



**University of
Nottingham**

UK | CHINA | MALAYSIA

Controlling Nonlinear PDEs Arising in Fluid Mechanics:

Open loop-, feedback control, and neural networks

A thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy in Mathematics
by

Sattam Alrashidy

School of Mathematical Sciences
University of Nottingham

May 18, 2026

Abstract

This thesis addresses the control of systems governed by nonlinear partial differential equations, combining mathematical modelling, optimisation, and neural-network tools within a unified framework. The primary application is thin liquid films flowing over flexible surfaces, where the film and the underlying wall affect one another. A central question is how external actuation should be applied so that the film evolves in a controlled and predictable way. To study this, the thesis develops mathematical frameworks for PDE control together with numerical and neural-network approaches that support actuator placement and feedback design. The neural-network methods provide flexible representations for control strategies and help to incorporate the governing equations into the optimisation process. The work is organised in two parts: the first focuses on open-loop control of thin-film systems, and the second introduces actuator-design and feedback-control techniques using neural-network tools.

The first part of the thesis develops a PDE-constrained optimisation framework for open-loop control of thin-film flow over a flexible substrate. The model couples a nonlinear lubrication equation with an elastic-damping equation for the wall, capturing the interaction between the film and the flexible surface. The system allows for film rupture and coalescence and includes a global energy-dissipation law that reflects the physical structure of the problem. The control problem seeks a distributed forcing of the wall that drives the film towards a desired state. Optimality conditions are derived at the continuous level and solved numerically using finite-element discretisation and implicit-explicit time integration. The numerical simulations show how wall forcing can suppress instabilities, stabilise dewetting, and accelerate convergence towards stable film shapes. This formulation provides a foundation for PDE-constrained optimisation of thin-film systems.

The second part extends this framework to actuator design and feedback control. A one-shot optimisation formulation is introduced, in which actuator placement, control inputs, and state variables are optimised within a single PDE-constrained problem. This avoids the nested and adjoint structures of classical bilevel approaches. In the open-loop setting, physics-informed neural networks are used as a trial space for the thin-film dynamics, with actuator parameters trained alongside the control signal. Numerical experiments show that the framework identifies suitable actuator locations and control inputs, with finite-element simulations used for validation. In the feedback setting, the Hamilton-Jacobi-Bellman framework is extended to include actuator design parameters. The value function is represented by a neural network whose training jointly optimises the feedback law and the actuator configuration. This removes the need for explicit HJB solvers or precomputed datasets. Numerical experiments on the heat equation confirm that the learned controllers achieve performance comparable to Riccati-based feedback whilst enabling flexible ac-

tuator adaptation.

Overall, the thesis combines mathematical analysis, PDE modelling, and learning-based optimisation. It develops open-loop control and actuator-design tools for thin-film systems on flexible substrates and introduces a learning-based feedback framework in a simplified PDE setting. The feedback methodology is demonstrated for the heat equation, indicating a possible direction for future feedback control of thin-film flows.

Acknowledgements

Alhamdulillah, all praise is due to Allah for granting me the strength, patience, and ability to complete this PhD journey.

During these four years of my PhD, many people offered support, for which I am grateful. First, I would like to express my sincere gratitude to my principal supervisor, Professor Kris van der Zee, for his steady support and thoughtful guidance throughout my PhD. Kris's supportive supervision and his emphasis on constructive and positive feedback created an environment in which I felt encouraged to develop and improve. His patience and understanding were particularly meaningful during the challenging moments of my study, especially around the time my son, Salem, was born.

I would also like to thank my supervisor, Dr Anna Kalogirou, for her consistent support throughout my PhD. Her clear feedback, thoughtful suggestions, and careful reading of my work have been invaluable. I greatly appreciated the time and effort she devoted to helping me refine my ideas and improve the quality of this thesis.

I am also grateful to my supervisor, Dr Dante Kalise from Imperial College London, for his support and guidance throughout my PhD. Despite his many commitments, he consistently made time to provide insightful comments and valuable suggestions, particularly on the control aspects of this project.

I would also like to thank my thesis examiners, Dr Susana Gomes and Dr Luis Espath, for their careful reading of this thesis and for their constructive comments and suggestions. Their feedback helped improve both the presentation and the overall quality of this work.

My thanks also extend to my colleagues in Kris's group, whose company made meetings and research discussions both enjoyable and motivating. I am especially grateful to Ignacio Brevis for the many valuable discussions, particularly on machine learning, and for his thoughtful comments on my work.

I am also thankful to my friends in Nottingham, including the Friday group and the Barnum group, for their kindness, companionship, and the many enjoyable moments we shared throughout this journey. I especially appreciate my friend and colleague, Hamad Alsobhi, whose company and conversations

at the university made the PhD journey more manageable.

On a personal level, I am deeply grateful to my parents for their constant prayers, encouragement, and support throughout my studies. I would also like to thank my siblings for their encouragement and kindness. I would also like to express my sincere appreciation to the government of Saudi Arabia and to the Islamic University of Madinah for providing the scholarship and financial support that made my studies possible.

My deepest gratitude goes to my wife, whose patience, understanding, and unwavering support sustained me throughout this journey. Her encouragement was especially meaningful during the more challenging moments of my PhD, and during the time when we welcomed our son, Salem. I am especially grateful for her constant belief in me throughout this journey. I also thank Salem, whose arrival brought great joy and perspective during the final year of my PhD.

Contents

Abstract	i
Acknowledgements	iii
1 Introduction	1
1.1 Motivation	2
1.2 Objectives and Contributions	5
1.3 Outline of the Thesis	7
 I Optimal Control of Thin-Film Flow on a Flexible Topography	 8
2 Introduction to Optimal Control of PDEs	9
2.1 Introduction	10
2.2 Control Designs	11
2.3 Differentiation in function spaces	13
2.4 Functional setting and weak formulation	15
2.5 Optimality conditions	17
2.6 Numerical solution strategies	19
2.7 Numerical methods	22
2.7.1 All-at-once methods	22
2.7.2 Iterative methods	23
2.8 Conclusion	26
3 Optimal Control the Thin-Film Flow on a Flexible Topography	28
3.1 Introduction	29

3.2	Model of thin-film flow over a flexible topography.	31
3.2.1	State equation	31
3.2.2	Weak form of the state equations	34
3.2.3	Boundedness of weak form and disjoining pressure as- sumptions	35
3.3	Optimal Control Problem	37
3.4	First-order optimality conditions	37
3.4.1	Reduced gradient approach	39
3.5	Numerical approximation and tests	40
3.5.1	IMEX-FEM discretisation of the model	40
3.5.2	Numerical tests	41
3.5.3	Non-rupture scenarios	42
3.5.4	Rupture scenarios	51
3.6	Conclusion	54
II	Actuator design and learned feedback control	57
4	Neural Networks for Solving and Controlling PDEs	58
4.1	Introduction	59
4.2	Neural Networks	60
4.2.1	Feedforward Neural Networks	61
4.2.2	Approximation Theory and Activation	62
4.2.3	Training and Optimisation	64
4.3	PINNs for Solving Nonlinear PDEs	67
4.3.1	Case study: thin-film flow over a flexible topography . .	69
4.4	Classical Optimal Control of PDEs	71
4.5	PINNs for Controlling Nonlinear PDEs	72
4.6	Data-Driven Discovery of PDEs	74
5	Optimal Actuator Design for Thin-Film Flow on a Flexible Topography	77
5.1	Introduction	78

5.2	Model and optimality conditions	79
5.3	Supervised-learning optimisation	82
5.4	Direct actuator design optimisation	84
5.5	Numerical results	86
5.6	Conclusion	91
6	Direct Learning feedback control for actuator design problems	92
6.1	Introduction	93
6.2	Classical Feedback Formulations	95
6.3	Continuous-time Feedback Control	97
6.4	Actuator Design through Value Function	98
6.5	Joint Learning of Feedback Control and Actuator Design	100
6.6	Numerical results	102
6.6.1	One-dimensional scalar test problem	103
6.6.2	Two-dimensional heat equation	106
6.7	Conclusion	112
7	Conclusions	115
7.1	Summary and Discussion	116
7.2	Future Work	117
	Bibliography	120
	Appendices	132
A	Appendices of Chapter 3	132
A.1	Derivation of the thin film flow on a flexible topography	132
A.2	Free energy dissipation	134
A.3	Boundedness of $(\phi'(h), w)$	135
A.4	IMEX-FEM discretisation	136
B	Appendices of Chapter 5	138
B.1	Supervised learning approach	138
B.2	Model Order Reduction (MOR)	140
B.2.1	First Steps into MOR	141

B.2.2	Proper Orthogonal Decomposition (POD)	142
B.2.3	Numerical Result	142
C	Appendices for Chapter 6	145
C.1	LQR closed-form optimal actuator position	145
C.2	Implementing the 2D Heat Equation	146
C.3	Finite-horizon discrete LQR (Riccati)	147

List of Tables

3.1	Summary of numerical examples (shaded rows indicate rupture cases).	43
4.1	Common activation functions and their definitions.	63
4.2	Hyperparameters employed in the PINN approach.	70
5.1	Key neural-network parameters	87
6.1	Neural-network training parameters	105
6.2	Model and training parameters.	109
A.1	Variable scaling (asterisks denote dimensional quantities). Here, $\bar{\mu}$ is the viscosity and U_0 is the characteristic velocity used in the scaling.	133
A.2	Nondimensional parameters. Here, ρ is the density, g is the gravity, σ is the surface tension, A^* is the dimensional Hamaker constant, γ_w is the wall damping, and T_w is the wall tension. . .	134

List of Figures

3.1	Illustration of a two-dimensional fluid flow on a flexible substrate, where $h(x, t)$ denotes the film thickness, $s(x, t)$ is the shape of the topography, and $\mathcal{H}(x, t) = h(x, t) + s(x, t)$ is the height of the one-dimensional free surface. The profile of the topography is controlled by a forcing function $f(x, t)$, which is represented by the colour of the curve, where red indicates downward (positive) forcing and blue indicates upward (negative) forcing. g is the gravity vector.	31
3.2	Example 1. Snapshots of the optimal solution, controlled to a desired surface shape. The solution is extended symmetrically in the domain $[-L, L]$	45
3.3	Example 1. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively. (f) Cost function $\mathcal{J}(f^*)$ against number of iterations k	45
3.4	Example 1. L^∞ -norms for (a) the controlled and uncontrolled solutions, and (b) various values of α	45
3.5	Example 2. Snapshots of the optimal solution, controlled to a desired surface shape.	47
3.6	Example 2. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.	47

3.7	Example 2. (a) L^∞ -norm for the controlled and uncontrolled solutions. (b) Cost function $\mathcal{J}(f^*)$ against number of iterations k	47
3.8	Example 3. Snapshots of the optimal solution, controlled to a desired surface shape. The solution is extended symmetrically in the domain $[-L, L]$	48
3.9	Example 3. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.	48
3.10	Example 3. Cost function $\mathcal{J}(f^*)$ against number of iterations k	48
3.11	Example 4. Snapshots of the optimal solution, controlled to a uniform state. The solution is extended symmetrically in the domain $[-L, L]$	50
3.12	Example 4. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.	50
3.13	Example 4. (a) L^∞ -norm for controlled and uncontrolled solutions. (b) Cost function $\mathcal{J}(f^*)$ against number of iterations k	50
3.14	Example 5. Snapshots of the optimal solution, controlled to remain at the initial surface shape. The solution is extended symmetrically in the domain $[-L, L]$	52
3.15	Example 5. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively. (f) Cost function $\mathcal{J}(f^*)$ against number of iterations k	52
3.16	Example 5. L^∞ -norm of controlled vs uncontrolled solutions relative to $\bar{h}$	52

3.17	Example 6. Snapshots of the optimal solution, controlled towards a ruptured state. The solution is extended symmetrically in the domain $[-L, L]$	53
3.18	Example 6. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.	53
3.19	Example 6. (a) L^∞ -norm of the variation $h^* - \bar{h}_{\text{rupt}}$. (b) Cost function $\mathcal{J}(f^*)$ against number of iterations k	53
3.20	Example 7. Snapshots of the optimal solution, controlled from a ruptured state to a uniform film thickness. The solution is extended symmetrically in the domain $[-L, L]$	55
3.21	Example 7. (a-d) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively. (e) Cost function $\mathcal{J}(f^*)$ against number of iterations k	55
4.1	A sketch of a feedforward neural network with one hidden layer.	61
4.2	Activation functions: ReLU, tanh, sigmoid, and softplus. This figure illustrates the output curves for each activation function over a range of x values, allowing for a visual comparison of their nonlinearity and range.	64
4.3	Comparison of FEM (solid) and PINN (dashed) solutions for $h(x, t)$, $s(x, t)$, the combined field $h + s$, and the corresponding L_2 error over time.	71
5.1	Evolution of the total and individual loss components during training.	87
5.2	Initial (dashed) and optimised (solid) actuator supports $\chi_i(x; w_i)$. The learned positions shift toward the central region, where they most effectively influence the film dynamics.	88
5.3	Learned control inputs $f_1(t)$ and $f_2(t)$ over time.	89

5.4	Evolution of (top) film thickness $h(x, t)$ and (bottom) substrate displacement $s(x, t)$ at selected times.	89
5.5	Evolution of the total free surface $\mathcal{H} = h + s$ compared with the target profile $\bar{h}(x)$ (dashed).	90
5.6	Cross-validation with an independent FEM forward simulation. (a) The FEM solution driven by the learned controls converges toward the target state. (b) The discrepancy between the PINN and FEM free surfaces remains below 10^{-1} , confirming close agreement.	90
6.1	Training losses (total, control, and terminal) versus epoch on a log-log scale during neural network training. The total-loss curve is partially obscured where it nearly overlaps with the control-loss curve.	106
6.2	Evolution of actuator centre position w during neural network training.	106
6.3	State trajectories: neural-network (solid) vs. analytical (dashed) for various h_0	107
6.4	Control trajectories: neural-network (solid) vs. analytical (dashed) for various h_0	107
6.5	Absolute final error $ h(T) $ vs. initial height h_0	108
6.6	Training loss evolution for the single-actuator case.	110
6.7	The dashed square corresponds to the initial actuator position, and the solid square corresponds to the learned position after training.	110
6.8	Comparison between neural and Riccati control signals for the single actuator.	110

6.9	Snapshots of state evolution for the single actuator: (top) no control, (middle) neural feedback, (bottom) Riccati solution. In the controlled cases (middle and bottom rows), the black square marks the actuator support centred at the learned position w^* .	111
6.10	Temporal evolution of the energy norm $\ \mathbf{h}(t)\ _2^2$ for the single-actuator case. The blue curve shows the natural diffusion of the system. The orange curve corresponds to feedback control using a fixed, non-optimal actuator location. The green curve shows the performance achieved after jointly learning both the feedback law and the actuator position.	111
6.11	Training loss evolution for the two-actuator case.	112
6.12	Initial (dashed) and learned (solid) actuator positions for the two-actuator case.	112
6.13	Snapshots of state evolution for the two-actuator case: (top) no control, (middle) neural feedback, (bottom) Riccati solution.	113
6.14	Comparison between neural and Riccati control signals for the two actuators.	113
6.15	Temporal evolution of the energy norm $\ \mathbf{h}(t)\ _2^2$ for the two-actuator case.	113
B.1	Distribution of the computed costs $\mathcal{J}(h^*, s^*, \{f_i^*\}, \omega)$ over the sampled actuator locations (w_1, w_2) . The low-cost region corresponds to actuator configurations producing free surfaces close to the target.	139
B.2	Evolution of the training and validation losses during supervised learning. Both losses decrease by roughly two orders of magnitude, although the validation curve remains consistently above the training loss, suggesting moderate generalisation error.	139

B.3	Comparison of the true cost surface (left) and the ANN-predicted surface (right). The predicted minimum closely matches the true optimum, though slight deviations appear in less-sampled regions.	140
B.4	Optimal actuator supports obtained from the supervised learning model. Each actuator acts over a compact region of width $2\delta = 1$ centred at (w_1^*, w_2^*)	140
B.5	FOM solution and error analysis of the POD solutions.	143
B.6	POD solutions with basis sizes $N_r = 3$ and $N_r = 7$ and their corresponding L_∞ -norm errors, effectively demonstrating low-dimensional approximations of the FOM.	144

Chapter 1

Introduction

This chapter introduces the motivation for studying the control of systems governed by partial differential equations, with particular focus on thin-film flow over a flexible substrate. It outlines the main applications and briefly reviews the mathematical background. The objectives and key contributions of the thesis are then presented, followed by an overview of its structure and chapter organisation.

1.1 Motivation

Partial differential equations (PDEs) are essential to numerous scientific and engineering disciplines. One of the areas where they are beneficial is in fluid dynamics, with a special interest in thin films [85, 87, 115]. A thin film is a liquid whose height is markedly smaller than its typical length [136]. Thin-film flows occur in many industrial and natural processes, including lubrication [159], coating and cooling [137], and biological transport [60]. In mechanical systems, thin layers of lubricating liquid reduce friction between moving parts [159]. Similarly, in biological systems, these mechanisms are observed in mucus transport, tear films, and blood flow through narrow vessels [60]. Whilst much of the previous theoretical and experimental work on thin-film flow has involved a rigid (flat) topography, many of the substrates which occur in practice are complex and may be flexible.

The extension of thin-film flow over a flat substrate to the case of a flexible topography introduces a coupled fluid-structure interaction. In this setting, the evolving free surface and the deformation of the substrate influence each other dynamically. This coupling generates complex flow patterns and surface instabilities that are central to many physical systems [4]. Biological examples include pulmonary flows [60, 101], where a viscous film coating the airways interacts with the gas phase and the elastic airway walls. Moreover, similar behaviour arises in blood flow through deformable vessels [20]. In engineering, flexible materials are employed to manipulate fluid behaviour, for instance in drag reduction [105], where wall deformation can lower hydrodynamic resistance and energy consumption. Examples include bio-inspired elastic coatings and adaptive hull surfaces designed to reduce skin friction on ships and underwater vehicles [4]. Comparable coupling mechanisms also appear in geophysical and industrial processes, such as ice sheets floating on oceans [146] and polymer processing flows [94]. These examples motivate the mathematical study of fluid-structure interaction systems, which give rise to coupled evolution equa-

tions. In such settings, the dynamics of the fluid flow and the elastic deflection of the flexible substrate are linked through the forces acting at the fluid-solid interface, such as pressure and viscous stresses [4].

The behaviour of thin films over a flexible topography is typically described by reduced models that couple the evolution of the film height with the response of the substrate [4]. These models are typically derived from long-wave approximations of the Navier-Stokes equations and capture the interplay between viscous, capillary, and elastic forces [113]. The resulting system is leading to nonlinear dynamics that are sensitive to both geometry and material of the liquid and solid substrate properties. Even small variations in the wall shape can lead to significant changes in the film profile or flow stability [115]. This sensitivity naturally leads to the question of how such flows can be controlled.

Controlling thin-film flow over a flexible topography involves influencing how the film spreads or settles by adjusting the substrate or applying external actuation. For example, coating processes favour stable, flat film flows [152], whereas applications in heat and mass transfer often require interfacial deformations [48]. Two main types of controls are commonly used: open-loop control and feedback control [112]. In an open-loop control, the actuation pattern is designed in advance – often based on the initial state of the film – and then applied as the system evolves, without further modification [11]. This approach provides insight into how actuation affects the coupled flow of the fluid and the substrate, and is useful when the system is well understood or operates under predictable conditions. However, open-loop control does not account for variations or disturbances during system evolution [80]. This limitation motivates feedback control, in which the input depends explicitly on the evolving state.

Designing feedback laws for systems governed by PDEs is particularly challenging [154]. In feedback control, the input must continuously respond to the evolving state of the system, which requires constructing a control law

that depends on the entire spatial configuration [80]. For nonlinear or high-dimensional systems, this task leads to an equation known as the Hamilton-Jacobi-Bellman (HJB) equation, whose solution characterises the optimal feedback law [96]. In practice, solving the HJB equation directly is rarely feasible. The equation becomes intractable as the number of spatial variables increases or when the underlying dynamics are strongly coupled [80]. Moreover, the performance of feedback control also depends on the nature of the actuation itself [81]. Consequently, addressing these computational challenges is important for practical applications that rely on optimal actuator placement, such as vibration control in flexible structures, thermal regulation, and the control of fluid flows [83, 121, 127].

In many controlled systems, particularly in thin-film flows, actuators may correspond to localised forces or substrate deformations that shape the film profile [10, 152]. Determining not only the optimal control signal but also how and where the actuation is applied introduces a coupled design-control problem [81]. The system should learn both an effective feedback law and the location of the actuation that realises it most efficiently [83]. Addressing this coupled problem requires flexible approaches capable of learning directly from the governing PDEs.

Solving the control problems for PDEs is often demanding, especially when the behaviour of the system changes over time or depends on complex geometries, such as thin film flows over flexible topography [66]. Traditional methods can become slow or difficult to use when the system has a large number of degrees of freedom or involves complex nonlinear PDEs [38]. In recent years, neural networks have offered a new way forward. They can capture the behaviour of complex systems and learn how the dynamics evolve directly from the governing physics [21]. This makes it possible to combine modelling, data, and control in a single, flexible framework [134].

The challenges outlined above motivate the development of new mathematical and computational frameworks for controlling systems governed by PDEs,

with particular attention to thin films moving over a flexible topography.

1.2 Objectives and Contributions

Objective of the Thesis

The main objective of this thesis is to develop mathematical and computational frameworks for the control of nonlinear partial differential equation systems, with particular emphasis on thin-film flow over a flexible wall. The study aims to understand how the interaction between fluid flow and wall deformation can be exploited to control and stabilise the film. It first aims to develop an open-loop control framework in which distributed forcing acts on the flexible wall to guide the film dynamics. This is then extended to actuator design, to determine effective actuator locations and coordinate the control with the mechanical response of the wall. Moreover, the thesis aims to develop learning-based open-loop and feedback control strategies, using a one-shot framework in which the actuator design and the control law are determined together within a single process. Together, these objectives seek to unify mathematical modelling, analysis, and computation within a coherent framework for the control of thin-film systems.

Main Contributions

This thesis develops three main research contributions to the control of nonlinear PDE systems, with thin-film flow over flexible substrates as the principal application. In addition, the work brings together the essential background in PDE-constrained optimisation and neural-network methods, which support the following main contributions:

1. **Open-loop control of thin-film flow over a flexible topography.**

A PDE-constrained optimisation framework is developed for thin-film

flow coupled to the deformation of an elastic substrate. The thin-film model accommodates rupture and bubble coalescence and is provided with a global energy-dissipation law. An optimal control problem is formulated for distributed forcing acting on the substrate, and the first-order optimality system is derived at the continuous level. A stable time discretisation and a gradient-based optimisation scheme are employed to compute the control. Numerical experiments show that the resulting strategy can steer the free-surface profile towards prescribed targets, stabilise dewetting, and remain effective in rupture regimes.

2. **One-shot actuator design for thin-film systems.** A direct one-shot optimisation framework is proposed that couples the state, control, and actuator design variables within a single differentiable formulation. The classical bilevel structure is reformulated as a single-level optimisation in which all variables are optimised simultaneously. The approach avoids explicit adjoint derivations, surrogate models, and nested iterations. A computed example demonstrates that the framework identifies effective actuator locations and produces control inputs that guide the free-surface evolution to the desired state.
3. **Learned feedback control with actuator design via value-function approximation.** The direct value-function learning approach is extended to design-actuator feedback-control problems governed by PDEs. A one-shot, data-free formulation is introduced in which the value function and actuator parameters are optimised jointly within a single training stage. A neural approximation of the value function implicitly represents the feedback law, removing the need to solve the Hamilton-Jacobi-Bellman equation explicitly. Applications to heat equations demonstrate that the method can learn stabilising feedback controls and actuator locations with performance comparable to Riccati-based controllers.

1.3 Outline of the Thesis

The thesis is organised into two main parts. Part I contains both foundational material on PDE-constrained optimal control and novel results on the open-loop control of thin-film flow over a flexible topography (Ch. 3), which has been accepted for publication in the *SIAM Journal on Applied Mathematics* [6]. Part II combines a methodological overview of neural-network-based approaches with new developments on actuator design and feedback control. All chapters are self-contained and include their own introduction, development, and concluding remarks.

Part I – Optimal control of thin-film flow on a flexible topography. Chapter 2 reviews the foundations of PDE-constrained optimal control, including admissible controls, functional differentiation, and weak optimality conditions, together with standard numerical approaches. Chapter 3 develops the open-loop control of thin-film flow over a flexible substrate by formulating the coupled lubrication–elastic model and the associated optimality problem, and by illustrating its behaviour through numerical examples.

Part II – Actuator design and learned feedback control. Chapter 4 reviews neural-network-based approaches for PDE solution and control, focusing on physics-informed neural networks as a link between analytical and learning-based methods. Chapter 5 extends the thin-film control framework to actuator design and introduces a one-shot formulation that determines actuator placement and control inputs simultaneously. Chapter 6 develops a learned feedback-control approach based on value-function approximation, combining actuator design with a neural representation of the Hamilton–Jacobi–Bellman equation.

Finally, Chapter 7 summarises the principal findings and outlines possible directions for future work.

Part I

Optimal Control of Thin-Film Flow on a Flexible Topography

Chapter 2

Introduction to Optimal Control of PDEs

This chapter reviews the mathematical and numerical foundations of optimal control problems governed by partial differential equations, using the one-dimensional heat equation as a representative example. We present the continuous and discrete formulations of the first-order optimality system. The analysis covers the functional framework, weak formulation, and derivation of adjoint equations. Finite-element discretisation in space is employed to obtain the discrete system. Two numerical strategies are discussed. The optimise-then-discretise and discretise-then-optimise approaches are highlighted, with their equivalence demonstrated for linear-quadratic problems and their differences in general settings. We then consider iterative methods for PDE-constrained optimisation, focusing on the gradient-descent algorithm with backtracking line search, and briefly compare it with all-at-once formulations.

2.1 Introduction

The objective of this chapter is to deliver a brief mathematical introduction to control systems governed by PDEs. We focus on the linear one-dimensional heat equation as a representative example, though the same framework extends to a broad class of (nonlinear) PDEs. Most of the material presented here is well-known and can be found in several textbooks. In particular, we mention [8, 40, 46, 66, 98, 112, 154, 155].

Let $\Omega \subset \mathbb{R}$ be a bounded domain and let $T > 0$ denote the final time. We define the space-time domain $Q := \Omega \times (0, T)$, where $x \in \Omega$ represents the spatial coordinate and $t \in (0, T)$ denotes time. We then consider the linear parabolic equation: find $h : Q \rightarrow \mathbb{R}$ such that

$$\begin{cases} \partial_t h - \partial_{xx} h = f(x, t), & (x, t) \in Q, \\ h = 0, & (x, t) \in \partial\Omega \times (0, T), \\ h(\cdot, 0) = h_0, & x \in \Omega, \end{cases} \quad (2.1)$$

where $h(x, t)$ is the temperature and f is the control input. The symbols ∂_t , ∂_x and ∂_{xx} denote differentiation with respect to time t and space x , and will occasionally be abbreviated by h_t , h_x and h_{xx} . The precise functional setting for h and f will be introduced in Section 2.4.

If a control $f(x, t)$ in (2.1) exists such that $h = \bar{h}$ for some target state $\bar{h}$, the system is said to be exactly controllable. In practice, this is not always achievable due to actuation limits, nonlinearities, geometric constraints or energy bounds (see, for example, [19, 55] for further details on controllability). Instead, an optimal control problem can be formulated by minimising a cost functional. This functional balances tracking a desired trajectory $\bar{h}$ whilst penalising the control effort [112]. In general, this leads to a quadratic functional of the form

$$\mathcal{J}(h, f) = \frac{1}{2} \|h - \bar{h}\|_{L^2(Q)}^2 + \frac{\alpha}{2} \|f\|_U^2, \quad (2.2)$$

where $\alpha > 0$ is a regularisation parameter ensuring well-posedness and preventing singular (non-unique or unstable) controls by penalising large control inputs [129, Sec. 2.2]. The precise definition of the norms and a control space U depends on the chosen control design (for example, distributed, localised, and boundary control). This will be discussed in the next section.

Thus, the full optimal control problem is

$$\min_{h,f} \mathcal{J}(h, f) \quad \text{subject to (2.1).} \quad (2.3)$$

Control actions can be applied in either an open or a feedback (closed) loop [11, 112, pp. 73]. The former is prescribed in advance and applied without adjustment during the evolution, whilst the latter depends on the current state and is updated as the system evolves. In this part of the thesis, we focus on open-loop control, whilst the feedback control will be addressed later in Part II.

The remainder of this chapter is organised as follows. Section 2.2 introduces the control designs and admissible formulations. Section 2.3 recalls differentiation in function spaces. Section 2.4 presents the functional setting and the weak formulation of the heat equation. Section 2.5 derives the first-order optimality system. Finally, Section 2.6 discusses solution strategies based on discretisation and their numerical realisation.

2.2 Control Designs

Control problems for PDEs differ according to how the control enters the system [8, 112]. The aim is to choose f so that the state h reaches a desired temperature $\bar{h}$ at the final time whilst limiting control effort. This balance is described by the quadratic cost functional

$$\mathcal{J}(h, f) = \frac{1}{2} \|h(T) - \bar{h}\|_{L^2(\Omega)}^2 + \frac{\alpha}{2} \int_0^T \|f\|_U^2 dt, \quad (2.4)$$

where the control space U depends on the actuation type [68].

If heating or cooling can be applied throughout the domain, the control enters the equation as a distributed source [112, Sec. 1.2]:

$$\partial_t h - \partial_{xx} h = f(x, t), \quad f \in L^2(\Omega \times (0, T)). \quad (2.5)$$

This case is known as distributed control, with $U = L^2(\Omega \times (0, T))$.

In many applications, the control acts on a subregion $\omega \subset \Omega$, leading to the model

$$\partial_t h - \partial_{xx} h = \chi_\omega(x) f(x, t), \quad f \in L^2(\omega \times (0, T)), \quad (2.6)$$

where χ_ω is the characteristic function of ω ; here $U = L^2(\omega)$. This setting is referred to as localised control [81].

The control may also act through the boundary. For example, prescribing a Neumann condition at $x = L$ gives

$$\begin{cases} \partial_t h - \partial_{xx} h = 0, & x \in (0, L), t \in (0, T), \\ h(0, t) = 0, \quad -\partial_x h(L, t) = f(t), & t \in (0, T), \\ h(x, 0) = h_0(x), & x \in (0, L), \end{cases} \quad (2.7)$$

with $f \in L^2(0, T)$ and $U = L^2(0, T)$. This configuration is called boundary control [155, Sec. 3.2].

In all the above formulations, it is also possible to impose limits on the control to reflect actuator and physical constraints [69, Ch. 1]. In the unconstrained setting, the control may take arbitrary values within the control space U , which is often defined as a Lebesgue space or Hilbert space like $U = L^2(\Omega \times (0, T))$ [68]. In contrast, constrained formulations restrict their range to satisfy $f_{\min} \leq f(x, t) \leq f_{\max}$, which defines the admissible set

$$\mathcal{U}_{\text{ad}} = \{ f \in U : f_{\min} \leq f(x, t) \leq f_{\max} \}. \quad (2.8)$$

The above optimal control problems are then solved over $\mathcal{U}_{\text{ad}}$ [8].

2.3 Differentiation in function spaces

This section reviews essential concepts from functional analysis and optimal control that underpin the analytical treatment of PDE-constrained optimisation. The discussion follows mainly [155, Sec. 2.4-2.7], and similar material can be found in [8, 40].

Directional derivative. Let U, V be Banach spaces. Denote by $\mathcal{L}(U, V)$ the space of bounded linear operators from U to V . For a mapping $F : U \rightarrow V$, the directional derivative of F at $u \in U$ in direction $g \in U$ is

$$\delta F(u; g) = \lim_{\theta \rightarrow 0} \frac{F(u + \theta g) - F(u)}{\theta}, \quad (2.9)$$

if the limit exists in V for all $g \in U$.

Gâteaux derivative. The mapping F is said to be Gâteaux differentiable at u if the directional derivative $\delta F(u; g)$ exists for all $g \in U$ and is linear and continuous in g . In that case, there exists $F'_G(u) \in \mathcal{L}(U, V)$ such that

$$\delta F(u; g) = F'_G(u)g. \quad (2.10)$$

Fréchet derivative. The mapping F is Fréchet differentiable at u if there exists $F'(u) \in \mathcal{L}(U, V)$ satisfying

$$F(u+g) = F(u) + F'(u)g + r(u, g), \quad \frac{\|r(u, g)\|_V}{\|g\|_U} \rightarrow 0 \quad \text{as } \|g\|_U \rightarrow 0. \quad (2.11)$$

Fréchet differentiability implies Gâteaux differentiability, but not conversely [155, Sec. 2.6].

Gradients in Hilbert spaces. In Hilbert spaces, derivatives can be identified with elements of the space itself via the Riesz representation theorem. Let U be a Hilbert space with inner product $(\cdot, \cdot)_U$. For a Fréchet differentiable mapping $F : U \rightarrow \mathbb{R}$, the derivative $F'(u) \in U^*$ corresponds to a unique element $\nabla F(u) \in U$ such that

$$\delta F(u; g) = (\nabla F(u), g)_U \quad \forall g \in U. \quad (2.12)$$

This element $\nabla F(u)$ is the gradient of F at u .

Example 1. Let U be a Hilbert space with inner product $(\cdot, \cdot)_U$ and define $F(u) = \|u\|_U^2$. Then

$$\delta F(u; g) = 2(u, g)_U, \quad \nabla F(u) = 2u. \quad (2.13)$$

Hence, F is Fréchet differentiable with gradient $2u$.

Adjoint operator. The concept of an adjoint operator links derivatives in different spaces. For Hilbert spaces U and V , and any bounded linear operator $A \in \mathcal{L}(U, V)$, the adjoint operator $A^* : V \rightarrow U$ is defined by

$$(Au, v)_V = (u, A^*v)_U \quad \forall u \in U, v \in V. \quad (2.14)$$

The adjoint A^* provides a way to transfer information between dual spaces and is essential in gradient representations for optimisation problems [155, Sec 2.7].

Example 2. Let $S : U \rightarrow H$ be a bounded linear operator between Hilbert spaces and fix $u_d \in H$. Consider $E(z) = \|Sz - u_d\|_H^2$. By the chain rule and the definition of the adjoint,

$$\delta E(z; h) = 2(Sz - u_d, Sh)_H, \quad \nabla E(z) = 2S^*(Sz - u_d), \quad (2.15)$$

where $S^* : H \rightarrow U$ is the adjoint of S . This illustrates the role of adjoints in the computation of gradients, a central step in PDE-constrained optimisation [155, Theorem 2.20].

Convexity [155, Sec 2.4]. A functional $J : U \rightarrow \mathbb{R}$ is convex if

$$J(\lambda u_1 + (1 - \lambda)u_2) \leq \lambda J(u_1) + (1 - \lambda)J(u_2) \quad \forall u_1, u_2 \in U, \lambda \in [0, 1], \quad (2.16)$$

and it is strictly convex if equality holds only when $u_1 = u_2$.

The quadratic cost functionals considered in this chapter are strictly convex in the control variable. Together with the linearity of the state equation, this ensures that the corresponding optimal control problems are well posed. Well-posedness of our problem in this chapter (2.3) is established in [112, Ch. 7] for the distributed case and in [81] for the localised case. In the next section, we define the functional spaces and derive the weak formulation of the heat equation (2.1).

2.4 Functional setting and weak formulation

We consider a weak formulation of the state problem (2.1), wherein h and f are considered independent variables. We start by establishing the functional context for the problem considered, which can be found in most functional analysis and control PDE books, for example [8, 40, 112, 158]. We introduce the Hilbert spaces $\hat{L}$ and $\mathcal{V}$ of functions depending on time $t \in (0, T)$, with values in $L^2(\Omega)$ and $H^1(\Omega)$, respectively, where $H^1(\Omega)$ denotes the Sobolev space of functions in $L^2(\Omega)$ with square-integrable first derivatives, together with their associated norms:

$$\hat{L} := L^2(0, T; L^2(\Omega)), \quad \|v\|_{\hat{L}}^2 := \int_0^T \|v(t)\|_{L^2(\Omega)}^2 dt, \quad (2.17a)$$

$$\mathcal{V} := L^2(0, T; H^1(\Omega)), \quad \|v\|_{\mathcal{V}}^2 := \int_0^T \|v(t)\|_{H^1(\Omega)}^2 dt, \quad (2.17b)$$

The space $\hat{L}$ is fit for f . A suitable space for the state variable h is then defined by

$$\mathcal{W} := \{v \in \mathcal{V} : v_t \in \mathcal{V}^* := L^2(0, T; H^1(\Omega)^*)\}, \quad (2.18)$$

provided with the norm

$$\|v\|_{\mathcal{W}}^2 := \|v\|_{\mathcal{V}}^2 + \|v_t\|_{\mathcal{V}^*}^2.$$

Note that $\mathcal{W} \subset \mathcal{V}$. For $v \in \mathcal{W}$, we have $v \in C([0, T]; L^2(\Omega))$ by a classical embedding result [120, p. 473]. Then, for $v, w \in \mathcal{W}$, the integration by parts formula holds:

$$\int_0^T \langle w_t(t), v(t) \rangle dt = (w(T), v(T)) - (w(0), v(0)) - \int_0^T \langle v_t(t), w(t) \rangle dt, \quad (2.19)$$

where $(\cdot, \cdot)$ denotes the spatial $L^2(\Omega)$ inner product. Here $\langle \cdot, \cdot \rangle$ denotes the duality pairing between $\mathcal{V}^*$ and $\mathcal{V}$, used to represent the action of the weak time derivative h_t (which belongs to $\mathcal{V}^*$) on a test function $v \in \mathcal{V}$; if $h_t \in L^2(\Omega)$, this coincides with the L^2 inner product [40, 155, Sec. 2.1, Sec. 2.7].

To impose the initial conditions for h , we establish

$$\mathcal{W}_{v_0} := \{v \in \mathcal{W} : v(0) = v_0\}, \quad (2.20)$$

for any $v_0 \in L$.

Now, we can outline the weak formulation of (2.1): given $f \in L^2(Q)$, find $h \in \mathcal{W}_{h_0}$ such that

$$\int_0^T \langle \partial_t h(t), v(t) \rangle dt + \int_0^T (\partial_x h(t), \partial_x v(t)) dt = \int_0^T (f(t), v(t)) dt \quad \forall v \in \mathcal{V}. \quad (2.21)$$

We write $h(t)$ and $v(t)$ instead of $h(x, t)$ and $v(x, t)$, so $(h_x(t), v_x(t))$ denotes the spatial $L^2(\Omega)$ inner product [112, Sec. 7.4]. This convention treats time as the evolution variable and the solution values in a spatial Hilbert space,

leading to a compact weak formulation [155, Sec. 3.4].

Formulation (2.21) relaxes differentiability requirements on the solution, thereby ensuring existence and uniqueness in Sobolev spaces [40, 155, Sec. 2.2, Sec. 2.3]. It forms the basis for numerical discretisation methods such as Finite Element Method (FEM) [98, 135]. It also provides the natural framework for deriving the first-order necessary optimality conditions, as we will discuss next.

2.5 Optimality conditions

We now derive the first-order optimality system using the Lagrangian functional associated with the distributed control problem (2.3). This functional $\mathcal{L} : \mathcal{W}_{v_0} \times L^2(Q) \times \mathcal{W}_{v_T} \rightarrow \mathbb{R}$, incorporates the cost (2.4) and the weak state equation (2.21), and reads

$$\mathcal{L}(h, f, p) = \mathcal{J}(h, f) + \int_0^T (\langle \partial_t h, p \rangle + (\partial_x h, \partial_x p) - (f, p)) dt, \quad (2.22)$$

where $p \in \mathcal{W}_{v_T}$ is the adjoint variable acting as a Lagrange multiplier for the state constraint [40]. The Lagrangian reformulates the constrained optimisation problem into an unconstrained one, so that necessary conditions are obtained by differentiation with respect to h , p , and f . This formal approach provides a systematic way to derive the adjoint equation and the optimality system [155, Ch. 1, Ch. 6]. The next step is to compute the Gâteaux derivatives of $\mathcal{L}$ with respect to h , p , and f .

The derivative with respect to p in direction $\delta p \in \mathcal{W}_{v_T}$ is

$$\begin{aligned} \delta_p \mathcal{L}(h, f, p; \delta p) &= \lim_{\epsilon \rightarrow 0} \frac{\mathcal{L}(h, f, p + \epsilon \delta p) - \mathcal{L}(h, f, p)}{\epsilon} \\ &= \int_0^T (\langle \partial_t h, \delta p \rangle + (\partial_x h, \partial_x \delta p) - (f, \delta p)) dt, \end{aligned} \quad (2.23)$$

which is precisely the weak form of the state equation (2.1) as expressed in equation (2.21).

