{(i)} + \gamma Q^{\pi'}\left(\boldsymbol{s}(k+1)^{(i)}, \boldsymbol{\pi}'\left(\boldsymbol{s}(k+1)^{(i)}\right)\right) - Q^{\pi}\left(\boldsymbol{s}(k)^{(i)}, \boldsymbol{a}(k)^{(i)}\right), \tag{6.34}$$

for $i = 1, 2, \ldots, N_b$ for a batch of size N_b, which quantifies how well the learned Q-function satisfies the Bellman equation (6.7). The Q-loss is then computed over the batch

$$L_Q = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(\delta^{(i)}\right)^2. \tag{6.35}$$

The goal of training the actor is then to find the set of weights that maximise the learned Q-function. This is achieved through the policy loss,

$$L_\pi = -\frac{1}{N_b}\sum_{i=1}^{N_b} Q^\pi\left(\boldsymbol{s}(k)^{(i)}, \boldsymbol{\pi}\left(\boldsymbol{s}(k)^{(i)}\right)\right). \tag{6.36}$$

which is then differentiated with respect to the actor weights θ^π to compute the policy gradient (See Alg. 6). The policy loss is minimised via gradient descent using the policy gradient. We apply one gradient update of the actor and critic weights at each time step, though this can be increased.

Validation error of ESN We evaluate the closed-loop performance of the ESN on a validation dataset, consisting of 5 unseen episodes, in order to choose its hyperparameters. We repeat this over 3 realisations of the randomly generated input and state matrices of the ESN, 3 folds, i.e., trajectories, of length 500 time steps (approximately 1 LT) starting from random time steps in the dataset. We set the relative ℓ_2-error between the prediction $\hat{\boldsymbol{u}}$ and the ground truth $\boldsymbol{u}^{\text{true}}$ as the error metric for validation,

$$e = \frac{\sqrt{\sum_{k=1}^{N_{\text{val}}}\sum_{i=1}^{n_u}\left(u_i(k)-\hat{u}_i(k)\right)^2}}{\sqrt{\sum_{k=1}^{N_{\text{val}}}\sum_{i=1}^{n_u}u_i^2(k)}}, \tag{6.37}$$

where N_{val} is the number of validation steps, which we average over the number of folds, number of validation episodes, and number of realisations.

6.5 Baseline analysis of model-free reinforcement learning

We analyse the effectiveness of the model-free RL algorithm, DDPG, (without state estimation) for stabilising the KS system. We examine the agent's performance metrics (§ 6.4.1) by varying (i) the number of sensors, (ii) the location of sensors and actuators, (iii) the level of observation noise, and (iv) the frequency of observation and actuation (Fig. 6.4).

Fig. 6.4 (a) shows the training performance for different number of uniformly placed sensors for $n_o = \{4, 6, 7, 8, 12, 64\}$, where $n_o = n_f^{\text{true}} = 64$ corresponds to the full state measurement case. We find that the model-free algorithm cannot discover stabilising policies with $n_o < 8$ sensors. In this context, the same number of sensors, $n_o = 8$ was employed in Bucci et al. (2019). Similar decrease in performance when the number of sensors is below a certain number was observed by Weissenbacher et al. (2025) for the KS equation in different chaotic regimes, which was referred to as the threshold for partial observability. Although the threshold number of sensors $n_o = 8$ fulfills the Nyquist-Shannon sampling criterion for the first four Fourier modes, which dominate the energy spectrum, this criterion alone is not conclusive to determine the observability threshold. Empirical studies (Gupta et al., 2024; Weissenbacher et al., 2025) suggest that, in chaotic systems, this threshold is related to the dimension of the inertial manifold, which is coincidentally equal to 8 for the $\nu = 0.08$ regime (Cvitanović et al., 2010), or the number of sensors needed for chaos synchronisation. But, formalising an observability condition remains an open question, as the observability may differ for the actuated system and depend on the sensor positions.

Table 6.1: Training parameters for data-assimilated model-informed reinforcement learning (DA-MIRL) agent and control-aware ESN (CA-ESN).

Parameter	**Value/Configuration**
DA-MIRL Agent - DDPG	
Episode length (N_{ep})	1000
Control-free steps (N_{start})	500
Number of random actuation episodes	5
Number of training episodes	95
Buffer capacity	10^5
Actor hidden units	[256, 256]
Critic hidden units	[256, 256]
Activation function	ReLU
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Discount factor (γ)	0.99
Soft update rate (α_{soft})	0.005
Batch size (n_b)	256
Exploration covariance ($\boldsymbol{C}_{aa}$)	$0.2^2 \boldsymbol{I}_{n_a \times n_a}$
Control-aware ESN (CA-ESN)	
Reservoir size (n_h)	1000
Connectivity (n_{conn})	3
Washout time steps (N_{wash})	100
Training time steps (N_{train})	56×10^3
Leak rate (α_h)	0.23
Tikhonov coefficient (λ)	10^{-6}
Spectral radius (ρ_h)	0.07
Input state scaling (ξ_y)	0.23
Input action scaling (ξ_a)	0.51

To test whether observability is affected by the position of actuators and sensors, we repeat the same experiment for $n_o = n_a = 8$ sensors and actuators in different grid configurations. The results for alternating grid locations for the sensors and actuators, $\boldsymbol{x}_{g_1} = [0, 1, \ldots, 7]^\top L/8$ and $\boldsymbol{x}_{g_2} = [1, 3, \ldots, 15]^\top L/16$ are shown in Fig. 6.4 (b). Optimal performance is achieved when the sensors are placed at the same grid locations as the actuators, i.e., $\boldsymbol{x}_a = \boldsymbol{x}_o$. On the other hand, when they are placed in staggered configurations, i.e., $\boldsymbol{x}_a \neq \boldsymbol{x}_o$, the convergence is slower and the performance is suboptimal.

Next, Fig. 6.4 (c) quantifies the impact of observation noise on policy learning in the fully-observable case, i.e., $n_o = 64$. Noise corrupts the state in the Bellman updates and thus, the agent must sample more trajectories to average out the effect of noise. Although the agent stabilises for noise levels for $\sigma_o \leq 5.0\%$ with increased sample complexity, higher variances ($\sigma_o \geq 10\%$) critically degrade the policy learning and destabilises the approximation of the value function, as evidenced by L_Q divergence in Fig. 6.4 II (c).

Lastly, we investigate the effect of the observation frequency $\Delta {k_o}^{-1}$ in the fully-observable noise-free case in Fig. 6.4 (d). (The agent interacts with the environment at the time of observation and then we apply the same action in a ZOH manner until the next observation.) We vary the observation

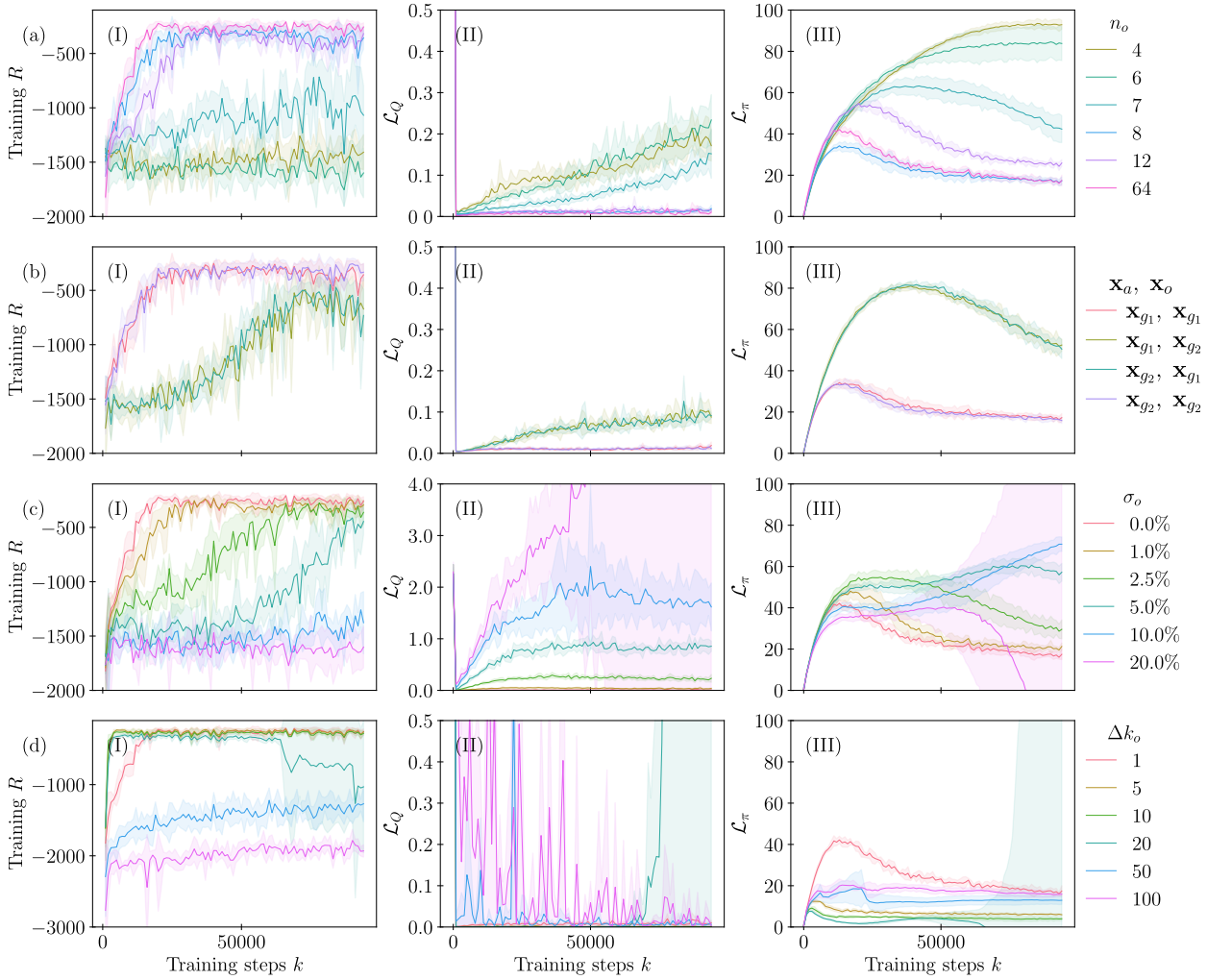


Figure 6.4: Training performance of model-free RL on the KS equation for (a) different number of sensors; (b) different placements of actuators, $\boldsymbol{x}_a$, and sensors, $\boldsymbol{x}_o$, with $\boldsymbol{x}_{g_1} = [0, 1, \ldots, 7]L/8$ and $\boldsymbol{x}_{g_2} = [1, 3, \ldots, 15]L/16$; (c) varying levels of observation error level, $\boldsymbol{\sigma}_o$; and (d) varying frequency of observations (Δk_o). Shown the mean and standard deviation across 5 different runs of (I) non-discounted returns of training episodes, (II) Q-loss, (III) policy loss.

interval between $\Delta k_o \in [1, 100]$, which are under the Lyapunov time for the system. For $\Delta k_o = 5$ and $\Delta k_o = 10$ time steps, we obtain similar performance as interacting with the environment every time step. In these cases, the decrease in sample complexity can be due to exploring more of the phase space with fewer samples as we have downsampled the data, or that the ZOH strategy simplifies the training. The agent's performance decreases for less frequent interactions, i.e., for $\Delta k_o = \{20, 50, 100\}$ time steps. In comparison to the sparse or noisy measurement cases, the Q-loss shows bigger spikes but in contrast decreases over training.

To summarise, the performance of the model-free RL algorithm in finding an optimal stabilising control policy for a chaotic flow depends on (i) the number and the placement of the available sensors, (ii) the noise level on the observations, and (iii) the frequency of observations/interactions. These conditions affect the training of the RL agent, which is captured by the evolution of the Q-loss and the policy loss during training. In the next section, we apply the DA-MIRL to overcome these limitations, which arise in real-life applications.

Table 6.2: Summary of the relation between the environment quantities and their model approximations in DA-MIRL.

	Environment	Environment model $\boldsymbol{F}$		
		General form	Truncated Fourier	Control-aware ESN
Full state	$\boldsymbol{u}^{\text{true}}$ (unknown)	$\hat{\boldsymbol{u}}$	$\mathscr{F}^{-1}(\bar{\boldsymbol{c}})[\boldsymbol{x}_{n_f}]$	$\boldsymbol{W}_{\text{out}}\left[\bar{\boldsymbol{h}}; 1\right]$
Ensemble	–	$\boldsymbol{s}_m$	$\boldsymbol{c}_m$	$\boldsymbol{h}_m$
Observable	$\boldsymbol{o} = \boldsymbol{u}^{\text{true}}[\boldsymbol{x}_o] + \boldsymbol{\epsilon}$	$\boldsymbol{M}(\boldsymbol{s}_m)$	$\mathscr{F}^{-1}(\boldsymbol{c}_m)[\boldsymbol{x}_o]$	$\boldsymbol{M}^{\text{true}}\boldsymbol{W}_{\text{out}}[\boldsymbol{h}_m; 1]$

6.6 Controlling spatiotemporal chaos from partial observations with DA-MIRL

We evaluate the DA-MIRL framework using the physics-based, i.e., truncated Fourier, model (§ 6.3.2(a)) and data-driven, i.e., CA-ESN, model (§ 6.3.2(b)) to approximate the higher-fidelity environment. The results for the truncated Fourier model are presented in § 6.6.1, and those for the CA-ESN are shown in § 6.6.2. Tab. 6.2 summarises the notation of the model and the reference environment quantities. As outlined in § 6.2.4, we employ an ensemble to infer the first two statistical moments. We find that an ensemble size of $n_{\text{en}} = 50$ offers good trade-off between computational efficiency and performance. Increasing the ensemble size from $n_{\text{en}} = 10$ to $n_{\text{en}} = 50$ yields an approximate 50% improvement in the mean episode return after convergence, whereas increasing from $n_{\text{en}} = 50$ to $n_{\text{en}} = 100$ gives only a marginal improvement of $\sim 5\%$. We update the ensemble prediction by assimilating sparse and noisy observations from the environment. The observations are unbiased samples from the environment with a noise level 10%, i.e, $\sigma_o = 0.1$ in (6.15). The EnKF assumes that the environment's model and the observation noise are unbiased. To overcome biased state estimates that can occur due to modelling errors or sensor drifts, a bias-aware data assimilation method (Nóvoa et al., 2024b) can be employed. Observations are assimilated at intervals of $\Delta k_o = 10$, i.e., the observation frequency is $\Delta k_o = 10$. The default system's configurations are listed in Tab. 6.3.

6.6.1 Results of DA-MIRL with truncated Fourier model

In this section, we consider the physics-based model, i.e, the Fourier model with a truncated number of modes in Eq. 3.23. The influence of the truncation level on the accuracy and stability of the simulation are shown in Fig. 6.5. We simulate the KS equation while applying random actuations in the $\nu = 0.08$ regime using models (3.23) with varying number of Fourier modes $n_f = \{14, 16, 20, 32, 64\}$ (the solver is unstable for $n_f < 14$). The initial condition is obtained by randomly initialising the $n_f^{\text{true}} = 64$ model (the environment), and letting it evolve for a transient time until the system settles on the chaotic attractor. We then truncate this initial condition for the respective lower order models

$$c_j^{n_f}(t_0) = \frac{n_f}{n_f^{\text{true}}} c_j^{\text{true}}(t_0), \tag{6.38}$$

Table 6.3: System's configurations of the controlled Kuramoto-Sivashinsky equation for Figs. 6.6-6.15 unless stated otherwise.

Parameter	**Symbol**	**Value**
Number of actuators	n_a	8
Actuators' locations	$\boldsymbol{x}_a$	$(0, L/8, \ldots, 7L/8)^\top$
Number of sensors	n_o	4
Sensors' locations	$\boldsymbol{x}_o$	$(0, L/4, L/2, 3L/4)^\top$
Observation interval	Δk_o	10
Control interval	Δk_a	1
Environment Fourier modes	n_f^{true}	64
Truncated Fourier modes	n_f	16
Reservoir size	n_h	1000
Ensemble size	m	50
Ensemble inflation	ρ_I	1.02
KS viscosity	ν	0.08

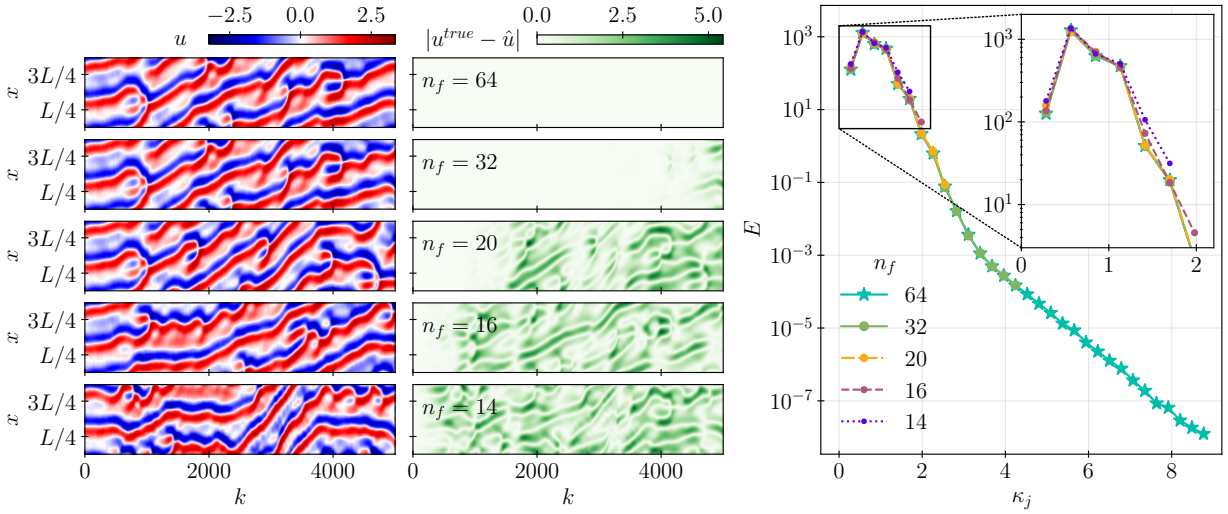


Figure 6.5: Numerical simulations and energy spectra of the KS equation with varying number of Fourier modes using random actuations. Shown spatiotemporal evolution of the KS equation and reconstruction error using (a) $n_f^{\text{true}} = 64$ and (b – e) truncated modes $n_f = \{32, 20, 16, 14\}$, and (f) fnergy spectrum $E(\kappa_j) = \frac{1}{5000}\sum_{k=1}^{5000} |c_j(k)|^2$, where $j = 1, 2, \ldots, n_f/2 - 1$.

for $j = 1, 2, \ldots, n_f$ so that the magnitude of the respective mode is preserved after the truncation. As the number of Fourier modes decreases, the solution diverges faster from the highest fidelity solution, i.e., the true environment solution. The plot of the energy spectra in Fig. 6.5 shows how much of the true energy spectrum each model captures. Lower wavenumber modes dominate the true environment's energy (shown in the zoomed-in portion of the spectrum). Whilst the dominance of these modes is captured by the lower order models ($n_f < 20$), the models' estimates of the energy of these modes show some error from the truth.

