Reinforcement Twinning: Towards a Bidirectional Learning Framework for Model Updating and Policy Optimization

Miguel A. Mendez*, Lorenzo Schena, Romain Poletti,
Sebastiano Randino, Yannick Lecomte
von Karman Institute for Fluid Dynamics

February 2026

Reinforcement Twinning (RT) is a cyber–physical learning framework that integrates a digital twin and a control agent into a unified feedback loop surrounding a physical system. Its defining characteristic is the simultaneous adaptation of both predictive models and control policies, which supports bidirectional information exchange between modeling and decision-making.

This chapter formalizes RT as a coupled dynamical system comprising a physical plant, a continuously updated twin with parametric closures, and two policy branches—model-based and model-free—coordinated by a supervisory referee. We show how RT subsumes both modeling-to-control and control-to-modeling paradigms, and specify when it reduces to classical optimal control (perfect twin) or to purely data-driven reinforcement learning.

We present two algorithmic implementations. The first combines adjoint-based optimal control for the model-based branch with actor–critic learning (DDPG with prioritized experience replay) for the model-free branch, exposing a gradient-level symmetry that enables gradient exchange, demonstration-driven learning, and policy cloning. The second uses a multi-fidelity probabilistic surrogate of the reward, fusing low-fidelity twin evaluations with sparse high-fidelity experiments via an autoregressive Gaussian-process model. In both cases, we examine the referee's role, conditions for consistency, and how each branch can accelerate or regularize the other.

Redefining the reward as a model-mismatch cost yields a principled control-to-modeling regime in which policy gradients and adjoint sensitivities optimize the same functional along complementary directions. RT thus offers a unified framework in which optimal control, adaptive modeling, and reinforcement learning become interacting components of a single learning architecture rather than competing methodologies.

*mendez@vki.ac.be

Contents

A note on notation and style

1 Engineering in the Age of Real-World Data

Over the past two decades, engineered systems have entered what may reasonably be described as an age of real-world data. Advances in sensing, embedded electronics, wireless communication, and Internet-of-Things (IoT) infrastructures have dramatically increased our ability to observe physical processes in situ and in real time (Lee, 2008; Lee and Seshia, 2015; Kagermann et al., 2013; Ficili et al., 2025). Modern industrial assets, energy systems, transportation platforms, manufacturing lines, and environmental infrastructures are densely instrumented with sensor networks that continuously stream measurements of temperature, pressure, vibration, flow, position, and operational states.

Surveys of IoT and cyber-physical systems document the rapid proliferation of connected devices. Industrial IoT emphasizes real-time monitoring, predictive maintenance, and data-driven optimization as core features of current engineering practice (Atzori et al., 2010; Gubbi et al., 2013; Sicari et al., 2015; Tao et al., 2019). Systems once sparsely measured are now continuously observed at high spatial and temporal resolution.

The central question is how to leverage this observational richness. A data-rich environment has the potential to reshape the entire engineering workflow—from design and modeling to monitoring, diagnosis, and control.

This expanded sensing capability arrives at a moment when engineering systems are being pushed into increasingly demanding operational regimes, under off-design, uncertain, and rapidly evolving conditions. For example, offshore wind farms on floating platforms are exposed to wave-induced loads, and atmospheric variability (Veers et al., 2019; Lackner, 2009). Unmanned aerial vehicles must maneuver reliably in disturbed and cluttered environments with rapidly changing flow conditions (Kumar and Michael, 2012). Emerging green propulsion systems, including cryogenic storage and transport technologies, operate under large thermal and dynamical excursions that challenge traditional design assumptions (Adler and Martins, 2023). These examples, drawn primarily from thermofluid systems, illustrate a broader trend. Similar challenges arise across power grids integrating intermittent renewables, electrochemical storage systems subject to degradation, chemical process plants operating under fluctuating feed conditions, or autonomous systems navigating uncertain environments to mention examples from other fields.

In all these engineering systems, the dominant physical processes can be described by first-principles models derived from conservation laws and constitutive relations. However, high-fidelity simulations of these processes are typically too computationally expensive for operational decision-making and real-time flow control. Fast reduced-order models are needed. Among the available options, the most effective are those that retain as much physical structure as possible, embedding conservation principles together with phenomenological coefficients (e.g. aerodynamic load models, damping terms, turbulence closures, reaction rates, or heat- and mass-transfer coefficients). Yet these coefficients are often derived from empirical correlations obtained under laboratory conditions that may differ significantly from operational regimes. Regardless of the approach, the predictive performance of any reduced-order model deteriorates far from the conditions in which it is derived. This raises a central challenge: how can predictive models be updated or adapted online so that control decisions remain both physically consistent and operationally effective?

The technical ingredients to address this challenge are largely available. However, a

shift in mindset is required. Models, measurements, and control laws can not be treated as loosely coupled components assembled in sequence but must be embedded within adaptive feedback architectures capable of continuously integrating information from the physical system. It is within this broader transformation of engineering practice that the concept of the digital twin has gained prominence.

In its most general formulation, a digital twin is a virtual representation of a physical asset that is continuously updated using real-time data and intended to support monitoring, prediction, and decision-making throughout the asset's lifecycle (Tao et al., 2019; Jones et al., 2020; Wagg et al., 2025). Unlike traditional simulation models, which are typically constructed offline and used for scenario analysis, a digital twin is meant to remain dynamically coupled to its physical counterpart during operation. Crucially, the twin is not merely descriptive: as emphasized in recent analyses of the concept (Korenhof et al., 2021; Wagg et al., 2025), it must become an acting—or steering—representation, capable of taking decisions and influence the evolution of the physical process itself. This closes the loop between representation and intervention.

From a control-theoretic perspective, this is a fundamental architectural shift. The digital model becomes an active element of the closed-loop system, and the real system becomes a source of epistemic gain. The twin can drive control actions and control actions can improve the twin. The paradigm thus shifts from modeling to enable control toward control-informed modeling. To operationalize this bidirectional coupling, model adaptation and control improvement must be coordinated within a unified learning architecture. Reinforcement twinning, first introduced in (Schena et al., 2024) and developed within the ERC project "Re-Twist", formalizes this idea by embedding policy optimization and model refinement within a unified feedback structure.

The following sections define reinforcement twinning (Section 2) and examine its conceptual roots (Section3).

2 What is Reinforcement Twinning?

Reinforcement twinning is a cyber-physical learning architecture in which a digital twin and a control agent operate as a coupled unit around a physical system. Its defining feature is the simultaneous adaptation of the predictive model and the control policy within the same feedback loop.

Figure 1 illustrates the reinforcement twinning (RT) unit and its interaction with the real system. The RT unit is composed of two adaptive components: a digital twin and an acting agent, persistently coupled to a physical system. Their role is described below.

The Real System. The real system represents the physical process or engineering asset under consideration. It evolves according to unknown or partially known dynamics and produces real-time measurements. These measurements enable model–data alignment within the twin and provide performance feedback—such as reward, cost, constraint violation, or task completion signals—to the agent. The system receives real actions from the agent, which determine its trajectory and the operating regimes that are explored.

The Digital Twin. The digital twin offers a predictive model of the real system. It embodies a closure structure or model class whose parameters are tuned to match observed behavior, aiming to improve predictive accuracy. The twin generates internal predictions

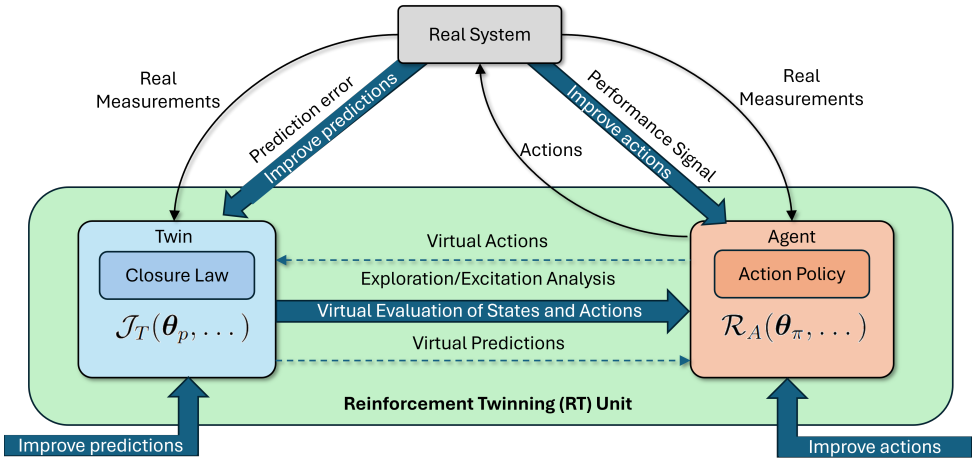

Figure 1: Schematic overview of the *Reinforcement Twinning* (RT) framework. The reinforcement twinning unit links a digital twin and an acting agent to a real system. The real system provides two feedback signals: measurement data for correcting the model and performance signals for improving the policy. The agent queries the twin to evaluate candidate actions, while its chosen actions shape the system's operating conditions and thus the data available to update the twin.

of state evolution and system response, which the agent can use for simulation, planning, and counterfactual analysis.

The Acting Agent. The acting agent implements a policy that selects actions applied to the real system. Its objective is to improve performance according to a specified criterion. The agent receives performance feedback from the system and may exploit the twin's predictions for virtual evaluation of candidate actions.

Two limiting operating regimes can be distinguished within this architecture, illustrated in Figures 2 and 3. In the *modeling-to-control* regime (Figure 2), the digital twin primarily serves the control agent: predictive simulations and model updates are exploited to improve control performance. This configuration encompasses established methodologies from adaptive control, model predictive control, and model-based reinforcement learning. In the *control-to-modeling* regime (Figure 3), the emphasis is reversed: the agent seek to excite the system to generate more informative data, thereby accelerating model refinement. This regime connects to dual control, active system identification, and information-driven exploration strategies. These conceptual roots are analyzed in more detail in Sections 3.1 and 3.2.