The derivative with respect to h in direction $\delta h \in \mathcal{W}_0$ is

$$\begin{aligned}\delta_h \mathcal{L}(h, f, p; \delta h) &= \lim_{\epsilon \rightarrow 0} \frac{\mathcal{L}(h + \epsilon \delta h, f, p) - \mathcal{L}(h, f, p)}{\epsilon} \\ &= (h - \bar{h}, \delta h)_{L^2(\Omega)} + \int_0^T \langle \partial_t \delta h, p \rangle dt + \int_0^T (\partial_x \delta h, \partial_x p) dt.\end{aligned}\tag{2.24}$$

Using integration-by-parts,

$$\int_0^T \langle \partial_t \delta h, p \rangle dt = (p(T), \delta h(T))_{L^2(\Omega)} - \int_0^T \langle \partial_t p, \delta h \rangle dt, \tag{2.25}$$

and applying Green's identity (see [112, p. 382]) in space, subject to homogeneous Dirichlet boundary conditions

$$\int_0^T (\partial_x \delta h, \partial_x p) dt = - \int_0^T (\delta h, \partial_{xx} p) dt, \tag{2.26}$$

this expression becomes

$$\delta_h \mathcal{L}(h, f, p; \delta h) = (p(T), \delta h(T))_{L^2(\Omega)} + \int_0^T (h - \bar{h} - \partial_t p - \partial_{xx} p, \delta h) dt. \tag{2.27}$$

The derivative with respect to f in direction $\delta f \in L^2(Q)$ is

$$\delta_f \mathcal{L}(h, f, p; \delta f) = \lim_{\epsilon \rightarrow 0} \frac{\mathcal{L}(h, f + \epsilon \delta f, p) - \mathcal{L}(h, f, p)}{\epsilon} = (\alpha f - p, \delta f)_{L^2(Q)}. \tag{2.28}$$

We now impose stationarity conditions of the Lagrangian by setting (2.23), (2.27), and (2.28) to vanish for all admissible variations [8]. We then obtain

the following optimality system in strong form:

$$\begin{cases} \partial_t h - \partial_{xx} h = f & \text{in } Q, \\ h = 0 & \text{on } \partial\Omega \times (0, T), \\ h(0) = h_0 & \text{in } \Omega, \end{cases} \quad (2.29a)$$

$$\begin{cases} -\partial_t p - \partial_{xx} p = \bar{h} - h & \text{in } Q, \\ p = 0 & \text{on } \partial\Omega \times (0, T), \\ p(T) = 0 & \text{in } \Omega. \end{cases} \quad (2.29b)$$

together with the control condition

$$f = \frac{1}{\alpha} p \quad \text{in } Q, \quad (2.30)$$

in the unconstrained case, and in the constrained case, the variational inequality

$$\int_Q (\alpha f^* - p^*)(g - f^*) \, dx \, dt \geq 0 \quad \forall g \in \mathcal{U}_{\text{ad}}. \quad (2.31)$$

This structure is typical of parabolic optimal control problems. The state equation (2.29a) evolves forward from the initial condition, the adjoint equation (2.29b) evolves backwards from its terminal condition, and the control (2.30) links the two.

The continuous optimality system (2.29)-(2.30) is thus established. In practice, its numerical solution requires discretisation and optimisation techniques, which we discuss next.

2.6 Numerical solution strategies

We now discuss strategies for numerical solutions. Two main approaches are employed in PDE-constrained optimisation [36]. The optimise-then-discretise strategy, where the continuous optimality system is first derived and then

discretised. In contrast, the discretise-then-optimise strategy, where the state equation and the cost functional are discretised first and optimality conditions are then derived at the discrete level [112, Ch. 8].

For the spatial discretisation, we adopt the Finite Element Method (FEM) [98]. Let $\mathcal{V}_h \subset \mathcal{V}$ and $U_h \subset L^2(\Omega)$ denote finite-dimensional subspaces for the state and control, respectively, and let $\{\hat{\phi}_i\}_{i=1}^N$ and $\{\psi_j\}_{j=1}^{N_u}$ be their corresponding basis functions. The discrete state, adjoint, and control are represented as linear combinations of these basis functions with time-dependent coefficients:

$$h_h(x, t) = \sum_{i=1}^N h_{h,i}(t) \hat{\phi}_i(x), \quad (2.32)$$

$$p_h(x, t) = \sum_{i=1}^N p_{h,i}(t) \hat{\phi}_i(x), \quad (2.33)$$

$$f_h(x, t) = \sum_{j=1}^{N_u} f_{h,j}(t) \psi_j(x). \quad (2.34)$$

The associated coefficient vectors are

$$\mathbf{h}(t) = (h_{h,1}(t), \dots, h_{h,N}(t))^{\top}, \quad (2.35)$$

$$\mathbf{p}(t) = (p_{h,1}(t), \dots, p_{h,N}(t))^{\top}, \quad (2.36)$$

$$\mathbf{f}(t) = (f_{h,1}(t), \dots, f_{h,N_u}(t))^{\top}, \quad (2.37)$$

each collecting the nodal unknowns corresponding to the state, adjoint, or control, respectively. This corresponds to a standard Galerkin finite element discretisation in space. More precisely, by inserting the finite element expansions into the weak formulation and then testing with the basis functions, one obtains a system of ordinary differential equations for the coefficient vectors [98].

Therefore, the finite element matrices are defined by

$$\begin{aligned} M_{ij} &= (\hat{\phi}_i, \hat{\phi}_j)_{L^2(\Omega)}, & K_{ij} &= (\partial_x \hat{\phi}_i, \partial_x \hat{\phi}_j)_{L^2(\Omega)}, \\ B_{ij} &= (\psi_j, \hat{\phi}_i)_{L^2(\Omega)}, & W_{ij} &= (\psi_i, \psi_j)_{L^2(\Omega)}. \end{aligned} \quad (2.38)$$

where $M \in \mathbb{R}^{N \times N}$ is the mass matrix, $K \in \mathbb{R}^{N \times N}$ the stiffness matrix, $B \in \mathbb{R}^{N \times N_u}$ the control-to-state matrix, and $W \in \mathbb{R}^{N_u \times N_u}$ is the control-space mass matrix in the discrete cost functional.

In the optimise-then-discretise approach, applying the above discretisation to the continuous optimality system (2.29)-(2.30), gives

$$M\dot{\mathbf{h}}(t) + K\mathbf{h}(t) = B\mathbf{f}(t), \quad (2.39a)$$

$$-M\dot{\mathbf{p}}(t) + K\mathbf{p}(t) = M(\bar{\mathbf{h}}(t) - \mathbf{h}(t)), \quad \mathbf{p}(T) = \mathbf{0}, \quad (2.39b)$$

$$\alpha W\mathbf{f}(t) = B^\top \mathbf{p}(t). \quad (2.39c)$$

Equations (2.39) form a coupled forward-backward ODE system. A time discretisation, for instance backward Euler or Crank-Nicolson, yields the fully discrete problem [98, Ch. 5].

In the discretise-then-optimise approach, one first discretises the state equation and the cost functional. The discrete cost associated with (2.4) is

$$J_h(\mathbf{h}, \mathbf{f}) = \frac{1}{2} \int_0^T (\mathbf{h}(t) - \bar{\mathbf{h}}(t))^\top M (\mathbf{h}(t) - \bar{\mathbf{h}}(t)) dt + \frac{\alpha}{2} \int_0^T \mathbf{f}(t)^\top W \mathbf{f}(t) dt, \quad (2.40)$$

subject to (2.39a), where $\bar{\mathbf{h}}(t)$ denotes the projection of the desired state onto the finite element space. Since the problem is linear-quadratic and the same Galerkin space is used for state and adjoint, the optimality system obtained by differentiating J_h coincides with (2.39) [112, Sec.6.2.3]. However, this equivalence does not hold in general [112, Ch.6]. If stabilisation techniques are used (for example in advection-dominated problems), the discrete operators for state and adjoint differ and the two strategies produce distinct optimality systems [36]. For time-dependent problems, differences also arise from the quadrature used to approximate time integrals in the cost, which affects only the discretise-then-optimise formulation [112, Sec 8.1].

The choice between them depends on the specific control problem, the desired accuracy, computational resources, and the availability of numerical

tools. For example, the optimise-then-discretise strategy preserves strong consistency with the continuous formulation and provides accurate adjoint approximations [36]. Also, it supports the mesh-independence principle whereby discrete convergence reflects the continuous problem [40, Ch. 4]. Discretise-then-optimise may yield symmetric linear systems that are favourable for iterative solvers [135], but the adjoint accuracy is generally inferior [36]. Further discussion can be found in [112, Chs. 6,8,9] and [40, Ch. 4].

2.7 Numerical methods

We now discuss two main strategies for the numerical solution of the optimal control problem (2.3): the all-at-once approach, which solves the discrete optimality system as a single coupled problem [112]; and iterative methods, which eliminate the PDE constraint and minimise a reduced cost functional [66].

2.7.1 All-at-once methods

All-at-once methods, also referred to as one-shot approaches, treat the optimisation problem by keeping the PDE constraint explicitly in the formulation. All variables (the state, adjoint, and control variables) of the optimal control problem (2.39) are treated simultaneously as independent variables within a single large system [112, Sec. 6.1]. This avoids the repeated solution of the state equation within an outer optimisation loop [66].

We recall the semi-discrete system (2.39) obtained after finite element discretisation in space, with matrices M , K , B , and W defined in (2.38). Applying a backward Euler scheme with time step Δt gives, for each $n = 1, \dots, N_t$,

$$\begin{cases} \frac{1}{\Delta t} M(\mathbf{h}^n - \mathbf{h}^{n-1}) + K \mathbf{h}^n = B \mathbf{f}^n, \\ -\frac{1}{\Delta t} M(\mathbf{p}^n - \mathbf{p}^{n+1}) + K \mathbf{p}^n = M(\bar{\mathbf{h}}^n - \mathbf{h}^n), \\ \alpha W \mathbf{f}^n = B^\top \mathbf{p}^n, \end{cases} \quad (2.41)$$

with $\mathbf{h}^0$ prescribed by the initial condition and $\mathbf{p}^{N_t} = \mathbf{0}$ by the terminal condition. For the unconstrained linear-quadratic problem, the optimality condition is given by the algebraic relation

$$\alpha W \mathbf{f}^n = B^\top \mathbf{p}^n.$$

If control constraints are imposed, this relation is replaced by the corresponding discrete variational inequality, or equivalently by a projection formula [40].

In the all-at-once formulation, instead of solving (2.41) sequentially at each time step, the state, adjoint, and control variables are coupled into a single algebraic system. For one time step, this can be written as

$$\begin{bmatrix} \frac{1}{\Delta t}M + K & -B & 0 \\ 0 & \alpha W & -B^\top \\ -M & 0 & K - \frac{1}{\Delta t}M \end{bmatrix} \begin{bmatrix} \mathbf{h}^n \\ \mathbf{f}^n \\ \mathbf{p}^n \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}M \mathbf{h}^{n-1} \\ 0 \\ M \bar{\mathbf{h}}^n - \frac{1}{\Delta t}M \mathbf{p}^{n+1} \end{bmatrix}, \quad (2.42)$$

where $M, K \in \mathbb{R}^{N \times N}$, $W \in \mathbb{R}^{N_u \times N_u}$, and $B \in \mathbb{R}^{N \times N_u}$. A similar block structure arises when all time steps are assembled into a global system. For time-dependent problems, however, the all-at-once formulation becomes increasingly challenging, as all time levels must be coupled simultaneously. This leads to systems whose dimension scales with the product of the number of time steps, the spatial degrees of freedom, and the number of optimisation variables [112, 130, Secs. 8.4 and 4.1]. This can be challenging because coupling all time levels simultaneously yields an extensive algebraic system, often exceeding the storage and computational limits of direct solvers and standard computing resources [112, Sec. 8.4].

2.7.2 Iterative methods

In this approach, the PDE constraint is satisfied implicitly by considering the state variable h as a function of the control variable f [66]. This effectively

reduces the optimisation problem to minimising the following reduced cost functional

$$\hat{\mathcal{J}}(f) = \frac{1}{2} \|h(f) - \bar{h}\|_{L^2(Q)}^2 + \frac{\alpha}{2} \|f\|_{L^2(Q)}^2, \quad f \in U, \quad (2.43)$$

which is only over the control variable f [112, Ch. 6]. Here $h(f)$ is computed by solving the state equation once f is available. Recalling the definition of the gradient in a Hilbert space from Section 2.3, the reduced gradient is characterised by the relation

$$\hat{\mathcal{J}}'(f)v = (\nabla \hat{\mathcal{J}}(f), v)_{L^2(Q)} \quad \forall v \in U. \quad (2.44)$$

Moreover, as illustrated by Example 2 in Section 2.3, the adjoint variable provides a convenient representation of the derivative of the reduced functional [155, p. 74]. Using the optimality condition (2.30), we therefore obtain the reduced gradient

$$\nabla \hat{\mathcal{J}}(f) = \alpha f - p. \quad (2.45)$$

Discretising (2.45) using the FEM as in Section 2.6, with matrices B and W from (2.38), this becomes

$$\nabla \hat{J}_h(\mathbf{f}) = \alpha W \mathbf{f} - B^\top \mathbf{p}. \quad (2.46)$$

Given an initial guess $\mathbf{f}_0 \in \mathbb{R}^{N_u}$, descent methods generate a sequence $\{\mathbf{f}_k\}$ by

$$\mathbf{f}_{k+1} = \mathbf{f}_k + \tau_k \mathbf{d}_k, \quad k = 0, 1, 2, \dots \quad (2.47)$$

where $\mathbf{d}_k$ is a descent direction depending on the gradient and $\tau_k > 0$ a step size chosen to ensure decrease of the cost [112, Ch. 8]. The simplest choice is steepest descent, which sets

$$\mathbf{d}_k = -\nabla \hat{J}_h(\mathbf{f}_k), \quad (2.48)$$

so that

$$\mathbf{f}_{k+1} = \mathbf{f}_k - \tau_k \nabla \hat{J}_h(\mathbf{f}_k). \quad (2.49)$$

The gradient points in the direction of maximum increase of $\hat{J}_h$, and its negative therefore defines the steepest descent direction [40, Ch. 4]. Gradient descent is globally convergent but only of first order, relying solely on gradient information [66]. Second-order schemes such as Newton or quasi-Newton achieve faster local convergence, but at higher computational cost per iteration [40, 112, Ch. 3, Ch. 4]. Thanks to their simplicity and robustness, gradient methods remain widely used, particularly in large or nonlinear PDE control problems [155, Ch. 2].

The step size τ_k in (2.49) can be defined by exact line search,

$$\tau_k = \arg \min_{\tau > 0} \hat{J}_h(\mathbf{f}_k - \tau \nabla \hat{J}_h(\mathbf{f}_k)). \quad (2.50)$$

For more general problems, exact line search is too costly, and inexact line-search strategies are used instead [155, Sec. 2.12]. A widely used strategy is the backtracking Armijo rule. Starting from $\tau = \tau_0$, the step is reduced by $\tau \leftarrow \rho \tau$ with $\rho \in (0, 1)$ until

$$\hat{\mathcal{J}}(f^k - \tau g^k) \leq \hat{\mathcal{J}}(f^k) - c \tau \|g^k\|_{L^2(Q)}^2, \quad g^k = \nabla \hat{\mathcal{J}}(f^k), \quad (2.51)$$

is satisfied for some $c \in (0, 1)$ [112, Sec. 6.2]. Other approaches exist, such as fixed step rules or the Barzilai-Borwein method [15, 40], which can accelerate convergence in certain settings. In this thesis, the step size is selected using a backtracking Armijo rule. Thus, each iteration consists of solving the state equation (2.29a) for the current control, then the adjoint equation (2.29b), computing the reduced gradient from (2.45), and finally updating the control according to (2.49). A simple pseudocode for this iterative procedure is given in Algorithm 1.

Algorithm 1 Gradient descent with backtracking line search

- 1: Choose initial guess $f^0 \in L^2(Q)$.
 - 2: **for** $k = 0, 1, 2, \dots$ until convergence **do**
 - 3: Solve the state equation (2.29a) for h^k .
 - 4: Solve the adjoint equation (2.29b) for p^k .
 - 5: Compute the gradient $g^k = \alpha f^k - p^k$.
 - 6: Choose step length τ_k , for example, using backtracking Armijo rule.
 - 7: Update the control $f^{k+1} = f^k - \tau_k g^k$.
 - 8: **end for**
-

Compared with the all-at-once formulation, gradient-based methods are often preferred for their simplicity, global convergence, and low memory requirements. They are particularly suitable for time-dependent problems, where all-at-once systems can become prohibitively large, and for implementations that reuse existing PDE solvers without modification. A more detailed discussion of these approaches can be found in [66] and in the textbooks [40, 112].

2.8 Conclusion

In this chapter, we reviewed the basic framework for PDE-constrained optimisation using the linear heat equation as a representative model. The functional analytic tools, together with the weak and optimality system, were developed in a setting where the state equation is linear and coercive. This structure also enabled a clear presentation of the optimise-then-discretise approach, reduced gradients, and gradient-based iterative methods without additional technical complications.

In the next chapter, we turn to the optimal control of a nonlinear thin-film flow over a flexible topography. This model is considerably more complex: it couples multiple physical fields, involves nonlinear and higher-order differential operators, and captures behaviours such as rupture and coalescence that are absent in the heat equation. These features require a more careful analytical and numerical treatment. In particular, deriving the optimality system leads to coupled forward-backward nonlinear PDEs, and their discretisation must

balance accuracy with stability in the presence of strong nonlinearities.

Nevertheless, the framework developed in this chapter provides the basis for the analysis and numerical treatment in the next chapter. In particular, the thin-film control problem follows the same overall structure, namely an optimise–then–discretise formulation combined with a reduced-gradient strategy, but extends these ideas to a significantly more complex PDE model. The next chapter therefore builds on this framework and adapts it to the nonlinear setting.

Chapter 3

Optimal Control the Thin-Film Flow on a Flexible Topography

This chapter presents a mathematical model for the optimal control of thin-film flows over a flexible topography influenced by an external force. Our thin-film model allows for the rupture of films as well as the coalescence of bubbles. The objective is to find the optimal distributed force acting on the topography that minimises the differences between actual and desired thin-film profiles. A nonlinear lubrication equation governing the fluid dynamics and appropriate functional settings for this model are presented. It is also shown that this system satisfies a global energy-dissipation law for a suitable energy functional. Optimality conditions are derived for the solution of the minimisation problem of a specified cost function across a time horizon. These conditions are formulated at a continuous level as system of coupled, forward-backward PDEs, which are subsequently discretised for numerical investigation. To ensure computational efficiency and stability, first-order Implicit-Explicit (IMEX) time-stepping schemes are employed to handle the nonlinearities in the model, and a reduced gradient descent algorithm is applied to obtain a numerical approximation of the optimal control signal. Numerical results illustrate that controlling the thin film, even during rupture, achieves a precise film profile. This control strategy accelerates convergence towards a steady state, reduces instabilities, stabilises dewetting processes, and meets the desired profile specifications.

3.1 Introduction

Controlling thin-film flow has gained significant interest across a range of applications [47, 92, 119, 142, 161]. For example, coating processes favour stable, flat-flow films [152], whereas applications in heat and mass transfer often require interfacial deformations [48]. Motivation stems from the increasing number of thin-film devices manufactured across diverse industries such as microelectronics, displays, optical storage, and microfluidic devices [47, 92, 119, 141, 142]. Of particular interest are inverted film configurations where the liquid resides on the underside of a substrate [27]. Such configurations arise in liquid film coating in cooling towers [137], environmental applications such as glacier hydrology and cave pattern morphogenesis [32], and biomedical applications such as microbicide epithelial coatings and blood pattern analysis [75, 79]. In these inverted geometries, gravity destabilises the film and control of the thin layer deposition process is required to achieve desired free surface profiles [41]. However, efficient control of interfacial shape or surface area has been largely unexplored until recently, with most efforts focusing on varying external mechanisms like electric fields [7, 156], including a rigid topography [52, 53, 139, 140, 157] or surfactants [26], rather than using mathematically optimal or robust approaches [161].

The mathematical model of thin films is based on the lubrication approximation simplified from the Navier-Stokes equations [136], resulting in a time-dependent PDE for the height of the film. This model has been extensively studied for its ability to facilitate accurate analytical and numerical calculations [5, 99, 113, 114]. This indicates that the characteristic wavelengths of interfacial deformation are significantly larger than the film thickness ratio [126, 152]. Additionally, when combined with flexible topographies, further dynamics arise due to elastohydrodynamics [113, 114]. Craster and Matar’s review provides a comprehensive overview of these models [37]. Traditional models typically assume rigid substrates, but real-world applications increasingly involve flexible topographies [114].

Our work develops a PDE-constrained optimisation approach for the control of thin-film flows over a flexible topography. Our thin film model allows for the rupture of films as well as the coalescence of bubbles. We focus on finding the distributed

force influencing the flexible topography that minimises a cost function, for example, the difference between the observed height of the thin film and some desired function.

The main contributions of this chapter are as follows:

1. We first present an appropriate functional setting for the PDE system consisting of a thin-film equation coupled to a flexible substrate equation. We show that this system satisfies a global energy-dissipation law for a suitable energy functional. Next, we derive the optimality conditions at the continuous level. To achieve this, we formulate an optimal control problem consisting of the minimisation of a cost functional subject to the nonlinear evolution PDE for the thin-film flow controlled by the bottom topography. First-order optimality conditions are given by forward-backward, state-adjoint PDE system, closed with an optimality condition for the control.
2. Our second main contribution is that we propose a stable time discretisation for the optimality system. In particular, a careful treatment of the nonlinearity in the model (i.e., the disjoining-pressure term) is needed to balance computational efficiency and stability. For this, we explore first-order Implicit-Explicit (IMEX) time-stepping schemes, which decompose the nonlinear term into a contractive (convex) and expansive (concave) component [117, 162]. This scheme can be combined with any spatial discretisation method. We have employed standard continuous finite elements. Finally, gradient descent is used as an iterative method to solve the resulting discrete PDE-constrained optimisation problem [40, 112].

The structure of the rest of the chapter is as follows. Section 3.2 introduces the model for the flow of thin films over a flexible topography and its weak form. Section 3.3 shows the optimal control problem, the first-order necessary optimality conditions, and the gradient descent scheme for the optimisation problem. Section 3.5 presents the time discretisation using the IMEX time-stepping schemes and the numerical results. Finally, Section 3.6 summarises the findings of this work.

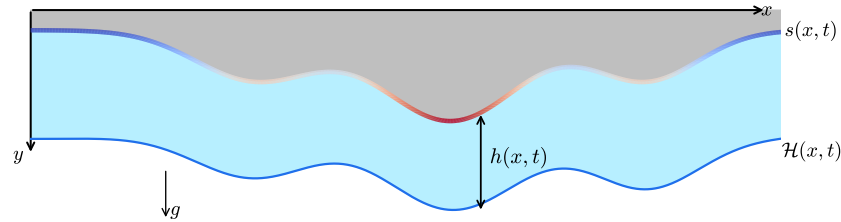


Figure 3.1: Illustration of a two-dimensional fluid flow on a flexible substrate, where $h(x, t)$ denotes the film thickness, $s(x, t)$ is the shape of the topography, and $\mathcal{H}(x, t) = h(x, t) + s(x, t)$ is the height of the one-dimensional free surface. The profile of the topography is controlled by a forcing function $f(x, t)$, which is represented by the colour of the curve, where red indicates downward (positive) forcing and blue indicates upward (negative) forcing. g is the gravity vector.

3.2 Model of thin-film flow over a flexible topography.

In this section, we present a mathematical model comprising two coupled PDEs that describe the evolution of the thin film and flexible substrate. Furthermore, we show the model's functional settings and weak form. The derivation of thin-film flow models on rigid substrates is well known and is based on lubrication theory [18, 157]. The inclusion of the disjoining pressure and a non-inclined flexible substrate is discussed in various forms in the literature [5, 37, 99, 113–115].

3.2.1 State equation

We consider a thin-film flowing on the underside of a flexible substrate; an abstract sketch of this setup is illustrated in Figure 3.1. The coordinate system is defined by (x, y) , with x being the horizontal coordinate and y being the vertical coordinate. We also define t as time. The force driving the flow is gravity g , which we take to act on the increasing y direction to model a film hanging beneath a substrate. This destabilised configuration generates the rich dynamics in which our control can play a meaningful role.

The following system governs the evolution of a thin film over a flexible topography, i.e.,

$$\mathcal{H}(x, t) := h(x, t) + s(x, t), \quad (3.1)$$

where $\mathcal{H}(x, t)$ is the observed film height relative to a flat horizontal reference, $h(x, t) > 0$ is the film thickness, and $s(x, t)$ is the substrate topography height:

$$h_t - \left(\frac{1}{3}h^3\mu_x\right)_x = 0, \quad (3.2a)$$

$$\mu = \phi'(h) - \frac{B_o}{C_a}\mathcal{H} - \frac{1}{C_a}\mathcal{H}_{xx}, \quad (3.2b)$$

$$s_t - c^2 s_{xx} - \frac{\gamma}{C_a}(h_{xx} + B_o h) = f(x, t). \quad (3.2c)$$

A derivation of the model equations is included in Appendix A.1. In equation (3.2a), we introduced μ to reduce the fourth-order thin film PDE to a coupled system of second-order equations¹. Equation (3.2b) includes an interface potential $\phi(h)$ capturing molecular interactions between liquid molecules and air [44, 114], which is commonly taken as

$$\phi(h) = -\frac{A}{2}h^{-2}. \quad (3.3)$$

This means there is a disjoining pressure $\Pi(h) = -\phi'(h) = Ah^{-3}$ in the model to allow for the possibility of film rupture. Here, A is the dimensionless Hamaker constant. The second term on the right of (3.2b) is associated with the action of gravity, and the last term reflects the effect of surface tension. Furthermore, C_a and B_o denote the capillary and Bond numbers, respectively. They represent the importance of viscous forces or gravitational forces compared to surface tension forces on a fluid's surface, respectively. See [86] for typical values these parameters can take in real-life scenarios. Equation (3.2c) represents a one-dimensional force balance on the topography, containing a damping term $c^2 s_{xx}$ arising from substrate tension, and the term $h_{xx} + B_o h$ which represents the pressure exerted on the substrate by the film. This formulation also includes an external forcing term f , which enables direct control of the substrate dynamics. The parameters c and γ are dimensionless measures of the topography tension and film damping, respectively [5]

To place our model (3.2) in context, we discuss how it relates with other models derived in the literature. In the absence of disjoining pressure $\phi'(h) = 0$, and when the substrate is flat ($s \equiv 0$) or rigid, equations (3.2a)-(3.2b) reduce to the

¹Furthermore, when integrating μ using the finite element method, we find that the highest derivative is of the first order when we apply test functions.

well-established long-wave thin-film equation on a horizontal substrate, as used in [62, 126] and [78, 157] in rigid substrates. Our wall equation (3.2c) extends existing flexible-wall thin-film models by including an external force $f(x, t)$, which allows the wall to be driven actively rather than responding only to fluid stresses. When $f \equiv 0$, our formulation reduces to a passive tension-damping membrane of the same type used in the lubrication-scale models of Matar et al. [115]. Moreover, it may be considered as a simplified limit of the general plate model of Alexander *et al.* [5] when bending, inertia and foundation stiffness are neglected, leaving only tension and damping.

The system is solved over a time interval $t \in [0, T]$, where T is the final time, and in a finite domain of arbitrary length L , i.e. $x \in \Omega := [0, L]$. The initial conditions are specified by setting

$$h(x, 0) = h_0(x), \quad s(x, 0) = s_0(x). \quad (3.4)$$

Furthermore, homogeneous Neumann boundary conditions are applied at the domain boundaries

$$h_x = \mu_x = s_x = 0, \quad \text{at } x = 0 \text{ and } x = L. \quad (3.5)$$

We now clarify the free-energy dissipation in our model (3.2). Understanding the energy dissipation mechanisms is important for ensuring the stability and physical relevance of the model. We begin by defining the total free-energy functional:

$$\mathcal{E}(h, s) = \int_{\Omega} \left[\phi(h) - \frac{B_o}{2C_a} h^2 + \frac{1}{2C_a} h_x^2 + \frac{c^2}{2\gamma} s_x^2 + \frac{1}{C_a} (h_x s_x - B_o h s) \right] dx, \quad (3.6)$$

which consists of the molecular free energy $\phi(h) - \frac{B_o}{2C_a} h^2$, surface free energy $\frac{1}{2C_a} h_x^2$, elastic free energy $\frac{c^2}{2\gamma} s_x^2$, and fluid–substrate interaction energy $\frac{1}{C_a} (h_x s_x - B_o h s)$. The latter is particularly important for the problem studied here, and describes how the combined variation of free surface and topography affects the energy—more details are provided in Appendix A.2.

This free energy functional satisfies the following law (the full derivation is shown

in Appendix A.2):

$$\underbrace{\frac{d}{dt}\mathcal{E}(h, s)}_{\text{Change in energy}} = - \underbrace{\left(\frac{1}{3} \int_{\Omega} h^3 \mu_x^2 dx + \frac{1}{\gamma} \int_{\Omega} s_t^2 dx \right)}_{\text{Dissipation}} + \underbrace{\frac{1}{\gamma} \int_{\Omega} f s_t dx}_{\text{External work}}, \quad (3.7)$$

where the terms on the right-hand side of (3.7) are the dissipation in the fluid and substrate, and the work done by the external force on the substrate. Note that bracket in the dissipation term is positive (since $h > 0$), hence, in the absence of external work, the total free energy is non-increasing in time. We also note that the Bond number B_o changes sign with g , but it affects only the potential-energy term in (3.6) and not the dissipation. As a result, the system's energy-decay property holds for either orientation of gravity.

It is well known that a free-energy dissipation law such as (3.7) is critical in the study of well-posedness of thin-film flow models, see for example, [22–24, 56, 93]. The well-posedness of system (3.2) is outside of the scope of this work. We believe such a study could be applied to the natural weak formulation of the problem, which we present next.

3.2.2 Weak form of the state equations

We consider a mixed weak formulation of the state system (3.2), wherein h , μ , and s are considered as independent variables. The functional spaces used throughout this chapter are the same as those introduced in Chapter 2 (see Section 2.4, equations (2.17), (2.18), and (2.20)). In particular, we use the Hilbert spaces $\hat{L}$ and $\mathcal{V}$, defined in (2.17), which contain functions of $t \in (0, T)$ with values in $L^2(\Omega)$ and $H^1(\Omega)$, respectively. A suitable space for the variables h and s is the space $\mathcal{W}$, defined earlier in (2.18), together with its subspace $\mathcal{W}_{v_0}$, introduced in (2.20), which allows one to strongly impose the initial conditions (3.4).

The weak formulation of (3.2) reads as follows (recall $\mathcal{H} = h + s$):

Find $(h, \mu, s) \in \mathcal{W}_{h_0} \times \mathcal{V} \times \mathcal{W}_{s_0}$, such that for $t \in (0, T]$,

$$\langle h_t, v \rangle + \frac{1}{3}(h^3 \mu_x, v_x) = 0, \quad \forall v \in H^1(\Omega), \quad (3.8a)$$

$$(\mu, w) - (\phi'(h), w) + \frac{B_o}{C_a}(\mathcal{H}, w) - \frac{1}{C_a}(\mathcal{H}_x, w_x) = 0, \quad \forall w \in H^1(\Omega), \quad (3.8b)$$

$$\langle s_t, \psi \rangle + c^2(s_x, \psi_x) + \frac{\gamma}{C_a}((h_x, \psi_x) - (B_o h, \psi)) = (f, \psi), \quad \forall \psi \in H^1(\Omega). \quad (3.8c)$$

In (3.8a), the first term represents the duality pairing between the time derivative and the test function, $\langle h_t, v \rangle$. Here $\langle \cdot, \cdot \rangle$ denotes the pairing between $H^1(\Omega)^*$ and $H^1(\Omega)$, that is, the natural extension of the $L^2(\Omega)$ inner product when the time derivative h_t belongs to the dual space $H^1(\Omega)^*$ rather than to $L^2(\Omega)$. The notation $(\cdot, \cdot)$ is reserved for the standard $L^2(\Omega)$ inner product [40, 155, Sec. 2.1, Sec. 2.7].

The system (3.8) can also be written as an aggregated space–time weak formulation:

Find $(h, \mu, s) \in \mathcal{W}_{h_0} \times \mathcal{V} \times \mathcal{W}_{s_0}$:

$$\begin{aligned} & \int_0^T \left(\langle h_t, v \rangle + \langle s_t, \psi \rangle - (\phi'(h), w) + \left(\frac{1}{3} h^3 \mu_x, v_x \right) \right) dt \\ & + \mathcal{A}((h, s, \mu), (v, w, \psi)) = \int_0^T (f, \psi) dt, \quad \forall (v, w, \psi) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}, \end{aligned} \quad (3.9a)$$

where the bilinear form $\mathcal{A}$ takes all linear terms:

$$\begin{aligned} \mathcal{A}((h, s, \mu), (v, w, \psi)) := & \int_0^T \left((\mu, w) + \frac{B_o}{C_a}(\mathcal{H}, w) - \frac{1}{C_a}(\mathcal{H}_x, w_x) \right. \\ & \left. + (c^2 s_x, \psi_x) + \frac{\gamma}{C_a}((h_x, \psi_x) - (B_o h, \psi)) \right) dt. \end{aligned} \quad (3.9b)$$

3.2.3 Boundedness of weak form and disjoining pressure assumptions

We now discuss the boundedness of the weak formulation, in particular suitable choices of the disjoining pressure $\phi'(h)$. First note that all the terms in the bilinear form $\mathcal{A}(\cdot, \cdot)$ can be shown to be bounded within the indicated function spaces (by using Cauchy-Schwarz inequalities). Next, observe that the time derivative terms $\langle h_t, v \rangle$ and $\langle s_t, \psi \rangle$ are bounded because $h_t, s_t \in H^1(\Omega)^*$ and $v, \psi \in H^1(\Omega)$. Further-

more, the term $(\frac{1}{3}h^3\mu_x, v_x)$ can be bounded as follows:

$$|(h^3\mu_x, v_x)| \leq \|h\|_{L^\infty(\Omega)}^3 \|\mu_x\|_{L^2(\Omega)} \|v_x\|_{L^2(\Omega)} \leq C \|h\|_{H^1(\Omega)}^3 \|\mu\|_{H^1(\Omega)} \|v\|_{H^1(\Omega)}, \quad (3.10)$$

for all $h, \mu, v \in H^1(\Omega)$, where in the last step we used the one-dimensional Sobolev embedding.

Finally, to ensure that the term $(\phi'(h), w)$ is bounded, we assume for simplicity that $\phi \in C^1(\mathbb{R})$ and satisfies the Lipschitz condition

$$|\phi'(\xi) - \phi'(\zeta)| \leq C_L |\xi - \zeta|, \quad \forall \xi, \zeta \in \mathbb{R}. \quad (3.11)$$

Using this property, one obtains (see Appendix A.3 for the full proof)

$$|(\phi'(h), w)| \leq C_L (\|h\|_{L^2(\Omega)} + C) \|w\|_{L^2(\Omega)}, \quad \forall h, w \in L^2(\Omega) \supset H^1(\Omega), \quad (3.12)$$

where the constant C can be written as $C = C_L |h_\star| + C^* \|1\|_{L^2(\Omega)}$, with $h_\star$ an arbitrary reference value.

The choice of potential $\phi(h) = -\frac{A}{2} h^{-2}$ introduced in (3.3) does not satisfy the Lipschitz condition and is not bounded, which causes the model (3.2) to break down when the film ruptures, i.e., $h^{-1}(x) \rightarrow \infty$ for some $x \in \Omega$. A common approach in the literature is to regularise $\phi(h)$. We adopt the approach proposed in [58] (see also [117]) as follows:

$$\phi(h) = \begin{cases} \frac{A}{2\varepsilon^4} h^2 - \frac{A}{\varepsilon^2} & \text{if } h < \varepsilon, \\ -\frac{A}{2} h^{-2} & \text{if } h \geq \varepsilon. \end{cases} \quad (3.13)$$

The idea is to use a small $\varepsilon > 0$ to make $\phi(h)$ quadratic for $h < \varepsilon$, whilst keeping its form as $\phi(h) = -\frac{A}{2} h^{-2}$ for $h \geq \varepsilon$, ensuring that $\phi(h)$ has a minimum at $h = 0$. The adopted approach also ensures that $\phi(h)$ and $\phi'(h)$ are continuous at all points, particularly at $h = \varepsilon$. In practice, ε should be small enough to resolve the rupture region whilst still ensuring numerical stability. Following [117], we set $\varepsilon = 0.1$ in our computations.

3.3 Optimal Control Problem

We formulate an optimal control problem for the thin film flow dynamics over a flexible topography (3.8). Our goal is to find a distributed force $f(x, t)$ that influences the topography $s(x, t)$ as in (3.2c) so that the optimal solution approaches the target state $\bar{h}(x)$. This optimal solution is either the height of the thin film $h(x, t)$ or the total observed surface $\mathcal{H}(x, t) = h(x, t) + s(x, t)$.

Formally, we seek a distributed force $f(x, t)$ that minimises the cost function $\bar{\mathcal{J}}(h, s, f)$:

$$\bar{\mathcal{J}}(h, s, f) := \frac{1}{2} \|(h + \beta s)(\cdot, T) - \bar{h}(\cdot)\|_{L^2(\Omega)}^2 + \frac{\alpha}{2} \int_0^T \|f(\cdot, t)\|_{L^2(\Omega)}^2 dt, \quad (3.14)$$

where $\beta \in \{0, 1\}$ and $\alpha > 0$ is a regularisation parameter. The first term in equation (3.14) measures the difference between the current state and the target state $\bar{h}$, whilst the second term, determines a penalty for the control energy. In the unconstrained control setting of this chapter, this control penalty is crucial for ensuring the problem's well-posedness [66]. We formulate the optimal control problem as:

$$\min_{(h, s, f)} \bar{\mathcal{J}}(h, s, f), \quad \text{subject to (3.8)}. \quad (3.15)$$

3.4 First-order optimality conditions

In this work, we follow an “optimise–then–discretise” approach, which is often advantageous because the resulting discrete problems remain closely aligned with the continuous optimality system and tend to behave more robustly under mesh refinement [40]. We begin by deriving first-order optimality conditions for (3.15). For this, we define the Lagrangian

$$\begin{aligned} \mathcal{L}((h, \mu, s), f, (p, q, r)) &:= \bar{\mathcal{J}}(h, s, f) + \int_0^T \langle h_t, p \rangle + \langle s_t, r \rangle dt \\ &\quad + \int_0^T \left(\frac{1}{3} h^3 \mu_x, p_x \right) - (f, r) - (\phi'(h), q) dt + \mathcal{A}\left((h, \mu, s), (p, q, r)\right). \end{aligned} \quad (3.16)$$

The optimality conditions arise from stationarity conditions of this Lagrangian with respect to state, adjoint, and control variables (see Sec. 2.5 for further details).

Consequently, the adjoint problem consists in finding $(p, q, r) \in \mathcal{W}^{h(T)} \times \mathcal{V} \times \mathcal{W}^{\beta s(T)}$ solving

$$\begin{aligned} - \int_0^T \left(\langle p_t, v \rangle + \langle r_t, \psi \rangle + (\phi''(h)q, w) - (h^2 \mu_x p_x, v) + \left(\frac{1}{3} h^3 p_x, w_x\right) \right) dt \\ + \mathcal{A}((v, \omega, \psi), (p, q, r)) = \int_0^T (r, \psi) dt, \end{aligned} \quad (3.17)$$

where $\mathcal{W}^{v_T} := \{v \in \mathcal{W} : v(T) = v_T\}$. The optimality conditions (3.17) can be expressed as a forward-backwards system. In other words, the optimal state variables, (h^*, μ^*, s^*) , and the corresponding optimal adjoint variables, (p^*, q^*, r^*) , satisfy the following strong form:

$$\begin{cases} h_t^* - \frac{1}{3} (h^{*3} \mu_x^*)_x = 0, \\ \mu^* = \phi'(h^*) - \frac{B_o}{C_a} (\mathcal{H}^*) - \frac{1}{C_a} (\mathcal{H}_{xx}^*), \\ s_t^* - c^2 s_{xx}^* - \frac{\gamma}{C_a} (h_{xx}^* + B_o h^*) = f^*, \end{cases} \quad (3.18a)$$

$$\begin{cases} -p_t^* + h^{*2} \mu_x^* p_x^* + \frac{1}{C_a} (q_{xx}^* + \gamma r_{xx}^*) + \frac{B_o}{C_a} (q^* - \gamma r^*) - \phi''(h^*) q^* = 0, \\ \left(-\frac{1}{3} h^{*3} p_x^* \right)_x + q^* = 0, \\ -r_t^* - c^2 r_{xx}^* + \frac{B_o}{C_a} q^* + \frac{1}{C_a} q_{xx}^* = 0, \end{cases} \quad (3.18b)$$

$$\begin{cases} h_x^* = \mu_x^* = s^* = p_x^* = q_x^* = r_x^* = 0, & \text{on } \Omega \times [0, T], \\ h^* = h_0, \quad s^* = s_0, & \text{on } \Omega \times \{t = 0\}, \\ p^* = r^* = (\bar{h} - (h + \beta s)), & \text{on } \Omega \times \{t = T\}. \end{cases} \quad (3.18c)$$

closed by the optimality condition

$$\alpha f^* - r^* = 0. \quad (3.19)$$

Since the control f acts directly through the substrate equation (3.2c), the stationarity condition for the Lagrangian (3.16) with respect to f gives (3.19). Accordingly, r^* is the adjoint variable associated with the substrate dynamics.

3.4.1 Reduced gradient approach

We solve the optimisation problem (3.15) by means of gradient descent. For this, the system dynamics (3.2) are used to define a nonlinear map from f to $h + \beta s$, leading to a reduced objective functional $\mathcal{J}$ expressed solely in terms of f :

$$\mathcal{J}(f) = \bar{\mathcal{J}}(h(f), s(f), f) = \frac{1}{2} \|(h + \beta s)(f; \cdot, T) - \bar{h}(\cdot)\|_{L^2(\Omega)}^2 + \frac{\alpha}{2} \int_0^T \|f(\cdot, t)\|_{L^2(\Omega)}^2 dt. \quad (3.20)$$

Here, $(h + \beta s)(f; \cdot, t)$ represents the solution of the system (3.2) for a given f . Expressed in terms of the reduced objective functional, the optimal control problem can be reformulated as follows:

$$\min_f \mathcal{J}(f), \quad \text{subject to (3.2)}. \quad (3.21)$$

It can be shown (see, e.g., Section 2.7.2), that the gradient of the reduced objective can be recovered from the optimality condition (3.19), that is,

$$\nabla \mathcal{J}(f) := \alpha f - r, \quad (3.22)$$

where the adjoint variable r is to be computed from the forward-backward system (3.18) which depends on f through the state equation. This gradient is the used in the iteration

$$f^{k+1} = f^k - \lambda^k \nabla \mathcal{J}(f^k), \quad k = 1, 2, \dots \quad (3.23)$$

where the stepsize λ^k is computed by backtracking. An elementary version of the pseudocode for this iterative loop is outlined in Algorithm 2.