We set the truncation to $n_f = 16$ and analyse the DA-MIRL training and control performance when using this coarse-grained model. The training performance of the DA-MIRL is shown in Fig. 6.6 for varying number of sensors below the $n_o = 8$ sensors limit we found for the model-free RL in § 6.5. We compare the episode returns of the true environment during training against the

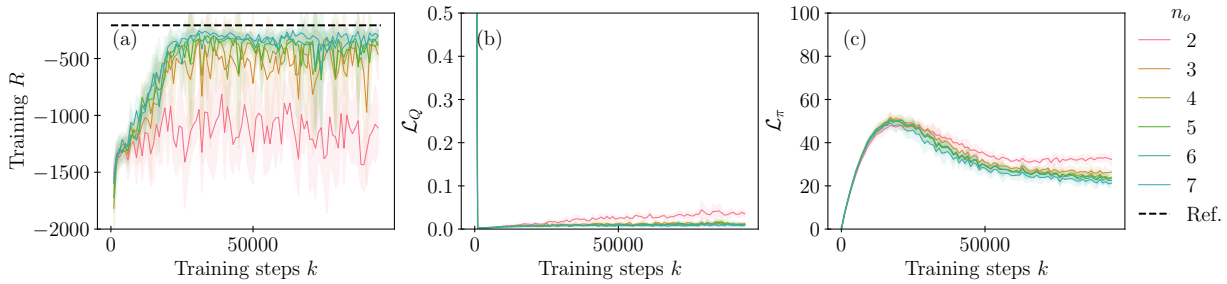


Figure 6.6: Truncated Fourier model: Training performance for varying number of sensors n_o. Mean and standard deviation over 5 runs of (a) return R, (b) critic loss L_Q, and (c) policy loss L_π. Ref. indicates the maximum return achieved by the model-free algorithm with noise-free full-observability, i.e., $n_o = 64$ in Fig. 6.4 (a).

maximum return achieved by the model-free agent when using the noise-free full state information $n_o = n_f^{\text{true}} = 64$. We find that the DA-MIRL stabilises the system and achieves comparable episodes returns with down to $n_o = 3$ sensors. This shows that low-order models with fewer degrees of freedom can be employed to learn stabilising control policies via RL when updated with partial, noisy observations using sequential DA. With a low number of sensors, e.g., $n_o \leq 2$, the provided information is not enough to synchronise the model with the environment and infer the correct state, particularly after the learning starts and the system is driven away from the attractor towards the zero solution. Nonetheless, the actor-critic losses converge with low variance across n_o, compared to the model-free RL results, demonstrating the DA-MIRL robustness to partial observability.

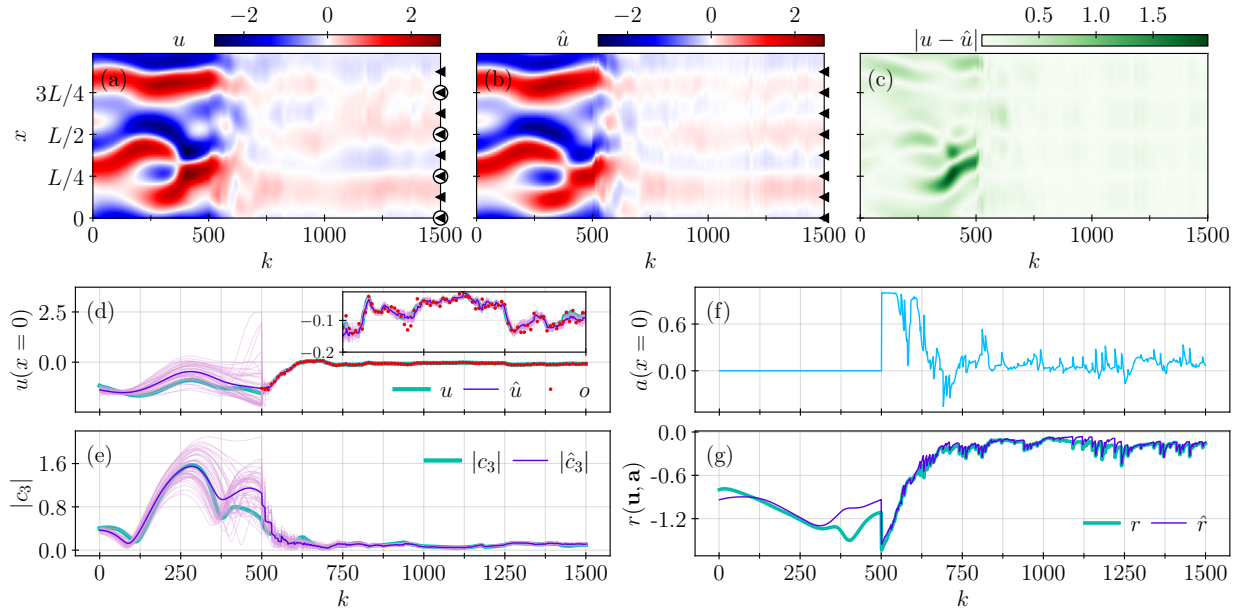


Figure 6.7: Truncated Fourier model: control performance of the DA-MIRL with $n_f = 16$. Shown in (a – c) spatiotemporal evolution of the environment (the truth), the model (ensemble mean), and the reconstruction error. The triangles and circles indicate the locations of the $n_a = 8$ actuators and $n_o = 4$ sensors, respectively. Shown in (d) and (e) the ensemble forecast (pink), ensemble mean (purple) and true (teal) velocity at $x = 0$ and magnitude of the third Fourier coefficient. Shown in (f) the agent's action at $x = 0$, and (g) the reward function of the environment and the model. The observations and actuations begin at $k = 500$.

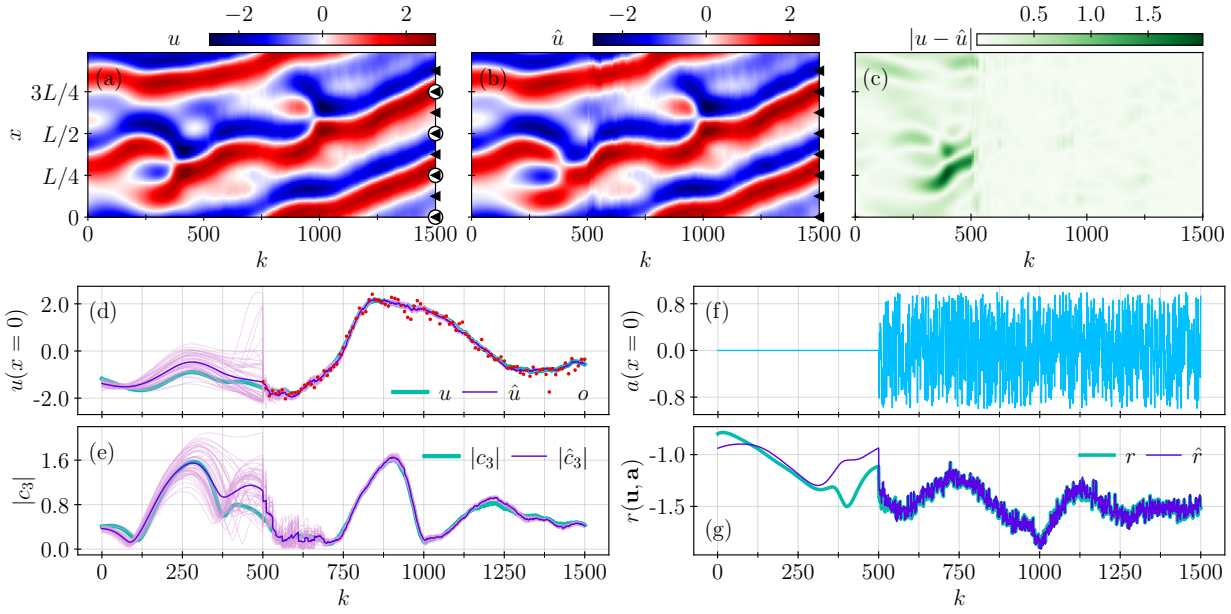


Figure 6.8: State estimation performance of the truncated Fourier model with $n_f = 16$ with random actuations. (a – g) same as Fig. 6.7.

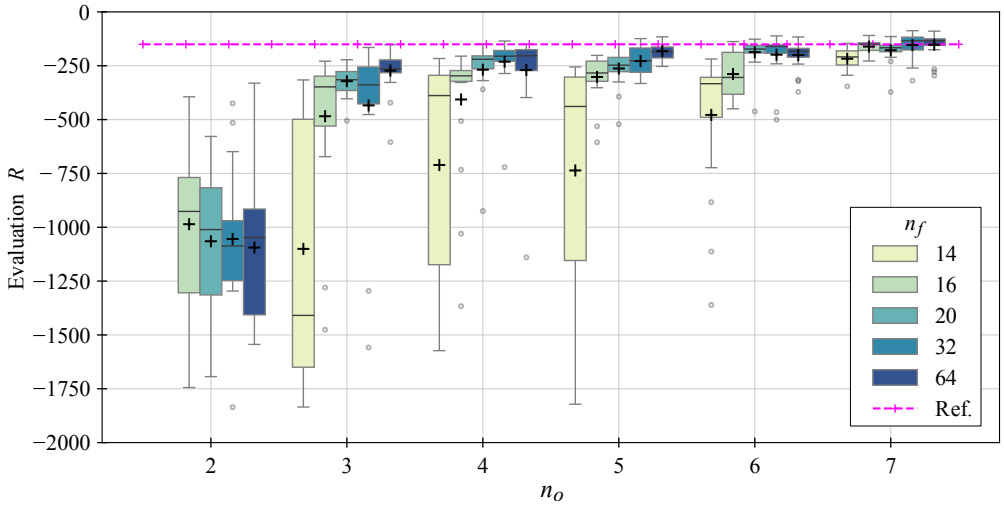


Figure 6.9: Truncated Fourier model: DA-MIRL performance 20 evaluation episodes for varying number of sensors n_o and number of Fourier modes n_f. (The model with $n_f = 14$ is numerically unstable with $n_o = 2$, hence it is not shown.)

An evaluation episode after training is visualised in Fig. 6.7 for the case of $n_o = 4$ sensors. The spatiotemporal plots in Fig. 6.7 (a – c) show how the model estimate of the environment remains accurate and rapidly converges to the stable solution when DA-MIRL algorithm starts at $k = 500$. The DA-MIRL successfully recovers the true state of the environment not only around the chaotic attactor (in comparison to the episode where we apply random actuations shown in Fig. 6.8 (a – c)) but also during the actuated transient. This is further illustrated in Fig. 6.7 (d, e), as the ensemble spread, i.e., the uncertainty of the model, decreases with each observation.

We assess the trained agent over 20 evaluation episodes for varying number of sensors, $n_o = \{2, 3, 4, 5, 6, 7\}$, and number of Fourier modes, $n_f = \{14, 16, 20, 32, 64\}$. The resulting statistics of the episode returns are shown in Fig. 6.9 and compared with the baseline obtained from the

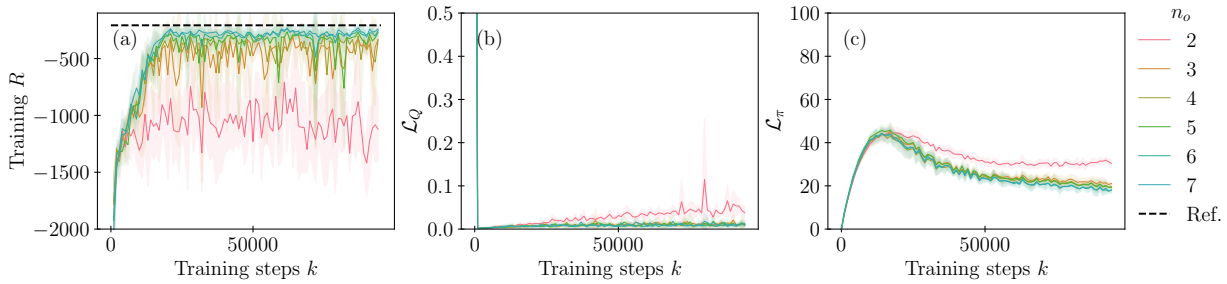


Figure 6.10: CA-ESN: DA-MIRL training for varying number of sensors n_o. Shown in (a) non-discounted returns of training episodes, (b) Q-loss, (c) policy loss across 5 runs (mean and 1 standard deviation in lighter colour). Ref. indicates the maximum return achieved by the model-free algorithm using the full-state observations, i.e., $n_o = n_f^{\text{true}} = 64$ in 6.4 (a).

model-free training with full state observability, i.e., $n_o = n_f^{\text{true}} = 64$. When the environment information is limited, i.e., fewer sensors, then the inaccuracies in the model have more impact in the overall performance. However, as the number of sensors is increased, the information from the environment corrects for the inaccuracy of the models with fewer Fourier modes resulting in similarly good performance across the models. Although lower-fidelity models intrinsically provide a limited representation of the environment, data assimilation compensates for this error, effectively bridging the gap between model predictions and environment.

6.6.2 Results of DA-MIRL with control-aware echo state network *(CA-ESN)*

The performance of the DA-MIRL using the data-driven modelling approach, i.e, the control-aware ESN detailed in § 6.3.2(b), is tested next. The control-aware ESN with a reservoir state size of $n_h = 1000$ is trained offline on the full state data, i.e., $\boldsymbol{u}^{\text{true}} \in \mathbb{R}^{64}$, generated by applying random actuations to the system. We generate 50 such simulations of length $k = 1500$ time steps with different initial conditions, where we use $N_{\text{wash}} = 100$ time steps for washout. The dataset is split into 80% of the simulations for training, 10% for the validation of hyperparameters, and 10% for testing. On the test set, the ESN achieves a mean relative ℓ_2 error (6.37) of approximately 3% over short trajectories of one Lyapunov time, indicating accurate short-time prediction capability. The optimal hyperparameters of the CA-ESN are provided in Tab. 6.1.

The training performance is shown in Fig. 6.10 for varying number of sensors below the $n_o = 8$ sensors limit we found for the model-free RL in § 6.5, where we achieve stabilisation down to $n_o = 3$ sensors. In reference to Fig. 6.6, we find that the data-driven model (control-aware ESN) matches the performance of the physics-based coarse-grained model (truncated Fourier model), which shows the efficacy of the DA-MIRL algorithm across different choices of models. The method is also robust to different configurations of sensors, and the results obtained with uniformly spaced sensors are representative. An evaluation episode after training is visualised in Fig. 6.11 for the case of $n_o = 4$ sensors. After an initial divergence of the ensemble, the reservoir state dynamics synchronise with the true dynamics as observations become available, and the DA-MIRL algorithm successfully stabilises the chaotic flow. The control-aware ESN accurately predicts the spatiotemporal evolution of the controlled KS equation. While the uncertainty of the estimate is higher at the unobserved locations, the reconstruction converges to the ground truth across the spatial domain (Fig. 6.11

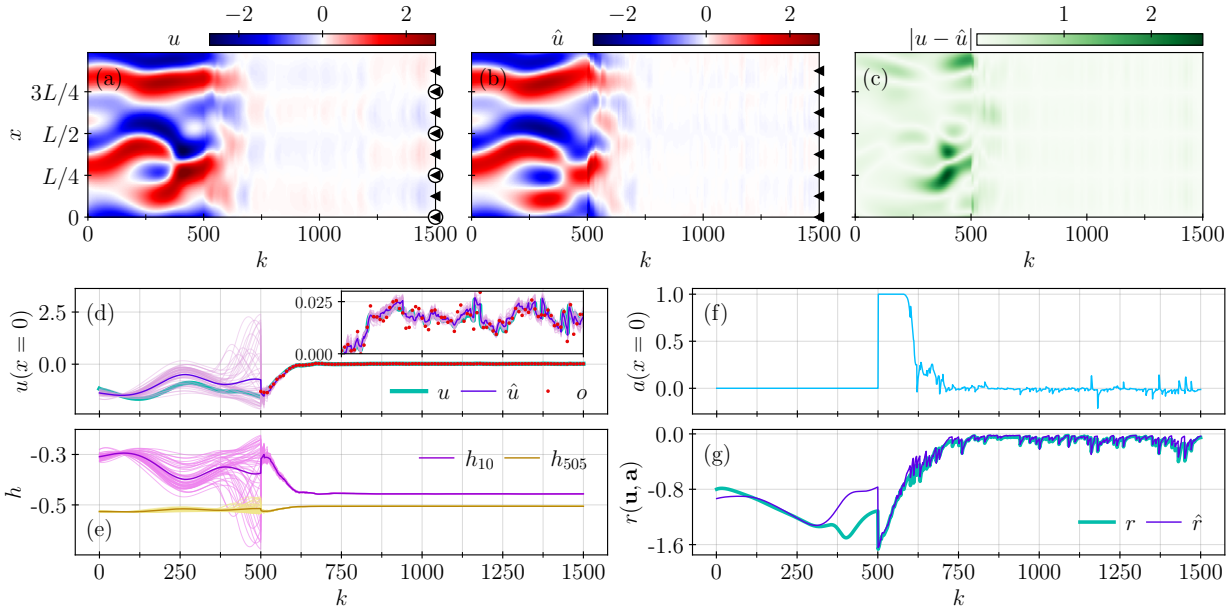


Figure 6.11: CA-ESN control performance of DA-MIRL with $n_h = 1000$. Shown in (a – c) the spatiotemporal evolution of the environment (the truth), the model (ensemble mean), and the reconstruction error. The triangles and circles indicate the locations of the $n_a = 8$ actuators and $n_o = 4$ sensors, respectively. Shown in (d) and (e) the ensemble forecast (pink), ensemble mean (purple) and true (teal) velocity at $x = 0$ and two reservoir states which take $u(x = 0)$ as input (no reference true value). Shown in (f) the agent's action at $x = 0$, and (g) the reward function of the environment and the model. The observations and actuations begin at $k = 500$.