Internal Virtual Interaction. Within the RT unit, the twin and agent interact via a virtual channel. The agent evaluates states and actions with the twin's predictive model, while twin updates alter the landscape in which the agent optimizes its policy. This complements physical feedback and enables simultaneous policy optimization and model refinement.

External Learning Coupling. An RT unit may run alone or within a network of twinning units. External feedback can involve shared model updates (Marques et al., 2024), shared policy updates, or aggregated performance signals across assets. This modularity lets reinforcement twinning scale from individual systems to fleets of interacting assets.

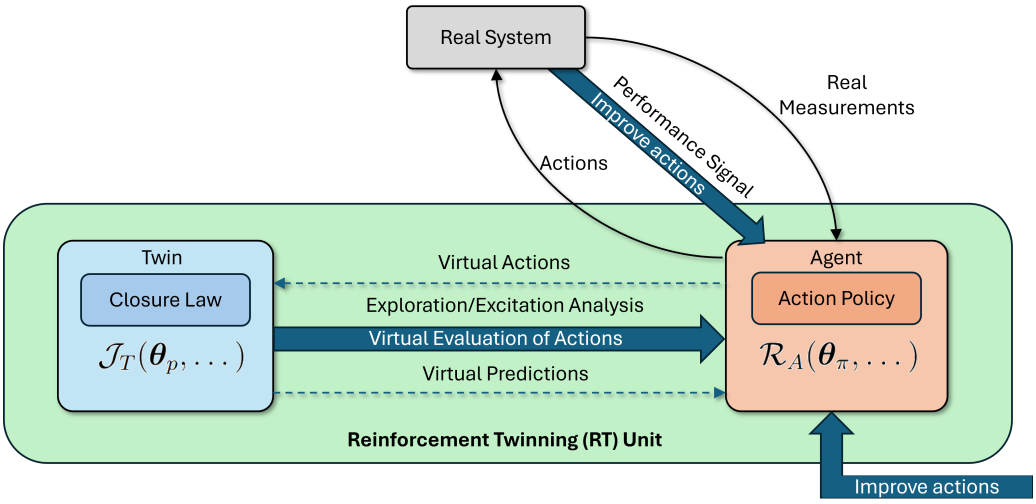

Figure 2: Modeling-to-control regime of reinforcement twinning. The digital twin primarily serves the acting agent: predictive simulations and model updates are exploited to improve control performance. This mode encompasses adaptive control, model predictive control, and model-based reinforcement learning, where models are used to synthesize or optimize control actions (see Section 3.1).

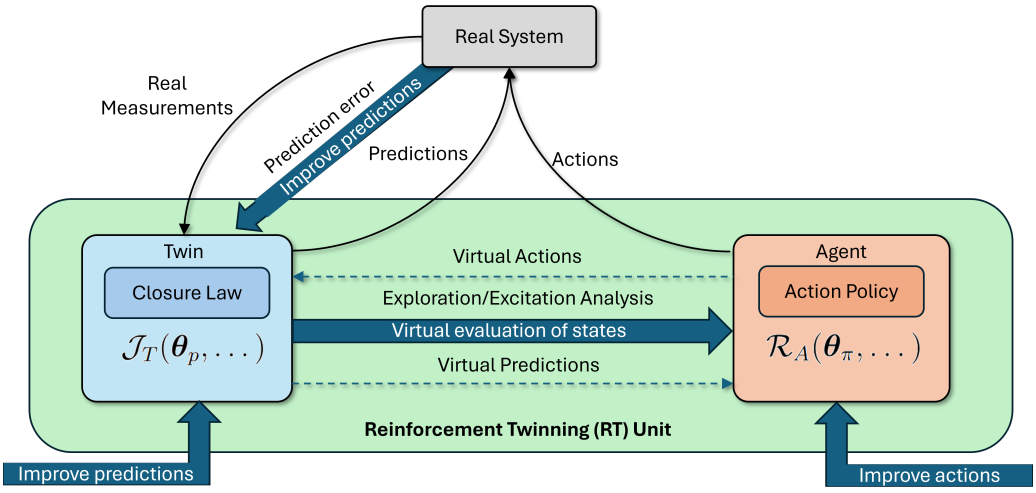

Figure 3: Control-to-modeling regime of reinforcement twinning. Action selection emphasizes informative excitation of the system to improve model fidelity. This mode includes dual control, active identification, and information-theoretic or curiosity-driven exploration approaches (see Section 3.2.

Reinforcement twinning is defined by the coexistence and coordination of these mechanisms within a single cyber-physical learning structure: (i) predictive correction through model–data alignment, (ii) performance-driven policy improvement, and (iii) virtual model-based evaluation and exploration shaping.

3 Conceptual Roots

The reinforcement twinning architecture lies at the intersection of several established research traditions concerned with the interaction between modeling and control. Historically, these traditions have emphasized one of two complementary directions. In the *modeling-to-control* view, predictive models are built or updated to design or improve control actions. In the *control-to-modeling* view, control actions are chosen to improve system identification, reduce uncertainty, or clarify system structure.

Reinforcement twinning offers a unifying framework in which these two directions appear as limiting cases of a single cyber-physical learning architecture. Instead of treating modeling and control as sequential or separate, it couples predictive refinement and policy optimization within a persistent feedback loop. The main methods for each direction are reviewed in Sections 3.1 and 3.2.

3.1 Modeling-to-Control

The modeling-to-control paradigm is historically the dominant framework in control theory, and particularly in flow control. In this perspective, a predictive model of the underlying dynamics is constructed and subsequently used to derive an optimal control law. We covered this framework in lecture 5, Section 2.

In fluid mechanics, this tradition spans a broad spectrum of modeling strategies. At one end lie analytical or semi-analytical reductions of the governing equations, enabling optimal control formulations directly at the PDE level (Sritharan, 1998; Gunzburger, 2002). The associated control problems are typically addressed using Pontryagin's Maximum Principle or adjoint-based optimization techniques (Dixon, 1981; Carnarius et al., 2011), or, in principle, through Hamilton–Jacobi–Bellman (HJB) formulations (Peng, 1992; Bardi and Capuzzo-Dolcetta, 1997; Stengel, 1994). These approaches provide a rigorous mathematical foundation, with explicit optimality conditions.

At higher Reynolds numbers or in complex geometries, direct optimal control of the full-order system becomes computationally prohibitive. This has motivated the development of projection-based reduced-order models (ROMs), which approximate the high-dimensional flow dynamics on low-dimensional manifolds extracted from dominant coherent structures (Ravindran, 2000; Rowley and Dawson, 2017). Model reduction enables tractable design of linear-quadratic regulators (LQR) and model predictive controllers (MPC) for fluid flows (see Leugering et al. (2011); Schwenzer et al. (2021) among others). In these settings, the model acts as a surrogate for the true dynamics, allowing control design to proceed offline with limited direct interaction with the physical system.

The strength of this modeling-to-control family lies in its mathematical structure and sample efficiency. Once a sufficiently accurate model is available, optimal control policies can be synthesized with minimal experimental interaction. However, practical performance depends critically on model fidelity. In real engineering systems, parameters drift, boundary conditions change, and unmodeled phenomena emerge, giving rise to the well-known model–plant mismatch problem (Badwe et al., 2010). The resulting "reality gap" between simulation and operation can degrade performance.

To mitigate these effects, system identification and data assimilation techniques are commonly employed to update model parameters or states using measurement data

(Ljung, 1999; Asch et al., 2016). In fluid flows, data-driven closure modeling and machine-learning-enhanced reduced-order models further extend this idea (Chiuso and Pillonetto, 2019; Buizza et al., 2022; Abarbanel et al., 2018). Alternatively, adaptive control strategies adjust controller parameters online to compensate for model uncertainties (Åström and Wittenmark, 2008). In all these cases, the objective remains performance-oriented: model updates are performed to preserve or improve control quality.

More recently, model-based reinforcement learning (MBRL) has revived the modeling-to-control principle within the machine-learning community (Sutton and Barto, 1998; Polydoros and Nalpantidis, 2017; Moerland et al., 2023). Learned dynamics models are exploited to generate synthetic rollouts, warm-start policies as we saw in tutorial 4 of lecture 5, or guide value-function estimation. In fluid mechanics, reinforcement learning has been successfully applied to active flow control problems (Werner et al., 2023; Pino et al., 2023), often relying on simulators for policy training before deployment. Hybrid approaches further combine model-based structure with model-free flexibility, for instance through residual policy learning or model-guided initialization (Bansal et al., 2017; Nagabandi et al., 2018; Johannink et al., 2019).

Across this broad spectrum of methodologies—from PDE-constrained optimal control to reduced-order modeling, adaptive control, and model-based reinforcement learning—the unifying principle remains consistent: predictive models are constructed and refined in order to synthesize effective control actions. Reinforcement twinning encompasses this regime when the digital twin primarily serves as a predictive engine supporting policy optimization (Figure 2).

3.2 Control-to-Modeling

In the control-to-modeling paradigm, control actions are chosen to improve system knowledge. Standard adaptive control adopts this idea but largely separates model refinement from control synthesis. This separation leads to the certainty-equivalence principle, where estimated parameters are treated as true when computing control actions (Åström and Wittenmark, 1995; Ioannou and Sun, 1996). As a result, it often ignores that actions stabilizing the system may not reduce model uncertainty. Learning can then remain passive and confined to a narrow range of operating conditions (Feldbaum, 1960), as discussed in Lecture 5, Section 4.

Many approaches acknowledge uncertainty without actively seeking to reducing it. Early examples include central-tendency controllers, which bias adaptive laws toward conservative parameter estimates to mitigate risk (Ryall and Moore, 1989; Moore et al., 1989). More recent uncertainty-aware control and learning methods aim to maximize the probability of meeting performance and safety objectives given explicit uncertainty estimates, e.g., robust or risk-sensitive MPC and uncertainty-aware reinforcement learning (Dehkordi et al., 2025; Kahn et al., 2017). Although effective for safety, these frameworks are mostly exploitative: they manage uncertainty without explicitly choosing actions to gain information.

