From optimization to learning: a unified perspective on control algorithms

Miguel A. Mendez*, Romain Poletti, Lorenzo Schena,
Sebastiano Randino, Francisco Monteiro,
Tommaso De Maria, Yannick Lecomte

von Karman Institute for Fluid Dynamics

February 2026

This lecture develops a unified perspective on control and learning as two complementary forms of adaptation through feedback. It examines the interplay between model-based control, data-driven model identification, and model-free reinforcement learning, emphasizing how assumptions about system knowledge shape both optimization strategies and closed-loop behavior.

We begin with classical optimal control, where the dynamics are assumed known and policy improvement exploits analytical structure and adjoint sensitivities. We then consider data-driven identification and digital twinning, highlighting how model updating and control interact when learning occurs under feedback.

The second part shifts to model-free methods, where optimal policies are learned directly from data. We contrast policy-based approaches—where a structured control law is optimized using techniques such as Bayesian optimization—with value-based methods such as Q-learning, where the long-term value of state–action pairs is learned and the policy emerges implicitly. This comparison clarifies the trade-off between structural bias and representational flexibility, as well as the practical challenges of exploration and function approximation.

Simple tutorial problems inspired by thermal and fluid systems illustrate the common structure underlying these approaches. The central message is that control and learning are not competing paradigms, but complementary strategies that differ primarily in what is assumed known, what is inferred from data, and how optimization is carried out within the closed loop.

*mendez@vki.ac.be

Contents

List of Symbols

State-Space Dynamics and Environment

$\boldsymbol{s}_\circ(t)$	Continuous-time state vector (white-box)
$\boldsymbol{a}_\circ(t)$	Continuous-time control action vector
$\boldsymbol{z}(t)$	Exogenous input vector (disturbances, references)
$\mathcal{S}, \mathcal{A}, \mathcal{Z}$	State, action, and exogenous input spaces
$f_\circ(\cdot)$	Continuous-time system dynamics operator
$F_\circ(\cdot)$	Discrete-time one-step flow map
Δt	Time step size
t_k	Discrete time step ($k\Delta t$)
$F_\circ^{\mathrm{EE}}, F_\circ^{\mathrm{RK4}}$	Euler and Runge–Kutta 4th-order integration maps
k_1, k_2, k_3, k_4	coefficients in the in RK4 integration

Optimal Control and Adjoint Calculus

$J, \mathcal{R}$	Cumulative reward or cost functional
$r(\boldsymbol{s}, \boldsymbol{a})$	Instantaneous (running) reward
$\boldsymbol{\theta}_\pi$	Policy parameter vector
$\pi(\boldsymbol{s} \mid \boldsymbol{\theta}_\pi)$	Parametric control policy
T	Finite time horizon
$\mathbb{E}$	Expectation operator
$\mathcal{I}_0$	Distribution of initial states
$\mathcal{P}_z$	Process distribution for exogenous inputs
β	Exponential discount factor ($e^{-\beta t}$)
$\Phi(\boldsymbol{s}(T))$	Terminal penalty or reward
$\boldsymbol{\lambda}_\circ(t)$	Continuous-time adjoint (costate) variable
$H_\circ$	Hamiltonian functional
$\boldsymbol{S}_\pi(t)$	State sensitivity Jacobian ($\mathrm{d}\boldsymbol{s}/\mathrm{d}\boldsymbol{\theta}$)
$\widetilde{\mathcal{R}}$	Augmented reward functional

Identification and Digital Twinning

$\boldsymbol{o}_\bullet(t)$	Measured output (black-box/real system)
$\boldsymbol{o}_\circ(t)$	Model-predicted output (white-box/digital twin)
$h(\cdot)$	Observation or measurement mapping
$\boldsymbol{p}$	Physical model parameters (e.g., friction, stiffness)
$\boldsymbol{\theta}_p$	Parameters of the closure law or regressor
$g(\cdot)$	Closure law mapping $\boldsymbol{s}$ to $\boldsymbol{p}$
$\ell(\cdot)$	Loss function for identification mismatch
$\boldsymbol{\Phi}(\boldsymbol{s})$	Regressor matrix for linear-in-parameter models
T_o	Duration of learning/identification episode

Wind Turbine Tutorial Case Study

$v_w(t)$	Wind inflow velocity
$\omega(t)$	Rotor angular speed
$\xi(t)$	Tip-speed ratio (TSR)
$C_p(\xi)$	Power coefficient
R	Rotor radius
ρ, A	Air density and rotor swept area
T_a	Aerodynamic torque
T_g	Generator (control) torque
T_{ff}	Feed-forward equilibrium torque
$\xi_\star$	Optimal tip-speed ratio
K	Feedback gain (from LQR)

Reinforcement Learning (Policy and Value Based)

$V(\boldsymbol{s})$	State-value function
$Q(\boldsymbol{s}, \boldsymbol{a})$	State-action value function
γ	Discount factor ($0 \leq \gamma < 1$)
ε	Exploration probability (ε-greedy)
$m(\cdot), \kappa(\cdot, \cdot)$	Gaussian Process mean and covariance functions
$\mathbf{K}$	Training covariance matrix
μ_*, σ_*^2	Posterior predictive mean and variance (GP)
$n_s, n_{\dot{s}}, n_a$	Number of discretization bins (tabular RL)

1 Adaptation Through Feedback

Every intelligent system, whether biological or artificial, must be able to adapt to a variable environment in order to achieve its goals. Adaptation implies some form of learning, which can be described as the process of modifying internal representations of the system's environment or decision rules based on experience so as to improve future performance on a specific task.

Any form of learning requires some feedback. Historically, the problem of manipulating feedback mechanisms to regulate the behaviour of physical systems has originated in the domain of optimal control theory, where feedback from state measurements or suitable estimates is used to compute control actions that steer a dynamical system towards a prescribed objective. In control theory, performance is quantified through a cost functional, typically expressing tracking error, energy expenditure, or long-term operating efficiency.

On a much broader scale, the problem of feedback-driven adaptation has become a central theme in machine learning, where discrepancies between predictions and observations provide feedback signals that drive the adjustment of parameters in statistical models. Machine learning spans a wide range of applications and communities, and has led to a large ecosystem of methods and terminology focused on data-driven modelling and decision making. In this settings, performance is assessed through metrics such as prediction error, classification accuracy, or expected reward.

Although machine learning and control theory have branched into specialised fields, their conceptual links and common roots can be traced back to Norbert Wiener's idea of cybernetics (from the Greek word κυβερνάω (kybernáō), meaning "to steer") in 1948 (Wiener, 2019). Cybernetics sought to build a unified framework and language for systems—biological, mechanical, or computational—that can sense their environment, process information, and act so as to achieve a goal.

At a fundamental level, control theory and machine learning rely on the same core mechanism: feedback from past performance is used to adapt future actions or internal representations. This shared reliance on feedback naturally leads both control and learning problems to be formulated as optimization problems under uncertainty. Making this connection explicit is essential for understanding how control and learning can be combined effectively, and motivates the unified perspective developed in this lecture.

A direct connection between control and learning appears in reinforcement learning (Sutton and Barto, 2018), where the objective is to construct a policy—a feedback control law—that maximizes cumulative reward from data generated through interaction with the environment. This formulation is closely related to classical control ideas and has deep roots in dynamic programming and optimal control (Bellman, 1957; Sutton et al., 1992). Despite these early connections, reinforcement learning and optimal control evolved for many years within largely separate research communities, with limited cross-fertilisation. More recent advances have renewed interest in their common structure and complementary strengths (Bertsekas, 2012; Recht, 2019). A similar convergence is now taking place between control theory and statistical learning, driven by progress in data-driven modelling, system identification, and sample-efficient optimization (Annaswamy, 2023; Dean et al., 2019; Tsiamis and Pappas, 2022).

Much of the recent progress at the interface between control and learning is driven by hybrid approaches that blend ideas from both fields. In reinforcement learning, increasing

attention is devoted to exploiting dynamical models—whether known, approximated, or learned—to improve data efficiency and recover analytical structure (Luo et al., 2024; Li and Han, 2022). Conversely, in control applications where accurate models are unavailable, learning tools are used to identify, refine, or adapt system models directly from data (Brunke et al., 2022). These developments point toward approaches in which models and policies are jointly informed by data (Pinosky et al., 2022; Schena et al., 2024). We return to hybrid methods in Lecture 13.

The scope of this lecture is to provide a general and, where possible, unifying overview of tools from model-based control, data-driven model updating, and model-free control. Many of these topics are presented at a deliberately high level, as they are treated in greater depth in other dedicated lectures of the course. The lecture also includes hands-on experience on a relatively simple problem that mimics key challenges encountered in the control of thermal and fluid systems.

The lecture is structured as follows. Section 2 introduces the model-based control problem, in which a perfect model of the system is assumed to be available and the objective is to determine an optimal control strategy to achieve a prescribed goal. Section 4 then shifts perspective to the problem of learning a model from data, focusing on system identification methods aimed at inferring dynamical models from observations. In Section 5, these two perspectives are brought together by considering the joint problem of identification and control, in which the system model and the control strategy are learned simultaneously. Section 6 is devoted to the model-free setting, where the objective is to control a system optimally without explicitly constructing a model of its dynamics.

2 The Model Based Control Problem

We begin with fully model-based control problem, in which the governing equations are known and the full system state is available for feedback. In this "white-box" setting (in Wiener (2019)'s terminology), we denote all model-based quantities with a subscript $\circ$. A schematic illustrating the key elements of this setting is shown in Figure 1. This consists of (1) the environment (plant) to be controlled, (2) an agent/controller that actuates on it informed by observation of the system state, (3) a reward function that evaluates the quality of the actions according to a desired goal and (4) an optimizer which drives the improvement of the agent/controller.

We consider a continuous dynamical system with state $\boldsymbol{s}_\circ(t) \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$, evolving in time $t \in \mathbb{R}$ under the influence of control actions $\boldsymbol{a}_\circ(t) \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ and exogenous inputs $\boldsymbol{z}(t) \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}$. The closed-loop system is given by[1]

$$\begin{cases} \dot{\boldsymbol{s}}_\circ(t) = f_\circ\Big(\boldsymbol{s}_\circ(t),\, \boldsymbol{z}(t), \boldsymbol{a}_\circ(t; \boldsymbol{\theta}_{\circ,\pi})\Big), \\ \boldsymbol{a}_\circ(t) = \pi\Big(\boldsymbol{s}_\circ(t) \mid \boldsymbol{\theta}_{\circ,\pi}\Big), \\ \boldsymbol{s}_\circ(0) = \boldsymbol{s}_{\circ,0}. \end{cases} \tag{1}$$

The evolution map $f_\circ : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \to \mathcal{S}$ specifies how the system state changes over time according to known physical laws, given the current state, control action, and exogenous

[1] We here follow the reinforcement-learning-inspired notation $(\boldsymbol{s}, \boldsymbol{a})$ for states and actions, rather than the classical $(\boldsymbol{x}, \boldsymbol{u})$ of control theory. Apologies to the control community!

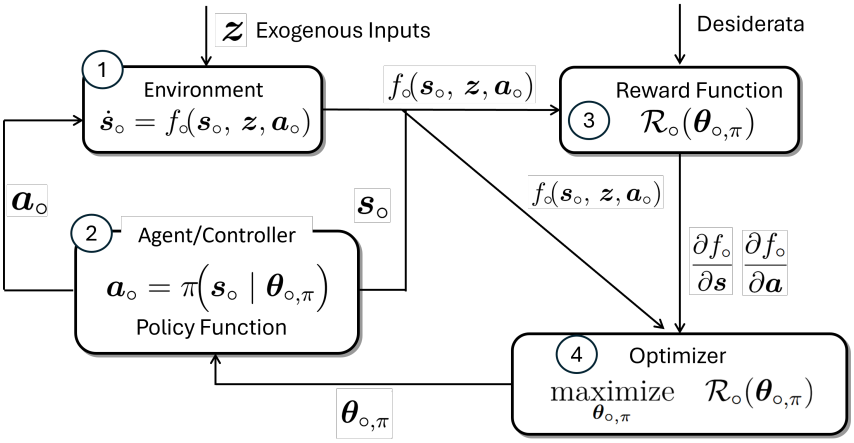


Figure 1: Fully model-based ("white-box") control setting. The environment evolves according to known deterministic dynamics under the action of exogenous inputs and control actions provided by a feedback policy. Performance is evaluated through a reward functional, and an optimizer updates the policy parameters. Crucially, the dynamics map f_o is known and can be used to predict closed-loop trajectories, to evaluate the reward functional, and to compute all necessary gradients and the updates produced by the optimizer tuning the policy parameters.

inputs. All uncertainties enter only through the exogenous input, which includes reference signals to be tracked, external disturbances to be rejected, and parametric variations represented as additional inputs.

Focusing on closed loop control[2], the action vector $\boldsymbol{a}_\circ(t)$ is prescribed by a *policy* (or control law) $\pi_\circ : \mathcal{S} \to \mathcal{A}$, which maps the current[3] (fully observed) state to a control actuation designed to drive the system toward a desired trajectory or operating condition. We treat $\pi_\circ$ as a parametric function, depending on a set of *policy parameters* $\boldsymbol{\theta}_{\circ,\pi} \in \mathbb{R}^{n_\pi}$. These parameters are initially unknown, and determining (or optimizing) them to achieve the desired behavior constitutes the essence of the *control synthesis problem*.

Depending on the control task, the policy function can take different forms. In regulation or stabilization problems, the control action is often expressed as a function of an *error signal* (e.g. deviation from a desired state or setpoint). In more general settings, the policy is taken as a function of the full state as in (1).

To synthesize the policy, we must specify what it means for the controller to behave "well". In classical optimal control, this is expressed through a *instantaneous reward*[4] $r_{\circ,\pi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which assigns credit to each state–action pair along the trajectory generated by the policy. We here restrict the evaluation of the policy to a finite time horizon $T > 0$ and thus define the associated optimization problem as

$$\underset{\boldsymbol{\theta}_{\circ,\pi}}{\text{maximize}} \quad \mathcal{R}_\circ(\boldsymbol{\theta}_{\circ,\pi}) = \int_0^T r_{\circ,\pi}(\boldsymbol{s}_\circ(t),\ \boldsymbol{z}(t), \boldsymbol{a}_\circ(t; \boldsymbol{\theta}_{\circ,\pi}))\ \mathrm{d}t, \tag{2}$$

[2]In the open loop control problem, the action is solely a function of time and is agnostic to the system state. Such controllers cannot react to unknown disturbances or modelling errors and are not considered in this lecture, where the emphasis is on feedback (closed-loop) policies.

[3]In a slightly more general approach, we could make the policy history-dependent, that is $\boldsymbol{a}_\circ(t) = \pi(\boldsymbol{s}_\circ(t), \dots \boldsymbol{s}(t - \Delta t))$. For the purposes of this section, we do not consider this case.

[4]or a *running cost*, if the problem is to be set as a minimization.

where the state $\boldsymbol{s}_\circ(t)$ and control $\boldsymbol{a}_\circ(t)$ evolve according to the closed-loop dynamics (1). The functional $\mathcal{R}_\circ$ can thus be interpreted as a cumulative reward, which depends on the policy parameters $\boldsymbol{\theta}_{\circ,\pi}$ only through the induced trajectory.

A more robust alternative is to account for the uncertainty in the exogenous input, described as a process $\boldsymbol{z}(\cdot) \sim \mathcal{P}_z$ and allowing the initial condition to be random with $\boldsymbol{s}_\circ(0) \sim \mathcal{I}_0$, could be

$$\underset{\boldsymbol{\theta}_{\circ,\pi}}{\text{maximize}} \quad \mathcal{R}_\circ(\boldsymbol{\theta}_{\circ,\pi}) = \mathbb{E}_{\boldsymbol{s}_{\circ,0}\sim\mathcal{I}_0,\, \boldsymbol{z}(\cdot)\sim\mathcal{P}_z}\left[\int_0^T r_{\circ,\pi}(\boldsymbol{s}_\circ(t),\, \boldsymbol{z}(t),\, \boldsymbol{a}_\circ(t;\boldsymbol{\theta}_{\circ,\pi}))\, \mathrm{d}t\right], \tag{3}$$

that is averaging over all possible realisations of the exogenous input and initial condition. This yields a control policy that is robust in expectation, reducing sensitivity to specific disturbance scenarios and initial states and promoting consistent performance across operating conditions. In practice, the expectation in (3) is typically approximated empirically using a Monte Carlo method, by evaluating the integral over multiple independent rollouts corresponding to different realisations of the exogenous input and initial condition.

Many variants of (2) or (3) are possible. For example, the instantaneous reward can be made explicitly dependent on time, by introducing an exponential discount factor $r_{\circ,\pi}(t, \boldsymbol{s}, \boldsymbol{a}) = e^{-\beta t}\, \tilde{r}_{\circ,\pi}(\boldsymbol{s}, \boldsymbol{a})$ with $\beta > 0$, to emphasise near-term performance. Such formulation allows also to consider infinite time horizons $T \to \infty$ using the same tools described in these notes. Other variants could include an additional terminal penalty/reward $\Phi(\boldsymbol{s}_\circ(T))$ on the final state or state and control constraints.

In the reinforcement-learning terminology, to which we return in Section 6, the approach presented in this section is on-policy, since the data used to evaluate and improve the control policy are generated by the policy itself. Alternative formulations avoid learning a direct mapping from state to action and instead determine the control action by solving an optimization problem at each time step. This receding-horizon philosophy underlies both traditional Model Predictive Control (MPC) and off-policy reinforcement-learning methods, which rely on evaluating or improving a policy using data generated by a different decision rule (see Bertsekas (2024) for a recent discussion of the links between these approaches). These topics are discussed in the following lectures.

Finally, we close this section with the discrete variant of (1)-(2), which could be obtained either from sampling a physical system or from numerically discretising the continuous-time dynamics (1). Let $t_k = k\Delta t$ denote a uniform time grid with time step $\Delta t > 0$. We then introduce a discrete-time state $\boldsymbol{s}_{\circ,k} \approx \boldsymbol{s}_\circ(t_k)$, control input $\boldsymbol{a}_{\circ,k} \approx \boldsymbol{a}_\circ(t_k)$, and disturbance $\boldsymbol{z}_k \approx \boldsymbol{z}(t_k)$, and write the closed-loop system as

$$\begin{cases} \boldsymbol{s}_{\circ,k+1} = F_\circ\big(\boldsymbol{s}_{\circ,k},\, \boldsymbol{z}_k,\, \boldsymbol{a}_{\circ,k}\big) \\ \boldsymbol{a}_{\circ,k} = \pi_\circ\big(\boldsymbol{s}_{\circ,k} \mid \boldsymbol{\theta}_{\circ,\pi}\big), \\ \boldsymbol{s}_{\circ,0} = \boldsymbol{s}_{\circ,0}, \end{cases} \tag{4}$$

where $F_\circ$ denotes the one-step flow map induced by integrating the continuous dynamics over the interval $[t_k, t_{k+1}]$ with a suitable time integration scheme (e.g. explicit or implicit Runge–Kutta methods). The discrete-time counterpart of the performance criterion (3)

is then given by

$$\underset{\boldsymbol{\theta}_{\circ,\pi}}{\text{maximize}} \quad \mathcal{R}_\circ(\boldsymbol{\theta}_{\circ,\pi}) = \mathbb{E}_{\boldsymbol{s}_{\circ,0}\sim\mathcal{I}_0,\, \boldsymbol{z}(\cdot)\sim\mathcal{P}_z}\left[\sum_{k=0}^{N-1} r_{\circ,\pi}(\boldsymbol{s}_{\circ,k},\, \boldsymbol{z}_k,\, \boldsymbol{a}_{\circ,k})\right], \tag{5}$$

with $N = T/\Delta t$ the number of time steps.

With a slight abuse of notation, we use the same symbols for rewards and policies in the continuous- and discrete-time settings, since we do not consider problems in which both appear simultaneously. It is nevertheless important to stress that the relation between the continuous-time flow map $f_\circ$ in (1) and its discrete counterpart $F_\circ$ in (4) depends strongly on the chosen time-integration scheme. For example, using an explicit Euler scheme to discretise the continuous-time dynamics $\dot{\boldsymbol{s}}_\circ = f_\circ(\boldsymbol{s}_\circ, \boldsymbol{a}_\circ, \boldsymbol{z})$, one obtains the discrete flow map

$$\boldsymbol{s}_{\circ,k+1} = \boldsymbol{s}_{\circ,k} + \Delta t\, f_\circ(\boldsymbol{s}_{\circ,k},\, \boldsymbol{a}_{\circ,k},\, \boldsymbol{z}_k) = F_\circ^{\mathrm{EE}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}). \tag{6}$$

For a classical fourth-order Runge–Kutta scheme one has

$$\boldsymbol{s}_{\circ,k+1} = \boldsymbol{s}_{\circ,k} + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \tag{7}$$

with

$$\begin{aligned} k_1 &= f_\circ(\boldsymbol{s}_{\circ,k}, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k), & k_2 &= f_\circ\Big(\boldsymbol{s}_{\circ,k} + \tfrac{\Delta t}{2} k_1, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k\Big), \\ k_3 &= f_\circ\Big(\boldsymbol{s}_{\circ,k} + \tfrac{\Delta t}{2} k_2, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k\Big), & k_4 &= f_\circ(\boldsymbol{s}_{\circ,k} + \Delta t\, k_3, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k). \end{aligned} \tag{8}$$

Hence, in this case, the discrete flow map $F_\circ^{\mathrm{RK4}}$ is defined by the above composition of evaluations of $f_\circ$.

In practice, control design and implementation are often carried out in the discrete setting (4)–(5), whereas many of the theoretical tools we rely on (such as variational calculus, adjoint methods, or Pontryagin's principle) are derived in continuous time from (1)–(2). Working in continuous time avoids committing to any particular discretisation and thus yields results that remain valid regardless of the numerical integration approach used to obtain $F_\circ$. We encourage the reader to familiarize with both.

2.1 Pontryagin's principle of optimality

How to optimize the parameters $\boldsymbol{\theta}_{\circ,\pi}$ in Figure 1? Pontryagin's maximum (minimum) principle provides necessary conditions for optimality in continuous-time control problems. It introduces an adjoint (or costate) variable and leads to a coupled system in which the state evolves forward in time, the adjoint evolves backward, and the optimal control satisfies a pointwise optimality condition on a Hamiltonian. This framework enables the efficient computation of gradients of performance functionals without explicitly propagating forward sensitivities (Stengel, 1994).

In this lecture, Pontryagin's framework is used primarily as an adjoint calculus to compute gradients with respect to policy parameters, rather than to solve the Hamiltonian minimisation explicitly. The same adjoint-based gradient computation applies to each realisation of the stochastic inputs in (3), and the overall gradient is obtained by averaging these contributions, as in a Monte Carlo approximation of the expectation. To look for

stationary points of the cumulative reward function, we compute the Jacobian (gradient) of the reward with respect to the policy parameter and use the chain rule to obtain:

$$\frac{\mathrm{d}\mathcal{R}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} = \nabla_{\theta_{\circ,\pi}}\mathcal{R}_\circ = \int_0^T \left[\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_\circ}\frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_\circ}\left(\frac{\partial \pi_\circ}{\partial \boldsymbol{s}_\circ}\frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial \pi_\circ}{\partial \boldsymbol{\theta}_{\circ,\pi}}\right)\right]\mathrm{d}t \tag{9}$$

$$= \int_0^T \left[\left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_\circ} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_\circ}\frac{\partial \pi_\circ}{\partial \boldsymbol{s}_\circ}\right)\frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_\circ}\frac{\partial \pi_\circ}{\partial \boldsymbol{\theta}_{\circ,\pi}}\right]\mathrm{d}t. \tag{10}$$

The reader is encouraged to pause and analyze the nature and dimensions of all Jacobians that appear in the chain rule. The instantaneous reward $r_{\circ,\pi}(\boldsymbol{s},\boldsymbol{a})$ and the policy $\pi_\circ(\boldsymbol{s} \mid \boldsymbol{\theta}_{\circ,\pi})$ are ordinary multivariable functions of their arguments, so their derivatives

$$\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}} \in \mathbb{R}^{1\times n_s}, \qquad \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}} \in \mathbb{R}^{1\times n_a}, \qquad \frac{\partial \pi_\circ}{\partial \boldsymbol{s}} \in \mathbb{R}^{n_a\times n_s}, \qquad \frac{\partial \pi_\circ}{\partial \boldsymbol{\theta}_{\circ,\pi}} \in \mathbb{R}^{n_a\times n_\pi},$$

are all *partial* derivatives taken with respect to one argument while the others are held fixed. Given explicit expressions for the instantaneous reward and the policy, these Jacobians can be computed directly[5]. By contrast, the state sensitivity,

$$\frac{\mathrm{d}\boldsymbol{s}_\circ(t)}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \in \mathbb{R}^{n_s\times n_\pi},$$

is a *total* derivative, because $\boldsymbol{s}_\circ(t)$ is not an explicit function of $(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta})$ but the outcome of solving the differential equation $\dot{\boldsymbol{s}}_\circ(t) = f_\circ(\boldsymbol{s}_\circ(t), \pi_\circ(\boldsymbol{s}_\circ(t);\boldsymbol{\theta}_{\circ,\pi}), \boldsymbol{z}(t))$. A perturbation of the parameters $\boldsymbol{\theta}_{\circ,\pi}$ alters the control action, which in turn modifies the entire state trajectory through the dynamics. Thus the mapping $\boldsymbol{\theta}_{\circ,\pi} \mapsto \boldsymbol{s}_\circ(t)$ is a composite, trajectory-dependent operator, and its derivative must be taken in the total sense.

Computing these sensitivities is difficult. The state sensitivity can, in principle, be computed directly by differentiating the dynamical system with respect to the policy parameters. Let

$$\boldsymbol{S}_\pi(t) := \frac{\mathrm{d}\boldsymbol{s}_\circ(t)}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \in \mathbb{R}^{n_s\times n_\pi}$$

offer a short hand notation for the Jacobian of the state trajectory with respect to the parameters. Differentiating the ODE (1) with respect to $\boldsymbol{\theta}_{\circ,\pi}$ and exchanging the order of differentiation yields

$$\dot{\boldsymbol{S}}_\pi(t) = \frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ}\boldsymbol{S}_\pi(t) + \frac{\partial f_\circ}{\partial \boldsymbol{a}_\circ}\left(\frac{\partial \pi_\circ}{\partial \boldsymbol{s}_\circ}\boldsymbol{S}_\pi(t) + \frac{\partial \pi_\circ}{\partial \boldsymbol{\theta}_{\circ,\pi}}\right), \tag{11}$$

with initial condition

$$\boldsymbol{S}_\pi(0) = \frac{\mathrm{d}\boldsymbol{s}_\circ(0)}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} = \boldsymbol{0}. \tag{12}$$

Equation (11) constitutes the *forward sensitivity system*: a system of coupled ODEs featuring $n_s \times n_\pi$ equations to be solved in addition to the original state dynamics. Its

[5]For instance, if $r_{\circ,\pi}(\boldsymbol{s},\boldsymbol{a}) = -\|\boldsymbol{e}(\boldsymbol{s})\|_2^2$ for an error vector $\boldsymbol{e}(\boldsymbol{s}) = \boldsymbol{s} - \boldsymbol{s}_{\mathrm{ref}}$, then $\partial r_{\circ,\pi}/\partial \boldsymbol{s} = -2\,\boldsymbol{e}^\top$. Similarly, if $\pi_\circ$ is a neural network, the Jacobians $\partial \pi_\circ/\partial \boldsymbol{s}$ and $\partial \pi_\circ/\partial \boldsymbol{\theta}_{\circ,\pi}$ are obtained via standard back-propagation.

solution provides the state sensitivities $\mathrm{d}\boldsymbol{s}_\circ(t)/\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}$. In many applications the number of parameters n_π is large, making this approach computationally demanding and motivating the introduction of the adjoint method (Givoli, 2021; Bradley, 2024). This offers an alternative route to compute the gradient of $\mathcal{R}_\circ$, based on the idea of augmenting the functional (2) with a time-dependent adjoint (or costate) $\boldsymbol{\lambda}_\circ(t) \in \mathbb{R}^{n_s}$. We define the augmented reward

$$\widetilde{\mathcal{R}}_\circ(\boldsymbol{\theta}_{\circ,\pi}, \boldsymbol{\lambda}_\circ) = \int_0^T \Big[r_{\circ,\pi}\Big(\boldsymbol{s}_\circ(t), \boldsymbol{a}_\circ(t)\Big) + \underbrace{\boldsymbol{\lambda}_\circ(t)^\top \Big(f_\circ(\boldsymbol{s}_\circ(t), \boldsymbol{z}(t), \boldsymbol{a}_\circ(t)) - \dot{\boldsymbol{s}}_\circ(t)\Big)}_{=0 \text{ along admissible trajectories}} \Big] \mathrm{d}t. \tag{13}$$

The second term vanishes identically regardless of the choice of $\boldsymbol{\lambda}_\circ(t)$, because the state trajectory $\boldsymbol{s}_\circ(t)$ satisfies the dynamics $\dot{\boldsymbol{s}}_\circ(t) = f_\circ(\boldsymbol{s}_\circ(t), \boldsymbol{a}_\circ(t), \boldsymbol{z}(t))$. Therefore

$$\widetilde{\mathcal{R}}_\circ(\boldsymbol{\theta}_{\circ,\pi}, \boldsymbol{\lambda}_\circ) = \mathcal{R}_\circ(\boldsymbol{\theta}_{\circ,\pi}) \quad \text{for all admissible trajectories and all } \boldsymbol{\lambda}_\circ(t). \tag{14}$$

The gradient of $\widetilde{\mathcal{R}}_\circ$ with respect to $\boldsymbol{\theta}_{\circ,\pi}$ coincides with that of $\mathcal{R}_\circ$, regardless of how $\boldsymbol{\lambda}_\circ$ is chosen.