(c)). Fig. 6.11 (e) also illustrates how the ESN's internal dynamics echo the observed environment: the evolution of two reservoir state ensembles, h_{10} and h_{505}, which take $u(x = 0)$ as input, are a transformation of the physical state $u(x = 0)$ (Fig. 6.11 (d)). The close overlap between the true reward and the model estimated reward (Fig. 6.11 (g)), quantitatively validates the proposed DA-MIRL framework's ability to maintain policy performance despite partial observability. For comparison of the uncontrolled performance, Fig. 6.12 visualises a random episode, where the actuations are sampled uniformly; in this case, the system is perturbed by random actuations near the chaotic attractor. This episode represents the conditions that the control-aware ESN has been trained on. Similar synchronisation of the reservoir dynamics as in the controlled case is observed with resulting low reconstruction error.

Similarly to the truncated Fourier model, we evaluate the trained agent over 20 evaluation episodes for varying number of sensors, $n_o = \{2, 3, 4, 5, 6, 7\}$, this time varying the reservoir sizes, $n_h = \{500, 1000, 3000\}$ to assess the impact of model complexity. The resulting statistics of the episode return are shown in Fig. 6.13 and compared with the baseline obtained from the model-free training with full state observability, i.e., $n_o = n_f^{\text{true}} = 64$. Whilst the performance increases with the addition of more sensors, it quickly saturates with increasing reservoir size. Ultimately, for the initial condition and experiment setup depicted in Figs. 6.7 and 6.11, the data-driven model with $n_h = 1000$ yields a higher episode return ($R = -226$) than the coarse-grained model with $n_f = 16$ ($R = -296$), with a smaller error from the target of zero state. However, their evaluation across different initial conditions (Figs. 6.9 and 6.13) indicate that their performances are comparable on average, where increasing the number of observations compensates for lack of model fidelity.

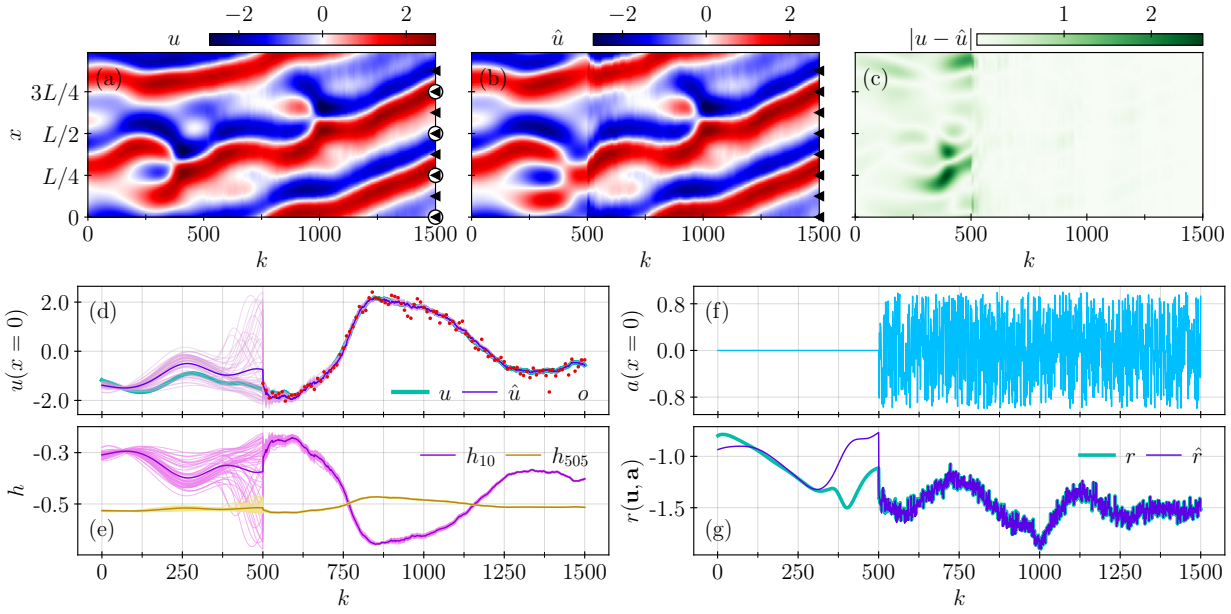


Figure 6.12: State estimation performance of the CA-ESN with $n_h = 1000$ in an episode with random actuations. (a–g) same as Fig. 6.11.

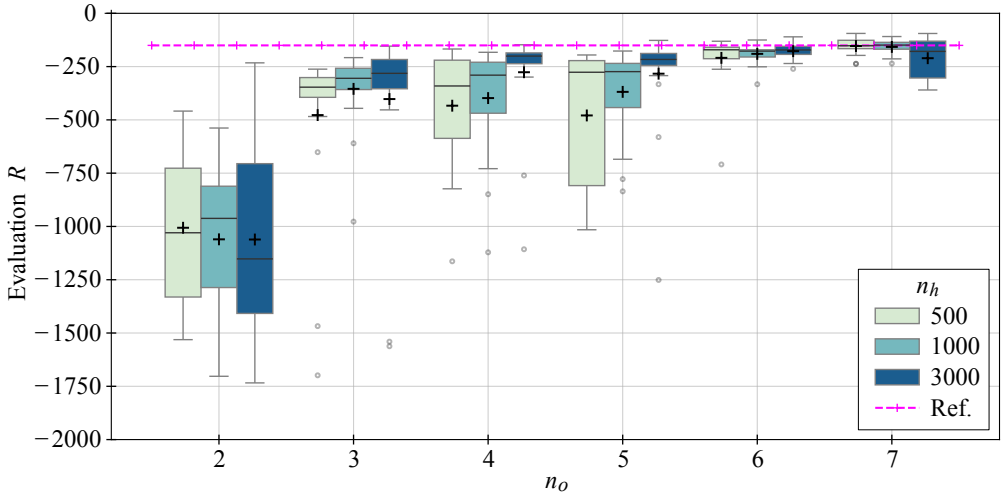


Figure 6.13: CA-ESN: DA-MIRL performance 20 evaluation episodes with varying number of sensors n_o and reservoir size n_h.

The forecasting capability of the ESN in closed-loop configuration (without the analysis step) is shown in Fig. 6.14 for a test case under different control scenarios, where the actuations are: (a) random, (b – d) given by the actor after $k = \{5, 10, 15\} \times 10^3$ learning steps, i.e., {5, 10, 15} episodes, and (e) given by the actor using the best actor weights, $\theta^{\pi\star}$, according to the evaluation. To generate the truth for this figure, we use the actor weights from the DA-MIRL training with $n_o = 4$ sensors. The system is initialised with the same initial condition as Figs. 6.12 (corresponds to Fig. 6.14 (a)) and Fig. 6.11 (corresponds to Fig. 6.14 (e)). We let it evolve for one LT without actuation, and record one episode using the actuations determined as (a – e). After generating this data, we perform the washout just before the actuations so that the ESN predicts the actuated system. The reconstruction error is smallest with random actuations because the control-aware ESN is trained on this regime (Fig. 6.14 (a)). As the actor learns to control the environment, the

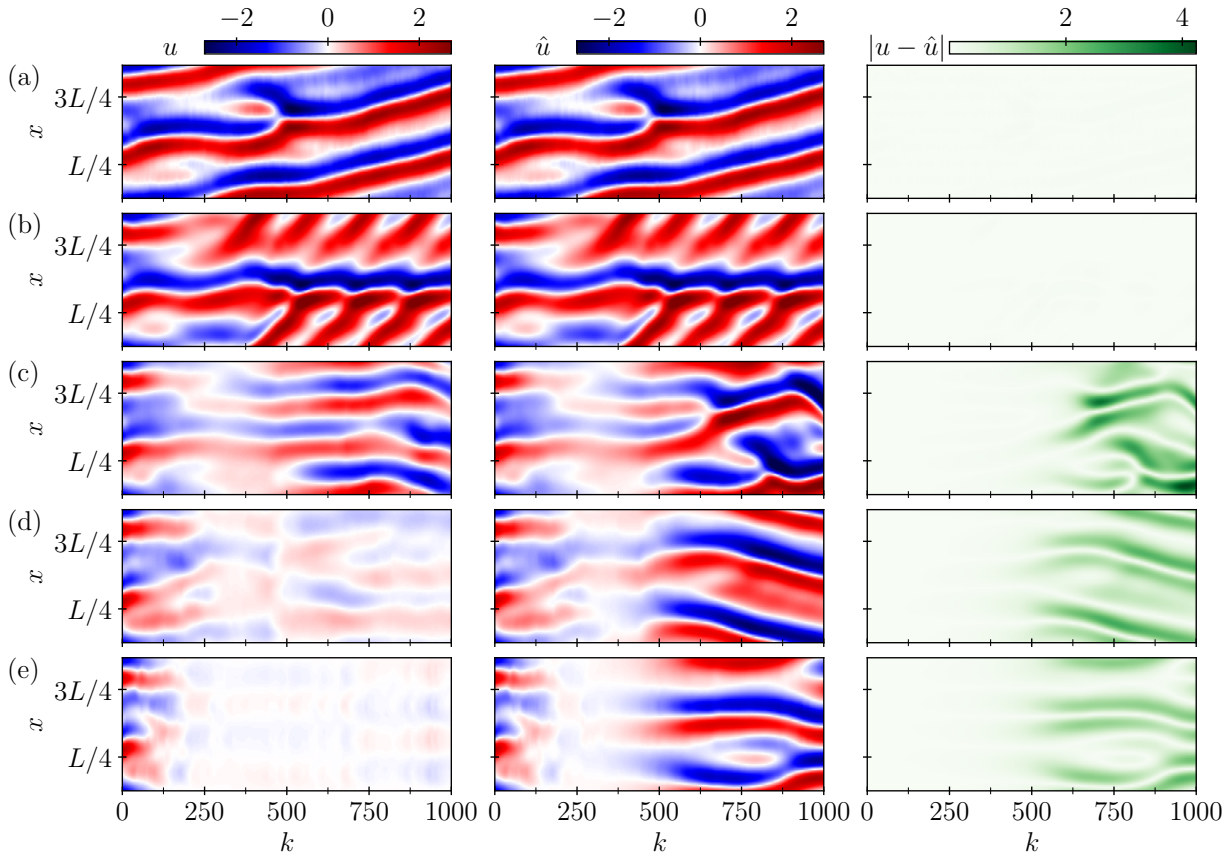


Figure 6.14: CA-ESN prediction in closed loop (without data assimilation) throughout RL training. Comparison between the environment state, the prediction of a control-aware ESN after a washout, and the mean reconstruction error. The actuations are selected (a) randomly (the condition in which the ESN has been trained offline), (b – d) by the actor after 5, 10, 15 learning episodes where the actor is trained, and (e) by the actor using the best actor weights, $\boldsymbol{\theta}^{\pi\star}$, based on the evaluation error. The actuations in (e) lead to stabilisation of the flow. The predictability horizon of the ESN decreases in this unseen regime. In the online DA-MIRL training, this deviation is corrected by the data assimilation.

control-aware ESN is exposed to increasingly unfamiliar regimes, particularly as the learned policy stabilises the flow. The reconstruction error remains small during the first $k = 500$ (one LT) for the controlled system with policies learned at different stages of the learning process (Fig. 6.14 (b – d)) as well as with the optimal control policy (Fig. 6.14 (e)). This validates the control-aware architecture proposed for the echo state network. The accurate prediction horizon decreases as the controller drives the system towards the unstable zero solution, where the dynamics have not been seen by the ESN. In the DA-MIRL, we overcome this limitation in the analysis step without explicitly retraining the ESN. In an experimental setting, this corresponds to training a cheap surrogate model offline based on experimental or high-fidelity simulation data, which is deployed for online reinforcement learning training with real-time observations without retraining. Integration of online learning methods for the ESN (e.g., Nóvoa and Magri, 2025) to improve its forecasting capability in unseen regimes is scope for future work.

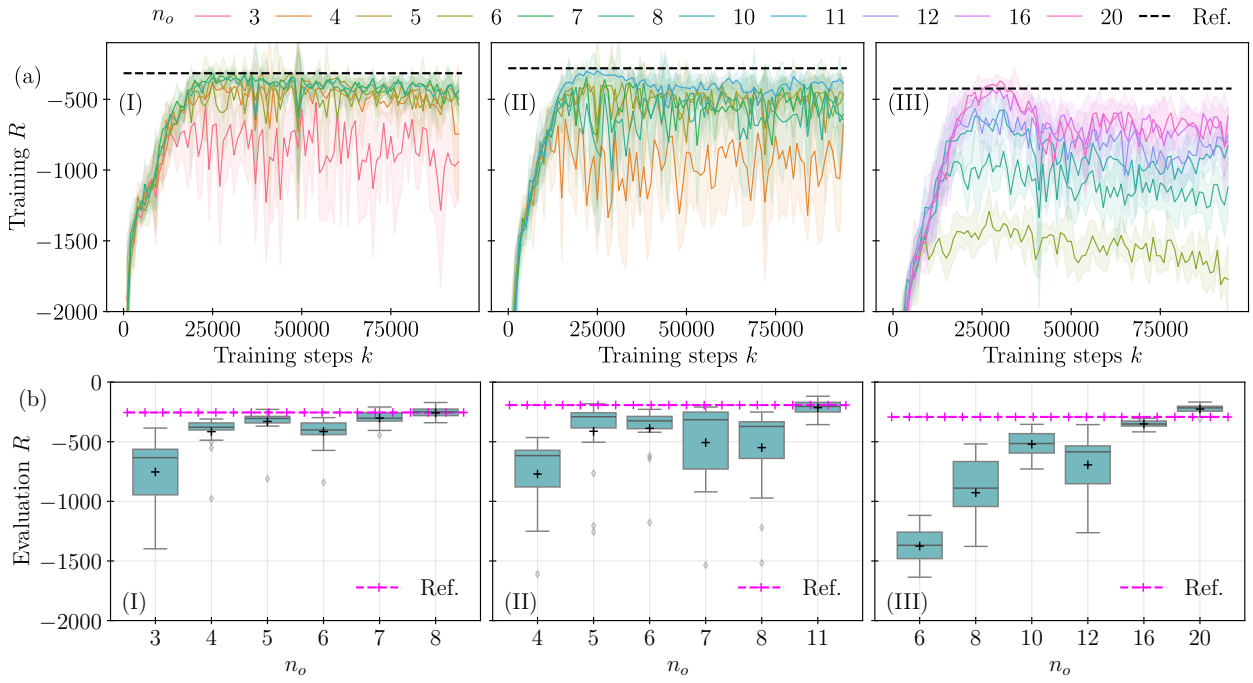


Figure 6.15: CA-ESN: DA-MIRL performance over increasingly chaotic regimes with viscosity (I) $\nu = 0.05$, (II) $\nu = 0.03$, (II) $\nu = 0.01$. Shown episode return in (a) evolution during training, and (b) statistics over 20 evaluation episodes.

6.6.3 Generalisation to more chaotic regimes

Lastly, we assess robustness to varying levels of chaos in the system. We evaluate the methodology in more chaotic regimes which are obtained by decreasing the viscosity parameter $\nu = \{0.05, 0.03, 0.01\}$ ($L \approx \{28, 36, 63\}$, respectively). We note that when not actuated $\nu = 0.05$ regime displays transient chaos before converging to a fixed point, whereas the other regimes are unstable and increasingly chaotic with $LT \approx \{250, 235\}$ time steps, which we use as our initial propagation time for the ensemble until the first observation. For the $\nu = 0.05$ regime we set this as 350 time steps. We determine the number of actuators as the number of sensors required for chaos synchronisation in the respective regime, which yield $n_a = \{9, 12, 21\}$, respectively. We find that as the flow becomes more chaotic and the dominant wavenumber increases, the number of actuators required for stabilisation also increase. Gomes et al. (2017) show that when using linear state-feedback control the number of point actuators (small σ_a) needed to control arbitrary steady states of the generalised KS equation is proportional to the number of unstable modes of the target solution. However, since the focus of this chapter is observability, we do not extensively explore the controllability as a function of the number or position of the actuators. We repeat the experiments using a control-aware ESN of reservoir size $n_h = 1000$ trained on data from each regime so as to show that it can capture more chaotic dynamics. As for the truncated Fourier model, the minimum number of Fourier modes required for the solver stability increases with the level of chaos, whereas we can use the same size ESN with similar performance. We find that the state estimation from low number of sensors is especially challenging for the most chaotic regime $\nu = 0.01$ after the flow starts stabilising. In this case, increasing the ensemble size and the inflation factor ($n_{\text{en}} = 200$ and $\rho_I = 1.05$) improves the state estimation. The results for varying number of sensors where we focus on sensors below the observable threshold are shown in Fig. 6.15. As the system exhibits increas-

ingly chaotic behaviour, state estimation and control tasks become more challenging, with a higher minimum number of sensors required for accurate reconstruction. However, DA-MIRL generalises well over the increasingly chaotic regimes, and is robust to partial and noisy observability across different dynamics.