Another line of work enforces exploration via *persistent excitation*. Classic examples include adaptive pole-positioning (Anderson and Johnstone, 1985) and extremum-seeking control, where dithering signals probe and improve performance (Krstić, 2000; Ariyur and Krstic, 2003). A key limitation is the lack of clear principles for *when* to excite

the system and *how* to design probing signals; excitation remains heuristic and most importantly, problem-dependent.

In the case of sloshing dynamics, for example small-amplitude oscillations may suffice for feedback regulation but are generally inadequate for identifying damping or nonlinear free-surface effects. In contrast, deliberate excitation over selected frequency bands improves model fidelity by exposing more of the system's dynamic response. The same applies to variable-pitch propellers: steady operation at an optimal thrust reveals little about aerodynamic coefficients, whereas transient pitch maneuvers produce richer dynamics that facilitate parameter estimation. Likewise, running a wind turbine only near its optimal tip-speed ratio yields nor information about off-design aerodynamics (see tutorials 2 and 3 of lecture 5), while deliberate excursions—such as brief operation at high induction factors—reveal nonlinear load responses that enable more accurate models. Similarly, tightly regulated cryogenic storage units, though vital for safety and nominal performance, can mask heat-transfer nonlinearities that appear only during controlled thermal transients or dynamic excitations such as sloshing.

A more principled framework linking regulation and learning is *dual control*, which chooses control actions to jointly optimize performance and improve state and/or parameter estimates (Feldbâum, 1963; Wittenmark, 1995; Mesbah, 2018). In principle, dynamic programming addresses this trade-off by valuing actions that reduce uncertainty when this lowers expected long-term cost (Bertsekas, 2005; Chen et al., 2021). Probing becomes optimal whenever its expected future benefit exceeds its short-term performance loss. However, exact dynamic programming is computationally intractable for most coupled estimation–control problems (Åström and Helmersson, 1986; Bernhardsson, 1989).

Practical methods approximate the exploration–exploitation trade-off with surrogate mechanisms. A common approach is to augment the control objective with explicit estimation-quality terms, coupling regulation and identification in one criterion (Goodwin and Payne, 1977; Wittenmark and Elevitch, 1985). In many applications, the balance between performance and information gathering is enforced indirectly via surrogate costs, separate identification phases, or heuristic excitation.

Beyond dual control, related fields also exemplify the control-to-modeling principle. In adaptive system identification and optimal experiment design, input trajectories are optimized to maximize information or parameter identifiability (Goodwin and Payne, 1977; Ljung, 1999), with control signals chosen to improve the model rather than regulate. In reinforcement learning, curiosity-driven and intrinsic-motivation methods reward actions that reduce prediction error or increase model uncertainty, formalizing information-seeking in high-dimensional spaces (Schmidhuber, 2010; Pathak et al., 2017). Though often framed within AI, these approaches share a core idea: control shapes the data distribution to improve predictions.

Model-based fault detection and diagnosis provide a related view (Isermann, 2006; Blanke et al., 2016). Comparing model predictions with measurements enables residual-based fault detection, and in active fault diagnosis, control inputs are designed to increase diagnostic sensitivity. The goal shifts from regulation to structural discrimination. In sloshing tanks, specific excitation patterns can amplify residuals indicating structural changes such as altered boundary conditions or internal damping. In propulsion systems, deliberate pitch modulation in a VPP can increase sensitivity to blade asymmetries or actuator degradation. Here, control actions enhance diagnostic observability.

Across these traditions, the idea is consistent: actions influence knowledge. The control-to-modeling regime of reinforcement twinning generalizes this by embedding information seeking behavior.

4 General Mathematical Formulation

Reinforcement twinning can be formalized in discrete time as a coupled cyber–physical dynamical system composed of three layers: the real system, the digital twin, and a set of candidate policies evaluated within the twin. Figure 4 summarizes this structure schematically, highlighting the coexistence of a physical trajectory and multiple virtual rollouts within the twin, together with the referee mechanism that selects which action is applied to the plant. The extension of this formulation to continuous time is natural, replacing the discrete-time updates of the plant, twin, and candidate policies with the corresponding flows generated by their continuous-time dynamics.

4.1 Structural Components

Real System (Black-Box Layer). The physical system evolves according to unknown dynamics,

$$\begin{cases} \boldsymbol{s}_{\bullet,k+1} = F_{\bullet}(\boldsymbol{s}_{\bullet,k},\, \boldsymbol{z}_k,\, \boldsymbol{a}_k), \\ \boldsymbol{o}_{\bullet,k} = h_{\bullet}(\boldsymbol{s}_{\bullet,k}), \end{cases} \tag{1}$$

where $\boldsymbol{s}_{\bullet,k} \in \mathbb{R}^{n_s}$ denotes the true and is usually not available, $\boldsymbol{z}_k \in \mathbb{R}^{n_z}$ denotes exogenous inputs or disturbances and $\mathbf{a}_k \in \mathbb{R}^{n_a}$ denotes the control action. We return to this later. The policy governing these actions and the training of the digital twin can only rely on partial, indirect and usually noisy observations $\boldsymbol{o}_{\bullet,k} \in \mathbb{R}^{n_o}$, which are linked to the actual state via an unknown observation function $h_{\bullet}()$.

Digital Twin (White-Box Layer). The digital twin provides a structured predictive model in which the dominant physics are represented explicitly, while unresolved or unmodeled effects are captured through a data-driven closure law. In the notation of Lecture 5, we introduce a closure variable $\boldsymbol{p}_k$ generated by a parametric closure model

$$\boldsymbol{p}_k = g(\boldsymbol{s}_{\circ,k},\, \boldsymbol{z}_k,\, \boldsymbol{a}_k \mid \boldsymbol{\theta}_p), \tag{2}$$

where $\boldsymbol{\theta}_p$ denotes the closure parameters to be learned and $\boldsymbol{s}_{\circ,k}$ denotes the full state of the system. This could be inferred from the partial observation using techniques from data assimilation or could be provided by the digital twin, which evolves it according to a fully known dynamics

$$\begin{cases} \boldsymbol{s}_{\circ,k+1} = F_{\circ}(\boldsymbol{s}_{\circ,k},\, \boldsymbol{z}_k,\, \boldsymbol{a}_k), \\ \boldsymbol{o}_{\circ,k} = h_{\circ}(\boldsymbol{s}_{\circ,k}), \end{cases} \tag{3}$$

These "twin" states could also be corrected using the real observations $\boldsymbol{o}_{\bullet,k}$ using standard statistical tools such as Kalman filters. The virtual observations $\boldsymbol{o}_{\circ,k} = h_{\circ}(s_{\circ,k})$ are designed to mimic the measurement process from the real world and one could expect $h_{\circ} \neq h_{\bullet}$ in general.

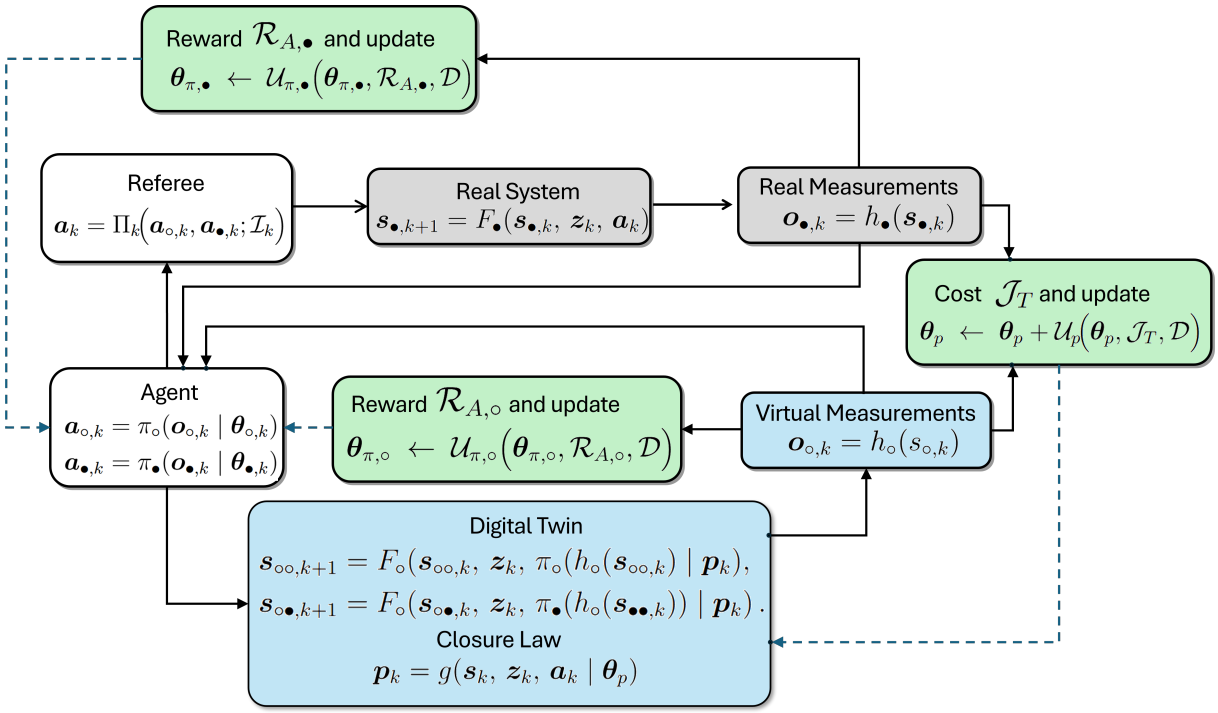

Figure 4: Reinforcement twinning architecture coupling a digital twin and two candidate policies around a real system. Prediction error updates the twin via $\mathcal{J}_T$, while rewards $\mathcal{R}_{A,\circ}$ and $\mathcal{R}_{A,\bullet}$ update the policies.