It is convenient to introduce the *Hamiltonian*

$$H_\circ\Big(\boldsymbol{s}_\circ, \boldsymbol{z}, \boldsymbol{a}_\circ, \boldsymbol{\lambda}_\circ, \Big) = r_{\circ,\pi}\Big(\boldsymbol{s}_\circ, \boldsymbol{a}_\circ\Big) + \boldsymbol{\lambda}_\circ^\top f_\circ(\boldsymbol{s}_\circ, \boldsymbol{z}, \boldsymbol{a}_\circ)\,, \tag{15}$$

to rewrite the augmented functional (13) as

$$\widetilde{\mathcal{R}}_\circ = \int_0^T \Big[H_\circ\Big(\boldsymbol{s}_\circ, \boldsymbol{z}, \boldsymbol{a}_\circ, \boldsymbol{\lambda}_\circ\Big) - \boldsymbol{\lambda}_\circ^\top \dot{\boldsymbol{s}}_\circ \Big] \mathrm{d}t\,. \tag{16}$$

Pontryagin's framework can be viewed as choosing the adjoint trajectory $\boldsymbol{\lambda}_\circ(t)$ so as to simplify the gradient of this augmented functional. Differentiating $\widetilde{\mathcal{R}}_\circ$ with respect to the policy parameters and using the chain rule yields

$$\begin{aligned} \frac{\mathrm{d}\widetilde{\mathcal{R}}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} = \nabla_{\theta_{\circ,\pi}} \widetilde{\mathcal{R}}_\circ = \int_0^T \Bigg[& \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_\circ} \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_\circ} \frac{\mathrm{d}\boldsymbol{a}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \\ & + \boldsymbol{\lambda}_\circ^\top \left(\frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ} \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial f_\circ}{\partial \boldsymbol{a}_\circ} \frac{\mathrm{d}\boldsymbol{a}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \right) \Bigg] \mathrm{d}t. \end{aligned} \tag{17}$$

Using integration by parts on the last term,

$$-\int_0^T \boldsymbol{\lambda}_\circ^\top \frac{\mathrm{d}}{\mathrm{d}t} \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \mathrm{d}t = -\boldsymbol{\lambda}_\circ^\top \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}}\bigg|_0^T + \int_0^T \dot{\boldsymbol{\lambda}}_\circ^\top \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \mathrm{d}t,$$

and assuming that the initial condition is independent of the parameters ($\mathrm{d}\boldsymbol{s}_\circ(0)/\mathrm{d}\boldsymbol{\theta}_{\circ,\pi} = \mathbf{0}$), we can then regroup the terms multiplying the state sensitivity to obtain

$$\frac{\mathrm{d}\widetilde{\mathcal{R}}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} = \int_0^T \left[\left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_\circ} + \boldsymbol{\lambda}_\circ^\top \frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ} + \dot{\boldsymbol{\lambda}}_\circ^\top \right) \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_\circ} + \boldsymbol{\lambda}_\circ^\top \frac{\partial f_\circ}{\partial \boldsymbol{a}_\circ} \right) \frac{\mathrm{d}\boldsymbol{a}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \right] \mathrm{d}t - \boldsymbol{\lambda}_\circ(T)^\top \frac{\mathrm{d}\boldsymbol{s}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}}\bigg|_T$$

We now exploit the freedom in the choice of the adjoint trajectory $\boldsymbol{\lambda}_\circ(t)$. Imposing that it satisfies the backward (adjoint) equation

$$-\dot{\boldsymbol{\lambda}}_\circ(t) = \left(\frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ} \right)^\top \boldsymbol{\lambda}_\circ(t) + \left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_\circ} \right)^\top = \left(\frac{\partial H_\circ}{\partial \boldsymbol{s}_\circ} \right)^\top, \qquad \boldsymbol{\lambda}_\circ(T) = \mathbf{0}, \tag{18}$$

we simplify the gradient equation to

$$\nabla_{\theta_{\circ,\pi}} \mathcal{R}_\circ = \int_0^T \left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_\circ} + \boldsymbol{\lambda}_\circ^\top \frac{\partial f_\circ}{\partial \boldsymbol{a}_\circ} \right) \frac{\mathrm{d}\boldsymbol{a}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \, \mathrm{d}t = \int_0^T \frac{\partial H_\circ}{\partial \boldsymbol{a}_\circ} \frac{\mathrm{d}\boldsymbol{a}_\circ}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \, \mathrm{d}t. \tag{19}$$

Thus, instead of solving a forward sensitivity system of size $n_s \times n_\pi$, we solve a single backward adjoint system for $\boldsymbol{\lambda}_\circ(t)$ and obtain the gradient by a single time integration.

Finally, we close this section with the discrete version of (19) for the case of the discrete system (4). Taking the gradient of (5) with respect to the control parameters and using the chain rule (disregarding for the moment the expectation operator) gives

$$\nabla_{\boldsymbol{\theta}_{\circ,\pi}} \mathcal{R}_\circ = \sum_{k=0}^{N-1} \left[\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{s}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{a}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \right]. \tag{20}$$

As in the continuous case, the difficulty lies in the state sensitivities $\mathrm{d}\boldsymbol{s}_{\circ,k}/\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}$, which satisfy a forward sensitivity recursion obtained by differentiating the state update:

$$\frac{\mathrm{d}\boldsymbol{s}_{\circ,k+1}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} = \frac{\partial F_\circ}{\partial \boldsymbol{s}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{s}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial F_\circ}{\partial \boldsymbol{a}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{a}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}}, \qquad \frac{\mathrm{d}\boldsymbol{s}_{\circ,0}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} = \boldsymbol{0}. \tag{21}$$

Solving (21) explicitly requires propagating an $n_s \times n_\pi$ Jacobian forward in time, which becomes prohibitively expensive when n_π is large. As for the continuous case, the discrete adjoint method seeks to avoid the explicit computation of these sensitivities. Expanding the reward gradient (20) backward in time and using the chain rule, we can rewrite

$$\nabla_{\boldsymbol{\theta}_{\circ,\pi}} \mathcal{R}_\circ = \sum_{k=0}^{N-1} \left[\underbrace{\left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_{\circ,k}} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_{\circ,k+1}} \frac{\partial \boldsymbol{s}_{\circ,k+1}}{\partial \boldsymbol{s}_{\circ,k}} + \cdots + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_{\circ,N-1}} \frac{\partial \boldsymbol{s}_{\circ,N-1}}{\partial \boldsymbol{s}_{\circ,k}} \right)}_{\text{total influence of } \boldsymbol{s}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{s}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{a}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \right]. \tag{22}$$

Introducing the forward mapping $\boldsymbol{s}_{\circ,k+1} = F_\circ(\boldsymbol{s}_{\circ,k}, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k)$, we recognize that the Jacobians $\partial \boldsymbol{s}_{\circ,j}/\partial \boldsymbol{s}_{\circ,k}$ for $j > k$ can be expressed in terms of products of the local sensitivities $\partial F_\circ/\partial \boldsymbol{s}_{\circ,\ell}$ for $\ell = k, \ldots, j-1$. Rather than keeping track of all these products explicitly, it is convenient to introduce a sequence of adjoint (costate) variables $\{\boldsymbol{\lambda}_{\circ,k}\}_{k=0}^N$, defined backwards in time by

$$\boldsymbol{\lambda}_{\circ,N} = \boldsymbol{0}, \qquad \boldsymbol{\lambda}_{\circ,k} = \left(\frac{\partial F_\circ}{\partial \boldsymbol{s}_{\circ,k}} \right)^\top \boldsymbol{\lambda}_{\circ,k+1} + \left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_{\circ,k}} \right)^\top, \quad k = N-1, \ldots, 0. \tag{23}$$

By construction, $\boldsymbol{\lambda}_{\circ,k}^\top$ collects the total influence of $\boldsymbol{s}_{\circ,k}$ on all future rewards, so that (22) can be rewritten compactly as

$$\nabla_{\boldsymbol{\theta}_{\circ,\pi}} \mathcal{R}_\circ = \sum_{k=0}^{N-1} \left[\boldsymbol{\lambda}_{\circ,k}^\top \frac{\mathrm{d}\boldsymbol{s}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} + \frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{a}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}} \right]. \tag{24}$$

Next, we substitute the forward sensitivity recursion (21) into (24). Collecting all terms that multiply $\mathrm{d}\boldsymbol{s}_{\circ,k}/\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}$, one finds that definition (23) allows to cancel telescopically all contributions involving the state sensitivities. The remaining term reads

$$\nabla_{\boldsymbol{\theta}_{\circ,\pi}} \mathcal{R}_\circ = \sum_{k=0}^{N-1} \left(\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{a}_{\circ,k}} + \boldsymbol{\lambda}_{\circ,k+1}^\top \frac{\partial F_\circ}{\partial \boldsymbol{a}_{\circ,k}} \right) \frac{\mathrm{d}\boldsymbol{a}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}}. \tag{25}$$

It is again convenient to introduce a discrete-time Hamiltonian,

$$H_{\circ,k}\Big(\boldsymbol{s}_{\circ,k}, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k, \boldsymbol{\lambda}_{\circ,k+1}\Big) = r_{\circ,\pi}\Big(\boldsymbol{s}_{\circ,k}, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k\Big) + \boldsymbol{\lambda}_{\circ,k+1}^{\top} F_{\circ}(\boldsymbol{s}_{\circ,k}, \boldsymbol{a}_{\circ,k}, \boldsymbol{z}_k), \tag{26}$$

so that the adjoint recursion and the gradient can be written compactly as

$$\boldsymbol{\lambda}_{\circ,N} = \mathbf{0}, \qquad \boldsymbol{\lambda}_{\circ,k} = \left(\frac{\partial H_{\circ,k}}{\partial \boldsymbol{s}_{\circ,k}}\right)^{\top}, \quad k = N-1, \dots, 0, \tag{27}$$

and (25) becomes

$$\nabla_{\boldsymbol{\theta}_{\circ,\pi}} \mathcal{R}_{\circ} = \sum_{k=0}^{N-1} \frac{\partial H_{\circ,k}}{\partial \boldsymbol{a}_{\circ,k}} \frac{\mathrm{d}\boldsymbol{a}_{\circ,k}}{\mathrm{d}\boldsymbol{\theta}_{\circ,\pi}}. \tag{28}$$

As in the continuous-time case, the discrete adjoint method replaces explicit forward sensitivity computation with a single backward sweep of the adjoint recursion (27), followed by the accumulation of the gradient contributions in (28). In practice, this is reverse-mode automatic differentiation (backpropagation through time) applied to the time-stepping scheme defined by $F_{\circ}$ and the policy $\pi_{\circ}$.

In the tutorials presented in this lecture, the adjoint computations are carried out using automatic differentiation, specifically through the JAX framework. This allows gradients to be computed efficiently without explicitly deriving or implementing the adjoint equations by hand. Nevertheless, understanding the adjoint formulation remains essential for interpreting the resulting gradients, assessing computational cost and memory requirements, and extending these methods beyond standard automatic-differentiation pipelines.

2.2 The adjoint based policy search algorithm

The adjoint-based policy search procedure in the model-based setting is summarized in Algorithm 1. At each iteration, the current policy parameters $\boldsymbol{\theta}_{\circ,\pi}^{(n)}$ define a closed-loop system that is simulated forward in time. In continuous time, this means integrating (1) over $t \in [0, T]$; in discrete time, it means iterating the state recursion (4) for $k = 0, \dots, N-1$. The resulting state and action trajectories are stored, and the cumulative reward $\mathcal{R}_{\circ}(\boldsymbol{\theta}_{\circ,\pi}^{(n)})$ is computed.

Given these trajectories, the sensitivity of the cumulative reward to the policy parameters is computed via the adjoint formulation, which requires backward-in-time propagation of the adjoint variables. In continuous time, the adjoint equation (18) is integrated from $t = T$ to $t = 0$, while in discrete time the adjoint sequence $\{\boldsymbol{\lambda}_{\circ,k}\}_{k=0}^{N}$ is computed by the backward recursion (23) with terminal condition $\boldsymbol{\lambda}_{\circ,N} = \mathbf{0}$. The state, action, and adjoint trajectories are then combined to form the gradient of the cumulative reward with respect to the policy parameters. The explicit expressions for this gradient are given by (19) in continuous time and (28) in discrete time.

Finally, the policy parameters are updated using a gradient-based optimization rule. While simple gradient ascent can be used, adaptive methods such as ADAM (Kingma and Ba, 2015) are typically preferred in data-driven control and reinforcement learning, where gradients may be noisy due to stochastic disturbances, numerical errors, or mini-batch evaluations. Compared to quasi-Newton methods such as BFGS or L–BFGS, which

Algorithm 1: Adjoint-based policy search

Input: Initial policy parameters $\boldsymbol{\theta}_{\circ,\pi}^{(0)}$, horizon T or N, learning rate η (or optimizer $\mathcal{U}$)

Initialise $n \leftarrow 0$;
while *stopping criterion not met* **do**
  Forward pass;
  Simulate the closed-loop dynamics with $\boldsymbol{\theta}_{\circ,\pi}^{(n)}$ with (1) or (4);
  Store state and action trajectories $\{\boldsymbol{s}_\circ, \boldsymbol{a}_\circ\}$;
  Compute cumulative reward $\mathcal{R}_\circ(\boldsymbol{\theta}_{\circ,\pi}^{(n)})$ with (2) or (5);
  Backward pass;
  Solve backward for the adjoint states $\{\boldsymbol{\lambda}_\circ\}$ with (18) or (23);
  Gradient evaluation;
  Compute $\boldsymbol{g}^{(n)} = \nabla_{\boldsymbol{\theta}_{\circ,\pi}} \mathcal{R}_\circ(\boldsymbol{\theta}_{\circ,\pi}^{(n)})$ using (19) or (28);
  Parameter update;
  $\boldsymbol{\theta}_{\circ,\pi}^{(n+1)} \leftarrow \mathcal{U}\left(\boldsymbol{\theta}_{\circ,\pi}^{(n)}, \boldsymbol{g}^{(n)}\right)$;
  $n \leftarrow n+1$;

require smooth, accurate gradients and incur high memory costs (Nocedal and Wright, 2006; Chong and Żak, 2013; Martins and Ning, 2022), ADAM is more robust and scalable for large, non-convex problems.

Denoting by $\boldsymbol{g}^{(n)} = \nabla_{\boldsymbol{\theta}_{\circ,\pi}} \mathcal{R}_\circ(\boldsymbol{\theta}_{\circ,\pi}^{(n)})$ the adjoint-computed gradient at iteration n, the ADAM update first proceeds with a moving average of the gradients and its squared value

$$\boldsymbol{m}^{(n+1)} = \beta_1 \boldsymbol{m}^{(n)} + (1-\beta_1)\, \boldsymbol{g}^{(n)}, \tag{29}$$

$$\boldsymbol{v}^{(n+1)} = \beta_2 \boldsymbol{v}^{(n)} + (1-\beta_2)\, \boldsymbol{g}^{(n)} \odot \boldsymbol{g}^{(n)}, \tag{30}$$

with $\beta_1, \beta_2 \in (0,1)$ and $\odot$ denoting element-wise multiplication. These are bias-corrected as

$$\hat{\boldsymbol{m}}^{(n+1)} = \boldsymbol{m}^{(n+1)} / (1-\beta_1^{n+1}), \tag{31}$$

$$\hat{\boldsymbol{v}}^{(n+1)} = \boldsymbol{v}^{(n+1)} / (1-\beta_2^{n+1}), \tag{32}$$

and the policy parameters are updated as

$$\boldsymbol{\theta}_{\circ,\pi}^{(n+1)} = \boldsymbol{\theta}_{\circ,\pi}^{(n)} + \eta \frac{\hat{\boldsymbol{m}}^{(n+1)}}{\sqrt{\hat{\boldsymbol{v}}^{(n+1)}} + \varepsilon}, \tag{33}$$

where $\eta > 0$ is the learning rate and $\varepsilon > 0$ a small regularisation constant.

The iteration is repeated until a stopping criterion is met, such as a maximum number of iterations, a sufficiently small gradient norm, or stagnation of the cumulative reward.

3 Classic linear(ized) Recipes

In the previous sections we formulated optimal control for general nonlinear systems and showed how adjoint-based optimization provides a scalable solution method. There are,

however, two important scenarios in which the optimization problem admits a closed-form solution and does not require running a gradient-based solver. These correspond to systems that are linear (or linearised) about an equilibrium operating condition. When the measurable operating parameter $\boldsymbol{q}$ is frozen, the resulting deviation dynamics become linear time-invariant (LTI), and in infinite horizon admit closed-form solutions leading to the classical Linear Quadratic Regulator (LQR) and H^∞ robust control formulations(Stengel, 1994; Bąsar and Bernhard, 1995; Zhou et al., 1996; Skogestad and Postlethwaite, 2005; Hespanha, 2018).

These methods arise as specialisations of the general framework introduced earlier, obtained by linearizing the dynamics about an equilibrium operating condition and adopting quadratic reward structures over an infinite time horizon.

In this section, we write the exogenous input as the partition $\boldsymbol{z}(t) = \Big(\boldsymbol{q}(t), \boldsymbol{w}(t)\Big)$, where $\boldsymbol{q}(t)$ denotes a measurable operating condition or scheduling parameter (e.g. reference signals or slowly varying environmental parameters) and $\boldsymbol{w}(t)$ denotes an unmeasurable disturbance to be attenuated or rejected.

We assume that $\boldsymbol{q}(t)$ varies sufficiently slowly so that $\dot{\boldsymbol{q}}(t) \approx 0$, which corresponds to a frozen-parameter or quasi-steady approximation.

For each fixed value of the measurable operating condition $\boldsymbol{q}$ and in the absence of disturbances ($\boldsymbol{w} = 0$), we define the equilibrium maps $(\boldsymbol{s}^*(\boldsymbol{q}), \boldsymbol{a}^*(\boldsymbol{q}))$ as solutions of

$$f_\circ\Big(\boldsymbol{s}^*(\boldsymbol{q}), \boldsymbol{a}^*(\boldsymbol{q}), \boldsymbol{q}, \boldsymbol{0}\Big) = 0. \tag{34}$$

These maps define an equilibrium manifold parameterised by $\boldsymbol{q}$. Classical regulation about a fixed point corresponds to the special case where this manifold reduces to a single equilibrium. We introduce deviation variables

$$\delta\boldsymbol{s}(t) = \boldsymbol{s}(t) - \boldsymbol{s}^*(\boldsymbol{q}(t)), \qquad \delta\boldsymbol{a}(t) = \boldsymbol{a}(t) - \boldsymbol{a}^*(\boldsymbol{q}(t)). \tag{35}$$

Under the frozen-parameter assumption and linearization about the manifold, the deviation dynamics become

$$\dot{\delta\boldsymbol{s}} = \boldsymbol{A}(\boldsymbol{q})\,\delta\boldsymbol{s} + \boldsymbol{B}(\boldsymbol{q})\,\delta\boldsymbol{a} + \boldsymbol{E}(\boldsymbol{q})\,\boldsymbol{w}, \tag{36}$$

where

$$\boldsymbol{A}(\boldsymbol{q}) = \left.\frac{\partial f_\circ}{\partial \boldsymbol{s}}\right|_{(\boldsymbol{s}^*(\boldsymbol{q}), \boldsymbol{a}^*(\boldsymbol{q}), \boldsymbol{q}, \boldsymbol{0})}, \quad \boldsymbol{B}(\boldsymbol{q}) = \left.\frac{\partial f_\circ}{\partial \boldsymbol{a}}\right|_{(\boldsymbol{s}^*(\boldsymbol{q}), \boldsymbol{a}^*(\boldsymbol{q}), \boldsymbol{q}, \boldsymbol{0})}, \quad \boldsymbol{E}(\boldsymbol{q}) = \left.\frac{\partial f_\circ}{\partial \boldsymbol{w}}\right|_{(\boldsymbol{s}^*(\boldsymbol{q}), \boldsymbol{a}^*(\boldsymbol{q}), \boldsymbol{q}, \boldsymbol{0})}.$$

Equation (36) defines a linear parameter-varying (LPV) system. If $\boldsymbol{q}(t)$ is held constant, the system reduces to a linear time-invariant model, for which classical LQR and H^∞ synthesis apply. We now present two classical reward structures for (36), corresponding to average-performance (LQR) and worst-case (robust) objectives.

3.1 Regulate deviations about an equilibrium: LQR

A natural objective is to regulate the deviation dynamics (36) while penalising excessive control effort. In deviation variables, the quadratic reward reads

$$r_\pi^{\mathrm{LQR}} = -\delta\boldsymbol{s}^\top \boldsymbol{Q}(\boldsymbol{q})\,\delta\boldsymbol{s} - \delta\boldsymbol{a}^\top \boldsymbol{R}(\boldsymbol{q})\,\delta\boldsymbol{a}, \tag{37}$$

where $\boldsymbol{Q}(\boldsymbol{q}) \succeq 0$ and $\boldsymbol{R}(\boldsymbol{q}) \succ 0$ are user-defined matrices.

Importantly, the reward penalizes deviations $\delta\boldsymbol{a}$ and not the equilibrium feedforward $\boldsymbol{a}^*(\boldsymbol{q})$ itself. The corresponding state-feedback policy takes the form

$$\boldsymbol{a}(t) = \boldsymbol{a}^*(\boldsymbol{q}(t)) - \boldsymbol{K}(\boldsymbol{q}(t)) \Big(\boldsymbol{s}(t) - \boldsymbol{s}^*(\boldsymbol{q}(t))\Big). \tag{38}$$

For fixed $\boldsymbol{q}$ and infinite time horizon, the optimal gain $\boldsymbol{K}(\boldsymbol{q})$ is obtained by solving the algebraic Riccati equation associated with (36) and (37) (Stengel, 1994; Başar and Bernhard, 1995; Zhou et al., 1996; Skogestad and Postlethwaite, 2005; Hespanha, 2018). The resulting closed-loop matrix is

$$\boldsymbol{A}_c(\boldsymbol{q}) = \boldsymbol{A}(\boldsymbol{q}) - \boldsymbol{B}(\boldsymbol{q})\boldsymbol{K}(\boldsymbol{q}),$$

which must be asymptotically stable.

Frequency-domain interpretation. Assume that $\boldsymbol{q}$ is frozen and that the closed-loop system is asymptotically stable. The closed-loop deviation dynamics read

$$\dot{\delta\boldsymbol{s}} = \boldsymbol{A}_c(\boldsymbol{q})\, \delta\boldsymbol{s} + \boldsymbol{E}(\boldsymbol{q})\, \boldsymbol{w}, \qquad \boldsymbol{A}_c(\boldsymbol{q}) = \boldsymbol{A}(\boldsymbol{q}) - \boldsymbol{B}(\boldsymbol{q})\boldsymbol{K}(\boldsymbol{q}).$$

To analyze disturbance amplification in the frequency domain, we introduce the Laplace transform of a signal $f(t)$:

$$\widehat{f}(s) = \int_0^\infty e^{-st} f(t)\, \mathrm{d}t, \tag{39}$$

where $s = \sigma + j\omega \in \mathbb{C}$ denotes the complex frequency.[6]

Applying the Laplace transform under zero initial conditions[7] gives

$$s\, \widehat{\delta\boldsymbol{s}}(s) = \boldsymbol{A}_c(\boldsymbol{q})\, \widehat{\delta\boldsymbol{s}}(s) + \boldsymbol{E}(\boldsymbol{q})\, \widehat{\boldsymbol{w}}(s).$$

Solving for $\widehat{\delta\boldsymbol{s}}(s)$ yields

$$\widehat{\delta\boldsymbol{s}}(s) = \Big(s\boldsymbol{I} - \boldsymbol{A}_c(\boldsymbol{q})\Big)^{-1} \boldsymbol{E}(\boldsymbol{q})\, \widehat{\boldsymbol{w}}(s).$$

The operator

$$\Big(s\boldsymbol{I} - \boldsymbol{A}_c(\boldsymbol{q})\Big)^{-1}$$

is the *resolvent* of the closed-loop matrix. It characterizes how disturbances at complex frequency s are propagated through the closed-loop dynamics.

Defining the performance output

$$\boldsymbol{y}(t) = \begin{bmatrix} \boldsymbol{Q}(\boldsymbol{q})^{1/2}\, \delta\boldsymbol{s}(t) \\ \boldsymbol{R}(\boldsymbol{q})^{1/2}\, \delta\boldsymbol{a}(t) \end{bmatrix},$$

[6] The Laplace transform is usually indicated with a capital letter, hence $F(s) \in \mathbb{C}$ would be the transform of $f(t) \in \mathbb{R}$ and $\boldsymbol{F}(s) \in \mathbb{C}^n$ would denote the transform of $\boldsymbol{f}(t) \in \mathbb{R}^n$. Do not confuse the complex scalar s with the state vector $\boldsymbol{s}$. Careful... a notational tornado is approaching!

[7] Here is another reason why control engineers traditionally use $\boldsymbol{x}$ for the state and $\boldsymbol{u}$ for the input. Using $\boldsymbol{s}$ and $\boldsymbol{a}$ risks collision with the complex frequency s and the system matrix $\boldsymbol{A}$. We nevertheless keep the reinforcement-learning notation for consistency with the rest of these notes.

the disturbance-to-output transfer function becomes

$$\boldsymbol{T}_{\boldsymbol{w}\to\boldsymbol{y}}(s;\boldsymbol{q}) = \boldsymbol{C}(\boldsymbol{q})\Big(s\boldsymbol{I} - \boldsymbol{A}_c(\boldsymbol{q})\Big)^{-1}\boldsymbol{E}(\boldsymbol{q}).$$

The infinite-horizon cumulative reward is proportional to the squared H^2 norm

$$\|\boldsymbol{T}_{\boldsymbol{w}\to\boldsymbol{y}}(\cdot;\boldsymbol{q})\|_2^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty} \operatorname{trace}\Big(\boldsymbol{T}(j\omega;\boldsymbol{q})^H\boldsymbol{T}(j\omega;\boldsymbol{q})\Big)\,\mathrm{d}\omega\,,$$

where ^{H} denotes Hermitian transposition. Thus LQR minimizes the *average* amplification induced by the closed-loop resolvent across all frequencies.

3.2 Robust Control

While LQR averages disturbance amplification over all frequencies, robust control constrains the *largest* amplification produced by the same closed-loop transfer operator.

Let

$$\boldsymbol{y}(t) = \boldsymbol{l}\Big(\boldsymbol{s}(t), \boldsymbol{a}(t), \boldsymbol{q}(t)\Big) \tag{40}$$

denote a performance output collecting the quantities to be kept small. In the reward-maximization framework, a robust instantaneous reward can be written as

$$r_\pi^{H_\infty} = -\|\boldsymbol{y}(t)\|_2^2 + \gamma^2\|\boldsymbol{w}(t)\|_2^2, \tag{41}$$

where $\gamma > 0$ is a prescribed performance level.

The corresponding optimization problem is the max–min game

$$\max_{\boldsymbol{\theta}_\pi}\ \min_{\boldsymbol{w}(\cdot)}\ \int_0^T r_\pi^{H_\infty}(t)\,\mathrm{d}t. \tag{42}$$

This formulation can be interpreted as a two-player game. The controller (through $\boldsymbol{\theta}_\pi$) attempts to maximize the cumulative reward, while an adversarial disturbance $\boldsymbol{w}(\cdot)$ attempts to minimize it by injecting the most harmful disturbance signal allowed within the admissible class. Unlike the LQR setting, where disturbances are treated statistically or in an average sense, the H^∞ objective explicitly considers the *worst possible disturbance trajectory*. The controller is therefore designed to perform satisfactorily even under the most unfavourable disturbance. In the linear time-invariant infinite-horizon case, this time-domain game admits an equivalent frequency-domain characterization. The max–min problem is equivalent to requiring the induced gain bound

$$\|\boldsymbol{T}_{\boldsymbol{w}\to\boldsymbol{y}}(\cdot;\boldsymbol{q})\|_\infty = \sup_{\omega\in\mathbb{R}} \sigma_{\max}(\boldsymbol{T}_{\boldsymbol{w}\to\boldsymbol{y}}(j\omega;\boldsymbol{q})) < \gamma. \tag{43}$$

Thus, whereas the H^2 norm integrates the Frobenius norm of the transfer operator over all frequencies, the H^∞ norm measures its largest singular value at the most amplified frequency.

At a fixed frequency ω, the worst amplification is obtained by solving

$$\max_{\|\widehat{\boldsymbol{w}}\|_2=1} \Big\|\boldsymbol{T}_{\boldsymbol{w}\to\boldsymbol{y}}(j\omega;\boldsymbol{q})\widehat{\boldsymbol{w}}\Big\|_2.$$

This optimization problem is solved by the singular value decomposition

$$\boldsymbol{T}_{\boldsymbol{w}\to\boldsymbol{y}}(j\omega;\boldsymbol{q}) = \boldsymbol{U}(\omega)\,\boldsymbol{\Sigma}(\omega)\,\boldsymbol{V}(\omega)^H.$$

The largest singular value $\sigma_{\max}(\omega)$ gives the maximal gain from disturbance to performance output at that frequency, while the corresponding right singular vector in $\boldsymbol{V}(\omega)$ identifies the unit-energy disturbance component that achieves this maximal amplification. The H^∞ norm is therefore the supremum of the spectral norm of the resolvent-weighted input–output operator over all frequencies. It captures the worst disturbance waveform that the closed loop can amplify.

For linear systems this worst-case formulation reduces to Riccati equations or Riccati inequalities (Hespanha, 2018; Skogestad and Postlethwaite, 2005; Bąsar and Bernhard, 1995; Zhou et al., 1996), providing reliable and efficient "off-the-shelf" synthesis tools.

3.3 Tutorial 1: An actuated tuned mass damper system

Linear and Nonlinear Approaches for TMD systems

In this exercise we consider a two-degree-of-freedom mechanical oscillator, shown in Fig. 2. Systems of this form arise in many areas of structural dynamics and vibration control, and are widely used as benchmark models for tuned mass dampers, vibration absorbers, and active or semi-active control devices in civil and mechanical engineering, for example for mitigating oscillations in tall buildings, bridges, and flexible structures. Variants of this model have also been used to describe sloshing mitigation in passive settings (Gligor et al., 2024), while actively controlled configurations are currently under investigation within the PhD project of T. De Maria.

The model considered in this exercise consists of two masses, m_1 and m_2, with generalized coordinates $s_1(t)$ and $s_2(t)$ denoting their displacements. The first mass is attached to a fixed support through a spring–damper pair (k_1, c_1) and represents the primary structure whose motion is not directly actuated. The second mass is connected to the opposite support through (k_2, c_2) and is equipped with an actuator that applies the control input $a(t)$ through the right support. The two subsystems are dynamically coupled by an interconnecting spring and damper (k_c, c_c), which transmit forces proportional to relative displacement and relative velocity. In the problem of sloshing control, one of the systems describes the sloshing masses while the other describes the damper system.