6.7 Conclusion

Closed-loop control of chaotic flows is desirable for increasing the efficiency in many applications, but it is challenging due to the high dimensionality and chaotic dynamics. Furthermore, in real-life scenarios we only have access to sensor measurements, which are incomplete and noisy, to determine the control inputs. Reinforcement learning (RL) methods have been successfully applied for the control of chaotic flows, however they may fail under partially observed conditions. We propose a framework, data-assimilated model-informed reinforcement learning (DA-MIRL) which seamlessly integrates three components: (i) a predictive model of the system's dynamics, which can be physics-based or data-driven; here we propose a control-aware echo state network (CA-ESN), (ii) sequential data assimilation for real-time state estimation from observations; here we use an ensemble Kalman filter (EnKF), (iii) an off-policy actor-critic RL algorithm for learning optimal control policies given the state estimates; here we use deep deterministic policy gradient (DDPG).

We demonstrate the effectiveness of the DA-MIRL framework on the Kuramoto-Sivashinsky (KS) equation, which is a prototypical system with spatiotemporal chaos. The objective of control is to stabilise the flow around the zero solution, whilst spending the minimal actuation power. Focusing on scenarios in which the number of available sensors is limited, i.e., under partial observability, we find that the off-the-shelf model-free RL agent fails to learn a stabilising policy. In contrast, we test the DA-MIRL framework on two models: (i) a physics-based low-order model obtained by truncating the Fourier basis of the original dynamics, and (ii) a control-aware ESN trained offline on full state data generated by randomly actuating the system. Although the control-aware ESN is trained on the KS system with random actuations, the network generalises to the dynamics under different control strategies, where its forecast is corrected by the data assimilation step as observations become available. For both choices of approximate models, the DA-MIRL framework successfully stabilises the chaotic flow below the threshold of number of sensors required by the model-free agent, even when the measurements are noisy and not available at every time step. The framework is further validated on increasingly chaotic conditions, in which it shows robustness to noise and partial observability across a range of chaotic regimes.

This work opens up possibilities for RL-based control of chaotic systems from partial and noisy observations.

Chapter 7

Conclusion and Outlook

7.1 Conclusion

Hard-constrained neural networks for modelling nonlinear acoustics

In Chapter 4, we develop acoustic neural networks that model pressure and velocity oscillations in time and space from synthetic sensor measurements. These networks perform the tasks of (i) extrapolation in time, which is challenging for standard neural networks, and (ii) reconstruction of full acoustic state from partial and noisy observations. These tasks are achieved by incorporating prior physical knowledge by:

1. *Employing periodic activations (periodically activated neural networks, P-FNNs).* The choice of sinusoidal activations is physically motivated as the eigenfunctions of acoustic and thermoacoustic systems are Fourier modes, and this choice enables long-term extrapolation in periodic through quasiperiodic and chaotic regimes capturing the dominant frequency content. This architecture outperforms standard activations (such as tanh and ReLU) both in terms of accuracy and network size for thermoacoustic datasets.

2. *Physics-informed loss.* This term regularises the problem in the case of partial and noisy observations and outperforms other regularisation techniques such as ℓ_2 and ℓ_1. It enables reconstruction of full velocity fields from only partial pressure measurements, which is a typical set up of thermoacoustics experiments.

3. *Hard-constraining the architecture in a physically-motivated solution space (Galerkin neural networks, GalNNs).* These networks are composed of a temporal branch modelled by P-FNNs and a spatial branch spanned by the natural acoustic eigenmodes. Additional spatial modes successfully capture velocity discontinuities and nonideal boundary conditions. Imposing the solution structure narrows down the search space, improves the robustness in uncertain conditions and offers interpretability as a reduced-order model.

Data-driven computation of adjoint sensitivities without adjoint solvers

In Chapter 5, we propose a framework that infers adjoint sensitivities from data by utilising an echo state network (ESN). The ESN predicts temporal input-output relationships by dynamically

evolving a hidden state, i.e., the reservoir state. A parameter-aware extension to the ESN, PA-ESN, learns parameterised dynamics and generalises to unseen nonlinear regimes. We derive the adjoint of the PA-ESN, which infers the sensitivities of the original system to (i) the system's parameters, and (ii) the system's initial conditions. (As a byproduct the sensitivities to the trainable weights can also be computed.) The proposed framework applies to any dynamical system, and it is extended to chaotic systems by the ensemble adjoint method.

1. We demonstrate it on the Lorenz system, in which we compute the sensitivity of a time-averaged state variable, $\bar{z}_{\mathrm{L}}$, at varying parameter values to all parameters, σ_{L}, ρ_{L}, and β_{L}. The estimated sensitivities match closely with the ground-truth values obtained by integrating the adjoint of the governing equations.

2. We design a parameter-aware ESN for nonlinear, time-delayed thermoacoustic systems, T-ESN, by (i) making the ESN explicitly time-delayed, and (ii) designing the input matrix in consideration of acoustic physics. The derivation of the adjoint is extended to the time-delayed equations. The sensitivity of the acoustic energy to the flame parameters is computed and matches closely with the ground truth. We suppress a nonlinear oscillation by minimising the acoustic energy with steepest gradient-descent, in which the gradients are computed by only one T-ESN.

3. We perform a bifurcation analysis of the T-ESN and find that it infers the Hopf bifurcation, at which the system transitions from fixed point to limit cycle attractor, even though it has only been trained on limit cycle data. It also infers the route to chaos with bifurcations to period-doubling, quasiperiodic, and chaotic regimes. The extrapolation accuracy depends on the range of training parameters.

Data-assimilated model-informed reinforcement learning

In Chapter 6, we propose a framework that tackles partial observability in the real-time active feedback control of spatiotemporally chaotic systems. The framework, data-assimilated model-informed reinforcement learning (DA-MIRL), integrates:

(i) *Predictive, approximate models for forecasting.* Two options of predictive models are developed (i) a low-order coarse-grained model based on the Galerkin projection of the physical PDE, and (ii) a fully data-driven model based on a control-aware echo state network (CA-ESN), which is designed here for computationally efficient inference and forecasting of actuated dynamics. These models are time-marched and forecast the dynamics.

(ii) *Sequential data assimilation for state estimation.* The model predictions and the partial, noisy observations from the true system (environment) are fed to a state estimator. The chosen data assimilation method is the ensemble Kalman filter (EnKF), which extends the linear Kalman filter to nonlinear dynamics, and approximates the high-dimensional covariance matrix with a low-dimensional ensemble. The EnKF outputs the maximum a posteriori state estimate.

(iii) *Off-policy actor-critic reinforcement learning for optimal control.* The deep deterministic policy gradient (DDPG) discovers the optimal feedback control policy based on the state

estimates.

This modular framework overcomes the difficulties of end-to-end training of observer and controller, which is the case with model-free RL algorithms trained on partially observed data.

We show the methodology on the Kuramoto-Sivashinsky (KS) equation with the goal of stabilising the chaotic dynamics by actuating the system at multiple locations. First, we show that the performance of the model-free RL agent depends on (i) the number of sensors available, (ii) the placement of the sensors, (iii) the level of noise on the measurements, and (iv) the number of time steps between each observation. Second, we validate DA-MIRL with both choices of approximate models under these uncertain conditions and demonstrate that it discovers stabilising policies below the threshold of number of sensors required by the model-free agent, e.g., down to three sensors in contrast to eight in the weakly chaotic conditions, even when these measurements are noisy and sparse in time. Third, the framework is validated across increasing chaotic conditions demonstrating robustness. We find that because the CA-ESN is trained offline on randomly actuated data, its prediction horizon decreases as the agent learns to stabilise the flow and the system enters unseen dynamics. However, the EnKF corrections counteract the model error resulting in accurate reference tracking.

7.2 Outlook

This work opens up possibilities to develop SciML methods for data-driven modelling, reconstruction, sensitivity analysis, and control of nonlinear systems with spatiotemporal dynamics and multiphysics effects. Whilst the developed frameworks are relatively general, we show how they can be tailored to specific datasets of engineering interest, for example in thermoacoustics. A discussion of future research directions is provided here.

Scaling up to engineering systems. Although we have convincing evidence of the developed methods' validity on benchmark systems, it remains to scale them up to higher-dimensional and more complex systems, including turbulent flows such as the flow behind the bluff body problem, or more complex geometries such as cross-section changes in the Rijke tube.

Extensions to GalNNs. In the GalNN, we have fixed the spatial modes; however, the framework also allows for learning them at the same time, where one can consider adding trainable weights in the spatial branch. The physical basis can be considered as a good starting point for the training. Furthermore, GalNNs impose a linear model decomposition. The decomposition can be made nonlinear by adding a nonlinear activation layer, e.g., tanh, before the output layer. Another option is to learn a closure term that is a standard FNN or a P-FNN.

Extensions to PA-ESN. Fukami and Taira (2023) show that the dynamics of extreme aerodynamics from different parameter regimes are captured in the latent space of an autoencoder trained on data from all of these regimes. For high-dimensional systems, the PA-ESN can be combined with an autoencoder. Specifically, a convolutional autoencoder-ESN approach has been applied for turbulent flows (Racca et al., 2023; Özalp and Magri, 2025); the parameter-aware training can be

a natural extension. This approach can also be useful for mapping a high-dimensional parameter space to a lower dimension.

Extensions to the data-driven adjoint framework. Currently, the data-driven adjoint framework is written in `NumPy`/`SciPy`. Whilst this is efficient for the propagation of the ESN, it may not be efficient for the Jacobian computation which requires the multiplication for two $n_h \times n_h$ matrices, where n_h is the reservoir size, which is typically large. Code optimisation possibly utilising the `JAX`-GPU capabilities should be considered.

Extensions to DA-MIRL. In DA-MIRL, we choose the off-the-shelf DDPG and EnKF algorithms to demonstrate the methodology without additional algorithmic complexity. Alternative RL or ensemble DA algorithms (e.g., Soft Actor-Critic or square-root filters) can be implemented for improving the performance whilst keeping the same theoretical background. In the training, we only used the mean of the state ensemble, the covariance of the ensemble can also used to inform the RL training; possibly synergising with stochastic policies. Online learning or hybrid approaches for the system modelling can be integrated in the framework (e.g., Hafner et al., 2019; Nóvoa et al., 2024b; Nóvoa and Magri, 2025).

Learning from partial observations. When learning from experimental data, most of the time we only have access to partial observations of the state. The training of both RNNs for time series forecasting, and RL agents for control is challenging on partially observable datasets for off-the-shelf methods.

- In Chapter 5, we trained the PA-ESN on the full state data, i.e., the Galerkin amplitudes. The PA-ESN and subsequently the adjoint sensivity framework based on that should be assessed in partially observable conditions, for example on pressure measurements in the Rijke tube, to emulate experimental setups.
- For both RNN and RL tasks, formalising an observability condition for learning chaotic systems from data would be useful. (For example, in Chapter 6, we observed that the model-free RL agent could stabilise the system with down to eight sensors.) This will extend the observability Gramian in linear control theory for learning tasks.

Inference of bifurcations. In Chapter 5, we showed that the ESN successfully infers bifurcations. A key question is how to choose the training regimes for this purpose. This can be addressed with linear stability analysis around the fixed points and generalised with the Lyapunov exponent analysis (e.g., Huhn and Magri, 2020). The stability analysis of the ESN trained in a single regime has been performed and has revealed that it accurately captures the Lyapunov exponent spectra (Pathak et al., 2017; Margazoglou and Magri, 2023a; Özalp and Magri, 2025). This analysis can be extended to the PA-ESN over the parameterised dynamics to give insights about the extrapolation capability of the PA-ESN beyond the training regime.

Model predictive control with the CA-ESN. The adjoint of the PA-ESN is derived in Chapter 5. This framework can be readily extended to the CA-ESN for control purposes. Specifically,

the adjoint of the CA-ESN can be utilised in a model predictive control routine, which finds the optimal control trajectory over a receding horizon. A similar control application without the adjoint was proposed in Racca and Magri (2023).

Combining data-driven adjoints with gradient-free/model-free methods. The data-driven adjoint framework can be compared with gradient-free/model-free methods such as Bayesian optimisation or RL. Combined hybrid methodologies (e.g., Li and Zhang, 2022; Schena et al., 2024) are likely to outperform either approach.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

Aguilar, J. G. (2021). Flame Transfer Functions - Theory, analysis and application. https://xoeg.github.io/Flame-Transfer-Function-Tutorial/.

Aguilar, J. G. and Juniper, M. P. (2018). Adjoint Methods for Elimination of Thermoacoustic Oscillations in a Model Annular Combustor via Small Geometry Modifications. In *Volume 4A: Combustion, Fuels, and Emissions*, page V04AT04A054, Oslo, Norway. American Society of Mechanical Engineers.

Aguilar Perez, J. G. (2018). *Sensitivity Analysis and Optimization in Low Order Thermoacoustic Models*. PhD thesis, Apollo - University of Cambridge Repository.

Anderson, J. L. (2001). An Ensemble Adjustment Kalman Filter for Data Assimilation. *Monthly Weather Review*.

Åström, K. J. and Murray, R. M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton.

Baisero, A., Daley, B., and Amato, C. (2022). Asymmetric DQN for partially observable reinforcement learning. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 107–117. PMLR.

Balasubramanian, K. and Sujith, R. I. (2008). Thermoacoustic instability in a Rijke tube: Non-normality and nonlinearity. *Physics of Fluids*, 20(4):044103.

Bauerheim, M., Parmentier, J.-F., Salas, P., Nicoud, F., and Poinsot, T. (2014). An analytical model for azimuthal thermoacoustic modes in an annular chamber fed by an annular plenum. *Combustion and Flame*, 161(5):1374–1389.

Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control*. Athena Scientific Optimization and Computation Series. Athena scientific, Belmont (Mass.), 3e ed edition.

Bewley, T. R. (2001). Flow control: New challenges for a new Renaissance. *Progress in Aerospace Sciences*, 37(1):21–58.

Bewley, T. R., Moin, P., and Temam, R. (2001). DNS-based predictive control of turbulence: An optimal benchmark for feedback algorithms. *Journal of Fluid Mechanics*, 447:179–225.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: Composable transformations of Python+NumPy programs.

Brunton, S. L. and Noack, B. R. (2015). Closed-Loop Turbulence Control: Progress and Challenges. *Applied Mechanics Reviews*, 67(5):050801.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. (2020). Machine Learning for Fluid Mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937.

Bucci, M. A., Semeraro, O., Allauzen, A., Wisniewski, G., Cordier, L., and Mathelin, L. (2019). Control of chaotic systems by deep reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2231):20190351.

Cai, S., Wang, Z., Fuest, F., Jeon, Y. J., Gray, C., and Karniadakis, G. E. (2021). Flow over an espresso cup: Inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915:A102.

Callaham, J. L., Maeda, K., and Brunton, S. L. (2019). Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10):103907.

Camarri, S. (2015). Flow control design inspired by linear stability analysis. *Acta Mechanica*, 226(4):979–1010.

Capecelatro, J., Bodony, D. J., and Freund, J. B. (2019). Adjoint-based sensitivity and ignition threshold mapping in a turbulent mixing layer. *Combustion Theory and Modelling*, 23(1):147–179.

Chandramoorthy, N., Fernandez, P., Talnikar, C., and Wang, Q. (2017). An Analysis of the Ensemble Adjoint Approach to Sensitivity Analysis in Chaotic Systems. In *23rd AIAA Computational Fluid Dynamics Conference*, Denver, Colorado. American Institute of Aeronautics and Astronautics.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Chen, Y., Huang, D., Zhang, D., Zeng, J., Wang, N., Zhang, H., and Yan, J. (2021). Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method. *Journal of Computational Physics*, 445:110624.

Cheng, S., Quilodrán-Casas, C., Ouala, S., Farchi, A., Liu, C., Tandeo, P., Fablet, R., Lucor, D., Iooss, B., Brajard, J., Xiao, D., Janjic, T., Ding, W., Guo, Y., Carrassi, A., Bocquet, M., and Arcucci, R. (2023). Machine Learning With Data Assimilation and Uncertainty Quantification for Dynamical Systems: A Review. *IEEE/CAA Journal of Automatica Sinica*, 10(6):1361–1387.

Cho, K., Van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Culick, F. and Kuentzmann, P. (2006). Unsteady Motions in Combustion Chambers for Propulsion Systems. *NATO RTO-AG-AVT-039, AGARDograph*.

Cvitanović, P., Davidchack, R. L., and Siminos, E. (2010). On the State Space Geometry of the Kuramoto–Sivashinsky Flow in a Periodic Domain. *SIAM Journal on Applied Dynamical Systems*, 9(1):1–33.

Dee, D. P. (2005). Bias and data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 131(613):3323–3343.

Doan, N., Polifke, W., and Magri, L. (2020a). Physics-informed echo state networks. *Journal of Computational Science*, 47:101237.

Doan, N. A. K., Polifke, W., and Magri, L. (2020b). Learning Hidden States in a Chaotic System: A Physics-Informed Echo State Network Approach. In Krzhizhanovskaya, V. V., Závodszky, G., Lees, M. H., Dongarra, J. J., Sloot, P. M. A., Brissos, S., and Teixeira, J., editors, *Computational Science – ICCS 2020*, volume 12142, pages 117–123. Springer International Publishing, Cham.

Doan, N. A. K., Polifke, W., and Magri, L. (2021). Short- and long-term predictions of chaotic flows and extreme events: A physics-constrained reservoir computing approach. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2253):20210135.

Dong, S. and Ni, N. (2021). A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, 435:110242.