Candidate Control Policies. Two candidate policies are considered: one relying from interaction with the real system and another leveraging the digital twin. These are defined as follows

$$\boldsymbol{a}_{\circ,k} = \pi_\circ(\boldsymbol{o}_{\circ,k} \mid \boldsymbol{\theta}_{\pi,\circ}), \tag{4}$$
$$\boldsymbol{a}_{\bullet,k} = \pi_\bullet(\boldsymbol{o}_{\bullet,k} \mid \boldsymbol{\theta}_{\pi,\bullet}), \tag{5}$$

In general, the two policies need not coincide. For instance, one could choose $\pi_\circ$ to be a structured model-based controller (e.g., LQR, MPC), while letting $\pi_\bullet$ denote an artificial neural network trained via reinforcement learning. In the remainder, we streamline the exposition by focusing on the case in which these two policies are represented by the same function, a setting that admits a substantially broader spectrum of possible interaction mechanisms during their joint synthesis.

Parallel Twin Rollouts. The twin might be used to evaluate both the model-free and the model-based policies. This means that at each time step this can be evolved under both candidate actions, hence giving rise to two virtual trajectories:

$$\boldsymbol{s}_{\circ\circ,k+1} = F_\circ(\boldsymbol{s}_{\circ\circ,k}, \boldsymbol{z}_k, \boldsymbol{a}_{\circ\circ,k} = \pi_\circ(h_\circ(\boldsymbol{s}_{\circ\circ,k}) \mid \boldsymbol{\theta}_{\pi,\circ}) \mid \boldsymbol{p}_k), \tag{6}$$
$$\boldsymbol{s}_{\circ\bullet,k+1} = F_\circ(\boldsymbol{s}_{\circ\bullet,k}, \boldsymbol{z}_k, \boldsymbol{a}_{\circ\bullet,k} = \pi_\bullet(h_\circ(\boldsymbol{s}_{\circ\bullet,k}) \mid \boldsymbol{\theta}_{\pi,\bullet}) \mid \boldsymbol{p}_k)\,. \tag{7}$$

Computing for each of these the reward $\mathcal{R}_{A,\circ}$ allows to evaluate the quality of the two policies according to the digital twin; we return to the reward definition in the following section. It is worth stressing that only one of the two actions is actually applied to the real system (1) according to the referee mechanism described below. Therefore, only the

rollout associated with the action applied to the real world can be expected to be aligned with the physical evolution and thus used to compute tracking errors and the digital twin performance.

Referee Mechanism (Live vs. Idle Policy). At the beginning of each episode, the referee identifies the policy that will interact with the real system and thus define the actions in (1). Many options are possible and deserves detailed investigation. Similarly to Schena et al. (2024); Poletti et al. (2025), the approach presented in these notes considers

$$\boldsymbol{a}_k = \Pi_{(e)}\Big(\pi_\bullet(\boldsymbol{o}_{\bullet,k} \mid \boldsymbol{\theta}_{\pi,\bullet}), \pi_\circ(\boldsymbol{o}_{\bullet,k} \mid \boldsymbol{\theta}_{\pi,\circ}) \mid \mathcal{I}_{(e)}\Big), \tag{8}$$

where $\mathcal{I}_{(e)}$ denotes supervisory information such as safety constraints, confidence measures, or predicted performance. In this formulation the selected policy is applied to real observations and remains active for the full episode. Alternative strategies may allow step-wise switching or arbitration based on virtual states or state-estimation outputs. The non-selected policy remains idle with respect to the plant but continues to be evaluated within the twin. The overall idea is to allow for the possibility that, during the policy identification one of the

The rationale behind maintaining two policies in parallel is that the idle policy, although not interacting directly with the real system, continues to improve through twin rollouts and indirect information. When its predicted or observed performance exceeds that of the live policy, the referee can promote it to govern the plant. This virtuous loop, in which the learner may eventually surpass the master, was observed in Schena et al. (2024).

4.2 Learning Objectives and Parameter Updates

The current reinforcement twinning framework is formulated in an episodic setting. Over an episode involving n_t iterations (assumed for simplicity to be equally spaced in time), the real system generates a sequence of observations $\{\boldsymbol{o}_{\bullet,k}\}_{k=1}^{n_t}$ under the applied action sequence $\{\boldsymbol{a}_k\}_{k=1}^{n_t}$ selected by the referee according to the unknown dynamics (1) and driven by a realized disturbance sequence $\{\boldsymbol{z}_k\}_{k=1}^{n_t}$. In parallel, the digital twin generates the two virtual trajectories (6)–(7).

Digital Twin cost functional. Denoting by $\star \in \{\circ, \bullet\}$ the index of the live policy selected at the beginning of the episode, and by $\{\boldsymbol{s}_{\circ\star,k}\}_{k=1}^{n_t}$ the twin trajectory computed using the same applied actions and disturbance inputs, the identification cost for a single episode is defined as

$$\mathcal{J}_T^{(e)}(\boldsymbol{\theta}_p \mid \boldsymbol{\theta}_{\pi,\circ}, \boldsymbol{\theta}_{\pi,\bullet}) = \sum_{k=0}^{n_t} \ell_{\text{pred}}\Big(\boldsymbol{o}_{\bullet,k}, \boldsymbol{o}_{\circ,k}(\boldsymbol{\theta}_p)\Big), \tag{9}$$

where $\boldsymbol{o}_{\circ,k} = h_\circ(\boldsymbol{s}_{\circ\star,k})$ and $\ell_{\text{pred}}(\cdot)$ measures the prediction error. If the disturbance sequence is not directly measurable, a disturbance model must be introduced (e.g. via estimation or data assimilation), and the expected identification objective over a disturbance distribution $\boldsymbol{z} \sim \mathcal{Z}$ may be considered.

The training of the digital twin aims to minimize this cost. Hence twin parameter updates take the form

$$\boldsymbol{\theta}_p \leftarrow \mathcal{U}_p\Big(\boldsymbol{\theta}_p, \mathcal{J}_T, \mathcal{D}_T\Big), \tag{10}$$

where $\mathcal{D}_T$ denotes the dataset used for identification. To increase the learning stability this could consist of a large set of transitions $\mathcal{D}_T^{(e)} = \{\boldsymbol{o}_{\bullet,k}, \boldsymbol{a}_k, \boldsymbol{z}_k\}_{k=1}^{n_t}$. collected at each episode, hence

$$\mathcal{D}_T = \bigcup_{e=1}^{N_e} \mathcal{D}_T^{(e)},$$

Sampling from this dataset allows for stochastic gradient updates and mitigation of catastrophic forgetting, in complete analogy with the memory buffer mechanism employed in actor-critic reinforcement learning.

Policy Reward functionals. At the same time, the acting policies are driven by accumulated rewards reflecting control performance. Denoting as $r(\boldsymbol{s}, \boldsymbol{a})$ the stage reward encoding encode regulation quality, tracking accuracy, energy efficiency, constraint satisfaction, or economic performance, the cumulative reward driving the model-free policy search is

$$\mathcal{R}_{A,\bullet} = \sum_{k=0}^{n_t-1} r\Big(\boldsymbol{s}_{\bullet,k}, \boldsymbol{a}_k\Big). \tag{11}$$

This drives the update of the model-free policy parameter

$$\boldsymbol{\theta}_{\pi,\bullet} \leftarrow \mathcal{U}_{\pi,\bullet}\Big(\boldsymbol{\theta}_{\pi,\bullet}, \mathcal{R}_{A,\bullet}, \mathcal{D}_\bullet\Big), \tag{12}$$

with transitions obtained from the memory buffer

$$\mathcal{D}_\bullet = \Big\{(\boldsymbol{o}_{\bullet,k}, \boldsymbol{a}_k, r_k, \boldsymbol{o}_{\bullet,k+1})\Big\}_{k=1}^{N_{\text{tot}}}, \tag{13}$$

where N_{tot} denotes the total number of collected transitions across episodes on the real system. Note that the model-free update is carried out using data from the real system regardless of whether the interaction is provided by the model-free or the model based policy; this is one of the main mechanism of interaction between the two policy search methods. Similarly, the cumulative reward driving the model-based policy search is computed from the "white rollout" in (6), i.e.

$$\mathcal{R}_{A,\circ} = \sum_{k=0}^{n_t-1} r\Big(\boldsymbol{s}_{\circ\circ,k}, \boldsymbol{a}_{\circ,k}\Big). \tag{14}$$

Policy parameters are updated according to

$$\boldsymbol{\theta}_{\pi,\circ} \leftarrow \mathcal{U}_{\pi,\circ}\Big(\boldsymbol{\theta}_{\pi,\circ}, \mathcal{R}_{A,\circ}, \mathcal{D}_\circ\Big), \tag{15}$$

where $\mathcal{D}_\circ$ denotes the virtual transition buffer generated by the twin rollouts,

$$\mathcal{D}_\circ = \Big\{(\boldsymbol{o}_{\circ,k}, \boldsymbol{a}_{\circ,k}, r_{\circ,k}, \boldsymbol{o}_{\circ,k+1})\Big\}_{k=1}^{N_\circ}, \tag{16}$$

with $r_{\circ,k} = r(\boldsymbol{s}_{\circ\circ,k}, \boldsymbol{a}_{\circ,k})$.

In contrast to $\mathcal{D}_\bullet$, which collects real transitions, the buffer $\mathcal{D}_\circ$ is generated entirely within the digital twin and may be expanded arbitrarily at negligible cost.

5 Current Reinforcement Twinning Schemes

The structural elements introduced in the previous section define a general Reinforcement Twinning architecture that is independent of the specific algorithms used in each block. The separation between real environment, digital twin, policy update, and shared learning feedback provides a modular framework in which different identification and control tools can operate coherently.

In practice, the closure update driven by $\mathcal{J}_T$ may rely on any nonlinear identification method, while maximization of $\mathcal{R}_{A,\circ}$ typically exploits analytic structure available inside the twin (e.g. linearization-based synthesis, adjoint methods, or MPC). In parallel, maximization of $\mathcal{R}_{A,\bullet}$ may be performed using model-free approaches such as policy search, actor–critic methods, or Bayesian optimization. Improvements in $\mathcal{R}_{A,\circ}$ provide inexpensive but potentially biased guidance on operational performance, whereas $\mathcal{R}_{A,\bullet}$ provides reliable feedback from the real system, possibly revealing behaviors beyond the model formulation, albeit at the cost of a large number of interaction with the real system. The central design question is therefore how structured model-based updates and data-driven updates should be integrated so as to exchange information in a mutually supportive manner.