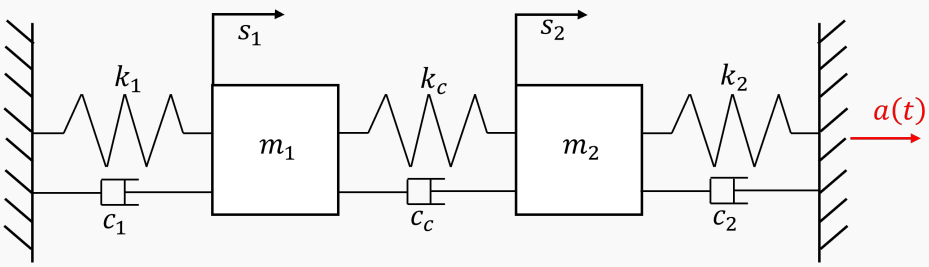


Figure 2: Two-degree-of-freedom actuated tuned mass damper model.

The system is initialised from a perturbed state with $s_1(t_0) = 0.1$, while all other

states are initially zero. In this tutorial, we consider a linear controller of the form $a(t) = -\mathbf{K}\boldsymbol{s}(t)$, where $\boldsymbol{s}(t) = [\, s_1,\ \dot{s}_1,\ s_2,\ \dot{s}_2 \,]^\top$ is the state vector, with $s_{1,2}$ denoting the displacements of the two masses and the dot denoting time derivatives, and $\mathbf{K} \in \mathbb{R}^{1\times 4}$ is the gain vector to be identified. For the system, we consider two scenarios.

1. **Linear dynamics.** All springs and dampers are assumed to have constant coefficients, leading to linear system and linear closed-loop dynamics. The continuous-time state-space model can be written as

$$\dot{\boldsymbol{s}}(t) = \boldsymbol{A}\,\boldsymbol{s}(t) + \boldsymbol{B}\,a(t),$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the system and input matrices given by

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\dfrac{k_1+k_c}{m_1} & -\dfrac{c_1+c_c}{m_1} & \dfrac{k_c}{m_1} & \dfrac{c_c}{m_1} \\ 0 & 0 & 0 & 1 \\ \dfrac{k_c}{m_2} & \dfrac{c_c}{m_2} & -\dfrac{k_2+k_c}{m_2} & -\dfrac{c_2+c_c}{m_2} \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dfrac{1}{m_2} \end{bmatrix}.$$

This case serves as a reference setting in which the system behaviour and the effect of feedback can be analyzed using standard linear control tools.

2. **Nonlinear dynamics.** In the second scenario, the control law remains linear, but the system dynamics are made nonlinear by introducing a cubic stiffness term acting on the first mass. Physically, this term represents a weak nonlinear restoring force, as commonly used to model geometric or material nonlinearities. The resulting state-space model takes the form

$$\dot{s}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1+k_c}{m_1} & -\frac{c_1+c_c}{m_1} & \frac{k_c}{m_1} & \frac{c_c}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k_c}{m_2} & \frac{c_c}{m_2} & -\frac{k_2+k_c}{m_2} & -\frac{c_2+c_c}{m_2} \end{bmatrix} \begin{bmatrix} s_1 \\ \dot{s}_1 \\ s_2 \\ \dot{s}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{k_3}{m_1} s_1^3 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_2} \end{bmatrix} a(t) \quad (44)$$

This scenario allows for assessing the robustness of linear feedback control in the presence of mild nonlinearities, and highlights the limitations of linear models when the system operates away from small-amplitude regimes.

Control laws and objective. In both scenarios we adopt the quadratic H^2-style reward as in section 3.1, and we seek a stabilising linear feedback policy

$$a_\circ(t) = \pi_\circ\Big(\boldsymbol{s}_\circ(t) \mid \boldsymbol{\theta}_{\circ,\pi}\Big) = -\boldsymbol{K}\,\boldsymbol{s}_\circ(t), \qquad \boldsymbol{K} \in \mathbb{R}^{1\times 4},$$

where $\boldsymbol{\theta}_{\circ,\pi} \equiv \boldsymbol{K}$ in this tutorial. Using the reward–maximization convention of Section 2, maximising $\mathcal{R}_\circ$ is equivalent to minimising the cost

$$\mathcal{J} = -\mathcal{R}_\circ = \int_0^\infty \Big(\boldsymbol{s}_\circ(t)^\top \boldsymbol{Q}\, \boldsymbol{s}_\circ(t) + R\, a_\circ(t)^2 \Big)\, \mathrm{d}t.$$

The matrix $\boldsymbol{Q}$ is constructed to penalise selected *performance quantities* rather than the state components individually. Specifically, we penalise the displacement and velocity of the primary mass, together with a coupled-motion measure involving both subsystems. To this end, we introduce the linear mapping $\boldsymbol{y}(t) = \boldsymbol{C}\,\boldsymbol{s}_\circ(t)$ with

$$\boldsymbol{y}(t) = \begin{bmatrix} s_1 \\ \dot{s}_1 \\ e \\ \dot{e} \end{bmatrix}, \qquad e := s_1 + s_2, \quad \dot{e} := \dot{s}_1 + \dot{s}_2, \qquad \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

Assigning independent weights to the components of $\boldsymbol{z}$ via

$$\boldsymbol{W} = \mathrm{diag}\big(w_{s_1},\, w_{\dot{s}_1},\, w_e,\, w_{\dot{e}}\big),$$

we obtain the state penalty

$$\boldsymbol{Q} = \boldsymbol{C}^\top \boldsymbol{W}\, \boldsymbol{C},$$

so that the running cost can be written explicitly as

$$\ell(t) = w_{s_1}\, s_1(t)^2 + w_{\dot{s}_1}\, \dot{s}_1(t)^2 + w_e \left(s_1(t) + s_2(t)\right)^2 + w_{\dot{e}} \left(\dot{s}_1(t) + \dot{s}_2(t)\right)^2 + R\, a_\circ(t)^2.$$

The scalar $R > 0$ weights the control effort and sets the trade-off between vibration reduction and actuation intensity. In the simulations we use $w_{s_1} = 50$, $w_{\dot{s}_1} = 2$, $w_e = 1$, $w_{\dot{e}} = 0.1$, and $R = 0.05$.

Computation of the feedback gain. The identification of a suitable gain $\boldsymbol{K}$ is carried out using two approaches, depending on the scenario.

1. **Linear case (Riccati/LQR).** For the linear model, the optimal gain is obtained from the standard LQR solution via the discrete-time algebraic Riccati equation (DARE); see Hespanha (2018). The continuous-time matrices $\mathbf{A}$ and $\mathbf{B}$ are discretized into $\mathbf{A_d}$ and $\mathbf{B_d}$ using the Euler discretization with $dt = 0.01\,\mathrm{s}$. In practice, this can be computed using the `scipy` routine `solve_discrete_are`:

```
from scipy.linalg import solve_discrete_are
def lqr_gain(Ad, Bd, Q, R):
    P = solve_discrete_are(Ad, Bd, Q, R)
        BtP = Bd.T @ P
        K = np.linalg.solve(R + BtP @ Bd, BtP @ Ad)
        return K
```

2. **General case (adjoint-based policy search).** For both the linear and nonlinear models, we can compute gradients with respect to the policy parameters $\boldsymbol{\theta}_{\circ,\pi} \equiv \boldsymbol{K}$ using the adjoint formulation developed in Section 2.2; see in particular (18) and the gradient expression (19). For the linear closed-loop dynamics with $a_\circ(t) = -\boldsymbol{K}\,\boldsymbol{s}_\circ(t)$, one has

$$\frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}, \qquad \frac{\partial f_\circ}{\partial a_\circ} = \boldsymbol{B}, \qquad \frac{\partial a_\circ}{\partial \boldsymbol{K}} = -\,\boldsymbol{s}_\circ^\top,$$

and the reward derivatives follow directly from Section 3.1 (with $\boldsymbol{e} = \boldsymbol{s}_\circ$ and constant $\boldsymbol{Q}$, R):

$$\frac{\partial r_{\circ,\pi}}{\partial \boldsymbol{s}_\circ} = -2\,\boldsymbol{s}_\circ^\top \boldsymbol{Q}, \qquad \frac{\partial r_{\circ,\pi}}{\partial a_\circ} = -2\,R\,a_\circ.$$

Substituting these quantities into (18)–(19) yields $\mathrm{d}\mathcal{R}_\circ/\mathrm{d}\boldsymbol{K}$, which is then used in a gradient-based update of $\boldsymbol{K}$ as in Algorithm 1.

In this work we focus on a specific implementation of the adjoint method based on a fixed time discretisation using an Euler explicit scheme. Once a discretisation is chosen, and assuming that all required quantities (state, control, and intermediate stages) are available at the corresponding time steps, it is natural to adopt the discrete adjoint formulation introduced earlier in this section. This avoids explicit manipulation of continuous-time adjoint equations and aligns directly with the numerical time-stepping scheme used in the forward simulation.

In practice, the discrete adjoint is implemented using the JAX framework, which allows the forward simulation, adjoint integration, and gradient assembly to be combined efficiently within a single just-in-time (JIT) compiled function. A schematic implementation is illustrated below:

```
@jax.jit
def cost_and_grad_manual(K):
        # Forward dynamics
        S = forward_trajectory(K)
        # Compute cost
        J, a, _ = cost_from_traj(K, S)
        # Compute the adjoint  variable  backward
        lam = adjoint_from_traj(K, S)
        # Compute the gradient
        gK = grad_wrt_K(K, S, a, lam)
        return J, gK # cost and gradient!
```

The reader is referred to the provided Python codes for the detailed implementation of these routines. We emphasise that the explicit computation of the adjoint variables is included here for didactic purposes: in practice, the same gradients could be obtained automatically using JAX's built-in reverse-mode differentiation.

Results, Part I: Comparison adjoint vs. LQR Figure 3 compares the time evolution of the displacements s_1 and s_2, together with the control input $a(t)$, for the uncontrolled system and for three feedback laws. The controlled cases correspond to the state-feedback gain obtained from the solution of the discrete-time algebraic Riccati equation, denoted K_{LQR}, and to gains obtained via adjoint-based policy optimization. In the latter case, we consider a full-state feedback gain, denoted $K_{\mathrm{adj,full}}$, and a reduced-structure gain, denoted $K_{\mathrm{adj,red}}$, obtained using exactly the same adjoint-based optimization procedure but restricting the feedback to a subset of the state variables (equivalent to forcing $K_3 = K_4 = 0$). All responses are computed from the

same initial condition and use identical model parameters and cost weights, so that differences in the trajectories can be attributed solely to the choice of feedback gain.

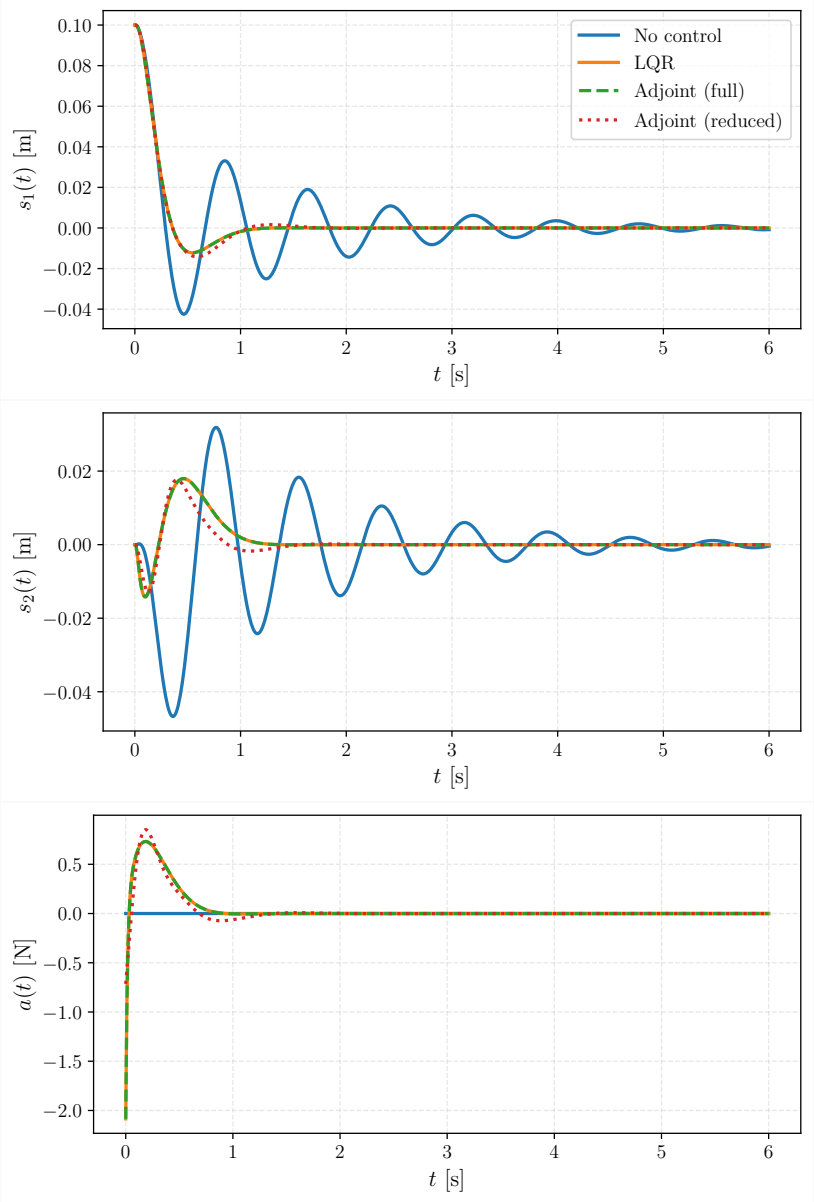


Figure 3: Time histories of the displacement s_1 (top) and control input a (bottom) for the uncontrolled system and for three feedback controllers: K_{LQR}, adjoint-optimized full-state gain $K_{\mathrm{adj,full}}$, and adjoint-optimized reduced-state gain $K_{\mathrm{adj,red}}$.

While the uncontrolled response shows lightly damped oscillations, all controllers significantly increase the closed-loop damping and drive the primary displacement s_1 rapidly to zero. The corresponding feedback gains and achieved cost values are reported in Table 1. It is immediately evident that the gains obtained through LQR and through the adjoint method coincide. This occurs because, under the same time discretization, both approaches solve the same optimization problem, namely the minimization of the identical cost function. In the LQR case, the linear structure of the

system together with the quadratic form of the cost function guarantees a known analytical solution that is unique and globally optimal. This agreement therefore provides a direct validation of the adjoint implementation: with the same temporal discretization, the adjoint algorithm converges to the LQR solution, for which the global minimum is known.

	K_1	K_2	K_3	K_4	J
LQR	20.851	6.063	10.417	4.218	0.121072
Adjoint full state	20.851	6.063	10.417	4.218	0.121072
Adjoint reduced state	8.26	3.15	0	0	0.129168

Table 1: Comparison of feedback gains and cost J for LQR and adjoint-optimized controllers.

The reader is encouraged to study the convergence plot of the ADAM optimizer from the files `2_Tut_1_Linear_Full` and `3_Tut_1_Linear_Reduced.py`.

The adjoint full-state controller achieves the same cost of the discrete LQR, because of the same gain. The close agreement between the full-state and reduced-state adjoint solutions further suggests that, for the considered excitation and weighting, several feedback channels have a lower impact on performance.

Results, Part II: Adjoint in the non-linear system The nonlinear variant is handled with *no modification* to the adjoint-based optimization algorithm: only the forward dynamics are changed by the addition of the cubic stiffness term. This highlights an important advantage of the adjoint formulation, namely that the same solver infrastructure applies seamlessly to both linear and nonlinear systems. The cubic term introduces an amplitude-dependent restoring force, providing a simple test of robustness of the feedback law outside the strictly linear regime. Full details of the implementation and numerical results are provided in `4_Tut_1_Nonlinear.py`.

4 Model Identification and Digital Twinning

We now turn to the complementary problem of *model identification*, in which the objective is to adapt a dynamical model so that it reproduces the observed behaviour of a physical system. As in Section 2, we formulate an optimization problem constrained by a dynamical system, with the parameters being adjusted now describing the model dynamics rather than the control policy and the performance criterion now measuring model-data mismatch. This problem is central to *model calibration* and forms a solid basis for updating what is commonly referred to as a *digital twin* (Xu et al., 2024; Wagg et al., 2025; Hartmann and Van der Auweraer, 2025).

Despite this shift in emphasis, the underlying mathematical structure remains essentially the same. Gradient-based optimization and adjoint methods therefore play the same key role in efficiently handling trajectory-dependent sensitivities, as discussed in Section 2; see also Mendez (2025); Mendez et al. (2025) for a broader perspective on physics-constrained learning. A schematic overview of the real-time identification problem considered in this section is shown in Figure 4.

We view this as a problem involving two dynamical systems. One is "black"– this is the

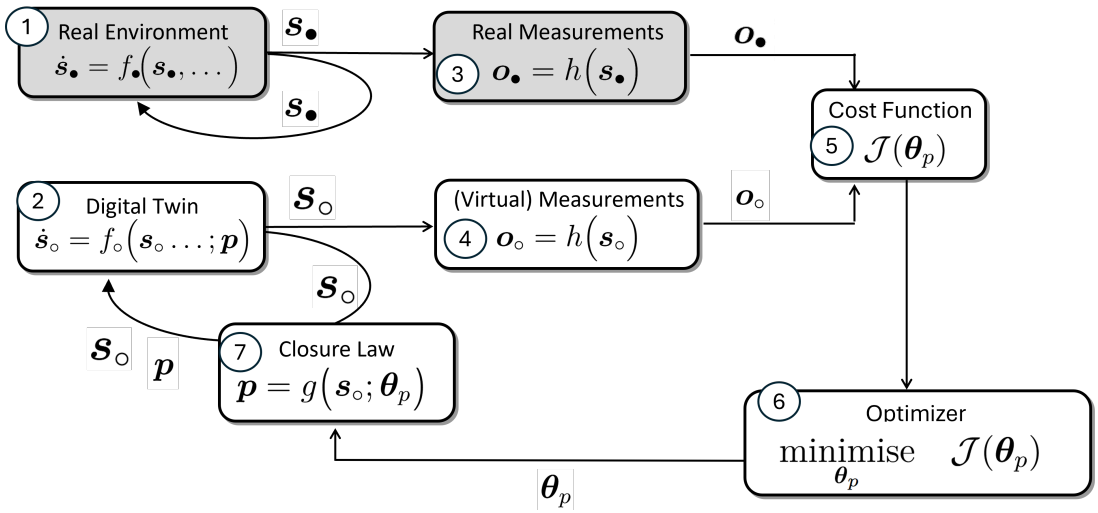


Figure 4: Real-time system identification setting. The physical environment evolves according to unknown dynamics and is observed through real measurements. In parallel, a digital twin with tunable parameters runs alongside the real system. A cost function quantifies the mismatch between real and virtual measurements, and an optimizer updates the closure-law parameters so as to continuously adapt the digital twin to the observed system behavior.

true system or plant (labelled 1 in Figure 4. Its evolution map is unknown and we denote its dynamics by $f_\bullet$. We only observe quantities $\boldsymbol{o}_\bullet \in \mathbb{R}^{n_o}$, via a measurement process (labelled 3 in Figure 4) related to the hidden states $\boldsymbol{s}_\bullet \in \mathbb{R}^{n_s}$ through an observation function $h(\cdot)$. This system is thus

$$\begin{cases} \dot{\boldsymbol{s}}_\bullet(t) = f_\bullet\big(\boldsymbol{s}_\bullet(t)\big), \\ \boldsymbol{o}_\bullet(t) = h\big(\boldsymbol{s}_\bullet(t)\big), \\ \boldsymbol{s}_\bullet(0) = \boldsymbol{s}_{\bullet,0}. \end{cases} \tag{45}$$

It is worth stressing that the system under consideration may, in practice, operate in closed loop under a fixed feedback policy $\boldsymbol{a} = \pi(\boldsymbol{s}; \dots)$. However, since this policy is assumed to be known and is not subject to optimization in this section, its effect can be absorbed into the system dynamics. Accordingly, we treat the observed system as an autonomous dynamical system with an effective evolution map $f_\bullet$.

The second system is the "white" system, our model of the system under study (labelled 2 in Figure 4). We assume full knowledge of its governing vector field $f_\circ$ and state $\boldsymbol{s}_\circ$. Here, $f_\circ$ comes from an engineering model, typically based on first principles (e.g., mechanical or thermodynamic conservation laws). This model contains unknown parameters—such as heat-transfer coefficients, friction factors, aerodynamic coefficients, or other closure terms—linked to the underlying physics through a closure law (labelled 7 in Figure 4), denoted generically by $g(\cdot)$. The model predictions are given by

$$\begin{cases} \dot{\boldsymbol{s}}_\circ(t) = f_\circ\Big(\boldsymbol{s}_\circ(t); \boldsymbol{p}\Big), \\ \boldsymbol{p} = g\Big(\boldsymbol{s}_\circ(t); \boldsymbol{\theta}_p\Big), \\ \boldsymbol{o}_\circ(t) = h\Big(\boldsymbol{s}_\circ(t)\Big), \\ \boldsymbol{s}_\circ(0) = \boldsymbol{s}_{\circ,0}. \end{cases} \tag{46}$$

Note that the problem formulation does not strictly require an explicit definition of the closure function; we could simply write $\dot{\boldsymbol{s}}_\circ(t) = f_\circ\Big(\boldsymbol{s}_\circ(t); \boldsymbol{\theta}_p\Big)$. However, the distinction between physics-based equations and data-driven closure laws will become relevant later, especially in Lecture 13.

We refer to the black-box system (45) as the *real environment* and to the white-box model (46) as its *digital twin* or *virtual environment*. The real-time adaptation, with the tailoring of the predictions to a specific physical system through continuous parameter updates distinguishes a digital twin from a static or offline-calibrated model. Because many physical systems change over time due to wear, ageing, or environmental conditions, our goal is to continuously track the evolution of the closure parameters $\boldsymbol{\theta}_p$ so that the digital twin adapts to the real system.

Identification and model updating form a single online optimization process. Over learning episodes of duration $T_o > 0$, we collect data to assess performance and update $\boldsymbol{\theta}_p$ so that the digital twin's predicted observations match those of the real system. Given observations $\{\boldsymbol{o}_\bullet(t)\}_{t\in[0,T_o]}$ and the digital twin dynamics (46), which yield predicted observations $\boldsymbol{o}_\circ(t) = h(\boldsymbol{s}_\circ(t))$, we measure performance with an integral cost functional of the form[8]

$$\underset{\boldsymbol{\theta}_p}{\text{minimize}} \quad \mathcal{J}(\boldsymbol{\theta}_p) = \mathbb{E}_{\boldsymbol{s}_{\circ,0}\sim\mathcal{I}_0,\, \boldsymbol{z}(\cdot)\sim\mathcal{P}_z}\left[\int_0^{T_o} \ell(\boldsymbol{o}_\circ(t;\boldsymbol{\theta}_p), \boldsymbol{o}_\bullet(t), t)\ \mathrm{d}t\right]. \tag{47}$$

This is mathematically analogous to (2) in the model-based control setting (labelled 5 in Figure 4). Expectation over possible disturbances can easily be introduced with no loss of generality (see Tutorial 2). The optimizer driving the model update (labelled 6 in Figure 4) can therefore proceed with the same adjoint tools introduced in the previous section.

In the identification setting, the cost is defined in terms of the discrepancy between observed quantities and the corresponding outputs predicted by the digital twin. Accordingly, we define the identification Hamiltonian

$$H_{\mathrm{ID}}\Big(\boldsymbol{s}_\circ, \boldsymbol{\lambda}_\circ, \boldsymbol{\theta}_p, t\Big) = \ell\Big(h(\boldsymbol{s}_\circ), t\Big) + \boldsymbol{\lambda}_\circ^\top f_\circ\Big(\boldsymbol{s}_\circ; \boldsymbol{p}\Big), \qquad \boldsymbol{p} = g(\boldsymbol{s}_\circ, \boldsymbol{\theta}_p), \tag{48}$$

where $\boldsymbol{\lambda}_\circ(t) \in \mathbb{R}^{n_s}$ denotes the adjoint variables enforcing the digital-twin dynamics.

As in Section 2, we choose the adjoint trajectory so as to eliminate the dependence on the state sensitivity $\mathrm{d}\boldsymbol{s}_\circ/\mathrm{d}\boldsymbol{\theta}_p$. Imposing the backward adjoint equation

$$-\dot{\boldsymbol{\lambda}}_\circ(t) = \left(\frac{\partial \ell}{\partial \boldsymbol{o}_\circ}\frac{\partial h}{\partial \boldsymbol{s}_\circ}\right)^\top + \left(\frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ}\right)^\top \boldsymbol{\lambda}_\circ(t), \qquad \boldsymbol{\lambda}_\circ(T_o) = \mathbf{0}, \tag{49}$$

[8] In the most general setting, this function could also explicitly depend on $\boldsymbol{\theta}_p$, hence $\ell(\boldsymbol{s}_\circ(t;\boldsymbol{\theta}_p),\, t;\, \boldsymbol{\theta}_p)$ via a virtual measurement process (labelled 4 in Figure 4). That is the case in which this term has a regularization on the parameters.

which follows from the chain rule since the loss depends on the state only through the observation operator h.

The adjoint system is therefore closed by the chain rule

$$\frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ} = \left.\frac{\partial f_\circ}{\partial \boldsymbol{s}_\circ}\right|_{\boldsymbol{p}} + \frac{\partial f_\circ}{\partial \boldsymbol{p}}\frac{\partial g}{\partial \boldsymbol{s}_\circ}, \tag{50}$$

since $f_\circ$ depends on $\boldsymbol{s}_\circ$ both directly and through the closure $\boldsymbol{p} = g(\boldsymbol{s}_\circ, \boldsymbol{\theta}_p)$.

Once $\boldsymbol{\lambda}_\circ(t)$ is available, the gradient of $\mathcal{J}$ with respect to the closure parameters follows as

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{\theta}_p} = \int_0^{T_o} \boldsymbol{\lambda}_\circ(t)^\top \frac{\partial f_\circ}{\partial \boldsymbol{\theta}_p}\,\mathrm{d}t. \tag{51}$$

Because $\boldsymbol{\theta}_p$ enters the dynamics only through $\boldsymbol{p} = g(\boldsymbol{s}_\circ, \boldsymbol{\theta}_p)$, the term $\partial f_\circ/\partial \boldsymbol{\theta}_p$ must be expanded as

$$\frac{\partial f_\circ}{\partial \boldsymbol{\theta}_p} = \frac{\partial f_\circ}{\partial \boldsymbol{p}}\frac{\partial g}{\partial \boldsymbol{\theta}_p}, \tag{52}$$

and therefore

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{\theta}_p} = \int_0^{T_o} \boldsymbol{\lambda}_\circ(t)^\top \frac{\partial f_\circ}{\partial \boldsymbol{p}}\Big(\boldsymbol{s}_\circ(t), \boldsymbol{p}(t)\Big)\frac{\partial g}{\partial \boldsymbol{\theta}_p}\Big(\boldsymbol{s}_\circ(t), \boldsymbol{\theta}_p\Big)\,\mathrm{d}t. \tag{53}$$

Equivalently, using the Hamiltonian definition (48), the dependence on the closure parameters can be made explicit through the chain rule

$$\frac{\partial H_{\mathrm{ID}}}{\partial \boldsymbol{\theta}_p} = \frac{\partial H_{\mathrm{ID}}}{\partial \boldsymbol{p}}\frac{\partial g}{\partial \boldsymbol{\theta}_p}, \qquad \frac{\partial H_{\mathrm{ID}}}{\partial \boldsymbol{p}} = \boldsymbol{\lambda}_\circ^\top \frac{\partial f_\circ}{\partial \boldsymbol{p}}. \tag{54}$$

The gradient of the identification cost can therefore be written as

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{\theta}_p} = \int_0^{T_o} \boldsymbol{\lambda}_\circ(t)^\top \frac{\partial f_\circ}{\partial \boldsymbol{\theta}_p}\,\mathrm{d}t = \int_0^{T_o} \frac{\partial H_{\mathrm{ID}}}{\partial \boldsymbol{p}}\frac{\partial g}{\partial \boldsymbol{\theta}_p}\,\mathrm{d}t. \tag{55}$$

The discrete-time derivation follows exactly the same steps as in Section 2 and is therefore omitted. Once the cost-function gradient is available, model identification and digital twin updating can be carried out using the same optimization strategies discussed in Section 2.

4.1 Tutorial 2: Identification of Wind Turbine Dynamics

Nonlinear Identification of Wind Turbine Dynamics

We consider a simplified problem in non-linear identification for a horizontal-axis wind turbine. The turbine has rotor diameter D and radius R, and we assume for simplicity it operates under uniform inflow with free-stream velocity $v_w(t)$ (see Figure 5, on the right). The rotor dynamics can be modeled by a single rotational degree of freedom, with angular velocity $\omega(t)$ governed by

$$\dot{\omega}(t) = \frac{T_a(\omega(t), v_w(t)) - T_g(t)}{J}, \tag{56}$$

where J is the rotor inertia, T_g is the generator torque, and T_a is the aerodynamic torque produced by the rotor, which is a control action for the turbine.

The instantaneous aerodynamic power extracted from the flow can be written as

$$P_a(t) = \frac{1}{2}\,\rho\, A\, C_p(\xi(t))\, v_w^3(t) = T_a(t)\,\omega(t), \tag{57}$$

where ρ is the air density, $A = \pi R^2$ is the swept area, and C_p is the *power coefficient*, a dimensionless quantity measuring the fraction of kinetic energy converted into mechanical power. This is function of the *tip–speed ratio*, defined as[a] $\xi = \omega R / v_w$.

In practice, the C_p is not known exactly and depends on the blade geometry, airfoil characteristics, and operating conditions such as the blade's pitch angle, which is another possible control action. We assume for simplicity that the blade pitch is kept constant, hence C_p is solely function of ξ. For a given turbine, C_p is estimated using low-order aerodynamic models such as Blade Element Momentum Theory (BEMT), possibly combined with empirical corrections for tip losses and stall (see Hansen (2015) for a concise introduction to wind turbine aerodynamics). More refined estimates may be obtained using lifting-line methods or three dimensional RANS or LES calculations, but these approaches remain computationally expensive and are not suitable for real-time applications.