Dowling, A. P. (1999). A kinematic model of a ducted flame. *Journal of Fluid Mechanics*, 394:51–72.

Dowling, A. P. and Morgans, A. S. (2005). Feedback control of combustion oscillations. *Annual Review of Fluid Mechanics*, 37(1):151–182.

Dowling, A. P. and Stow, S. R. (2003). Acoustic Analysis of Gas Turbine Combustors. *Journal of Propulsion and Power*, 19(5):751–764.

Duan, X.-Y., Ying, X., Leng, S.-Y., Kurths, J., Lin, W., and Ma, H.-F. (2023). Embedding theory of reservoir computing and reducing reservoir network using time delays. *Physical Review Research*, 5(2):L022041.

Duraisamy, K., Iaccarino, G., and Xiao, H. (2019). Turbulence Modeling in the Age of Data. *Annual Review of Fluid Mechanics*, 51(1):357–377.

Eckmann, J. P. and Ruelle, D. (1985). Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, 57(3):617–656.

Eivazi, H., Wang, Y., and Vinuesa, R. (2024). Physics-informed deep-learning applications to experimental fluid mechanics. *Measurement Science and Technology*, 35(7):075303.

Evensen, G. (2009). *Data Assimilation: The Ensemble Kalman Filter*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Eyink, G. L., Haine, T. W. N., and Lea, D. J. (2004). Ruelle's linear response formula, ensemble adjoint schemes and Lévy flights. *Nonlinearity*, 17(5):1867–1889.

Fathi, M. F., Perez-Raya, I., Baghaie, A., Berg, P., Janiga, G., Arzani, A., and D'Souza, R. M. (2020). Super-resolution and denoising of 4D-Flow MRI using physics-Informed deep neural nets. *Computer Methods and Programs in Biomedicine*, 197:105729.

Foures, D. P. G., Caulfield, C. P., and Schmid, P. J. (2014). Optimal mixing in two-dimensional plane Poiseuille flow at finite Péclet number. *Journal of Fluid Mechanics*, 748:241–277.

Fukami, K., Fukagata, K., and Taira, K. (2021a). Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909.

Fukami, K., Hasegawa, K., Nakamura, T., Morimoto, M., and Fukagata, K. (2021b). Model Order Reduction with Neural Networks: Application to Laminar and Turbulent Flows. *SN Computer Science*, 2(6):467.

Fukami, K., Nakamura, T., and Fukagata, K. (2020). Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. *Physics of Fluids*, 32(9):095110.

Fukami, K. and Taira, K. (2023). Grasping extreme aerodynamics on a low-dimensional manifold. *Nature Communications*, 14(1).

Gad-el-Hak, M. (2000). *Flow Control: Passive, Active, and Reactive Flow Management*. Cambridge University Press, 1 edition.

Gao, H., Sun, L., and Wang, J.-X. (2021). Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids*, 33(7):073603.

Gelbert, G., Moeck, J. P., Paschereit, C. O., and King, R. (2012). Feedback control of unstable thermoacoustic modes in an annular Rijke tube. *Control Engineering Practice*, 20(8):770–782.

Geneva, N. and Zabaras, N. (2022). Transformers for modeling physical systems. *Neural Networks*, 146:272–289.

Giannetti, F. and Luchini, P. (2007). Structural sensitivity of the first instability of the cylinder wake. *Journal of Fluid Mechanics*, 581:167–197.

Giles, M. B. and Pierce, N. A. (2000). An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion*, 65(3/4):393–415.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Gomes, S. N., Papageorgiou, D. T., and Pavliotis, G. A. (2017). Stabilizing non-trivial solutions of the generalized Kuramoto–Sivashinsky equation using feedback and optimal control: Lighthill–Thwaites Prize. *IMA Journal of Applied Mathematics*, 82(1):158–194.

Gouasmi, A., Parish, E. J., and Duraisamy, K. (2017). *A Priori* estimation of memory effects in reduced-order models of nonlinear systems using the Mori–Zwanzig formalism. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170385.

Goyal, A. and Bengio, Y. (2022). Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478(2266):20210068.

Griewank, A. and Walther, A. (2000). Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software*, 26(1):19–45.

Grigoryeva, L. and Ortega, J.-P. (2018). Echo state networks are universal. *Neural Networks*, 108:495–508.

Guastoni, L., Rabault, J., Schlatter, P., Azizpour, H., and Vinuesa, R. (2023). Deep reinforcement learning for turbulent drag reduction in channel flows. *The European Physical Journal E*, 46(4):27.

Gunzburger, M. D. (2002). *Perspectives in Flow Control and Optimization*. Society for Industrial and Applied Mathematics.

Gupta, V., Chen, Y., and Wan, M. (2024). Predictability of weakly turbulent systems from spatially sparse observations using data assimilation and machine learning. *arXiv:2407.10088*.

Haeringer, M., Merk, M., and Polifke, W. (2019). Inclusion of higher harmonics in the flame describing function for predicting limit cycles of self-excited combustion instabilities. *Proceedings of the Combustion Institute*, 37(4):5255–5262.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2019). Dream to Control: Learning Behaviors by Latent Imagination.

Hammoud, M. A. E. R., Raboudi, N., Titi, E. S., Knio, O., and Hoteit, I. (2024). Data Assimilation in Chaotic Systems Using Deep Reinforcement Learning. *Journal of Advances in Modeling Earth Systems*, 16(8):e2023MS004178.

Hart, A., Hook, J., and Dawes, J. (2020). Embedding and approximation theorems for echo state networks. *Neural Networks*, 128:234–247.

Hart, A. G., Olding, K. R., Cox, A. M. G., Isupova, O., and Dawes, J. H. P. (2021). Using Echo State Networks to Approximate Value Functions for Control. *arXiv:2102.06258*.

Hasegawa, K., Fukami, K., Murata, T., and Fukagata, K. (2020). CNN-LSTM based reduced order

modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers. *Fluid Dynamics Research*, 52(6):065501.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Santiago, Chile. IEEE.

Heckl, M. A. (1990). Nonlinear Acoustic Effects in the Rijke Tube. *Acustica*, 72(1):63–71.

Heidenreich, H. S., Vlachas, P. R., and Koumoutsakos, P. (2024). Deconstructing Recurrence, Attention, and Gating: Investigating the transferability of Transformers and Gated Recurrent Neural Networks in forecasting of dynamical systems. *arXiv:2410.02654*.

Heyder, F. and Schumacher, J. (2021). Echo state network for two-dimensional turbulent moist Rayleigh-B\'enard convection. *Physical Review E*, 103(5):053107.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Holmes, P., Lumley, J. L., and Berkooz, G. (1996). *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 1 edition.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G. M. (2022). CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, 23(274):1–18.

Huhn, F. and Magri, L. (2020). Stability, sensitivity and optimisation of chaotic acoustic oscillations. *Journal of Fluid Mechanics*, 882:A24.

Huhn, F. and Magri, L. (2022). Gradient-free optimization of chaotic acoustics with reservoir computing. *Physical Review Fluids*, 7(1):014402.

Jaeger, H. and Haas, H. (2004). Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667):78–80.

Jaensch, S. and Polifke, W. (2017). Uncertainty encountered when modelling self-excited thermoacoustic oscillations with artificial neural networks. *International Journal of Spray and Combustion Dynamics*, 9(4):367–379.

Jameson, A. (1988). Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260.

Jameson, A. (1999). Re-Engineering the Design Process Through Computation. *Journal of Aircraft*, 36(1):36–50.

Jameson, A., Martinelli, L., and Pierce, N. (1998). Optimum Aerodynamic Design Using the Navier-Stokes Equations. *Theoretical and Computational Fluid Dynamics*, 10(1-4):213–237.

Joslin, R. D., Miller, D. N., and American Institute of Aeronautics and Astronautics, editors (2009). *Fundamentals and Applications of Modern Flow Control*. Number v. 231 in Progress in Astronautics and Aeronautics. American Institute of Aeronautics and Astronautics, Reston, Va.

Juniper, M. P. (2011). Triggering in the horizontal Rijke tube: Non-normality, transient growth and bypass transition. *Journal of Fluid Mechanics*, 667:272–308.

Juniper, M. P. and Sujith, R. (2018). Sensitivity and Nonlinearity of Thermoacoustic Oscillations. *Annual Review of Fluid Mechanics*, 50(1):661–689.

Kabiraj, L., Sujith, R. I., and Wahi, P. (2012). Bifurcations of Self-Excited Ducted Laminar Premixed Flames. *Journal of Engineering for Gas Turbines and Power*, 134(3):031502.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45.

Kalogirou, A., Keaveny, E. E., and Papageorgiou, D. T. (2015). An in-depth numerical study of the two-dimensional Kuramoto–Sivashinsky equation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179):20140932.

Kar, S. K. (2006). A Semi-Implicit Runge–Kutta Time-Difference Scheme for the Two-Dimensional Shallow-Water Equations. *Monthly Weather Review*, 134(10):2916–2926.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440.

Kelshaw, D. and Magri, L. (2024). Physics-constrained convolutional neural networks for inverse problems in spatiotemporal partial differential equations. *Data-Centric Engineering*, 5:e43.

Kim, J. and Bewley, T. R. (2007). A Linear Systems Approach to Flow Control. *Annual Review of Fluid Mechanics*, 39(1):383–417.

Kissas, G., Yang, Y., Hwuang, E., Witschey, W. R., Detre, J. A., and Perdikaris, P. (2020). Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118.

Krishnamurthy, V. (2016). *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 1 edition.

Kuramoto, Y. (1978). Diffusion-Induced Chaos in Reaction Systems. *Progress of Theoretical Physics Supplement*, 64:346–367.

Lagaris, I., Likas, A., and Fotiadis, D. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000.

Landau, L. D., Lifšic, E. M., Sykes, J. B., and Reid, W. H. (1987). *Fluid Mechanics*. Number 6 in Course of Theoretical Physics. Pergamon press, Oxford, 2nd ed edition.

Lea, D. J., Allen, M. R., and Haine, T. W. N. (2000). Sensitivity analysis of the climate of a chaotic system. *Tellus A*, 52(5):523–532.

Levine, H. and Schwinger, J. (1948). On the Radiation of Sound from an Unflanged Circular Pipe. *Physical Review*, 73(4):383–406.

Li, J. and Morgans, A. S. (2015). Time domain simulations of nonlinear thermoacoustic behaviour in a simple combustor using a wave-based approach. *Journal of Sound and Vibration*, 346:345–360.

Li, J. and Zhang, M. (2022). Reinforcement-learning-based control of confined cylinder wakes with stability analyses. *Journal of Fluid Mechanics*, 932.

Liao, P., Song, W., Du, P., and Zhao, H. (2021). Multi-fidelity convolutional neural network surrogate model for aerodynamic optimization based on transfer learning. *Physics of Fluids*, 33(12).

Lieuwen, T. C. and Yang, V. (2006). *Combustion Instabilities In Gas Turbine Engines: Operational Experience, Fundamental Mechanisms, and Modeling*. American Institute of Aeronautics and Astronautics, Reston ,VA.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, {ICLR} 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. arXiv.

Ling, J., Kurzawski, A., and Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166.

Linot, A. J., Zeng, K., and Graham, M. D. (2023). Turbulence control in plane Couette flow using low-dimensional neural ODE-based models and deep reinforcement learning. *International Journal of Heat and Fluid Flow*, 101:109139.

Liu, X.-Y. and Wang, J.-X. (2021). Physics-informed Dyna-style model-based deep reinforcement learning for dynamic control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2255):20210618.

Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141.

Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., and Ott, E. (2017). Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4):041102.

Luchini, P. and Bottaro, A. (2014). Adjoint Equations in Stability Analysis. *Annual Review of Fluid Mechanics*, 46(1):493–517.

Luenberger, D. (1971). An introduction to observers. *IEEE Transactions on Automatic Control*, 16(6):596–602.

Lukoševičius, M. (2012). A Practical Guide to Applying Echo State Networks. In Montavon, G., Orr, G. B., and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade*, volume 7700, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg.

Magri, L. (2019). Adjoint Methods as Design Tools in Thermoacoustics. *Applied Mechanics Reviews*, 71(2):020801.

Magri, L. (2020). Physics-constrained adaptive learning for multi-physics optimization. *ERC Grant No. 949388*.

Magri, L. (2023). Introduction to neural networks for engineering and computational science. *zenodo.org/doi/10.5281/zenodo.7655872*.

Magri, L., Bauerheim, M., and Juniper, M. P. (2016a). Stability analysis of thermo-acoustic nonlinear eigenproblems in annular combustors. Part I. Sensitivity. *Journal of Computational Physics*, 325:395–410.

Magri, L., Bauerheim, M., Nicoud, F., and Juniper, M. P. (2016b). Stability analysis of thermo-acoustic nonlinear eigenproblems in annular combustors. Part II. Uncertainty quantification. *Journal of Computational Physics*, 325:411–421.

Magri, L. and Juniper, M. P. (2013). Sensitivity analysis of a time-delayed thermo-acoustic system via an adjoint-based approach. *Journal of Fluid Mechanics*, 719:183–202.

Magri, L. and Juniper, M. P. (2014). Global modes, receptivity, and sensitivity analysis of diffusion flames coupled with duct acoustics. *Journal of Fluid Mechanics*, 752:237–265.

Magri, L., Schmid, P. J., and Moeck, J. P. (2023). Linear Flow Analysis Inspired by Mathematical Methods from Quantum Mechanics. *Annual Review of Fluid Mechanics*, 55(1):541–574.

Manneschi, L., Ellis, M. O. A., Gigante, G., Lin, A. C., Del Giudice, P., and Vasilaki, E. (2021). Exploiting Multiple Timescales in Hierarchical Echo State Networks. *Frontiers in Applied Mathematics and Statistics*, 6:616658.

Margazoglou, G. and Magri, L. (2023a). Data-Driven Stability Analysis of a Chaotic Time-Delayed System. In Mikyška, J., De Mulatier, C., Paszynski, M., Krzhizhanovskaya, V. V., Dongarra, J. J., and Sloot, P. M., editors, *Computational Science – ICCS 2023*, volume 10476, pages 406–413. Springer Nature Switzerland, Cham.

Margazoglou, G. and Magri, L. (2023b). Stability analysis of chaotic systems from data. *Nonlinear Dynamics*, 111(9):8799–8819.

Marino, F. and Guglieri, G. (2024). Beyond Static Obstacles: Integrating Kalman Filter with Reinforcement Learning for Drone Navigation. *Aerospace*, 11(5):395.

Matveev, K. I. (2003). *Thermoacoustic Instabilities in the Rijke Tube: Experiments and Modeling*. PhD thesis, California Institute of Technology.

Mensah, G. A., Magri, L., and Moeck, J. P. (2018). Methods for the Calculation of Thermoacoustic Stability Boundaries and Monte Carlo-Free Uncertainty Quantification. *Journal of Engineering for Gas Turbines and Power*, 140(061501).

Mensah, G. A. and Moeck, J. P. (2017). Acoustic Damper Placement and Tuning for Annular Combustors: An Adjoint-Based Optimization Study. *Journal of Engineering for Gas Turbines and Power*, 139(6):061501.

Mo, S., Zabaras, N., Shi, X., and Wu, J. (2019). Deep Autoregressive Neural Networks for High-Dimensional Inverse Problems in Groundwater Contaminant Source Identification. *Water Resources Research*, 55(5):3856–3881.

Mo, Y. and Magri, L. (2025). Reconstructing unsteady flows from sparse, noisy measurements with a physics-constrained convolutional neural network. *Physical Review Fluids*, 10(3):034901.

Mo, Y., Traverso, T., and Magri, L. (2024). Decoder decomposition for the analysis of the latent space of nonlinear autoencoders with wind-tunnel experimental data. *Data-Centric Engineering*, 5:e38.

Moeck, J. P., Paul, M., and Paschereit, C. O. (2010). Thermoacoustic Instabilities in an Annular Rijke Tube. In *ASME Turbo Expo 2010: Power for Land, Sea, and Air*, pages 1219–1232. American Society of Mechanical Engineers Digital Collection.

Mohammadi, B. and Mohammadi, B. (1997). Optimal shape design, reverse mode of automatic differentiation and turbulence. In *35th Aerospace Sciences Meeting and Exhibit*, Reno,NV,U.S.A. American Institute of Aeronautics and Astronautics.

Mohan, A. T., Lubbers, N., Chertkov, M., and Livescu, D. (2023). Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence. *Physical Review Fluids*, 8(1):014604.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. (2018). Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566, Brisbane, QLD. IEEE.

Noiray, N., Durox, D., Schuller, T., and Candel, S. (2008). A unified framework for nonlinear combustion instability analysis based on the flame describing function. *Journal of Fluid Mechanics*, 615:139–167.

Noiray, N. and Schuermans, B. (2012). Theoretical and experimental investigations on damper performance for suppression of thermoacoustic oscillations. *Journal of Sound and Vibration*, 331(12):2753–2763.

Nóvoa, A. (2024). Real-time bias-aware data assimilation. https://github.com/MagriLab/real-time-bias-aware-DA.

Nóvoa, A. and Magri, L. (2022). Real-time thermoacoustic data assimilation. *Journal of Fluid Mechanics*, 948:A35.

Nóvoa, A. and Magri, L. (2025). Online Model Learning with Data-Assimilated Reservoir Computers. In *Lecture Notes in Computer Science*, pages 57–64. Springer Nature Switzerland, Cham.

Nóvoa, A., Noiray, N., Dawson, J. R., and Magri, L. (2024a). A real-time digital twin of azimuthal thermoacoustic instabilities. *Journal of Fluid Mechanics*, 1001:A49.

Nóvoa, A., Racca, A., and Magri, L. (2023). Bias-aware thermoacoustic data assimilation. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(6):1924–1931.

Nóvoa, A., Racca, A., and Magri, L. (2024b). Inferring unknown unknowns: Regularized bias-aware ensemble Kalman filter. *Computer Methods in Applied Mechanics and Engineering*, 418:116502.