In this section, we focus on the modeling-to-control regime, in which improvements in $\mathcal{R}_{A,\star}$ are guided by the predictive structure of the twin and interaction with the real system while $\mathcal{J}_T$ maintains consistency with reality. The reverse control-to-modeling regime, where actions are selected to accelerate reduction of $\mathcal{J}_T$, is only briefly outlined. We restrict attention to a single reinforcement twinning unit and postpone collective feedback mechanisms to a later discussion. Two approaches are considered: one blending optimal control with actor–critic learning (Sec. 5.1) and the other based on multi-fidelity policy search (Sec. 5.2).

In both schemes, closure identification (minimization of $\mathcal{J}_T$) and model-based policy improvement (maximization of $\mathcal{R}_{A,\circ}$) are carried out using the adjoint-based formalism introduced in Lecture 5. Gradients with respect to closure and policy parameters are obtained through the twin dynamics and are therefore not discussed further. The distinguishing elements lie instead in (i) how the model-free policy associated with $\mathcal{R}_{A,\bullet}$ is constructed, and (ii) how it interacts with the model-based branch.

5.1 Adjoint Optimal Control and Actor–Critic Learning

5.1.1 Formulation for the model free search

The formulation presented in this section is the one originally introduced in Schena et al. (2024) and refined in Poletti et al. (2025). In this setting the model-free search for the policy is carried out using the Deep Deterministic Policy Gradient (DDPG) algorithm Lillicrap et al. (2015), augmented with Prioritized Experience Replay (Schaul et al., 2015). This method falls into the actor–critic reinforcement learning formalism (Grondman et al., 2012), hence combining the ideas introduced in Section 5 of Lecture 5: the algorithm trains both a critic (i.e. a model seeking to approximate the state–action value function Q) and an actor (that is, a policy) mapping states to actions. Both are typically parametrized using artificial neural networks.

Parametrization. The actor is represented as a deterministic policy

$$\boldsymbol{a}_{\bullet,k} = \pi_\bullet(\boldsymbol{o}_{\bullet,k} \mid \boldsymbol{\theta}_{\pi,\bullet}), \tag{17}$$

where $\boldsymbol{\theta}_{\pi,\bullet}$ denotes the weights of the neural network. The critic approximates the action–value function

$$Q(\boldsymbol{o}, \boldsymbol{a}; \boldsymbol{\theta}_Q), \tag{18}$$

where $\boldsymbol{\theta}_Q$ collects the weights of the critic network.

Critic update (Bellman regression). A replay buffer (memory) is used to store transitions

$$(\boldsymbol{o}_{\bullet,k}, \boldsymbol{a}_k, r_k, \boldsymbol{o}_{\bullet,k+1})\,.$$

The critic is then trained by minimizing the Bellman residual on the collected transitions

$$\mathcal{L}_Q(\boldsymbol{\theta}_Q) = \frac{1}{N} \sum_k \left(Q(\boldsymbol{o}_{\bullet,k}, \boldsymbol{a}_k; \boldsymbol{\theta}_Q) - y_k\right)^2, \tag{19}$$

where the target is defined as

$$y_k = r_k + \gamma\, Q\Big(\boldsymbol{o}_{\bullet,k+1}, \pi_\bullet(\boldsymbol{o}_{\bullet,k+1} \mid \boldsymbol{\theta}^{\text{targ}}_{\pi,\bullet}); \boldsymbol{\theta}^{\text{targ}}_Q\Big). \tag{20}$$

Here $\boldsymbol{\theta}^{\text{targ}}_{\pi,\bullet}$ and $\boldsymbol{\theta}^{\text{targ}}_Q$ denote target-network parameters that evolve slowly according to

$$\boldsymbol{\theta}^{\text{targ}} \leftarrow \tau\, \boldsymbol{\theta} + (1-\tau)\, \boldsymbol{\theta}^{\text{targ}}, \qquad 0 < \tau \ll 1, \tag{21}$$

which improves numerical stability by reducing oscillations in the Bellman targets.

Sampling the next $\boldsymbol{\theta}_{\pi,\bullet}$ Once the critic approximates Q, the actor is updated following a stochastic gradient descent approach like ADAM (see lecture 5). Using a batch of size N, the gradient can be computed as

$$\nabla_{\boldsymbol{\theta}_{\pi,\bullet}} \mathcal{R}_{A,\bullet} \approx \frac{1}{N} \sum_k \nabla_{\boldsymbol{\theta}_{\pi,\bullet}} \pi_\bullet(\boldsymbol{o}_{\bullet,k}) \nabla_{\boldsymbol{a}} Q(\boldsymbol{o}_{\bullet,k}, \boldsymbol{a}; \boldsymbol{\theta}_Q)\Big|_{\boldsymbol{a}=\pi_\bullet(\boldsymbol{o}_{\bullet,k})}. \tag{22}$$

The update follows the deterministic policy gradient introduced in previous lectures and avoids direct differentiation through the system dynamics. The gradient based update has the same structure for the model based policy, which follows the adjoint based formalism introduced in lecture 5. That is, the virtual reward $\mathcal{R}_{A,\circ}$ (14) is differentiated through the twin dynamics $\boldsymbol{s}_{\circ,k+1} = F_\circ(\boldsymbol{s}_{\circ,k}, \boldsymbol{z}_k, \boldsymbol{a}_{\circ,k}; \boldsymbol{\theta}_p)$, leading to the gradient expression

$$\nabla_{\boldsymbol{\theta}_{\pi,\circ}} \mathcal{R}_{A,\circ} = \sum_{k=0}^{n_t-1} \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{a}_{\circ,k}} \frac{\partial \pi_\circ(\boldsymbol{o}_{\circ,k})}{\partial \boldsymbol{\theta}_{\pi,\circ}}, \tag{23}$$

where the discrete Hamiltonian is defined as

$$\mathcal{H}_k = r(\boldsymbol{s}_{\circ,k}, \boldsymbol{a}_{\circ,k}) + \boldsymbol{\xi}_{k+1}^\top F_\circ(\boldsymbol{s}_{\circ,k}, \boldsymbol{z}_k, \boldsymbol{a}_{\circ,k}; \boldsymbol{\theta}_p),$$

and the adjoint variables $\boldsymbol{\xi}_k$ evolve backward in time according to

$$\boldsymbol{\xi}_k = \frac{\partial r}{\partial \boldsymbol{s}_{\circ,k}} + \left(\frac{\partial F_\circ}{\partial \boldsymbol{s}_{\circ,k}}\right)^\top \boldsymbol{\xi}_{k+1}, \qquad \boldsymbol{\xi}_{n_t} = 0, \tag{24}$$

as detailed in Lecture 5.

Since both branches ultimately produce gradients with respect to policy parameters, and both are typically optimized using batched stochastic optimizers (e.g. ADAM), their updates possess a remarkable structural symmetry which can be used for simple interactions within the reinforcement twinning architecture, for example exchanging or combining gradient updates, produce weight cloning or policy transfer between branches (see Schena et al. (2024)). To ensure sufficient exploration (which is critical for the value function learning!), noise is injected in the policy output during data collection,

$$\boldsymbol{a}_{\bullet,k} = \pi_{\bullet}(\boldsymbol{o}_{\bullet,k} \mid \boldsymbol{\theta}_{\pi,\bullet}) + \boldsymbol{\varepsilon}_k,$$

with $\boldsymbol{\varepsilon}_k$ typically sampled from an Ornstein–Uhlenbeck or Gaussian process.

Experience Replay and Prioritization. Transitions are stored in a replay buffer and sampled in mini-batches to decorrelate data and improve sample efficiency. In prioritized replay (Schaul et al., 2015), transitions are sampled with probability proportional to their temporal-difference error, thus concentrating learning effort on informative experiences. For implementation details and design choices in the reinforcement twinning context, the reader is referred to Lillicrap et al. (2015); Schena et al. (2024); Poletti et al. (2025).

5.1.2 Reciprocal Reinforcement of Model-Based and Model-Free Policies

The interaction between the two policy branches occurs through the executed trajectories and the replay buffer. At each time step the referee selects one of the candidate actions,

$$\boldsymbol{a}_k \in \{\boldsymbol{a}_{\circ,k}, \boldsymbol{a}_{\bullet,k}\},$$

thereby determining the state–action distribution $d^{\pi}(\boldsymbol{s}, \boldsymbol{a})$ visited in the real system. The resulting transitions $(\boldsymbol{o}_k, \boldsymbol{a}_k, r_k, \boldsymbol{o}_{k+1})$ populate the replay buffer $\mathcal{B}$ and therefore define the sampling measure under which the critic and actor updates are computed.

When the model-based policy $\pi_{\circ}$ dominates execution, $\mathcal{B} \sim d^{\pi_{\circ}}$, and the critic loss

$$\mathcal{L}_Q = \mathbb{E}_{\mathcal{B}}\Big[(Q - y)^2\Big]$$

trains Q primarily on transitions generated by $\pi_{\circ}$. The actor gradient update (22) is therefore shaped by a value landscape induced by the model-based branch. In this sense the model-free policy is driven by demonstration: it initially tracks $\pi_{\circ}$ through value shaping and may subsequently surpass it as the critic generalizes beyond demonstrated actions.

Conversely, when the model-free policy dominates execution, $\mathcal{B} \sim d^{\pi_{\bullet}}$, the visited state distribution allows for more exploration and the model-based policy search is carried out in the state space explored by $\pi_{\bullet}$. This prevents the model driven and adjoint-based search from being confined to narrow regions of the state space.

A further level of exchange is enabled by the symmetry of the policy update, which offers multiple mechanisms to escape local minima and to compensate for biased gradient information. The idea is to exploit the structural equivalence between the two learning loops: the critic approximates the value landscape that drives the model-free actor, just

as the digital twin provides the performance landscape that drives the model-based actor: the critic is to the model-free actor what the digital twin is to the model-based actor.