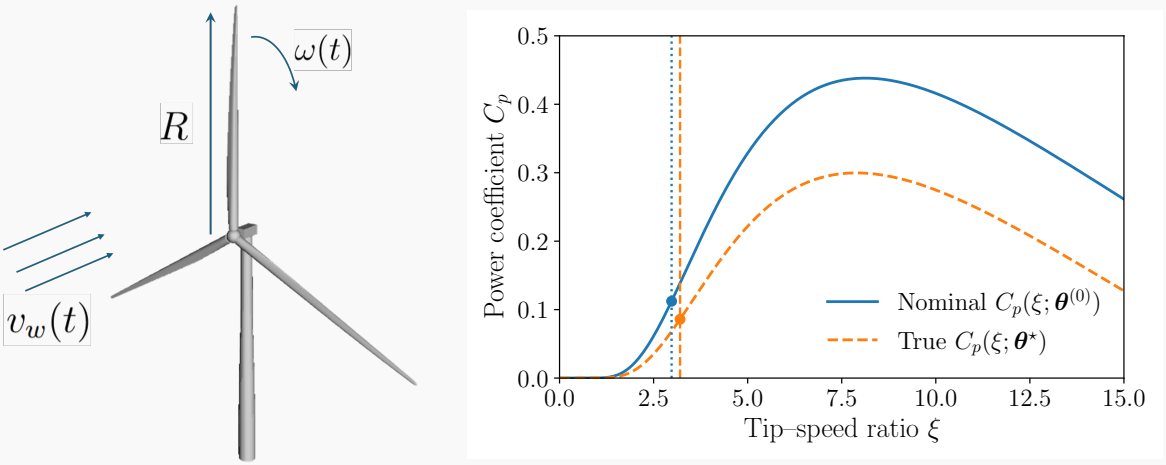


Figure 5: Schematic of the horizontal-axis wind turbine considered in this exercise (left) and expected shape of the power coefficient C_p as a function of the tip–speed ratio ξ (right). The curve exhibits a single maximum corresponding to the optimal operating point.

Scope of the exercise. For the purpose of this exercise, we assume that the turbine is well described by the following parametric form:

$$C_p(\xi;\boldsymbol{\theta}_p) = \left[c_1\left(\frac{c_2}{\xi} - c_3\right)\exp\left(-\frac{c_4}{\xi}\right)\right]_+, \quad \boldsymbol{\theta}_p = [c_1, c_2, c_3, c_4]^\top, \tag{58}$$

where $\boldsymbol{\theta}_p \in \mathbb{R}^4$ denotes the vector of coefficients identifying the turbine and $[x]_+ = \max(x, 0)$ enforces non-negativity. This expression recovers the commonly used analytical shape for C_p curves (single peak at moderate tip–speed ratio and decay for large ξ), while remaining differentiable almost everywhere and inexpensive to evaluate.

Based on prior simulations or design-stage aerodynamic estimates, our current best model of the turbine uses the nominal coefficients

$$\boldsymbol{\theta}^{(0)} = \left(c_1^{(0)},\, c_2^{(0)},\, c_3^{(0)},\, c_4^{(0)}\right)^\top = [0.22, 116, 5, 12.5]^\top. \tag{59}$$

However, we assume that the turbine is better described by a different (unknown) parameter vector $\boldsymbol{\theta}_p^\star = [0.2, 112, 5.9, 13.5]^\top$. The discrepancy could be due to modeling assumptions, unmodeled aerodynamic effects, or changes in rotor performance (e.g. surface roughness, leading-edge erosion, or blade contamination).

The corresponding curves $C_p(\xi; \boldsymbol{\theta}_p^{(0)})$ and $C_p(\xi; \boldsymbol{\theta}_p^\star)$ are illustrated in Figure 5, on the right. Clearly, the turbine performs significantly worse than expected, and the optimal operating point predicted by the nominal model does not coincide with the true one. The objective is therefore to exploit time-resolved operational data to correct the discrepancy between the model and the real system. Starting from the initial parameter estimate $\boldsymbol{\theta}_p^{(0)}$, we seek an updated parameter vector $\boldsymbol{\theta}_p$ such that the closed-loop turbine model reproduces the observed dynamics.

In this exercise, we assume that measurements of the rotor angular velocity are available, so that the observation operator reduces to the identity and the measured signal is $\omega_\bullet(t)$ in the notation of the previous section. The discrepancy between model predictions and measurements is quantified by a cost functional of the form (47), which in the present setting reads

$$\underset{\boldsymbol{\theta}_p}{\text{minimize}} \quad \mathcal{J}(\boldsymbol{\theta}_p) = \mathbb{E}_{v_w}\left[\frac{1}{2}\int_0^{T_o}\Big(\omega_\circ(t;\boldsymbol{\theta}_p) - \omega_\bullet(t)\Big)^2\,\mathrm{d}t\right], \tag{60}$$

where the expectation is approximated by an empirical average over an ensemble of sampled wind realizations.

Excitation scenarios and closed-loop stability. We consider a time-varying incoming wind speed, sampled from a stationary Gaussian process with a randomly chosen mean value in the range $\mu_w \in [4, 10]$ m/s, superimposed fluctuations at two distinct temporal scales, and an additive white-noise component. The wind covariance structure is modeled by a Gaussian kernel of the form

$$\kappa(t,t') = \sigma_{f,1}^2 \exp\!\left(-\frac{(t-t')^2}{2\,\ell_{T,1}^2}\right) + \sigma_{f,2}^2 \exp\!\left(-\frac{(t-t')^2}{2\,\ell_{T,2}^2}\right) + \sigma_n^2\,\delta(t-t'), \tag{61}$$

where $\delta(\cdot)$ denotes the Dirac delta function.

Figure 6 shows, on the top panel, a sample of wind profile for an observation of $T_o = 100$s with $dt = 0.005$s, $\mu_w = 10.58$m/s and settings $[\sigma_{f1}, \sigma_{f2}, \sigma_n] = [1, 0.4, 0.1]$ $\mathrm{m}^2/\mathrm{s}^2$, and $[\ell_{T,1}, \ell_{T,2}] = [5, 0.4]$s. We assume that these wind conditions lie in the so-called region II for the turbine, that is, below the rated wind speed, where variable-speed operation is used to maximize power capture (Pao and Johnson, 2009).

The operating condition of the turbine is regulated by a classic $K\omega^2$ torque controller (see Pao and Johnson (2009); Bianchi et al. (2007); Pao et al. (2024)), in which the generator torque is prescribed as $T_g = K\,\omega^2$.

This control law is designed under the assumption of constant wind velocity v_w and aims to enforce steady operation at a prescribed tip–speed ratio $\xi_\star$. In particular, the gain K is chosen such that $\dot{\omega} = 0$ when $\xi = \xi_\star$, thereby balancing aerodynamic and generator torques at the desired operating point and maintaining the turbine near maximum power capture. Setting $T_a = T_g$ in (56), and recalling from (57) that $T_a = P_a/\omega$, the generator torque required to balance the aerodynamic torque reads

$$T_g = \frac{1}{2}\,\rho A\, R^3 \frac{C_p(\xi_\star)}{\xi_\star^3}\,\omega^2 \;\equiv\; K\,\omega^2, \tag{62}$$

which leads to the closed-loop rotor dynamics

$$\dot{\omega} = \frac{\rho A R}{2J}\left(\frac{C_p(\xi)}{\xi^3} - \frac{C_p(\xi_\star)}{\xi_\star^3}\right)\omega^2. \tag{63}$$

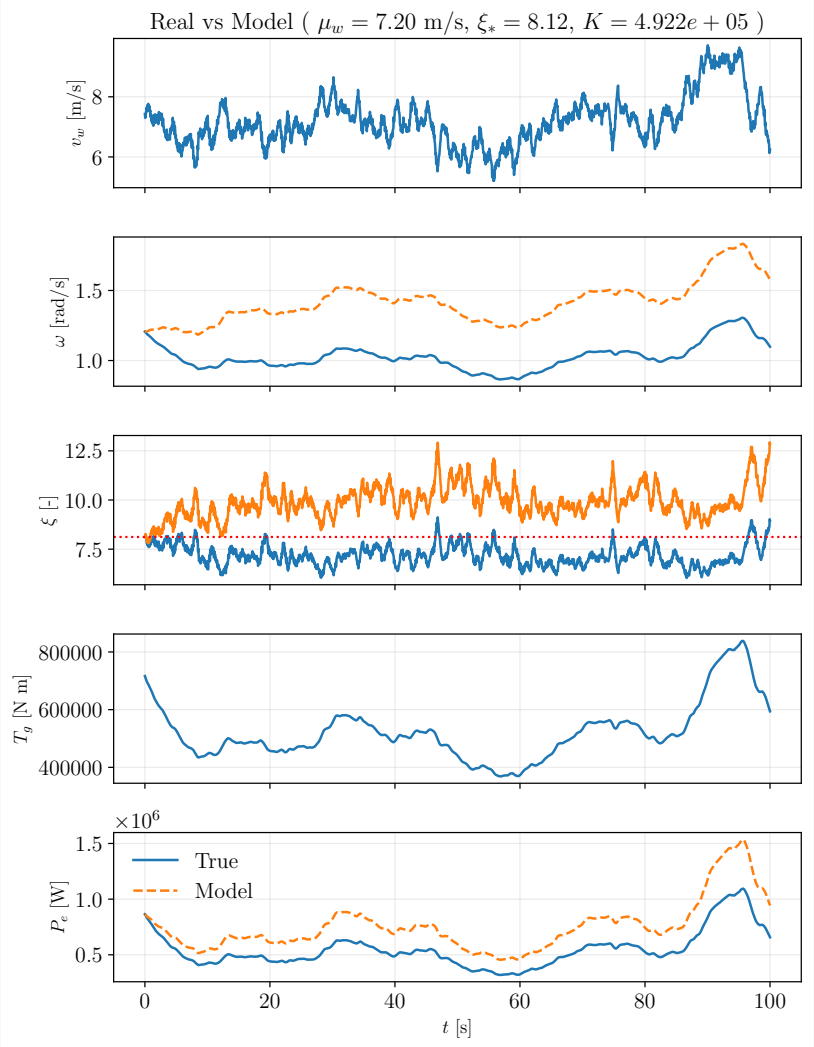


Figure 6: Example of a sampled episode of duration $T_o = 100\,\mathrm{s}$. The top panel shows a realization of the incoming wind speed generated from the Gaussian-process model. The remaining panels report, from top to bottom, the rotor angular velocity, the tip-speed ratio, the applied generator torque, and the extracted electrical power, all obtained under the $K\omega^2$ torque control law. When shown, solid and dashed lines correspond to the true system and the digital-twin prediction, respectively.

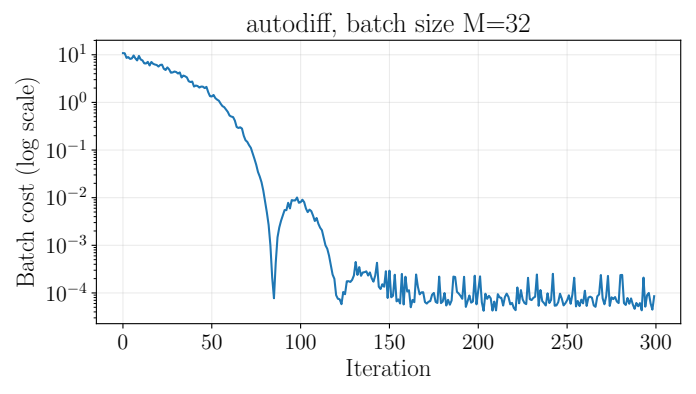
Under the simplifying assumption of constant wind velocity, the induced tip–speed–ratio dynamics $\dot{\xi} = F(\xi)$ admits a local linear stability analysis around $\xi = \xi_\star$. In this setting, stability requires

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{C_p(\xi)}{\xi^3}\right)\bigg|_{\xi=\xi_\star} < 0, \tag{64}$$

which defines a stability region in terms of the chosen operating point $\xi_\star$. The corresponding stability thresholds for the nominal and true turbine models are reported in Figure 5.

In practice, the strong wind-velocity variations considered here add forcing terms to the tip–speed–ratio dynamics that are absent from the constant-wind analysis. The stability condition above is therefore an optimistic guideline rather than a strict limit: turbine run-down may occur even above the threshold predicted for constant wind. A full stability analysis with time-varying wind, in terms of $\dot{\xi}$ and explicitly including $\dot{v}_w$, is postponed to the control exercise.

For this tutorial, it is enough to note that the $K\omega^2$ controller does not risk uncontrolled rotor acceleration. If generator torque temporarily exceeds aerodynamic torque, the rotor decelerates and may run down, but over-speeding cannot occur. One could add safety measures, such as disabling the torque controller to avoid shutdown, but numerical tests under the wind conditions used here show they are unnecessary: the closed-loop dynamics remain well behaved.



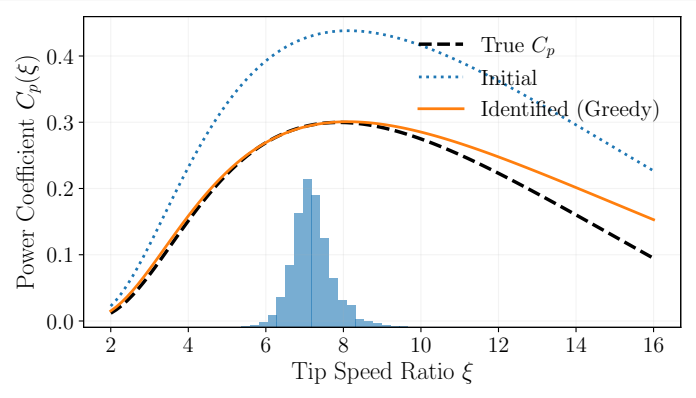


Figure 7: Top: convergence plot during the identification. Bottom: initial and identified power coefficient curves together with the ground truth and the distribution of sampled operating conditions under the $K\omega^2$ controller.

Although the $K\omega^2$ controller does not keep the turbine exactly at the optimal tip–speed ratio under strong wind variations, it yields dynamics that are sufficiently rich and stable for the identification tasks considered. Moreover, wind-induced dispersion naturally broadens the range of explored operating conditions compared with a tighter controller enforcing $\xi \approx \xi_\star$. This point will be revisited in the next exercise with more advanced control laws.

Offline identification and remarks on online extensions. In the present implementation, model identification is performed offline. A fixed dataset of 200 episodes is first collected, after which the optimization problem (60) is solved to estimate the model parameters. Extending this approach to online identification requires only minor modifications. Rather than collecting all data in advance, parameter updates can be interleaved with data acquisition, for example by running a fixed number of ADAM iterations after each new batch of episodes. From an algorithmic standpoint, the cost function and gradient evaluations remain unchanged; the distinction between offline and online identification primarily concerns the frequency of parameter updates and the available computational budget.

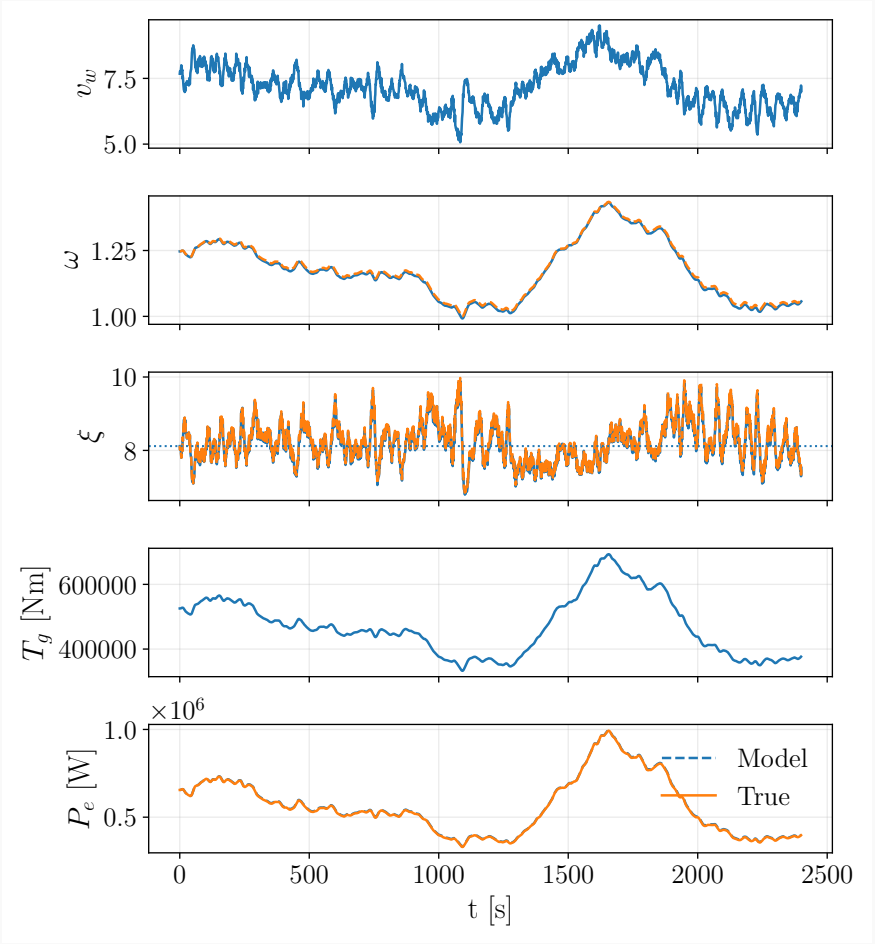


Figure 8: Same as Fig. 6, with the $K\omega^2$ controller driven by the identified model.

Results Figure 7 shows, on the top, the cost function evolution as a function of the number of iterations of the ADAM optimizer using a batch size of $M = 50$ episodes. Despite the spikes, the cost function clearly shows a good convergence.

On the bottom, the figure shows the initial power coefficient curve and the final one, together with the underlying truth. An histogram of all visited points is overlaid to the plots, showing the distribution of visited operating conditions. Despite the large variations of wind velocity, the $K\omega^2$ controller keeps the turbine reasonably close to the optimum while at the same time allowing for collecting some data in suboptimal conditions. Almost no conditions with $\xi < 5.5$ or $\xi > 10$ are observed. Avoid low values is a virtue, since these bring the higher risk of shut down, while higher values could be more interesting but not explorable with the current policy. However, since the range of explored conditions is sufficiently rich, the identification is successful and mismatches in unexplored conditions do not represent a limit. Figure 8 shows the same plots as in Figure 6 on a random episode, but this time with the identified model: within the range of operating conditions explored during the training the model matching is excellent and, accordingly, the controller performances.

4.2 The Linear Setting in Model Identification

We learned in the previous exercise that the trajectories collected for model identification must be sufficiently informative. The notion of "informativeness"—and, more generally, of identifiability—can be made more precise in the linear setting, namely in the scenario where the model dynamics are nonlinear in the state but *linear in the unknown parameters* $\boldsymbol{\theta}_p$. Then the vector field can be written as

$$f_\circ\Big(\boldsymbol{s}_\circ(t);\boldsymbol{\theta}_p\Big) = \boldsymbol{\Phi}\Big(\boldsymbol{s}_\circ(t)\Big)\,\boldsymbol{\theta}_p, \tag{65}$$

where $\boldsymbol{\Phi}(\boldsymbol{s}) \in \mathbb{R}^{n_s \times n_p}$ is a regressor matrix depending (possibly nonlinearly) on the state, and $\boldsymbol{\theta}_p \in \mathbb{R}^{n_p}$ is the vector of unknown parameters. This structure is common in practice, for instance when modelling unknown source terms, parametric forcing, or linear-in-parameter closure relations (see for instance Åström and Wittenmark (1994)).

When (65) holds, the parameter derivative appearing in the adjoint-based gradient (51) takes the simple form

$$\frac{\partial f_\circ}{\partial \boldsymbol{\theta}_p} = \boldsymbol{\Phi}\Big(\boldsymbol{s}_\circ(t)\Big).$$

Consequently, the gradient of the identification cost function (51) becomes

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{\theta}_p} = \int_0^{T_o} \boldsymbol{\lambda}_\circ(t)^\top \boldsymbol{\Phi}\Big(\boldsymbol{s}_\circ(t)\Big)\,\mathrm{d}t. \tag{66}$$

The adjoint equation remains unchanged and encodes how observation mismatches propagate backward through the nonlinear state dynamics.

Consider for the moment the idealised case in which the state trajectory $\boldsymbol{s}_\circ(t)$ (and thus the regressor $\boldsymbol{\Phi}(\boldsymbol{s}_\circ(t))$) is known and fixed over an episode $t \in [0, T_o]$. Using the linear-in-parameter structure (65), we can write the data generating equation as

$$\dot{\boldsymbol{s}}_\circ(t) = \boldsymbol{\Phi}\Big(\boldsymbol{s}_\circ(t)\Big)\,\boldsymbol{\theta}_p, \tag{67}$$

so that parameter identification reduces to fitting $\boldsymbol{\theta}_p$ to measured (or reconstructed) quantities on the left-hand side. Denoting the measured time signal to be matched by $\boldsymbol{y}(t) \in \mathbb{R}^{n_s}$ (for instance $\boldsymbol{y}(t) \approx \dot{\boldsymbol{s}}_\bullet(t)$, or a suitable projected/filtered version thereof), the batch least-squares problem reads

$$\min_{\boldsymbol{\theta}_p} \int_0^{T_o} \Big\|\boldsymbol{\Phi}\Big(\boldsymbol{s}_\circ(t)\Big)\,\boldsymbol{\theta}_p - \boldsymbol{y}(t)\Big\|_2^2\,\mathrm{d}t. \tag{68}$$

The optimality condition $\mathrm{d}\mathcal{J}/\mathrm{d}\boldsymbol{\theta}_p = 0$ yields the (normal) equations

$$\underbrace{\left(\int_0^{T_o} \boldsymbol{\Phi}(t)^\top \boldsymbol{\Phi}(t)\,\mathrm{d}t\right)}_{\boldsymbol{G}(T_o)\ \in\ \mathbb{R}^{n_p \times n_p}} \boldsymbol{\theta}_p = \int_0^{T_o} \boldsymbol{\Phi}(t)^\top \boldsymbol{y}(t)\,\mathrm{d}t, \qquad \boldsymbol{\Phi}(t) \equiv \boldsymbol{\Phi}\Big(\boldsymbol{s}_\circ(t)\Big). \tag{69}$$

Whenever the Gramian matrix

$$\boldsymbol{G}(T_o) = \int_0^{T_o} \boldsymbol{\Phi}(t)^\top \boldsymbol{\Phi}(t)\,\mathrm{d}t \tag{70}$$

[a] The symbol ξ is used instead of the more common λ to avoid confusion with adjoint variables.

is nonsingular, the least-squares estimate is uniquely defined as

$$\boldsymbol{\theta}_p^\star = \boldsymbol{G}(T_o)^{-1} \left(\int_0^{T_o} \boldsymbol{\Phi}(t)^\top \boldsymbol{y}(t)\, \mathrm{d}t \right). \tag{71}$$

The invertibility (or conditioning) of $\boldsymbol{G}(T_o)$ quantifies *identifiability*. If $\boldsymbol{G}(T_o)$ is singular, then distinct parameter vectors produce indistinguishable effects along the observed trajectory and the identification problem is ill-posed. Conversely, if $\boldsymbol{G}(T_o)$ is uniformly positive definite over time windows, i.e. there exist constants $\alpha > 0$ and $T_o > 0$ such that

$$\boldsymbol{G}(T_o) \succeq \alpha \boldsymbol{I},$$

then the data are sufficiently informative to identify $\boldsymbol{\theta}_p$. This condition is closely related to the classical notion of *persistent excitation* (Åström and Wittenmark, 1994): the trajectory must excite enough independent directions of the regressor $\boldsymbol{\Phi}(t)$ for the parameters to be observable.

In the fully coupled setting of this section, the state trajectory $\boldsymbol{s}_\circ(t)$ depends on $\boldsymbol{\theta}_p$, so the problem is not a linear regression. Nevertheless, the linear-in-parameter structure implies that, for a fixed trajectory, the objective is quadratic in $\boldsymbol{\theta}_p$ with normal equations of the form (69). The adjoint method developed earlier can then be interpreted as an efficient way of computing the gradient of this (quadratic) misfit while consistently accounting for how $\boldsymbol{\theta}_p$ perturbs the trajectory through the dynamics.

4.3 On identification challenges in nonlinear settings

In the linear-in-parameter setting, least-squares problems are well understood and have clear identifiability conditions. Nonlinear model identification, however, raises fundamental challenges that go beyond computational complexity and persist regardless of the optimization algorithm.

First, convexity is lost. Because the state trajectory $\boldsymbol{s}_\circ(t)$ depends nonlinearly on the parameters through the dynamics, the cost functional $\mathcal{J}(\boldsymbol{\theta}_p)$ is generally non-convex. Multiple local minima may therefore occur, and gradient-based methods can converge to solutions that depend strongly on the initialisation. This contrasts with the linear case, where the objective is quadratic and has a unique global minimizer when the regressor Gramian is nonsingular (Ljung, 1999; Söderström and Stoica, 1989).

Second, there is strong coupling between state and parameter errors. In nonlinear systems, parameter errors change the state trajectory, which then affects observations and the gradient used for updates. This feedback can yield poorly conditioned optimization landscapes, especially under large model mismatch. The adjoint method is crucial to maximize efficiency of the exploration but multiple restarts are usually necessary (Lions, 1971; Plessix, 2006).

Third, identifiability becomes trajectory-dependent. Unlike linear regression, where it is determined by the rank of a fixed regressor matrix, nonlinear models may reveal certain parameters only along specific trajectories or in particular regions of state space. If the system remains near an equilibrium or on a low-dimensional manifold, some parameters may be effectively unobservable. The notion of persistent excitation must therefore be viewed dynamically, as a property of the realised trajectory rather than of the model structure alone (Narendra and Annaswamy, 1989; Åström and Wittenmark, 1994).

Nonlinear dynamics are often highly sensitive to parameter perturbations: in systems with transient instabilities, bifurcations, or chaos, small changes in $\boldsymbol{\theta}_p$ can cause large deviations in the state trajectory. The adjoint variables then grow rapidly backward in time, producing noisy or numerically unstable gradients for long observation windows, as known in variational data assimilation and PDE-constrained optimization (Talagrand and Courtier, 1987). Additional issues arise when the observation map $h(\cdot)$ is nonlinear or low-dimensional. In many applications, only part of the state is observed and the state–measurement relation is nonlinear, so different parameter values can be observationally indistinguishable, hindering identifiability and often necessitating regularisation or prior information (Stengel, 1994; Law et al., 2015).

In addition, many parameters change slowly over time due to wear, aging, or environmental shifts, making the identification problem nonstationary. Long-window batch estimation may then be unsuitable, requiring a trade-off between tracking ability and noise sensitivity. This motivates online or episodic approaches with finite windows, forgetting factors, or temporal regularisation, connecting model identification to adaptive control and data assimilation methods (Åström and Wittenmark, 1994; Simon, 2006).

4.4 Identification vs Data Assimilation... and their combination

The identification problem formulated above focuses on estimating unknown *parameters* $\boldsymbol{\theta}_p$ of a dynamical model so as to improve its predictive capability. A closely related problem arises when the parameters are assumed known and the objective is instead to infer the *state* $\boldsymbol{s}_\circ$ of the system from incomplete and noisy observations. This latter task is commonly referred to as *state estimation* or *data assimilation*.

In its simplest variational form, the data assimilation problem can be posed as the optimization problem

$$\underset{\boldsymbol{s}_\circ(\cdot)}{\text{minimize}} \quad \int_0^{T_o} \ell\Big(\boldsymbol{s}_\circ(t), t\Big)\,\mathrm{d}t, \qquad \text{subject to } \dot{\boldsymbol{s}}_\circ(t) = f_\circ\Big(\boldsymbol{s}_\circ(t)\Big), \tag{72}$$

where the loss ℓ penalises mismatches between the predicted observations $h(\boldsymbol{s}_\circ(t))$ and the measured data $\boldsymbol{o}_\bullet(t)$, as in (47). In contrast to model identification, the decision variable here is the state trajectory (or, equivalently, the initial condition $\boldsymbol{s}_\circ(0)$), while the model parameters are assumed fixed.

From this perspective, the difference between identification and assimilation lies mainly in *what is inferred*: in *state estimation* (data assimilation), one seeks the state trajectory that best explains the observations for a known model; in *model identification*, one seeks parameters that make the model trajectory match the observations. Mathematically, the two problems share the same structure: a parameterised dynamical system, an observation operator, and a cost functional measuring data misfit. Thus, the same adjoint-based machinery can compute gradients in both cases.

The separation between state and parameter estimation is partly artificial. Unknown but constant (or slowly varying) parameters can be treated as additional state variables with trivial dynamics, $\dot{\boldsymbol{\theta}}_p = \mathbf{0}$ (or $\dot{\boldsymbol{\theta}}_p \approx \mathbf{0}$), yielding an *augmented-state* formulation. In this view, model identification and data assimilation differ only in which components of the augmented state are updated, and on what time scales.

Two main families of data assimilation methods are commonly used (Asch et al., 2016; Law et al., 2015; Bocquet and Farchi, 2014) and can be classified into sequential or variational methods.

Sequential methods, such as the Kalman filter and its nonlinear extensions (extended and ensemble Kalman filters), estimate the state *locally in time* by recursively combining the current model prediction with new observations (Simon, 2006). They are well suited to online and real-time applications but rely on local linearization and prescribed error statistics. Examples of these applications include the work of Ahizi et al. (2026), who employs an augmented-state extended Kalman filter for the real-time identification of heat transfer coefficients in sloshing tanks, while Schena et al. (2026) utilizes Kalman fusion to combine reduced-order models for the full-field reconstruction of wind turbine blade vibrations from sparse sensor data. Variational methods instead seek a state trajectory (or initial condition) that minimizes a cost functional over a finite time window. In this framework, adjoint equations appear as Lagrange multipliers enforcing the dynamics, leading to *four-dimensional variational data assimilation* (4D-Var) (Lions, 1971; Talagrand and Courtier, 1987). The adjoint-based formulation in this lecture belongs to this class. Unlike sequential filters, 4D-Var is *global in time*: it uses all observations in the window at once to infer the initial state (and possibly parameters), after which the model uniquely determines the evolution.

Modern digital-twin and forecasting systems often combine both approaches: sequential filters for rapid state updates, and variational or adjoint-based methods for parameter estimation and long-term consistency. This illustrates the close link between learning, identification, and assimilation, all viewed as optimization problems constrained by dynamical systems.

5 Modeling and Controlling at the same time

We now turn to the setting in which *model identification* and *control* are carried out simultaneously. This situation arises naturally in many practical applications, where a controller must act on a system whose dynamics are only partially known and may evolve over time. In such cases, the model used for control must be continuously updated from data, while the control actions themselves influence the data that become available for learning. This is the essence of *adaptive control* (Narendra and Annaswamy, 1989; Åström and Wittenmark, 1994; Ioannou and Sun, 1996). The setting is depicted in Figure 9.