Olver, P. J. (2020). *Introduction to Partial Differential Equations*. Undergraduate Texts in Mathematics. Springer, Cham Heidelberg New York Dordrecht London, second corrected printing edition.

Orchini, A. and Juniper, M. P. (2016). Flame Double Input Describing Function analysis. *Combustion and Flame*, 171:87–102.

Orchini, A., Mensah, G. A., and Moeck, J. P. (2019). Effects of Nonlinear Modal Interactions on the Thermoacoustic Stability of Annular Combustors. *Journal of Engineering for Gas Turbines and Power*, 141(2):021002.

Orchini, A., Rigas, G., and Juniper, M. P. (2016). Weakly nonlinear analysis of thermoacoustic bifurcations in the Rijke tube. *Journal of Fluid Mechanics*, 805:523–550.

Ott, E. (2008). *Chaos in Dynamical Systems*. Cambridge Univ. Press, Cambridge, 2. ed., transf. in digital print edition.

Özalp, E. and Magri, L. (2025). Stability analysis of chaotic systems in latent spaces. *Nonlinear Dynamics*, 113(11):13791–13806.

Özalp, E., Margazoglou, G., and Magri, L. (2023). Reconstruction, forecasting, and stability of chaotic dynamics from partial data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(9):093107.

Ozan, D. E. (2023). Thermoacoustics datasets. *zenodo.org/doi/10.5281/zenodo.10021354*.

Ozan, D. E. and Magri, L. (2023a). Hard-constrained neural networks for modeling nonlinear acoustics. *Physical Review Fluids*, 8(10):103201.

Ozan, D. E. and Magri, L. (2023b). Physics-aware learning of nonlinear limit cycles and adjoint limit cycles. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(6):1191–1199.

Ozan, D. E. and Magri, L. (2024a). Adjoint Sensitivities of Chaotic Flows Without Adjoint Solvers: A Data-Driven Approach. In *Lecture Notes in Computer Science*, pages 345–352. Springer Nature Switzerland, Cham.

Ozan, D. E. and Magri, L. (2024b). Data-driven computation of adjoint sensitivities without adjoint solvers: An application to thermoacoustics. *Physical Review Fluids*, 9(10):103902.

Ozan, D. E., Nóvoa, A., and Magri, L. (2025a). Data-Assimilated Model-Based Reinforcement Learning for Partially Observed Chaotic Flows. In *Lecture Notes in Computer Science*, pages 65–72. Springer Nature Switzerland, Cham.

Ozan, D. E., Nóvoa, A., Rigas, G., and Magri, L. (2025b). Data-assimilated model-informed reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 481(2327):20250476.

Page, J. (2025). Super-resolution of turbulence with dynamics in the loss. *Journal of Fluid Mechanics*, 1002.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. (2018). Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Physical Review Letters*, 120(2):024102.

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., and Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102.

Peter, J. E. and Dwight, R. P. (2010). Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391.

Pironneau, O. (1974). On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1):97–110.

Poinsot, T. (2017). Prediction and control of combustion instabilities in real engines. *Proceedings of the Combustion Institute*, 36(1):1–28.

Qadri, U. A., Magri, L., Ihme, M., and Schmid, P. J. (2021). Using adjoint-based optimization to enhance ignition in non-premixed jets. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2245):20200472.

Rabault, J., Kuchta, M., Jensen, A., Réglade, U., and Cerardi, N. (2019). Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302.

Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Jimenez Rezende, D., Puigdomènech Badia, A., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., and Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Racca, A. (2022). Echo State Networks. https://github.com/alberacca/Echo-State-Networks.

Racca, A., Doan, N. A. K., and Magri, L. (2023). Predicting turbulent dynamics with the convolutional autoencoder echo state network. *Journal of Fluid Mechanics*, 975:A2.

Racca, A. and Magri, L. (2021). Robust Optimization and Validation of Echo State Networks for learning chaotic dynamics. *Neural Networks*, 142:252–268.

Racca, A. and Magri, L. (2023). Control-aware echo state networks (Ca-ESN) for the suppression of extreme events. *arXiv:2308.03095*.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. (2019). On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5301–5310. PMLR.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019a). 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.

Raissi, M., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E. (2019b). Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137.

Raissi, M., Yazdani, A., and Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030.

Rayleigh (1878). THE EXPLANATION OF CERTAIN ACOUSTICAL PHENOMENA 1. *Nature*, 18(455):319–321.

Raynaud, G., Houde, S., and Gosselin, F. P. (2022). ModalPINN: An extension of physics-informed Neural Networks with enforced truncated Fourier decomposition for periodic flow reconstruction using a limited number of imperfect sensors. *Journal of Computational Physics*, 464:111271.

Reddy, S. B., Magee, A. R., Jaiman, R. K., Liu, J., Xu, W., Choudhary, A., and Hussain, A. A. (2019). Reduced Order Model for Unsteady Fluid Flows via Recurrent Neural Networks. In *Volume 2: CFD and FSI*, page V002T08A007, Glasgow, Scotland, UK. American Society of Mechanical Engineers.

Renganathan, S. A., Maulik, R., and Ahuja, J. (2021). Enhanced data efficiency using deep neural networks and Gaussian processes for aerodynamic design optimization. *Aerospace Science and Technology*, 111:106522.

Rowley, C. W. and Dawson, S. T. (2017). Model Reduction for Flow Analysis and Control. *Annual Review of Fluid Mechanics*, 49(1):387–417.

Roy, M., Mandal, S., Hens, C., Prasad, A., Kuznetsov, N. V., and Dev Shrimali, M. (2022). Model-free prediction of multistability using echo state network. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(10):101104.

Sayadi, T., Le Chenadec, V., Schmid, P. J., Richecoeur, F., and Massot, M. (2014). Thermoacoustic instability – a dynamical system and time domain analysis. *Journal of Fluid Mechanics*, 753:448–471.

Schena, L., Marques, P. A., Poletti, R., Ahizi, S., Van Den Berghe, J., and Mendez, M. A. (2024). Reinforcement Twinning: From digital twins to model-based reinforcement learning. *Journal of Computational Science*, 82:102421.

Selimefendigil, F. and Polifke, W. (2011). A Nonlinear Frequency Domain Model for Limit Cycles in Thermoacoustic Systems with Modal Coupling. *International Journal of Spray and Combustion Dynamics*, 3(4):303–330.

Semeraro, O. (2024). Reinforcement Learning for Fluid Mechanics: An overview on Fundamentals from a Control Perspective. *hal-04794452v1f*.

Shawki, K. and Papadakis, G. (2020). Feedback control of chaotic systems using multiple shooting shadowing and application to Kuramoto–Sivashinsky equation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2240):20200322.

Shokar, I. J. S., Kerswell, R. R., and Haynes, P. H. (2024). Stochastic Latent Transformer: Efficient Modeling of Stochastically Forced Zonal Jets. *Journal of Advances in Modeling Earth Systems*, 16(6):e2023MS004177.

Shukla, K., Oommen, V., Peyvan, A., Penwarden, M., Plewacki, N., Bravo, L., Ghoshal, A., Kirby, R. M., and Karniadakis, G. E. (2024). Deep neural operators as accurate surrogates for shape optimization. *Engineering Applications of Artificial Intelligence*, 129:107615.

Silva Garzon, C. F., Bonnaire, P., Doan, N. A. K., Niebler, K., and Silva, C. F. (2023). Towards reconstruction of acoustic fields via physics-informed neural networks. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(3):4773–4782.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994). Learning Without State-Estimation in Partially Observable Markovian Decision Processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier.

Sipp, D., Marquet, O., Meliga, P., and Barbagallo, A. (2010). Dynamics and Control of Global Instabilities in Open-Flows: A Linearized Approach. *Applied Mechanics Reviews*, 63(3):030801.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. I. Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571.

Sitte, M. P. and Doan, N. A. K. (2022). Velocity reconstruction in puffing pool fires with physics-informed neural networks. *Physics of Fluids*, 34(8):087124.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc.

Sivashinsky, G. I. (1980). On Flame Propagation Under Conditions of Stoichiometry. *SIAM Journal on Applied Mathematics*, 39(1):67–82.

Sivashinsky, G. I. and Michelson, D. M. (1980). On Irregular Wavy Flow of a Liquid Film Down a Vertical Plane. *Progress of Theoretical Physics*, 63(6):2112–2114.

Solera-Rico, A., Sanmiguel Vila, C., Gómez-López, M., Wang, Y., Almashjary, A., Dawson, S. T. M., and Vinuesa, R. (2024). β-Variational autoencoders and transformers for reduced-order modelling of fluid flows. *Nature Communications*, 15(1):1361.

Sparrow, C. (1982). *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors*, volume 41 of *Applied Mathematical Sciences*. Springer New York, New York, NY.

Srinivasan, P. A., Guastoni, L., Azizpour, H., Schlatter, P., and Vinuesa, R. (2019). Predictions of turbulent shear flows using deep neural networks. *Physical Review Fluids*, 4(5):054603.

Storm, L., Gustavsson, K., and Mehlig, B. (2022). Constraints on parameter choices for successful time-series prediction with echo-state networks. *Machine Learning: Science and Technology*, 3(4):045021.

Strogatz, S. (2019). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. A Chapman & Hall Book. CRC Press, Boca Raton London New York, second edition, first issued in hardback edition.

Subramanian, P., Mariappan, S., Sujith, R. I., and Wahi, P. (2010). Bifurcation Analysis of Thermoacoustic Instability in a Horizontal Rijke Tube. *International Journal of Spray and Combustion Dynamics*, 2(4):325–355.

Sun, L. and Wang, J.-X. (2020). Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, 10(3):161–169.

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163.

Sutton, R. S. and Barto, A. (2018). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts London, England, second edition edition.

Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, D. and Young, L.-S., editors, *Dynamical Systems and Turbulence, Warwick 1980*, volume 898, pages 366–381. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tathawadekar, N., Doan, N. A. K., Silva, C. F., and Thuerey, N. (2021). Modeling of the nonlinear flame response of a Bunsen-type flame via multi-layer perceptron. *Proceedings of the Combustion Institute*, 38(4):6261–6269.

Thuburn, J. (2005). Climate sensitivities via a Fokker–Planck adjoint approach. *Quarterly Journal of the Royal Meteorological Society*, 131(605):73–92.

Towne, A., Schmidt, O. T., and Colonius, T. (2018). Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. *Journal of Fluid Mechanics*, 847:821–867.

Trefethen, L. N. (2000). *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ukasz Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Verma, S., Novati, G., and Koumoutsakos, P. (2018). Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854.

Vignon, C., Rabault, J., and Vinuesa, R. (2023). Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions. *Physics of Fluids*, 35(3):031301.

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., and Koumoutsakos, P. (2018). Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844.

Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., and Koumoutsakos, P. (2020). Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217.

Wang, H., Liu, Y., and Wang, S. (2022). Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network. *Physics of Fluids*, 34(1):017116.

Wang, Q. (2013). Forward and adjoint sensitivity computation of chaotic dynamical systems. *Journal of Computational Physics*, 235:1–13.

Wang, X., Yin, P., Zhang, B., and Yang, C. (2024). AONN-2: An adjoint-oriented neural network method for PDE-constrained shape optimization. *Journal of Computational Physics*, 513:113160.

Wang, Z., Xiao, D., Fang, F., Govindan, R., Pain, C. C., and Guo, Y. (2018). Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, University of Cambridge.

Waugh, I. C., Kashinath, K., and Juniper, M. P. (2014). Matrix-free continuation of limit cycles and their bifurcations for a ducted premixed flame. *Journal of Fluid Mechanics*, 759:1–27.

Weissenbacher, M. (2024). CHAROT. https://github.com/maxweissenbacher/charot.

Weissenbacher, M., Borovykh, A., and Rigas, G. (2025). Reinforcement Learning of Chaotic Systems Control in Partially Observable Environments. *Flow, Turbulence and Combustion*.

Xia, C., Zhang, J., Kerrigan, E. C., and Rigas, G. (2024). Active flow control for bluff body drag reduction using reinforcement learning with partial measurements. *Journal of Fluid Mechanics*, 981:A17.

Xiao, C., Deng, Y., and Wang, G. (2021a). Deep-Learning-Based Adjoint State Method: Methodology and Preliminary Application to Inverse Modeling. *Water Resources Research*, 57(2):e2020WR027400.

Xiao, D. and Papadakis, G. (2017). Nonlinear optimal control of bypass transition in a boundary layer flow. *Physics of Fluids*, 29(5).

Xiao, R., Kong, L.-W., Sun, Z.-K., and Lai, Y.-C. (2021b). Predicting amplitude death with machine learning. *Physical Review E*, 104(1):014205.

Xu, K. and Darve, E. (2022). Physics constrained learning for data-driven inverse modeling from sparse observations. *Journal of Computational Physics*, 453:110938.

Zaki, T. A. and Wang, M. (2021). From limited observations to the state of turbulence: Fundamental difficulties of flow reconstruction. *Physical Review Fluids*, 6(10):100501.

Zeng, K. and Graham, M. D. (2021). Symmetry reduction for deep reinforcement learning active control of chaotic spatiotemporal dynamics. *Physical Review E*, 104(1):014210.

Zeng, K., Linot, A. J., and Graham, M. D. (2022). Data-driven control of spatiotemporal chaos with reduced-order neural ODE-based models and reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478(2267):20220297.

Zhang, D., Guo, L., and Karniadakis, G. E. (2020). Learning in Modal Space: Solving Time-Dependent Stochastic PDEs Using Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 42(2):A639–A665.

Zhang, X., Xie, F., Ji, T., Zhu, Z., and Zheng, Y. (2021). Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 373:113485.

Zhao, D. (2012). Transient growth of flow disturbances in triggering a Rijke tube combustion instability. *Combustion and Flame*, 159(6):2126–2137.

Zighed, I., Thome, N., Gallinari, P., and Sayadi, T. (2025). UP-dROM : Uncertainty-Aware and Parametrised dynamic Reduced-Order Model, application to unsteady flows. *arXiv:2503.23236*.

Zinn, B. T. and Lores, M. E. (1971). Application of the Galerkin Method in the Solution of Non-linear Axial Combustion Instability Problems in Liquid Rockets. *Combustion Science and Technology*, 4(1):269–278.

Ziyin, L., Hartwig, T., and Ueda, M. (2020). Neural Networks Fail to Learn Periodic Functions and How to Fix It. In *Advances in Neural Information Processing Systems*, volume 33, pages 1583–1594. Curran Associates, Inc.

Appendix A

Thermoacoustics

The derivation of the natural acoustic eigenfunctions (§ 3.4.2) is provided in this Appendix. The governing equations of the Rijke tube are projected onto these eigenfunctions to obtain a low-order model (§ 3.4.3). These eigenfunctions are also incorporated in the spatial branch of Galerkin neural networks (GalNNs) tailored for thermoacoustics (§ 4.4).

A.1 Derivation of the thermoacoustic eigenmodes

We derive the natural acoustic eigenfunctions (3.11) similarly to (Zhao, 2012; Magri and Juniper, 2014). The eigenfunctions satisfy the wave equation in each duct segment l

$$\frac{\partial^2 p'}{\partial t^2} = \frac{1}{\rho_l}\frac{\partial^2 p'}{\partial x^2} \quad l = 1, 2 \tag{A.1}$$

where superscripts $(\cdot)^{(1)}$ and $(\cdot)^{(2)}$ denote the segments up- and downstream of the flame, and ρ_1 $(0 \leq x < x_f)$ and ρ_2 $(x_f < x \leq 1)$ are the respective densities that are constant over the segment. Applying the separation of variables $p'(x,t) = \mu^{(l)}(t)\Pi^{(l)}(x)$ yields

$$\frac{d^2\left(\mu^{(l)}(t)\right)}{dt^2}\frac{1}{\mu^{(l)}(t)} = \frac{1}{\rho_l}\frac{d^2\left(\Pi^{(l)}(x)\right)}{dx^2}\frac{1}{\Pi^{(l)}(x)} = -\kappa^2, \tag{A.2}$$

This results in the ODE

$$\frac{d^2\left(\Pi^{(l)}(x)\right)}{dx^2} + \rho_l\kappa^2\Pi^{(l)}(x) = 0. \tag{A.3}$$

The roots of the characteristic polynomial of the ODE (A.3) are purely imaginary. Therefore, for each segment, the ODE (A.3) has the solution

$$\Pi^{(1)}(x) = A\sin(\sqrt{\rho_1}\kappa x) + B\cos(\sqrt{\rho_1}\kappa x) \tag{A.4}$$

$$\Pi^{(2)}(x) = C\sin(\sqrt{\rho_2}\kappa(1-x)) + D\cos(\sqrt{\rho_2}\kappa(1-x)) \tag{A.5}$$

We impose the boundary conditions $p(0,t) = p(1,t) = 0$ and find

$$\Pi^{(1)}(x=0) = 0 \;\Rightarrow\; \Pi^{(1)}(x) = -\sin(\sqrt{\rho_1}\kappa x) \tag{A.6}$$

$$\Pi^{(2)}(x=1) = 0 \;\Rightarrow\; \Pi^{(2)}(x) = C\sin(\sqrt{\rho_2}\kappa(1-x)) \tag{A.7}$$

We impose the continuity of pressure at the flame location $p(x_f^-,t) = p(x_f^+,t)$ and find

$$\Pi^{(1)}(x=x_f) = \Pi^{(2)}(x=x_f) \;\Rightarrow\; \Pi^{(2)}(x) = \frac{-\sin(\sqrt{\rho_1}\kappa x_f)}{\sin(\sqrt{\rho_2}\kappa(1-x_f))}\sin(\sqrt{\rho_2}\kappa(1-x)) \tag{A.8}$$

For the modes to be orthogonal κ must satisfy

$$\int_0^1 \Pi_j(x)\Pi_i(x)dx \neq 0 \iff i = j, \tag{A.9}$$

where κ_i and κ_j are the wavenumbers associated with modes $\Pi_i(x)$ and $\Pi_j(x)$. We split this integral as

$$\int_0^1 \Pi_j(x)\Pi_i(x)dx = \underbrace{\int_0^{x_f} \Pi_j^{(1)}(x)\Pi_i^{(1)}(x)dx}_{(1)} + \underbrace{\int_{x_f}^1 \Pi_j^{(2)}(x)\Pi_i^{(2)}(x)dx}_{(2)}. \tag{A.10}$$

By substituting the modes (A.7) and (A.8) in the integral (A.10) and using trigonometric identities, we find

$$(1) = \frac{1}{\sqrt{\rho_1}(\kappa_j^2-\kappa_i^2)}\left(\kappa_j\sin(\gamma_j)\cos(\gamma_i) - \kappa_i\cos(\gamma_j)\sin(\gamma_i)\right), \tag{A.11}$$

$$(2) = \frac{1}{\sqrt{\rho_2}(\kappa_j^2-\kappa_i^2)}\frac{\sin(\gamma_j)\sin(\gamma_i)}{\sin(\beta_j)\sin(\beta_i)}\left(\kappa_j\sin(\beta_j)\cos(\beta_i) - \kappa_i\cos(\beta_j)\sin(\beta_i)\right), \tag{A.12}$$

where $\gamma_i = \sqrt{\rho_1}\kappa_i x_f$, $\gamma_j = \sqrt{\rho_1}\kappa_j x_f$, $\beta_i = \sqrt{\rho_2}\kappa_i(1-x_f)$, and $\beta_j = \sqrt{\rho_2}\kappa_j(1-x_f)$. Summing (1) and (2), we find

$$\begin{aligned} 0 =& \frac{\kappa_j}{\kappa_j^2-\kappa_i^2}\frac{\sin(\gamma_j)}{\sin(\beta_i)}\underbrace{\left(\sqrt{\rho_2}\sin(\beta_i)\cos(\gamma_i) + \sqrt{\rho_1}\sin(\gamma_i)\cos(\beta_i)\right)}_{=0\ dispersion} \\ &- \frac{\kappa_i}{\kappa_j^2-\kappa_i^2}\frac{\sin(\gamma_i)}{\sin(\beta_j)}\underbrace{\left(\sqrt{\rho_2}\sin(\beta_j)\cos(\gamma_j) + \sqrt{\rho_1}\sin(\gamma_j)\cos(\beta_j)\right)}_{=0\ dispersion}, \end{aligned} \tag{A.13}$$

which enforces the dispersion relationship (3.12). By substituting the pressure modes $\Pi_j(x)$ in the momentum equation (3.8a) and imposing the continuity of velocity condition $u(x_f^-,t) = u(x_f^+,t)$, we obtain the velocity modes $\Upsilon_j(x)$ (3.10b).