Therefore, on model-based side, one could introduce critic-induced gradients for the model-based policy, i.e.

$$\nabla_{\boldsymbol{\theta}_{\pi,\circ}} \mathcal{R}^{Q}_{A,\bullet} \approx \mathbb{E}\left[\nabla_{\boldsymbol{\theta}_{\pi,\circ}} \pi_{\circ}(\boldsymbol{o})\, \nabla_{\boldsymbol{a}} Q(\boldsymbol{o}, \boldsymbol{a}; \boldsymbol{\theta}_Q) \Big|_{\boldsymbol{a}=\pi_{\circ}(\boldsymbol{o})} \right]. \tag{25}$$

This has the same deterministic policy-gradient structure as the model-free update but is evaluated on the model-based policy parameters. It injects directional information extracted from real-world transitions into the model-based adjoint-based branch, thereby correcting possible biases in the twin-induced gradient field. By symmetry, the adjoint-based gradient computed within the twin, can be evaluated on the model-free parametrization, i.e.

$$\nabla_{\boldsymbol{\theta}_{\pi,\bullet}} \mathcal{R}_{A,\circ} = \sum_k \nabla_{\boldsymbol{\theta}_{\pi,\bullet}} \boldsymbol{a}_k \, \frac{\partial H_k}{\partial \boldsymbol{a}_k}, \tag{26}$$

thus providing a physics-informed descent direction for the model-free actor.

This gradient is typically smoother and less noisy than the critic-based estimate and may therefore act as a structural regularizer of the model-free update.

A hybrid update may then be defined as

$$\nabla_{\boldsymbol{\theta}_{\pi,\circ}} \mathcal{R}_{\text{hybrid}} = (1-\eta) \nabla_{\boldsymbol{\theta}_{\pi,\circ}} \mathcal{R}_{A,\circ} + \eta \nabla_{\boldsymbol{\theta}_{\pi,\circ}} \mathcal{R}^{Q}_{A,\bullet}, \tag{27}$$

where $\eta \in [0,1]$ controls the influence of real-world value information on the model-based update. For $\eta = 0$, the update reduces to the pure adjoint-based descent driven entirely by the twin. For $\eta = 1$, the model-based policy follows exclusively the critic-induced gradient, effectively aligning its update direction with the real-data value landscape. Intermediate values interpolate smoothly between these two regimes. This bidirectional gradient exchange result in a coupled ascent system in which real-data value information and physics-based structure mutually constrain and enhance each other.

Finally, in the most intrusive scenario, if one branch becomes completely stalled or unstable, a reset mechanism may be activated whereby the parameters themselves are swapped or cloned,

$$\boldsymbol{\theta}_{\pi,\circ} \leftarrow \boldsymbol{\theta}_{\pi,\bullet} \quad \text{or} \quad \boldsymbol{\theta}_{\pi,\bullet} \leftarrow \boldsymbol{\theta}_{\pi,\circ},$$

thus restoring symmetry and allowing learning to resume from a more favorable region of policy space.

Refereeing Mechanism Only one action can be applied to the real system at each time step, $\boldsymbol{a}_k = \Pi_k(\pi_{\circ}(\boldsymbol{o}_k), \pi_{\bullet}(\boldsymbol{o}_k))$, and the referee determines which branch governs exploration and memory filling.

Two fundamentally different arbitration principles may be considered.

(i) Model-based arbitration (twin-driven). Since both candidate policies can be evaluated inside the digital twin, the referee may compare their predicted cumulative rewards, $\mathcal{R}^{(\circ)}_{A,\circ}$ and $\mathcal{R}^{(\bullet)}_{A,\circ}$, obtained by rolling out $\pi_{\circ}$ and $\pi_{\bullet}$ within the twin. The live action is then selected as

$$\boldsymbol{a}_k = \begin{cases} \pi_{\circ}(\boldsymbol{o}_k), & \mathcal{R}^{(\circ)}_{A,\circ} \geq \mathcal{R}^{(\bullet)}_{A,\circ}, \\ \pi_{\bullet}(\boldsymbol{o}_k), & \text{otherwise.} \end{cases}$$

This is the strategy adopted in Schena et al. (2024); Poletti et al. (2025): arbitration is based entirely on the predictive model and is reliable only insofar as the digital twin provides sufficiently accurate performance estimates.

(ii) Value-based arbitration (critic-driven). Alternatively, the referee may rely on the critic and compare predicted state–action values,

$$\Delta Q_k = Q(\boldsymbol{o}_k, \pi_\bullet(\boldsymbol{o}_k)) - Q(\boldsymbol{o}_k, \pi_\circ(\boldsymbol{o}_k)).$$

The policy associated with the larger Q value is selected. This approach bases arbitration on real-data value estimates rather than model predictions and is reliable only if the value-function approximator provides sufficiently accurate evaluations.

(iii) Confidence-weighted blending. Since the critic provides a temporal-difference error signal and the twin provides model-based sensitivity information, the referee may combine both actions using a confidence weight,

$$\boldsymbol{a}_k = (1 - \zeta_k)\, \pi_\circ(\boldsymbol{o}_k) + \zeta_k\, \pi_\bullet(\boldsymbol{o}_k), \tag{28}$$

where $\zeta_k \in [0, 1]$ may depend on relative value predictions, critic uncertainty, or prediction error of the twin. In the limit $\zeta_k = 0$ the system is fully model-based; for $\zeta_k = 1$ it is fully model-free.

(iv) Value-gap driven arbitration. A more structured rule relies on the value difference

$$\Delta Q_k = Q(\boldsymbol{o}_k, \pi_\bullet(\boldsymbol{o}_k)) - Q(\boldsymbol{o}_k, \pi_\circ(\boldsymbol{o}_k)), \tag{29}$$

and activates the model-free policy only when this gap exceeds a prescribed threshold. This prevents premature takeover due to critic noise and stabilizes policy switching.

In all cases, the referee acts not merely as a switch but as a distribution-shaping mechanism: by selecting which policy generates real transitions, it determines how the replay buffer is populated and ultimately which gradients dominate subsequent learning.

Degenerate limit for perfect twin If the twin captures all the relevant dynamics and the closure parameters converge to $\boldsymbol{\theta}_p^\star$ such that

$$F_\circ(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{a}; \boldsymbol{\theta}_p^\star) = F_\bullet(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{a})$$

for all admissible states and actions, then for any policy π,

$$\mathcal{R}_{A,\circ}(\pi) = \mathcal{R}_{A,\bullet}(\pi),$$

and the adjoint gradient satisfies

$$\nabla_{\boldsymbol{\theta}_{\pi,\circ}} \mathcal{R}_{A,\circ} = \nabla_{\boldsymbol{\theta}_{\pi,\circ}} \mathcal{R}_{A,\bullet}.$$

In this regime the model-based gradient is exact, the referee becomes inactive, and the architecture collapses to classical adjoint-based optimal control. The model-free branch adds no new information and eventually aligns with the model-based solution. This proves the consistency property: the modeling to control operation of reinforcement twinning should reduce to classical model-based control when the model is perfect.

A note on the "control to modeling operation" A control-to-modeling regime may be obtained by redefining the stage reward as

$$r_k = -\,\ell_{\text{pred}}\Big(\boldsymbol{o}_{\bullet,k}, \boldsymbol{o}_{\circ,k}\Big), \tag{30}$$

so that the cumulative reward satisfies $\mathcal{R}_A = -\mathcal{J}_T$. In this case the critic approximates

$$Q(\boldsymbol{o}_k, \boldsymbol{a}_k) \approx \mathbb{E}\left[-\sum_{t=k}^{n_t} \ell_{\text{pred}}\right], \tag{31}$$

and therefore encodes the expected future reduction of model mismatch.

The deterministic policy gradient driving the actor becomes

$$\nabla_{\boldsymbol{\theta}_\pi} \mathcal{R}_A \approx \mathbb{E}\left[\nabla_{\boldsymbol{\theta}_\pi} \pi(\boldsymbol{o})\, \nabla_{\boldsymbol{a}} Q(\boldsymbol{o}, \boldsymbol{a})\right] \;\approx\; -\mathbb{E}\left[\nabla_{\boldsymbol{\theta}_\pi} \pi(\boldsymbol{o})\, \frac{\partial \mathcal{J}_T}{\partial \boldsymbol{a}}\right], \tag{32}$$

so that action selection is indirectly driven by the sensitivity of the twin mismatch with respect to control inputs.

At the same time, the closure parameters are updated via the adjoint gradient

$$\nabla_{\boldsymbol{\theta}_p} \mathcal{J}_T = \sum_k \frac{\partial \ell_{\text{pred}}}{\partial \boldsymbol{o}_{\circ,k}} \frac{\partial \boldsymbol{o}_{\circ,k}}{\partial \boldsymbol{\theta}_p}. \tag{33}$$

Hence both the actor and the twin updates act on the same functional $\mathcal{J}_T$, but along complementary directions: the actor modifies the trajectory through $\partial \mathcal{J}_T / \partial \boldsymbol{a}$, while the adjoint modifies the model through $\partial \mathcal{J}_T / \partial \boldsymbol{\theta}_p$.

The reinforcement loop therefore shapes the distribution of visited states in order to influence the modeling gradient itself, providing the bidirectional interaction between control and modeling that characterizes reinforcement twinning.

5.2 Multi-Fidelity Policy Search

We now consider a scheme in which the two policy branches shown in Figure 4 communicate through a multi-fidelity probabilistic model of the operational reward. The first steps of this formulation were tested within the research master project of Yannick Lecomte (Lecomte et al., 2024, 2025). A generalization is currently under development and the general ideas are shared in these notes.

In this formulation we assume that the digital twin provides inexpensive evaluations of the virtual reward $\mathcal{R}_{A,\circ}(\boldsymbol{\theta}_\pi)$ over a wide range of policy parameters, while the real system produces sparse but reliable evaluations of $\mathcal{R}_{A,\bullet}(\boldsymbol{\theta}_\pi)$. These two signals may be interpreted as low- and high-fidelity observations of the same underlying performance landscape. This implies that the same policy parametrization is used for both the model-based and the model-free control policies. We thus use $\boldsymbol{\theta}_\pi \in \mathbb{R}^{n_\pi}$ to denote a generic policy parameter vector.