This setting combines all the elements in Figures 1 and 4. The real environment (1) evolves according to unknown dynamics and generates measurements (3), while a digital twin (2) produces corresponding virtual measurements (4) based on a parameterised model. The mismatch between real and virtual observations is evaluated by a cost function (5), which drives an optimization process (6) to update the model parameters through a closure law (7), thereby improving the predictive accuracy of the digital twin. At the same time, a control agent (8) interacts with the digital twin, where full state information is available, and selects control actions according to a parameterised policy. The same action is passed to the real environment which should thus be closely aligned with the digital twin if the modeling is accurate. The resulting closed-loop behaviour is assessed by a reward function (9), and the policy parameters are updated by an optimiser (10). Con-

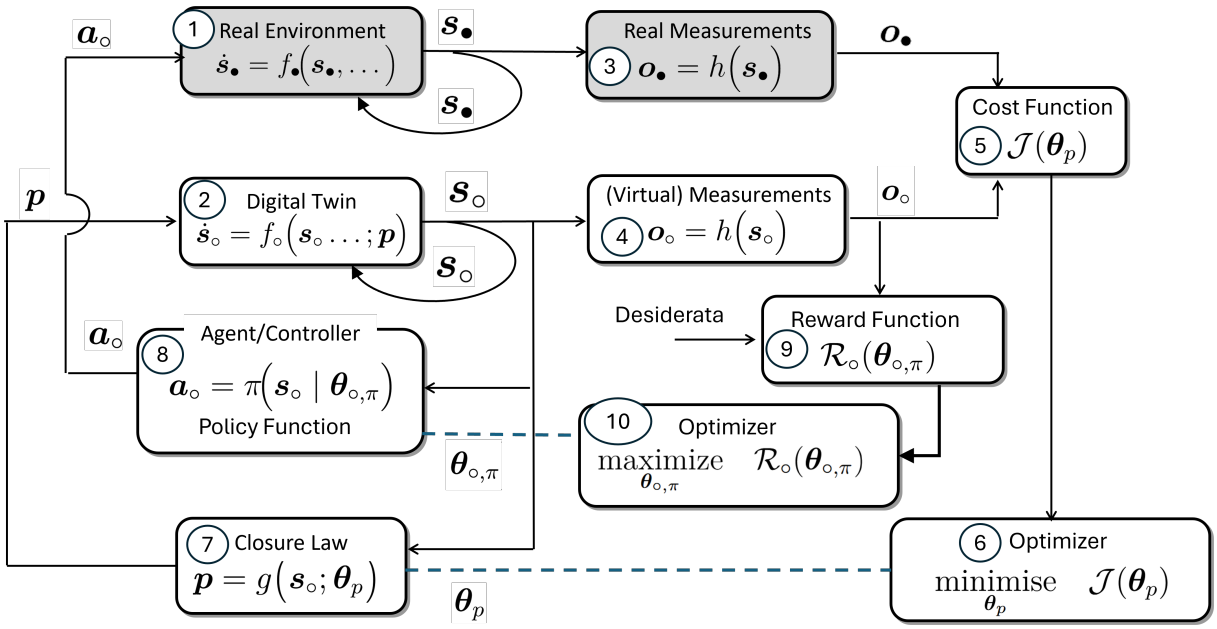


Figure 9: Schematic of model-based control with real-time model adaptation. The digital twin is continuously calibrated from measurements of the real environment, while a control policy is optimised using the adapted model. Control and identification are formulated as coupled optimization problems acting on the policy parameters $\boldsymbol{\theta}_\pi$ and model parameters $\boldsymbol{\theta}_p$, respectively.

trol and identification are therefore tightly coupled: the digital twin informs the control strategy, while the control actions influence the trajectories and data used to update the model. This joint loop enables model-based control with continuous, data-driven model adaptation.

The coupling between optimizers can be interpreted in several equivalent ways. One may view the combined problem as a single optimization problem over both the control policy and the model parameters, constrained by the system dynamics. Alternatively, one may consider a hierarchical or time-scale–separated structure, in which control is updated at a fast rate based on the current model, while model parameters are updated more slowly using accumulated data. In both cases, the adjoint-based sensitivity analysis developed in the previous sections provides a unifying computational framework for evaluating gradients efficiently.

An important consequence of this coupling is a *dual-control* effect: control actions must both achieve immediate performance and generate informative data for model identification. Aggressive regulation can suppress informative dynamics and make parameters unidentifiable, whereas exploratory actions improve the model over time but degrade control performance. Balancing these objectives is a central challenge in adaptive and learning-based control. A further difficulty, emphasised by Anderson (2008), is that identification and control cannot be combined naively without regard to robustness and safety. In closed loop, a model is validated only under the current controller and operating conditions; a model accurate for one controller may become poor once the controller changes. Aggressive controller updates based on such a model may destabilise the true system, even if the model-based design appears sound. This calls for cautious controller adapta-

tion, time-scale separation between learning and control, and mechanisms that prevent temporarily destabilising controllers. These considerations motivate *safe adaptive control*, in which stability is enforced throughout learning rather than only asymptotically.

In practice, many modern approaches address this challenge by alternating or interleaving control and identification steps, by augmenting the control objective with information-based terms, or by explicitly accounting for model uncertainty in the control design. These ideas connect classical adaptive control, reinforcement learning, and digital-twin methodologies, and will reappear in different forms throughout the remainder of the course.

5.1 Tutorial 3: Adaptive Control via Online Identification

Adaptive TSR control of a wind turbine via online identification

In this tutorial, we couple model identification and control design in a closed-loop setting. Starting from the parametric aerodynamic model introduced in Tutorial 2 (Section 4.1), we use closed-loop data to update the model parameters online, and at each stage redesign a model-based controller for power maximization. The objective is to illustrate how identification and control interact when both are embedded within the same feedback loop: improved model parameters lead to a better equilibrium operating point and controller tuning, while the controller shapes the data available for identification.

We consider the same wind turbine rotor dynamics as in Tutorial 2, described by (56), with the closure law for the power coefficient given by (58). However, instead of the $K\omega^2$ strategy (Equation (62)) used previously, we now adopt a tip-speed-ratio (TSR) reference-tracking approach. The goal is to regulate the rotor near the optimal TSR $\xi_\star$, which maximizes the power coefficient and typically provides improved transient behaviour (Stockhouse et al., 2024). In this type of control law, the reference is typically defined as

$$\omega_{\text{ref}} = \frac{\xi_\star \, \hat{v}_w}{R}, \tag{73}$$

where $\hat{v}_w$ denotes an estimated wind speed, usually provided by an observer, since direct wind-speed measurements are often difficult to obtain in practical applications (Abbas et al., 2022).

In this tutorial, however, we assume that the wind speed $v_w(t)$ is measurable at each time step and corresponds to the true (noise-free) value, so that the tip-speed ratio $\xi(t)$ can be computed directly. Under this assumption, it is convenient to express the controller directly in terms of TSR, rather than through an angular-speed reference ω_{ref}. The resulting control law reads

$$T_g = T_{\text{ff}}(v_w) - K\left(\xi - \xi_\star\right), \tag{74}$$

i.e., a TSR state-feedback term augmented with a wind-dependent feed-forward component T_{ff}. Crucially, the feed-forward torque T_{ff}, the optimal TSR $\xi_\star$ and the gain K are *model based* and therefore depend on the current model parameter estimate $\boldsymbol{\theta}_p$.

The role of T_{ff} is to provide the *equilibrium* generator torque required to operate at the optimal TSR for the current wind speed, so that the feedback term only needs to regulate deviations of $\xi(t)$ around $\xi_\star$. Starting from the aerodynamic power relation (57) and using $\xi(t) = R\omega(t)/v_w(t)$, the aerodynamic torque can be expressed as

$$T_a(t) = \frac{P_a(t)}{\omega(t)} = \frac{1}{2}\,\rho\,A\,R\,\frac{C_p\Big(\xi(t);\boldsymbol{\theta}_p\Big)}{\xi(t)}\,v_w^2(t). \tag{75}$$

Imposing operation at the model-optimal TSR $\xi_\star(\boldsymbol{\theta}_p)$, we define the feed-forward torque as the model-based aerodynamic torque evaluated at this operating condition:

$$T_{\text{ff}}\Big(v_w(t);\boldsymbol{\theta}_p\Big) \;\triangleq\; \frac{1}{2}\,\rho\,A\,R\,\frac{C_p\Big(\xi_\star(\boldsymbol{\theta}_p);\boldsymbol{\theta}_p\Big)}{\xi_\star(\boldsymbol{\theta}_p)}\,v_w^2(t). \tag{76}$$

This term compensates the dominant wind-dependent component of the torque at the desired operating point; consequently, the feedback gain K primarily shapes the transient response and robustness around $\xi_\star$.

In classical regulation problems, steady-state offsets induced by modelling errors are often mitigated by adding integral action. Here, however, we deliberately avoid an integral term: besides practical issues such as integrator windup under saturation and the associated tuning effort, integral action does not address the main limitation of this setting. Accurate operation near the maximum-C_p condition fundamentally relies on a sufficiently accurate model to compute both the equilibrium torque (76) and the optimal TSR $\xi_\star(\boldsymbol{\theta}_p)$.

Controller tuning. The feedback gain $K(\boldsymbol{\theta}_p)$ is designed via an LQR synthesis based on a local linear approximation of the nonlinear rotor dynamics around the optimal operating point $\xi_\star(\boldsymbol{\theta}_p)$. Although the plant model is naturally written in terms of the angular speed ω, we perform the linearization in tip-speed ratio (TSR) coordinates and the resulting regulation problem becomes a local setpoint-tracking problem about the constant reference $\xi_\star$.

The TSR dynamics follows by differentiating the definition $\xi(t) = R\omega(t)/v_w(t)$:

$$\dot{\xi}(t) = \frac{R}{v_w(t)}\,\dot{\omega}(t) - \frac{\xi(t)}{v_w(t)}\,\dot{v}_w(t). \tag{77}$$

For controller synthesis, we assume a constant nominal wind speed $v_w^\star$ (hence $\dot{v}_w^\star = 0$), so that the TSR dynamics reduces to

$$\dot{\xi}(t) \approx \frac{R}{v_w^\star}\,\dot{\omega}(t). \tag{78}$$

Combining this approximation with the rotor dynamics (56) yields a TSR-based model that can be linearised about the equilibrium $(\xi_\star,\, T_g^\star,\, v_w^\star)$. In particular, the equilibrium generator torque is provided by the feed-forward term,

$$T_g^\star = T_{\text{ff}}\Big(v_w^\star;\boldsymbol{\theta}_p\Big), \tag{79}$$

which corresponds to operation at $\xi_\star(\boldsymbol{\theta}_p)$. A first-order continuous-time linearization then yields

$$\delta\dot{\xi}(t) = A\,\delta\xi(t) + B\,\delta T_g(t), \tag{80}$$

where $\delta\xi(t) = \xi(t) - \xi_\star(\boldsymbol{\theta}_p)$ and $\delta T_g(t) = T_g(t) - T_g^\star$ denote deviations from the operating point.

In the actual implementation, the LQR gain is computed from a *discrete-time* linear model. Rather than discretising a continuous-time linearization, we linearise *directly* the one-step update induced by the chosen numerical integrator with sampling time Δt. Specifically, we write the closed-form one-step map in TSR coordinates as

$$\xi_{k+1} = F\Big(\xi_k,\, T_{g,k},\, v_{w,k};\boldsymbol{\theta}_p\Big), \tag{81}$$

and compute the corresponding Jacobians at the operating point using JAX automatic differentiation:

$$A_d = \left.\frac{\partial F}{\partial \xi}\right|_\star, \qquad B_d = \left.\frac{\partial F}{\partial T_g}\right|_\star, \tag{82}$$

where $(\cdot)|_\star$ denotes evaluation at $(\xi_\star(\boldsymbol{\theta}_p),\, T_g^\star,\, v_w^\star)$. The resultant reduced linear model reads

$$\delta\xi_{k+1} = A_d\,\delta\xi_k + B_d\,\delta T_{g,k}. \tag{83}$$

Finally, the optimal gain K is obtained from the standard LQR solution via the discrete-time algebraic Riccati equation (DARE) for the infinite-horizon quadratic cost, by using the reward–maximization convention of Section 2, which is equivalent to minimising the cost

$$\mathcal{J} = -\mathcal{R}_\circ = \sum_{k=0}^{\infty}\Big(Q\,(\xi_k - \xi_\star)^2 + R\,(T_{g,k} - T_g^\star)^2\Big), \tag{84}$$

where $Q > 0$ penalises TSR-tracking errors and $R > 0$ penalises control effort. In this tutorial, we choose

$$Q = \frac{1}{\xi_\star^2}, \qquad R = \frac{1}{2\,(T_g^\star)^2}, \tag{85}$$

i.e., we scale the weights by the squared nominal magnitudes of the state and input. With this choice, the state-regulation term is weighted approximately twice as much as the control-effort term in the normalised (dimensionless) cost.

Scope of the exercise. The goal of this tutorial is to couple identification and control. Starting from the identification procedure derived in Tutorial 2, we use closed-loop data to update the parameter vector $\boldsymbol{\theta}_p$. The updated model is then used to redesign the model-based power-maximizing control law (74). In this setting, identification and control form a feedback loop: improved parameters yield a more accurate equilibrium/feed-forward torque and TSR setpoint, while the controller keeps the system near the operating region of interest and provides persistently informative data for identification. Iterating this process progressively improves model fidelity and closed-loop performance around the target operating point.

Implementation and Case Study The numerical implementation follows a windowed closed-loop procedure. In contrast to Tutorial 2, we consider a single continuous simulation time series, mimicking real-time operation. The total simulation time is set to $T_{\text{sim}} = 300\,\text{s}$ and is divided into 10 consecutive windows of duration $T_w = 30\,\text{s}$ each, with sampling time $\Delta t = 0.05\,\text{s}$. Each window defines an *episode*.

We first generate the full wind-speed time series over $[0, T_{\text{sim}}]$ using (61), with $\Delta t = 0.05\,\text{s}$, mean wind speed $\mu_w = 8.4\,\text{m/s}$, and hyperparameters $[\sigma_{f1}, \sigma_{f2}, \sigma_n] = [1,\, 0.4,\, 0.1]\,\text{m}^2/\text{s}^2$ and $[\ell_{T,1}, \ell_{T,2}] = [5,\, 0.4]\,\text{s}$ (same conditions as in Tutorial 2).

We start from a nominal initial estimate of the power-coefficient model parameters,

$$\boldsymbol{\theta}_p^{(0)} = \left(c_1^{(0)},\, c_2^{(0)},\, c_3^{(0)},\, c_4^{(0)}\right)^\top = [0.22,\, 116,\, 9,\, 12.5]^\top, \tag{86}$$

which differs slightly from the initial estimate reported in (59), in order to deliberately exaggerate the initial modelling error and thus shift the predicted optimum $\xi_\star\left(\boldsymbol{\theta}^{(0)}\right)$ away from its true value.

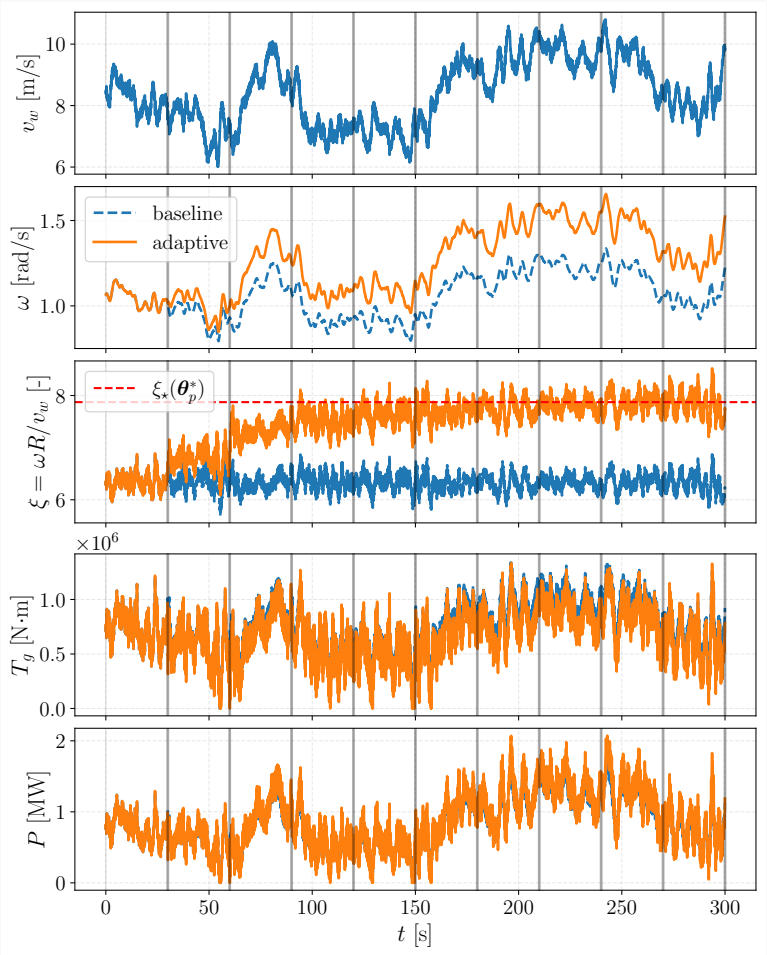


Figure 10: Closed-loop comparison between the adaptive identify–control loop (blue) and a fixed baseline controller fixed with the nominal model parameters $\boldsymbol{\theta}_{\boldsymbol{p}}^{(\mathbf{0})}$ (orange) over the full simulation horizon. From top to bottom: wind speed $v_w(t)$, rotor speed $\omega(t)$, tip-speed ratio $\xi(t) = \omega(t)R/v_w(t)$ (with the reference $\xi_\star$), commanded generator torque $T_g(t)$, and generated power $P(t)$. Vertical lines indicate the identification windows. The adaptive strategy progressively regulates $\xi(t)$ closer to $\xi_\star$ and yields higher power extraction than the baseline controller.

At the beginning of each window w, the controller is synthesized using the *current* model parameters $\boldsymbol{\theta}_p^{(w)}$. First, the optimal TSR $\xi_\star(\boldsymbol{\theta}_p^{(w)})$ is computed by maximizing the current $C_p(\xi; \boldsymbol{\theta}_p^{(w)})$. Then, the LQR gain $K(\boldsymbol{\theta}_p^{(w)})$ is obtained by linearizing the one-step discrete update (RK4) around a nominal operating point associated with $v_w^\star = 8.4\,\mathrm{m/s}$, and by solving the discrete Riccati equation with weights (Q, R). The resulting controller is applied in closed loop to the *plant* model over the full window, using the measured wind speed $v_w(t)$ to compute the feedforward term in (74).

At the end of each window, the model parameters are updated by minimising a simulation-error cost over the trajectory collected during that episode, thereby making the identification procedure *online*. We use the same objective introduced in (60), but restricted to a single window:

$$\underset{\boldsymbol{\theta}_p}{\text{minimize}} \quad \mathcal{J}(\boldsymbol{\theta}_p) = \frac{1}{2} \int_{w\,T_w}^{(w+1)\,T_w} \Big(\omega_{\text{sim}}(t; \boldsymbol{\theta}_p) - \omega_{\text{meas}}(t) \Big)^2 \, \mathrm{d}t. \tag{87}$$

The optimization is performed with ADAM for 300 iterations for each window. To improve numerical conditioning, we introduce scaled (dimensionless) optimization variables x such that

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{ref}} + \text{diag}(\boldsymbol{s})\, x, \tag{88}$$

where $\boldsymbol{\theta}_{\text{ref}}$ is a reference parameter vector and $\boldsymbol{s}$ is a user-defined vector of characteristic parameter scales. In the Adam iterations, we also apply the same diagonal scaling as a per-parameter step preconditioner, in order to balance update magnitudes across parameters with different physical units.

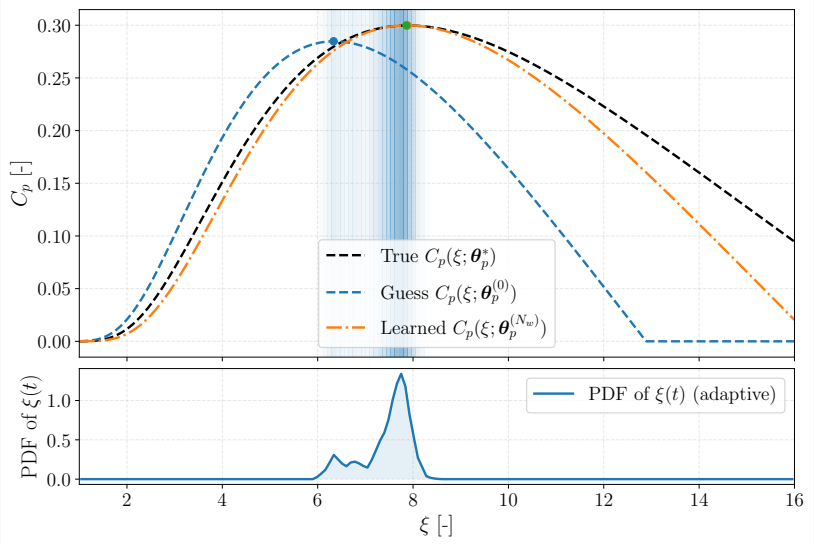


Figure 11: Power-coefficient curves and closed-loop operating region. Top: parametric $C_p(\xi;\boldsymbol{\theta}_p)$ computed using the nominal (initial), true, and final identified parameter vectors; markers indicate the corresponding maxima $\xi_\star$. The shaded vertical band encodes the empirical distribution of the tip-speed ratio visited in closed loop (darker bands correspond to higher probability density). Bottom: probability density function of the measured $\xi(t) = \omega(t)R/v_w(t)$ over the full time series, showing that the controller operates predominantly near $\xi_\star$ after adaptation.

Results Figure 10 summarises the closed-loop simulation over the full horizon $T_{\text{sim}} = 300\,\text{s}$ for both the baseline controller (orange, fixed nominal model) and the adaptive controller (blue, model updated at the end of each window). The simulation is partitioned into consecutive identification/control windows, whose boundaries are indicated by the vertical grey lines.

Both controllers are driven by the same wind realisation $v_w(t)$ (top panel). During the first window, the baseline and adaptive responses coincide, since the adaptive controller is initialised with the same nominal parameter vector $\boldsymbol{\theta}_p^{(0)}$ and therefore uses the same model-based feed-forward term and the same feedback gain. From the second window onward, the trajectories diverge: the adaptive controller is re-synthesised using the updated parameter estimate, whereas the baseline controller keeps the initial (incorrect) model.

This effect is most clearly visible in the TSR panel, where the true optimal TSR $\xi_\star(\boldsymbol{\theta}_p^\star)$ is indicated by the red dashed line. While the baseline controller regulates around an incorrect TSR level (consistent with the wrong initial model), the adaptive controller progressively shifts the operating TSR as the model improves. In the first few windows, the adaptive TSR is still biased because $\xi_\star(\boldsymbol{\theta}_p)$ is computed from a parameter estimate that has not converged yet; however, as identification proceeds, the adaptive TSR approaches the true optimum. In this test case, good agreement with $\xi_\star(\boldsymbol{\theta}_p^\star)$ is achieved after roughly five windows, after which the adaptive controller remains close to the true optimal TSR for the remainder of the simulation. This improved TSR regulation translates into higher power extraction than the baseline controller under the same wind forcing.

Figure 11 provides a complementary view in terms of the power coefficient map and the induced TSR statistics. The top panel compares: (i) the nominal $C_p(\xi;\boldsymbol{\theta}_p^{(0)})$ curve used by the baseline controller, (ii) the true $C_p(\xi;\boldsymbol{\theta}_p^\star)$ curve, and (iii) the final $C_p(\xi;\boldsymbol{\theta}_p^{(N_w)})$ obtained after the last window. The nominal curve peaks at a TSR that is noticeably shifted with respect to the true optimum, whereas the final identified curve closely matches the true curve around its maximum, implying that the identified model recovers both the location of the optimum and the local curvature relevant for feedback design close to the operating point $\xi_\star$.

The same figure also reports the distribution of the TSR achieved by the adaptive controller (bottom panel). The probability mass concentrates around the true optimal TSR, confirming the behaviour observed in Figure 10: after a few windows, the closed-loop operation spends most of the time in the vicinity of the maximum-C_p region. The residual probability away from the optimum is primarily associated with the initial window, in which the controller is still based on the nominal (mismatched) parameter set; accordingly, the TSR during the first episode is biased and contributes a lower-

probability tail in the distribution.

Comparing with Figure 7 from Tutorial 2, where a less aggressive control law was used, we observe that the standard deviation of the TSR distribution is markedly smaller in the present tutorial. This is consistent with the more aggressive feedback action, which keeps the operating TSR closer to the desired value, except for the initial deviations caused by the deliberately mismatched nominal parameters (as discussed above). At the same time, these results highlight an important limitation of closed-loop identification: the identification–control loop is most effective *locally*, i.e., in the vicinity of the operating point that the controller regulates. In comparison with Figure 7, the final identified $C_p(\xi)$ curve matches the true curve very well near the optimal TSR, but it does not necessarily remain accurate across the entire TSR range. As the controller becomes progressively better at regulation, the closed-loop trajectories concentrate around $\xi_\star$, which reduces excitation and therefore the amount of informative data away from the operating point. Consequently, the identified model tends to be accurate near $\xi_\star$ but may degrade at low and high TSR values that are seldom visited during closed-loop operation.

6 The Model-Free Control Problem

Model-free control rests on a simple observation: in many practical settings, no reliable mathematical model of the system is available, or existing models are too expensive to support real-time decision making or repeated policy optimization. This situation is common in industrial contexts where systems can be queried and measured in real time and candidate control laws can be tested online, while the underlying physics—particularly in fluid mechanics—are so complex that high-fidelity simulations are far more costly than direct interaction with the real system.

One may therefore think in terms of a simulation-to-interaction ratio: when simulating the system is significantly slower than running experiments or collecting data online, it becomes natural to bypass modelling altogether and instead rely on direct interaction. Control policies are then improved by "trying things out," observing the outcome, and iteratively updating the policy based solely on measured performance. This philosophy underlies model-free control and reinforcement learning approaches (Brunton and Noack, 2015; Duriez et al., 2017; Pino et al., 2023).

Of course, such exploration cannot be carried out blindly. In realistic settings, experimentation must remain safe: physical constraints, operational limits, and the risk of irreversible damage severely restrict which actions can be tested on the real system. Abandoning an explicit dynamical model also removes much of the analytical structure of model-based control, including stability guarantees and optimality conditions. Model-free methods therefore trade analytical transparency for practical flexibility, relying on constrained experimentation, performance evaluation, and iterative improvement. The setting investigated in this section is described in Figure 12.

The real environment (1) is treated as an opaque system whose internal dynamics are unknown. Measurements (2) are collected and used to evaluate performance through a reward function (3). This reward drives an optimiser (4), which updates the parameters

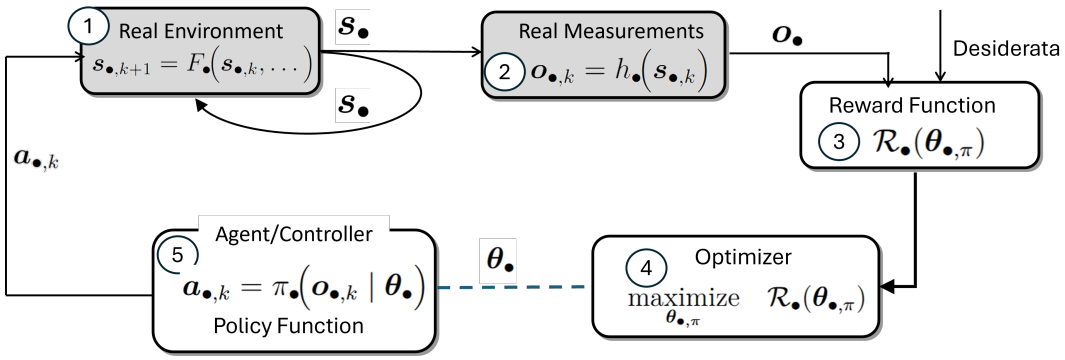


Figure 12: Real-time system identification setting. The physical environment evolves according to unknown dynamics and is observed through real measurements. In parallel, a digital twin with tunable parameters runs alongside the real system. A cost function quantifies the mismatch between real and virtual measurements, and an optimizer updates the closure-law parameters so as to continuously adapt the digital twin to the observed system behavior.

of an agent or controller (5) interacting directly with the environment. No explicit model of the system dynamics is constructed; the control law is synthesised purely through trial, evaluation, and adaptation.

In this section, we adopt a discrete-time formulation similar to eq. (4). This choice is purely pragmatic: in a model-free setting, neither control synthesis nor control deployment requires the numerical integration of the (unknown) system dynamics– learning and decision making operate directly on observed sequences of measurements and actions.

We denote by $\boldsymbol{s}_{\bullet,k} \approx \boldsymbol{s}_\bullet(t_k)$ the discrete-time state, generally not available, by $\boldsymbol{o}_{\bullet,k}$ the corresponding observation, and by $\boldsymbol{a}_{\bullet,k} \approx \boldsymbol{a}_\bullet(t_k)$ the applied control action. The exogenous input collecting disturbances or operating conditions is denoted by $\boldsymbol{z}_k \approx \boldsymbol{z}(t_k)$. The closed-loop system is thus

$$\begin{cases} \boldsymbol{s}_{\bullet,k+1} = F_\bullet\big(\boldsymbol{s}_{\bullet,k},\, \boldsymbol{z}_k,\, \boldsymbol{a}_{\bullet,k}\big), \\ \boldsymbol{o}_{\bullet,k} = h_\bullet\big(\boldsymbol{s}_{\bullet,k}\big), \\ \boldsymbol{a}_{\bullet,k} = \pi_\bullet\big(\boldsymbol{o}_{\bullet,k} \mid \boldsymbol{\theta}_\bullet\big), \\ \boldsymbol{s}_{\bullet,0} = \boldsymbol{s}_{\bullet,0}. \end{cases} \tag{89}$$

Here, $F_\bullet$ should not be interpreted as a dynamical model or state-transition map. It merely represents a black-box data-generating mechanism associated with the interaction between the controller and the system, with no assumed structure, regularity, or physical interpretation exploited in the control design. For the purposes of this section, the mapping $F_\bullet$ could equivalently be written directly in terms of observations, thereby incorporating the observation operator $h_\bullet$ and avoiding any explicit reference to the latent state $\boldsymbol{s}_{\bullet,k}$, which is neither observed nor used by the controller. We retain the state notation here only to facilitate later extensions of the framework, discussed in Lecture 13.