Appendix B

Adjoint equations

The derivations of the adjoint equations of a discrete-time system and a time-delayed discrete-time system are provided in this Appendix. The adjoint of the parameter-aware echo state network (PA-ESN) (5.14) and the thermoacoustic echo state network (T-ESN) (5.30) can be obtained based on this derivation.

B.1 Derivation of the adjoint of a discrete-time system

The following derivation of the adjoint sensitivity applies to any discrete map of choice that can be expressed as

$$\boldsymbol{s}(k) = \boldsymbol{F}(\boldsymbol{s}(k-1), \boldsymbol{\theta}), \tag{B.1}$$

where $\boldsymbol{s}(k) \in \mathbb{R}^{n_x}$ is the state vector at time step i. Equation (B.1) is rewritten as a constraint

$$\tilde{\boldsymbol{F}}(\boldsymbol{s}(k), \boldsymbol{s}(k-1), \boldsymbol{\theta}) \coloneqq \boldsymbol{s}(k) - \boldsymbol{F}(\boldsymbol{s}(k-1), \boldsymbol{\theta}) = 0. \tag{B.2}$$

We consider a time-averaged quantity as the objective functional

$$\mathcal{J}(\boldsymbol{s}, \boldsymbol{\theta}) \coloneqq \frac{1}{N} \sum_{k=1}^{N} \tilde{\mathcal{J}}(\boldsymbol{s}(k), \boldsymbol{\theta}). \tag{B.3}$$

Following Magri (2019), the problem of minimising the objective functional (B.3) under the constraints (B.2) can be formulated as a Lagrangian optimisation problem

$$\mathcal{L} \coloneqq \mathcal{J}(\boldsymbol{s}, \boldsymbol{\theta}) - \langle \boldsymbol{q}^+, \tilde{\boldsymbol{F}} \rangle, \tag{B.4}$$

where $\boldsymbol{q}^+$ are the Lagrange multipliers. The inner product $\langle \boldsymbol{q}^+, \tilde{\boldsymbol{F}} \rangle$ is given by

$$\langle \boldsymbol{q}^+, \tilde{\boldsymbol{F}} \rangle \coloneqq \sum_{k=1}^{N} \boldsymbol{q}^{+\top}(k) \tilde{\boldsymbol{F}}(k), \tag{B.5}$$

where we denote $\tilde{\boldsymbol{F}}(k) = \tilde{\boldsymbol{F}}(\boldsymbol{s}(k), \boldsymbol{s}(k-1), \boldsymbol{\theta})$ for brevity. Because $d\mathcal{J}/d\boldsymbol{\theta} = d\mathcal{L}/d\boldsymbol{\theta}$, we seek the gradient $d\mathcal{L}/d\boldsymbol{\theta}$

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{1}{N}\sum_{k=1}^{N}\frac{\partial\tilde{\mathcal{J}}}{\partial\boldsymbol{\theta}} + \frac{\partial\tilde{\mathcal{J}}}{\partial\boldsymbol{s}(k)}\frac{d\boldsymbol{s}(k)}{d\boldsymbol{\theta}} - \sum_{k=1}^{N}\boldsymbol{q}^{+\top}(k)\left(\frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{\theta}} + \frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{s}(k)}\frac{d\boldsymbol{s}(k)}{d\boldsymbol{\theta}} + \frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{s}(k-1)}\frac{d\boldsymbol{s}(k-1)}{d\boldsymbol{\theta}}\right). \tag{B.6}$$

We assume that the objective function does not explicitly depend on $\boldsymbol{\theta}$ and therefore $\partial\tilde{\mathcal{J}}/\partial\boldsymbol{\theta}$ is zero. Writing out the terms of the summation, we have

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \sum_{k=1}^{N} &- \boldsymbol{q}^{+\top}(k)\frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{\theta}} \\ &+ \frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(1))}{\partial\boldsymbol{s}(1)}\frac{d\boldsymbol{s}(1)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(1)\frac{\partial\tilde{\boldsymbol{F}}(1)}{\partial\boldsymbol{s}(1)}\frac{d\boldsymbol{s}(1)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(1)\frac{\partial\tilde{\boldsymbol{F}}(1)}{\partial\boldsymbol{s}(0)}\frac{d\boldsymbol{s}(0)}{d\boldsymbol{\theta}} \\ &+ \frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(2))}{\partial\boldsymbol{s}(2)}\frac{d\boldsymbol{s}(2)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(2)\frac{\partial\tilde{\boldsymbol{F}}(2)}{\partial\boldsymbol{s}(2)}\frac{d\boldsymbol{s}(2)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(2)\frac{\partial\tilde{\boldsymbol{F}}(2)}{\partial\boldsymbol{s}(1)}\frac{d\boldsymbol{s}(1)}{d\boldsymbol{\theta}} \\ &\vdots \\ &+ \frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(N))}{\partial\boldsymbol{s}(N)}\frac{d\boldsymbol{s}(N)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(N)\frac{\partial\tilde{\boldsymbol{F}}(N)}{\partial\boldsymbol{s}(N)}\frac{d\boldsymbol{s}(N)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(N)\frac{\partial\tilde{\boldsymbol{F}}(N)}{\partial\boldsymbol{s}(N-1)}\frac{d\boldsymbol{s}(N-1)}{d\boldsymbol{\theta}}.\end{aligned} \tag{B.7}$$

Because of Eq. (B.1), the terms $\partial\tilde{\boldsymbol{F}}(k)/\partial\boldsymbol{s}(k)$ are equal to the identity matrix, and $d\boldsymbol{s}(0)/d\boldsymbol{\theta}$ is equal to zero. We choose the Lagrange multipliers to eliminate the terms $d\boldsymbol{s}(k)/d\boldsymbol{\theta}$, the size of which grow with the dimension of $\boldsymbol{\theta}$. For this purpose, we gather the terms multiplying $d\boldsymbol{s}(k)/d\boldsymbol{\theta}$, thereby obtaining the evolution equations (5.14) for the Lagrange multipliers, i.e., the adjoint variables,

$$\begin{aligned}&\frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(1))}{\partial\boldsymbol{s}(1)} - \boldsymbol{q}^{+\top}(1) - \boldsymbol{q}^{+\top}(2)\frac{\partial\tilde{\boldsymbol{F}}(2)}{\partial\boldsymbol{s}(1)} = 0 \\ &\vdots \\ &\frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(N))}{\partial\boldsymbol{s}(N)} - \boldsymbol{q}^{+\top}(N) = 0,\end{aligned} \tag{B.8}$$

where from Eq. (B.1), $\partial\tilde{\boldsymbol{F}}(k)/\partial\boldsymbol{s}(k-1) = -\partial\boldsymbol{F}(k)/\partial\boldsymbol{s}(k-1)$.

B.2 Derivation of the adjoint of a time-delayed discrete-time system

The discrete map for the time-delayed can be expressed as

$$\tilde{\boldsymbol{F}}(\boldsymbol{s}(k), \boldsymbol{s}(k-1), \boldsymbol{s}(k-1-N_\tau), \boldsymbol{\theta}) = 0, \tag{B.9}$$

where N_τ is the time-delay in discrete time steps. The Lagrangian $\mathcal{L}$ is formulated similarly to Eq. (B.4).

The gradient $d\mathcal{L}/d\boldsymbol{\theta}$ is given by

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\theta}} =& \frac{1}{N}\sum_{k=1}^{N}\frac{\partial\tilde{\mathcal{J}}}{\partial\boldsymbol{\theta}} + \frac{\partial\tilde{\mathcal{J}}}{\partial\boldsymbol{s}(k)}\frac{d\boldsymbol{s}(k)}{d\boldsymbol{\theta}} \\ &- \sum_{k=1}^{N}\boldsymbol{q}^{+\top}(k)\Bigg(\frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{\theta}} + \frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{s}(k)}\frac{d\boldsymbol{s}(k)}{d\boldsymbol{\theta}} \\ &\qquad + \frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{s}(k-1)}\frac{d\boldsymbol{s}(k-1)}{d\boldsymbol{\theta}} + \frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{s}(k-1-N_\tau)}\frac{d\boldsymbol{s}(k-1-N_\tau)}{d\boldsymbol{\theta}}\Bigg)\end{aligned} . \tag{B.10}$$

Writing out the terms of the summation, we have

$$\begin{aligned}\frac{d\mathcal{L}}{dp} =& \sum_{k=1}^{N} - \boldsymbol{q}^{+\top}(k)\frac{\partial\tilde{\boldsymbol{F}}(k)}{\partial\boldsymbol{\theta}} \\ &+ \frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(1))}{\partial\boldsymbol{s}(1)}\frac{d\boldsymbol{s}(1)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(1)\frac{\partial\tilde{\boldsymbol{F}}(1)}{\partial\boldsymbol{s}(1)}\frac{d\boldsymbol{s}(1)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(1)\frac{\partial\tilde{\boldsymbol{F}}(1)}{\partial\boldsymbol{s}(0)}\frac{d\boldsymbol{s}(0)}{d\boldsymbol{\theta}} \\ &- \boldsymbol{q}^{+\top}(1)\frac{\partial\tilde{\boldsymbol{F}}(1)}{\partial\boldsymbol{s}(-N_\tau)}\frac{d\boldsymbol{s}(-N_\tau)}{d\boldsymbol{\theta}} \\ &+ \frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(2))}{\partial\boldsymbol{s}(2)}\frac{d\boldsymbol{s}(2)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(2)\frac{\partial\tilde{\boldsymbol{F}}(2)}{\partial\boldsymbol{s}(2)}\frac{d\boldsymbol{s}(2)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(2)\frac{\partial\tilde{\boldsymbol{F}}(2)}{\partial\boldsymbol{s}(1)}\frac{d\boldsymbol{s}(1)}{d\boldsymbol{\theta}} \\ &- \boldsymbol{q}^{+\top}(2)\frac{\partial\tilde{\boldsymbol{F}}(2)}{\partial\boldsymbol{s}(1-N_\tau)}\frac{d\boldsymbol{s}(1-N_\tau)}{d\boldsymbol{\theta}} \\ &\vdots \\ &+ \frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(2+N_\tau))}{\partial\boldsymbol{s}(2+N_\tau)}\frac{d\boldsymbol{s}(2+N_\tau)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(2+N_\tau)\frac{\partial\tilde{\boldsymbol{F}}(2+N_\tau)}{\partial\boldsymbol{s}(2+N_\tau)}\frac{d\boldsymbol{s}(2+N_\tau)}{d\boldsymbol{\theta}} \\ &- \boldsymbol{q}^{+\top}(2+N_\tau)\frac{\partial\tilde{\boldsymbol{F}}(2+N_\tau)}{\partial\boldsymbol{s}(1+N_\tau)}\frac{d\boldsymbol{s}(1+N_\tau)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(2+N_\tau)\frac{\partial\tilde{\boldsymbol{F}}(2+N_\tau)}{\partial\boldsymbol{s}(1)}\frac{d\boldsymbol{s}(1)}{d\boldsymbol{\theta}} \\ &\vdots \\ &+ \frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(N))}{\partial\boldsymbol{s}(N)}\frac{d\boldsymbol{s}(N)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(N)\frac{\partial\tilde{\boldsymbol{F}}(N)}{\partial\boldsymbol{s}(N)}\frac{d\boldsymbol{s}(N)}{d\boldsymbol{\theta}} \\ &- \boldsymbol{q}^{+\top}(N)\frac{\partial\tilde{\boldsymbol{F}}(N)}{\partial\boldsymbol{s}(N-1)}\frac{d\boldsymbol{s}(N-1)}{d\boldsymbol{\theta}} - \boldsymbol{q}^{+\top}(N)\frac{\partial\tilde{\boldsymbol{F}}(N)}{\partial\boldsymbol{s}(N-1-N_\tau)}\frac{d\boldsymbol{s}(N-1-N_\tau)}{d\boldsymbol{\theta}}.\end{aligned} \tag{B.11}$$

Because of Eq. (B.1), the terms $\partial\tilde{\boldsymbol{F}}(k)/\partial\boldsymbol{s}(k)$ are equal to the identity matrix, and due to the initial conditions $d\boldsymbol{s}(i \leq 0)/d\boldsymbol{\theta}$ are equal to zero. Eliminating $d\boldsymbol{s}/d\boldsymbol{\theta}$ terms, the evolution of the adjoint variables is given by

$$\begin{aligned}&\frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(1))}{\partial\boldsymbol{s}(1)} - \boldsymbol{q}^{+\top}(1) - \boldsymbol{q}^{+\top}(2)\frac{\partial\tilde{\boldsymbol{F}}(2)}{\partial\boldsymbol{s}(1)} - \boldsymbol{q}^{+\top}(2+N_\tau)\frac{\partial\tilde{\boldsymbol{F}}(2+N_\tau)}{\partial\boldsymbol{s}(1)} = 0, \\ &\vdots \\ &\frac{1}{N}\frac{\partial\tilde{\mathcal{J}}(\boldsymbol{s}(N))}{\partial\boldsymbol{s}(N)} - \boldsymbol{q}^{+\top}(N) = 0.\end{aligned} \tag{B.12}$$

We set $\boldsymbol{q}^{+}(k > N) = 0$ because there are no terms with $d\boldsymbol{s}(k > N)/d\boldsymbol{\theta}$.

Appendix C

Ensemble Kalman Filter

The derivation of the posterior update of the ensemble Kalman filter (§ 6.2.3) from a Bayesian perspective is provided in this Appendix.