Within this interpretation, the twin-supported policy $\pi_\circ$ generates a large set of virtual samples which we collect in a matrix

$$\boldsymbol{\Theta}_\circ = \left[\boldsymbol{\theta}^\circ_{\pi,1}, \ldots, \boldsymbol{\theta}^\circ_{\pi,n_\circ}\right] \in \mathbb{R}^{n_\pi \times n_\circ},$$

Similarly, we dispose of a set of candidate policies obtained while interacting with the real system stored in the matrix

$$\boldsymbol{\Theta}_\bullet = \left[\boldsymbol{\theta}^\bullet_{\pi,1}, \dots, \boldsymbol{\theta}^\bullet_{\pi,n_\bullet}\right] \in \mathbb{R}^{n_\pi \times n_\bullet},$$

The rewards observe in the digital twin in the first case and in the real system in the second are collected in the reward vectors

$$\mathbf{r}_\circ = \left[\mathcal{R}_{A,\circ}(\boldsymbol{\theta}^\circ_{\pi,1}), \dots, \mathcal{R}_{A,\circ}(\boldsymbol{\theta}^\circ_{\pi,n_\circ})\right]^\top \in \mathbb{R}^{n_\circ},$$
$$\mathbf{r}_\bullet = \left[\mathcal{R}_{A,\bullet}(\boldsymbol{\theta}^\bullet_{\pi,1}), \dots, \mathcal{R}_{A,\bullet}(\boldsymbol{\theta}^\bullet_{\pi,n_\bullet})\right]^\top \in \mathbb{R}^{n_\bullet}, \qquad n_\bullet \ll n_\circ.$$

5.2.1 Formulation for the model free search

The identification of the next best sample location can be carried out under two scenarios.

(1) Pure model-free search (no trust in the twin). If the digital twin is deemed unreliable for policy improvement, only the high-fidelity dataset $(\boldsymbol{\Theta}_\bullet, \mathbf{r}_\bullet)$ is used to build a single-fidelity surrogate of $\mathcal{R}_{A,\bullet}$. This is exactly the framework introduced in Lecture 4. Denoting by

$$\mathbf{K}_\bullet = k_\bullet(\boldsymbol{\Theta}_\bullet, \boldsymbol{\Theta}_\bullet) + \sigma^2_\bullet \mathbf{I} \in \mathbb{R}^{n_\bullet \times n_\bullet}, \qquad \mathbf{k}_\bullet(\boldsymbol{\theta}_\pi) = k_\bullet(\boldsymbol{\theta}_\pi, \boldsymbol{\Theta}_\bullet) \in \mathbb{R}^{n_\bullet},$$

with $\kappa_\bullet(\cdot,\cdot)$ the kernel function associated with the sampling from the real system, conditioning produces the predictive model

$$\mu_\bullet(\boldsymbol{\theta}_\pi) = \mathbf{k}_\bullet(\boldsymbol{\theta}_\pi)^\top \mathbf{K}_\bullet^{-1} \mathbf{r}_\bullet, \tag{34}$$
$$\sigma^2_\bullet(\boldsymbol{\theta}_\pi) = k_\bullet(\boldsymbol{\theta}_\pi, \boldsymbol{\theta}_\pi) - \mathbf{k}_\bullet(\boldsymbol{\theta}_\pi)^\top \mathbf{K}_\bullet^{-1} \mathbf{k}_\bullet(\boldsymbol{\theta}_\pi). \tag{35}$$

The next $\boldsymbol{\theta}_{\pi,\bullet}$ is then selected by maximizing an acquisition functional (e.g. Expected Improvement) built from $(\mu_\bullet, \sigma_\bullet)$.

(2) Multi-fidelity policy search (partial trust in the twin). If the twin is considered informative but biased, its evaluations are used as low-fidelity data to reduce the number of real experiments. We then use both datasets $(\boldsymbol{\Theta}_\circ, \mathbf{r}_\circ)$ and $(\boldsymbol{\Theta}_\bullet, \mathbf{r}_\bullet)$ and adopt a multifidelity model. For example the autoregressive multi-fidelity model by Kennedy and O'Hagan (1998) (see also Gratiet (2013)) proposes a relation

$$\mathcal{R}_{A,\bullet}(\boldsymbol{\theta}_\pi) = \rho\, \mathcal{R}_{A,\circ}(\boldsymbol{\theta}_\pi) + \delta(\boldsymbol{\theta}_\pi), \tag{36}$$

where ρ is a scalar scaling factor and $\delta(\boldsymbol{\theta}_\pi)$ captures systematic discrepancy. Both the low-fidelity reward and the discrepancy models are modelled as uncorrelated Gaussian process

$$\mathcal{R}_{A,\circ} \sim \mathcal{GP}\Big(0, k_\circ(\cdot,\cdot)\Big), \quad \text{and} \quad \delta \sim \mathcal{GP}\Big(0, k_\delta(\cdot,\cdot)\Big), \qquad \mathcal{R}_{A,\circ} \perp \delta\,,$$

where the kernel functions $k_\circ(\boldsymbol{\theta}, \boldsymbol{\theta}')$ and $k_\delta(\boldsymbol{\theta}, \boldsymbol{\theta}')$ define the covariance structure in policy space. Evaluation of the kernel functions over the sampled inputs $\boldsymbol{\Theta}_\circ$ and $\boldsymbol{\Theta}_\bullet$ induces the finite-dimensional covariance matrices required for conditioning.

Introducing the covariance matrices

$$\mathbf{K}_{\circ\circ} = k_\circ(\boldsymbol{\Theta}_\circ, \boldsymbol{\Theta}_\circ) \in \mathbb{R}^{n_\circ \times n_\circ},$$

$$\mathbf{K}_{\bullet\bullet} = k_\circ(\boldsymbol{\Theta}_\bullet, \boldsymbol{\Theta}_\bullet) \in \mathbb{R}^{n_\bullet \times n_\bullet},$$

$$\mathbf{K}_{\circ\bullet} = k_\circ(\boldsymbol{\Theta}_\circ, \boldsymbol{\Theta}_\bullet) \in \mathbb{R}^{n_\circ \times n_\bullet}, \qquad \mathbf{K}_{\bullet\circ} = \mathbf{K}_{\circ\bullet}^\top,$$

$$\mathbf{K}_{\delta\delta} = k_\delta(\boldsymbol{\Theta}_\bullet, \boldsymbol{\Theta}_\bullet) \in \mathbb{R}^{n_\bullet \times n_\bullet}.$$

The joint covariance matrix of the observed reward vector $\begin{bmatrix}\mathbf{r}_\circ \\ \mathbf{r}_\bullet\end{bmatrix}$ is therefore

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{\circ\circ} + \sigma_\circ^2 \mathbf{I} & \rho\, \mathbf{K}_{\circ\bullet} \\ \rho\, \mathbf{K}_{\bullet\circ} & \rho^2 \mathbf{K}_{\bullet\bullet} + \mathbf{K}_{\delta\delta} + \sigma_\bullet^2 \mathbf{I} \end{bmatrix} \in \mathbb{R}^{(n_\circ + n_\bullet)\times(n_\circ + n_\bullet)}. \tag{37}$$

To query the autoregressive model for a new candidate policy parameter vector $\boldsymbol{\theta}_\pi$ we further define

$$\mathbf{k}_\circ(\boldsymbol{\theta}_\pi) = k_\circ(\boldsymbol{\theta}_\pi, \boldsymbol{\Theta}_\circ) \in \mathbb{R}^{n_\circ}, \qquad \mathbf{k}_\bullet(\boldsymbol{\theta}_\pi) = k_\circ(\boldsymbol{\theta}_\pi, \boldsymbol{\Theta}_\bullet) \in \mathbb{R}^{n_\bullet},$$

$$\mathbf{k}_\delta(\boldsymbol{\theta}_\pi) = k_\delta(\boldsymbol{\theta}_\pi, \boldsymbol{\Theta}_\bullet) \in \mathbb{R}^{n_\bullet}.$$

The cross-covariance vector becomes

$$\mathbf{k}_*(\boldsymbol{\theta}_\pi) = \begin{bmatrix} \rho\, \mathbf{k}_\circ(\boldsymbol{\theta}_\pi) \\ \rho^2\, \mathbf{k}_\bullet(\boldsymbol{\theta}_\pi) + \mathbf{k}_\delta(\boldsymbol{\theta}_\pi) \end{bmatrix} \in \mathbb{R}^{n_\circ + n_\bullet},$$

and the prior variance at the query point is

$$k_{**}(\boldsymbol{\theta}_\pi) = \rho^2 k_\circ(\boldsymbol{\theta}_\pi, \boldsymbol{\theta}_\pi) + k_\delta(\boldsymbol{\theta}_\pi, \boldsymbol{\theta}_\pi).$$

Therefore, conditioning the auto-regressive model on the available data gives the multi-fidelity posterior model

$$\mu_\bullet^{\mathrm{MF}}(\boldsymbol{\theta}_\pi) = \mathbf{k}_*(\boldsymbol{\theta}_\pi)^\top \mathbf{K}^{-1} \begin{bmatrix}\mathbf{r}_\circ \\ \mathbf{r}_\bullet\end{bmatrix}, \tag{38}$$

$$\sigma_\bullet^{2,\mathrm{MF}}(\boldsymbol{\theta}_\pi) = k_{**}(\boldsymbol{\theta}_\pi) - \mathbf{k}_*(\boldsymbol{\theta}_\pi)^\top \mathbf{K}^{-1} \mathbf{k}_*(\boldsymbol{\theta}_\pi). \tag{39}$$

The next parameter $\boldsymbol{\theta}_{\pi,\bullet}$ is then selected by maximizing an acquisition functional constructed from $(\mu_\bullet^{\mathrm{MF}}, \sigma_\bullet^{\mathrm{MF}})$, thereby exploiting the twin to guide exploration while allowing high-fidelity data to correct systematic bias.