The policy parameters $\boldsymbol{\theta}_\bullet$ are identified by maximizing a cumulative reward similar to

(5), with the expectation taken over possible initial states and disturbances:

$$\underset{\boldsymbol{\theta}_\bullet}{\text{maximize}} \quad \mathcal{R}_\bullet(\boldsymbol{\theta}_\bullet) = \mathbb{E}_{\boldsymbol{s}_{\bullet,0}\sim\mathcal{I}_0,\, \boldsymbol{z}(\cdot)\sim\mathcal{P}_z}\left[\sum_{k=0}^{N-1} r_{\bullet,\pi}\Big(\boldsymbol{s}_{\bullet,k}(\boldsymbol{\theta}_\bullet), \boldsymbol{a}_{\bullet,k}(\boldsymbol{\theta}_\bullet)\Big)\right]. \tag{90}$$

In the case of stochastic policies, the expectation is additionally taken over the distribution of actions induced by the policy. Furthermore, in many reinforcement learning formulations a discount factor $\gamma \in [0, 1)$ is introduced in the summation of (90), replacing $r_{\bullet,\pi}$ by $\gamma^k r_{\bullet,\pi}$ so that future rewards are progressively downweighted. This is particularly important in infinite-horizon settings to ensure convergence of the return and to encode a preference for immediate rewards over distant ones. In the present lecture, however, we consider a finite-horizon episodic problem with fixed N. Since the cumulative reward is bounded by construction and no asymptotic behaviour is analyzed, the introduction of discounting is not essential. We therefore set $\gamma = 1$ and optimise the undiscounted return; the extension to $\gamma < 1$ follows directly from the same formulation.

Note that the policy parameters in (89) and (90) are denoted as $\boldsymbol{\theta}_\bullet$ and not $\boldsymbol{\theta}_{\bullet,\pi}$ as the policy need not be expressed explicitly as a parametric mapping from states (or observations) to actions. Within this broad class of methods, a natural classification arises based on whether the policy is learned *directly* or *indirectly*.

In *direct (policy-based) methods*, the control law is parametrized as a mapping of observations $\boldsymbol{a}_k = \pi_\bullet(\boldsymbol{s}_k \mid \boldsymbol{\theta}_{\bullet,\pi})$ similarly to the model based control framework introduced earlier. These approaches correspond to what is commonly referred to as *policy-based reinforcement learning*, where learning acts directly on the policy parameters.

In indirect methods, also known as *value-based* methods, the policy is not parametrized. Instead, one learns a surrogate function—typically a state–action value (or Q) function—that estimates the expected return (90) associated with taking a given action in a given state. The parameters $\boldsymbol{\theta}_\bullet$ then denote the parameters for this surrogate function. In indirect methods, the control law then emerges implicitly by selecting, at each decision step, the action that maximizes this learned value function. In this setting, learning targets the evaluation of actions rather than the policy mapping itself[9].

Both classes fall under the umbrella of *reinforcement learning*, and many algorithms exist within each category. Modern approaches often combine the two viewpoints, as in actor–critic methods. In these notes, we highlight one representative approach for each class in the following sections.

The field of model-free reinforcement learning is vast and rapidly evolving, and the material covered here represents only a small subset of the available literature. For further reading, we refer to Rabault and Kuhnle (2023) for an accessible introduction, to Bertsekas (2024); Khan et al. (2012); Lewis and Vrabie (2009); Recht (2019); Semeraro (2025) for deeper perspectives on the connections between optimal control and reinforcement learning, and to the classic textbooks (Sutton and Barto, 2018; Bertsekas, 2012).

[9]The lack of a direct policy and the need for an optimization to drive the action selection is reminiscent of model predictive control (MPC), discussed the Lecture by Prof. Discetti. Value-based methods can be interpreted as replacing this repeated online optimization with a learned surrogate of the long-term return: the optimization is performed implicitly during training, and the resulting value (or Q) function enables fast action selection at runtime via a simple maximization. In this sense, value-based model-free control trades online optimization for offline (or incremental) learning, while retaining the ability to select actions based on predicted future performance.

6.1 Policy search via Bayesian Optimization

Policy search via Bayesian Optimization (BO) formulates control design as a data-efficient black-box optimization problem, using a probabilistic surrogate to balance exploration and exploitation (Müller et al., 2021). The goal is here to find the parameters $\boldsymbol{\theta}_{\bullet,\pi} \in \mathbb{R}^{n_\pi}$ for the parametric policy $\boldsymbol{a}_k = \pi_\bullet(\boldsymbol{s}_k \mid \boldsymbol{\theta}_{\bullet,\pi})$. In the following, since all the focus is placed on model free optimization, we simplify the notation $\boldsymbol{\theta}_{\bullet,\pi}$ to $\boldsymbol{\theta}$.

BO is particularly effective when policy evaluations are expensive, because each new trial is chosen to maximize a principled acquisition criterion—typically trading predicted improvement against uncertainty reduction (Frazier, 2018; Candelieri, 2021). A key limitation is scalability with dimension: BO tends to degrade when $n_{\theta,\pi}$ is large, since surrogate fitting and acquisition optimization become difficult and the number of evaluations required to explore the parameter space grows rapidly (a manifestation of the curse of dimensionality). Consequently, BO is most effective for low- to moderate-dimensional policy parametrisations (up to $n_{\theta,\pi} \lesssim 20$), such as structured controllers or low-dimensional gain vectors. In control applications, BO has been used for automatic tuning of feedback laws and, more recently, in combination with digital twins to reduce reliance on real-system evaluations (Richter et al., 2025; Nobar et al., 2024).

To guide the search, BO constructs a probabilistic surrogate model of the (unknown) reward function $\mathcal{R}_\bullet(\boldsymbol{\theta}_\bullet) : \mathbb{R}^{n_{\theta,\pi}} \to \mathbb{R}$ based on a dataset of past evaluations $\mathcal{D}_n = \{(\boldsymbol{\theta}^{(i)}, \mathcal{R}^{(i)})\}_{i=1}^n$. This surrogate function is modelled as a Gaussian Process (GP, Rasmussen and Williams (2006)),

$$\mathcal{R}_\bullet(\boldsymbol{\theta}_\bullet) \sim \mathcal{GP}\Big(m((\boldsymbol{\theta}_\bullet)),\, \kappa(\boldsymbol{\theta}_{\bullet,1}, \boldsymbol{\theta}_{\bullet,2})\Big),$$

with mean function $m(\cdot)$ and covariance function $\kappa(\cdot,\cdot)$. This means that for any finite collection of inputs, the corresponding function values follow a multivariate Gaussian distribution

$$\boldsymbol{r} = \Big[f(\boldsymbol{\theta}^{(1)}),\ \cdots\ f(\boldsymbol{\theta}^{(n)})\Big]^\top \sim \mathcal{N}\Big(\boldsymbol{m},\, \mathbf{K}\Big), \qquad \mathbf{K}_{ij} = \kappa(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)}). \tag{91}$$

Here $\mathcal{N}(\boldsymbol{m}, \mathbf{K})$ denotes a multivariate normal distribution with mean vector $\boldsymbol{m} = [m(\boldsymbol{\theta}^{(1)}),\ \ldots,\ m(\boldsymbol{\theta}^{(n)})]^\top$ while the covariance matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ evaluated on the training data is defined by the kernel function $\kappa(\cdot,\cdot)$, with entries $\mathbf{K}_{ij} = \kappa(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)})$.

Given the prior GP model and the noisy observations $\mathcal{D}_n = \{(\boldsymbol{\theta}^{(i)}, \mathcal{R}^{(i)})\}_{i=1}^n$, Bayesian inference yields a posterior distribution over the latent function f conditioned on the data. This conditioning step is the core of Gaussian Process regression.

Let $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(n)}] \in \mathbb{R}^{n_\pi \times n}$ denote the matrix of training inputs and $\boldsymbol{r} = [\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(n)}]^\top$ the corresponding observed rewards. Under the Gaussian noise model, the joint distribution of the observed rewards and the latent function value at a new query point $\boldsymbol{\theta}_*$ is the multivariate normal,

$$\begin{bmatrix} \boldsymbol{r} \\ \mathcal{R}_{\bullet,\pi}(\boldsymbol{\theta}_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m} \\ m(\boldsymbol{\theta}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \boldsymbol{I} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right), \tag{92}$$

where the covariance blocks are defined as follows:

- $\mathbf{K} = \kappa(\boldsymbol{\Theta}, \boldsymbol{\Theta}) \in \mathbb{R}^{n\times n}$ is the *training covariance matrix*, whose entries $\mathbf{K}_{ij} = \kappa(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)})$ encode pairwise correlations between reward evaluations at the training policy parameters.

- $\mathbf{K}_* = \kappa(\boldsymbol{\Theta}, \boldsymbol{\theta}_*) \in \mathbb{R}^{n\times 1}$ is the *cross-covariance vector* between the training inputs and the new query point $\boldsymbol{\theta}_*$, with entries $(\mathbf{K}_*)_i = \kappa(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}_*)$.
- $\mathbf{K}_{**} = \kappa(\boldsymbol{\theta}_*, \boldsymbol{\theta}_*) \in \mathbb{R}$ is the *prior variance* of the latent function evaluated at the query point.
- $\sigma_n^2 \boldsymbol{I} \in \mathbb{R}^{n\times n}$ accounts for independent, identically distributed Gaussian observation noise affecting the measured rewards.

Conditioning the joint Gaussian distribution on the observed data yields the posterior predictive distribution at the query point $\boldsymbol{\theta}_*$

$$f(\boldsymbol{\theta}_*) \mid \mathcal{D}_n \sim \mathcal{N}\Big(\mu_*(\boldsymbol{\theta}_*),\, \sigma_*^2(\boldsymbol{\theta}_*)\Big), \tag{93}$$

with posterior mean and variance given by

$$\mu_*(\boldsymbol{\theta}_*) = m(\boldsymbol{\theta}_*) + \mathbf{K}_*^\top \Big(\mathbf{K} + \sigma_n^2 \boldsymbol{I}\Big)^{-1} \Big(\boldsymbol{r} - \boldsymbol{m}\Big), \tag{94}$$

$$\sigma_*^2(\boldsymbol{\theta}_*) = \mathbf{K}_{**} - \mathbf{K}_*^\top \Big(\mathbf{K} + \sigma_n^2 \boldsymbol{I}\Big)^{-1} \mathbf{K}_*. \tag{95}$$

These expressions follow directly from the conditioning properties of joint multivariate Gaussian distributions; see Appendices A.2–A.3 of Rasmussen and Williams (2006) for a detailed derivation.

The mean function of the prior is usually set to zero, so that the central role in the Gaussian process modelling is the kernel function. This encodes prior assumptions on smoothness and correlation of the unknown reward function over the policy-parameter space, so that nearby parameter vectors are expected to yield similar rewards.

In these notes, we primarily employ the Matérn kernel, which offers a flexible trade-off between smoothness and expressiveness and is widely used in Bayesian Optimization. The Matérn kernel is defined as

$$\kappa_{\text{Matérn}}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\, r}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}\, r}{\ell}\right), \qquad r = \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2, \tag{96}$$

where σ_f^2 is the signal variance, $\ell > 0$ is a characteristic length-scale, $\nu > 0$ is a smoothness parameter, $\Gamma(\cdot)$ denotes the Gamma function, and $K_\nu(\cdot)$ is the modified Bessel function of the second kind[10].

The parameter ℓ controls how rapidly correlations decay with distance in parameter space, while σ_f^2 sets the overall scale of reward variations. The smoothness parameter ν governs the regularity of sample functions drawn from the GP: smaller values of ν allow for rougher functions, whereas larger values recover smoother behaviour. Notably, as $\nu \to \infty$, the Matérn kernel converges to the squared-exponential (RBF) kernel, which assumes infinitely differentiable functions. In practice, fixed values such as $\nu \in \{3/2, 5/2\}$ are commonly used to balance flexibility and numerical robustness.

The kernel hyperparameters, together with the observation noise variance σ_n^2, hyperparameters, which can be collected in a vector $\boldsymbol{\eta} = (\sigma_f^2, \ell, \nu, \sigma_n^2)$. Although prior knowledge

[10]The Gamma function is defined as $\Gamma(\nu) = \int_0^\infty t^{\nu-1} e^{-t}\, \mathrm{d}t$, while the modified Bessel function of the second kind admits the integral representation $K_\nu(x) = \frac{1}{2}(\frac{x}{2})^\nu \int_0^\infty t^{-\nu-1} \exp(-t - x^2/(4t))\, \mathrm{d}t$.

of the problem (and user experience!) can allow for a reasonable guess for these parameters, these can also be easily inferred from data by maximising the log marginal likelihood of the observed rewards under the GP model.

For a dataset $\mathcal{D}_n = \{(\boldsymbol{\theta}^{(i)}, \mathcal{R}^{(i)})\}$, the log-marginal likelihood reads

$$\log p(\boldsymbol{r} \mid \boldsymbol{\Theta}, \boldsymbol{\eta}) = -\frac{1}{2}\boldsymbol{r}^\top \left(\mathbf{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{r} - \frac{1}{2}\log\left|\mathbf{K} + \sigma_n^2 \boldsymbol{I}\right| - \frac{n}{2}\log(2\pi), \tag{97}$$

where $\mathbf{K} = \kappa(\boldsymbol{\Theta}, \boldsymbol{\Theta})$ depends on the kernel hyperparameters and $|\cdot|$ denotes the determinant of a matrix.

The first term measures data fit, penalising models that explain the observed rewards poorly. The second term acts as a complexity penalty, discouraging overly flexible models that would fit the data only by inflating uncertainty. The final term is a normalisation constant. Maximising the log marginal likelihood thus naturally balances fidelity to the observed rewards against model complexity, providing an automatic and principled mechanism for tuning the GP hyperparameters (Rasmussen and Williams, 2006). Once a Gaussian Process surrogate has been fitted, Bayesian Optimization needs a rule to choose the next sample in the policy-parameter space: this is the role of the *acquisition function*. It assigns a scalar utility to each candidate $\boldsymbol{\theta}$ from the current GP posterior, measuring how informative or promising a new evaluation there would be.

The acquisition function balances *exploitation*, favouring regions with high predicted reward, and *exploration*, favouring regions with high predictive uncertainty. Because it is cheap to evaluate and typically differentiable, it can be maximized efficiently via numerical optimization, even when the true reward is expensive to evaluate.

Among many proposed acquisition strategies, *Expected Improvement* (EI) is one of the most widely used and also employed in the proposed exercises. For maximization, EI is the expected increase in reward obtained by sampling at $\boldsymbol{\theta}$ over the best reward observed so far, $r_{\text{best}} = \max_{i \le n} r_i$, under the current GP posterior.

Let $\mu_n(\boldsymbol{\theta})$ and $\sigma_n(\boldsymbol{\theta})$ denote the posterior mean and standard deviation of the GP at $\boldsymbol{\theta}$. The expected improvement is defined as

$$\mathrm{EI}_n(\boldsymbol{\theta}) = \begin{cases} \left(\mu_n(\boldsymbol{\theta}) - r_{\text{best}}\right)\Phi(Z) + \sigma_n(\boldsymbol{\theta})\,\phi(Z), & \sigma_n(\boldsymbol{\theta}) > 0, \\ 0, & \sigma_n(\boldsymbol{\theta}) = 0, \end{cases} \tag{98}$$

where

$$Z = \frac{\mu_n(\boldsymbol{\theta}) - r_{\text{best}} - \xi}{\sigma_n(\boldsymbol{\theta})},$$

and $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative distribution function and probability density function of the standard normal distribution, respectively. The parameter $\xi \ge 0$ explicitly controls the exploration–exploitation trade-off, with larger values encouraging exploration.

To conclude, algorithm 2 illustrates the full Bayesian Optimization algorithm.

Practical numerical aspects and scalability. A practical limitation of Gaussian-process-based Bayesian Optimization is its computational cost. At iteration n, assuming that n samples have been collected (i.e. no initialization), evaluating the GP posterior and the log marginal likelihood requires manipulating the covariance matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$,

Algorithm 2: Bayesian Optimization for Black-Box Policy Search

Input: Initial dataset $\mathcal{D}_{n_0} = \{(\boldsymbol{\theta}_i, r_i)\}_{i=1}^{n_0}$, evaluation budget $N > n_0$

Initialise $n \leftarrow n_0$

while $n < N$ **do**

  Hyperparameter update: Fit the GP hyperparameters $\boldsymbol{\eta}$ by maximising the marginal likelihood (equivalently, minimising the negative log marginal likelihood), cf. (97)

  Posterior update: Compute the GP posterior $p(f(\boldsymbol{\theta}) \mid \mathcal{D}_n)$ using the conditioning formulas (94)

  Acquisition maximization: Select the next policy parameters by maximising the acquisition function, for example the expected improvement, $\boldsymbol{\theta}_{n+1} = \arg\max_{\boldsymbol{\theta}} \mathrm{EI}_n(\boldsymbol{\theta})$, with EI_n defined in (98)

  Black-box evaluation: Roll out the policy with parameters $\boldsymbol{\theta}_{n+1}$ on the real system and observe the reward $r_{n+1} = \mathcal{R}_\bullet(\boldsymbol{\theta}_{n+1})$

  Dataset augmentation: $\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n \cup \{(\boldsymbol{\theta}_{n+1}, r_{n+1})\}$

  $n \leftarrow n + 1$

return $\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}_i \in \mathcal{D}_N} r_i$

which leads to a computational complexity $\mathcal{O}(n^3)$ and a memory cost $\mathcal{O}(n^2)$. This cost rapidly becomes dominant as the number of policy evaluations increases and is one of the main reasons why BO is typically restricted to a few tens or hundreds of samples.

In practice, explicit matrix inversion is never performed. Instead, numerical implementations rely on a Cholesky factorisation $\mathbf{K}+\sigma_n^2\boldsymbol{I} = \boldsymbol{L}\boldsymbol{L}^\top$, where $\boldsymbol{L}$ is a lower triangular matrix. This factorisation allows both the evaluation of the log marginal likelihood and the solution of linear systems of the form $(\mathbf{K}+\sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{r}$ in a numerically stable way, using forward and backward substitutions. Moreover, the log-determinant term appearing in the marginal likelihood is then obtained cheaply as

$$\log\left|\mathbf{K} + \sigma_n^2\boldsymbol{I}\right| = 2\sum_{i=1}^{n} \log \boldsymbol{L}_{ii}.$$

Since Bayesian Optimization proceeds sequentially, one new data point is added at each iteration. Rather than recomputing the Cholesky factorisation from scratch, it is possible to exploit the block structure of the augmented covariance matrix (see Rasmussen and Williams (2006) and Golub and Van Loan (2013))

$$\mathbf{K}_{n+1} = \begin{bmatrix} \mathbf{K}_n & \boldsymbol{k}_* \\ \boldsymbol{k}_*^\top & k_{**} \end{bmatrix}, \tag{99}$$

where $\boldsymbol{k}_* = \kappa(\boldsymbol{\Theta}_n, \boldsymbol{\theta}_{n+1})$ and $k_{**} = \kappa(\boldsymbol{\theta}_{n+1}, \boldsymbol{\theta}_{n+1}) + \sigma_n^2$. The Cholesky factor can then be updated using a rank-one (or block) update, reducing the cost of each iteration to $\mathcal{O}(n^2)$.

More specifically, given the Cholesky factorisation of the current kernel matrix $\mathbf{K}_n = \mathbf{L}_n\mathbf{L}_n^\top$, with $\mathbf{L}_n \in \mathbb{R}^{n\times n}$ lower triangular, we seek a Cholesky factorisation of the augmented matrix $\mathbf{K}_{n+1}$ of the form

$$\mathbf{L}_{n+1} = \begin{bmatrix} \mathbf{L}_n & \mathbf{0} \\ \boldsymbol{v}^\top & \alpha \end{bmatrix}, \tag{100}$$

such that $\mathbf{K}_{n+1} = \mathbf{L}_{n+1}\mathbf{L}_{n+1}^{\top}$.

The vector $\boldsymbol{v} \in \mathbb{R}^n$ is obtained by solving the triangular system

$$\mathbf{L}_n \, \boldsymbol{v} = \boldsymbol{k}_*, \tag{101}$$

which requires $\mathcal{O}(n^2)$ operations. The scalar α is then given by $\alpha = \sqrt{k_{**} - \boldsymbol{v}^{\top}\boldsymbol{v}}$, which is guaranteed to be real and positive provided that the augmented kernel matrix remains positive definite. This incremental update avoids recomputing the Cholesky factorisation from scratch and reduces the computational cost of incorporating a new data point from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ per iteration. Although the overall complexity of Gaussian Process regression remains cubic in the number of samples, such updates significantly reduce constant factors and are essential for efficient Bayesian Optimization implementations (Rasmussen and Williams, 2006; Golub and Van Loan, 2013).

6.2 Bellman's principle of optimality and the Q function

We now seek to learn functions that evaluate the long-term consequences of states and actions. In contrast to policy-search approaches, the policy itself is not optimised directly. Instead, we approximate *value functions* that characterise the expected cumulative reward introduced in (90), but conditioned on the state or on the state–action pair.

As in the previous sections, we consider finite-horizon episodes of length N. The environment is stochastic through the initial condition $\boldsymbol{s}_{\bullet,0} \sim \mathcal{I}_0$ and the exogenous inputs $\boldsymbol{z}(\cdot) \sim \mathcal{P}_z$, while actions are selected deterministically. All expectations below are therefore taken with respect to the same sources of randomness as in (90).

Because the horizon is finite, the value of a state depends on the time at which it is visited. We therefore introduce a time-indexed value function $V_k : \mathcal{S} \to \mathbb{R}$, defined as the maximum expected cumulative reward achievable from interaction k onward,

$$V_k(\boldsymbol{s}) \;:=\; \sup_{\{\boldsymbol{a}_t\}_{t=k}^{N-1}} \mathbb{E}\left[\sum_{t=k}^{N-1} \gamma^{\,t-k} r_{\bullet,\pi}\Big(\boldsymbol{s}_{\bullet,t}, \boldsymbol{a}_{\bullet,t}\Big) \;\middle|\; \boldsymbol{s}_{\bullet,k} = \boldsymbol{s}\right], \tag{102}$$

where the state evolves according to

$$\boldsymbol{s}_{\bullet,t+1} = F_{\bullet}(\boldsymbol{s}_{\bullet,t}, \boldsymbol{z}_t, \boldsymbol{a}_{\bullet,t}). \tag{103}$$

By convention, the terminal value satisfies

$$V_N(\boldsymbol{s}) = 0, \tag{104}$$

since no rewards remain beyond interaction $N-1$.

Thus, $V_k(\boldsymbol{s})$ represents the optimal expected return conditioned on being in state $\boldsymbol{s}$ at interaction k, with $N-k$ interactions remaining.

It is often useful to further condition the expected return on the first action taken at interaction k. This leads to the *state–action value function* (or *Q-function*) $Q_k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, defined as

$$Q_k(\boldsymbol{s}, \boldsymbol{a}) \;:=\; \sup_{\{\boldsymbol{a}_t\}_{t=k+1}^{N-1}} \mathbb{E}\left[\sum_{t=k}^{N-1} \gamma^{\,t-k} r_{\bullet,\pi}\Big(\boldsymbol{s}_{\bullet,t}, \boldsymbol{a}_{\bullet,t}\Big) \;\middle|\; \boldsymbol{s}_{\bullet,k} = \boldsymbol{s},\; \boldsymbol{a}_{\bullet,k} = \boldsymbol{a}\right]. \tag{105}$$

By construction, the two functions are related by

$$V_k(\boldsymbol{s}) = \sup_{\boldsymbol{a}\in\mathcal{A}} Q_k(\boldsymbol{s}, \boldsymbol{a}), \tag{106}$$

which states that the value of a state at interaction k is obtained by selecting the best available action at that time.

Intuitively, $V_k(\boldsymbol{s})$ measures how favourable it is to be in state $\boldsymbol{s}$ at interaction k in terms of achievable future performance over the remaining horizon, while $Q_k(\boldsymbol{s}, \boldsymbol{a})$ measures the quality of applying a specific control action $\boldsymbol{a}$ when in state $\boldsymbol{s}$ at interaction k. This decomposition is the foundation of Bellman's principle of optimality and underlies value-based reinforcement learning methods such as Q-learning, which aim to approximate these functions from data.

Learning the optimal state–action value function $Q_k(\boldsymbol{s}, \boldsymbol{a})$ is an ambitious objective and a major conceptual shift from policy-based methods. The function Q_k assigns to each state–action pair the expected long-term return under optimal behaviour over the remaining horizon. If it were known exactly, optimal control at interaction k would follow the greedy rule

$$\pi_k^\star(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}\in\mathcal{A}} Q_k(\boldsymbol{s}, \boldsymbol{a}), \tag{107}$$

since all planning and long-term reasoning would already be encoded in $Q_k^\star$. In words, the goal is not to teach the agent how to *act*, but rather to teach it the *value* of its actions.

The optimization in (107) is computationally inexpensive, and the resulting policy requires no explicit parametrisation: it may generate feedback laws that would be extremely difficult to parameterise, interpret, or even conceive a priori. This is one of the central merits of reinforcement learning. By learning the value of actions rather than prescribing behaviour directly, agents can, in principle, discover strategies that were never explicitly encoded by a human designer. This mechanism is what allowed deep reinforcement learning agents to "surpass the teacher" and reach superhuman performance in a variety of challenging settings: Atari 2600 games learned directly from raw pixels (Mnih et al., 2015), large-scale partially observable environments such as AlphaStar in StarCraft II (Vinyals et al., 2019), and the OpenAI Five system in Dota 2 (Berner et al., 2019).

From a broader (and somewhat romantic) perspective, the pursuit of a robust approximation of the Q-function can be seen as a pathway toward discovering decision strategies that transcend human intuition, not by mimicking expertise, but by systematically evaluating the long-term consequences of actions. We will see a leaner surpassing the teacher in the following tutorial.

The indirect action-selection mechanism closely resembles the philosophy of Model Predictive Control (MPC, Bertsekas (2024)), where an optimization problem is solved online at each time step to select the action that minimizes a predicted cost over a finite horizon, given the current state. The key difference with Q learning is that the optimization is performed over a learned value function rather than over a predictive model of the system dynamics.

6.3 Learning the Value of Actions

The definitions of the value and state–action value functions imply a fundamental consistency property, known as *Bellman's principle of optimality*. In the stochastic setting

considered throughout these notes, this principle states that an action is optimal if it maximizes the immediate reward plus the expected optimal future return from the next state. Applied to the state–action value function, Bellman's principle yields the optimality relation

$$Q(\boldsymbol{s}_k, \boldsymbol{a}_k) = r(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma\, \mathbb{E}\left[\sup_{\boldsymbol{a} \in \mathcal{A}} Q(\boldsymbol{s}_{k+1}, \boldsymbol{a}) \;\middle|\; \boldsymbol{s}_k, \boldsymbol{a}_k \right], \tag{108}$$

where the expectation is taken with respect to the conditional distribution of the next state $\boldsymbol{s}_{k+1}$ induced by the stochastic environment $\boldsymbol{s}_{k+1} = F_\bullet(\boldsymbol{s}_k, \boldsymbol{z}_k, \boldsymbol{a}_k)$.

Tabular Q-learning. In its simplest form, value-based reinforcement learning assumes that both the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are finite and of moderate size. In this case, the state–action value function can be stored explicitly in a *Q-table*,

$$Q(\boldsymbol{s}, \boldsymbol{a}) \in \mathbb{R}, \qquad \text{for all } (\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S} \times \mathcal{A}.$$

The Bellman optimality relation (108) then defines a fixed-point equation over this finite set of entries. Since the transition distribution is unknown, the expectation in (108) cannot be evaluated analytically. Instead, it is approximated using samples obtained from interaction with the system. Given an observed transition $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$, the resulting sample-based update reads

$$Q(\boldsymbol{s}_k, \boldsymbol{a}_k) \;\leftarrow\; Q(\boldsymbol{s}_k, \boldsymbol{a}_k) + \alpha_k \left[r_k + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q(\boldsymbol{s}_{k+1}, \boldsymbol{a}) - Q(\boldsymbol{s}_k, \boldsymbol{a}_k) \right], \tag{109}$$

where $\alpha_k \in (0, 1]$ is a learning rate and $\gamma \in [0, 1]$ a discount factor. The term in brackets is the *temporal-difference (TD) error*, which measures the discrepancy between the current estimate and the one-step Bellman target constructed from the observed sample. This tabular update constitutes the classical Q-learning algorithm (Watkins and Dayan, 1992).

For completeness, Algorithm 3 summarises the tabular procedure. Action selection during learning must balance exploitation of the current Q-function estimate with exploration of alternative actions in order to gather informative data. A common choice is the *ε-greedy policy*, defined as the stochastic policy

$$\pi_\varepsilon(a \mid \boldsymbol{s}_k) = \begin{cases} 1 - \varepsilon, & a \in \arg\max_{\tilde{a} \in \mathcal{A}} Q(\boldsymbol{s}_k, \tilde{a}), \\ \dfrac{\varepsilon}{|\mathcal{A}| - 1}, & \text{otherwise}, \end{cases} \tag{110}$$

where $|\mathcal{A}|$ denotes the number of admissible actions.

In words, the action that maximizes the current Q-value estimate is selected (exploitation) with probability $1 - \varepsilon$, while with probability ε an action is drawn uniformly at random from the remaining admissible actions (exploration). The parameter $\varepsilon \in [0, 1]$ therefore directly controls the exploration–exploitation trade-off and is typically decreased over the course of training (Sutton and Barto, 2018).

While conceptually simple, the tabular formulation suffers from a fundamental limitation: the number of Q-values grows proportionally to $|\mathcal{S}|\,|\mathcal{A}|$. For problems with continuous states, continuous actions, or even moderately high-dimensional discretisations, this quickly becomes computationally infeasible and statistically inefficient. In such settings, storing a separate value for each state–action pair is no longer practical, and one must resort to function approximation.