Algorithm 2 Gradient descent with backtracking line search

Require: $\text{tol} > 0, k_{\max}, f^0, \lambda, k = 0;$

- 1: **while** $\|\nabla \mathcal{J}(f^k)\| > \text{tol}$ and $k < k_{\max}$ **do**
- 2: Obtain (h^k, μ^k, s^k) with f^k from (3.18a).
- 3: Obtain (p^k, q^k, r^k) with (h^k, μ^k, s^k) from (3.18b).
- 4: Evaluate the gradient $\nabla \mathcal{J}(f^k)$ from (3.19).
- 5: Update $f^{k+1} = f^k - \lambda^k \nabla \mathcal{J}(f^k)$
- 6: **while** $\mathcal{J}(f^{k+1}) > \mathcal{J}(f^k)$ **do**
- 7: Obtain $(h^{k+1}, \mu^{k+1}, s^{k+1})$ with f^{k+1}
- 8: Evaluate the cost $\mathcal{J}(f^{k+1})$
- 9: **if** $\mathcal{J}(f^{k+1}) > \mathcal{J}(f^k)$ **then**
- 10: $\lambda^m = 0.5\lambda^k, m \geq k$
- 11: Update $f^{k+1} = f^k - \lambda^m \nabla \mathcal{J}(f^k)$
- 12: **end if**
- 13: **end while**
- 14: $k := k + 1$
- 15: **end while**

3.5 Numerical approximation and tests

3.5.1 IMEX-FEM discretisation of the model

For the computational treatment of the optimality conditions (3.18)-(3.19) derived above at a continuous level are discretised in space using the standard FEM for variables h, s, f , and μ , as well as the adjoint variables p, q , and r , subject to homogeneous Neumann boundary conditions. More details on the FEM formulation are provided in Appendix A.4. For the time discretisation we need more careful treatment as the nonlinearities in equations (3.9) and (3.17) require a particular choice for the time-discretisation scheme to balance computational efficiency and stability [162]. We explore first-order Implicit-Explicit (IMEX) time-stepping schemes, which decompose the nonlinear term ϕ in (3.13) as:

$$\phi(h; v) = \phi_+(h; v) + \phi_-(h; v), \quad (3.24)$$

where ϕ_+ and ϕ_- denote the contractive (convex) and expansive (concave) components of ϕ , respectively. The IMEX scheme treats the convex part implicitly and the concave part explicitly. For the convex term $\phi_+ = \frac{1}{2\varepsilon^4}h^2$, the derivative $\phi'_+(h) = \frac{1}{\varepsilon^4}h$ is linear. Treating it implicitly therefore requires only a linear solve at each time

step, retaining computational efficiency whilst ensuring stability as $h \rightarrow 0$.

The time interval $[0, T]$ is divided into N smaller intervals $0 = t_0 < t_1 < \dots < t_N = T$, with each sub-interval $\mathcal{I}_{k+1} = [t_k, t_{k+1}]$ having a uniform time step $\Delta t = t_{k+1} - t_k$ for $k = 0, 1, \dots, N-1$. The time discretisation for problem (3.9) is:

Find $(h_{k+1}, \mu_{k+1}, s_{k+1}) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$ such that for all $(v, \omega, \psi) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$

$$\begin{aligned} & \left(\frac{h^{k+1} - h^k}{\Delta t}, v \right) + \left(\frac{s^{k+1} - s^k}{\Delta t}, \psi \right) - \left(\phi'_+(h^{k+1}); v \right) - \left(\phi'_-(h^k); v \right) \\ & + \left(\frac{1}{3} (h^k)^3 \mu_x^{k+1}, v_x \right) + \mathcal{A}((h^{k+1}, s^{k+1}, \mu^{k+1}), (v, \omega, \psi)) = (f^{k+1}, \psi), \end{aligned} \quad (3.25)$$

for $k = 0, 1, \dots, N-1$, where the initial condition is

$$(h, v) = (h_0, v) \quad \text{and} \quad (s, \psi) = (s_0, \psi), \quad \forall v, \psi \in L^2(\Omega).$$

It can be shown that this scheme dissipates the total energy (3.6) using techniques shown in [162].

For the adjoint problem (3.17), we introduce an IMEX time-stepping scheme to discretise backwards in time:

Find $(p_k, q_k, r_k) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$ such that for all $(v, \omega, \psi) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$

$$\begin{aligned} & - \left(\frac{p^{k+1} - p^k}{\Delta t}, v \right) - \left(\frac{r^{k+1} - r^k}{\Delta t}, \psi \right) - \left(\phi''_+(h^{k+1}) q^{k+1}, w \right) - \left(\phi''_-(h^k) q^k, w \right) \\ & - \left((h^k)^2 \mu_x^{k+1} p_x^{k+1}, v \right) + \frac{1}{3} \left((h^k)^3 p_x^{k+1}, w_x \right) - (r^{k+1}, \psi) \\ & + \mathcal{A}((p^k, q^k, r^k), (v, \omega, \psi)) = 0, \end{aligned} \quad (3.26)$$

for $k = 0, 1, 2, \dots, N-1$, where the terminal condition is:

$$\begin{aligned} & (p^N, v) + (r^N, \psi) - \Delta t (\phi''_+(h^N), q^N) + \frac{\Delta t}{3} ((h^N)^3 p_x^N, v) \\ & + \Delta t \mathcal{A}((p^N, q^N, r^N), (v, \omega, \psi)) = (\bar{h}^N - (h + \beta s)^N, v). \end{aligned} \quad (3.27)$$

3.5.2 Numerical tests

The results were generated using the forward-backward optimisation algorithm (Algorithm 2) with a tolerance $\text{tol} = 10^{-4}$ and initial guess $f^0 = 0$. All simulations were carried out using a space-time discretisation with $n = 250$ nodes, a time step

$\Delta t = 0.05$, and the following fixed parameters: $C_a = B_o = 1$, $c = 0.1$, and the regularisation parameter $\alpha = 10^{-6}$. We set $\gamma = 0$ in all simulations, which neglects the pressure-coupling term from the substrate equation. The wall then responds only to the actuation $f(x, t)$ and its own elasticity, whilst still influencing the fluid through its deformation, which appears in the observed surface height $\mathcal{H} = h + s$. We solve the system (3.18) on $[0, L]$. The initial conditions used in the numerical simulations are, unless otherwise specified,

$$h_0(x) = 1 + h_A \cos\left(\frac{m\pi x}{L}\right), \quad s_0(x) = 0, \quad (3.28)$$

where the initial amplitude h_A will generally be chosen between $[0, 0.5]$ (with $h_A = 0$ corresponding to a flat state), and the frequency parameter will be set to $m = 1$ unless otherwise specified. We note that the respective flow with a flat substrate evolves to a nontrivial state whenever $L > 2\pi$, as predicted by linear stability analysis [62]. In addition to utilising the cost function and the norm of its gradient to validate our solution, we also employ the L_∞ -norm as a measure of accuracy; it is defined as $\max_{t \in [0, T]} \|(h^*(t) + \beta s^*(t)) - \bar{h}\|_\infty$. For each example, three sets of figures are presented. The first set of figures show the free-surface and substrate evolutions in a symmetrically extended domain $[-L, L]$; this is done for illustrative purposes and better visualisation of the results. More detailed results, including the time evolution of the forcing $f(x, t)$, are demonstrated in the rest of the figures in the computational domain $[0, L]$. Table 3.1 summarises the key parameters and comparison between uncontrolled and controlled states for all numerical examples.

3.5.3 Non-rupture scenarios

In this section, we set $\beta = 1$ in (3.20) because our goal is to control the observed surface, defined as $\mathcal{H} = h + s$.

Accelerated transition to desirable surface shape

In the first two examples, where no disjoining pressure is present (i.e. $\phi' \equiv 0$), the uncontrolled dynamics are obtained by solving the forward problem (3.2) until temporal variations in the free surface become negligibly small. In Example 1,

Table 3.1: Summary of numerical examples (shaded rows indicate rupture cases).

Common Parameters: $C_a = B_o = 1$ (except Ex. 4: $B_o = 0$), $c = 0.1$, $\gamma = 0$, $\alpha = 10^{-6}$, $n = 250$, $\Delta t = 0.05$, and $f^0 = 0$				
Example	Key Parameters	Uncontrolled State	Controlled State	Figures
1	$L = 3\pi$, $T = 5$, $h_A = 0.5$ $\beta = 1$, $\phi(h) = 0$	Quasi-steady state reached at $T = 900$	Target state solution from Hammond [62] achieved at $T = 5$	Figs. 3-5
2	$L = 3\pi$, $T = 5$, $h_A = 0$ $s_0 = -0.25[\tanh(\frac{x+c_1}{d}) - \tanh(\frac{x-c_2}{d})]$ $d = -0.2$ $\beta = 1$, $\phi(h) = 0$, $c_1 = -0.35L$ $c_2 = 0.65L$	Quasi-steady state reached at $T = 900$	Target state achieved at $T = 5$	Figs. 6-8
3	$L = 5$, $T = 1$, $h_A = 0$ $\beta = 1$, $\phi(h) = 0$	Remains flat (no evolution)	Wave-like shape at $T = 1$	Figs. 9-11
4	$L = 10$, $T = 5$ $h_0 = 1 + 10\exp(-(2x)^2)$ $\beta = 1$, $\phi(h) = 0$	Non-flat steady state (bump profile)	Flattened to $\bar{h} = 1$ at $T = 5$	Figs. 12-14
5	$L = \frac{15}{2}\pi$, $T = 10$, $h_A = 0.5$ $\beta = 1$, $\phi(h) = 0$	Nonlinear instability develops	Maintains initial shape	Figs. 15-17
6	$L = 3\pi$, $T = 30$, $h_A = 0.5$, $\beta = 0$, $\phi(h) \neq 0$, $\varepsilon = 0.1$, $A = 0.03$	Ruptured state reached at $T = 550$	Ruptured target state at $T = 30$	Figs. 18-20
7	$L = 3\pi$, $T = 30$ $\beta = 0$, $\phi(h) \neq 0$, $\varepsilon = 0.1$, $A = 0.03$	Remains ruptured	Bubbles merge into uniform surface at $T = 30$	Figs. 21-22

we consider $f = s = 0$, so that (3.2) reduces to the classical thin-film equation of Hammond [62], and the resulting quasi-steady profile is exported as the target state $\bar{h}$. In Example 2, we repeat the same procedure but over a prescribed, non-flat substrate $s(x)$, again integrating the uncontrolled system until the solution reaches a quasi-steady regime. These quasi-steady profiles provide the reference states against which the controlled evolution is compared.

Example 1

We begin the presentation by comparing the time required for uncontrolled and controlled free surfaces to reach a quasi-steady state. For the uncontrolled case (i.e., $s = f = 0$), the forward system (3.2) has been resolved by Hammond [62] for $L = 3\pi$, and reached a quasi-steady state after $T = 900$ time units. We aim to observe how fast the optimal state $\mathcal{H}^*$ approaches the target state $\bar{h}$ when control is applied.

We apply the forward-backwards optimisation method using the algorithm in Figure 2, where the control is applied over the time interval $[0, T]$, with final time $T = 5$, using initial conditions (3.28) with $h_A = 0.5$. For illustrative purposes, snapshot plots spanning the domain $[-L, L]$ are presented in Figure 3.2. The result of this computation is demonstrated in Figures 3.3 and 3.4. The quasi-steady state is exported as our target state $\bar{h}$, shown in panel 3.3a. The optimal free surface over topography, $\mathcal{H}^*$, shown in panel 3.3b, successfully achieves this target state fast. The plot in panel 3.3c illustrates how the optimal free surface, h^* , varies over time. Panel 3.3d shows that the topography s^* evolving from a flat state to an optimally adjusted profile with unexpected peaks and troughs, whilst panel 3.3e shows the control parameter f^* exhibits oscillatory behaviour to shape the optimal topography s^* . The cost function $\mathcal{J}(f^*)$ illustrated in panel 3.3f, decreases to zero by the final time. Lastly, Figure 3.4a shows the infinity norm of the controlled solution $(h^*(t) + \beta s^*(t)) - \bar{h}$ goes to zero much faster than the uncontrolled one, and Figure 3.4b explores the impact of the regularisation parameter α on the solution, showing that smaller values of α than 10^{-3} lead to negligible changes to the optimal solution.

To verify that the controlled free surface $\mathcal{H}^*$ is quasi-steady, we initialise a subsequent forward simulation with the terminal optimal states $h^*(\cdot, T)$ and $s^*(\cdot, T)$. As

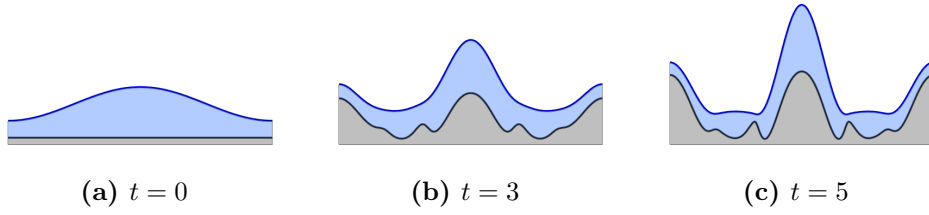


Figure 3.2: Example 1. Snapshots of the optimal solution, controlled to a desired surface shape. The solution is extended symmetrically in the domain $[-L, L]$.

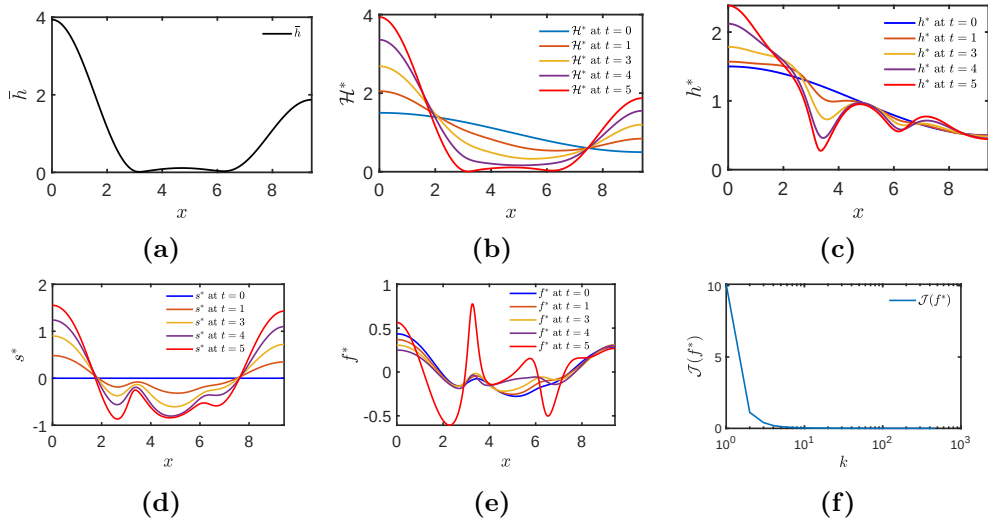


Figure 3.3: Example 1. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively. (f) Cost function $\mathcal{J}(f^*)$ against number of iterations k .

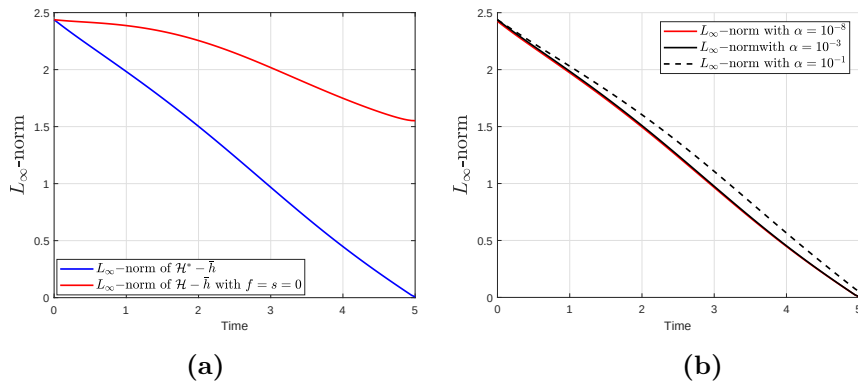


Figure 3.4: Example 1. L^∞ -norms for (a) the controlled and uncontrolled solutions, and (b) various values of α .

expected, the free surface remains unchanged, whilst the topography relaxes towards a flat state.

Example 2

We extend the first example by solving the forward system (3.2) over a non-flat topography, contrasting with the flat substrate examined previously. The uncontrolled system is solved with initial conditions comprising a flat surface profile (i.e., using $h_A = 0$) and a substrate profile $s_0(x) = a \left(\tanh\left(\frac{x+c_1}{d}\right) - \tanh\left(\frac{x-c_2}{d}\right) \right)$, where $a = -0.25$, $c_1 = -0.35L$, $c_2 = 0.65L$, $L = 3\pi$, and the topography steepness is $d = -0.2$. The aim remains the same as in the first example in achieving the target state much faster than the uncontrolled one. A quasi-steady state for the free surface is achieved at $T = 900$, and the optimal control solution is presented in Figures 3.5 and 3.6. Panel 3.6a shows the target state, and panels 3.6b and 3.6c illustrate the time evolution of $\mathcal{H}^*$ and h^* , respectively. The smoothing changes in amplitudes and phases of s^* are highlighted in panel 3.6d, and the oscillatory forces f^* that influence s^* are shown in panel 3.6e. Figure 3.7 demonstrates that both the L^∞ norm and the cost function converge to zero, although the convergence is slower than the case with a flat initial topography.

Example 3

In this section, we consider a case where the free surface quickly transitions from flat to a desired wave-like shape in a short time $t = [0, 1]$, as shown in Figures 3.8 and 3.9. Using a computational domain of length $L = 5$, we start with an initially flat surface by setting $h_A = 0$ in (3.28). Without control, the surface remains flat, but with control, it transforms into the desired wave-like shape as shown in Figure 3.9a. The optimal solution of $\mathcal{H}^*$ and h^* are shown in panels 3.9b and 3.9c, respectively. The topography s^* changes from flat to an oscillatory profile, enabling the desired wave-like target state. The control force f^* oscillates in sync with the topography changes, facilitating rapid adaptation, as seen in panel 3.9e. Figure 3.10 demonstrates the cost function decreases towards zero, achieving the minimum cost, and the optimal control reaches the target surface.

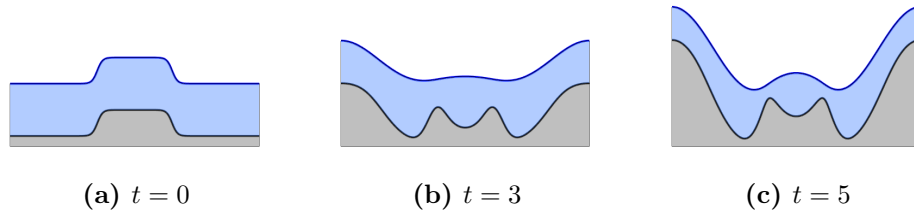


Figure 3.5: Example 2. Snapshots of the optimal solution, controlled to a desired surface shape.

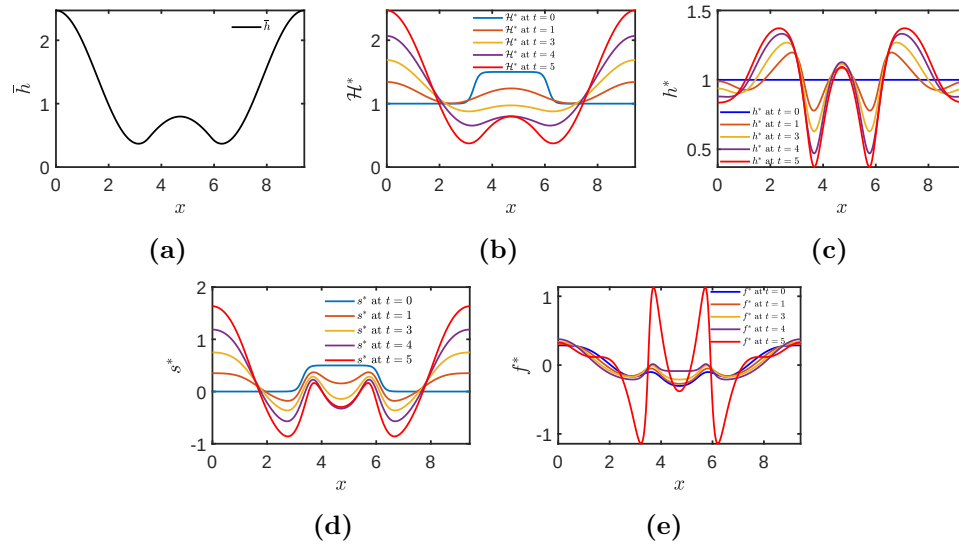


Figure 3.6: Example 2. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.

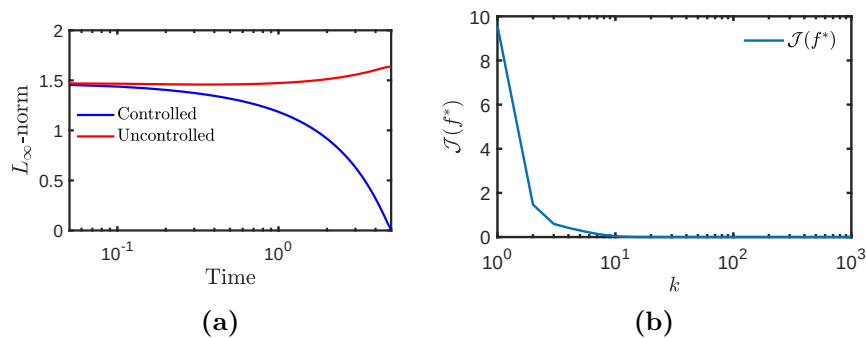


Figure 3.7: Example 2. (a) L^∞ -norm for the controlled and uncontrolled solutions. (b) Cost function $\mathcal{J}(f^*)$ against number of iterations k .

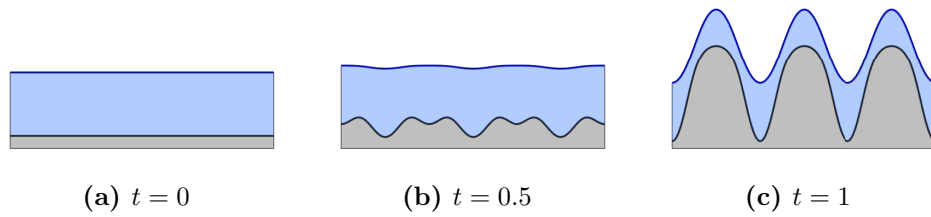


Figure 3.8: Example 3. Snapshots of the optimal solution, controlled to a desired surface shape. The solution is extended symmetrically in the domain $[-L, L]$.

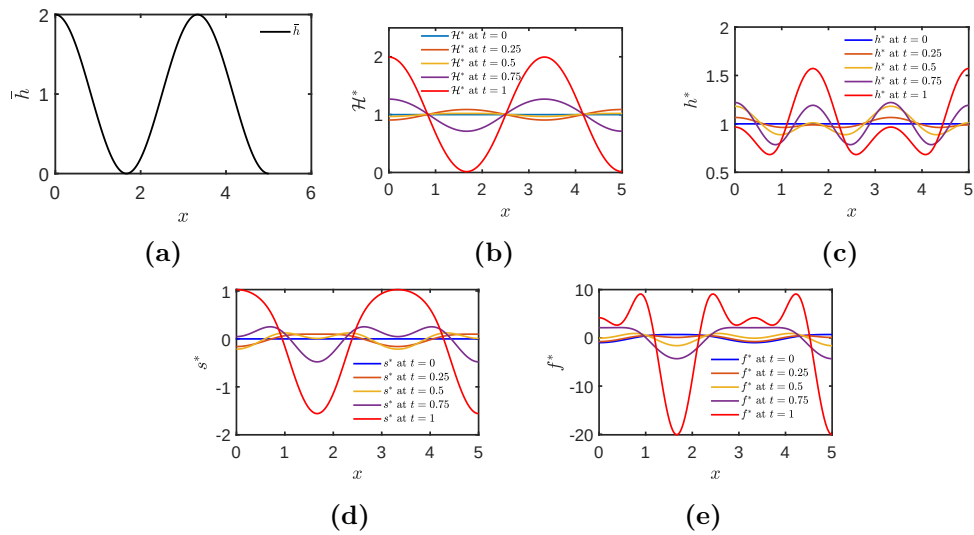


Figure 3.9: Example 3. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.

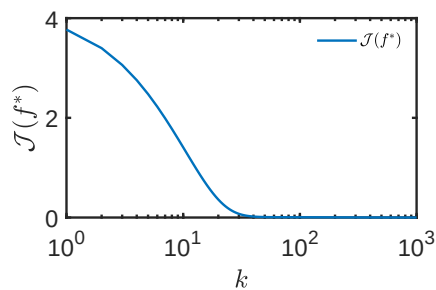


Figure 3.10: Example 3. Cost function $\mathcal{J}(f^*)$ against number of iterations k .

From non-flat to desired shapes

Example 4

The next example follows the work of [78] where a related (uncontrolled) evolution equation for thin-film flow over topography was solved and a non-flat quasi-steady state was achieved. We initiate our control problem using one of these non-flat states as the initial condition by taking $h_0(x) = 1 + 10 \exp(-(2x)^2)$, $s_0(x) = 0$, and set $L = 10$, $B_o = 0$. Our aim is to control the problem and flatten the surface (i.e., the target state is $\bar{h} = 1$). The time evolution of the various system variables is shown in Figures 3.11 and 3.12. Panel 3.12b shows that our optimal solution achieves a flat state, whilst panel 3.12c displays deviations in h^* , indicating that the film height alone does not attain the flat target state. Panels 3.12d and 3.12e reveal s^* and f^* adapting by $T = 5$ to counterbalance the variations in h^* , aligning $\mathcal{H}^* = h^* + s^*$ with the target state. Figure 3.13 shows the L^∞ -norm for the controlled and uncontrolled solutions over time, relative to the target state. The controlled solution moves towards zero, whilst the uncontrolled solution stabilises at a non-zero value.

Example 5

We also carried out simulations aiming to retain the surface at a linear state, whilst solving the nonlinear system (recall that this exhibits free-surface instabilities in the uncontrolled case if $L > 2\pi$). The solution of the uncontrolled forward system with initial conditions (3.28) and $h_A = 0.5$, $m = 3$, $L = \frac{15}{2}\pi$ results in nonlinear instability, but our objective is to maintain the free surface at the linear initial condition $\mathcal{H}(x, 0) = h_0 + s_0$ (i.e., $\bar{h} = \mathcal{H}(x, 0)$) throughout the simulation period $[0, T]$, with final simulation time $T = 10$. The results of the computation are depicted in Figures 3.14, 3.15 and 3.16. Panel 3.15a shows the target state $\bar{h}$, whereas panel 3.15b presents $\mathcal{H}^*$ maintaining its initial sinusoidal shape. Panel 3.15c depicts the free surface evolution, which naturally tends towards instability (non-flat state). In contrast, it is adjusted by the changing topography in panel 3.15d, which develops inverse undulations to h^* to maintain $\mathcal{H}^*$. The oscillatory behaviour of the control f^* is depicted in panel 3.15e. The cost function decreases as indicated in panel 3.15f, reflecting progress towards the target state. Finally, Figure 3.16 shows

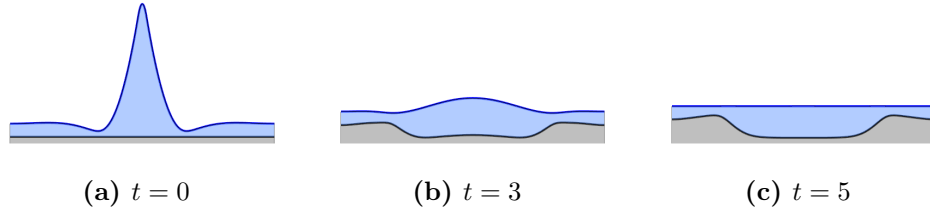


Figure 3.11: Example 4. Snapshots of the optimal solution, controlled to a uniform state. The solution is extended symmetrically in the domain $[-L, L]$.

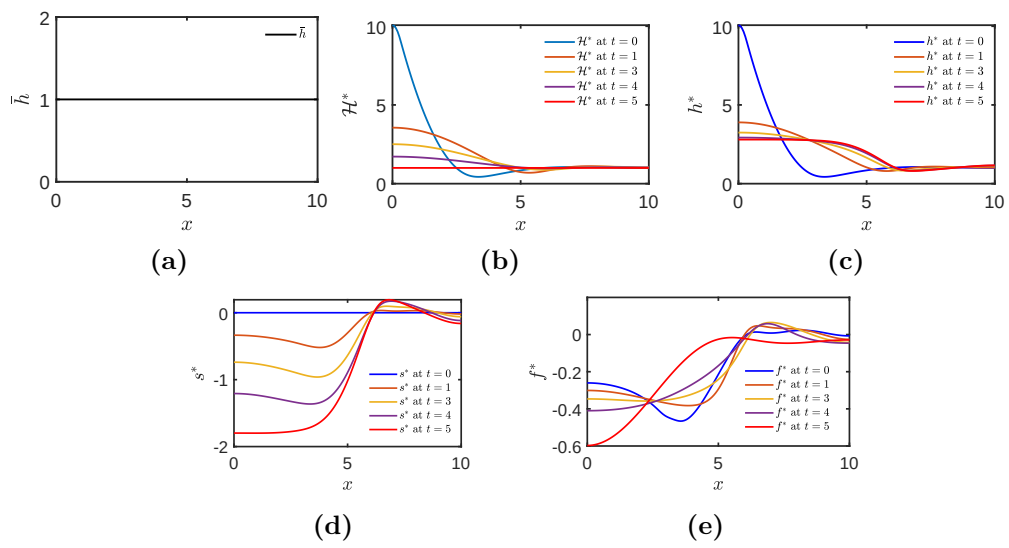


Figure 3.12: Example 4. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.

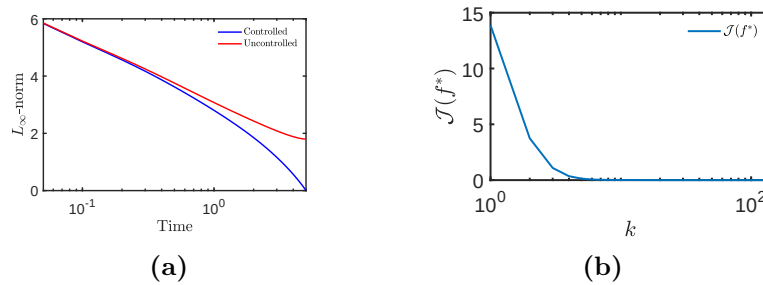


Figure 3.13: Example 4. (a) L^∞ -norm for controlled and uncontrolled solutions. (b) Cost function $\mathcal{J}(f^*)$ against number of iterations k .

a brief increase in the infinity norm due to strong instability, before it returns to near zero.

3.5.4 Rupture scenarios

In this section, we set $\beta = 0$ in (3.20), focusing on controlling $h(x, t)$ rather than the total surface $\mathcal{H} = h + s$. This approach lets $\mathcal{H}$ rupture naturally, driven by the interaction between the thin film $h(x, t)$ and the underlying topography $s(x, t)$, providing a realistic representation of physical rupture. If we were to control $\mathcal{H}$ directly, it could prevent rupture by limiting the film's ability to interact with the topography closely.

In the following examples, we aim to examine how thin-film flows evolve towards a rupture state. We set a non-zero rupture function $\phi(h)$ as in (3.24), with $\phi_+ = \frac{1}{2\varepsilon^4}h^2$, $\varepsilon = 0.1$, and $A = 0.03$. We integrate the forward system (3.2) with $f = s = 0$ and using the initial conditions (3.28) with $h_A = 0.5$, and within a spatial domain of length $L = 3\pi$. The simulations are carried out over $[0, T]$, with $T = 550$, until the temporal changes become sufficiently slow, and regard this limiting configuration as a quasi-steady state. This numerically obtained quasi-steady profile is then used as the target state $\bar{h}_{\text{rupt}}$ for the corresponding optimal-control problem.

Example 6

In the first example, we assess how quickly the optimal free surface h^* reaches the target rupture state $\bar{h} = \bar{h}_{\text{rupt}}$, relative to the uncontrolled case. The optimal solution is shown in Figures 3.17 and 3.18. whilst the uncontrolled solution necessitated $T = 550$ to achieve a quasi-steady ruptured thin-film state $\bar{h}_{\text{rupt}}$, shown in panel 3.18a, our controlled solution h^* achieves the desired state within just $T = 30$, as seen in panel 3.18b. Panel 3.18c depicts how the optimal thin-film flow over topography $\mathcal{H}^*$, including rupture effects, changes over time. The optimal topography s^* , and the control function f^* are illustrated in panels 3.18d and 3.18e, respectively. The significant reduction in convergence time is highlighted in Figure 3.19. Both L^∞ norm of $h^* - \bar{h}_{\text{rupt}}$ and cost function approach zero, though more slowly than in the non-rupture case.

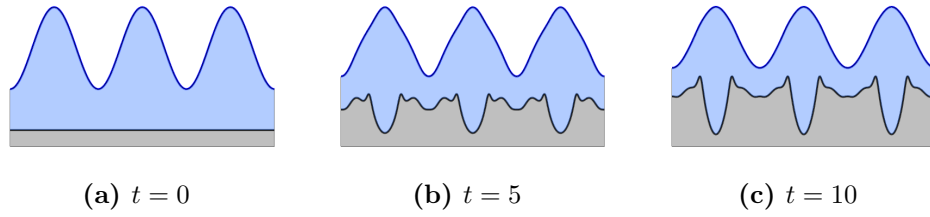


Figure 3.14: Example 5. Snapshots of the optimal solution, controlled to remain at the initial surface shape. The solution is extended symmetrically in the domain $[-L, L]$.

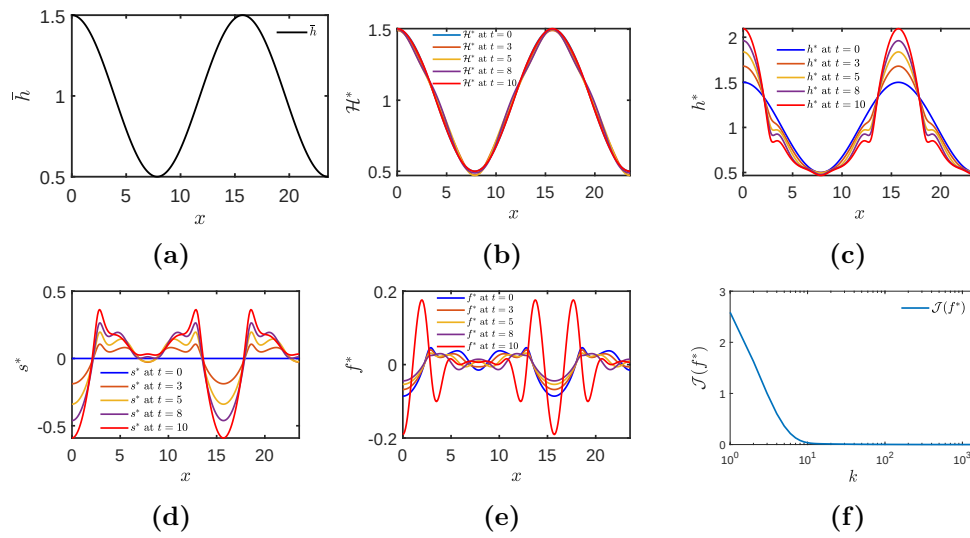


Figure 3.15: Example 5. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively. (f) Cost function $\mathcal{J}(f^*)$ against number of iterations k .

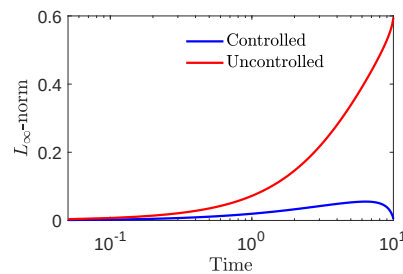


Figure 3.16: Example 5. L^∞ -norm of controlled vs uncontrolled solutions relative to $\bar{h}$.

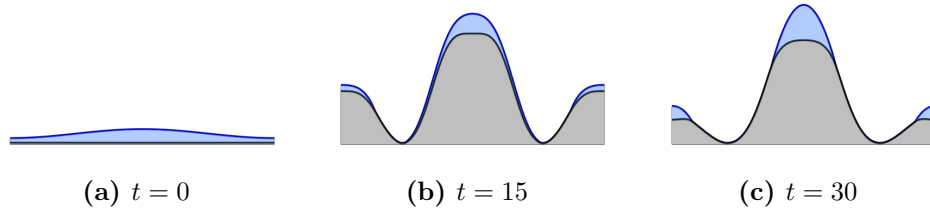


Figure 3.17: Example 6. Snapshots of the optimal solution, controlled towards a ruptured state. The solution is extended symmetrically in the domain $[-L, L]$.

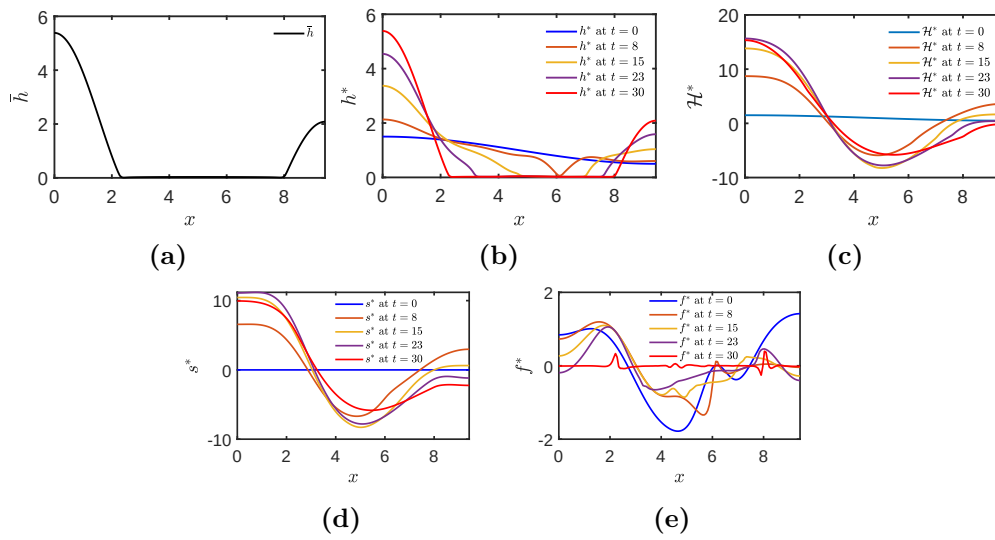


Figure 3.18: Example 6. (a) Target state $\bar{h}$. (b-e) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively.

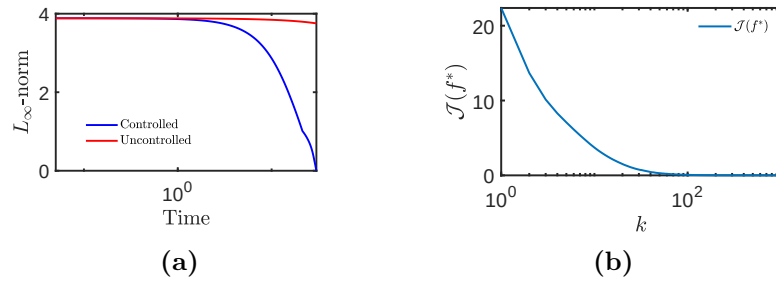


Figure 3.19: Example 6. (a) L^∞ -norm of the variation $h^* - \bar{h}_{\text{rupt}}$. (b) Cost function $\mathcal{J}(f^*)$ against number of iterations k .

Example 7

In the final example, we start with an initially ruptured state, $h_0(x) = \bar{h}_{\text{rupt}}$, and show that our control can halt the rupture by allowing the bubbles to merge into a single uniform surface, as shown in Figure 3.20. This demonstrates how control can guide the system from rupture to a stable state. Figure 3.21 depicts h^* , $\mathcal{H}^*$, the flexible topography s^* and control f^* , as well as the cost function which is seen to reach its minimum faster compared to the previous example (cf. Figure 3.19b).

3.6 Conclusion

We have developed a mathematical framework for the open-loop optimal control of thin-film flows over flexible topography, accounting for the rupture of films and the coalescence of bubbles. The focus of this chapter has been on finding the distributed force influencing the flexible topography to minimise the cost function. We have also established a functional setting for the coupled PDE system, comprising a thin film equation and a flexible substrate equation. This demonstrates that this system adheres to a global energy-dissipation law. Furthermore, we have derived the optimality conditions by formulating the mathematical control problem as a PDE-constrained optimisation problem to obtain the time-dependent forward and backward PDE system and the optimality condition. We used a stable time discretisation approach with IMEX schemes to address the model's nonlinearity, such as the disjoining-pressure term. This approach worked well in our numerical examples, demonstrating its effectiveness in solving the optimisation problem.

The effectiveness of our optimal control problem was demonstrated through a range of numerical examples, showing its ability to accelerate convergence to steady states, stabilise dewetting processes, achieve specific profiles, and even reverse rupture dynamics to restore uniformity. These results were achieved by allowing the adjacent wall to take a flexible shape, thereby affecting the surface profile. This non-standard approach provides a mechanism to control thin-film dynamics without altering the fluid properties (e.g., surface tension, stresses). It offers a useful alternative for addressing challenges in thin-film control and complements existing control

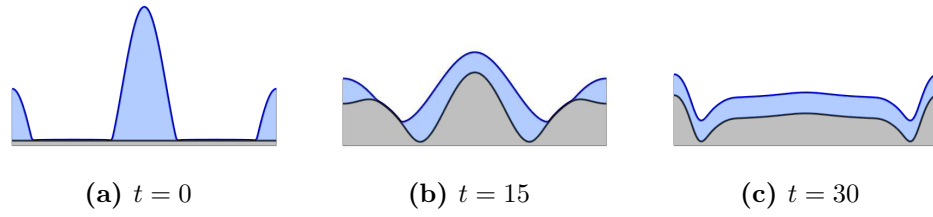


Figure 3.20: Example 7. Snapshots of the optimal solution, controlled from a ruptured state to a uniform film thickness. The solution is extended symmetrically in the domain $[-L, L]$.