C.1 Derivation of the posterior update

We recap the Bayes rule for the posterior (6.9)

$$\mathcal{P}(\tilde{\boldsymbol{s}} \mid \boldsymbol{o}) = \frac{\mathcal{P}(\boldsymbol{o} \mid \tilde{\boldsymbol{s}})\mathcal{P}(\tilde{\boldsymbol{s}})}{\mathcal{P}(\boldsymbol{o})}, \tag{C.1}$$

where we drop the time dependency for ease of notation and replace the state $\boldsymbol{s}$ with the augmented state $\tilde{\boldsymbol{s}} = [\boldsymbol{s}; \boldsymbol{M}(\boldsymbol{s})]$, which means that the observation operator is $\widetilde{\boldsymbol{M}} = [\boldsymbol{0}\ \boldsymbol{I}_{n_o \times n_o}]$. Assuming Gaussian prior and zero-mean Gaussian measurement noise

$$\mathcal{P}(\tilde{\boldsymbol{s}}) \propto \exp\left\{-\frac{1}{2}\left(\tilde{\boldsymbol{s}} - \tilde{\boldsymbol{s}}^{\mathrm{f}}\right)^{\top} \boldsymbol{C}_{\tilde{s}\tilde{s}}^{\mathrm{f}^{-1}} \left(\tilde{\boldsymbol{s}} - \tilde{\boldsymbol{s}}^{\mathrm{f}}\right)\right\}, \tag{C.2}$$

$$\mathcal{P}(\boldsymbol{o} \mid \tilde{\boldsymbol{s}}) \propto \exp\left\{-\frac{1}{2}\left(\boldsymbol{o} - \widetilde{\boldsymbol{M}}\tilde{\boldsymbol{s}}\right)^{\top} \boldsymbol{C}_{oo}^{-1} \left(\boldsymbol{o} - \widetilde{\boldsymbol{M}}\tilde{\boldsymbol{s}}\right)\right\}, \tag{C.3}$$

the posterior is also a Gaussian random variable

$$\begin{aligned} \mathcal{P}(\tilde{\boldsymbol{s}} \mid \boldsymbol{o}) &\propto \exp\left\{-\frac{1}{2}(\tilde{\boldsymbol{s}} - \tilde{\boldsymbol{s}}^{\mathrm{a}})^{\top} \boldsymbol{C}_{\tilde{s}\tilde{s}}^{\mathrm{a}^{-1}} (\tilde{\boldsymbol{s}} - \tilde{\boldsymbol{s}}^{\mathrm{a}})\right\}, \\ &= \exp\left\{-\frac{1}{2}\left(\left(\tilde{\boldsymbol{s}} - \tilde{\boldsymbol{s}}^{\mathrm{f}}\right)^{\top} \boldsymbol{C}_{\tilde{s}\tilde{s}}^{\mathrm{f}^{-1}} \left(\tilde{\boldsymbol{s}} - \tilde{\boldsymbol{s}}^{\mathrm{f}}\right) + \left(\boldsymbol{o} - \widetilde{\boldsymbol{M}}\tilde{\boldsymbol{s}}\right)^{\top} \boldsymbol{C}_{oo}^{-1} \left(\boldsymbol{o} - \widetilde{\boldsymbol{M}}\tilde{\boldsymbol{s}}\right)\right)\right\}, \end{aligned} \tag{C.4}$$

with analysis mean $\tilde{\boldsymbol{s}}^{\mathrm{a}}$ and analysis covariance $\boldsymbol{C}_{\tilde{s}\tilde{s}}^{\mathrm{a}}$. We determine the analysis covariance based on the terms quadratic in $\tilde{\boldsymbol{s}}$

$$\tilde{\boldsymbol{s}}^{\top} C_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}^{\mathrm{a}^{-1}} \tilde{\boldsymbol{s}} = \tilde{\boldsymbol{s}}^{\top} C_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}^{\mathrm{f}^{-1}} \tilde{\boldsymbol{s}} + \tilde{\boldsymbol{s}}^{\top} \widetilde{\boldsymbol{M}}^{\top} C_{oo}^{-1} \widetilde{\boldsymbol{M}} \tilde{\boldsymbol{s}} \Rightarrow C_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}^{a} = \left(C_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}^{\mathrm{f}^{-1}} + \widetilde{\boldsymbol{M}}^{\top} C_{oo}^{-1} \widetilde{\boldsymbol{M}}\right)^{-1}. \tag{C.5}$$

We determine the analysis mean as based on the terms linear in $\tilde{\boldsymbol{s}}$

$$\tilde{\boldsymbol{s}}^{\top} C_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}^{\mathrm{a}^{-1}} \tilde{\boldsymbol{s}}^{a} = \tilde{\boldsymbol{s}}^{\top} C_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}^{\mathrm{f}^{-1}} \tilde{\boldsymbol{s}}^{f} + \tilde{\boldsymbol{s}}^{\top} \widetilde{\boldsymbol{M}}^{\top} C_{oo}^{-1} \boldsymbol{o} \Rightarrow \tilde{\boldsymbol{s}}^{a} = \tilde{\boldsymbol{s}}^{f} + C_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}^{\mathrm{a}} \widetilde{\boldsymbol{M}}^{\top} C_{oo}^{-1} (\boldsymbol{o} - \widetilde{\boldsymbol{M}} \tilde{\boldsymbol{s}}^{f}). \tag{C.6}$$

Applying the Woodbury matrix identity to avoid inverting $C^{\mathrm{f}}_{\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}}$, which is typically large, and plugging in $\widetilde{\boldsymbol{M}}$, we obtain the posterior update equation (6.13).

Appendix D

Additional results

Additional results for Chapter 6 are provided in this Appendix, where the DA-MIRL is performed using a CA-ESN model with Lu readout.

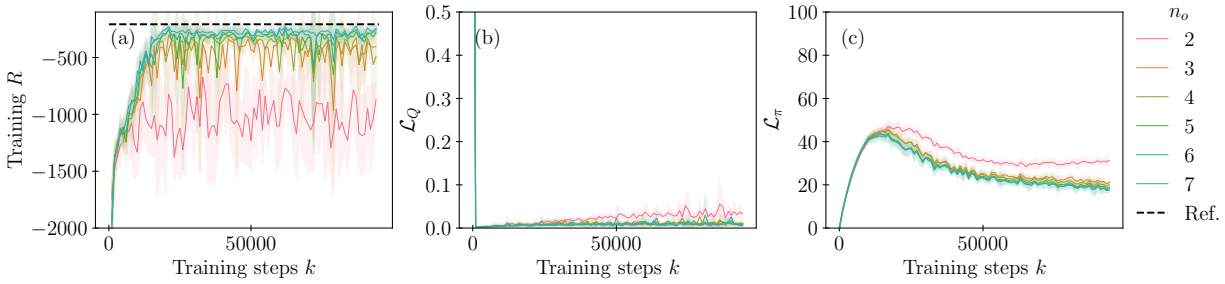


Figure D.1: Control-aware ESN: Training performance when using Lu readout (6.28) with varying number of sensors. In reference to 6.10, shown in (a) non-discounted returns of training episodes, (b) Q-loss, (c) policy loss across 5 runs (mean and 1 standard deviation in lighter colour). Ref. indicates the maximum return achieved by the model-free algorithm using the full-state observations, i.e., $n_o = n_s = 64$ in 6.4 (a)

D.1 Results of the DA-MIRL using the CA-ESN model with Lu readout

To break the antisymmetry of the tanh operator in the CA-ESN model, we either require (i) the addition of an input bias or (ii) the Lu readout (6.28) (Lu et al., 2017) which includes squared reservoir state variables in the readout layer (§ 6.3.2). The latter option (ii) is a nonlinear observation operator, although the original formulation of the EnKF requires linear observation operators, we circumvent this using the augmented state formulation (6.11). We prefer the option (i) in the results shown in the main text because the linear operator provides the exact solution to the filtering problem, however, here we provide the same set of results using the Lu readout with the same reservoir size $n_h = 1000$, because it is of interest that the algorithm still works in this case without any extra modifications. In reference to Fig. 6.10, Fig. D.1 shows the training performance across different number of sensors, where the algorithm successfully stabilizes the chaotic dynamics down

to $n_o = 2$ sensors. An episode with controlled actuations after training and an episode with random actuations are visualised in Figs. D.2 and D.3 in reference to Figs. 6.11 and 6.12, respectively.

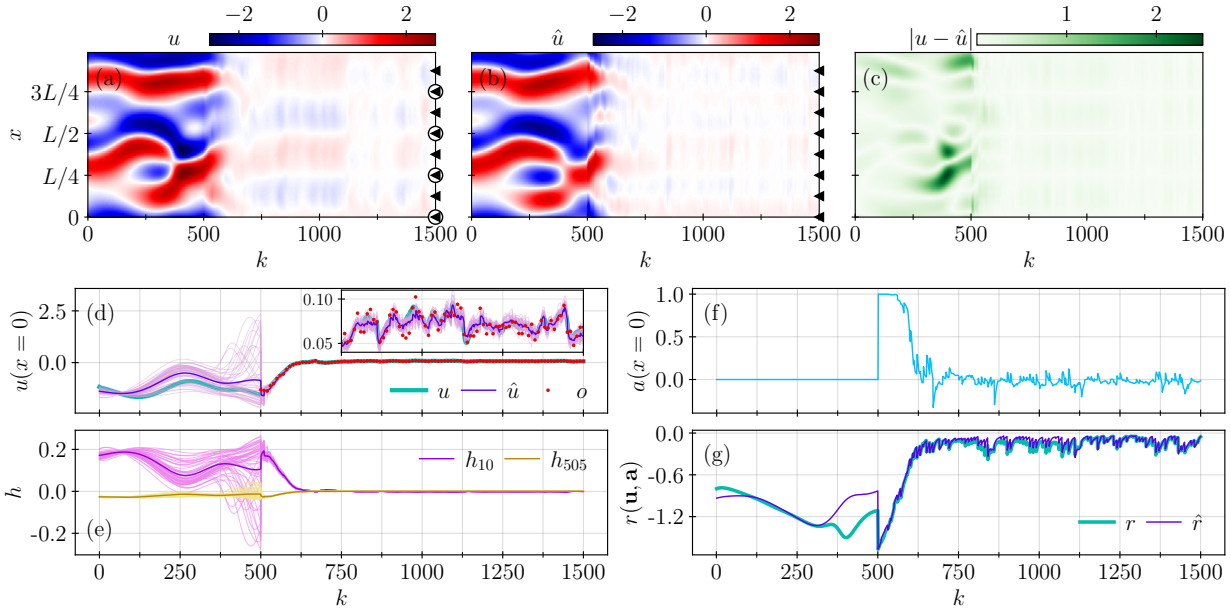


Figure D.2: Control-aware ESN control performance of DA-MIRL with the nonlinear observation operator defined by the Lu readout (6.28). In reference to Fig. 6.11, shown in (a – c) spatio-temporal evolution of the environment (the truth), the model (ensemble mean), and the reconstruction error. The triangles and circles indicate the location of the $n_a = 8$ actuators and $n_o = 4$ sensors, respectively. Shown in (d) and (e) ensemble forecast (pink), ensemble mean (purple) and true (teal) velocity at $x = 0$ and two reservoir states which take u_0 as input (no reference true value). Shown in (f) the agent's actions at $x = 0$, and (f) the reward function of the environment and the model. The observations and actuations begin at $k = 500$.

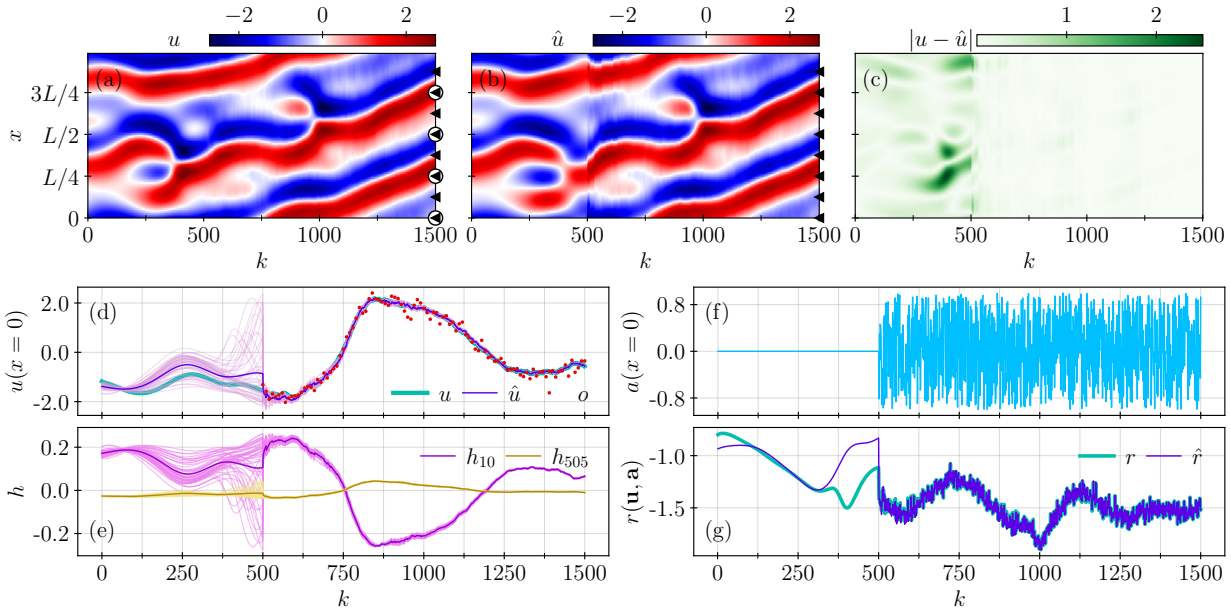


Figure D.3: State estimation of the control-aware ESN using the nonlinear observation operator defined by the Lu readout (6.28) in an episode with random actuations. In reference to Fig. 6.12, (a–g) same as Fig. D.2.

Appendix E

Computational costs

The computational costs associated with different algorithms introduced in this thesis are provided in this Appendix.

E.1 Computational costs in Chapter 4

The training costs associated with some of the characteristic cases in § 4 are provided below in Tabs. E.1 and E.2 for the twin experiments and higher-fidelity data experiments, respectively when using physics-informed Galerkin (GalNN) and feedforward neural networks (FNN). Hard-constraints allow for smaller neural networks which speeds up the training in the case of GalNNs. (Depending on the size of the dataset, the training times change between the different test cases.) The code is written using `TensorFlow` (Abadi et al., 2015) and run on CPU, since the network sizes are too small to profit from running on GPU due to the data transfer overhead.

Table E.1: Computation time of twin experiments with 13 pressure sensors using physics-informed periodic Galerkin neural network (PI-P-GalNN), Galerkin neural network (GalNN), and periodically activated feedforward neural network (PI-P-FNN). Training with a batch size of 32 for 2000 epochs on Intel Xeon(R) Gold 5218R CPU @ 2.10 Ghz x 80. (Ref. § 4.5.3).

Network name	**Description**	**Training time**
PI-P-GalNN	$n_g = 10$, 1 harmonic layer with 40 neurons	1 h 50 m
PI-GalNN	$n_g = 10$ Galerkin modes, 2 layers with 16 neurons	1 h 50 m
PI-P-FNN	3 layers with 32 neurons	2 h 40 m

Table E.2: Computation time of nonideal boundary condition higher-fidelity experiments with 5 pressure sensors using physics-informed periodic Galerkin neural network (PI-P-GalNN) and periodically activated feedforward neural network (PI-P-FNN). Training with a batch size of 32 for 10000 epochs on Intel Core(TM) i7-9750H CPU @ 2.60 Ghz x 6. (Ref. § 4.6.4).

Network name	**Description**	**Training time**
PI-P-GalNN	$n_g = 15$, 1 harmonic layer with 10 neurons	20 m
PI-P-FNN	4 layers with 32 neurons	35 m

E.2 Computational costs in Chapter 5

The computational performance of the thermoacoustic echo state network (T-ESN) in the test case discussed in § 5 in comparison to the original adjoint solver is provided below in Tab. E.3. The code is written in `NumPy`, leveraging the `SciPy` library, `scipy.sparse`, for sparse matrix operations. The original solver is also implemented in `NumPy` and integrated using `scipy.integrate.odeint`.

The training of the ESN is faster compared to neural networks that are trained with backpropagation. Thus, the training time scales roughly linearly with the number of regimes. (Training time includes both running the ESN in open-loop to generate the reservoir matrix $\boldsymbol{H}$ and perform ridge regression.) The validation procedure in total took roughly 2 hours for our case of training with 20 regimes and validating on 5 regimes, where we evaluated 250 different sets of hyperparameters (iteratively chosen with Bayesian optimization) for the seven hyperparameters considered in T-ESN. For the problem considered, the T-ESN runs faster than the original solver when simulating the system. Although this can be case dependent, in general the propagation of the ESN equations is fast thanks to the sparse matrices even though the reservoir state can be high dimensional (here, $n_h = 1200$). Because we have not optimised on GPU, the evaluation of the adjoint of the ESN, which requires multiplication of two 1200×1200 matrices for the Jacobian, is relatively slower. Indeed, we performed the following test using `JAX` (Bradbury et al., 2018) on a GPU (NVIDIA Quadro RTX 8000), where we compared the computational cost of the matrix multiplication required for the Jacobian for increasing matrix dimension, i.e., dimension of the reservoir state. In GPU computations, the cost for small matrices is dominated by the overhead (compilation, data transfer). However, we see a major speed up for larger matrices, e.g., in the case of 1200 up to 50 times, which opens up possibilities for future code optimisation.

Table E.3: Computational performance comparison between original solver and thermoacoustic echo state network (T-ESN) (mean ± standard deviation). Results obtained on Intel Xeon(R) Gold 5218R CPU at 2.10 GHz × 80. (Ref. § 5.6)

Operation	**Original Solver**	**T-ESN**
Training Time		
5 parameter sets	–	0.68 ± 0.11 s
10 parameter sets	–	1.24 ± 0.15 s
20 parameter sets	–	2.72 ± 0.24 s
Simulation	0.029 ± 0.006 s	0.015 ± 0.002 s
Parametric Sensitivity	*Cost / Time unit of integration*	
Adjoint method	0.055 ± 0.011 s	1.75 ± 0.350 s
Tangent linear	0.688 ± 0.020 s	1.67 ± 0.151 s
Central finite difference	0.094 ± 0.003 s	0.047 ± 0.009 s
Initial Condition Sensitivity	*Cost / Time unit of integration*	
Adjoint method	0.026 ± 0.001 s	1.67 ± 0.267 s
Tangent linear	0.035 ± 0.004 s	13.3 ± 0.926 s
Central finite difference	0.791 ± 0.019 s	39.6 ± 5.04 s

E.3 Computational costs in Chapter 6

The training times associated with model-free reinforcement learning (RL), data-assimilated model-informed reinforcement learning (DA-MIRL) and control-aware echo state network (T-ESN) on the test case discussed in § 6 are provided below in Tab. E.4. The training is quick thanks to the parallelisation of the ensemble operations and the optimisation capabilities of `JAX` (Bradbury et al., 2018) on GPU.

Table E.4: Training times for model-free reinforcement learning (RL) and data-assimilated model-informed reinforcement learning (DA-MIRL). Results obtained on one NVIDIA Quadro RTX 8000 GPU. (Ref. § 6.3).

Operation	**Description**	**Time**
Model-free RL	Full state $n_o = 64$	7 m
DA-MIRL with Fourier model	$n_f = 16$, $n_{\text{en}} = 50$	7 m
CA-ESN validation	$n_h = 1000$	4 m
CA-ESN training	$n_h = 1000$	5 m
DA-MIRL with CA-ESN	$n_h = 1000$, $n_{\text{en}} = 50$	14 m