Algorithm 3: Tabular Q-learning for finite state and action spaces

Input: Episode length N, learning rates $\{\alpha_k\}$, discount factor γ

Initialise $Q(\boldsymbol{s}, \boldsymbol{a})$ arbitrarily for all $(\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S} \times \mathcal{A}$;

for *each episode* **do**

  Observe initial state $\boldsymbol{s}_0 \sim \mathcal{I}_0$;

  for $k = 0, \ldots, N-1$ **do**

    Select action $\boldsymbol{a}_k$ using an exploratory policy (e.g. ε-greedy w.r.t. Q);

    Apply $\boldsymbol{a}_k$, observe reward r_k and next state $\boldsymbol{s}_{k+1}$;

    Update

$$Q(\boldsymbol{s}_k, \boldsymbol{a}_k) \leftarrow Q(\boldsymbol{s}_k, \boldsymbol{a}_k) + \alpha_k \Big(r_k + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q(\boldsymbol{s}_{k+1}, \boldsymbol{a}) - Q(\boldsymbol{s}_k, \boldsymbol{a}_k) \Big)$$

Parametric and deep Q-learning. A natural extension of tabular learning consists in introducing a parametric approximation $Q(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}_q)$, where $\boldsymbol{\theta}_q$ denotes a set of adjustable parameters. Typical choices include linear function approximators, radial-basis-function expansions, or neural networks. The tabular case is recovered as a special instance in which each state–action pair has its own independent parameter. When the approximation is realised by a neural network with multiple hidden layers, the resulting approach is commonly referred to as *Deep Q-learning*. For consistency with the Bellman relation introduced previously, we retain here the discount factor $\gamma \in [0, 1]$ explicitly in the temporal-difference target.

With function approximation, the Bellman optimality condition (108) cannot be enforced exactly, since the expectation over next states is unknown. Instead, it is approximated locally using observed transitions (Sutton and Barto, 2018). Given a transition $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ observed from interaction with the system, this leads to the temporal-difference error

$$\delta_k(\boldsymbol{\theta}_q) = r_k + \gamma \sup_{\boldsymbol{a} \in \mathcal{A}} Q(\boldsymbol{s}_{k+1}, \boldsymbol{a}; \boldsymbol{\theta}_q) - Q(\boldsymbol{s}_k, \boldsymbol{a}_k; \boldsymbol{\theta}_q), \tag{111}$$

which measures the violation of Bellman consistency for the current approximation of the Q-function. Q-learning updates the parameters $\boldsymbol{\theta}_q$ so as to reduce this TD error along observed trajectories, thereby driving the approximation toward a fixed point of the Bellman optimality operator. In the tabular case, this recovers the update (109); with function approximation, the update becomes a stochastic optimization problem over $\boldsymbol{\theta}_q$. Algorithm 4 summarises the Q-learning procedure for the finite-horizon stochastic setting considered here. Stochasticity may arise from process disturbances, partial observability, or measurement noise. In this stochastic setting, the expectation in (108) is approximated by Monte Carlo sampling, using the observed transitions generated by interaction with the environment. Q-learning can thus be interpreted as a sample-based stochastic approximation of Bellman's optimality equation. Exploration is commonly enforced either through an explicitly stochastic decision rule, such as an ε-greedy policy, or by perturbing the selected action with random noise.

Algorithm 4: Q-learning with function approximation for finite-horizon stochastic control

Input: Episode length N, discount factor γ, learning rates $\{\alpha_k\}$, Q-function approximator $Q(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}_q)$

Initialise parameters $\boldsymbol{\theta}_q$;

for *each episode* **do**

  Observe initial state $\boldsymbol{s}_0 \sim \mathcal{I}_0$;

  for $k = 0, \dots, N-1$ **do**

    Select action $\boldsymbol{a}_k$ using an exploratory policy (e.g. ε-greedy with respect to Q);

    Apply $\boldsymbol{a}_k$, observe reward r_k and next state $\boldsymbol{s}_{k+1}$;

    Compute TD error;

$$\delta_k = r_k + \gamma \sup_{\boldsymbol{a} \in \mathcal{A}} Q(\boldsymbol{s}_{k+1}, \boldsymbol{a}; \boldsymbol{\theta}_q) - Q(\boldsymbol{s}_k, \boldsymbol{a}_k; \boldsymbol{\theta}_q)$$

    Update parameters

$$\boldsymbol{\theta}_q \leftarrow \boldsymbol{\theta}_q + \alpha_k \, \delta_k \, \nabla_{\boldsymbol{\theta}_q} Q(\boldsymbol{s}_k, \boldsymbol{a}_k; \boldsymbol{\theta}_q)$$

Final Remarks We close this brief introduction to Q-learning by noting that these algorithms are *off-policy* methods. The temporal-difference target relies on a greedy maximization over actions at the next state and therefore assumes optimal future behaviour, independently of how actions are actually selected during learning. As a result, the Q-function is driven toward the value associated with the optimal policy, even when data are generated using exploratory or suboptimal actions (Sutton and Barto, 2018).

An alternative is provided by *on-policy* methods, which instead learn the value of the policy that is actually executed. A prominent example is the SARSA algorithm, whose name reflects the sequence of variables involved in the update (state–action–reward–state–action). In SARSA, the temporal-difference target uses the value of the next action selected by the current policy, rather than a greedy maximization, thereby explicitly accounting for exploratory behaviour (Rummery and Niranjan, 1994; Sutton and Barto, 2018). This distinction becomes particularly relevant when function approximation is employed, as on-policy methods often exhibit improved stability and more conservative behaviour in practice. The following tutorial explores both off-policy and on-policy approaches on a simple control problem, and contrasts them with the policy-based method introduced in Section 6.1.

6.4 Tutorial 4: Pressure Control in Energy Storage Systems

Policy-search and Value-Based Control of a Thermal System

Problem Set. In this exercise we investigate a nonlinear control problem in which the optimal policy may be either smooth or bang–bang, depending on the relative weighting between performance and control effort. The purpose of the exercise is to compare the performance of Bayesian optimization (policy search) and Q-learning (value-based reinforcement learning) on the same control task.

We consider the discrete-time system

$$\begin{aligned} s_{k+1} &= s_k + \Delta t\Big(q_k - c\,\tanh(\kappa v_k)\Big), \\ v_{k+1} &= v_k + \frac{\Delta t}{\tau}\,(a_k - v_k), \end{aligned} \tag{112}$$

where s_k denotes the controlled state, a_k the commanded control action, and v_k the realised actuator position. The disturbance $q_k \geq 0$ represents an exogenous forcing that drives the system upward, while the bounded nonlinear term $c\,\tanh(\kappa v_k)$ acts as a saturating sink. The second equation models first-order actuator dynamics with time constant τ, accounting for the finite response time of the command implementation.

The control objective is to maintain the state within a prescribed safe band while penalising excessive actuation. Because the disturbance is strictly non-negative, the system naturally drifts away from the safe region unless corrective action is applied. Depending on the relative penalty imposed on control effort, the optimal strategy may consist of gradual corrections or of rapid switching between extreme actuator values. In the latter case, the solution approaches a bang–bang policy.

Although introduced in abstract form, this system captures the essential structure of a practical engineering problem: pressure regulation in a cryogenic storage tank equipped with a *thermal venting system* (TVS). In such systems, heat ingress induces boil-off and ullage pressurisation, while pressure control is achieved by activating a secondary loop that extracts liquid, cools it through a heat exchanger, and reinjects it into the tank. The reinjected subcooled liquid enhances condensation at the liquid–vapour interface, thereby reducing ullage pressure without direct mass venting.

Within this abstraction, the disturbance q_k represents the net pressurisation rate due to heat ingress, the bounded sink term $c\,\tanh(\kappa v_k)$ models the finite cooling and condensation capacity of the TVS, and the actuator state v_k reflects the finite valve response and associated loop dynamics (Mer et al., 2016; Barsi and Kassemi, 2013b,a). Advanced hybrid modelling and control strategies for such systems are currently being investigated in the PhD project of F. Monteiro.

A baseline bang–bang policy As a first reference strategy, we consider a simple bang–bang (on–off) control law. The commanded valve opening is restricted to two

values, $a_k \in \{0, a_{\max}\}$, and is updated according to the current state:

$$a_k = \begin{cases} a_{\max}, & s_k > s_{ON}, \\ 0, & s_k < s_{OFF}, \\ a_{k-1}, & \text{otherwise}, \end{cases} \tag{113}$$

with $s_{OFF} < s_{ON}$.

The pair (s_{OFF}, s_{ON}) defines an internal control band strictly inside the safe region $[s_{\min}, s_{\max}]$, i.e. $s_{\min} < s_{OFF} < s_{ON} < s_{\max}$, to account for actuator inertia and nonlinear saturation. Because the realised actuator state v_k tracks the command a_k with a finite time constant τ, corrective action is delayed. If the thresholds were set at $s_{\min}$ and $s_{\max}$, this lag would routinely cause safety bounds violations. The internal band thus serves as a predictive buffer for dynamical delay.

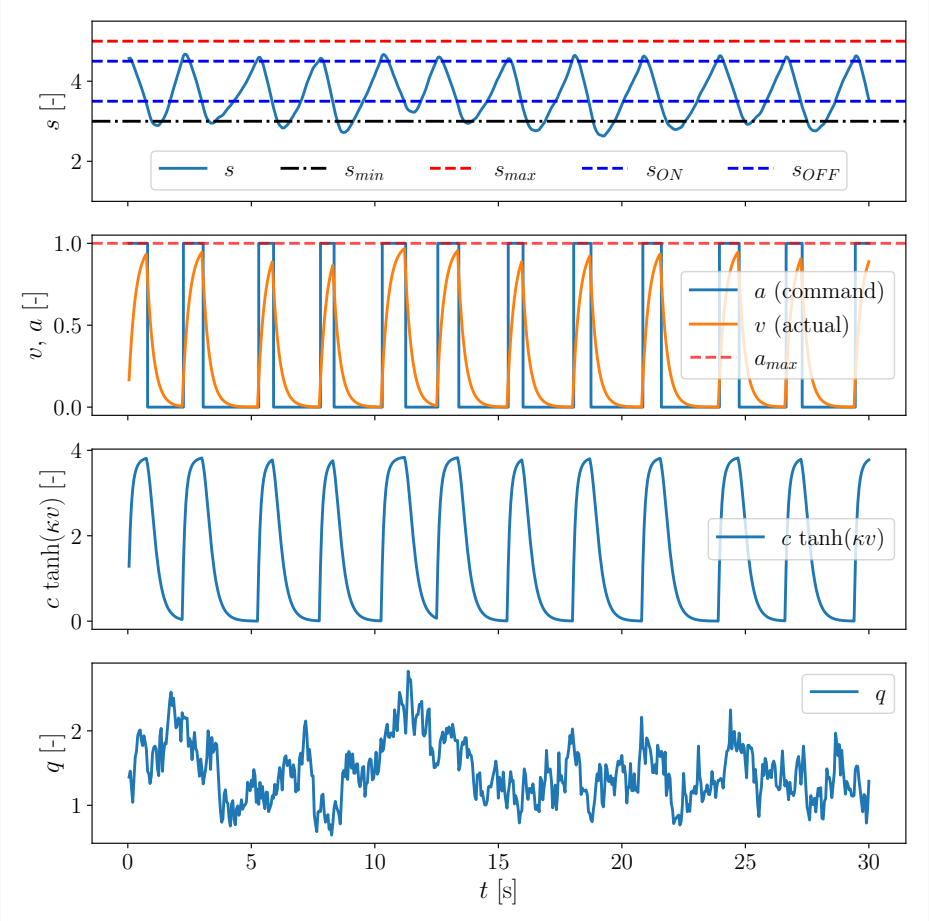


Figure 13: Baseline bang–bang (on–off) control with hysteresis for the valve environment over $N = 600$ steps with $\Delta t = 0.05$ s. Plant parameters: sink scale $c = 4.0$, saturation slope $\kappa = 2.0$, actuator time constant $\tau = 0.3$, and normalised command $a_k \in \{0, a_{max}\}$ with $a_{max} = 1.0$. The safe operating band is $s \in [s_{min}, s_{max}] = [3, 5.0]$ and the on/off band is $[s_{OFF}, s_{ON}] = [3.5, 4.5]$. Initial condition: $s_0 = 4.5$, $v_0 = 0.0$. The exogenous disturbance q is computed with $seed = 1.0$. This is treated as a Ornstein–Uhlenbeck process with two scales, and the reader is referred to the python codes for more details.

The hysteresis mechanism, which keeps the previous command when $s_k \in [s_{OFF}, s_{ON}]$, avoids rapid switching near the thresholds and reduces chattering.

Figure 13 shows a representative simulation of this policy: the trajectories of s_k, the command a_k and realised actuator position v_k, the effective removal term $c\tanh(\kappa v_k)$, and the external load q_k. Despite actuator lag and saturation, the hysteresis controller keeps the state within the safe region, with only brief overshoots during switching, mainly due to the finite actuator response time.

In many applications the lower bound $s_{\min}$ is more critical. After switching off ($a_k = 0$), v_k decays slowly, so $c\tanh(\kappa v_k)$ remains active and can keep driving the state downward. This makes switch-off inertia more harmful than switch-on inertia, and s_{OFF} may therefore need to be set more conservatively relative to $s_{\min}$. Despite its simplicity and robustness, the bang–bang policy in (113) has intrinsic limitations. The switching thresholds s_{OFF} and s_{ON} must be selected a priori and remain fixed, independently of the disturbance level and actuator dynamics. The controller therefore reacts only after the state crosses a prescribed boundary, rather than anticipating its future evolution. In the presence of rapid increases in the external load q_k, particularly when actuator delays are significant, this reactive behaviour may lead to degraded performance.

Moreover, the on–off nature of the policy induces repeated valve switching. Although hysteresis mitigates chattering, frequent transitions between extreme commands remain undesirable due to actuator wear and maintenance considerations. The restriction to $a_k \in \{0, a_{\max}\}$ limits the controller's ability to modulate its response smoothly and to balance performance against control effort in a nuanced manner.

A more refined strategy would allow continuous control actions $a_k \in [0, a_{\max}]$, enabling smoother regulation and potentially improved efficiency. If accurate dynamical models were available, such a problem could be addressed using model-based optimal control techniques such as Model Predictive Control (MPC). In the present exercise, however, we deliberately assume that such models are unavailable and instead investigate whether learning-based approaches can recover effective control strategies directly from interaction with the system.

Learning formulation and reward definition We study two complementary learning-based control strategies. The first is the policy-based approach introduced in 6.1, where the control law is explicitly parameterised and its parameters are optimised using Bayesian optimization (BO). The second is a value-based approach, where the quality of state–action pairs is learned directly using tabular Q-learning as introduced

in 6.2. For both strategies, the objective is to maximize a cumulative reward over a finite interaction episode. Given an episode of length N, the return is defined as

$$\mathcal{R} = \sum_{k=0}^{N-1} r(s_k, a_k, a_{k-1}), \tag{114}$$

where the instantaneous reward is defined as

$$r(s_k, a_k, a_{k-1}) = -\rho_s \left(\frac{d(s_{k+1})}{s_{\text{scale}}} \right)^2 - \rho_a \left(\frac{a_k}{a_{\max}} \right)^2 - \rho_{\Delta a} \left(\frac{a_k - a_{k-1}}{a_{\max}} \right)^2. \tag{115}$$

where

$$d(s) = \max(s_{\min} - s, 0) + \max(s - s_{\max}, 0) \tag{116}$$

measures violations of the prescribed safe operating range $s \in [s_{\min}, s_{\max}]$.

The quantity $s_{\text{scale}} = \alpha(s_{\max} - s_{\min})$ defines a characteristic state scale used to normalise band violations. In this tutorial we take $\alpha = 0.1$, so that safety violations are measured relative to 10% of the admissible band. Control magnitudes and variations are normalised with respect to $a_{\max}$, which renders the weights ρ_s, ρ_a, and $\rho_{\Delta a}$ dimensionless and directly comparable in magnitude.

The first term penalises violations of the prescribed safety band. The function $d(s)$ is zero whenever $s \in [s_{\min}, s_{\max}]$ and grows linearly outside this interval; the quadratic scaling therefore produces a progressively stronger penalty as the state moves further away from the admissible region. Inside the safe band no state penalty is applied, so the controller is free to trade actuation effort against proximity to the boundaries.

The second term penalises the magnitude of the control input and discourages unnecessarily large actuation. This term promotes energy efficiency and prevents the trivial strategy of permanently applying maximum control authority. The third term penalises rapid variations of the command signal. By discouraging large differences between successive control inputs, it mitigates aggressive switching behaviour and implicitly accounts for actuator wear and mechanical stress. The weights ρ_s, ρ_a, and $\rho_{\Delta a}$ therefore determine the trade-off between safety enforcement, control effort, and smoothness of operation. In this tutorial, we fix $\boldsymbol{\rho} = [\rho_s, \rho_a, \rho_{\Delta a}] = [5.0,\ 0.5,\ 5.0]$. Different choices of these parameters may lead to qualitatively different optimal policies, ranging from smooth regulation to bang–bang behaviour.

Direct policy search (Bayesian optimization). We now consider the policy-based approach introduced in Sec. 6.1, in which the control law is explicitly parameterised and its parameters $\boldsymbol{\theta}$ are tuned using Bayesian optimization (BO).

We adopt a non-dimensional sigmoid policy augmented with a derivative term, so that the same functional form can represent both smooth continuous actuation and near bang–bang switching:

$$a_k = a_{\max}\, \sigma\left(\alpha \left[\frac{s_k - s_{ref}}{s_{\text{scale}}} + \beta \frac{T_c\, \hat{\dot{s}}_k}{s_{\text{scale}}} \right] \right), \qquad \sigma(\xi) = \frac{1}{1 + e^{-\xi}}, \tag{117}$$

where $\boldsymbol{\theta}_\pi = [\alpha, s_{ref}, \beta]$ are the tunable policy parameters.

The quantity s_{scale} defines a characteristic state scale (in this tutorial taken as $s_{\text{scale}} = 0.1\,(s_{\max} - s_{\min})$), ensuring that state deviations are measured relative to the admissible band. The factor T_c is a characteristic time scale (taken here as the actuator time constant τ), which renders the derivative contribution dimensionless. With this normalisation, both terms inside the brackets are non-dimensional and directly comparable in magnitude.

The parameter $\alpha > 0$ controls the sharpness of the sigmoid transition: small values produce smooth, approximately linear behaviour near s_{ref}, whereas large values drive the policy toward bang–bang switching. The reference level s_{ref} defines the centre of the transition, and the coefficient β weights the anticipative (rate) contribution.

BO is used to solve

$$\boldsymbol{\theta}_\pi^\star = \arg\max_{\boldsymbol{\theta}_\pi} \mathcal{R}(\boldsymbol{\theta}_\pi),$$

where the return $\mathcal{R}$ is estimated through episodic interaction with the environment.

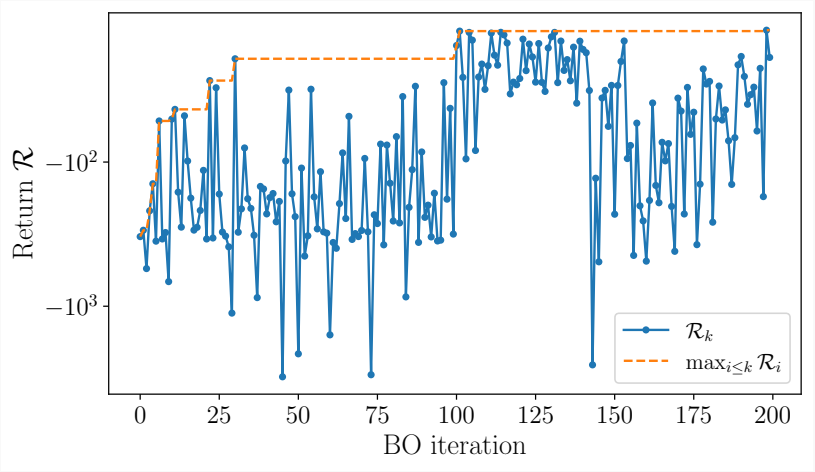


Figure 14: Bayesian optimization of the sigmoid policy parameters $\boldsymbol{\theta}_\pi = [\alpha, s_{ref}, \beta]$. The search space was defined as $\alpha \in [1, 10]$, $s_{ref} \in [s_{\min}, s_{\max}]$, and $\beta \in [0, 1]$. The optimization was performed for 200 function evaluations (100 random initial points followed by Gaussian-process-guided optimization using Expected Improvement). The solid curve shows the episodic return $\mathcal{R}_k$, while the dashed curve represents the best return obtained up to iteration k.

Figure 14 reports the evolution of the episodic return during the Bayesian optimization process, with the caption reporting on the details of the search space. Despite the stochastic disturbance affecting each episode, the best-so-far return improves steadily and eventually stabilises, indicating that BO successfully identifies a high-performing region of the parameter space. The variability of the individual returns reflects the exploration phase of the algorithm as well as the inherent variability of the environment.

The best performance reaches $\mathcal{R} = -18.63$, with the best combination of parameters identified to be $\boldsymbol{\theta}_\pi^\star = [\alpha,\ s_{ref},\ \beta] = [3.09, 4.86, 0.01]$. The identified slope $\alpha = 3.09$ corresponds to a moderately sharp transition, indicating that a smooth control law is preferred over near bang–bang switching under the chosen weight configuration. The

reference level s_{ref} lies close to the upper safety bound, suggesting that the optimal strategy maintains the state biased toward the upper part of the admissible band in order to mitigate the more critical dynamics associated with downward excursions. Finally, the very small value $\beta = 0.01$ indicates that the derivative term contributes only marginally to performance, so that the optimal controller behaves almost as a static nonlinear state feedback.

Finally, Figure 15 a representative closed-loop episode obtained with the optimised sigmoid policy. In contrast to the baseline bang–bang controller, the actuation signal is continuous and varies smoothly in time, with the actuator state v_k closely tracking the command a_k and no rapid switching events. The state remains safely within the admissible band and is maintained preferentially toward the upper part of the interval, consistent with the identified reference level $s_{ref} = 4.86$. This bias reflects the higher inertia associated with switching off the actuation, which may otherwise induce undesired excursions toward the lower safety bound. The removal rate adapts gradually to disturbance variations, so that regulation is achieved through smooth modulation of actuation intensity rather than through discrete switching.

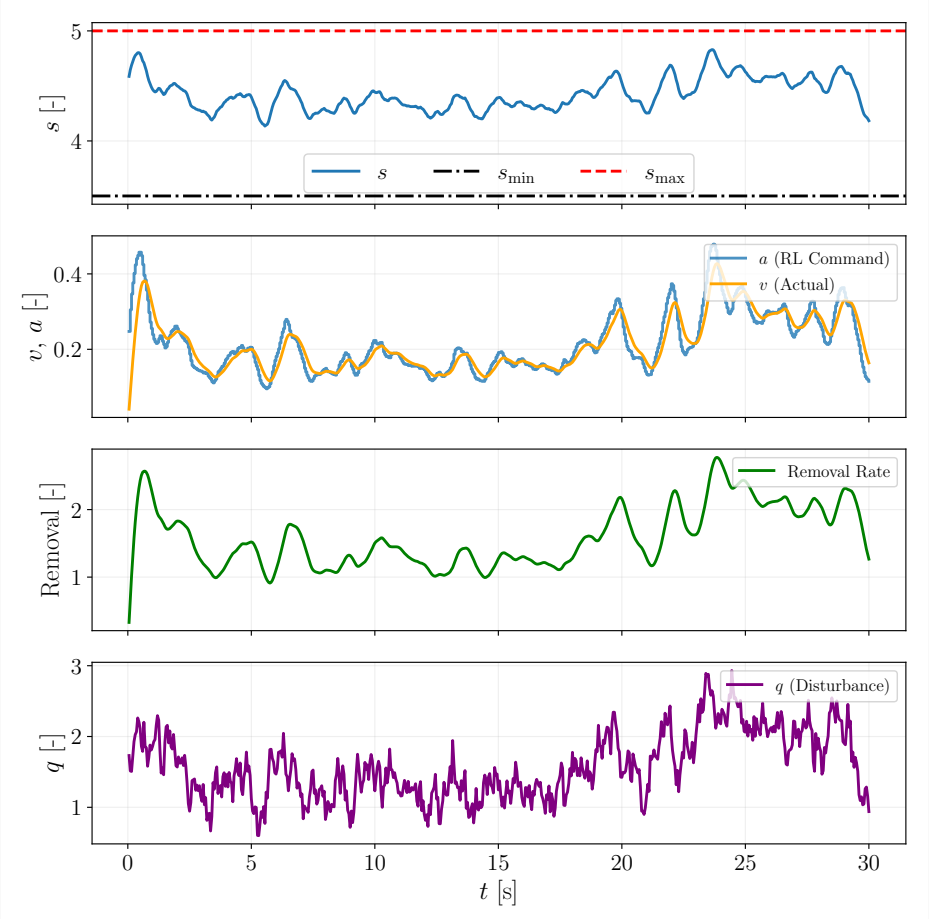


Figure 15: Closed-loop response under the policy obtained via Bayesian optimization, to be compared with the bang–bang baseline in Fig. 13. From top to bottom: state evolution s_k with safety bounds, commanded and realised actuator positions (a_k, v_k), effective removal rate $c\tanh(\kappa v_k)$, and disturbance q_k. The optimised sigmoid policy produces continuous actuation and maintains the state within the admissible band while avoiding aggressive switching.

The parameterised policy (117) offers substantial flexibility: by tuning α, s_{ref}, and β, it can approximate both smooth and near bang–bang behaviour, with or without anticipative action. Bayesian optimization selected the best controller within this prescribed functional class.

Can there be any better? We now consider an alternative in which no policy structure is imposed. With Q-learning, the agent learns the value of state–action pairs directly and derives its policy implicitly from the value function, potentially exploring behaviours beyond the confines of a predefined parametrisation. This is a *much* more challenging optimization.

Value-based learning via tabular Q-learning. We here implement the tabular learning introduced in Sec. 6.2. In this setting, the continuous observation $(s_k, \hat{\dot{s}}_k)$ and control input a_k are discretised into finite bins. Let n_s and $n_{\dot{s}}$ denote the number of bins for s and $\hat{\dot{s}}$, and n_a the number of discrete actions. The resulting Q-table has dimension

$$Q \in \mathbb{R}^{n_s \times n_{\dot{s}} \times n_a},$$

so that each entry $Q(i_s, i_{\dot{s}}, i_a)$ represents the estimated return obtained when selecting action i_a in the corresponding state bin.

In this tutorial we first consider a coarse discretisation $n_s \times n_{\dot{s}} \times n_a = 10 \times 10 \times 11$, and later increase the resolution to $20 \times 20 \times 11$.

This "cubic table" is updated according to the standard temporal-difference rule (109), here repeated for convenience

$$Q(s,a) \leftarrow Q(s,a) + \alpha_Q \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \tag{118}$$

where $\alpha_Q \in (0,1]$ is the learning rate and $\gamma \in [0,1]$ the discount factor.

The learning rate α_Q dictates the weight of the temporal difference error relative to existing knowledge. In the extreme case of $\alpha_Q = 1$, the agent becomes "memoryless," discarding previous estimates to match the most recent target $r + \gamma \max_{a'} Q(s',a')$. This induces heavy oscillations as the Q-function "chases" stochastic noise and outliers rather than averaging them into a stable expectation. Conversely, as $\alpha_Q \to 0$, the agent becomes overly "stubborn"; while this eliminates oscillations and ensures a smooth trajectory, it risks stalling the learning process on prior suboptimal beliefs and failing to adapt to the environment's dynamics. Finding the right α_Q is extremely difficult. The reader is encouraged to repeat the training with different values.

A second crucial parameter is the exploration parameter ε in the ε-greedy strategy in (110), according to which the agent takes the greedy action $\arg\max_a Q(s,a)$ with

probability $1-\varepsilon$ (exploitation) and a random action (exploration) with probability ε. This trade-off is particularly difficult because a high ε prevents the agent from refining a high-performance policy by constantly injecting random noise into its behavior. Conversely, an ε that is too low leads to premature convergence, where the agent becomes trapped in a sub-optimal local maximum because it has not explored the state-action space sufficiently to discover superior strategies.

As the reader will realize studying the provided codes, both α_Q and ε are adapted dynamically during the learning process. The trade-off that these two parameters identifies becomes more and more critical as the discretization of the action-space cube becomes finer. Increasing the resolution from $10 \times 10 \times 11$ to $20 \times 20 \times 21$ increases the number of Q-values from 1100 to 8400. This means that each state–action pair is visited far less frequently during training. Since Q-learning relies on repeated updates of individual entries, the effective sample density per entry drops dramatically.

After spending several hours with the provided codes, we report that direct Q learning appeared feasible on the $10\times10\times11$ cube but impossible (or at least the writer did not succeed!) in $20 \times 20 \times 21$ cube. To explore the challenges and opportunities offered by these methods, we here propose a four-phase approach, implemented in the script in the file `4_main_Q_LEARNING_PRO.py`.

Phase 0: Imitation (warm start). To avoid learning from a blank table, the agent first observes a bang–bang controller with hysteresis for a 2000 episodes. This controller acts as a teacher, selecting all actions while the learner updates its Q-table using the standard temporal-difference rule (118). This procedure serves as a form of behavioural cloning in the value space, ensuring the agent begins autonomous exploration with a competent baseline policy.

Phase 1: Autonomous "coarse" learning. After the imitation phase, the learner acts autonomously under an ε-greedy strategy with gently decaying exploration over 20 000 episodes. Figure 16 shows the evolution of the episodic reward during this stage. The horizontal lines indicate the average and best performance of the teacher. The learner rapidly surpasses the teacher baseline, demonstrating that even a coarse discretisation ($10 \times 10 \times 11$) can improve upon the heuristic bang–bang rule.

The corresponding policies are shown in ig. 17. Although the achieved returns are higher, the learned policy is irregular and exhibits abrupt variations between neighboring bins, with strong actuation next to weak actuation in the state space. This "spiky" structure is a typical artifact of limited sampling in tabular reinforcement learning: individual state–action pairs are updated unevenly, and no mechanism enforces spatial smoothness across neighboring bins.