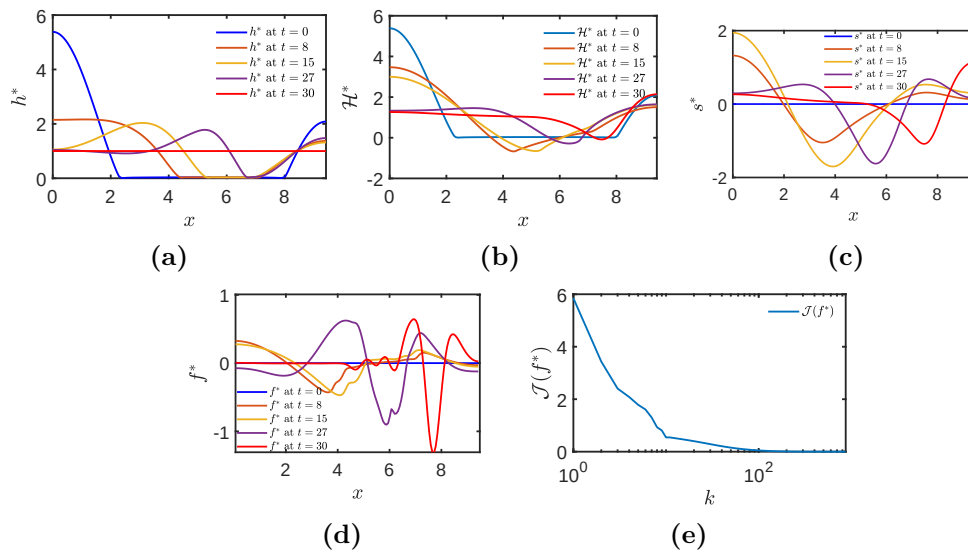


Figure 3.21: Example 7. (a-d) Evolution of the optimal surface profile $\mathcal{H}^*$, film height h^* , topography s^* , and control f^* , respectively. (e) Cost function $\mathcal{J}(f^*)$ against number of iterations k .

approaches; previous studies have focussed on using thermal feedback [51, 150], fluid injection and extraction [35, 57, 106, 149], or electric fields [153, 161].

The optimality system presented in this study was formulated as an open-loop control problem, namely the controller design is independent of the system output. This work can be extended to closed-loop control, also known as feedback control, which influences the control input directly [11] and helps navigate the system along the optimal path set by the open-loop control problem [110]. In thin film applications, feedback control not only stabilises contact line instabilities but also addresses complex problems, such as optimising the placement of actuators and sensors to suppress or enhance waves on a thin film [51, 57, 149]. In addition, feedback control using fluid injection and extraction techniques are employed to mitigate wave phenomena in thin films and optimise actuator placement [72, 149]. The extension of our work to feedback control and developing numerical methods for solving a closed-loop control problem is left as future work.

Part II

Actuator design and learned feedback control

Chapter 4

Neural Networks for Solving and Controlling PDEs

This chapter reviews the use of neural networks for solving and controlling systems governed by PDEs. It introduces the basic principles of neural networks and their role as function approximators for PDE solutions. Particular attention is given to Physics-Informed Neural Networks (PINNs). These networks incorporate the governing equations directly into the training process. The chapter then outlines extensions to PDE-constrained control, where neural networks represent the state or control functions. It concludes with a discussion on data-driven discovery of PDEs, showing how neural networks can learn governing equations directly from observations.

4.1 Introduction

Traditional numerical solution of PDEs has relied on schemes such as the finite difference and the FEM [102]. Similarly, classical formulations of PDE-constrained control typically involve analytical derivations of the optimality system followed by iterative numerical optimisation [40]. Whilst these methods are robust for many established problems, they encounter difficulties in high-dimensional domains or systems with strong nonlinear coupling [25, 38]. These limitations motivate the use of neural-network-based framework [3, 63, 88, 123].

Neural network methods for solving differential equations using continuous analytical representations were developed by Lagaris et al. [97]. They proposed a neural-network-based trial function for initial-boundary-value problems, constructed so that the boundary and initial conditions are satisfied. As a result, the differential equation can be reformulated as an optimisation problem over the network parameters, without introducing additional constraints during training. The network acts as a differentiable global approximator. Its derivatives are computed using automatic differentiation and are then used to minimise the PDE residual in the loss function [16, 97, 134].

Building on this idea, several subsequent methods embed PDEs directly into neural-network frameworks. The Physics-Informed Neural Networks (PINNs) introduced by Raissi et al. [134] is trained by minimising a composite loss combining data and PDE residuals. This enables forward simulation, parameter identification, and control design within a single optimisation framework [38, 123, 134]. The PINN approach has been applied to nonlinear PDEs and complex geometries where mesh-based solvers are less effective, showing promise for high-dimensional problems [38, 167].

However, PINNs are not the only neural-network framework proposed for solving PDEs. The Deep Ritz Method defines the loss through the energy functional [45], whereas Galerkin-type approaches weight the residual with test functions, leading to the Deep Galerkin Method [144]. Recent extensions, neural-weighted weak formulations improve the accuracy of PDE discretisation schemes, yielding high precision for quantities of interest even on coarse meshes [29, 30]. Neural surrogates, for example

DeepONet [107], learn the mapping between inputs and PDE solutions, replacing explicit solvers with efficient learned operators [59, 103, 165].

In this chapter, we focus on the PINN framework, which serves as a unifying approach among many neural-network-based methods for solving and controlling PDEs. PINNs combine the flexibility of neural networks with the structure of governing physical laws, providing a general basis for both forward simulation and control applications. The formulations presented here follow established developments [67, 123, 132, 134], whilst comprehensive surveys of deep learning methods for PDEs and control can be found in [65, 76, 148].

The remainder of this chapter is organised as follows. Section 4.2 recalls basic neural network concepts. Section 4.3 introduces the PINN framework for PDEs. Section 4.4 reviews classical PDE control, and Section 4.5 shows how PINNs can be employed in control settings. Finally, Section 4.6 discusses data-driven discovery of PDE models.

4.2 Neural Networks

Artificial neural networks are computational models inspired by the structure of biological neural systems¹. Neural networks rely on fundamental mathematical concepts such as linear algebra, calculus, optimisation, and approximation theory, making neural networks natural tools in applied mathematics [67]. They consist of layers of connected neurons that apply weighted transformations followed by nonlinear activation [38]. Several architectures exist, including recurrent networks, convolutional networks, and deep networks with multiple hidden layers (see [38, 163] for more details). Among these, the feedforward neural network, also known as a multi-layer perceptron, is the most widely used and forms the basis of the PINN framework [134].

In a feedforward network, the input variables (e.g. spatial and temporal coordinates of a PDE) are passed forward sequentially from the input layer, through one or more hidden layers (e.g. nonlinear transformations combining weighted inputs), to the output layer (e.g. the PDE solution or a parameter of interest) [163].

¹This section outlines the basic principles of neural network structure, training, and optimisation; see Higham and Higham [67] for an overview for applied mathematicians.

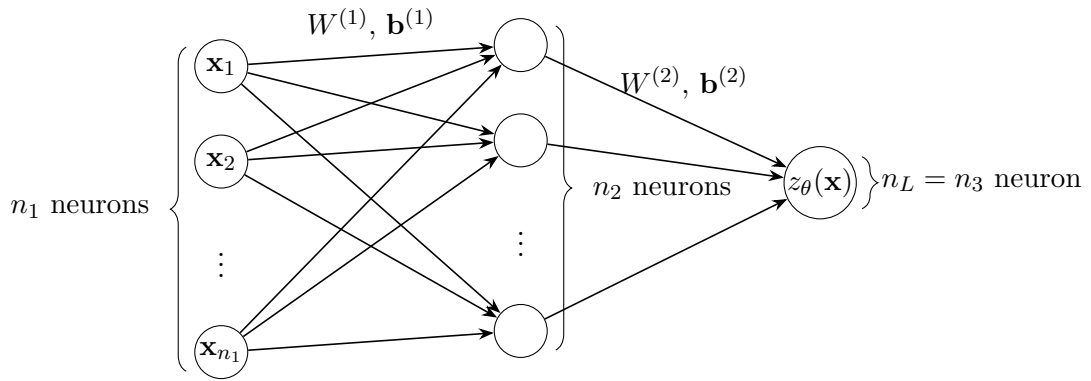


Figure 4.1: A sketch of a feedforward neural network with one hidden layer.

The following subsection introduces the standard formulation of a feedforward network, describing how inputs are propagated through successive layers of weighted and biased transformations. This provides the foundation for discussing activation functions, approximation properties, and training procedures in later sections.

4.2.1 Feedforward Neural Networks

A feedforward network is defined as a composition of affine maps and nonlinear activation functions [163]. Here we adopt the general formulation and notation presented by Higham and Higham [67]. Suppose the network has $\bar{L}$ layers, with layers 1 and $\bar{L}$ being the input and output, respectively. Let layer ℓ contain n_ℓ neurons; thus the input and output dimensions are n_1 and $n_{\bar{L}}$. The network realises a mapping

$$z_\theta : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{\bar{L}}. \quad (4.1)$$

For each layer $\ell = 2, \dots, \bar{L}$, we denote the weight matrix and bias vector by

$$W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, \quad \mathbf{b}^{(\ell)} \in \mathbb{R}^{n_\ell}, \quad (4.2)$$

where $w_{jk}^{(\ell)}$ is the weight applied by neuron j in layer ℓ to the output of neuron k in layer $\ell - 1$, and $b_j^{(\ell)}$ is the bias of neuron j in layer ℓ . The structure of such a network is shown in Figure 4.1.

The input layer is

$$\mathcal{N}^0(\mathbf{x}) = \mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^{n_1}. \quad (4.3)$$

For each hidden layer $\ell = 2, \dots, \bar{L} - 1$, the transformation is

$$\mathcal{N}^\ell(\mathbf{x}) = \sigma\left(W^{(\ell)} \mathcal{N}^{\ell-1}(\mathbf{x}) + \mathbf{b}^{(\ell)}\right), \quad (4.4)$$

where $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ denotes a nonlinear activation function, applied componentwise when acting on vectors. The output of each neuron is obtained by applying this function to the weighted sum of its inputs [38]. Finally, the output layer is typically linear:

$$\mathcal{N}^{\bar{L}}(\mathbf{x}) = W^{(\bar{L})} \mathcal{N}^{\bar{L}-1}(\mathbf{x}) + \mathbf{b}^{(\bar{L})}. \quad (4.5)$$

Thus, the full network is the composition

$$z_\theta(\mathbf{x}) = (\mathcal{N}^{\bar{L}} \circ \mathcal{N}^{\bar{L}-1} \circ \dots \circ \mathcal{N}^0)(\mathbf{x}), \quad (4.6)$$

with parameter set $\theta = \{W^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{\bar{L}}$.

The formulation above defines how a neural network maps inputs to outputs through affine transformations and nonlinear activations. To understand its effectiveness, it is essential to examine its ability to approximate general functions—a property central to representing PDE solutions [76]. This leads to the following universal approximation.

4.2.2 Approximation Theory and Activation

The universal approximation theorem, introduced by Cybenko [39] and further developed by Hornik et al. [74], guarantees that for any continuous function $G : \Omega \rightarrow \mathbb{R}$ on a compact domain $\Omega \subset \mathbb{R}^n$ and any $\varepsilon > 0$, there exists a network z_θ such that

$$\sup_{\mathbf{x} \in \Omega} |z_\theta(\mathbf{x}) - G(\mathbf{x})| < \varepsilon. \quad (4.7)$$

The theorem also asserts that multilayer feedforward networks, even those with as few as one hidden layer, include a class of universal approximators [74]. The validity of this result depends on the activation function. Universal approximation requires that the activation function is not a polynomial [100]. This theoretical foundation motivates the use of nonlinear activation functions, discussed next, which allow

networks to capture the complex and nonlinear behaviour of PDE-governed systems.

Activation Functions. As we discussed earlier, neural networks leverage nonlinear activation functions to introduce complexity in the model [100]. Without them, the network output is restricted to a linear transformation of the input, regardless of the number of layers [73]. In the PDE setting, such a restriction would prevent the network from capturing the nonlinear solution structures that arise in most models of interest [61].

The activation function affects training behaviour and approximation quality, making it a key design choice in PINN applications [38]. Commonly used activation functions include the Rectified Linear Unit (ReLU), the hyperbolic tangent (tanh), the sigmoid function, and the softplus function. Their definitions and plots are found in Table 4.1 and Figure 4.2, respectively. The sigmoid and tanh activation functions exhibit similar behaviour: the sigmoid maps inputs to the interval $[0, 1]$, whereas the tanh function spans $[-1, 1]$ [77]. In contrast, ReLU and Softplus are unbounded activations, allowing their outputs to grow without limit for large positive inputs [29, 67], whilst both still approach zero as the input becomes increasingly negative [67].

Name	Definition	Reference
ReLU	$\text{ReLU}(x) = \max\{0, x\}$	[124]
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	[67]
Tanh	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	[111]
Softplus	$\ln(1 + \exp(x))$	[54]

Table 4.1: Common activation functions and their definitions.

In the PINN framework, activation functions play an additional role because derivatives of the network output enter the loss functional [134]. Functions such as tanh are often preferred for their infinite differentiability, whereas ReLU is less suitable for second- or higher-order derivatives due to its non-smoothness at zero [160]. Smooth choices such as sigmoid and tanh also provide the regularity required for theoretical estimates of generalisation error in PINNs to hold [118]. When the output is required to remain positive, activation functions such as sigmoid or Softplus may be used. In practice, Softplus is often preferred when the quantity is positive but not bounded above, for example in enforcing non-negative parameters or

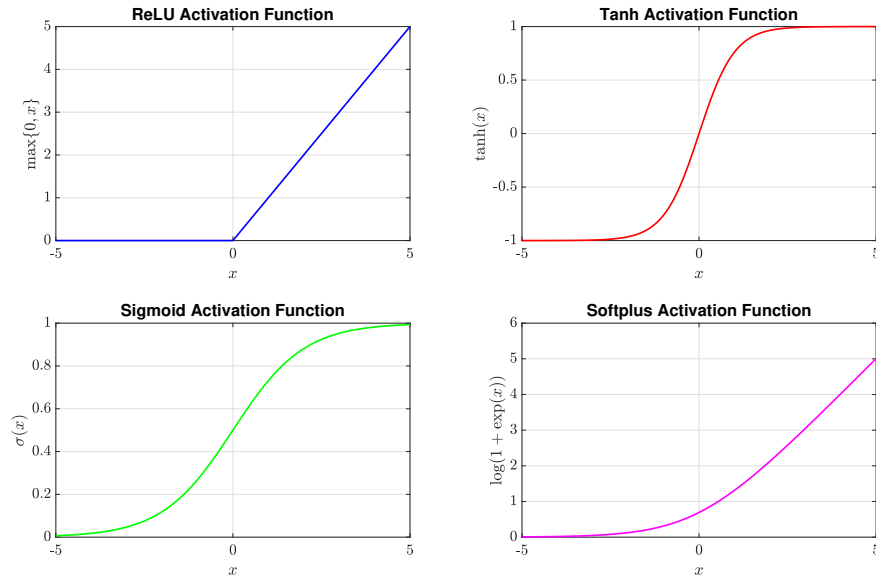


Figure 4.2: Activation functions: ReLU, tanh, sigmoid, and softplus. This figure illustrates the output curves for each activation function over a range of x values, allowing for a visual comparison of their nonlinearity and range.

weights [30, 160]. The choice of activation function determines how the network represents nonlinear relationships, but effective learning also depends on how its parameters are adjusted [160]. Training provides this mechanism by optimising the weights and biases to minimise a chosen loss function, typically through gradient-based methods [67]. We next outline the basic principles of network training and optimisation, which underpin all neural-network formulations used in this thesis.

4.2.3 Training and Optimisation

Training corresponds to adjusting the parameters θ so that the network output approximates the target values as closely as possible [163]. Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, the parameters are obtained by minimising a loss functional that measures the difference between predictions and targets $\mathbf{y}_i$,

$$\mathcal{J}(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} \|z_{\theta}(\mathbf{x}_i) - \mathbf{y}_i\|^2. \quad (4.8)$$

Although other loss functions are possible, this quadratic form is standard in regression tasks and in PINN settings [132]. Parameters θ are updated using gradient-based optimisation. We here consider the standard gradient descent, Stochastic

Gradient Descent (SGD), and the Adaptive Moment Estimation method (Adam).

In gradient descent, the network parameters are updated iteratively to minimise the loss function:

$$\theta_{t+1} = \theta_t - \lambda \nabla_{\theta} \mathcal{J}(\theta), \quad t = 1, 2, \dots \quad (4.9)$$

where $\lambda > 0$ is the learning rate. Each update occurs within an iteration, and one complete pass of the training dataset through the model is referred to as an epoch [123]. For large datasets, computing this full gradient at every iteration can be computationally expensive [67]. SGD addresses this by estimating the gradient from a randomly chosen subset of the data, called a mini-batch, reducing the computational cost per iteration [28], and given by

$$\theta_{t+1} = \theta_t - \lambda \nabla_{\theta} \mathcal{J}_{\mathcal{B}}(\theta_t), \quad (4.10)$$

where $\text{Loss}_{\mathcal{B}}$ denotes the loss evaluated on a mini-batch $\mathcal{B}$ of collocation, boundary, and initial points. This stochastic sampling reduces computational cost and often improves convergence in practice [67].

Building on SGD, adaptive methods such as Adam [90] are widely used in this thesis. At each iteration t , the gradient is computed using a mini-batch $\mathcal{B}_t$,

$$g_t = \nabla_{\theta} \mathcal{J}_{\mathcal{B}_t}(\theta_{t-1}), \quad (4.11)$$

where $\mathcal{J}_{\mathcal{B}_t}$ denotes the loss evaluated on the mini-batch. Adam then updates exponentially weighted estimates of the first and second moments of the gradient. Bias-corrected versions of these estimates are used to account for their initialisation at zero. The parameter update procedure is summarised in Algorithm 3. Adam provides robustness, low memory cost, and fast convergence, particularly in problems with noisy gradients or non-stationary objectives [89]. All such methods rely on computing $\nabla_{\theta} \mathcal{J}(\theta)$ via backpropagation, a special case of Automatic Differentiation (AD). AD applies the chain rule systematically through the network layers, producing exact derivatives at machine precision with a single forward and backward pass [16]. This is more efficient than finite-difference schemes, which require multiple evaluations and suffer from truncation errors [132]. Accurate derivatives

Algorithm 3 Adam optimiser [90]

Require: Step size $\alpha > 0$, decay rates $\beta_1, \beta_2 \in (0, 1)$, $\epsilon > 0$ **Require:** Objective function $\mathcal{J}(\theta)$, initial parameters θ_0 **Require:** Mini-batch sequence $\{\mathcal{B}_t\}$ 1: Initialise first and second moment vectors: $m_0 \leftarrow 0, v_0 \leftarrow 0$ 2: Initialise iteration index $t \leftarrow 0$ 3: **while** not converged **do**4: $t \leftarrow t + 1$

5: Compute gradient:

$$g_t \leftarrow \nabla_{\theta} \mathcal{J}_{\mathcal{B}_t}(\theta_{t-1})$$

6: Update biased moment estimates:

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

7: Compute bias-corrected estimates:

$$\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$$

8: Update parameters:

$$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

9: **end while**10: **return** θ_t

are especially important in PINNs, where spatial and temporal derivatives of the network output enter directly into the loss functional [123]. The overall training procedure is summarised in Algorithm 4.

Algorithm 4 Training a feedforward neural network

1: Initialise parameters $\theta = \{W^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^L$ 2: **for** iteration $k = 1, 2, \dots$ until stopping criterion (e.g. maximum iterations or small gradient norm) is met **do**3: Select batch (or mini-batch) of training points $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ 4: Forward pass: compute network output $z_{\theta}(\mathbf{x}_i)$ 5: Evaluate loss $\mathcal{J}(\theta)$ 6: compute gradient $\nabla_{\theta} \mathcal{J}(\theta)$ by AD

7: Update parameters using Gradient Descent, SGD, or Adam.

8: **end for**

Having introduced the basic structure and training of neural networks, we now turn to their application in the solution of PDEs and its control through the PINN framework.

4.3 PINNs for Solving Nonlinear PDEs

Throughout this chapter, we link all topics by using the general form of a time-dependent PDE:

$$\partial_t h(x, t) = \mathcal{F}[h, f](x, t), \quad (4.12)$$

for $x \in \Omega = (0, L) \subset \mathbb{R}$ and $t \in (0, T]$, subject to the initial condition

$$h(x, 0) = h_0(x), \quad x \in \Omega, \quad (4.13)$$

and boundary condition

$$h(x, t) = g(x, t), \quad (x, t) \in \partial\Omega \times (0, T). \quad (4.14)$$

Here h denotes the state variable, $\mathcal{F}$ is a spatial differential operator, and f represents a prescribed parameter or control input.

Traditional numerical solvers approximate PDE solutions through spatial or temporal discretisation [38, 98]. In contrast, the PINN method represents the solution as a differentiable neural network trained to satisfy the governing equations and boundary conditions [123, 134]. We illustrate the method for a general time-dependent PDE (4.12) with the form

$$\mathcal{F}[h, f](x, t) = A[h](x, t) + f(x, t). \quad (4.15)$$

Here h denotes the state variable, $A[h]$ is a (possibly nonlinear) differential operator acting on $h(x, t)$, and $f(x, t)$ denotes a forcing term. For example, in the thin-film equation, $A[h] = \partial_x(h^n \partial_x h)$, represents a nonlinear fourth-order diffusion operator [126]. If A is independent of h , the formulation reduces to the linear case, such as the heat equation. If A depends on h , the same framework also covers nonlinear PDEs. This general form therefore yields both linear and nonlinear PDEs within a unified framework.

The PDE then reads:

$$\partial_t h + A[h] = f \quad \text{in } Q := \Omega \times (0, T), \quad (4.16)$$

subject to the initial and the Dirichlet boundary conditions (4.13)-(4.14).

In the PINN framework, the unknown solution $h(x, t)$ is represented by a neural network surrogate

$$h(x, t) \approx h_\theta(x, t), \quad (4.17)$$

with trainable parameters θ [160]. The core idea of the PINN framework is to reformulate the solution of the differential equation (4.16), together with its initial and boundary conditions (4.13)-(4.14), as an unconstrained optimisation problem [134]. This is achieved by iteratively adjusting the network's trainable parameters θ (weights and biases) to minimise the following composite loss functional

$$\mathcal{J}_{\text{PINN}}(\theta) = \omega_{\text{PDE}} \mathcal{J}_{\text{PDE}}(\theta) + \omega_{\text{IC}} \mathcal{J}_{\text{IC}}(\theta) + \omega_{\text{BC}} \mathcal{J}_{\text{BC}}(\theta), \quad (4.18)$$

where the weights $\omega_{\text{PDE}}, \omega_{\text{IC}}, \omega_{\text{BC}}$ balance the contributions of the different terms [134].

The PDE term $\mathcal{J}_{\text{PDE}}$ measures the mean squared residual of the PDE (4.16) at collocation points in the domain by $\{(x^{(i)}, t^{(i)})\} \subset Q$

$$\mathcal{J}_{\text{PDE}}(\theta) = \frac{1}{N} \sum_{i=1}^N \left| \partial_t h_\theta(x^{(i)}, t^{(i)}) + A[h_\theta](x^{(i)}, t^{(i)}) - f(x^{(i)}, t^{(i)}) \right|^2. \quad (4.19)$$

where derivatives are computed using automatic differentiation [16]. This avoids mesh discretisation and yields exact derivatives at machine precision [38].

Similarly, the initial and boundary conditions are enforced in the least-squares sense:

$$\mathcal{J}_{\text{IC}}(\theta) = \frac{1}{N_0} \sum_{j=1}^{N_0} |h_\theta(x^{(j)}, 0) - h_0(x^{(j)})|^2, \quad (4.20)$$

$$\mathcal{J}_{\text{BC}}(\theta) = \frac{1}{N_b} \sum_{k=1}^{N_b} |h_\theta(x^{(k)}, t^{(k)}) - g(x^{(k)}, t^{(k)})|^2. \quad (4.21)$$

Training then amounts to solving the unconstrained optimisation problem

$$\min_{\theta} \mathcal{J}_{\text{PINN}}(\theta). \quad (4.22)$$

By minimising the multi-objective loss function (4.18) via gradient-based optimisation (e.g., Adam or SGD) [108], the network simultaneously learns the solution that satisfies the PDE constraint and the given conditions [9]. The overall training

Algorithm 5 PINN training algorithm

Require: Network h_θ ; loss weights $\omega_{\text{PDE}}, \omega_{\text{IC}}, \omega_{\text{BC}}$; training points for the PDE, initial, and boundary conditions.

- 1: Initialise θ .
- 2: **repeat**
- 3: Evaluate $\mathcal{J}_{\text{PDE}}, \mathcal{J}_{\text{IC}}, \mathcal{J}_{\text{BC}}$ from (4.19)-(4.21).
- 4: Form total loss $\mathcal{J}_{\text{PINN}}(\theta)$ using (4.18).
- 5: Update parameters θ using a gradient-based optimiser (e.g., Adam).
- 6: **until** stopping criterion is satisfied (loss tolerance or max epochs).

procedure is summarised in Algorithm 5.

The optimisation problem (4.22) is highly non-convex, and convergence depends on the optimiser and learning rate [160]. The weighting coefficients $\omega_{\text{PDE}}, \omega_{\text{IC}}, \omega_{\text{BC}}$ should also be chosen carefully to maintain a balanced training process [38, 123]. A well-trained network yields a set of parameters θ such that $h_\theta(x, t)$ serves as a mesh-free, differentiable surrogate for the true solution $h(x, t)$.

4.3.1 Case study: thin-film flow over a flexible topography

We apply the PINN framework to solve the one-dimensional thin-film flow on a flexible substrate, using the full-batch optimisation procedure described in Algorithm 5.

The governing equations are given by (3.2) with $\gamma = 0$ and $\phi(h) = 0$, posed on $x \in [0, L]$ with no-flux boundary conditions (3.5). In this example we focus on the forward problem and therefore set $f(x, t) = 0$. The initial conditions are prescribed

$$\begin{aligned} h(x, 0) &= 1 + 0.5 \cos(\pi x/L), \\ s(x, 0) &= 1 + 0.5 \cos(3\pi x/L). \end{aligned} \tag{4.23}$$

We approximate the solution triplet (h, μ, s) using a fully connected neural network with trainable parameters θ , and train it by minimising the composite loss (4.18). The relative weights and training hyperparameters (Table 4.2) were selected following small-scale tuning experiments. In particular, the penalty weights were selected through empirical tuning. Larger weights were assigned to the boundary conditions, since their enforcement was found to be important for stable training and accurate solutions in this example. Smaller weights were sufficient for the initial

Table 4.2: Hyperparameters employed in the PINN approach.

Quantity	Value
Domain length L	3π
Final time T	5.0
PDE weight ω_{PDE}	10.0
IC weights $\omega_{\text{IC,h}}, \omega_{\text{IC,s}}$	1.0, 1.0
BC weights $\omega_{\text{BC,h}}, \omega_{\text{BC,s}}$	10.0, 50.0
Collocation points N_r	20 000
IC points N_0	2 000
BC points N_b	5 000
Network architecture	[2, 64, 64, 64, 64, 64, 3]
Initial learning rate	10^{-3}
Training epochs	15 000

conditions. We employ the Adam optimiser for 15,000 epochs. For comparison, we also solve the system using a classical FEM. Although we did not perform an exact runtime comparison here, the PINN training is slower than the FEM solve for this example which is consistent with the literature [108].

Figure 4.3 shows the resulting FEM and PINN predictions for the film height $h(x, t)$, substrate deformation $s(x, t)$, the combined free-surface height $\mathcal{H}(x, t)$, and the corresponding L_2 error. The agreement between the two methods is remarkable: the PINN-based solution closely matches the FEM solution across all times. The slightly larger error in $\mathcal{H}$ than in h may be attributed to the training setup. Since $\mathcal{H} = h + s$, the error in $\mathcal{H}$ combines the approximation errors arising from both components. In addition, the larger penalty weight assigned to the substrate variable may also affect this balance during training. The small errors further indicate that the PINN model provides an accurate, mesh-free surrogate capable of capturing the thin-film dynamics. Overall, this case study illustrates that the PINN framework provides an accurate and efficient surrogate for nonlinear, coupled PDE systems, successfully capturing the evolution of thin-film profiles on flexible substrates.

The above PINN framework can be applied beyond forward problems. Inverse problems can be addressed by treating unknown parameters or operators as additional trainable variables, whilst control problems can be formulated by embedding control functions directly into the network or the loss functional [123, 133, 134]. This highlights the flexibility of PINNs in settings where traditional numerical solvers face limitations, as we will discuss next.

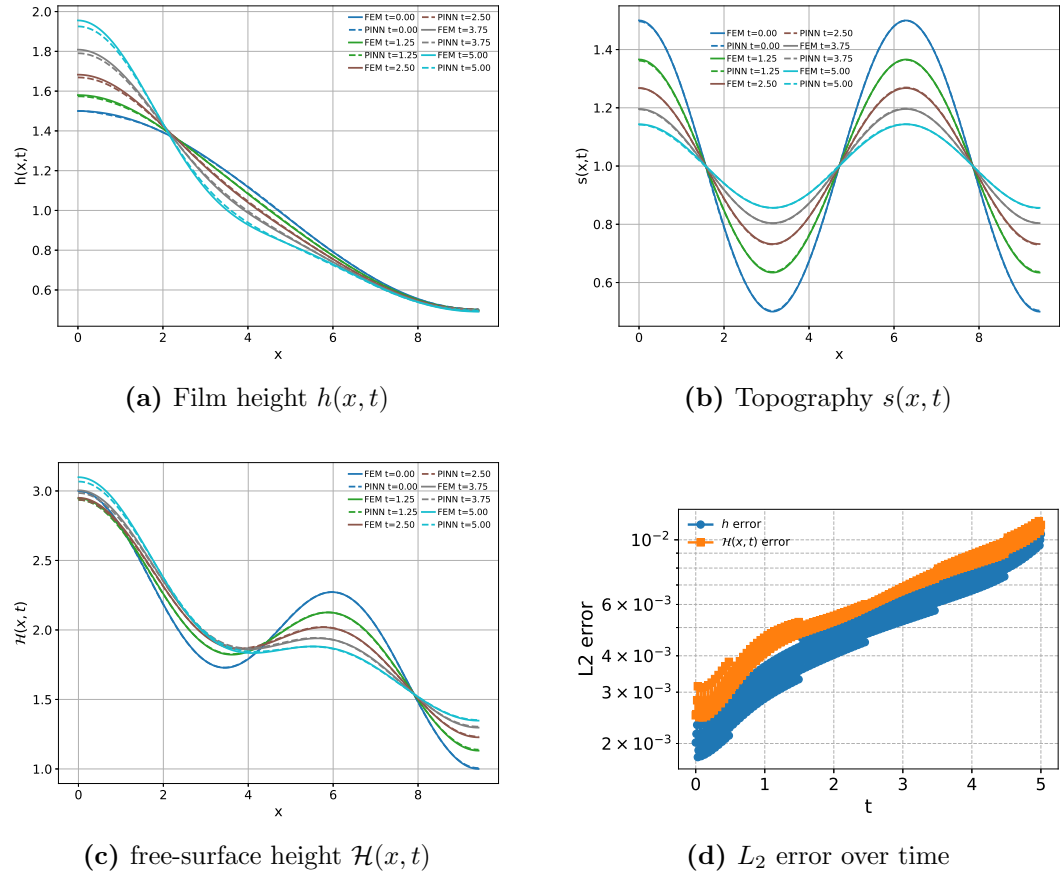


Figure 4.3: Comparison of FEM (solid) and PINN (dashed) solutions for $h(x, t)$, $s(x, t)$, the combined field $h + s$, and the corresponding L_2 error over time.

4.4 Classical Optimal Control of PDEs

We briefly recall the standard formulation of PDE-constrained optimal control problems. The goal is to determine a control input $f(x, t)$ that drives the system state $h(x, t)$ towards a desired target profile $\bar{h}(x)$ whilst penalising excessive control effort [66, 112]. A common quadratic cost functional is

$$\mathcal{J}_{\text{classical}}(f) = \frac{1}{2} \int_0^T \|h(x, t) - \bar{h}(x)\|_{L^2(\Omega)}^2 dt + \frac{\alpha}{2} \int_0^T \|f(x, t)\|_{L^2(\Omega)}^2 dt, \quad (4.24)$$

subject to the governing PDE (4.16) and the initial and boundary conditions (4.13)-(4.14). Here $\alpha > 0$ is a regularisation parameter.

Optimality conditions follow from the Lagrangian formalism [155], introducing an adjoint variable $p(x, t)$ associated with the state equation. The first-order necessary conditions couple the optimal state h^* , optimal adjoint p^* , and optimal control

f^* in the following system:

$$\partial_t h^* + A[h^*] = f^*, \quad h^*(x, 0) = h_0(x), \quad h^*|_{\partial\Omega} = g, \quad (4.25a)$$

$$-\partial_t p^* + (D_h A[h^*])^* p^* = h^* - \bar{h}, \quad p^*(x, T) = 0, \quad p^*|_{\partial\Omega} = 0, \quad (4.25b)$$

$$\alpha f^* + p^* = 0. \quad (4.25c)$$

Here $D_h A[h^*]$ denotes the Fréchet derivative of the operator $A[h]$ with respect to h , evaluated at h^* (cf. equation (2.11)), and $(D_h A[h^*])^*$ is its adjoint in the L^2 inner-product sense.

Classical numerical methods typically solve this system iteratively through forward-backward sweeps. Starting from an initial guess for the control, the state equation is integrated forward in time, the adjoint equation backward, and the control updated using the gradient condition [40]. Whilst theoretically well established and widely used [69, 112, 155], such approaches can be computationally demanding, as they require repeated PDE solves, careful mesh refinement, and full knowledge of the governing operators and system states [76, 123, 134]. These limitations motivate the development of neural-network-based approaches, which we discuss in the next section. In particular, we show how the PINN framework can be adapted to formulate and solve PDE-constrained optimal control problems directly.

4.5 PINNs for Controlling Nonlinear PDEs

The PINN framework provides an alternative approach to PDE-constrained optimal control, as developed by Mowlavi and Nabi [123]. The central idea is to embed the entire optimal control problem into a single learning process by representing both the unknown state and control with neural networks. Specifically, they introduce two networks:

$$h(x, t) \approx h_{\theta_h}(x, t), \quad f(x, t) \approx f_{\theta_f}(x, t), \quad (4.26)$$

which are trained simultaneously by minimising a composite loss functional.

This loss extends the standard PINN formulation (Section 4.3) by including

Algorithm 6 A PINN-Based PDE Control Problem Training Procedure

Require: Networks $h_{\theta_h}, f_{\theta_f}$; training points for PDE, initial, boundary, and terminal conditions; weights $\alpha, \omega_T, \omega_{\text{PDE}}, \omega_{\text{IC}}, \omega_{\text{BC}}$.

- 1: **repeat**
- 2: Compute PDE, initial, boundary, and terminal losses using (4.27).
- 3: Form total loss $\mathcal{J}_{\text{control}}(\theta_h, \theta_f)$.
- 4: Update parameters (θ_h, θ_f) using a gradient-based optimiser (e.g., Adam).
- 5: **until** convergence criterion is satisfied.

additional terms for the control cost and terminal tracking objective:

$$\begin{aligned}
 \mathcal{J}_{\text{control}}(\theta_h, \theta_f) = & \omega_T \frac{1}{N_T} \sum_{\ell=1}^{N_T} |h_{\theta_h}(x^{(\ell)}, T) - \bar{h}(x^{(\ell)})|^2 \\
 & + \omega_{\text{PDE}} \frac{1}{N} \sum_{i=1}^N \left| \partial_t h_{\theta_h}(x^{(i)}, t^{(i)}) + A[h_{\theta_h}](x^{(i)}, t^{(i)}) - f_{\theta_f}(x^{(i)}, t^{(i)}) \right|^2 \\
 & + \alpha \frac{1}{N} \sum_{i=1}^N |f_{\theta_f}(x^{(i)}, t^{(i)})|^2 \\
 & + \omega_{\text{IC}} \mathcal{J}_{\text{IC}}(\theta_h) + \omega_{\text{BC}} \mathcal{J}_{\text{BC}}(\theta_h).
 \end{aligned} \tag{4.27}$$

The remaining terms, corresponding to the initial, and boundary conditions, are defined as in the Section 4.3. Training then corresponds to solving the unconstrained optimisation problem

$$\min_{\theta_h, \theta_f} \mathcal{J}_{\text{control}}(\theta_h, \theta_f), \tag{4.28}$$

which is carried out using gradient-based methods such as Adam [89].

The physics-based terms ensure that $(h_{\theta_h}, f_{\theta_f})$ satisfy the PDE constraints, whilst the remaining terms guide the networks toward the optimal state and control [123]. Training, as shown in Algorithm 6, updates both parameter sets, θ_h and θ_f , so that $h_{\theta_h}(x, t)$ reproduces the system dynamics and $f_{\theta_f}(x, t)$ yields the optimal control input. This allows gradient-based training (such as with the Adam optimiser) to be applied directly to both networks [14].

Compared with classical adjoint-based optimisation (Section 4.4), the PINN formulation offers several advantages. It simplifies the optimisation routine by eliminating the need for complex analytical steps and iterative solution loops required by classical adjoint methods [123, 160]. The mesh-free nature of PINNs allows collocation points to be chosen flexibly, making the approach well suited for nonlinear

or high-dimensional control problems where traditional discretisation techniques become computationally demanding [25, 49, 108].

Beyond forward and control problems, the PINN framework can also be extended to data-driven discovery, where the objective is to infer unknown operators or model parameters directly from partial observations of the state.

4.6 Data-Driven Discovery of PDEs

Thus far, we have applied the PINNs setting in two contexts: solving forward problems, where the governing PDE and its operator $\mathcal{F}$ are fully known, and controlling PDEs, where the PDE is specified but the optimal control input must be determined. A further extension arises when the governing operator in the general form

$$\partial_t h(x, t) = \mathcal{F}[h, f](x, t) \quad (4.29)$$

is only partially known or completely unknown. In such cases, the PINN framework can be adapted to infer $\mathcal{F}$ directly from data whilst still enforcing physical structure through the loss functional. We adopt the formulations of Raissi [132], Raissi et al. [134], which incorporate either parametric identification of unknown coefficients or nonparametric discovery of the operator itself within the training objective.

Parametric discovery. Suppose $f \equiv 0$ and $\mathcal{F}$ is known up to a finite parameter vector $\xi \in \mathbb{R}^m$, so that we write $\mathcal{F}_\xi[h]$. Given scattered measurements $\{(t^i, x^i, h^i)\}_{i=1}^N$ and collocation points $\{(t_r^j, x_r^j)\}_{j=1}^{N_r}$, we train a state network h_{θ_h} whilst simultaneously inferring ξ . In general, the prediction accuracy improves as the number of measurements N increases, since the training loss is constrained by more information about the underlying dynamics [134].

We then minimise

$$\begin{aligned} \mathcal{J}_{\text{par}}(\theta_h, \xi) = & \frac{1}{N} \sum_{i=1}^N |h_{\theta_h}(t^i, x^i) - h^i|^2 \\ & + \frac{1}{N_r} \sum_{j=1}^{N_r} |\partial_t h_{\theta_h}(t_r^j, x_r^j) - \mathcal{F}_\xi[h_{\theta_h}](t_r^j, x_r^j)|^2 \\ & + \omega_{\text{IC}} \mathcal{J}_{\text{IC}}(\theta_h) + \omega_{\text{BC}} \mathcal{J}_{\text{BC}}(\theta_h). \end{aligned} \quad (4.30)$$

The first term enforces data fidelity, the second enforces the PDE residual using the parametrised operator $\mathcal{F}_\xi$, and the remaining terms penalise deviations from initial and boundary conditions as in Section 4.3. Training follows the same loop as in Algorithm 6, now with parameter set (θ_h, ξ) and loss $\mathcal{J}_{\text{par}}$.

Nonparametric discovery. If $\mathcal{F}$ is fully unknown, it can be modelled directly by a neural network $\mathcal{F}_{\theta_{\mathcal{F}}}$. Using the same data and collocation sets, we then learn $(\theta_h, \theta_{\mathcal{F}})$ by minimising

$$\begin{aligned} \mathcal{J}_{\text{np}}(\theta_h, \theta_{\mathcal{F}}) = & \frac{1}{N} \sum_{i=1}^N |h_{\theta_h}(t^i, x^i) - h^i|^2 \\ & + \frac{1}{N_r} \sum_{j=1}^{N_r} |\partial_t h_{\theta_h}(t_r^j, x_r^j) - \mathcal{F}_{\theta_{\mathcal{F}}}[h_{\theta_h}](t_r^j, x_r^j)|^2 \\ & + \omega_{\text{IC}} \mathcal{J}_{\text{IC}}(\theta_h) + \omega_{\text{BC}} \mathcal{J}_{\text{BC}}(\theta_h). \end{aligned} \quad (4.31)$$

Training again uses Algorithm 6, with parameters $(\theta_h, \theta_{\mathcal{F}})$ and loss $\mathcal{J}_{\text{np}}$.

A crucial validation step is to verify that the learned operator defines a consistent dynamical system. Following Raissi [132], the identified operator is substituted into the PDE, and the resulting trajectory h_{θ_h} is simulated and compared with reference data. Because $\mathcal{F}_{\theta_{\mathcal{F}}}$ is represented by a neural network rather than an explicit formula, traditional solvers such as finite differences or finite elements cannot be applied directly. Instead, automatic differentiation is used to evaluate derivatives within $\mathcal{F}_{\theta_{\mathcal{F}}}$, and the level of agreement with the reference evolution provides a practical validation of the discovery.