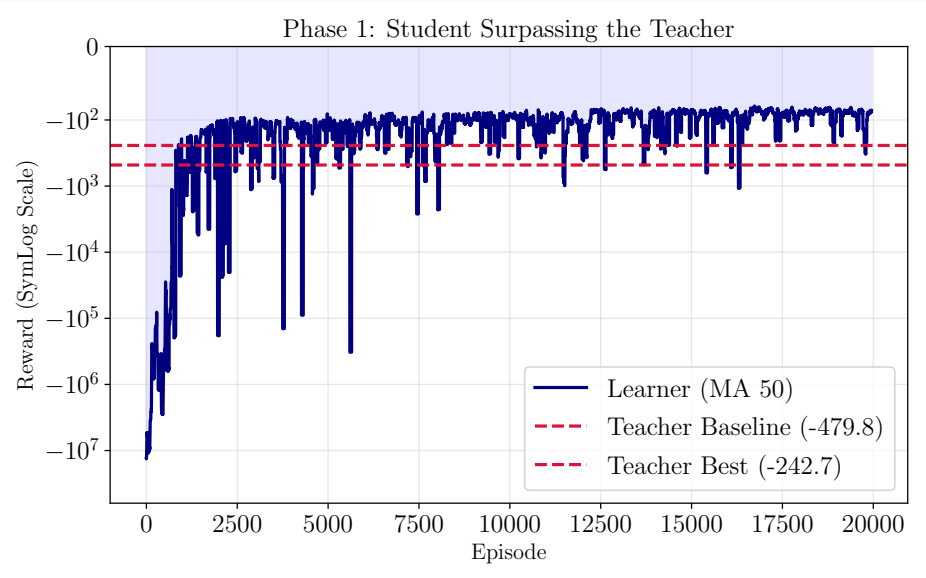


Figure 16: Phase 1 autonomous learning with coarse discretisation ($10 \times 10 \times 11$). The blue curve shows the moving average (window 50) of the episodic return obtained under ε-greedy Q-learning over 20 000 episodes. The horizontal dashed lines indicate the average and best performance achieved by the bang–bang teacher during the imitation phase.

Phase 2: "Growing up". Before further refinement, the grid resolution is increased to $20 \times 20 \times 11$. This progression follows a *curriculum learning* paradigm (Bengio et al., 2009), in which the agent is gradually exposed to increasing task complexity. After a teacher-guided warm start and a coarse autonomous phase, the refinement stage introduces a higher-resolution state space while preserving the knowledge already acquired (Narvekar et al., 2020).

The transition from $10 \times 10 \times 11$ to $20 \times 20 \times 11$ substantially increases the number of Q-values and reduces the visitation frequency of individual state–action pairs. Bins that were previously aggregated are now separated, and value estimates learned at the coarse level may no longer be locally consistent. Without additional regularization, this expansion can destabilize learning: updates become sparse, local inconsistencies are amplified, and neighboring bins may oscillate between competing actions.

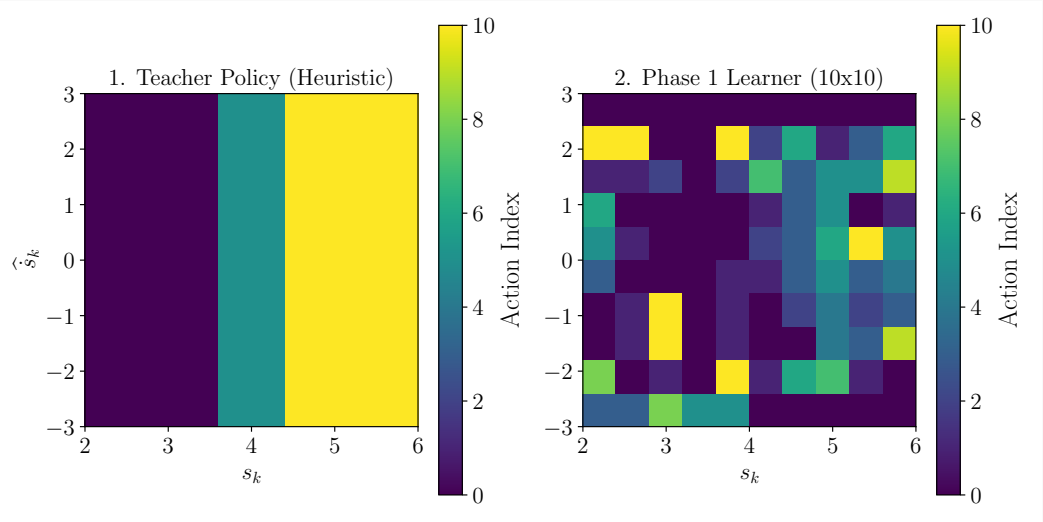


Figure 17: Teacher policy (left) and Phase 1 learner policy (right) on the coarse 10×10 state grid. Each pixel shows the greedy action $\arg\max_a Q(s, a)$. While the learner attains higher returns, the resulting policy is irregular, with abrupt changes between neighbouring bins, reflecting limited sampling in the tabular setting.

To mitigate these effects, we perform value-function transfer from the coarse grid to the refined grid using bi-linear interpolation combined with Gaussian smoothing (Taylor and Stone, 2009). This regularization enforces spatial coherence across neighboring states, consistent with principles developed in kernel-based reinforcement learning (Ormoneit and Sen, 2002).

The effect of value-function transfer is illustrated in Fig. 18. The left panel shows the greedy policy obtained at the end of Phase 1 on the coarse grid. When the grid is refined, these values are first smoothed and then interpolated onto the 20×20 state space. The resulting policy, shown on the right, preserves the large-scale structure of the coarse solution while removing isolated spikes and enforcing spatial coherence across neighboring bins. This interpolated map serves as the initial condition for the high-resolution refinement stage.

Phase 3: Coarse-to-fine curriculum refinement. This high-resolution training stage is itself divided into three successive sub-phases: *exploration*, *refinement*, and *polishing*. This additional structure is necessary because, after the grid expansion, the agent faces a substantially larger state–action space and must progressively stabilize its value estimates.

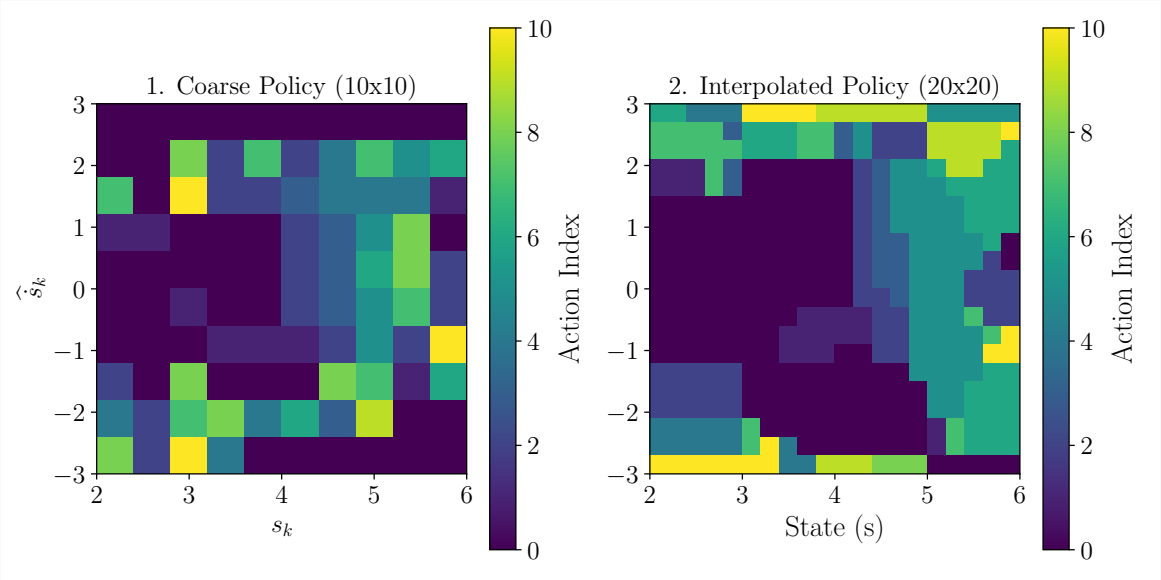


Figure 18: Coarse-to-fine policy transfer. The Phase 1 greedy map (left) is smoothed and interpolated to initialise the refined 20×20 grid (right). The procedure removes local irregularities while retaining the global structure of the learned policy.

Exploration. In the first sub-phase, which covers the first 4000 episodes, a relatively high exploration rate is maintained in order to populate the newly created bins of the refined grid. Although the interpolated policy provides a structured initialization, many state–action pairs have not yet been sufficiently visited. This stage allows the agent to redistribute value information across the expanded table and to correct inconsistencies introduced by interpolation.

Refinement. Once the coarse structure has been adapted to the fine grid, the exploration rate is gradually reduced and the learning rate is decayed. The objective in this stage is to stabilise the dominant switching boundaries and smooth transitions observed in the interpolated map, while still permitting moderate corrections. The policy begins to exhibit coherent large-scale regions in state space rather than fragmented patches. This phase begins at episode 4001 and ends at episode 16000.

Polishing. In the final sub-phase, covering the last 4000 episodes, exploration is suppressed ($\varepsilon \approx 0$) and a small learning rate is retained. Additionally, periodic Gaussian smoothing is applied to the Q-table to dampen local oscillations between neighboring bins. This stage acts as a regularization mechanism, consolidating spatial consistency and producing a stable high-resolution policy.

The complete learning process, including Phase 1 and the high-resolution curriculum of Phase 2, is shown in Fig. 16. The evolution shows the major impact of the transition to the finer grid at 20000 episodes, reflecting the sudden expansion of the state–action space and the loss of local consistency in the value estimates.

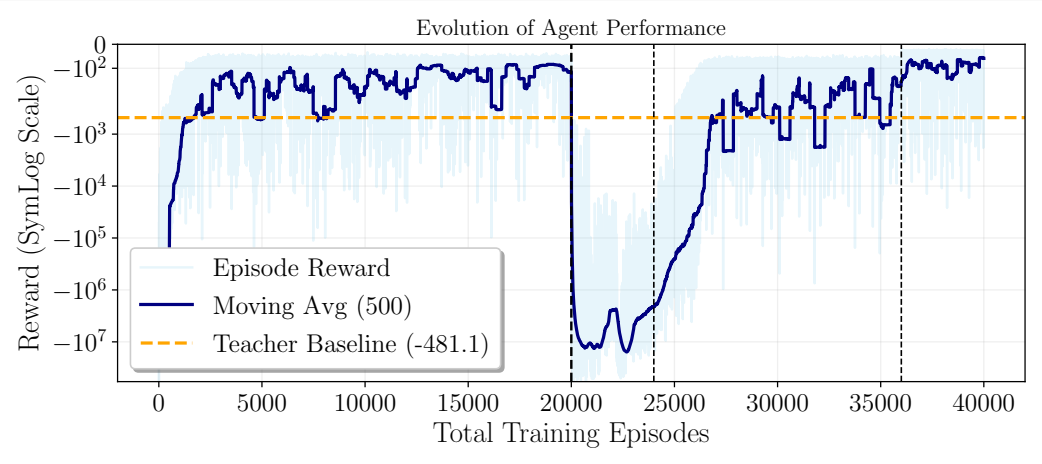


Figure 19: Reward evolution across all training stages: imitation baseline, coarse autonomous learning, and high-resolution curriculum refinement. Vertical dashed lines mark the grid transition and the successive exploration, refinement, and polishing sub-phases.

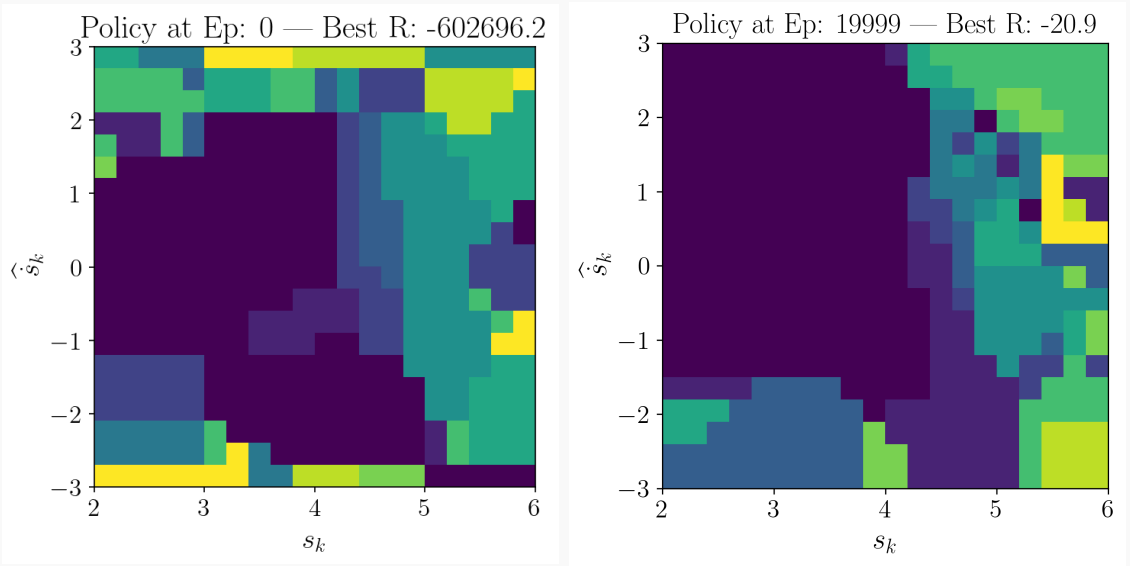


Figure 20: Policy before and after high-resolution curriculum training. The final map displays coherent switching regions structured in the $(s, \hat{s})$ plane, with actuation primarily triggered when the state drifts toward unsafe regions and suppressed when the system naturally recovers.

The agent must effectively re-stabilize its understanding of the environment at a higher resolution, and the subsequent exploration and refinement stages progressively recover the performance previously achieved on the coarse grid. Only after this consolidation does the polishing phase fully exploit the increased resolution, ultimately surpassing the performance of the coarse learner. This sequence conveys an important message: increasing representational capacity alone does not guarantee improved control. Without structured transfer and staged learning, higher resolution can destabilize value estimation. However, when combined with curriculum design and spatial

regularization, the additional resolution enables the discovery of more refined and higher-performing policies.

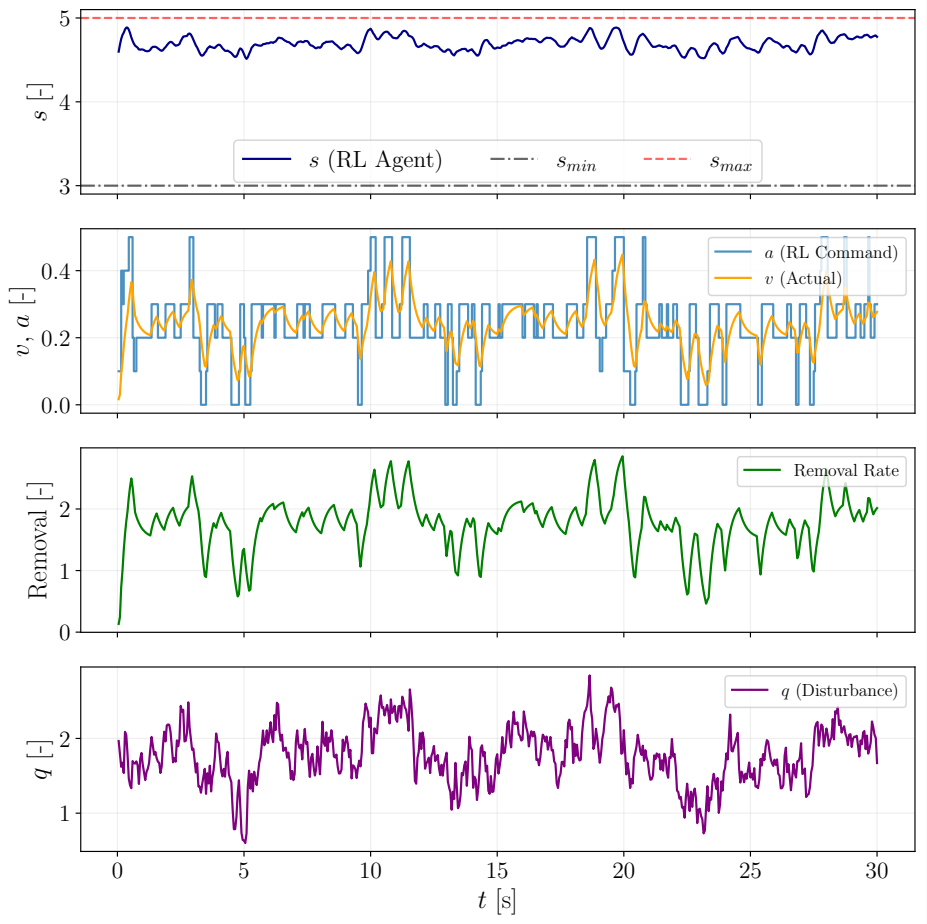


Figure 21: Closed-loop response under the policy obtained via Q learning, to be compared with the bang–bang baseline in Fig. 13.

Figure 20 shows the control policies at the beginning of the final refinement phase and at the end. The refined policy reveals a clear structure in the $(s, \hat{s})$ plane. Actuation is primarily activated when the state is low and decreasing, indicating that the agent has learned to anticipate downward drift toward the lower boundary. Conversely, little or no action is taken when the state is low but increasing, since the system is naturally recovering. Similarly, limited actuation is observed when the state is high but not drifting toward unsafe regions. The resulting policy encodes a geometric understanding of the system dynamics and exploits the derivative information to avoid unnecessary intervention. The final policy does not merely define thresholds; it encodes directional awareness in state space.

Finally, Figure 21 shows a representative closed-loop trajectory obtained with the trained Q-learning agent, yielding a return of $\mathcal{R} = -20.2$. Although this performance does not match the continuous policy obtained via Bayesian optimization (Fig. 15), it significantly improves upon the bang–bang baseline (Fig. 13). Remarkably, this behavior is achieved with an action space consisting of only 11 discrete levels. The

commanded signal is piece-wise constant, yet the actuator dynamics naturally smooth the effective valve position. The agent leverages this slow response to approximate quasi-continuous regulation using only a few low-amplitude action bins.

This reflects an efficient trade-off between safety and control effort: rather than relying on large corrective actions, the agent performs small anticipative adjustments guided by the state derivative. This illustrates how a coarse discrete action space, when combined with value-based learning and system dynamics, can produce refined and efficient closed-loop behavior.

7 Conclusion and Outlook

This chapter developed a unified view of control and learning as two faces of the same idea: *adaptation through feedback*. In all settings considered here, performance is quantified through a cost or reward functional, and feedback—from measurements, model–data mismatches, or realized rewards—is used to update either a control law, a model, or both. The key difference between the methods is therefore not whether they "optimize", but *what is assumed known*, *what is learned from data*, and *how information is propagated* through the closed-loop system.

We began with the fully model-based setting, where the dynamics are known and control synthesis exploits analytical structure. In this regime, optimality conditions and adjoint sensitivities provide an efficient route to policy improvement, and optimization remains tightly tied to the governing equations. We then relaxed the modeling assumptions and moved to data-driven model updating. The system identification and digital twinning examples highlighted a central practical point: modeling errors are unavoidable, and identification is often performed in closed loop, under the very controller that relies on the model. This coupling implies that the controller shapes the data available for learning, while the model accuracy directly constrains achievable performance. The windowed identification–control procedure made this interaction explicit by letting model fidelity and control performance evolve together. The second part of the lecture shifted to the model-free viewpoint, where the goal is to improve closed-loop behavior without relying on explicit model derivatives or structural assumptions on the dynamics. Through the comparison between Bayesian optimization and tabular Q-learning on the same nonlinear regulation problem, we illustrated two complementary learning paradigms. Policy search restricts optimization to a predefined functional family and can be highly effective when the parametrization is sufficiently expressive. Value-based reinforcement learning, by contrast, removes this structural constraint and attempts to estimate directly the long-term value of state–action pairs.

The Q-learning example highlighted both the promise and the difficulty of this approach. Even with a coarse discretization, the agent was able to surpass the heuristic bang–bang controller. However, increasing the resolution of the state space did not automatically improve performance; it initially degraded it. Only through structured transfer, smoothing, and staged curriculum learning did the higher-resolution representation translate into improved control. This emphasizes a central message of the lecture: increasing representational capacity without structure can destabilize learning, whereas carefully

managed refinement can unlock superior policies.

Taken together, the examples presented in this lecture suggest that control and learning should not be viewed as competing paradigms, but as complementary strategies for performance improvement. Model-based methods provide structure and efficiency when reliable equations are available; learning-based methods provide flexibility when modeling is incomplete. In practice, effective solutions for complex systems are likely to combine physical insight with adaptive, data-driven mechanisms rather than rely exclusively on either extreme. We return to this combination in Lecture 13.

Acknowledgements This work is part of the RE-TWIST project, which has received funding from the European Research Council (ERC) under the European Union's Horizon Europe programme (grant agreement No 101165479). The views expressed are those of the authors and do not necessarily reflect those of the European Union or the ERC. Additional support has been received for the PhD fellowships of Lorenzo Schena (FWO fellowship number 1567925N) and Sebastiano Randino (FNRS FRIA Fellowship number 40029825).

References

Abbas, N. J., Zalkind, D. S., Pao, L., and Wright, A. (2022). A reference open-source controller for fixed and floating offshore wind turbines. *Wind Energy Science*, 7(1):53–73.

Ahizi, S. A., Monteiro, F., Abarca, R., and Mendez, M. A. (2026). Real-time identification of parametric sloshing-induced heat and mass transfer in a horizontally oriented cylindrical tank. *International Journal of Heat and Mass Transfer*, 264:128707.

Anderson, B. D. O. (2008). Challenges of adaptive control—past, permanent and future. *Annual Reviews in Control*.

Annaswamy, A. M. (2023). Adaptive control and intersections with reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 6(1):65–93.

Asch, M., Bocquet, M., and Nodet, M., editors (2016). *Data Assimilation: Methods, Algorithms, and Applications*. SIAM, Philadelphia, PA, USA. Modern monograph on variational and statistical data assimilation.

Barsi, S. and Kassemi, M. (2013a). Investigation of tank pressurization and pressure control—part i: Experimental study. *Journal of Thermal Science and Engineering Applications*, 5(4).

Barsi, S. and Kassemi, M. (2013b). Investigation of tank pressurization and pressure control—part ii: Numerical modeling. *Journal of Thermal Science and Engineering Applications*, 5(4).

Bąsar, T. and Bernhard, P. (1995). *H-infinity Optimal Control and Related Minimax Design Problems*. Birkhäuser, 2nd edition.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, Belmont, MA, 4 edition.

Bertsekas, D. P. (2024). Model predictive control and reinforcement learning: A unified framework based on dynamic programming.

Bianchi, F. D., Mantz, R. J., and De Battista, H. (2007). *Wind Turbine Control Systems*. Springer London.

Bocquet, M. and Farchi, A. (2014). Introduction to the principles and methods of data assimilation in the geosciences. Lecture notes, École des Ponts ParisTech & EDF R&D, Revision 0.52, last revised 20 January 2025.

Bradley, A. M. (2024). Pde-constrained optimization and the adjoint method. Technical report, Stanford University.

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A. P. (2022). Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444.

Brunton, S. L. and Noack, B. R. (2015). Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67(5).

Candelieri, A. (2021). A gentle introduction to bayesian optimization. In *2021 Winter Simulation Conference (WSC)*, pages 1–16. IEEE.

Chong, E. K. P. and Żak, S. H. (2013). *An Introduction to Optimization*. Wiley-Interscience Texts and Monographs. Wiley, 4th edition.

Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. (2019). On the sample complexity of the linear quadratic regulator. In *Proceedings of the 2019 American Control Conference (ACC)*, pages 304–310.

Duriez, T., Brunton, S. L., and Noack, B. R. (2017). *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Springer International Publishing.

Frazier, P. I. (2018). A tutorial on bayesian optimization.

Givoli, D. (2021). A tutorial on the adjoint method for inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 380:113810.

Gligor, D., Marques, P. A., Salgado Sánchez, P., Porter, J., Méndez, M. A., and Ezquerro, J. M. (2024). Experiments on sloshing mitigation using tuned oscillating baffles. *Physics of Fluids*, 36(9).

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins University Press, 4 edition.

Hansen, M. O. L. (2015). *Aerodynamics of Wind Turbines*. Routledge.

Hartmann, D. and Van der Auweraer, H. (2025). Digital twins - a golden age for industrial mathematics. *Journal of Mathematics in Industry*, 15(1).

Hespanha, J. P. (2018). *Linear System Theory*. Princeton University Press.

Ioannou, P. A. and Sun, J. (1996). *Robust Adaptive Control*. Prentice Hall, Upper Saddle River, NJ.

Khan, S. G., Herrmann, G., Lewis, F. L., Pipe, T., and Melhuish, C. (2012). Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*, 36(1):42–59.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Law, K. J., Stuart, A. M., and Zygalakis, K. C. (2015). *Data Assimilation: A Mathematical Introduction*. Springer, Cham, Switzerland. Rigorous introduction to data assimilation.

Lewis, F. L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50.

Li, Y. and Han, S. (2022). Accelerating model-free policy optimization using model-based gradient: A composite optimization perspective.

Lions, J.-L. (1971). *Optimal Control of Systems Governed by Partial Differential Equations*, volume 170 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, Berlin, Germany. Foundational work on adjoint methods.

Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition. Classic text on system identification.

Luo, F.-M., Xu, T., Lai, H., Chen, X.-H., Zhang, W., and Yu, Y. (2024). A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2).

Martins, J. R. R. A. and Ning, A. (2022). *Engineering Design Optimization*. Cambridge University Press. Available online: https://mdobook.github.io.

Mendez, M. A. (2025). *Fundamentals of Regression*, pages 33–64. von Karman Institute for Fluid Dynamics. Also available as arXiv:2512.01920 [stat.ML].

Mendez, M. A., van den Berghe, J., Ratz, M., Fiore, M., and Schena, L. (2025). *Learning with Physical Constraints*, pages 65–87. von Karman Institute for Fluid Dynamics. Also available as arXiv:2512.00104 [stat.ML].

Mer, S., Fernandez, D., Thibault, J.-P., and Corre, C. (2016). Optimal design of a thermodynamic vent system for cryogenic propellant storage. *Cryogenics*, 80:127–137.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Müller, S., von Rohr, A., and Trimpe, S. (2021). Local policy search with bayesian optimization. Policy search method with Gaussian process surrogate models to improve sample efficiency.

Narendra, K. S. and Annaswamy, A. M. (1989). *Stable Adaptive Systems*. Prentice Hall, Englewood Cliffs, NJ, USA, 1st edition. Rigorous treatment of adaptive systems and excitation conditions.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. (2020). Curriculum learning for reinforcement learning: A survey. *Journal of Machine Learning Research*, 21(181):1–40.

Nobar, M., Keller, J., Rupenyan, A., Khosravi, M., and Lygeros, J. (2024). Guided bayesian optimization: Data-efficient controller tuning with digital twin. *arXiv preprint*. Combines BO with digital twins to reduce real-system experiments.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2nd edition.

Ormoneit, D. and Sen, Š. (2002). Kernel-based reinforcement learning. *Machine learning*, 49:161–178.

Pao, L. Y. and Johnson, K. E. (2009). A tutorial on the dynamics and control of wind turbines and wind farms. In *2009 American Control Conference*, pages 2076–2089. IEEE.

Pao, L. Y., Pusch, M., and Zalkind, D. S. (2024). Control co-design of wind turbines. *Annual Review of Control, Robotics, and Autonomous Systems*, 7(1):201–226.

Pino, F., Schena, L., Rabault, J., and Mendez, M. A. (2023). Comparative analysis of machine learning methods for active flow control. *Journal of Fluid Mechanics*, 958.

Pinosky, A., Abraham, I., Broad, A., Argall, B., and Murphey, T. D. (2022). Hybrid control for combining model-based and model-free reinforcement learning. *The International Journal of Robotics Research*, 42(6):337–355.

Plessix, R.-E. (2006). A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503. Adjoint methods in inverse problems.

Rabault, J. and Kuhnle, A. (2023). *Deep Reinforcement Learning Applied to Active Flow Control*, page 368–390. Cambridge University Press.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279.

Richter, A. V., le Roux, J. D., and Craig, I. K. (2025). Bayesian optimization for automatic tuning of a mimo controller of a flotation bank. *Journal of Process Control*, 147:103388. Example of BO applied to controller tuning in a multivariable control system.

Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*. PhD thesis, University of Cambridge.

Schena, L., Marques, P. A., Poletti, R., Ahizi, S., Van den Berghe, J., and Mendez, M. A. (2024). Reinforcement twinning: From digital twins to model-based reinforcement learning. *Journal of Computational Science*, 82:102421.

Schena, L., Munters, W., Helsen, J., and Mendez, M. A. (2026). Pod-based sparse stochastic estimation of dynamic wind turbine blade deflections. *Journal of Sound and Vibration*, page 119738.

Semeraro, O. (2025). *Reinforcement Learning for Fluid Mechanics: an overview on Fundamentals from a Control Perspective*, page 195–229. von Karman Institute for Fluid Dynamics.

Simon, D. (2006). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, Hoboken, NJ, USA. Comprehensive overview of estimation methods.

Skogestad, S. and Postlethwaite, I. (2005). *Multivariable Feedback Control: Analysis and Design*. Wiley, 2nd edition.

Stengel, R. F. (1994). *Optimal control and estimation*. Dover Publications.

Stockhouse, D., Zalkind, D., Ross, H., and Pao, L. (2024). Analysis of power-maximizing region 2 controllers for wind and marine turbines. In *Journal of Physics: Conference Series*, volume 2767, page 032051. IOP Publishing.

Sutton, R., Barto, A., and Williams, R. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2):19–22.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2 edition.

Söderström, T. and Stoica, P. (1989). *System Identification*. Prentice Hall, Englewood Cliffs, NJ, USA. Comprehensive treatment of linear and nonlinear identification.

Talagrand, O. and Courtier, P. (1987). Variational assimilation of meteorological observations with the adjoint vorticity equation. i: Theory. In *Quarterly Journal of the Royal Meteorological Society*, volume 113, pages 1311–1328. Early work on variational data assimilation.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains. *Journal of Machine Learning Research*, 10(7).

Tsiamis, A. and Pappas, G. J. (2022). Statistical learning theory for control. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:219–246.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Wagg, D. J., Burr, C., Shepherd, J., Xuereb Conti, Z., Enzer, M., and Niederer, S. (2025). The philosophical foundations of digital twinning. *Data-Centric Engineering*, 6.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3–4):279–292.

Wiener, N. (2019). *Cybernetics or Control and Communication in the Animal and the Machine*. The MIT Press.

Xu, Q., Wang, J., Gao, W., Ren, S., and Li, Z. (2024). Digital twin: State-of-the-art and future perspectives. In *Proceeding of the 2024 5th International Conference on Computer Science and Management Technology*, ICCSMT 2024, pages 731–740. ACM.

Zhou, K., Doyle, J. C., and Glover, K. (1996). *Robust and Optimal Control*. Prentice Hall.

Åström, K. J. and Wittenmark, B. (1994). *Adaptive Control*. Addison-Wesley, Boston, MA, USA, 2nd edition. Standard reference on adaptive control.