Conclusion

This chapter introduced the fundamental principles of neural networks and their use in approximating solutions of PDEs. Particular emphasis was placed on the PINNs framework, which embeds the governing physical laws directly into the training process, offering a mesh-free framework for solving nonlinear PDEs. The framework was then extended to PDE-constrained control problems, where both the state and control functions are learned simultaneously within a unified optimisation setting. Finally, the chapter discussed data-driven discovery of governing equations,

demonstrating how PINNs can infer unknown parameters or even entire differential operators directly from sparse observations. Building on these foundations, the next chapter applies the developed methodology to an actuator design optimisation problem.

Chapter 5

Optimal Actuator Design for Thin-Film Flow on a Flexible Topography

In this chapter, we study actuator placement and control of thin-film flow over a flexible substrate within a single optimisation framework. We propose a one-shot formulation that allows the state, control, and actuator locations to be optimised together. This removes the need for data generation or nested iterations and enables gradient-based optimisation directly from the physical model. The numerical results show that the learned actuator positions and control inputs drive the film surface toward the target profile with reasonable accuracy. Comparison with finite-element simulations confirms the validity of the solution and demonstrates the capability of the proposed method for nonlinear time-dependent PDE control.

5.1 Introduction

Many optimal control problems for PDEs involve not only finding an optimal control input but also designing how and where the control acts. This motivates actuator design, which can be formulated as a bilevel optimisation problem in which an upper-level design (actuator placement or shape) interacts with a lower-level PDE-constrained control problem [42, 81]. Such hierarchical formulations arise in actuator and sensor placement [33], imaging [95], and network optimisation [34].

In this chapter, we address actuator design for thin-film flow over a flexible substrate. Such a model is relevant to coating, environmental, and biomedical applications [32, 75, 137]. The outer design problem determines actuator configuration, whilst the inner control problem minimises a PDE-constrained cost functional. Although several studies have examined optimal control of thin-film systems [35, 72, 93, 110, 152], the coupled design-control setting has received little attention.

Solving bilevel PDE-constrained problems remains computationally demanding. Classical approaches typically either reformulate the lower-level optimisation problem through its first-order optimality conditions, thereby obtaining a single coupled system, or use nested iteration schemes [12, 42, 70]. In the latter approach, the inner problem is solved for each candidate design during the outer iteration [116, 143]. More recent work employs learning-based surrogates [64, 164]. These methods approximate the PDE solution operator or the control-to-state map, enabling efficient gradient-based optimisation [59, 103, 165]. However, these methods often require an additional surrogate-construction or training stage, or repeated PDE solves within nested loops.

We propose a direct one-shot optimisation framework that couples the state, control, and design variables within a single formulation. This approach avoids explicit adjoint derivations or surrogate pretraining whilst remaining applicable to general PDE-constrained design-control problems. Its effectiveness is demonstrated for the open-loop control of thin-film flow over a flexible substrate.

The main contributions of this chapter are as follows:

1. A bilevel optimisation setting for thin-film flow over a flexible topography is

introduced, where the actuator design forms the upper level and is evaluated through the performance of the optimal open-loop control problem at the lower level.

2. The bilevel structure is then reformulated as a single differentiable optimisation in which the state, control, and actuator positions are represented within a unified framework.
3. The resulting one-shot approach allows all variables to be optimised simultaneously without explicit adjoint equations, surrogate models, or nested iterations.

The remainder of this chapter is organised as follows. Section 5.2 introduces the governing PDE and formulates the bilevel optimisation problem. Section 5.3 outlines a supervised data-driven formulation, included to illustrate the existing treatment of bilevel optimisation. Section 5.4 presents the proposed one-shot design-control framework. Section 5.5 reports numerical results demonstrating the performance of the framework, and Section 5.6 concludes the chapter.

5.2 Model and optimality conditions

In this section, we formulate the bilevel PDE-constrained control problem. We begin by describing the governing thin-film equations and cost functional, followed by the derivation of the optimality system obtained via the adjoint method. We consider a two-dimensional thin-film flowing on the underside of a flexible substrate, described by the coupled system:

$$h_t - \frac{1}{3}(h^3 \mu_x)_x = 0, \quad (5.1a)$$

$$\mu + \frac{B_o}{C_a} \mathcal{H} + \frac{1}{C_a} \mathcal{H}_{xx} = 0, \quad (5.1b)$$

$$s_t - c^2 s_{xx} - \frac{\gamma}{C_a} (h_{xx} + B_o h) = \sum_{i=1}^2 f_i(t) \chi_i(x; w_i). \quad (5.1c)$$

We recall all variables introduced in Chapter 3, where the state variables are the film height $h(x, t)$, the auxiliary variable $\mu(x, t)$, and the substrate deformation

$s(x, t)$, defining the free surface as $\mathcal{H}(x, t) = h(x, t) + s(x, t)$. Equation (5.1a) represents mass conservation of the film. Equation (5.1b) defines the auxiliary variable through curvature and pressure contributions. For simplicity, we assume a vanishing Hamaker constant A , introduced in equation (3.3), so that the disjoining-pressure term is omitted from the model. Equation (5.1c) models the elastic response of the substrate driven by actuator forcing.

In contrast to the distributed control setting considered in Chapter 3, the control here is applied through two localised actuators centred at positions w_i and driven by time-dependent inputs $f_i(t)$. Each actuator acts on the substrate through a support function $\chi_i(x; w_i)$ of width 2δ . This function could be chosen, for example, as a Gaussian or rectangular profile. The free-energy dissipation law and the functional setting established in Sections 3.2.1-3.2.2 for this extension model (5.1) remain valid.

The system is solved for $t \in [0, T]$ on a finite domain $\Omega = [0, L]$, subject to the initial conditions

$$h(x, 0) = h_0(x), \quad s(x, 0) = s_0(x), \quad (5.2)$$

and homogeneous Neumann boundary conditions

$$h_x = \mu_x = s_x = 0, \quad \text{at } x = 0, L. \quad (5.3)$$

We define the cost functional to balance matching a target state $\bar{h}(x)$ at the final time and quadratic control regularisation,

$$\mathcal{J}(h, s, \{f_i\}, \{w_i\}) = \frac{1}{2} \int_{\Omega} (\mathcal{H}(x, T) - \bar{h}(x))^2 dx + \frac{\alpha}{2} \int_0^T \sum_{i=1}^2 f_i^2(t) dt, \quad (5.4)$$

where $\alpha > 0$ is a regularisation parameter. The resulting bilevel problem consists of an inner control optimisation over $\{f_i(t)\}$ for fixed actuator locations $\{w_i\}$, and an outer design optimisation over the actuator positions to minimise the overall cost functional:

$$\min_{\{w_i\}} \min_{\{f_i\}} \mathcal{J}(h, s, \{f_i\}, \{w_i\}) \quad \text{s.t. } (h, \mu, s) \text{ satisfy (5.1)-(5.3)}. \quad (5.5)$$

To derive the optimality conditions, we introduce the Lagrangian associated with

(5.5) [112]. With adjoint variables (p, q, r) for (h, μ, s) and $\mathcal{H} = h + s$, define

$$\begin{aligned} \mathcal{L}(h, \mu, s, \{f_i\}, \{w_i\}; p, q, r) := & \frac{1}{2} \int_{\Omega} (\mathcal{H}(x, T) - \bar{h}(x))^2 dx + \frac{\alpha}{2} \int_0^T \sum_{i=1}^2 f_i(t)^2 dt \\ & + \int_0^T \left[\langle p, h_t \rangle + \left(p, -\frac{1}{3}(h^3 \mu_x)_x \right) \right] dt \\ & + \int_0^T \left(q, \mu + \frac{B_o}{C_a} \mathcal{H} + \frac{1}{C_a} \mathcal{H}_{xx} \right) dt \\ & + \int_0^T \left[\langle r, s_t \rangle + \left(r, -c^2 s_{xx} - \frac{\gamma}{C_a} (h_{xx} + B_o h) \right) \right] dt \\ & - \int_0^T \sum_{i=1}^2 f_i(t) \left(r, \chi_i(\cdot; w_i) \right) dt, \end{aligned} \quad (5.6)$$

where $\langle \cdot, \cdot \rangle$ denotes the duality pairing in time and $(\cdot, \cdot)$ denotes the $L^2(\Omega)$ inner product for the remaining spatial terms. Imposing stationarity of (5.6) with respect to all variables, and using the homogeneous Neumann boundary conditions (5.3) to eliminate spatial boundary terms, yields the following forward-adjoint optimality system:

$$\begin{cases} h_t^* - \frac{1}{3}(h^{*3} \mu_x^*)_x = 0, \\ \mu^* + \frac{B_o}{C_a} \mathcal{H}^* + \frac{1}{C_a} \mathcal{H}_{xx}^* = 0, \\ s_t^* - c^2 s_{xx}^* - \frac{\gamma}{C_a} (h_{xx}^* + B_o h^*) = \sum_{i=1}^2 f_i^*(t) \chi_i(x; w_i^*), \end{cases} \quad (5.7a)$$

$$\begin{cases} -p_t^* + h^{*2} \mu_x^* p_x^* + \frac{1}{C_a} (q_{xx}^* + \gamma r_{xx}^*) + \frac{B_o}{C_a} (q^* - \gamma r^*) = 0, \\ \left(-\frac{1}{3} h^{*3} p_x^* \right)_x + q^* = 0, \\ -r_t^* - c^2 r_{xx}^* + \frac{B_o}{C_a} q^* + \frac{1}{C_a} q_{xx}^* = 0, \end{cases} \quad (5.7b)$$

$$\begin{cases} h_x^* = \mu_x^* = s_x^* = p_x^* = q_x^* = r_x^* = 0, & \text{on } \partial\Omega \times (0, T), \\ h^* = h_0, \quad s^* = s_0, & \text{on } \Omega \times \{t = 0\}, \\ p^* = r^* = (\mathcal{H}^* - \bar{h}), & \text{on } \Omega \times \{t = T\}. \end{cases} \quad (5.7c)$$

The optimality condition for the controls is

$$\alpha f_i^*(t) = \int_{\Omega} r^*(x, t) \chi_i(x; w_i^*) dx, \quad i = 1, 2, \quad (5.8)$$

and the gradient with respect to actuator locations is

$$\frac{\partial \mathcal{J}}{\partial w_i} = - \int_0^T f_i^*(t) \int_{\Omega} r^*(x, t) \partial_{w_i} \chi_i(x; w_i^*) dx dt, \quad i = 1, 2. \quad (5.9)$$

The system (5.7a) corresponds to the state dynamics, (5.7b) to the adjoint variables enforcing sensitivity of the cost functional, and (5.7c) to the boundary, initial, and terminal conditions. Together with the control and location conditions (5.8)-(5.9), they form the complete optimality system. Although we do not use the adjoint system in the proposed framework, we derive it here because it underpins the supervised-learning approach discussed next. These adjoint equations are solved offline for many sampled actuator locations to generate training data for a surrogate of the outer objective. By contrast, the direct one-shot formulation optimises the state, control, and actuator locations simultaneously in a single differentiable problem, without surrogate construction or repeated forward-adjoint solves.

5.3 Supervised-learning optimisation

We first review a data-driven approach based on supervised learning, where the control is trained on datasets generated from sampled optimal trajectories [1–3, 83]. In this setting, the upper-level objective in equation (5.5) is approximated using data obtained from offline solving, i.e., using precomputed solutions of the lower-level control problems (5.7)-(5.8).

For any fixed actuator configuration $w = \{w_i\}$, the inner control problem is defined as

$$\{f_i^*\}(t; w) = \arg \min_{\{f_i\}} \mathcal{J}(h, s, \{f_i\}, w) \quad \text{s.t.} \quad (h, \mu, s) \text{ satisfy (5.1) – (5.3)}, \quad (5.10)$$

whose solution requires a complete forward-adjoint solve of the optimality system (5.7)-(5.8). The supervised surrogate then constructs a parametric map $\mathcal{D}_{\theta} : \mathcal{W}_w \rightarrow \mathbb{R}$ that approximates the inner optimal cost $\mathcal{J}^*(w)$. A set of actuator configurations $\{w^{(k)}\}_{k=1}^{N_s} \subset \mathcal{W}_w$ is sampled, and for each sample the inner problem (5.10)

Algorithm 7 Supervised surrogate for the outer objective**Require:** sampling set size N_s ; design domain : $\mathcal{W}_w$.

-
- 1: **for** $k = 1, \dots, N_s$ **do**
 - 2: sample $w^{(k)} \in \mathcal{W}_w$
 - 3: solve inner problem (5.10) (via (5.7)-(5.8)) to obtain $\{f_i^*(\cdot; w^{(k)})\}$
 - 4: set $y^{(k)} = \mathcal{J}^*(w^{(k)})$ using (5.11)
 - 5: **end for**
 - 6: train $\mathcal{D}_\theta$ by (5.12)
 - 7: compute w^* by minimising $\mathcal{D}_\theta$ as in (5.13)
-

is solved to obtain the corresponding label

$$y^{(k)} = \mathcal{J}(h^*(w^{(k)}), s^*(w^{(k)}), \{f_i^*(\cdot; w^{(k)})\}, w^{(k)}). \quad (5.11)$$

The network parameters θ are trained by supervised regression on the dataset $\{(w^{(k)}, y^{(k)})\}_{k=1}^{N_s}$ via

$$\min_{\theta} \frac{1}{N_s} \sum_{k=1}^{N_s} (\mathcal{D}_\theta(w^{(k)}) - y^{(k)})^2, \quad (5.12)$$

and the trained model is subsequently used to approximate the outer objective

$$w^* \in \arg \min_{w \in \mathcal{W}_w} \mathcal{D}_\theta(w), \quad (5.13)$$

with gradients $\nabla_w \mathcal{D}_\theta(w)$ computed by automatic differentiation [16]. The complete procedure is summarised in Algorithm 7. A numerical example illustrating this supervised-learning baseline is provided in Appendix B.1.

This surrogate-based formulation replaces repeated PDE solves at the upper level with evaluations of the trained model. Although effective in principle, the supervised-learning approach incurs high offline cost due to the repeated forward-adjoint solves required for dataset construction [64]. In practice, one may reduce this cost by employing model order reduction techniques to approximate the forward PDE system [31, 131], as explained in Appendix B.2 for the thin-film dynamics (5.1) over a flat substrate without forcing. If such reduced models were extended to the entire optimality system, the overall supervised-learning framework would become substantially more efficient.

To avoid the dependence on precomputed data and adjoint evaluations altogether, we next introduce a direct differentiable optimisation framework that embeds

the bilevel control problem into a single unified formulation.

5.4 Direct actuator design optimisation

We now reformulate the actuator design problem introduced in Section 5.2 as a single-level PDE-constrained optimisation. To construct this formulation, the state variables $(h(x, t), \mu(x, t), s(x, t))$ and the control $f = (f_1(t), f_2(t))$ are approximated by functions depending on finite sets of parameters:

$$(h_\theta, \mu_\theta, s_\theta) : \Omega \times [0, T] \rightarrow \mathbb{R}^3, \quad f_\eta : [0, T] \rightarrow \mathbb{R}^2, \quad (5.14)$$

where θ and η denote vectors of optimisation parameters. These parameters may correspond to different families of parametrised functions, such as polynomial, spectral, neural, or reduced-order representations. The actuator centres are collected in $\omega = (w_1, w_2)$ and are treated as additional optimisation variables. Based on these parameterisations, we define the residuals of the governing PDE (5.1) together with the initial and boundary conditions (5.2)-(5.3):

$$e_{\text{PDE}}(\theta, \eta, \omega) : \begin{cases} \partial_t h_\theta - \frac{1}{3} \partial_x (h_\theta^3 \partial_x \mu_\theta) = 0, \\ \mu_\theta + \frac{B_o}{C_a} (h_\theta + s_\theta) + \frac{1}{C_a} (\partial_{xx} h_\theta + \partial_{xx} s_\theta) = 0, \\ \partial_t s_\theta - c^2 \partial_{xx} s_\theta - \frac{\gamma}{C_a} (\partial_{xx} h_\theta + B_o h_\theta) - \sum_{i=1}^2 f_{\eta,i}(t) \chi_i(x; w_i) = 0, \end{cases} \quad (5.15a)$$

$$e_{\text{BC/IC}}(\theta) : \begin{cases} h_\theta(x, 0) - h_0(x) = 0, \quad s_\theta(x, 0) - s_0(x) = 0, \\ \partial_x h_\theta = \partial_x s_\theta = \partial_x \mu_\theta = 0 \text{ on } \partial\Omega. \end{cases} \quad (5.15b)$$

The residuals vanish when the parameterised fields exactly satisfy the PDE system and its boundary and initial conditions. We then introduce a functional that combines the original cost (5.4) with penalty terms involving the PDE, boundary, and

initial residuals:

$$\mathcal{L}(\theta, \eta, \omega) = \lambda_{\text{cost}} \mathcal{J}(h_\theta, s_\theta, f_\eta, \omega) + \lambda_{\text{PDE}} \|e_{\text{PDE}}(\theta, \eta, \omega)\|^2 + \lambda_{\text{BC/IC}} \|e_{\text{BC/IC}}(\theta)\|^2, \quad (5.16)$$

where $\lambda_{\text{cost}}, \lambda_{\text{PDE}}, \lambda_{\text{BC/IC}} > 0$ are penalty weights balancing the importance of the tracking objective and constraint satisfaction. Minimising this functional allows the state, control, and actuator locations to be optimised simultaneously within a unified framework. The single-level optimisation problem is then written as

$$(\theta^*, \eta^*, \omega^*) = \arg \min_{\theta, \eta, \omega} \mathcal{L}(\theta, \eta, \omega), \quad (5.17)$$

where the parameter sets (θ, η, ω) act as joint optimisation variables and are updated simultaneously using a gradient-based scheme, for instance the Adam optimiser [89]. Here, θ and η are the parameters of the functions looking to approximate the PDE and the control solutions, respectively. The parameter ω updates the actuator positions according to the spatial gradients of $\chi_i(x; w_i)$, moving each actuator toward the region where it most effectively influences the objective. Gradients with respect to (θ, η, ω) can be computed either analytically or automatically through differentiation frameworks [16, 40, 66]. The optimisation loop is summarised in Algorithm 8. By

Algorithm 8 Direct differentiable one-shot optimisation framework

Require: Parameterised $(h_\theta, \mu_\theta, s_\theta)$ and f_η , initial actuator positions ω^0 , learning rate ℓ

- 1: **repeat**
 - 2: Sample collocation points $\{(x_i, t_i)\}_{i=1}^N \subset \Omega \times [0, T]$
 - 3: Evaluate $(h_\theta, \mu_\theta, s_\theta, f_\eta)$ and compute residuals $e_{\text{PDE}}, e_{\text{BC/IC}}$
 - 4: Compute total loss $\mathcal{L}(\theta, \eta, \omega)$ from (5.16)
 - 5: Evaluate gradients $\nabla_\theta \mathcal{L}$, $\nabla_\eta \mathcal{L}$, and $\nabla_\omega \mathcal{L}$
 - 6: Update (θ, η, ω) using a gradient-based scheme.
 - 7: **until** convergence
-

embedding state approximation, control synthesis, and actuator placement within a single optimisation process, the formulation determines all quantities simultaneously, avoiding hierarchical loops and repeated PDE solves.

5.5 Numerical results

The direct minimisation framework developed in the previous section is now applied to the thin-film control problem. The parameterised trial functions $(h_\theta, \mu_\theta, s_\theta)$ and the control inputs $\{f_i(t)\}$ are represented by neural networks, providing mappings for both the state and control variables. We refer the reader to Chapter 4 for more details on PINNs.

We solve the problem (5.17) with initial conditions

$$h_0(x) = 1 + 0.5 \cos\left(\frac{\pi x}{L}\right), \quad s_0(x) = 0, \quad (5.18)$$

and target state

$$\bar{h}(x) = 1 + 0.5 \cos\left(\frac{3\pi x}{L}\right), \quad (5.19)$$

and actuator function

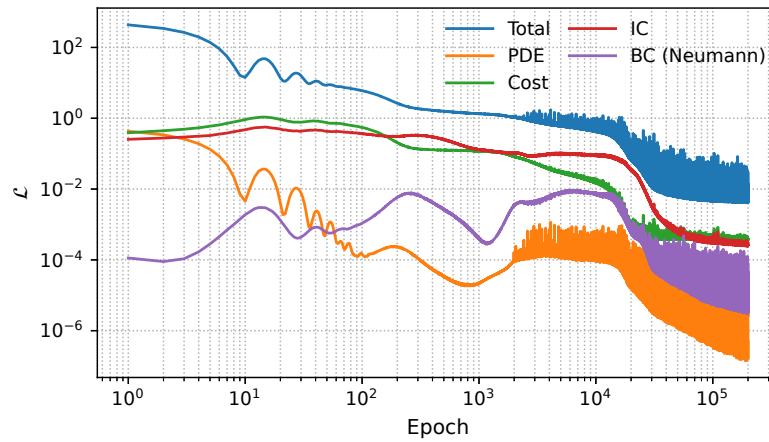
$$\chi_i(x; w_i) = A \exp\left[-\frac{(x - w_i)^2}{\sigma^2}\right], \quad A = 1, \sigma = 0.1L, i = 1, 2. \quad (5.20)$$

Each actuator centre w_i is optimised during training as described in Algorithm 8. Here we use a Gaussian actuator profile rather than the rectangular support used in Appendix B.1. This is because the actuator centres w_i are optimisation variables in the present one-shot formulation, so the loss must remain differentiable with respect to w_i for stable training. The numerical parameters were chosen empirically to obtain stable training. In particular, λ_{PDE} was taken largest to enforce the governing equations strongly, whilst most collocation points were placed in the interior to resolve the PDE residual across the space–time domain. A summary of key parameters is given in Table 5.1.

Figure 5.1 shows the evolution of the total and component losses. All terms decrease overall by several orders of magnitude, although the individual components are not monotone throughout training. This is expected in PINN training, since the different loss components are optimised simultaneously and a decrease in the total loss does not require each individual term to decrease at every stage [123, 160]. The PDE residual remains the largest contributor for most of training, reflecting the

Table 5.1: Key neural-network parameters

Quantity	Value / Description
Domain length	$L = 10$
Final time	$T = 5$
PDE parameters	$C_a = B_o = 1$, $c = 0.5$, and $\gamma = 0$
Regularisation weight	$\alpha = 10^{-3}$
Initial actuator centres	$w_1^0 = 0.1L$, $w_2^0 = 0.9L$
Network architectures:	State network: 2-32 ⁷ -3 (input-hidden-output); Control network: 1-32 ⁴ -2.
Activations	tanh (hidden), Softplus on h_θ
Loss weights	$(\lambda_{\text{PDE}}, \lambda_{\text{BC/IC}}, \lambda_{\text{Neu}}, \lambda_{\text{Cost}}) = (10^3, 1, 10, 10)$
Collocation points	$(N_r, N_b, N_0, N_c) = (2 \times 10^4, 5 \times 10^3, 5 \times 10^3, 1.5 \times 10^4)$
Optimiser	Adam, learning rate 10^{-3} , 2×10^5 iterations

**Figure 5.1:** Evolution of the total and individual loss components during training.

model's continued enforcement of the governing equations. The cost and boundary terms also decrease overall, indicating that the learned controls gradually refine the state toward the target.

The interplay between the actuators, control signals, and resulting film dynamics is summarised in Figures 5.2-5.5. The learned actuator centres shift inward to $w_1^* = 2.6$ and $w_2^* = 7.04$ (Figure 5.2). This is consistent with the supervised result in Appendix B.1, in the sense that both approaches place the optimal actuator centres away from the boundary and into the interior of the domain. We do not expect the locations to coincide exactly, since the two examples use different PDE parameters, actuator supports, and optimisation formulations. These positions align with the x -locations where the substrate displacement $s(x, t)$ reaches its largest values. The corresponding control trajectories (Figure 5.3) show that $f_1(t)$ initially decreases

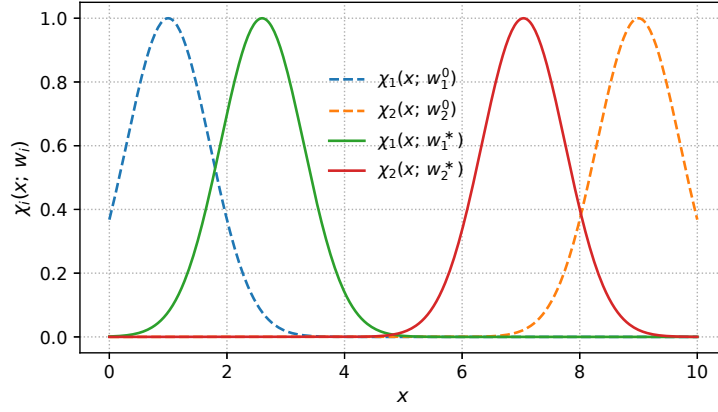


Figure 5.2: Initial (dashed) and optimised (solid) actuator supports $\chi_i(x; w_i)$. The learned positions shift toward the central region, where they most effectively influence the film dynamics.

before rising again after $t \approx 3$, whereas $f_2(t)$ increases steadily throughout the interval. These differing trends generate an uneven forcing on the substrate, which in turn produces a coupled response in the state fields (Figure 5.4): the substrate displacement $s(x, t)$ bends downward on the left and upward on the right, whilst the film thickness $h(x, t)$ shows the opposite behaviour. The resulting free surface $\mathcal{H} = h + s$ (Figure 5.5) approaches the target $\bar{h}(x)$ and nearly coincides with it by $t = T$, demonstrating that the coupled design-control optimisation successfully drives the system toward the target free-surface profile.

The optimised controls were inserted into an independent FEM solver of (5.1) to verify that the resulting state matches the target configuration. The L^2 error between FEM solution and $\bar{h}$ is shown in Figure 5.6, which presents that the FEM solution converges toward the target state, whilst the L^2 difference between the PINN and FEM free surfaces remains below 10^{-1} . This moderate gap is expected: the PINN imposes PDE, boundary, and initial conditions as soft residual penalties over randomly sampled collocation points [164], whereas FEM enforces them exactly in a variational sense on a fixed mesh. Additional discrepancies arise from the smooth activations (`tanh`, `Softplus`) that limit the sharpness of spatial gradients, finite collocation density, and the non-adaptive fixed loss weights [38, 123]. Despite these factors, both models yield the same qualitative behaviour and quantitatively consistent trajectories of the free surface. The results confirm that the proposed direct minimisation formulation can accurately capture the coupled film-substrate

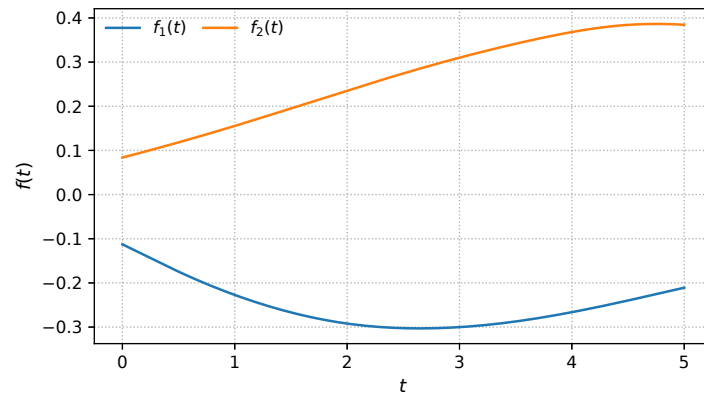


Figure 5.3: Learned control inputs $f_1(t)$ and $f_2(t)$ over time.

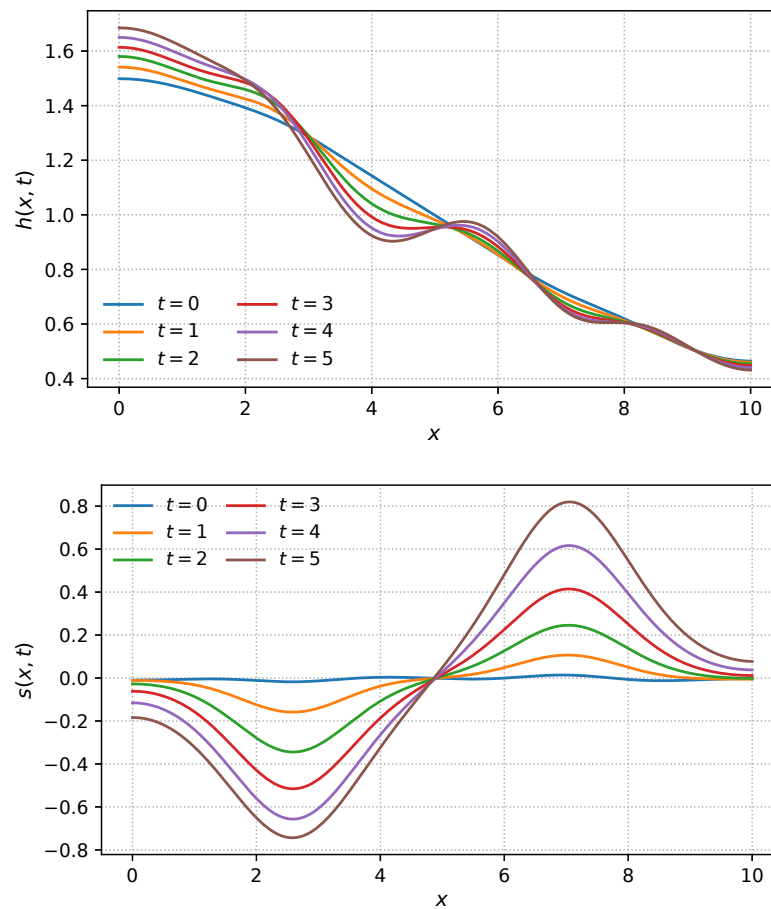


Figure 5.4: Evolution of (top) film thickness $h(x,t)$ and (bottom) substrate displacement $s(x,t)$ at selected times.

dynamics and synthesise effective control and actuator designs within a single differentiable optimisation procedure.

Overall, the direct optimisation achieves stable convergence and physically consistent control of the thin-film system.

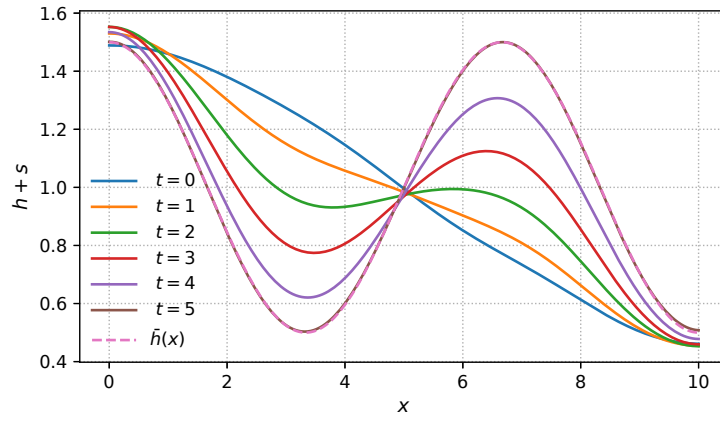
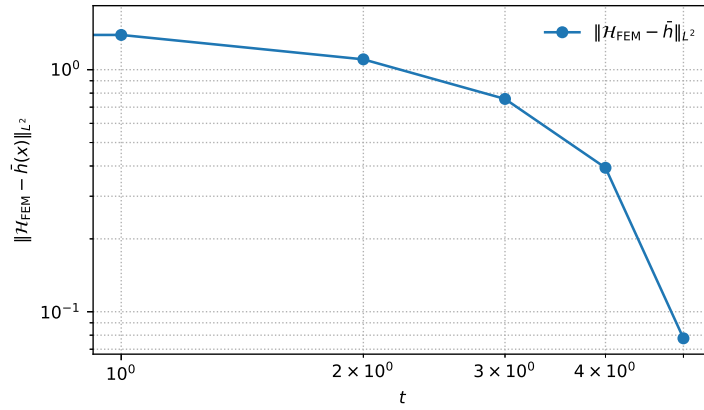
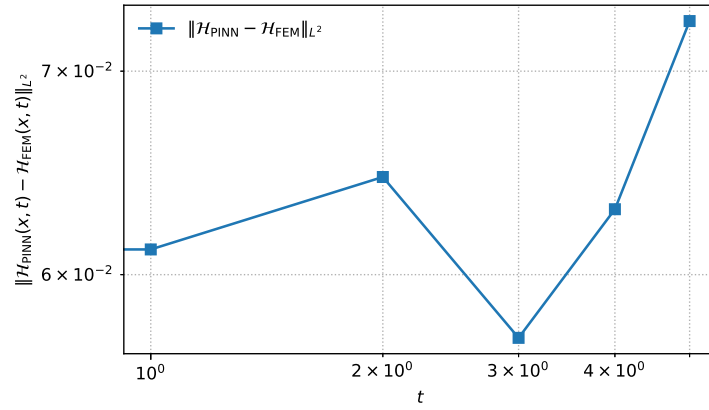


Figure 5.5: Evolution of the total free surface $\mathcal{H} = h + s$ compared with the target profile $\bar{h}(x)$ (dashed).



(a) $\|\mathcal{H}_{\text{FEM}} - \bar{h}\|_{L^2}$ versus time



(b) $\|\mathcal{H}_{\text{PINN}} - \mathcal{H}_{\text{FEM}}\|_{L^2}$ versus time

Figure 5.6: Cross-validation with an independent FEM forward simulation. (a) The FEM solution driven by the learned controls converges toward the target state. (b) The discrepancy between the PINN and FEM free surfaces remains below 10^{-1} , confirming close agreement.

5.6 Conclusion

This chapter presented a direct formulation for bilevel open-loop control problems, in which the state, control, and actuator variables are optimised simultaneously within a single objective. By embedding the governing PDE constraints as differentiable residuals, the approach removes the need for adjoint equations, surrogate models, and nested inner-outer iterations. The resulting one-shot framework provides a unified treatment of control and design, enabling efficient optimisation over both temporal and spatial variables.

The method was applied to the control of thin-film flow over a flexible substrate, demonstrating stable convergence and physically consistent solutions. The learned actuator positions concentrated in regions of dominant film-substrate interaction, whilst the corresponding controls successfully guided the system toward the desired surface profile. Comparison with finite-element forward simulations confirmed that the proposed formulation captures the essential dynamics with high accuracy despite the soft enforcement of the PDE constraints.

In contrast to the supervised data-driven strategy introduced in Section 5.3 and illustrated in Appendix B.1, the present approach achieves comparable accuracy without offline training or repeated adjoint solves. Whilst the formulation avoids adjoint equations and surrogate data, it remains sensitive to the selection of loss weights and collocation density. Future work will include a systematic comparison with the supervised-learning approach illustrated in Appendix B.1 under matched actuator supports, PDE parameters, and optimisation settings. It will also examine strategies to adaptively weight the loss terms or to normalise gradients to mitigate this issue (see [38, 160] and references therein). The framework therefore establishes a computationally efficient foundation for coupled design-control optimisation, upon which the next chapter builds to develop a feedback-based formulation for actuator placement and control synthesis.

Chapter 6

Direct Learning feedback control for actuator design problems

This chapter develops a unified framework for learning feedback control laws and optimising actuator locations in distributed dynamical systems. Starting from the principles of feedback control, we extend the formulation to include actuator design, where the placement of the actuator directly affects the closed-loop dynamics. The value function is approximated by a neural network, enabling simultaneous learning of feedback and actuator parameters in a single optimisation problem. The learning process is implemented through closed-loop rollouts, where both the control policy and actuator positions are updated jointly to minimise the expected cost. The framework is applied to two-dimensional diffusion systems discretised by spectral methods, demonstrating accurate regulation and efficient actuator placement. Comparisons with the analytical Riccati solution show that the proposed approach achieves acceptable performance whilst improving robustness and flexibility under joint optimisation.

6.1 Introduction

Actuator design problems appear in applications such as vibration suppression with piezoelectric devices [127] and active noise control in automotive and aerospace systems [121]. These actuator design applications highlight the need for feedback strategies that remain robust to model uncertainties and disturbances [11, 125]. Unlike open-loop strategies, feedback control achieves this objective by continuously adjusting the control input according to the current system state [91].

Classical approaches to optimal control include Pontryagin’s Minimum Principle and Dynamic Programming [17, 128]. Pontryagin’s Minimum Principle provides necessary optimality conditions in the form of a coupled system for the state and costate variables, which leads to a two-point boundary value problem [11]. This approach is generally associated with open-loop control, where the control depends on time and the initial condition, but not directly on the current state of the system.

By contrast, Dynamic Programming introduces the value function, which represents the minimum cost that can be achieved from a given state and time [43]. This function satisfies the Hamilton-Jacobi-Bellman (HJB) equation, which is a nonlinear partial differential equation posed in the time-state variables [46, 104]. The HJB equation expresses the principle of optimality and characterises the optimal feedback law [13]. Although this framework provides a general description of optimal feedback control, solving the HJB equation for systems arising from PDE discretisation is generally computationally difficult because of the curse of dimensionality [82, 125].

Several approaches have been proposed to address the computational challenges arising from high-dimensional HJB equations. Some key examples include polynomial expansions [80], tensor decompositions [43], and sparse-grid discretisations [50]. Such methods provide low-dimensional surrogates, but remain limited to moderate dimensions [125]. Recently, neural network methods have gained increasing attention. For example, residual minimisation strategies approximate HJB solutions directly [63, 145], whilst data-driven approaches learn feedback policies from numerically generated trajectories [83, 138, 147]. Moreover, adaptive algorithms [125] and Riccati-based structures [1] have been proposed to improve supervised learning of value functions. However, most of these approaches rely on optimality systems,

precomputed datasets, or nested optimisation loops.

A recent approach was proposed by Kunisch and Walter [96]. They showed that the value function of a given dynamical system can be approximated directly by a neural network. This approach is trained from on-policy trajectories and does not require precomputed data. In this work, we extend this direct value-function learning approach to actuator design feedback control problems [81, 83]. More precisely, we develop a formulation in which the value function and actuator design parameters are optimised simultaneously within a single training loop. Gradients are obtained by automatic differentiation through the closed-loop simulations rather than by deriving a separate adjoint system (adjoint-free). The method does not rely on a precomputed training dataset (data-free). In addition, the control and design variables are updated together within a unified optimisation procedure rather than through nested loops (one-shot).

The main contributions of this chapter are as follows:

1. The classical feedback and HJB formulation is extended to include actuator design parameters, providing a unified theoretical foundation that links optimal feedback control with actuator placement.
2. A single-level learning framework is proposed that jointly optimises the value function and actuator configuration in a unified training process. This formulation removes the nested structure of conventional bilevel designs and eliminates the need for precomputed datasets.
3. A neural approximation of the value function is employed that implicitly encodes the optimal feedback law, allowing training without explicitly solving the HJB equation.
4. The approach is applied to the heat equation and compared it with analytical Riccati controllers, showing enhanced performance and robustness through joint optimisation.

The remainder of this chapter is organised as follows. Section 6.2 reviews the classical formulation of dynamic programming and the HJB PDE for nonlinear control systems. Section 6.3 extends this framework to control-affine systems arising

from PDE discretisation and introduces the dependence of system dynamics on actuator locations. Building on this, Section 6.4 formulates actuator placement as an optimisation problem over the value function, leading to a unified treatment of control and actuator design. Section 6.5 presents a joint learning algorithm that simultaneously trains the value-function network and optimises actuator locations through closed-loop simulations. Section 6.6 reports numerical experiments illustrating the performance of the learned feedback laws. Finally, Section 6.7 concludes the chapter.

6.2 Classical Feedback Formulations

We briefly recall the standard formulation of Dynamic Programming (DP), which provides the theoretical foundation for feedback control design; see, e.g., [13, 17, 46, 91]. Consider a general nonlinear control system

$$\dot{\mathbf{h}}(t) = \mathcal{F}(t, \mathbf{h}(t), \mathbf{u}(t)), \quad \mathbf{h}(t_0) = \mathbf{h}_0, \quad (6.1)$$

where $\mathbf{h}(t) \in \mathbb{R}^d$ is the system state, $\mathbf{h}_0 \in \mathbb{R}^d$ is the initial condition, $\mathbf{u}(t) \in \mathbb{R}^m$ is the control, and $\mathcal{F} : [t_0, T] \times \mathbb{R}^d \times \mathbb{R}^m \rightarrow \mathbb{R}^d$ is a sufficiently smooth vector field describing the dynamics. In the previous chapters, the symbol f was used to denote the open-loop forcing term. In this chapter, we use $\mathbf{u}$ for the feedback control input.

Over the finite horizon $[t_0, T]$, the performance index is defined as

$$\mathcal{J}(\mathbf{u}; t_0, \mathbf{h}_0) = g(\mathbf{h}(T)) + \int_{t_0}^T [\ell(\mathbf{h}(t)) + \beta \|\mathbf{u}(t)\|_2^2] dt, \quad \beta > 0, \quad (6.2)$$

where $\ell : \mathbb{R}^d \rightarrow \mathbb{R}$ and $g : \mathbb{R}^d \rightarrow \mathbb{R}$ are the running and terminal cost functions, respectively, and β weights the control effort. For a fixed initial condition $\mathbf{h}(t_0) = \mathbf{h}_0$, an optimal open-loop control $\mathbf{u}^*(t; \mathbf{h}_0)$ can be computed to minimise (6.2). This is the open-loop setting studied in Chapter 5. In the present chapter, we move instead to a feedback setting, where the control is sought as a function of the current state and time. However, such open-loop solutions remain valid only for the prescribed initial condition and are highly sensitive to disturbances or modelling errors [125]. To improve robustness, feedback control seeks a state-dependent law $\mathbf{u}^*(t, \mathbf{h})$ that

adjusts according to the current state [91]. This shift leads to the introduction of the value function and the HJB framework.

We define the value function $V : [t_0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}$, which determines the optimal feedback controls that minimise (6.2) for all admissible states and times $t_0 \leq t \leq T$, as

$$V(t, \mathbf{h}) = \inf_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{u}; t, \mathbf{h}). \quad (6.3)$$

This function represents the minimum cost-to-go from the state $\mathbf{h}$ at time t up to the terminal time T [91].

To derive the corresponding HJB equation, one applies the dynamic programming principle over a short interval $[t, t + \Delta t]$. This gives

$$V(t, \mathbf{h}) = \inf_{\mathbf{u}(\cdot)} \left\{ \int_t^{t+\Delta t} \left[\ell(\mathbf{h}(\tau)) + \beta \|\mathbf{u}(\tau)\|_2^2 \right] d\tau + V(t + \Delta t, \mathbf{h}(t + \Delta t)) \right\}. \quad (6.4)$$

By considering a first-order expansion of $V(t + \Delta t, \mathbf{h}(t + \Delta t))$, and then dividing by Δt and letting $\Delta t \rightarrow 0$, we arrive at the following HJB equation; see [46, 91] for the full derivation.

$$-\partial_t V(t, \mathbf{h}) - \min_{\mathbf{u} \in \mathbb{R}^m} \left\{ \ell(\mathbf{h}) + \beta \|\mathbf{u}\|_2^2 + \nabla_{\mathbf{h}} V(t, \mathbf{h})^\top \mathcal{F}(t, \mathbf{h}, \mathbf{u}) \right\} = 0, \quad V(T, \mathbf{h}) = g(\mathbf{h}). \quad (6.5)$$

The minimisation term in the equation (6.5) defines the Hamiltonian

$$H(t, \mathbf{h}, \mathbf{u}, \nabla_{\mathbf{h}} V) = \ell(\mathbf{h}) + \beta \|\mathbf{u}\|_2^2 + \nabla_{\mathbf{h}} V^\top \mathcal{F}(t, \mathbf{h}, \mathbf{u}), \quad (6.6)$$

and the optimal feedback control is obtained from

$$\mathbf{u}^*(t, \mathbf{h}) = \arg \min_{\mathbf{u} \in \mathbb{R}^m} H(t, \mathbf{h}, \mathbf{u}, \nabla_{\mathbf{h}} V). \quad (6.7)$$

Once $\nabla_{\mathbf{h}} V(\cdot)$ is known, the feedback control can be recovered by solving an optimisation problem in $\mathbf{u}$ [46].

These formulations provide the foundation for extending optimal feedback control to actuator design.

6.3 Continuous-time Feedback Control

We consider a dynamical system whose evolution depends on the actuator configuration w , and which is described by

$$\dot{\mathbf{h}}(t) = A(\mathbf{h}(t)) + B(w)\mathbf{u}(t). \quad (6.8)$$

Such systems arise from the spatial discretisation of PDE control problems, where $A(\mathbf{h})$ denotes the discretised system dynamics and $B(w) \in \mathbb{R}^{d \times N_a}$ is the input operator associated with the actuator configuration $w = (w_1, \dots, w_{N_a})$.

For a fixed location w , the performance of a control trajectory $\mathbf{u}(\cdot)$ is measured by

$$\mathcal{J}(\mathbf{u}; t, \mathbf{h}, w) = g(\mathbf{h}(T; w)) + \int_t^T [\ell(\mathbf{h}(\tau; w)) + \beta \|\mathbf{u}(\tau)\|_2^2] d\tau, \quad (6.9)$$

where $\mathbf{h}(\tau; w)$ evolves according to (6.8) with initial condition $\mathbf{h}(t) = \mathbf{h}_0$. The integration variable τ denotes time within the horizon $[t, T]$.

Following the dynamic programming principle (as in the previous section), the minimum cost achievable from state $\mathbf{h}$ at time t defines the value function

$$V(t, \mathbf{h}; w) = \inf_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{u}; t, \mathbf{h}, w). \quad (6.10)$$

For each fixed w , this function satisfies the following HJB PDE equation

$$\begin{aligned} -\partial_t V(t, \mathbf{h}; w) &= \min_{\mathbf{u} \in \mathbb{R}^{N_a}} \left\{ \ell(\mathbf{h}) + \beta \|\mathbf{u}\|_2^2 + \nabla_{\mathbf{h}} V(t, \mathbf{h}; w)^\top [A(\mathbf{h}) + B(w)\mathbf{u}] \right\}, \\ V(T, \mathbf{h}; w) &= g(\mathbf{h}). \end{aligned} \quad (6.11)$$

The minimisation term in (6.11) defines the Hamiltonian

$$H(\mathbf{h}, \mathbf{u}, \nabla_{\mathbf{h}} V; w) = \ell(\mathbf{h}) + \beta \|\mathbf{u}\|_2^2 + \nabla_{\mathbf{h}} V(t, \mathbf{h}; w)^\top [A(\mathbf{h}) + B(w)\mathbf{u}]. \quad (6.12)$$

We now derive the optimal control $\mathbf{u}^*$ that minimises this Hamiltonian, that is,

$$\frac{\partial H}{\partial \mathbf{u}} = 0. \quad (6.13)$$

Since H is quadratic in $\mathbf{u}$, this condition gives the explicit feedback law [91]

$$\mathbf{u}^*(t, \mathbf{h}; w) = -\frac{1}{2\beta} B(w)^\top \nabla_{\mathbf{h}} V(t, \mathbf{h}; w), \quad (6.14)$$

which expresses the optimal control directly in terms of the gradient of the value function. Substituting (6.14) into the system dynamics (6.8) yields the closed-loop evolution:

$$\dot{\mathbf{h}}(t) = A(\mathbf{h}(t)) - \frac{1}{2\beta} B(w)B(w)^\top \nabla_{\mathbf{h}} V(t, \mathbf{h}(t); w), \quad (6.15)$$

which shows how the actuator configuration w influences the dynamics through the operator $B(w)$.

6.4 Actuator Design through Value Function

The feedback law (6.14) characterises the optimal control for a fixed actuator location w . In practice, however, the actuator placement also influences the system dynamics and, consequently, the achievable performance. This motivates the optimisation of the configuration w itself to minimise the resulting closed-loop cost, leading to

$$w^* = \arg \min_w \left\{ \underbrace{\inf_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{u}; t_0, \mathbf{h}_0, w)}_{V(t_0, \mathbf{h}_0; w)} \right\}. \quad (6.16)$$

The inner problem defines the value function for a fixed configuration w , as in (6.10), whilst the outer optimisation selects the actuator configuration w^* that minimises the total cost.

Solving this bilevel structure is computationally expensive, as discussed in Chapter 5, since each evaluation of the inner problem requires a full control solution. Recent surrogate methods approximate the value function across actuator configurations to reduce this cost [83], yet they still rely on multiple control solves and large datasets. To overcome these limitations, we propose a direct learning framework that unifies control and design in one process. It jointly updates the feedback law and actuator configuration through a single optimisation.

We approximate the value function using a neural network representation¹:

$$V(t, \mathbf{h}; w) \approx V^{\text{NN}}(t, \mathbf{h}; \theta), \quad (6.17)$$

where θ denotes the network parameters. The actuator configuration w is not an explicit network input; it influences the approximation parametrically through the system dynamics and cost, both dependent on $B(w)$. The corresponding learned feedback control is

$$\mathbf{u}^{\text{NN}}(t, \mathbf{h}; \theta, w) = -\frac{1}{2\beta} B(w)^\top \nabla_{\mathbf{h}} V^{\text{NN}}(t, \mathbf{h}; \theta), \quad (6.18)$$

which provides a continuous-time mapping from the state $\mathbf{h}$ to the control input $\mathbf{u}$.

Using this feedback control law, the inner optimisation in (6.16) can be eliminated by evaluating the cost functional (6.9) directly under the closed-loop dynamics (6.15). Consequently, the actuator design problem reduces to a single joint optimisation over (θ, w) :

$$(\theta^*, w^*) = \arg \min_{\theta, w \in \mathcal{W}} \left\{ g(\mathbf{h}(T)) + \int_{t_0}^T \left[\ell(\mathbf{h}(t)) + \beta \|\mathbf{u}^{\text{NN}}(t, \mathbf{h}(t); \theta, w)\|_2^2 \right] dt \right\}, \quad (6.19)$$

subject to the closed-loop dynamics

$$\dot{\mathbf{h}}(t) = A(\mathbf{h}(t)) + B(w) \mathbf{u}^{\text{NN}}(t, \mathbf{h}(t); \theta, w), \quad \mathbf{h}(t_0) = \mathbf{h}_0. \quad (6.20)$$

Note that $\mathbf{h}(t)$ depends on the parameters θ and w , i.e., $\mathbf{h}(t) = \mathbf{h}(t; \theta, w)$. During training, (θ, w) are updated jointly so that V^{NN} remains consistent with the dynamics induced by the current w . At convergence, V^{NN} approximates the value function associated with the optimal actuator configuration w^* , as we discuss in the next section. Gradients of the objective with respect to (θ, w) are computed automatically via differentiation through the closed-loop dynamics [16], removing the need for manual derivation of adjoint equations.

¹We refer the reader to Chapter 4 for a general discussion of neural network architectures, training procedures, and optimisation methods, including automatic differentiation and the Adam optimiser, which underpin the learning framework presented here.

6.5 Joint Learning of Feedback Control and Actuator Design

We now describe the training procedure for jointly learning the feedback control law and actuator locations. The value-function parameters θ and actuator configuration w are optimised jointly by minimising a discrete approximation of the continuous-time cost (6.19). The cost is evaluated along trajectories generated by numerically integrating the closed-loop dynamics over the time horizon.

We begin by defining the training set of initial conditions. Let

$$\mathcal{B} = \{(t_0^{(i)}, \mathbf{h}_0^{(i)})\}_{i=1}^{N_{\text{train}}} \quad (6.21)$$

denote a collection of N_{train} initial states from which the closed-loop system will be simulated. During training, we draw a mini-batch from $\mathcal{B}$, and for each element of the batch we simulate the closed-loop dynamics.

The time interval $[t_0, T]$ is discretised into N uniform time steps with step size $\Delta t = (T - t_0)/N$. For each initial condition $(t_0^{(i)}, \mathbf{h}_0^{(i)})$ in the mini-batch, we generate a discrete trajectory

$$\{\mathbf{h}_n^{(i)}\}_{n=0}^N, \quad \mathbf{h}_0^{(i)} = \mathbf{h}_0^{(i)}, \quad (6.22)$$

where the superscript i indexes the sample in the batch and the subscript n denotes the time step. At each time step, the feedback control is computed from the value-network gradient via the discrete analogue of (6.18):

$$\mathbf{u}_n^{\text{NN},(i)} = -\frac{1}{2\beta} B(w)^\top \nabla_{\mathbf{h}} V^{\text{NN}}(t_n, \mathbf{h}_n^{(i)}; \theta). \quad (6.23)$$

This requires evaluating the neural network V^{NN} and its gradient at the current state $\mathbf{h}_n^{(i)}$. The state is then advanced using the closed-loop dynamics,

$$\mathbf{h}_{n+1}^{(i)} = \mathbf{h}_n^{(i)} + \Delta t \left[A(\mathbf{h}_n^{(i)}) + B(w) \mathbf{u}_n^{\text{NN},(i)} \right]. \quad (6.24)$$

Thus, both θ and w influence every trajectory: θ enters through the network gradient, and w enters through the actuator operator $B(w)$.

For each trajectory, the discrete approximation of the cost in (6.19) is

$$\widehat{\mathcal{J}}(\theta, w; t_0^{(i)}, \mathbf{h}_0^{(i)}) = g(\mathbf{h}_N^{(i)}) + \sum_{n=0}^{N-1} \left[\ell(\mathbf{h}_n^{(i)}) + \beta \|\mathbf{u}_n^{\text{NN},(i)}\|_2^2 \right] \Delta t. \quad (6.25)$$

Computing $\widehat{\mathcal{J}}$ therefore requires evaluating V^{NN} and $\nabla_{\mathbf{h}} V^{\text{NN}}$ at all time steps along the simulated trajectory. However, this is different from solving the HJB equation directly. Here, the value function is approximated by a neural network and evaluated only along the closed-loop rollouts generated from the training initial conditions, rather than on a global discretisation of the full time-state domain.

Finally, the empirical loss over the mini-batch is

$$\mathcal{L}(\theta, w) = \frac{1}{B} \sum_{i=1}^B \widehat{\mathcal{J}}(\theta, w; t_0^{(i)}, \mathbf{h}_0^{(i)}). \quad (6.26)$$

The joint optimisation problem then is

$$(\theta^*, w^*) = \arg \min_{\theta, w} \mathcal{L}(\theta, w). \quad (6.27)$$

The optimisation is carried out using gradient-based updates (Adam optimiser [89]) on both θ and w , with gradients computed automatically through the closed-loop dynamics via automatic differentiation [16].

Algorithm 9 outlines the joint learning procedure. At each iteration, the neural feedback control (6.23) generates closed-loop trajectories according to the dynamics (6.24). The discrete cost (6.25) is accumulated along each trajectory and averaged over the mini-batch to form the empirical loss (6.26). Gradients of this loss with respect to (θ, w) are then obtained by differentiating through the rollout equations, and parameters are updated using Adam [16, 89]. This unified optimisation removes the nested structure inherent in classical actuator design approaches.

Algorithm 9 Joint training of feedback control and actuator configuration

1. Initialise network parameters θ and actuator positions $w_i \in [\delta, L - \delta]$.
 2. **Repeat** until convergence:
 - (a) Sample a mini-batch of initial conditions $\{(t_0^{(i)}, \mathbf{h}_0^{(i)})\}_{i=1}^B$.
 - (b) Generate closed-loop rollouts using equations (6.23)-(6.24).
 - (c) Evaluate the empirical loss $\mathcal{L}(\theta, w)$ from (6.26).
 - (d) Compute gradients $\nabla_{\theta}\mathcal{L}$ and $\nabla_w\mathcal{L}$ via automatic differentiation.
 - (e) Update θ, w using Adam.
 3. **Return** (θ^*, w^*) .
-

6.6 Numerical results

We consider a prototypical example related to optimal actuator placement for a heating system, following the modelling approaches in [81, 83]. For completeness, we recall the underlying continuous model

$$\begin{cases} \partial_t h(x, t) - \nu \Delta h(x, t) = \chi_{\omega(w)}(x) u(t), & \text{in } \Omega \times (0, T], \\ h(x, t) = 0, & \text{on } \partial\Omega \times (0, T], \\ h(x, 0) = h_0(x), & \text{on } \Omega. \end{cases} \quad (6.28)$$

where $\chi_{\omega(w)}$ denotes the actuator support depending on the configuration w . After semi-discretisation in space, this system reduces to a finite-dimensional linear system of the form

$$\dot{\mathbf{h}}(t) = A \mathbf{h}(t) + B(w) \mathbf{u}(t). \quad (6.29)$$

where A denotes the semi-discrete diffusion operator and $B(w)$ is the input operator corresponding to actuators centred at positions w . This finite-dimensional system is obtained by applying the finite element method to the heat equation (6.28).

The numerical experiments serve two purposes. First, we examine a one-dimensional heat equation whose finite element discretisation reduces to a scalar system. This provides a benchmark problem in which the optimal feedback and actuator location are known analytically. This allows us to verify that the learned value function

and actuators recover the exact solution. Second, we apply the method to a two-dimensional heat equation to assess its behaviour on a higher-dimensional PDE model.

In all cases, the discrete training objective is

$$\widehat{\mathcal{J}} = g(\mathbf{h}_N) + \sum_{n=0}^{N-1} \beta \|\mathbf{u}_n\|_2^2 \Delta t, \quad (6.30)$$

with terminal cost $g(\mathbf{h}) = \frac{\alpha}{2} \|\mathbf{h}\|_2^2$. The well-posedness of this control setting is discussed in [81]. In all numerical experiments, the control objective is to regulate the temperature field toward its equilibrium state. All examples follow the joint optimisation procedure in Algorithm 9.

6.6.1 One-dimensional scalar test problem

As a first test, we consider the one-dimensional setting, in which the dynamics in (6.29) reduce to a scalar equation obtained by applying a finite-element discretisation with a single basis function, i.e. a piecewise linear hat function defined on $[0, 1]$:

$$\hat{\phi}(x) = \begin{cases} 2x, & 0 \leq x \leq \frac{1}{2}, \\ 2(1-x), & \frac{1}{2} \leq x \leq 1, \\ 0, & \text{otherwise,} \end{cases} \quad (6.31)$$

and modelling the actuator as an interval of half-width δ centred at w yields the scalar system

$$\dot{h}(t) = A h(t) + B(w) u(t), \quad (6.32)$$

where $A = M^{-1}K$ (cf. equation (2.38)) and

$$B(w) = \int_{w-\delta}^{w+\delta} \hat{\phi}(x) dx, \quad w \in [\delta, 1-\delta]. \quad (6.33)$$

Here we intentionally use a single finite element basis function in order to obtain a scalar benchmark problem. The aim is not to provide an accurate spatial approximation of the heat equation, but rather to construct a reduced setting in which the

optimal feedback law and actuator location can be derived analytically. With this choice, the semi-discrete system has a single degree of freedom and therefore reduces to the scalar equation (6.32).

Since the dynamics are linear and the cost functional is quadratic, this problem falls within the Linear Quadratic Regulator framework [91]. In this setting, the value function is sought in quadratic form, and substitution into the HJB equation yields a Riccati differential equation for the coefficient of that quadratic form [1, 154]. The resulting closed-form solution provides an analytic reference for validating the learned value function and feedback law; further details are given in Appendix C.1. For fixed w , the optimal-feedback problem with cost

$$J_w(u) = \frac{\alpha}{2} h(T)^2 + \frac{\beta}{2} \int_0^T u(t)^2 dt, \quad (6.34)$$

reduces to solving the scalar Riccati equation. With the quadratic ansatz $V(t, h; w) = \frac{1}{2} P(t; w) h^2$, one obtains the backward Riccati ODE (more details can be found in Appendix C.1)

$$P'(t) = \frac{B(w)^2}{\beta} P(t)^2 - 2A P(t), \quad P(T; w) = \alpha. \quad (6.35)$$

Introducing the variable $Q(t; w) = 1/P(t; w)$ yields the linear ODE

$$Q'(t) - 2A Q(t) = -\frac{B(w)^2}{\beta}, \quad Q(T; w) = \frac{1}{\alpha}, \quad (6.36)$$

with explicit solution

$$Q(t; w) = \frac{B(w)^2}{2A\beta} + \left(\frac{1}{\alpha} - \frac{B(w)^2}{2A\beta} \right) e^{-2A(T-t)}. \quad (6.37)$$

Evaluating the value function at $t = 0$ gives the optimal cost

$$J(w) = V(0, h_0; w) = \frac{h_0^2}{2} P(0; w) = \frac{h_0^2}{2Q(0; w)}. \quad (6.38)$$

Because $J(w)$ is strictly decreasing in $B(w)^2$, minimising (6.38) with respect to w yields the closed-form optimal actuator location (more details are given in Ap-

pendix C.1):

$$w^* = \frac{1}{2}, \quad (6.39)$$

together with the exact optimal trajectory and feedback law. This analytical solution provides a benchmark for assessing the accuracy of the learned value function, feedback, and actuator update.

We now turn to the numerical results, where we apply the proposed joint learning framework to recover the optimal feedback law and optimal actuator position in this scalar setting. A learning-rate scheduler was also used during training to improve convergence [160]. Training parameters are listed in Table 6.1. Figure 6.1 plots the

Domain Length L	1
Time horizon T	1
Time-step Δt	0.01
Hidden layers	[32, 32, 32, 32] (Tanh)
Epochs	2×10^4
Learning rate (weights)	10^{-2}
Learning rate (actuator w)	10^{-2}
Scheduler	decay 0.9 every 500 epochs
Cost weights	$\alpha = 400, \beta = 0.1$
Actuator domain	$w \in [L + \delta, L - \delta]$

Table 6.1: Neural-network training parameters

total loss $\mathcal{L}$ together with its control-cost and terminal-cost components on a log-log scale versus epoch. The rapid initial decrease of the terminal loss shows effective enforcement of the terminal condition $h(T) \approx 0$, whilst the slower refinement of the control loss reflects optimisation of the feedback effort. By roughly 10^3 epochs all curves become nearly flat, indicating convergence of both the neural approximation and the actuator parameter w . Figure 6.2 demonstrates the evolution of the learned actuator position w during training. The neural network was initialised with $w = 0.1$ and converged to $w = \frac{1}{2}$, which is in exact agreement with the analytical result as in equation (6.39). To verify that the learned controller reproduces the analytical LQR solution, we compare the neural-network outputs with the corresponding closed-form solutions. Figures 6.3-6.4 display the state $h(t)$ and the control trajectories $u(t)$ for initial states $h_0 \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, respectively. The excellent agreement shows that the network accurately reproduces both the dynamics and the optimal

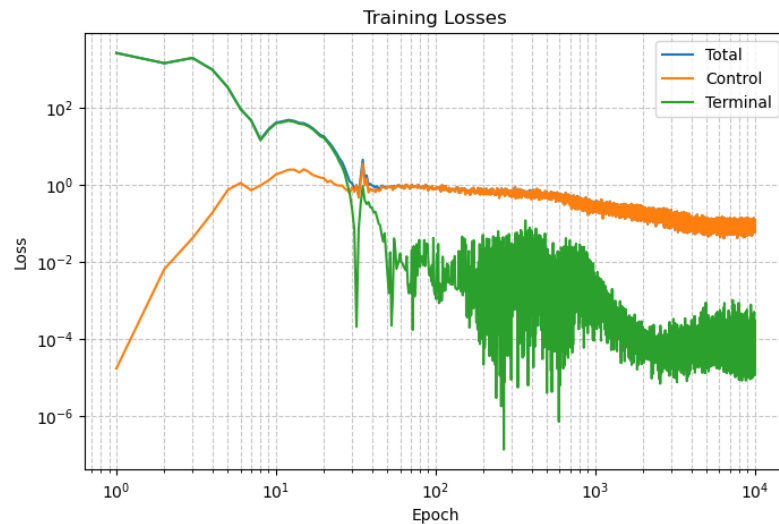


Figure 6.1: Training losses (total, control, and terminal) versus epoch on a log-log scale during neural network training. The total-loss curve is partially obscured where it nearly overlaps with the control-loss curve.

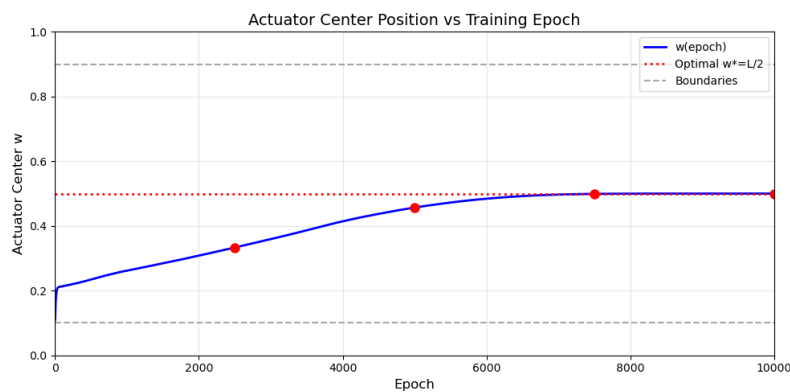


Figure 6.2: Evolution of actuator centre position w during neural network training.

feedback across this range of h_0 . Finally, Figure 6.5 plots the absolute error $|h(T)|$ versus initial conditions h_0 . Across a hundred $h_0 \in [0, 0.5]$, the error remains below 2×10^{-3} , demonstrating that the neural-network controller achieves high-accuracy regulation even when evaluated on a denser set of 100 initial conditions than the set used during training.

6.6.2 Two-dimensional heat equation

We next consider the spatial discretisation of the two-dimensional heat equation, which leads to a system of the form (6.29), posed on the unit square $\Omega = [0, 1]^2$ with homogeneous Dirichlet boundary conditions. The spatial discretisation follows the

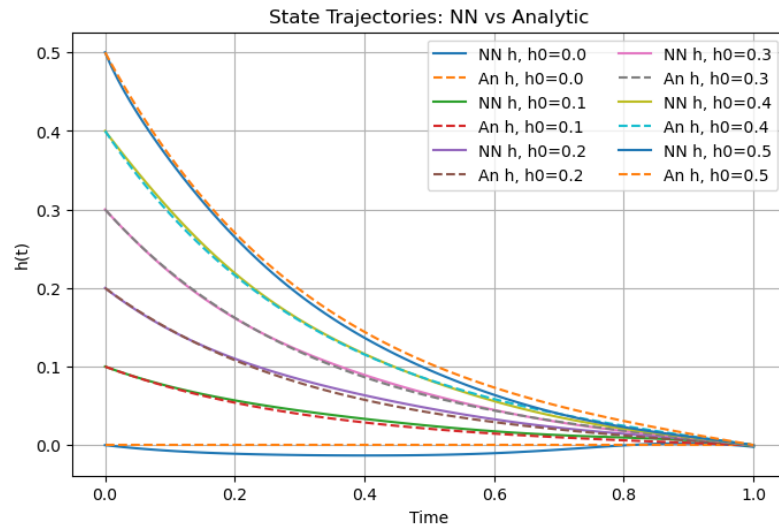


Figure 6.3: State trajectories: neural-network (solid) vs. analytical (dashed) for various h_0 .

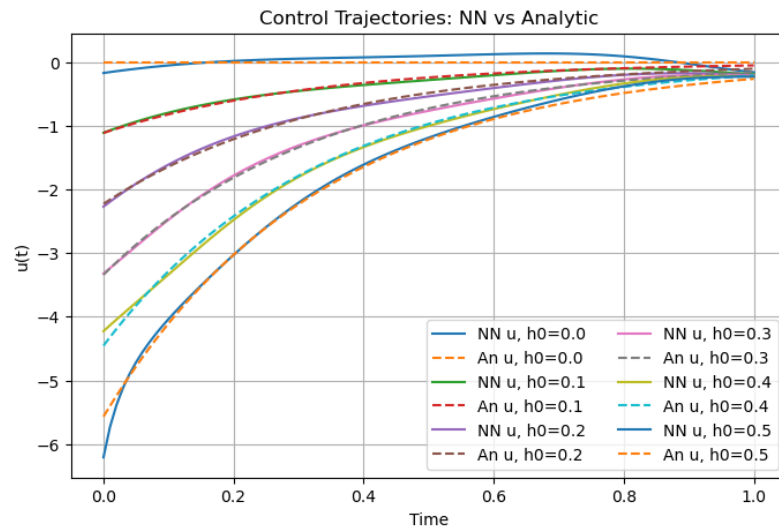


Figure 6.4: Control trajectories: neural-network (solid) vs. analytical (dashed) for various h_0 .

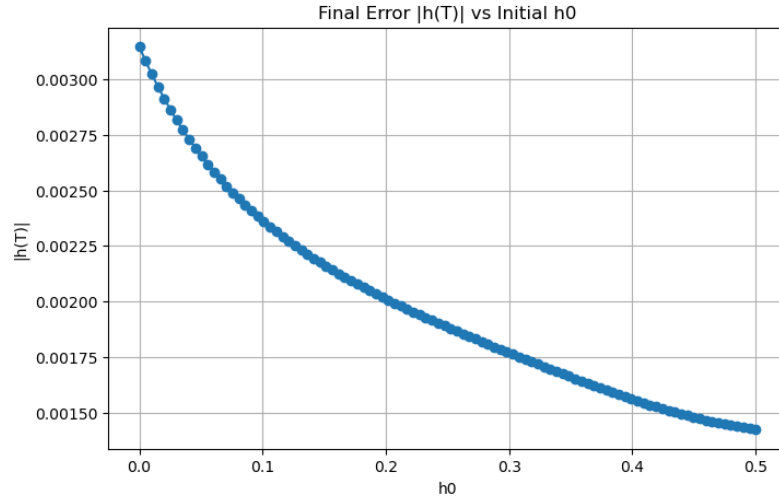


Figure 6.5: Absolute final error $|h(T)|$ vs. initial height h_0 .

sine-Galerkin scheme described in Appendix C.2. The basis functions are

$$\phi_{k\ell}(x, y) = 2 \sin(k\pi x) \sin(\ell\pi y), \quad 1 \leq k, \ell \leq m, \quad (6.40)$$

where m denotes the number of retained sine modes in each spatial direction (i.e. $m_x = m_y = m$). The semi-discrete state therefore has dimension $d = m^2$. Projecting the heat equation onto this basis yields a finite-dimensional system of the form (6.29), where the diffusion operator is diagonal, $A = -\nu \text{diag}(\lambda_{k\ell})$, where $\nu > 0$ is the diffusion coefficient, and $\lambda_{k\ell} = (k\pi)^2 + (\ell\pi)^2$. The domain contains N_a square actuators. Actuator j has centre $w_j = (w_{x,j}, w_{y,j})$, and half-widths $\delta_x > 0$, $\delta_y > 0$. The corresponding influence matrix $B_j(w_j)$ is computed via the modal integrals (see Appendix C.2), and the full matrix is given by $B(w) = [B_1(w_1), \dots, B_{N_a}(w_{N_a})]$. Initial conditions for each trajectory in the training batch are generated as smooth random combinations of the sine basis.

To prevent actuators from approaching the boundary, we apply a smooth penalty to each centre $w_j = (w_{x,j}, w_{y,j})$. For a single coordinate $x \in [0, L]$ we use

$$p(x) = \frac{4}{\left(\tanh\left(\frac{x-\delta_x}{0.1}\right) + 1\right) \left(\tanh\left(\frac{L-\delta_x-x}{0.1}\right) + 1\right)}, \quad (6.41)$$

and define $p_j(w_j) = p(w_{x,j}) p(w_{y,j})$. This function satisfies $p_j(w_j) \geq 1$ and grows smoothly as w_j approaches the boundary, allowing gradient-based optimisation of w .

Table 6.2: Model and training parameters.

Parameter	Symbol / Value
Domain	$\Omega = [0, 1]^2$
Diffusion coefficient	$\nu = 0.01$
Time horizon	$T = 2$
Basis size	$m_x = m_y = 4$
Actuator width	$\delta_x = \delta_y = 0.08$
Terminal cost weight	$\alpha = 100$
Control cost weight	$\beta = 5 \times 10^{-3}$
Time step	$\Delta t = 0.01$
Training epochs	2×10^4
Hidden layers	$[64, 64]$ (Tanh)
Optimiser	Adam, learning rate 10^{-3}

In the discrete cost function $\hat{\mathcal{J}}$ given by equation (6.30), the running control penalty is

$$\sum_{j=1}^{N_a} r_j(w) u_j(t)^2, \quad r_j(w) = \beta p_j(w), \quad (6.42)$$

This penalisation is introduced to discourage boundary-near actuator locations and to ensure that the actuator support remains inside the admissible region during optimisation. The Riccati feedback from Appendix C.3 is used as the reference solution for fixed actuator locations. All model and training parameters are summarised in Table 6.2.

Single actuator. Figure 6.6 shows stable convergence of the loss over 2×10^4 epochs. Starting from $w^0 = (0.1, 0.1)$, the actuator migrates toward the centre of the domain as shown in Figure 6.7, where it can most effectively influence the dominant, slowly decaying temperature patterns. This central migration behaviour agrees with the analytical result of the scalar LQR model in Appendix C.1, where the optimal actuator position is shown to coincide with the domain midpoint. The learned control in Figure 6.8 applies strong early action to suppress the initial disturbance, followed by a gradual decay as the state stabilises. The neural policy slightly deviates from the Riccati reference at later times due to finite-horizon training and limited network capacity, but the main transient dynamics are accurately reproduced. Snapshots in figure 6.9 confirm this behaviour: the control quickly redistributes heat away from the corners, smoothing the field. Figure 6.10 shows

the time evolution of the state norm $\|\mathbf{h}(t)\|_2^2$, which illustrates the overall energy decay of the system. In the absence of control, diffusion causes a slow natural decay. A fixed actuator provides limited improvement due to its suboptimal placement. The learned actuator achieves the fastest decay by relocating toward dynamically dominant regions, allowing stronger suppression of the slowest-decaying temperature modes.

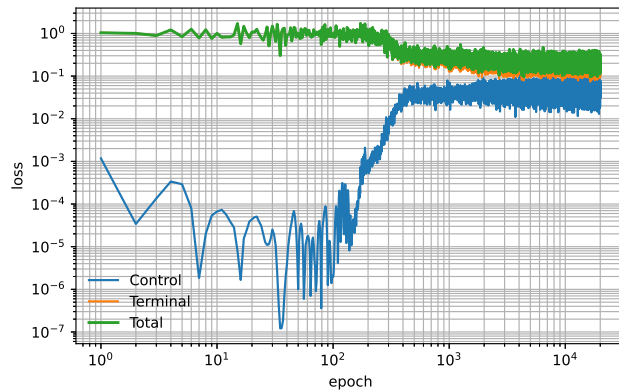


Figure 6.6: Training loss evolution for the single-actuator case.

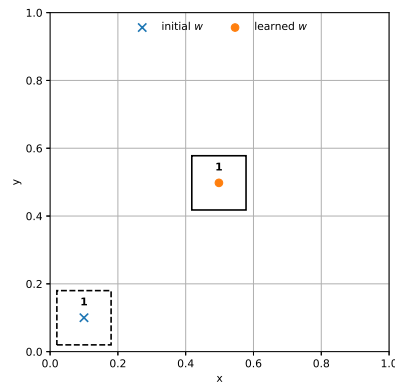


Figure 6.7: The dashed square corresponds to the initial actuator position, and the solid square corresponds to the learned position after training.

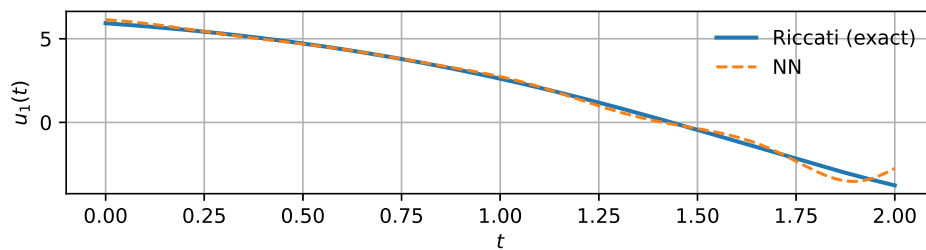


Figure 6.8: Comparison between neural and Riccati control signals for the single actuator.

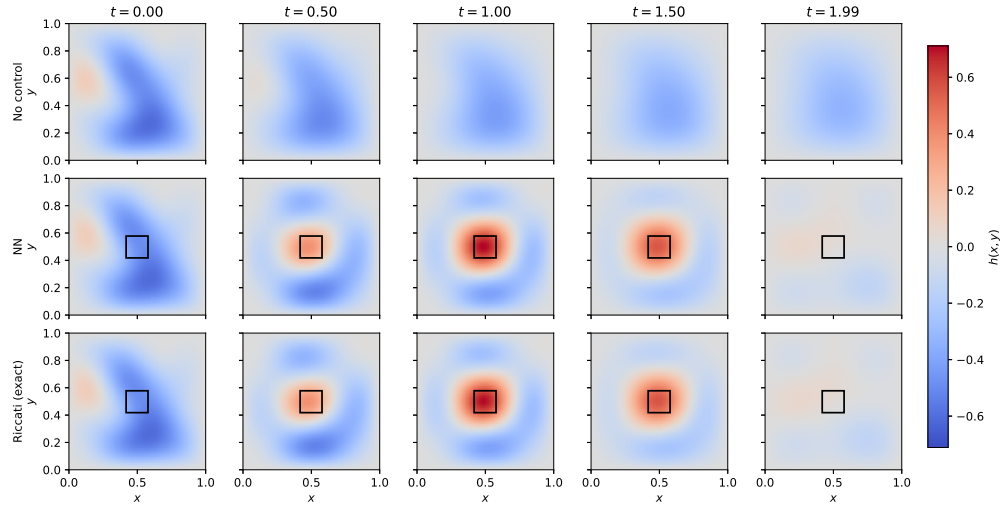


Figure 6.9: Snapshots of state evolution for the single actuator: (top) no control, (middle) neural feedback, (bottom) Riccati solution. In the controlled cases (middle and bottom rows), the black square marks the actuator support centred at the learned position w^* .

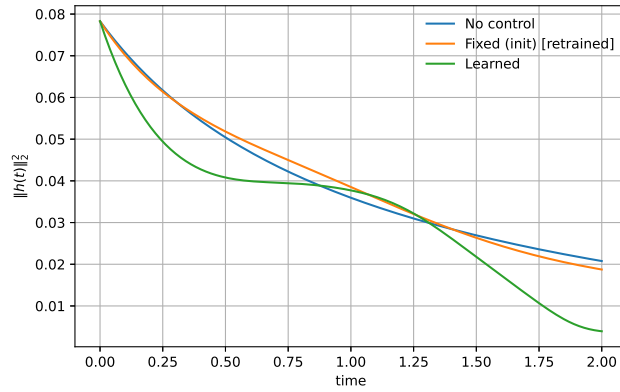


Figure 6.10: Temporal evolution of the energy norm $\|\mathbf{h}(t)\|_2^2$ for the single-actuator case. The blue curve shows the natural diffusion of the system. The orange curve corresponds to feedback control using a fixed, non-optimal actuator location. The green curve shows the performance achieved after jointly learning both the feedback law and the actuator position.

Two actuators. The loss components as shown in figure 6.11 converge smoothly, reflecting stable joint optimisation of the control and actuator parameters. With initial centres at $(0.1, 0.1)$ and $(0.9, 0.9)$, the actuators shift toward $(0.5, 0.35)$ and $(0.5, 0.65)$ as shown in Figure 6.12, achieving near-symmetric coverage of dominant spatial modes. Figure 6.13 shows more uniform damping and faster energy dissipation of the resulting trajectories than the single-actuator case. The control signals $u_1(t)$ and $u_2(t)$ seen in Figure 6.14 are initially symmetric, matching the rapid early suppression in the snapshots. Both follow the Riccati reference closely, with mi-

nor deviations at the final stage due to neural approximation. Figure 6.15 shows the energy decay that confirms the learned configuration achieves the most efficient stabilisation.

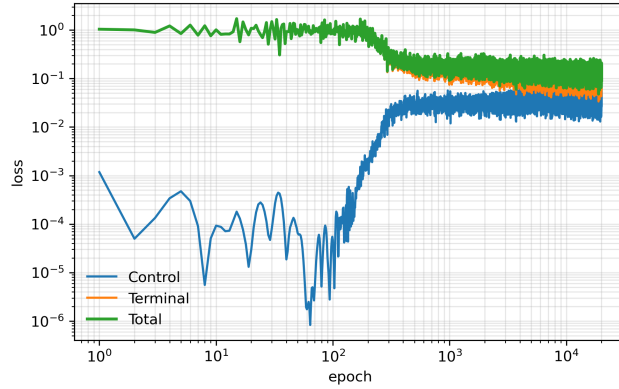


Figure 6.11: Training loss evolution for the two-actuator case.

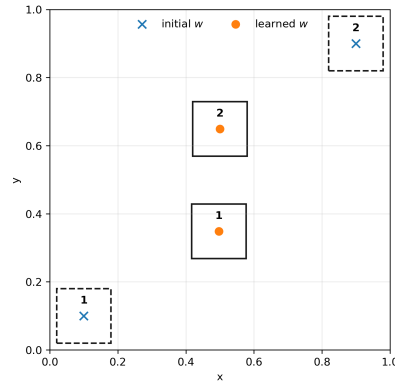


Figure 6.12: Initial (dashed) and learned (solid) actuator positions for the two-actuator case.

In both cases, the learned placements concentrate control authority in dynamically influential regions, producing trajectories and energy decay comparable to the optimal Riccati benchmark. The dual-actuator system achieves faster energy dissipation and smoother transient behaviour, as the additional actuator extends spatial coverage and enables simultaneous damping of multiple dominant modes.

6.7 Conclusion

This chapter introduced a direct learning framework for feedback-based actuator design. By representing the value function with a neural network and optimising

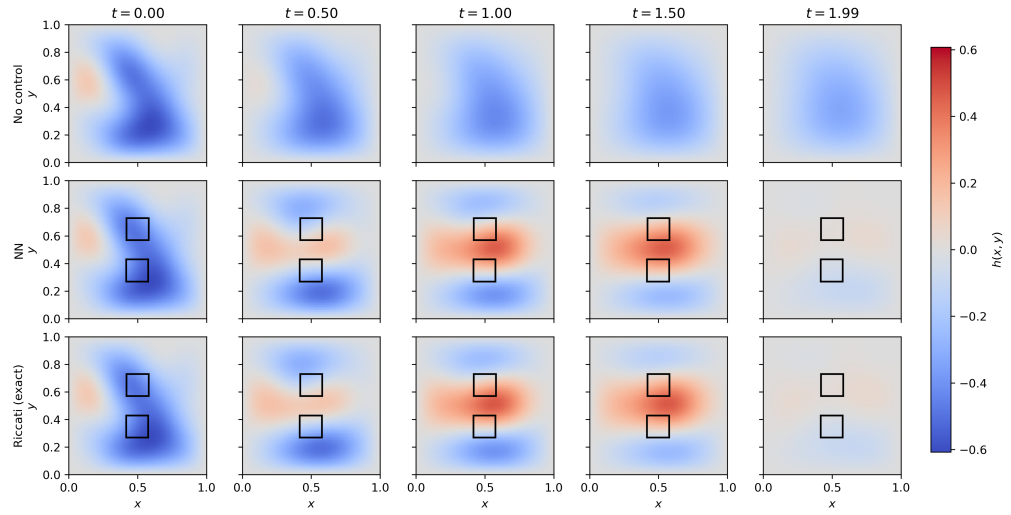


Figure 6.13: Snapshots of state evolution for the two-actuator case: (top) no control, (middle) neural feedback, (bottom) Riccati solution.

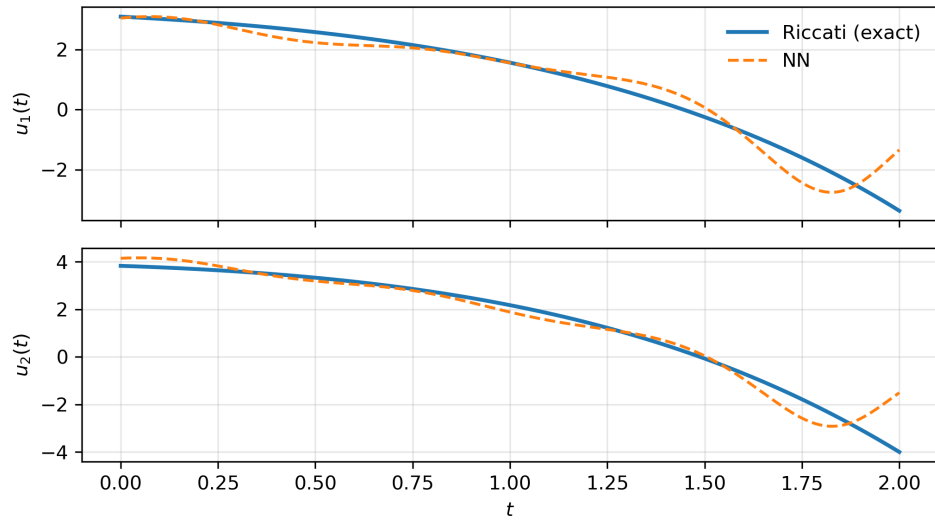


Figure 6.14: Comparison between neural and Riccati control signals for the two actuators.

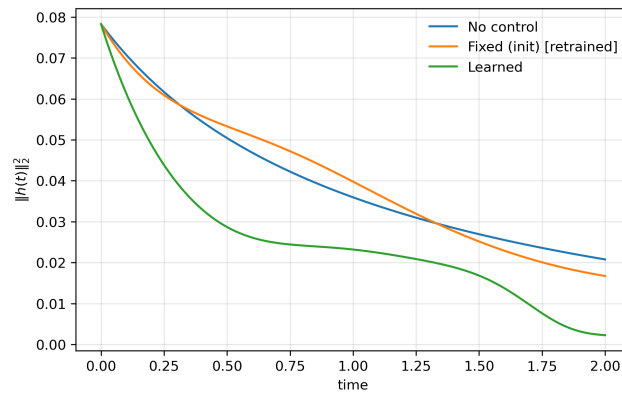


Figure 6.15: Temporal evolution of the energy norm $\|\mathbf{h}(t)\|_2^2$ for the two-actuator case.

the actuator positions jointly within the same cost functional, the approach eliminates the need for adjoint equations, nested iterations, or offline data generation. The resulting one-shot formulation learns both the feedback law and the actuator configuration directly from the governing dynamics and objective.

Numerical experiments on the heat equation confirmed that the learned feedback laws achieve performance comparable to that of Riccati-based solutions. The actuator locations adapt automatically to dynamically influential regions, enhancing stabilisation efficiency and spatial coverage.

We plan to extend this approach to nonlinear problems such as thin-film flow on flexible topographies, as discussed in the previous chapters. Future work will also focus on improving the optimisation procedure and exploring the stability, convergence, and robustness of the learned feedback laws.

Chapter 7

Conclusions

This chapter summarises the main aims and findings of the thesis, which addressed the control of thin-film flow over flexible substrates through PDE-constrained and learning-based formulations. The key conclusions from both the open-loop and feedback studies are highlighted, emphasising the integration of actuator design, optimisation, and neural-network representations. Finally, the chapter outlines several directions for future research, including model extensions, adaptive optimisation, and learning-based feedback for nonlinear systems.

7.1 Summary and Discussion

In this thesis, the control of systems governed by nonlinear partial differential equations was studied, with a particular focus on thin liquid films flowing over flexible surfaces. In such systems, the evolution of the free surface and the deformation of the substrate are coupled, and this interaction creates an opportunity to influence, stabilise, or reshape the film through suitable actuation. The work aimed to understand how the dynamics of nonlinear PDE systems can be influenced through suitable actuation, and how mathematical models and optimisation frameworks can be developed to support such control strategies.

We developed a coupled thin-film model that captures the main physical mechanisms governing both the free-surface evolution and the deformation of the substrate. The formulation includes surface-tension effects and disjoining pressure that allows rupture. Moreover, it uses a mixed representation that makes the system suitable for finite-element implementation. The model also satisfies a natural energy-dissipation law, which reflects the physical consistency of the dynamics.

The first part of the thesis dealt with the open-loop control of thin-film flow over flexible topographies. A mathematical framework for PDE-constrained control was developed and analysed, demonstrating how the deformation of a flexible wall can be influenced through distributed forces applied along the substrate. The optimality conditions were derived at the continuous level to provide analytical insight into the control structure. They were then implemented computationally through spatial and temporal discretisation. The forward and adjoint PDEs were discretised using finite elements and IMEX time stepping. The optimisation was then carried out with gradient descent equipped with a backtracking line search. Numerical experiments showed that wall flexibility can suppress dewetting instabilities, accelerate convergence to steady states, and guide the free-surface profile toward desired configurations. These findings illustrate how a deformable substrate can influence, stabilise, and guide thin-film evolution, a mechanism that is relevant to applications such as coating flows and heat transfer control [48, 152].

The second part of the thesis extended the open-loop framework toward actuator design and learned feedback control. The control problem was reformulated to

optimise not only the input signals but also the placement of localised actuators that most effectively influence the flow. To achieve this, a one-shot optimisation approach was introduced, embedding the governing equations directly within the objective so that actuator design and distributed control were treated simultaneously. This formulation can be implemented using both standard numerical discretisations and neural networks. Numerical experiments using physics-informed neural networks (PINNs) showed that the method learned actuator locations and generated controls that drove the free-surface profile toward the desired target state. This offers a practical and efficient way to unify actuator placement and distributed forcing within a single optimisation stage, avoiding repeated solves and nested iterations.

Building on this framework, a learned feedback-control approach was developed in which the value function of the Hamilton-Jacobi-Bellman equation was represented by a neural network. The trained network directly produced the feedback law, eliminating the need for adjoint equations, precomputed datasets, or nested optimisation. Numerical studies on a semi-discrete heat equation confirmed that the learned feedback achieved stabilisation comparable to classical Riccati-based solutions, with actuator positions adapting to the most influential regions. These results demonstrate how neural value-function approximations can support model-based feedback control in settings where analytical or adjoint-based methods become computationally demanding.

Overall, the thesis shows how to combine modelling, optimisation, and learning to advance the control of nonlinear PDE systems. These ingredients support a unified view of the open-loop, actuator design, and feedback problems studied here. The work progresses from distributed open-loop control to the joint design and optimisation of actuators. It then advances to the synthesis of feedback laws learned directly from the governing dynamics. This progression provides a coherent and model-based route to controlling complex PDE systems.

7.2 Future Work

Future work will extend the one-shot actuator-design and feedback-control formulations to higher-order thin-film models, such as the Benney [18] and weighted-residual

equations [84]. These models incorporate inertial and curvature effects that are not captured by the present lubrication framework. They also connect more directly with existing feedback-control studies for thin films, for example [35, 71, 149, 151, 161]. In practice, this extension will require adapting the mixed formulation and the numerical discretisation to the richer PDE structure. It will also require maintaining stable optimisation in the presence of stronger nonlinear effects.

Future work will also examine adaptive weighting and gradient-normalisation strategies for the PINN-based optimisation [38, 122]. These techniques aim to balance the different components of the loss, including the PDE residuals and boundary terms, and may improve both the efficiency and the robustness of training. A practical approach is to monitor the residual terms during optimisation and adjust the weights dynamically so that no single component dominates the learning process.

Another direction is to compare the one-shot optimisation approach with supervised learning strategies. In the present setting, the one-shot method treats the actuator locations, control inputs, and state variables as unknowns that are optimised simultaneously. By contrast, supervised learning relies on a large collection of precomputed solutions. Such a comparison would require a careful discretisation of the governing PDE and the generation of a sufficiently accurate dataset for training. This would provide a clearer assessment of the trade-offs between data-based and model-based approaches in design-control problems.

An important direction for future work is to extend the analysis of the control problems developed in this thesis. This includes deriving error estimates between the infinite-dimensional problems and their semi-discrete and fully discrete approximations. It also involves studying the convergence of the discretisation and optimisation schemes. It would also be of interest to establish further analytical results, including second-order optimality conditions and stability properties. In the feedback setting, additional analysis is needed to understand how errors in the approximation of the value function affect the resulting closed-loop dynamics.

Future work should also address the computational complexity of the proposed methods. In particular, it is important to study how the computational cost scales with the spatial discretisation, the number of actuator parameters, and the size of the neural network. This is especially relevant in the feedback framework, where the

dimension of the discretised PDE remains a major challenge. Reduced-order models and neural network approximations for high-dimensional value functions may provide a practical approach, but their effect on accuracy and computational cost requires careful assessment. Such analysis would give a clearer picture of the practical complexity of the framework and help determine to what extent the learning-based approach alleviates the curse of dimensionality in practice.

Bibliography

- [1] G. Albi, S. Bicego, and D. Kalise. Gradient-augmented supervised learning of optimal feedback laws using state-dependent riccati equations. *IEEE Control Systems Letters*, 6:836–841, 2021.
- [2] G. Albi, S. Bicego, and D. Kalise. Supervised learning for kinetic consensus control. *IFAC-PapersOnLine*, 55(30):103–108, 2022.
- [3] G. Albi, S. Bicego, M. Herty, Y. Huang, D. Kalise, and C. Segala. Data/moment-driven approaches for fast predictive control of collective dynamics. In *Model Predictive Control: Engineering Methods for Economists*, pages 29–54. Springer, 2025.
- [4] J. Alexander. *Systems of nonlinear PDEs arising from multi-dimensional and multi-fluid viscous shear flows*. PhD thesis, Imperial College London, 2021.
- [5] J. P. Alexander, T. L. Kirk, and D. T. Papageorgiou. Stability of falling liquid films on flexible substrates. *Journal of Fluid Mechanics*, 900:A40, 2020.
- [6] S. Alrashidy, A. Kalogirou, D. Kalise, and K. van der Zee. Optimal control of thin-film flow on a flexible topography. *arXiv preprint arXiv:2506.15340*, 2025.
- [7] T. G. Anderson, R. Cimpeanu, D. T. Papageorgiou, and P. G. Petropoulos. Electric field stabilization of viscous liquid layers coating the underside of a surface. *Physical Review Fluids*, 2(5):054001, 2017.
- [8] H. Antil and D. Leykekhman. A brief introduction to pde-constrained optimization. *Frontiers in PDE-constrained optimization*, 163:434, 2018.
- [9] K. Antonion, X. Wang, M. Raissi, and L. Joshie. Machine learning through physics-informed neural networks: Progress and challenges. *Academic Journal of Science and Technology*, 9(1):1–16, 2024.
- [10] K. Argyriadi, M. Vlachogiannis, and V. Bontozoglou. Experimental study of inclined film flow along periodic corrugations: The effect of wall steepness. *Physics of Fluids*, 18(1), 2006.
- [11] B. Azmi, D. Kalise, and K. Kunisch. Optimal feedback law recovery by gradient-augmented sparse polynomial regression. *Journal of Machine Learning Research (JMLR)*, 22, 2021.
- [12] J. F. Bard and J. E. Falk. An explicit solution to the multi-level programming problem. *Computers & Operations Research*, 9(1):77–100, 1982.
- [13] M. Bardi, I. C. Dolcetta, et al. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, volume 12. Springer, 1997.

- [14] J. Barry-Straume, A. Sarshar, A. A. Popov, and A. Sandu. Physics-informed neural networks for pde-constrained optimization and control. *arXiv preprint arXiv:2205.03377*, 2022. URL <https://arxiv.org/abs/2205.03377>.
- [15] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.
- [16] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18:1–43, 2018.
- [17] R. Bellman. Dynamic programming, princeton univ. *Press Princeton, New Jersey*, 39, 1957.
- [18] D. Benney. Long waves on liquid films. *Journal of mathematics and physics*, 45(1-4):150–155, 1966.
- [19] A. Bensoussan, G. Da Prato, M. C. Delfour, and S. K. Mitter. *Representation and control of infinite dimensional systems*, volume 1. Springer, 2007.
- [20] S. Berger and L.-D. Jou. Flows in stenotic vessels. *Annual review of fluid mechanics*, 32(1):347–382, 2000.
- [21] J. Berner, P. Grohs, and A. Jentzen. A survey on deep learning for partial differential equations. *Journal of Computational Physics*, 447:110686, 2021.
- [22] F. Bernis and A. Friedman. Higher order nonlinear degenerate parabolic equations. *Journal of differential equations*, 83(1):179–206, 1990.
- [23] A. L. Bertozzi and M. Pugh. The lubrication approximation for thin viscous films: the moving contact line with a porous media cut-off of van der waals interactions. *Nonlinearity*, 7(6):1535, 1994.
- [24] J. Binard, P. Noble, and P. Degond. Well-posedness and stability analysis of a landscape evolution model. *arXiv preprint arXiv:2211.09629*, 2022.
- [25] J. Blechschmidt and O. G. Ernst. Three ways to solve partial differential equations with neural networks—a review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021. doi: 10.1002/gamm.202100006.
- [26] M. Blyth and C. Pozrikidis. Effect of surfactant on the stability of film flow down an inclined plane. *Journal of Fluid Mechanics*, 521:241–250, 2004.
- [27] M. G. Blyth, T.-S. Lin, and D. Tseluiko. On the transition to dripping of an inverted liquid film. *Journal of Fluid Mechanics*, 958:A46, 2023.
- [28] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010: 19th International Conference on Computational Statistics Paris France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pages 177–186. Springer, 2010.
- [29] I. Brevis, I. Muga, and K. G. van der Zee. Neural control of discrete weak formulations: Galerkin, least squares & minimal-residual methods with quasi-optimal weights. *Computer Methods in Applied Mechanics and Engineering*, 402:115716, 2022.

- [30] I. Brevis, I. Muga, D. Pardo, O. Rodriguez, and K. G. Van Der Zee. Learning quantities of interest from parametric pdes: An efficient neural-weighted minimal residual approach. *Computers & Mathematics with Applications*, 164: 139–149, 2024.
- [31] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
- [32] C. Camporeale. An asymptotic approach to the crenulation instability. *Journal of Fluid Mechanics*, 826:636, 2017.
- [33] J. Castro, J. C. De los Reyes, and C.-B. Schönlieb. Bilevel learning for optimal sensor placement in variational data assimilation. *Inverse Problems*, 35(8): 085003, 2019. doi: 10.1088/1361-6420/ab208b.
- [34] M. Cerulli. *Bilevel optimization and applications*. PhD thesis, Institut Polytechnique de Paris, 2021.
- [35] R. Cimpanu, S. N. Gomes, and D. T. Papageorgiou. Active control of liquid film flows: beyond reduced-order models. *Nonlinear Dynamics*, 104(1):267–287, 2021.
- [36] S. S. Collis and M. Heinkenschloss. Analysis of the streamline upwind/petrov galerkin method applied to the solution of optimal control problems. *CAAM TR02-01*, 108, 2002.
- [37] R. V. Craster and O. K. Matar. Dynamics and stability of thin liquid films. *Reviews of modern physics*, 81(3):1131, 2009.
- [38] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what’s next. *Journal of Scientific Computing*, 92(3):88, 2022.
- [39] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274.
- [40] J. C. De los Reyes. *Numerical PDE-constrained optimization*. Springer, 2015.
- [41] M. M. Décré and J.-C. Baret. Gravity-driven flows of viscous liquids over two-dimensional topographies. *Journal of Fluid Mechanics*, 487:147–166, 2003.
- [42] S. Dempe. Bilevel optimization: theory, algorithms, applications and a bibliography. In *Bilevel optimization: advances and next challenges*, pages 581–672. Springer, 2020.
- [43] S. Dolgov, D. Kalise, and K. K. Kunisch. Tensor decomposition methods for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(3):A1625–A1650, 2021.
- [44] M. A. Durán-Olivencia, R. S. Gvalani, S. Kalliadasis, and G. A. Pavliotis. Instability, rupture and fluctuations in thin liquid films: Theory and computations. *Journal of statistical physics*, 174(3):579–604, 2019.
- [45] W. E and B. Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

- [46] L. C. Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.
- [47] P. D. Fowler, C. Ruscher, J. D. McGraw, J. A. Forrest, and K. Dalnoki-Veress. Controlling marangoni-induced instabilities in spin-cast polymer films: How to prepare uniform films. *The European Physical Journal E*, 39(9):1–8, 2016.
- [48] D. P. Frisk and E. J. Davis. The enhancement of heat transfer by waves in stratified gas-liquid flow. *International Journal of Heat and Mass Transfer*, 15(8):1537–1552, 1972.
- [49] C. J. García-Cervera, M. Kessler, and F. Periago. Control of partial differential equations via physics-informed neural networks. *Journal of Optimization Theory and Applications*, 196(2):391–414, 2023.
- [50] J. Garcke and A. Kröner. Suboptimal feedback control of pdes by solving hjb equations on adaptive sparse grids. *Journal of Scientific Computing*, 70(1): 1–28, 2017.
- [51] N. Garnier, R. O. Grigoriev, and M. F. Schatz. Optical manipulation of microscale fluid flow. *Physical Review Letters*, 91(5):054501, 2003.
- [52] P. Gaskell. Gravity-driven flow of continuous thin liquid films on non-porous substrates with topography. *Journal of Fluid Mechanics*, 509:253–280, 2004. doi: 10.1017/S0022112004009425.
- [53] P. Gaskell, P. Jimack, M. Sellier, H. Thompson, and M. Wilson. Gravity-driven flow of continuous thin liquid films on non-porous substrates with topography. *Journal of Fluid Mechanics*, 509:253–280, 2004.
- [54] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.
- [55] R. Glowinski and J.-L. Lions. Exact and approximate controllability for distributed parameter systems. *Acta numerica*, 4:159–328, 1995.
- [56] M. V. Gnann and M. Petrache. The navier-slip thin-film equation for 3d fluid films: existence and uniqueness. *Journal of Differential Equations*, 265(11): 5832–5958, 2018.
- [57] S. N. Gomes, D. T. Papageorgiou, and G. A. Pavliotis. 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, 2017.
- [58] H. Gomez and K. G. van der Zee. Computational phase-field modeling. In E. Stein, R. de Borst, and T. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics, Second Edition*. Wiley, 2017. Part 1 Fluids.
- [59] L. Gonon, L. Grigoryeva, and J.-P. Ortega. An overview on machine learning methods for partial differential equations: from physics informed neural networks to deep operator learning. *arXiv preprint arXiv:2408.13222*, 2024.

- [60] J. B. Grotberg and O. E. Jensen. Biofluid mechanics in flexible tubes. *Annu. Rev. Fluid Mech.*, 36(1):121–147, 2004.
- [61] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.
- [62] P. S. Hammond. Nonlinear adjustment of a thin annular film of viscous fluid surrounding a thread of another within a circular cylindrical pipe. *Journal of fluid Mechanics*, 137:363–384, 1983.
- [63] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.
- [64] Z. Hao, C. Ying, H. Su, J. Zhu, J. Song, and Z. Cheng. Bi-level physics-informed neural networks for pde constrained optimization using broyden’s hypergradients. *arXiv preprint arXiv:2209.07075*, 2022.
- [65] L. Herrmann and S. Kollmannsberger. Deep learning in computational mechanics: a review. *Computational Mechanics*, 74(2):281–331, 2024.
- [66] R. Herzog and K. Kunisch. Algorithms for pde-constrained optimization. *GAMM-Mitteilungen*, 33(2):163–176, 2010.
- [67] C. F. Higham and D. J. Higham. Deep learning: An introduction for applied mathematicians. *SIAM Review*, 61(4):860–891, 2019.
- [68] M. Hinze and F. Tröltzsch. Discrete concepts versus error analysis in pde-constrained optimization. *GAMM-Mitteilungen*, 33(2):148–162, 2010.
- [69] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE constraints*, volume 23. Springer Science & Business Media, 2008.
- [70] G. Holler, K. Kunisch, and R. C. Barnard. A bilevel approach for parameter learning in inverse problems. *Inverse Problems*, 34(11):115012, 2018.
- [71] O. A. Holroyd, R. Cimpeanu, and S. N. Gomes. Linear quadratic regulation control for falling liquid films. *arXiv preprint arXiv:2301.11379*, 2023.
- [72] O. A. Holroyd, R. Cimpeanu, and S. N. Gomes. Linear quadratic regulation control for falling liquid films. *SIAM Journal on Applied Mathematics*, 84(3): 940–960, 2024.
- [73] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.
- [74] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- [75] B. Hu. Effects of surface tension and viscoelastic behavior on the thin film coating flow of microbicide gels. 2016.
- [76] S. Huang, W. Feng, C. Tang, Z. He, C. Yu, and J. Lv. Partial differential equations meet deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

- [77] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop. Neural networks for control systems—a survey. *Automatica*, 28(6):1083–1112, 1992.
- [78] O. E. Jensen, G. Chini, and J. King. Thin-film flows near isolated humps and interior corners. *Journal of engineering mathematics*, 50:289–309, 2004.
- [79] N. Kabaliuk, M. Jermy, K. Morison, T. Stotesbury, M. Taylor, and E. Williams. Blood drop size in passive dripping from weapons. *Forensic science international*, 228(1-3):75–82, 2013.
- [80] D. Kalise and K. Kunisch. Polynomial approximation of high-dimensional hamilton–jacobi–bellman equations and applications to feedback control of semilinear parabolic pdes. *SIAM Journal on Scientific Computing*, 40(2):A629–A652, 2018.
- [81] D. Kalise, K. Kunisch, and K. Sturm. Optimal actuator design based on shape calculus. *Mathematical Models and Methods in Applied Sciences*, 28(13):2667–2717, 2018.
- [82] D. Kalise, S. Kundu, and K. Kunisch. Robust feedback control of nonlinear pdes by numerical approximation of high-dimensional hamilton–jacobi–isaacs equations. *SIAM Journal on Applied Dynamical Systems*, 19(2):1496–1524, 2020.
- [83] D. Kalise, E. Loayza-Romero, K. A. Morris, and Z. Zhong. Multi-level optimal control with neural surrogate models. *IFAC-PapersOnLine*, 58(17):292–297, 2024.
- [84] S. Kalliadasis, C. Ruyer-Quil, B. Scheid, and M. G. Velarde. *Falling liquid films*, volume 176. Springer Science & Business Media, 2011.
- [85] A. Kalogirou. Instability of two-layer film flows due to the interacting effects of surfactants, inertia, and gravity. *Physics of Fluids*, 30(3), 2018.
- [86] A. Kalogirou and M. G. Blyth. Nonlinear dynamics of two-layer channel flow with soluble surfactant below or above the critical micelle concentration. *Journal of Fluid Mechanics*, 900, 2020.
- [87] A. Kalogirou and M. G. Blyth. Nonlinear dynamics of unstably stratified two-layer shear flow in a horizontal channel. *Journal of Fluid Mechanics*, 955:A32, 2023.
- [88] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.
- [89] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014.
- [90] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.
- [91] D. E. Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.
- [92] S. F. Kistler and P. M. Schweizer. Coating science and technology: An overview. *Liquid Film Coating*, pages 3–15, 1997.

- [93] M. Klein and A. Prohl. Optimal control for the thin film equation: Convergence of a multi-parameter approach to track state constraints avoiding degeneracies. *Computational Methods in Applied Mathematics*, 16(4):685–702, 2016.
- [94] V. Kumaran and R. Muralikrishnan. Spontaneous growth of fluctuations in the viscous flow of a fluid past a soft interface. *Physical review letters*, 84(15):3310, 2000.
- [95] K. Kunisch and T. Pock. A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences*, 6(2):938–983, 2013. doi: 10.1137/120900046.
- [96] K. Kunisch and D. Walter. Optimal feedback control of dynamical systems via value-function approximation. *Comptes Rendus. Mécanique*, 351(S1):535–571, 2023.
- [97] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. doi: 10.1109/72.712178.
- [98] M. G. Larson and F. Bengzon. *The finite element method: theory, implementation, and applications*, volume 10. Springer Science & Business Media, 2013.
- [99] Y. Lee, H. Thompson, and P. Gaskell. Dynamics of thin film flow on flexible substrate. *Chemical Engineering and Processing: Process Intensification*, 50(5-6):525–530, 2011.
- [100] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- [101] R. Levy, D. B. Hill, M. G. Forest, and J. B. Grotberg. Pulmonary fluid flow challenges for experimental and mathematical modeling, 2014.
- [102] Z. Li, Z. Qiao, and T. Tang. *Numerical solution of differential equations: introduction to finite difference and finite element methods*. Cambridge University Press, 2017.
- [103] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- [104] D. Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press, 2011.
- [105] G. Liu, Z. Yuan, Z. Qiu, S. Feng, Y. Xie, D. Leng, and X. Tian. A brief review of bio-inspired surface technology and application toward underwater drag reduction. *Ocean Engineering*, 199:106962, 2020.
- [106] Y. Lou and P. D. Christofides. Optimal actuator/sensor placement for nonlinear control of the kuramoto-sivashinsky equation. *IEEE Transactions on Control Systems Technology*, 11(5):737–745, 2003.

- [107] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning non-linear operators via deepnet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.
- [108] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021. doi: 10.1137/19M1274067.
- [109] D. M. Luchtenburg. Data-driven science and engineering: machine learning, dynamical systems, and control (brunton, steven l. and kutz, j. nathan; 2020)[bookshelf]. *IEEE Control Systems Magazine*, 41(4):95–102, 2021.
- [110] D. Lunz. Minimizing deformation of a thin fluid film driven by fluxes of momentum and heat. *Physical Review E*, 103(3):033105, 2021.
- [111] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.
- [112] A. Manzoni, A. Quarteroni, and S. Salsa. *Optimal control of partial differential equations*. Springer, 2021.
- [113] O. Matar, R. Craster, and S. Kumar. Falling films on flexible inclines. *Physical Review E*, 76(5):056301, 2007.
- [114] O. K. Matar and S. Kumar. Rupture of a surfactant-covered thin liquid film on a flexible wall. *SIAM Journal on Applied Mathematics*, 64(6):2144–2166, 2004.
- [115] O. K. Matar and S. Kumar. Dynamics and stability of flow down a flexible incline. *Journal of Engineering Mathematics*, 57:145–158, 2007.
- [116] J.-A. Mejía-de Dios and E. Mezura-Montes. A physics-inspired algorithm for bilevel optimization. In *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–6. IEEE, 2018.
- [117] C. Miles, K. G. v. d. Zee, M. E. Hubbard, and R. MacKenzie. Thermomechanically-consistent phase-field modeling of thin film flows. *Numerical Methods for Flows: FEF 2017 Selected Contributions*, pages 121–129, 2020.
- [118] S. Mishra and R. Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes. *IMA Journal of Numerical Analysis*, 42(2):981–1022, 2022.
- [119] A. Miyara. Numerical analysis on flow dynamics and heat transfer of falling liquid films with interfacial waves. *Heat and Mass Transfer*, 35(4):298–306, 1999.
- [120] P. Möbius. Dautray, r.; lions, j.-l., mathematical analysis and numerical methods for science and technology. vol. 5: Evolution problems i. berlin etc., springer-verlag 1992. xiv, 709 pp., dm 198, oo. isbn 3-540-50205-x, 1994.
- [121] K. Morris. Noise reduction in ducts achievable by point control. 1998.

- [122] S. Mowlavi and S. Nabi. Optimal control of pdes using physics-informed neural networks. *arXiv preprint arXiv:2111.09880*, 2021. URL <https://arxiv.org/abs/2111.09880>. Mitsubishi Electric Research Laboratories Technical Report TR2022-143.
- [123] S. Mowlavi and S. Nabi. Optimal control of pdes using physics-informed neural networks (pinns). In *APS Division of Fluid Dynamics Meeting Abstracts*, pages H23–005, 2021.
- [124] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [125] T. Nakamura-Zimmerer, Q. Gong, and W. Kang. Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021.
- [126] A. Oron, S. H. Davis, and S. G. Bankoff. Long-scale evolution of thin liquid films. *Reviews of modern physics*, 69(3):931, 1997.
- [127] F. Peng, A. Ng, and Y.-R. Hu. Actuator placement optimization and adaptive vibration control of plate smart structures. *Journal of Intelligent Material Systems and Structures*, 16(3):263–271, 2005.
- [128] L. Pontryagin. *Mathematical Theory of Optimal Processes*. Classics of Soviet Mathematics. Taylor & Francis, 1987. ISBN 9782881240775. URL <https://books.google.co.uk/books?id=kqwzq0F4cBVAC>.
- [129] F. Pörner. *Regularization methods for ill-posed optimal control problems*. BoD–Books on Demand, 2018.
- [130] U. Prüfert. *Solving optimal PDE control problems: optimality conditions, algorithms and model reduction*. PhD thesis, Habilitationsschrift, Freiberg, Technische Universität Bergakademie Freiberg . . . , 2016.
- [131] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: an introduction*. Springer, 2015.
- [132] M. Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.
- [133] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2017. doi: 10.1016/j.jcp.2017.03.039.
- [134] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- [135] T. Rees, H. S. Dollar, and A. J. Wathen. Optimal solvers for pde-constrained optimization. *SIAM Journal on Scientific Computing*, 32(1):271–298, 2010.

- [136] O. Reynolds. Iv. on the theory of lubrication and its application to mr. beauchamp tower's experiments, including an experimental determination of the viscosity of olive oil. *Philosophical transactions of the Royal Society of London*, (177):157–234, 1886.
- [137] W. Rohlf, P. Pischke, and B. Scheid. Hydrodynamic waves in films flowing under an inclined plane. *Physical Review Fluids*, 2(4):044003, 2017.
- [138] C. Sánchez-Sánchez and D. Izzo. Real-time optimal control via deep neural networks: study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41(5):1122–1135, 2018.
- [139] M. Sellier. Substrate design or reconstruction from free surface data for thin film flows. *Physics of Fluids*, 20(6), 2008.
- [140] M. Sellier. Inverse problems in free surface flows: a review. *Acta Mechanica*, 227(3):913–935, 2016.
- [141] M. Sellier and S. Panda. Beating capillarity in thin film flows. *International journal for numerical methods in fluids*, 63(4):431–448, 2010.
- [142] K. Serifi, N. A. Malamataris, and V. Bontozoglou. Transient flow and heat transfer phenomena in inclined wavy films. *International journal of thermal sciences*, 43(8):761–767, 2004.
- [143] H. K. Singh, M. M. Islam, T. Ray, and M. Ryan. Nested evolutionary algorithms for computationally expensive bilevel optimization problems: Variants and their systematic analysis. *Swarm and Evolutionary Computation*, 48:329–344, 2019.
- [144] J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.
- [145] M. Sperl, L. Saluzzi, D. Kalise, and L. Grüne. Separable approximations of optimal value functions and their representation by neural networks. *arXiv preprint arXiv:2502.08559*, 2025.
- [146] V. A. Squire. Synergies between vlfs hydroelasticity and sea ice research. *International Journal of Offshore and Polar Engineering*, 18(04), 2008.
- [147] D. Tailor and D. Izzo. Learning the optimal state-feedback via supervised imitation learning. *Astrodynamics*, 3(4):361–374, 2019.
- [148] D. N. Tanyu, J. Ning, T. Freudenberger, N. Heilenkötter, A. Rademacher, U. Iben, and P. Maass. Deep learning methods for partial differential equations and related parameter identification problems. *Inverse Problems*, 39(10):103001, 2023.
- [149] A. B. Thompson, S. N. Gomes, G. A. Pavliotis, and D. T. Papageorgiou. Stabilising falling liquid film flows using feedback control. *Physics of Fluids*, 28(1):012107, 2016.
- [150] A. B. Thompson, S. N. Gomes, F. Denner, M. C. Dallaston, and S. Kalliadasis. Robust low-dimensional modelling of falling liquid films subject to variable wall heating. *Journal of Fluid Mechanics*, 877:844–881, 2019. doi: 10.1017/jfm.2019.580.

- [151] R. J. Tomlin and S. N. Gomes. Point-actuated feedback control of multidimensional interfaces. *IMA Journal of Applied Mathematics*, 84(6):1112–1142, 2019.
- [152] R. J. Tomlin, S. N. Gomes, G. A. Pavliotis, and D. T. Papageorgiou. Optimal control of thin liquid films and transverse mode effects. *SIAM Journal on Applied Dynamical Systems*, 18(1):117–149, 2019.
- [153] R. J. Tomlin, R. Cimpeanu, and D. T. Papageorgiou. Instability and dripping of electrified liquid films flowing down inverted substrates. *Physical Review Fluids*, 5(1):013703, 2020.
- [154] D. Tonon, M. S. Aronna, and D. Kalise. *Optimal control: Novel directions and applications*, volume 1. Springer, 2017.
- [155] F. Tröltzsch. *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, volume 112 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2010.
- [156] D. Tseluiko and D. T. Papageorgiou. Wave evolution on electrified falling films. *Journal of Fluid Mechanics*, 556:361–386, 2006.
- [157] D. Tseluiko, M. G. Blyth, and D. T. Papageorgiou. Stability of film flow over inclined topography based on a long-wave nonlinear model. *Journal of Fluid Mechanics*, 729:638–671, 2013. doi: 10.1017/jfm.2013.331.
- [158] K. G. van der Zee, J. Tinsley Oden, S. Prudhomme, and A. Hawkins-Daarud. Goal-oriented error estimation for cahn–hilliard models of binary phase transition. *Numerical Methods for Partial Differential Equations*, 27(1):160–196, 2011.
- [159] D. Wang, Y. Song, J. Tian, G. Haidak, et al. Research on the fluid film lubrication between the piston-cylinder interface. *AIP Advances*, 8(10), 2018.
- [160] S. Wang, S. Sankaran, and H. Wang. An expert’s guide to training physics-informed neural networks. *Preprint*, 2023.
- [161] A. W. Wray, R. Cimpeanu, and S. N. Gomes. Electrostatic control of the navier-stokes equations for thin films. *Physical Review Fluids*, 7(12):L122001, 2022.
- [162] X. Wu, K. G. van der Zee, G. Simsek, and E. Van Brummelen. A posteriori error estimation and adaptivity for nonlinear parabolic equations using imex-galerkin discretization of primal and dual equations. *SIAM Journal on Scientific Computing*, 40(5):A3371–A3399, 2018.
- [163] Y. Wu, B. Sicard, and S. A. Gadsden. Physics-informed machine learning: A comprehensive review on applications in anomaly detection and condition monitoring. *Expert Systems with Applications*, 255:124678, 2024.
- [164] R. Z. Zhang, X. Xie, and J. S. Lowengrub. Bilo: Bilevel local operator learning for pde inverse problems. *arXiv preprint arXiv:2404.17789*, 2024.
- [165] Y. Zhao and J. Han. Offline supervised learning vs online direct policy optimization: A comparative study and a unified training paradigm for neural network-based optimal feedback control. *Physica D: Nonlinear Phenomena*, 462:134130, 2024.

- [166] Y. B. Zhou. *Model reduction for nonlinear dynamical systems with parametric uncertainties*. PhD thesis, Massachusetts Institute of Technology, 2012.
- [167] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

Appendix A

Appendices of Chapter 3

A.1 Derivation of the thin film flow on a flexible topography

We consider a Newtonian liquid film of typical thickness h_0 flowing beneath a flexible substrate over a horizontal length scale $L = h_0/\varepsilon$, where $\varepsilon := \frac{h_0}{L} \ll 1$ is the small aspect ratio. The coordinate system is defined by (x, y) , with x being the horizontal coordinate and y being the vertical coordinate. Substitute the scalings from Table A.1 into the dimensional Navier-Stokes equations and the tangential and normal stress balance conditions. Expanding the equations in powers of ε and retaining leading-order terms yields:

$$p_x = u_{yy} \quad (\text{A.1})$$

$$p_y = \frac{B_o}{C_a} \quad (\text{A.2})$$

$$u_x + v_y = 0 \quad (\text{A.3})$$

where $p(x, y, t)$ and $\mathbf{u} = (u(x, y, t), v(x, y, t))$ are, respectively, the pressure and velocity of the fluid. All dimensionless quantities in this and subsequent equations (C_a , B_o , etc.) are defined in Table A.2. The leading-order tangential stress and the normal stress balance conditions at the free surface $y = \mathcal{H}(x, t) = h(x, t) + s(x, t)$ are

$$u_y = 0, \quad (\text{A.4a})$$

$$p = -\Pi(h) - \frac{1}{C_a} \mathcal{H}_{xx}, \quad (\text{A.4b})$$

where the dimensionless disjoining pressure is written in terms of the potential ϕ as

$$\Pi(h) = -\phi'(h) = Ah^{-3}. \quad (\text{A.5})$$

The conditions on the wall $y = s(x, t)$ are

$$u = 0, \quad v = s_t, \quad (\text{A.6a})$$

$$s_t - c^2 s_{xx} = f(x, t) - \gamma p(x, s, t), \quad (\text{A.6b})$$

where the actuation pressure acting on the back of the wall appears in the nondimensional wall equation as the forcing term $f(x, t)$, obtained from $p_{\text{ext}} = f$. The

Table A.1: Variable scaling (asterisks denote dimensional quantities). Here, $\bar{\mu}$ is the viscosity and U_0 is the characteristic velocity used in the scaling.

Dimensional variable	Nondimensional scaling
x^*	$x^* = (h_0/\varepsilon) x$
y^*	$y^* = h_0 y$
u^*	$u^* = U_0 u$
v^*	$v^* = \varepsilon U_0 v$
t^*	$t^* = h_0/(\varepsilon U_0) t$
p^*	$p^* = (\bar{\mu} U_0/(\varepsilon h_0)) p$
Π^*	$\Pi^* = (\bar{\mu} U_0/(\varepsilon h_0)) \Pi$
p_{ext}^*	$p_{ext}^* = \frac{p_{ext}}{\bar{\mu} U_0/(\varepsilon h_0)}$

kinematic condition

$$\mathcal{H}_t + u\mathcal{H}_x = v, \quad (\text{A.7})$$

can be written in the form

$$h_t + q_x = 0, \quad (\text{A.8})$$

where q is the flux defined by

$$q(x, t) = \int_s^{s+h} u \, dy. \quad (\text{A.9})$$

To find u , we integrate (A.2) and use the condition (A.4b), which gives

$$p(x, y, t) = -\Pi(h) - \frac{1}{C_a} \mathcal{H}_{xx} + \frac{B_o}{C_a} (y - \mathcal{H}). \quad (\text{A.10})$$

Differentiating in x yields

$$p_x = -\left[\Pi(h) + \frac{1}{C_a} \mathcal{H}_{xx} + \frac{B_o}{C_a} \mathcal{H}\right]_x, \quad (\text{A.11})$$

which shows that p_x is independent of y and hence we can simply integrate equation (A.1) and use the boundary condition (A.4a). Then we integrate the result once again and use the no-slip boundary condition (A.6a) to get the final result

$$u(x, y, t) = p_x (y - s) \left(\frac{1}{2} (y - s) - h \right). \quad (\text{A.12})$$

Substituting the velocity into equation (A.9) gives the flux

$$q(x, t) = \int_s^{s+h} u \, dy = -\frac{h^3}{3} p_x. \quad (\text{A.13})$$

Using equation (A.11) and equation (A.8), we obtain

$$h_t + \frac{1}{3} \left(h^3 \left[\Pi(h) + \frac{1}{C_a} \mathcal{H}_{xx} + \frac{B_o}{C_a} \mathcal{H} \right]_x \right)_x = 0, \quad (\text{A.14})$$

which is the same as (3.2a), and (3.2b) is defined as the term in the square brackets.

To complete the coupling with the deformable substrate, we evaluate the pressure

Table A.2: Nondimensional parameters. Here, ρ is the density, g is the gravity, σ is the surface tension, A^* is the dimensional Hamaker constant, γ_w is the wall damping, and T_w is the wall tension.

Parameter	Definition
Capillary	$C_a = \frac{\bar{\mu}U_0}{\sigma\varepsilon^3}$
Bond	$B_o = \frac{\rho gh_0^2}{\sigma\varepsilon^2}$
Hamaker	$A = \frac{A^*\varepsilon h_0^2}{\bar{\mu}U_0}$
Wall tension	$c^2 = \frac{\sigma + T_w}{\gamma_w U_0 L}$
Film damping	$\gamma = \frac{\bar{\mu}}{\varepsilon^2 \gamma_w^* h_0}$

field (A.10) at the wall position $y = s$, which gives

$$p(x, s, t) = -\Pi(h) - \frac{1}{C_a} \mathcal{H}_{xx} - \frac{B_o}{C_a} h. \quad (\text{A.15})$$

Substituting this expression into the wall equation (A.6b) yields the corresponding evolution equation for the substrate deflection. In this step we neglect the term $\gamma\Pi(h)$, which is asymptotically small compared with the hydrodynamic loading. The resulting relation coincides with the flexible-wall model presented in (3.2c) of (3.2).

A.2 Free energy dissipation

Here we derive the identity (3.7). We recall the energy

$$\mathcal{E}(h, s) = \int_{\Omega} \left[\phi(h) - \frac{B_o}{2C_a} h^2 + \frac{1}{2C_a} h_x^2 + \frac{c^2}{2\gamma} s_x^2 + \frac{1}{C_a} (h_x s_x - B_o h s) \right] dx. \quad (\text{A.16})$$

The molecular energy term $\phi(h)$ represents forces between molecules in the liquid film and the substrate, such as van der Waals forces. The film naturally moves toward thicknesses where $\phi(h)$ is smallest. In the rupture examples we study, we use $\phi'(h) = Ah^{-3}$ with $A > 0$, which represents attractive van der Waals forces. For this choice, $\phi(h) \rightarrow -\infty$ as $h \rightarrow 0$, meaning the energy becomes lower when the film gets thinner. This makes thinning energetically favourable, so rupture can occur. The hydrostatic term $-\frac{B_o}{2C_a} h^2$ represents the effect of gravity. Since the film hangs beneath the substrate and this term is negative, the energy decreases when the film becomes thicker. Therefore, when $B_o > 0$, gravity encourages the film to form thicker bulges rather than staying uniform. The capillary term $\frac{1}{2C_a} h_x^2$ penalises sharp slopes in the film surface, favouring smoother and flatter shapes. Finally, the fluid-wall interaction energy $\frac{1}{C_a} (h_x s_x - B_o h s)$ couples the film and substrate together. The cross-slope term $h_x s_x$ reduces the energy when the film and wall tilt in the same direction, encouraging them to deform together. The hydrostatic coupling $-B_o h s$ is reduced when thicker parts of the film sit above downward bends in the wall, representing the weight of the film pushing down on the substrate.

The goal is to derive the rate of change of this energy in time

$$\frac{d\mathcal{E}}{dt} = \frac{d}{dt} \int_{\Omega} \left[\phi(h) + \frac{1}{2C_a} h_x^2 + \frac{c^2}{2\gamma} s_x^2 + \frac{1}{C_a} h_x s_x - \frac{B_o}{2C_a} h^2 - \frac{B_o}{C_a} h s \right] dx. \quad (\text{A.17})$$

We apply the chain rule and integrate by parts on the terms involving spatial derivatives, then apply the boundary conditions (3.5) to remove the boundary terms, yielding

$$\frac{d\mathcal{E}}{dt} = \int_{\Omega} \left[\left(\phi'(h) - \frac{1}{C_a} h_{xx} - \frac{B_o}{C_a} h \right) h_t - \frac{c^2}{\gamma} s_{xx} s_t + \frac{1}{C_a} (h_x s_x)_t - \frac{B_o}{C_a} (hs)_t \right] dx. \quad (\text{A.18})$$

We use equations (3.2) to simplify the above expression, resulting in the dissipation law

$$\frac{d}{dt} \mathcal{E}(h, s) = - \left(\frac{1}{3} \int_{\Omega} h^3 \mu_x^2 dx + \frac{1}{\gamma} \int_{\Omega} s_t^2 dx \right) + \frac{1}{\gamma} \int_{\Omega} f s_t dx. \quad (\text{A.19})$$

This equation represents the rate of change of the free energy. The first term on the right-hand side corresponds to the dissipation in the fluid, the second term represents dissipation in the substrate, and the last term accounts for the work done by the external force on the substrate. Thus, without external forces (i.e., $f = 0$), the total free energy is non-increasing in time.

A.3 Boundedness of $(\phi'(h), w)$

We prove that for any $h, w \in L^2(\Omega)$,

$$|(\phi'(h), w)| \leq C_L (\|h\|_{L^2(\Omega)} + C) \|w\|_{L^2(\Omega)}, \quad (\text{A.20})$$

where C_L is the Lipschitz constant of ϕ' and C is a positive constant. Let $h_{\star} \in \mathbb{R}$ be any fixed reference value. We decompose

$$(\phi'(h), w) = (\phi'(h) - \phi'(h_{\star}), w) + (\phi'(h_{\star}), w). \quad (\text{A.21})$$

Using Lipschitz continuity of ϕ' ,

$$|\phi'(h(x)) - \phi'(h_{\star})| \leq C_L |h(x) - h_{\star}|, \quad (\text{A.22})$$

and applying Hölder's inequality gives

$$|(\phi'(h) - \phi'(h_{\star}), w)| \leq C_L \|h - h_{\star}\|_{L^2(\Omega)} \|w\|_{L^2(\Omega)}. \quad (\text{A.23})$$

Since $h_{\star}$ is a constant,

$$|(\phi'(h_{\star}), w)| = |\phi'(h_{\star})| \|w\|_{L^1(\Omega)} \leq |\phi'(h_{\star})| \|1\|_{L^2(\Omega)} \|w\|_{L^2(\Omega)}. \quad (\text{A.24})$$

Combining both bounds,

$$|(\phi'(h), w)| \leq (C_L \|h - h_{\star}\|_{L^2(\Omega)} + |\phi'(h_{\star})| \|1\|_{L^2(\Omega)}) \|w\|_{L^2(\Omega)} \quad (\text{A.25})$$

$$\leq (C_L (\|h\|_{L^2} + |h_{\star}|) + C^* \|1\|_{L^2(\Omega)}) \|w\|_{L^2(\Omega)}. \quad (\text{A.26})$$

Thus the inequality (A.20) holds with

$$C = C_L |h_{\star}| + C^* \|1\|_{L^2(\Omega)}, \quad (\text{A.27})$$

where h_* may be chosen arbitrarily.

A.4 IMEX-FEM discretisation

We discretise the spatial domain $\Omega \subset \mathbb{R}$ into $n - 1$ elements with n nodes. The variational space $V = H^1(\Omega)$ is approximated by the finite-dimensional subspace $V_h \subset V$ spanned by the standard piecewise-linear hat functions $\{\hat{\phi}_i\}_{i=1}^n$. The solutions can therefore be characterised as linear combinations of $\hat{\phi}_i$ with real coefficients, such that $\mathbf{F}_h(x, t) = \sum_{i=1}^n F_i(t) \hat{\phi}_i(x)$. Here, $\mathbf{F}$ represents a vector that holds the unknowns at each node and is one of the state, adjoint or control variables, i.e. $\mathbf{F} = \{\mathbf{h}, \boldsymbol{\mu}, \mathbf{s}, \mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{f}\}$, with corresponding coefficients $F_i = \{h_i, \mu_i, s_i, p_i, q_i, r_i, f_i\}$.

We define the real $n \times n$ matrices

$$\begin{aligned} \mathbf{M}_{ij} &= \int_{\Omega} \hat{\phi}_i \hat{\phi}_j \, dx, & \mathbf{K}_{ij} &= \int_{\Omega} \hat{\phi}'_i \hat{\phi}'_j \, dx, \\ \widehat{\mathbf{M}}_{ij}^k &= \int_{\Omega} m(h_h^k) \hat{\phi}_i \hat{\phi}_j \, dx, & \widehat{\mathbf{K}}_{ij}^k &= \int_{\Omega} m(h_h^k) \hat{\phi}'_i \hat{\phi}'_j \, dx, \\ \widehat{\mathbf{C}}_{ij}^k &= \int_{\Omega} m(h_h^k) \hat{\phi}'_i \hat{\phi}_j \, dx. \end{aligned}$$

The IMEX time discretisation uses uniform time steps $t^k = k\Delta t$. All linear terms are evaluated implicitly at t^{k+1} , whilst nonlinear terms are evaluated explicitly at t^k . In particular, $\phi'_+(h_h^{k+1})$ (implicit), and $\phi'_-(h_h^k)$ (explicit), and likewise for $\phi''_{\pm}$. Here, $m(h)$ can adopt nonlinear forms such as h^2 or h^3 . We write the semi-discrete optimality system in the following form:

$$\text{(State)} \quad \mathbf{M} \frac{\mathbf{h}^{k+1} - \mathbf{h}^k}{\Delta t} + \frac{1}{3} \widehat{\mathbf{K}}^k ((\mathbf{h}^k)^3) \boldsymbol{\mu}^{k+1} = 0, \quad (\text{C.1})$$

$$\mathbf{M} \boldsymbol{\mu}^{k+1} - \frac{A}{\varepsilon^4} \widehat{\mathbf{M}}^{k+1} (\phi'_+(\mathbf{h}^{k+1})) \mathbf{h}^{k+1} - \frac{1}{C_a} (\mathbf{K} + B_o \mathbf{M}) \boldsymbol{\chi}^{k+1} \quad (\text{C.2})$$

$$= \frac{A}{\varepsilon^4} \widehat{\mathbf{M}}^k (\phi'_-(\mathbf{h}^k)) \mathbf{h}^k, \quad (\text{C.3})$$

$$\mathbf{M} \frac{\mathbf{s}^{k+1} - \mathbf{s}^k}{\Delta t} + c^2 \mathbf{K} \mathbf{s}^{k+1} + \frac{\gamma}{C_a} (\mathbf{K} - B_o \mathbf{M}) \mathbf{h}^{k+1} = \mathbf{M} \mathbf{f}^{k+1}, \quad (\text{C.4})$$

$$\text{(Adjoint)} \quad -\mathbf{M} \frac{\mathbf{p}^{k+1} - \mathbf{p}^k}{\Delta t} + \widehat{\mathbf{C}}^k \mathbf{p}^k + \left(\frac{B_o}{C_a} \mathbf{M} - \frac{1}{C_a} \mathbf{K} - \frac{A}{\varepsilon^4} \widehat{\mathbf{M}}^{k+1} (\phi''_+(\mathbf{h}^{k+1})) \right) \mathbf{q}^{k+1} \quad (\text{C.5})$$

$$+ \frac{\gamma}{C_a} (B_o \mathbf{M} - \mathbf{K}) \mathbf{r}^{k+1} = \frac{A}{\varepsilon^4} \widehat{\mathbf{M}}^k (\phi''_-(\mathbf{h}^k)) \mathbf{q}^k, \quad (\text{C.6})$$

$$\frac{1}{3} \widehat{\mathbf{K}}^k ((\mathbf{h}^k)^3) \mathbf{p}^{k+1} + \mathbf{M} \mathbf{q}^{k+1} = 0, \quad (\text{C.7})$$

$$-\mathbf{M} \frac{\mathbf{r}^{k+1} - \mathbf{r}^k}{\Delta t} + c^2 \mathbf{K} \mathbf{r}^{k+1} - \frac{1}{C_a} (B_o \mathbf{M} - \mathbf{K}) \mathbf{q}^{k+1} = 0, \quad (\text{C.8})$$

$$\text{(Optimality)} \quad \alpha \mathbf{M} \mathbf{f}^{k+1} - \mathbf{M} \mathbf{r}^{k+1} = 0. \quad (\text{C.9})$$

The forward problem is initialised with $\mathbf{h}^0$ and $\mathbf{s}^0$ from (3.28). The adjoint variables satisfy the terminal conditions obtained from the discrete optimality system, namely

$$\mathbf{p}^N = \mathbf{r}^N = \mathbf{h}^N + \beta \mathbf{s}^N - \bar{\mathbf{h}}, \quad (\text{A.29})$$

with $\mathbf{q}^N$ given by (C.7) at $k = N$.

Appendix B

Appendices of Chapter 5

B.1 Supervised learning approach

To illustrate the supervised learning strategy discussed in Section 5.3, we consider the same physical setting and the governing equations (5.1)-(5.3), with model parameters $L = 10$, $C_a = 1$, $B_o = 0.1$, $c = 0.5$, and $\gamma = 0.1$. In this approach, we use offline data generated from the lower-level optimisation problems to approximate the cost functional $\mathcal{D}(\omega) = \mathcal{J}(h^*, s^*, \{f_i^*\}, \omega)$, where $(h^*, s^*, \{f_i^*\})$ are the optimal state and control trajectories corresponding to each actuator configuration $\omega = (w_1, w_2)$. Once the surrogate map $\mathcal{D}(\omega)$ has been trained, the upper-level design problem reduces to a direct minimisation of this learned cost surface with respect to ω .

The actuator profile is chosen as a rectangular function of width $2\delta = 1$,

$$\chi_i(x; w_i) = \begin{cases} 1, & |x - w_i| \leq \delta \\ 0, & \text{otherwise.} \end{cases} \quad (\text{B.1})$$

The film height and substrate displacement are initialised by

$$h_0(x) = 1 + 0.5 \cos\left(\frac{\pi x}{L}\right), \quad s_0(x) = 0, \quad (\text{B.2})$$

and the target state is given by

$$\bar{h}(x) = 1 + 0.5 \cos\left(\frac{3\pi x}{L}\right). \quad (\text{B.3})$$

To construct the dataset, actuator centres (w_1, w_2) are uniformly sampled over $[0, L]$ with a non-overlapping condition $|w_1 - w_2| > 2\delta$. For each configuration, the lower-level optimal control problem (5.5) is solved for $\{f_i^*(t)\}$ using the adjoint system (5.7)-(5.9), and the resulting cost $\mathcal{J}(h^*, s^*, \{f_i^*\}, \omega)$ is recorded. A total of approximately 300 valid actuator configurations are obtained, forming the dataset $\{(w_1^{(j)}, w_2^{(j)}), \mathcal{J}^{(j)}\}_{j=1}^{300}$.

Figure B.1 shows the computed cost landscape. The cost varies smoothly with respect to the actuator positions, and regions of low cost are clearly identifiable. These values serve as the reference data for the supervised regression stage.

A fully-connected neural network with two hidden layers of 64 neurons each and **tanh** activations is trained to approximate $\mathcal{D}(\omega)$. The mean-squared error between predicted and computed costs is minimised using the Adam optimiser with learning rate 10^{-3} . Eighty percent of the samples are used for training and the remaining for validation.

Figure B.2 presents the training and validation losses. The network achieves a

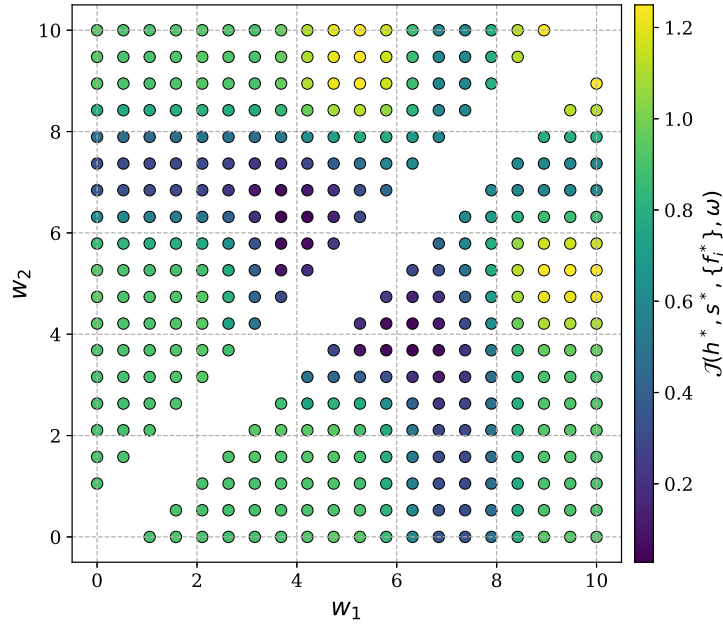


Figure B.1: Distribution of the computed costs $\mathcal{J}(h^*, s^*, \{f_i^*\}, \omega)$ over the sampled actuator locations (w_1, w_2) . The low-cost region corresponds to actuator configurations producing free surfaces close to the target.

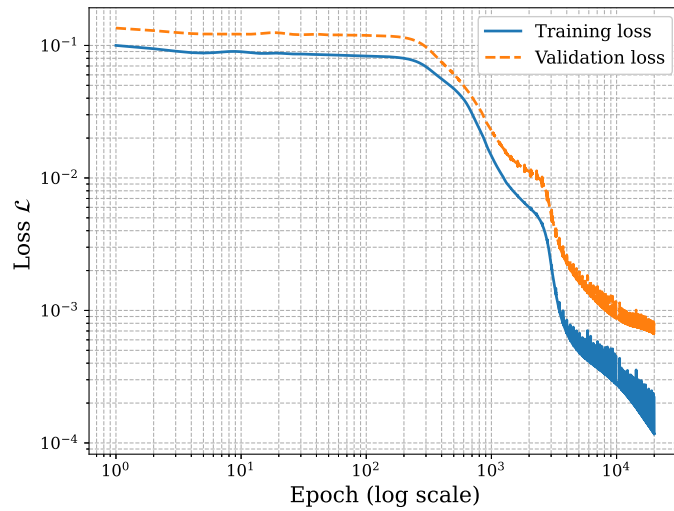


Figure B.2: Evolution of the training and validation losses during supervised learning. Both losses decrease by roughly two orders of magnitude, although the validation curve remains consistently above the training loss, suggesting moderate generalisation error.

satisfactory fit within 10^4 epochs, although the convergence rate slows down at later stages. The training was performed on a single GPU and required several hours, highlighting the considerable offline cost of data generation and training.

The quality of the trained surrogate is assessed in Figure B.3, which compares the true and predicted cost surfaces. The two landscapes share a similar topology and location of minima, although small discrepancies are visible in the outer regions, indicating limited extrapolation capability beyond the sampled domain.

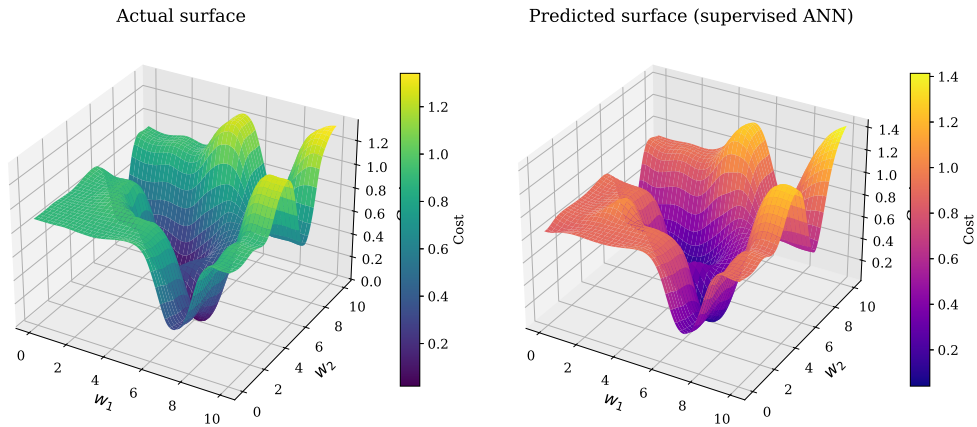


Figure B.3: Comparison of the true cost surface (left) and the ANN-predicted surface (right). The predicted minimum closely matches the true optimum, though slight deviations appear in less-sampled regions.

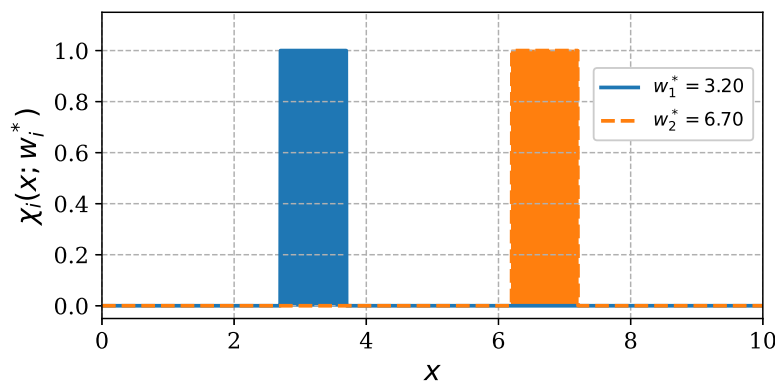


Figure B.4: Optimal actuator supports obtained from the supervised learning model. Each actuator acts over a compact region of width $2\delta = 1$ centred at (w_1^*, w_2^*) .

Minimising the surrogate surface in (5.13) yields

$$w^* \approx (w_1^*, w_2^*) = (6.7, 3.2), \quad (\text{B.4})$$

obtained directly from the optimiser applied to the learned cost map $\mathcal{D}_\theta(w)$.

This example demonstrates how supervised learning can approximate the upper-level objective $\mathcal{D}(\omega)$ using data generated from adjoint-based optimisation. The method eliminates the need for online PDE solves during optimisation, but shifts the computational burden to the offline stage, where a large number of high-fidelity simulations must be performed. Even with several hundred samples, the learned model exhibits limited accuracy outside the training range and remains expensive to construct. Nevertheless, the results confirm that a moderately trained surrogate can capture the general structure of the cost landscape and recover physically reasonable actuator locations.

B.2 Model Order Reduction (MOR)

In this appendix, we illustrate the concept of MOR by attempting to reduce the forward system (3.2). It is also worth mentioning that we follow [109, 166] to derive the model reduction and extend it to our nonlinear problem.

B.2.1 First Steps into MOR

The spatial discretisation of (3.2) leads to the following system of N nonlinear ODEs (for simplicity, we have assumed $\phi'(h) = \gamma = 0, B_o = 1$):

$$M\dot{\mathbf{h}} = -\frac{1}{3}\hat{K}\boldsymbol{\mu}, \quad (\text{B.5a})$$

$$M\boldsymbol{\mu} = M\boldsymbol{\mathcal{H}} + \frac{1}{C_a}K\boldsymbol{\mathcal{H}}, \quad (\text{B.5b})$$

$$M\dot{\mathbf{s}} + cK\mathbf{s} = M\mathbf{f}, \quad (\text{B.5c})$$

$$\mathbf{h}(0) = \mathbf{h}_0, \mathbf{s}(0) = \mathbf{s}_0 \quad (\text{B.5d})$$

where $\mathbf{h}, \boldsymbol{\mu}, \mathbf{s}, \mathbf{f}$ and $\boldsymbol{\mathcal{H}} := \mathbf{h} + \mathbf{s}$ are $\in \mathbb{R}^N$, and M, K and $\hat{K}(\mathbf{h})$ are $\in \mathbb{R}^{N \times N}$.

The infinite-dimensional problem (3.2) is thus replaced by the finite-dimensional system (B.5). In practice, however, the dimension N may still be large. Model order reduction seeks a lower-dimensional approximation of dimension N_r , where $N_r \ll N$, whilst preserving the main features of the full system. Throughout this discussion, N_r denotes the dimension of the reduced-order model.

Suppose a matrix $\Phi \in \mathbb{R}^{N \times N_r}$ with its columns being orthonormal vectors. This implies $\Phi^T \Phi = I \in \mathbb{R}^{N_r \times N_r}$. In the case of Galerkin projection, the columns $\{\phi\}_{i=1}^{N_r}$ of Φ establish a basis for an N_r -dimensional subspace, denoted as $V^{N_r} = \text{span}\{\phi_1, \dots, \phi_{N_r}\}$ which is a subset of $\mathbb{R}^N$ [166]. To initiate the derivation of the projection-based reduced order model (ROM), we express states like $\mathbf{h}, \boldsymbol{\mu}, \mathbf{s}$, and $\mathbf{f}$ as linear combinations of N_r basis vectors, bearing in mind that $N_r \ll N$:

$$\mathbf{h} \approx \sum_{i=0}^{N_r} \tilde{h}_i \phi_i, \quad \boldsymbol{\mu} \approx \sum_{i=0}^{N_r} \tilde{\mu}_i \phi_i, \quad (\text{B.6a})$$

$$\mathbf{s} \approx \sum_{i=0}^{N_r} \tilde{s}_i \phi_i, \quad \mathbf{f} \approx \sum_{i=0}^{N_r} \tilde{f}_i \phi_i. \quad (\text{B.6b})$$

In matrix form:

$$\mathbf{h} \approx \Phi \mathbf{h}_r, \quad \boldsymbol{\mu} \approx \Phi \boldsymbol{\mu}_r, \quad (\text{B.7a})$$

$$\mathbf{s} \approx \Phi \mathbf{s}_r, \quad \mathbf{f} \approx \Phi \mathbf{f}_r, \quad (\text{B.7b})$$

where $\Phi = [\phi_1, \phi_2, \dots, \phi_{N_r}] \in \mathbb{R}^{N \times N_r}$, and

$$\begin{aligned} \mathbf{h}_r &= [\tilde{h}_1, \tilde{h}_2, \dots, \tilde{h}_{N_r}] \in \mathbb{R}^{N_r \times 1}, & \boldsymbol{\mu}_r &= [\tilde{\mu}_1, \tilde{\mu}_2, \dots, \tilde{\mu}_{N_r}] \in \mathbb{R}^{N_r \times 1} \\ \mathbf{s}_r &= [\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_{N_r}] \in \mathbb{R}^{N_r \times 1}, & \mathbf{f}_r &= [\tilde{f}_1, \tilde{f}_2, \dots, \tilde{f}_{N_r}] \in \mathbb{R}^{N_r \times 1}. \end{aligned}$$

Here, $\mathbf{h}_r, \mathbf{s}_r, \boldsymbol{\mu}_r$ and $\mathbf{f}_r$, also $\boldsymbol{\mathcal{H}}_r := \mathbf{h}_r + \mathbf{s}_r$ are column vectors of size N_r that describe the state of the system in a reduced form.

Substituting (B.7) into the system (B.5), we obtain:

$$M\Phi\dot{\mathbf{h}}_r = -\frac{1}{3}\hat{K}(\Phi\mathbf{h}_r)\Phi\boldsymbol{\mu}_r, \quad (\text{B.9a})$$

$$M\Phi\boldsymbol{\mu}_r = M\Phi\boldsymbol{\mathcal{H}}_r + \frac{1}{C_a}K\Phi\boldsymbol{\mathcal{H}}_r, \quad (\text{B.9b})$$

$$M\Phi\dot{\mathbf{s}}_r + cK\Phi\mathbf{s}_r = M\Phi\mathbf{f}_r, \quad (\text{B.9c})$$

$$\mathbf{h}_r(0) = \Phi\mathbf{h}_0, \mathbf{s}_r(0) = \Phi\mathbf{s}_0, \quad (\text{B.9d})$$

This still gives a system of N equations in N_r unknowns. To obtain a closed reduced-

order system, we project the equations onto the reduced subspace, which yields

$$M_r \dot{\mathbf{h}}_r = -\frac{1}{3} \hat{K}_r (\Phi \mathbf{h}_r) \boldsymbol{\mu}_r, \quad (\text{B.10a})$$

$$M_r \boldsymbol{\mu}_r = M_r \boldsymbol{\mathcal{H}}_r + \frac{1}{C_a} K_r \boldsymbol{\mathcal{H}}_r, \quad (\text{B.10b})$$

$$M_r \dot{\mathbf{s}}_r + c K_r \mathbf{s}_r = M_r \mathbf{f}_r, \quad (\text{B.10c})$$

$$\mathbf{h}_r(0) = \Phi \mathbf{h}_0, \mathbf{s}_r(0) = \Phi \mathbf{s}_0, \quad (\text{B.10d})$$

where $M_r = \Phi^T M \Phi$, $K_r = \Phi^T K \Phi$, $\hat{K}_r = \Phi^T \hat{K}(\Phi \mathbf{h}_r) \Phi \in \mathbb{R}^{N_r \times N_r}$. Then, the high-order solution is approximated as (B.7).

B.2.2 Proper Orthogonal Decomposition (POD)

The quality of ROM relies on the appropriate matrix Φ used for projection in (B.7). In the context of POD model reduction, basis vectors are derived from samples (termed snapshots matrix, $\mathbf{H} \in \mathbb{R}^{N \times N_s}$, N_s is the number of snapshots) of the full-order model solutions from (B.5). The POD approach frames this as a minimisation problem [109, 166]. Specifically, the N_r basis vectors, represented by $\Phi = \{\phi_i\}_{i=1}^{N_r}$, are determined to minimise the least-squares approximation errors across the N_s snapshots

$$\min_{\{\phi_i\}_{i=1}^{N_r}} \sum_{j=1}^{N_s} \left\| \mathbf{H}_j - \sum_{i=1}^{N_r} (\mathbf{H}_j^T \phi_i) \phi_i \right\|_2^2, \quad (\text{B.11})$$

such that $\phi_i^T \phi_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta. Within the POD framework for $\mathbb{R}^N$, the optimisation problem (B.11) targets an orthonormal basis $\{\phi_i\}_{i=1}^{N_r}$. The aim is to narrow the differences between the snapshots and their projections on this basis. It has been proven that the solution to (B.11) is directly linked to the singular value decomposition (SVD) of $\mathbf{H}$ [109], which can be stated as:

$$\mathbf{H} = \Phi \Sigma \mathbf{W}^T, \quad (\text{B.12})$$

where $\Phi \in \mathbb{R}^{N \times N_r}$, $\Sigma \in \mathbb{R}^{N_r \times N_r}$, and $\mathbf{W} \in \mathbb{R}^{N_r \times N_r}$. The POD basis are the first N_r columns $\{\phi_i\}_{i=1}^{N_r}$ of the matrix Φ . If we selected N_r mode well that means it is possible to do the evolution of (B.7) with only N_r -modes. The algorithm (10) summarises the procedure.

Algorithm 10 Projection-based ROM

Require: Collect snapshot matrix $\mathbf{H} \in \mathbb{R}^{N \times N_s}$

- 1: Compute singular value decomposition $[\Phi, \Sigma, \mathbf{W}] = \text{svd}(\mathbf{H})$
 - 2: Define $\Phi = \{\phi_i\}_{i=1}^{N_r}$.
 - 3: Compute M_r , $\hat{K}_r$, and K_r .
 - 4: Solve (B.10)
 - 5: Obtain the high-order solution from (B.7)
-

B.2.3 Numerical Result

In our computational study of the Full Order Model (FOM), as defined by Equation (B.5), we use initial conditions $s_0 = 1 + 0.1 \sin\left(\frac{5\pi x}{L}\right)$ and $h_0 = 1 + 0.5 \cos\left(\frac{2\pi x}{L}\right)$. The

parameters are $f = 0$, $L = 5$, $N = 250$, $T = 5$, $C_a = c = 1$, and $dt = 0.05$. The FOM solution is shown in Figure B.5a.

For dimensional reduction, we employ the POD method, varying the basis size N_r from the set $\{1, 3, 5, 7, 9, 11, 13, 15\}$. The errors are depicted in Figure B.5. We

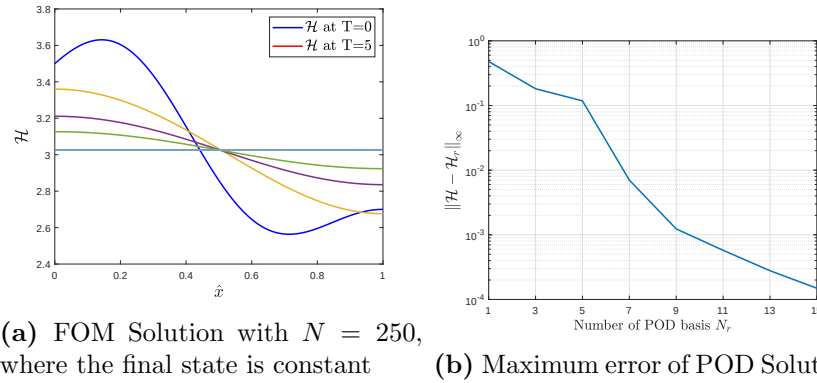


Figure B.5: FOM solution and error analysis of the POD solutions.

quantify the fidelity using the L_∞ -norm, $\max_{t \in [0, T]} \|\mathcal{H}(t) - \mathcal{H}_r(t)\|_\infty$. As N_r increases, the L_∞ -error decreases, indicating that the POD approximation improves as more modes are retained. Figure B.6a shows that the reduced model with $N_r = 3$ already captures the main behaviour of the full-order solution, although a visible discrepancy remains. By contrast, the $N_r = 7$ approximation in Figure B.6c is much closer to the FOM, with the corresponding error in Figure B.6d remaining significantly smaller over the whole time interval. These results indicate that a relatively small number of POD modes is sufficient to reproduce the dominant dynamics of the system.

In summary, POD provides a robust, low-dimensional approximation of the FOM, demonstrating its utility in complex simulations.

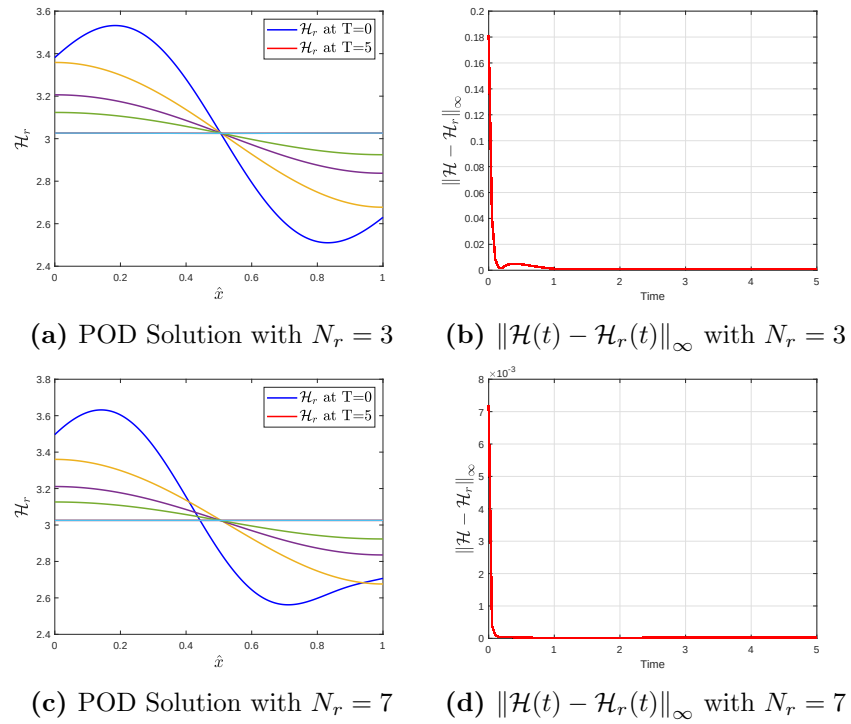


Figure B.6: POD solutions with basis sizes $N_r = 3$ and $N_r = 7$ and their corresponding L_∞ -norm errors, effectively demonstrating low-dimensional approximations of the FOM.

Appendix C

Appendices for Chapter 6

C.1 LQR closed-form optimal actuator position

Consider the scalar system obtained by projecting a diffusion PDE onto a finite-element basis with a single basis, i.e, a single piecewise linear hat function defined on $[0, 1]$:

$$\dot{h}(t) = A h(t) + B(w) u(t), \quad t \in [0, T], \quad h(0) = h_0, \quad (\text{C.1})$$

with quadratic cost

$$J_w(u) = \frac{\alpha}{2} h(T)^2 + \frac{\beta}{2} \int_0^T u(t)^2 dt, \quad \alpha, \beta > 0. \quad (\text{C.2})$$

For fixed w , the value function $V(t, h; w)$ solves the HJB

$$V_t + A h V_h - \frac{B(w)^2}{2\beta} V_h^2 = 0, \quad V(T, h; w) = \frac{\alpha}{2} h^2. \quad (\text{C.3})$$

With the quadratic ansatz $V(t, h; w) = \frac{1}{2} P(t; w) h^2$, the Riccati ODE is

$$P'(t) = \frac{B(w)^2}{\beta} P(t)^2 - 2A P(t), \quad P(T; w) = \alpha. \quad (\text{C.4})$$

Let $Q = 1/P$. Then

$$Q'(t) - 2A Q(t) = -\frac{B(w)^2}{\beta}, \quad Q(T) = \frac{1}{\alpha}, \quad (\text{C.5})$$

with solution

$$Q(t) = \frac{B(w)^2}{2A\beta} + \left(\frac{1}{\alpha} - \frac{B(w)^2}{2A\beta} \right) e^{-2A(T-t)}. \quad (\text{C.6})$$

The optimal cost at $(0, h_0)$ is

$$J(w) = V(0, h_0; w) = \frac{h_0^2}{2} P(0; w) = \frac{h_0^2}{2} \frac{1}{Q(0)}. \quad (\text{C.7})$$

Let the hat basis be

$$\phi(x) = \begin{cases} 2x, & 0 \leq x \leq \frac{1}{2}, \\ 2(1-x), & \frac{1}{2} \leq x \leq 1, \\ 0, & \text{otherwise,} \end{cases} \quad (\text{C.8})$$

and model the actuator as a box of half-width δ centred at $w \in [\delta, 1 - \delta]$: $b(x; w) = \mathbf{1}_{[w-\delta, w+\delta]}(x)$. The scalar gain is

$$B(w) = \int_{w-\delta}^{w+\delta} \phi(x) dx = \begin{cases} 4\delta w, & w \leq \frac{1}{2} - \delta, \\ 2\delta - 2\delta^2, & \frac{1}{2} - \delta < w < \frac{1}{2} + \delta, \\ 4\delta(1-w), & w \geq \frac{1}{2} + \delta. \end{cases} \quad (\text{C.9})$$

Writing $J(w) = \Phi(B(w)^2)$ with

$$\Phi(s) = \frac{h_0^2}{2} \left[\frac{s}{2A\beta} + \left(\frac{1}{\alpha} - \frac{s}{2A\beta} \right) e^{-2AT} \right]^{-1}, \quad (\text{C.10})$$

one checks $\Phi'(s) < 0$ for $s > 0$; hence $\text{sign } J'(w) = -\text{sign } B'(w)$. Since $B'(w) > 0$ for $w < 1/2$, $B'(w) = 0$ at $w = 1/2$, and $B'(w) < 0$ for $w > 1/2$, it follows that J is strictly decreasing on $[\delta, 1/2]$ and strictly increasing on $[1/2, 1 - \delta]$. Therefore,

$$\boxed{w^* = \frac{1}{2}} \quad (\text{C.11})$$

is the unique minimiser. The optimal feedback is $u^*(t, h; w) = -\frac{B(w)}{\beta} P(t; w) h$.

C.2 Implementing the 2D Heat Equation

Consider the dynamic system as given in equation (6.29) on $\Omega = [0, L]^2$ with homogeneous Dirichlet boundary conditions. The state is expanded in a truncated sine basis as

$$h(x, y, t) = \sum_{k=1}^{m_x} \sum_{\ell=1}^{m_y} h_{k\ell}(t) \phi_{k\ell}(x, y), \quad \phi_{k\ell}(x, y) = \frac{2}{L} \sin\left(\frac{k\pi x}{L}\right) \sin\left(\frac{\ell\pi y}{L}\right). \quad (\text{C.12})$$

Since $-\Delta \phi_{k\ell} = \lambda_{k\ell} \phi_{k\ell}$ with

$$\lambda_{k\ell} = \left(\frac{k\pi}{L}\right)^2 + \left(\frac{\ell\pi}{L}\right)^2, \quad (\text{C.13})$$

the semi-discrete dynamics are diagonal in modal coordinates:

$$\dot{h}_{k\ell}(t) = -\nu \lambda_{k\ell} h_{k\ell}(t) + \sum_{j=1}^{N_a} B_{(k,\ell),j}(w) u_j(t). \quad (\text{C.14})$$

Collecting all modes into $\mathbf{h}(t) \in \mathbb{R}^d$, with $d = m_x m_y$, yields

$$\dot{\mathbf{h}}(t) = A \mathbf{h}(t) + B(w) \mathbf{u}(t), \quad A = \text{diag}(-\nu \lambda_{k\ell}). \quad (\text{C.15})$$

Actuator influence. Each actuator j is modelled as a rectangular region centred at $w_j = (w_{x,j}, w_{y,j})$, with half-widths (δ_x, δ_y) . Its modal influence coefficients are

$$B_{(k,\ell),j}(w) = \iint_{\Omega_j} \phi_{k\ell}(x, y) dx dy = \frac{2}{L} I_k(w_{x,j}, \delta_x) I_\ell(w_{y,j}, \delta_y), \quad (\text{C.16})$$

where

$$I_k(c, \delta) = \int_{c-\delta}^{c+\delta} \sin\left(\frac{k\pi x}{L}\right) dx = \frac{L}{k\pi} \left[\cos\left(\frac{k\pi(c-\delta)}{L}\right) - \cos\left(\frac{k\pi(c+\delta)}{L}\right) \right]. \quad (\text{C.17})$$

Boundary penalty. To prevent actuators from approaching the domain boundaries, a smooth differentiable penalty is applied to each actuator centre w_j . For a single coordinate $x \in [0, L]$, the penalty is defined as

$$p(x) = \frac{4}{\left(\tanh\left(\frac{x-\delta_x}{\varepsilon}\right) + 1\right) \left(\tanh\left(\frac{L-\delta_x-x}{\varepsilon}\right) + 1\right)}, \quad \varepsilon = 0.05L, \quad (\text{C.18})$$

and the total penalty for actuator j is

$$p_j(w_j) = p(w_{x,j}) p(w_{y,j}). \quad (\text{C.19})$$

This function satisfies $p_j(w_j) \geq 1$ and grows smoothly near the boundaries, discouraging placements too close to the domain edges whilst maintaining differentiability for gradient-based optimisation.

Time discretisation. An implicit-Euler step with time step Δt gives

$$(I - \Delta t A) \mathbf{h}_{n+1} = \mathbf{h}_n + \Delta t B(w) \mathbf{u}_n, \quad (\text{C.20})$$

that is,

$$\mathbf{h}_{n+1} = A_d \mathbf{h}_n + B_d(w) \mathbf{u}_n, \quad A_d = (I - \Delta t A)^{-1}, \quad B_d(w) = \Delta t (I - \Delta t A)^{-1} B(w). \quad (\text{C.21})$$

Initial condition generation. Each training batch consists of B randomly generated smooth initial conditions. For each sample $i = 1, \dots, B$, modal coefficients are drawn as

$$h_{k\ell}^{(i)}(t_0) = \lambda^{(i)} \xi_{k\ell}^{(i)} \frac{1}{1 + k^2 + \ell^2}, \quad (\text{C.22})$$

where $\xi_{k\ell}^{(i)} \sim \mathcal{N}(0, 1)$ are samples taken from the standard normal distribution, and $\lambda^{(i)}$ is a random scaling factor drawn uniformly from an interval $[\lambda_{\min}, \lambda_{\max}]$ (typically $[0, 1]$). The decay factor $(1 + k^2 + \ell^2)^{-1}$ suppresses high-frequency modes, ensuring spatial smoothness of $h(x, y, 0)$ and simulating physically realistic temperature profiles. These randomised yet smooth initial conditions promote generalisation of the learned feedback law over diverse initial states.

C.3 Finite-horizon discrete LQR (Riccati)

For the discrete dynamics

$$\mathbf{h}_{k+1} = A_d \mathbf{h}_k + B_d(w) \mathbf{u}_k, \quad (\text{C.23})$$

and quadratic cost

$$J = \frac{1}{2} \alpha \|\mathbf{h}_N\|_2^2 + \frac{1}{2} \sum_{k=0}^{N-1} \mathbf{u}_k^\top R(w) \mathbf{u}_k, \quad R(w) = \beta \operatorname{diag}(p(w)), \quad (\text{C.24})$$

the Bellman principle with the quadratic ansatz $V_k(\mathbf{h}) = \frac{1}{2} \mathbf{h}^\top P_k \mathbf{h}$ gives the backward Riccati recursion

$$P_N = \alpha I, \quad P_k = A_d^\top P_{k+1} A_d - A_d^\top P_{k+1} B_d (R + B_d^\top P_{k+1} B_d)^{-1} B_d^\top P_{k+1} A_d. \quad (\text{C.25})$$

The corresponding optimal feedback gain is

$$\mathbf{u}_k^* = -K_k \mathbf{h}_k, \quad K_k = (R + B_d^\top P_{k+1} B_d)^{-1} B_d^\top P_{k+1} A_d. \quad (\text{C.26})$$

This Riccati feedback serves as the exact finite-horizon solution for a fixed actuator layout w , and provides the benchmark reference in Section 6.6.