As in the single-fidelity case, the dominant computational cost arises from inversion (or factorization) of the covariance matrix; numerical strategies for efficient sequential updates, including rank-one Cholesky updates, were discussed in Lecture 4. Similarly, the optimization of the kernel hyperparameters (e.g. length scales, variances, and the scaling factor ρ in the multi-fidelity case) is carried out using standard maximum marginal likelihood estimation, as discussed in Lecture 4. In the multi-fidelity setting this corresponds to maximizing the joint log-marginal likelihood associated with the covariance matrix $\mathbf{K}$.

5.2.2 Reciprocal Reinforcement of Model-Based and Model-Free Policies

While the digital twin clearly accelerates the model-free exploration by providing inexpensive low-fidelity samples, the reverse interaction is equally important. The multi-fidelity posterior $\mu_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_\pi)$ may be interpreted as an improved estimate of the real operational response, corrected by experimental evidence. Instead of optimizing the model-based policy exclusively with respect to the virtual reward $\mathcal{R}_{A,\circ}$, one may update $\boldsymbol{\theta}_{\pi,\circ}$ by maximizing the corrected surrogate $\mu_{\bullet}^{\mathrm{MF}}$, possibly penalized by its associated uncertainty:

$$\boldsymbol{\theta}_{\pi,\circ} \leftarrow \arg\max_{\boldsymbol{\theta}_\pi} \Big(\mu_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_\pi) - \xi\, \sigma_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_\pi) \Big), \tag{40}$$

where $\xi \geq 0$ controls risk aversion.

Such a strategy effectively injects high-fidelity information into the model-based loop. To avoid unstable extrapolation in poorly sampled regions of parameter space, this update may be restricted to trust regions where $\sigma_{\bullet}^{\mathrm{MF}}$ remains below a prescribed threshold

$$\sigma_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_\pi) \leq \bar{\sigma}, \tag{41}$$

or regularized through a quadratic penalty $\|\boldsymbol{\theta}_\pi - \boldsymbol{\theta}_\pi^{\mathrm{ref}}\|^2$.

In this manner, experimental data does not merely validate the twin but actively reshapes the objective landscape perceived by the model-based optimizer.

Refereeing Mechanisms In the multi-fidelity formulation the referee acquires a richer role than in the formulation in Section 5.1. Besides selecting which policy branch provides the live action, it now has access to a probabilistic description of the expected operational performance through the posterior quantities $\mu_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_\pi)$ and $\sigma_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_\pi)$. These quantities encode also a measure of epistemic uncertainty inherited from the joint low- and high-fidelity datasets.

This additional information enables several forms of arbitration. First, the referee may select between $\pi_\circ$ and $\pi_\bullet$ based on risk-adjusted criteria, privileging the branch whose predicted reward remains favorable under uncertainty.

A simple risk-adjusted decision rule consists in comparing

$$S_\circ = \mathcal{R}_{A,\circ}(\boldsymbol{\theta}_{\pi,\circ}) - \beta_\circ\, \sigma_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_{\pi,\circ}), \qquad S_\bullet = \mu_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_{\pi,\bullet}) - \beta_\bullet\, \sigma_{\bullet}^{\mathrm{MF}}(\boldsymbol{\theta}_{\pi,\bullet}), \tag{42}$$

and selecting the branch with the larger score. More generally, the applied action may be blended,

$$\boldsymbol{a}_k = (1-\alpha_k)\, \boldsymbol{a}_{\circ,k} + \alpha_k\, \boldsymbol{a}_{\bullet,k}, \tag{43}$$

where the mixing coefficient may depend on relative confidence, e.g.

$$\alpha_k = \frac{\sigma_{\circ,k}^2}{\sigma_{\circ,k}^2 + \sigma_{\bullet,k}^2}, \tag{44}$$

in analogy with Kalman-type gain formulas. In this interpretation the referee becomes a probabilistic fusion layer rather than a purely logical switch.

Degenerate limit for perfect digital twin If the digital twin becomes highly accurate, the discrepancy process $\delta(\boldsymbol{\theta}_\pi)$ approaches zero and the scaling factor ρ approaches unity. In that regime, the high-fidelity observations $(\boldsymbol{\Theta}_\bullet, \mathbf{r}_\bullet)$ and the low-fidelity observations $(\boldsymbol{\Theta}_\circ, \mathbf{r}_\circ)$ effectively describe the same performance landscape. The multi-fidelity posterior then collapses to a single-fidelity surrogate driven predominantly by the twin and the entire reinforcement twinning structure effectively reduces to optimizing a prescribed parametric controller against a reliable model. In that regime, maintaining a separate model-free search becomes unnecessary, and the framework consistently degenerates into classical model-based control. This proves the consistency property.

A note on the "control to modeling operation" The multi-fidelity surrogate also provides a mechanism to couple policy exploration with refinement of the twin parameters $\boldsymbol{\theta}_p$ appearing in the prediction cost $\mathcal{J}_T$. Rather than selecting policies exclusively to maximize operational performance, one may deliberately choose parameter configurations that are informative for the identification of the digital twin.

A first mechanism relies on the discrepancy signal

$$\Delta(\boldsymbol{\theta}_\pi) = \mathcal{R}_{A,\bullet}(\boldsymbol{\theta}_\pi) - \rho\, \mathcal{R}_{A,\circ}(\boldsymbol{\theta}_\pi), \tag{45}$$

which identifies regions of policy space where the twin fails in performance-relevant ways. Policies that induce large $|\Delta|$ correspond to trajectories along which the model error is structurally significant. Such policies may be prioritized to generate data that accelerate the minimization of $\mathcal{J}_T(\boldsymbol{\theta}_p)$ through the adjoint-based update of $\boldsymbol{\theta}_p$.

A second mechanism exploits the uncertainty encoded in the surrogate. Instead of defining $\mathcal{J}_T$ directly as a function of the reward, one may use the surrogate variance as a proxy for modeling uncertainty and select

$$\boldsymbol{\theta}_\pi^{\text{info}} = \arg\max_{\boldsymbol{\theta}_\pi} \sigma_\bullet^{\text{MF}}(\boldsymbol{\theta}_\pi), \tag{46}$$

subject to safety constraints. The data collected under such policies enrich the dataset $(\boldsymbol{\Theta}_\bullet, \mathbf{r}_\bullet)$ in regions where the twin is uncertain, thereby reshaping the landscape of $\mathcal{J}_T$ and improving the subsequent estimation of $\boldsymbol{\theta}_p$.

More generally, a composite objective may be considered,

$$\max_{\boldsymbol{\theta}_\pi}\ \mu_\bullet^{\text{MF}}(\boldsymbol{\theta}_\pi) - \xi\, \mathcal{J}_T(\boldsymbol{\theta}_p) + \gamma\, \sigma_\bullet^{\text{MF}}(\boldsymbol{\theta}_\pi), \tag{47}$$

where ξ and γ balance operational performance, model consistency, and information gain.

6 A tutorial test case: stabilization of a Floating Wind Turbine (FWT)

General Context . Wind energy is increasingly shifting offshore, where water depths exceed the limits of fixed-bottom structures and floating wind turbines (FWT) become necessary. The dynamics of a FWT are dominated by strong coupling between wave-induced platform motions and rotor aerodynamics (Jonkman, 2007; Stockhouse et al., 2022), leading to well-known instabilities such as negative damping (Larsen and Hanson,

2007; Lackner, 2009; Karikomi et al., 2015). While many advanced control strategies exist—model predictive control (MPC) (Chaaban and C-P., 2014), platform actuation (Stockhouse et al., 2022), structural damping (Namik et al., 2013), or H_∞ control (Li and Gao, 2015)—they typically rely on simplified models whose validity degrades in strongly off-design conditions.

The challenge is intrinsically difficult because the turbine is subject to stochastic disturbances arising from both wind and waves, while the available actuation authority in conventional designs remains limited to blade pitch and generator torque, unless more sophisticated concepts such as active ballast systems or controlled mooring-line tension are introduced. We assume in this tutorial that the floating wind turbine does not dispose of such additional devices and that stabilization must rely solely on blade pitch and generator torque. As we will see, a classical land-based control strategy—designed primarily for power maximization—can induce large oscillations of the floating platform and significant load fluctuations.

The problem studied here is a drastic simplification of real floating wind turbine physics. Actual systems feature coupled aero-hydro-servo-elastic dynamics with many structural modes, nonlinear hydrodynamics, wake interactions, and complex sensing and actuation. The goal of this tutorial is not to reproduce this complexity, but to construct a minimal dynamical model that still captures the key aspects of the stabilization problem. In this controlled setting, we fairly compare three approaches: a fully model-based strategy, a fully model-free reinforcement learning strategy, and the reinforcement twinning framework introduced in this chapter.

The aim is pedagogical: to build intuition about how hybrid model-based/model-free methods behave in a coupled, disturbed dynamical system and to evaluate their strengths and weaknesses in a controlled yet representative scenario.

Problem set and simulated "real" environment. The configuration of interest is illustrated in Figure 5. It consists of a floating wind turbine mounted on a semi-submersible platform. The turbine has a rotor of radius R rotating at angular velocity Ω and a tower of height H_t. This is subjected to an incoming wind flow with hub-height velocity v_∞, as well as to incoming waves (assumed to arrive with zero incident angle), denoted by η_{waves}.

In general, a floating platform possesses six rigid-body degrees of freedom: surge, sway, and heave (translations), and roll, pitch, and yaw (rotations). In this tutorial, we restrict attention to a single dominant degree of freedom, namely the pitch motion of the platform about its center of gravity (C.G.). This simplification captures the primary aero–hydro coupling mechanism responsible for negative damping effects, while keeping the dynamical model minimal.

The aerodynamic forces generated by the rotor produce an overall thrust force T_{aero}, acting approximately along the rotor axis, and an aerodynamic torque τ_{aero} acting on the low-speed shaft. The generator applies an opposing torque τ_g, while blade pitch control modifies the aerodynamic loading through the pitch angle β.

For modelling purposes, the distributed aerodynamic loading along the blades is represented by an equivalent resultant thrust force T_{aero} acting at hub height. This resultant force generates a pitching moment about the platform C.G., thereby exciting the pitch motion. The corresponding pitch angle, angular velocity, and angular acceleration are denoted by θ, $\dot{\theta}$, and $\ddot{\theta}$, respectively.

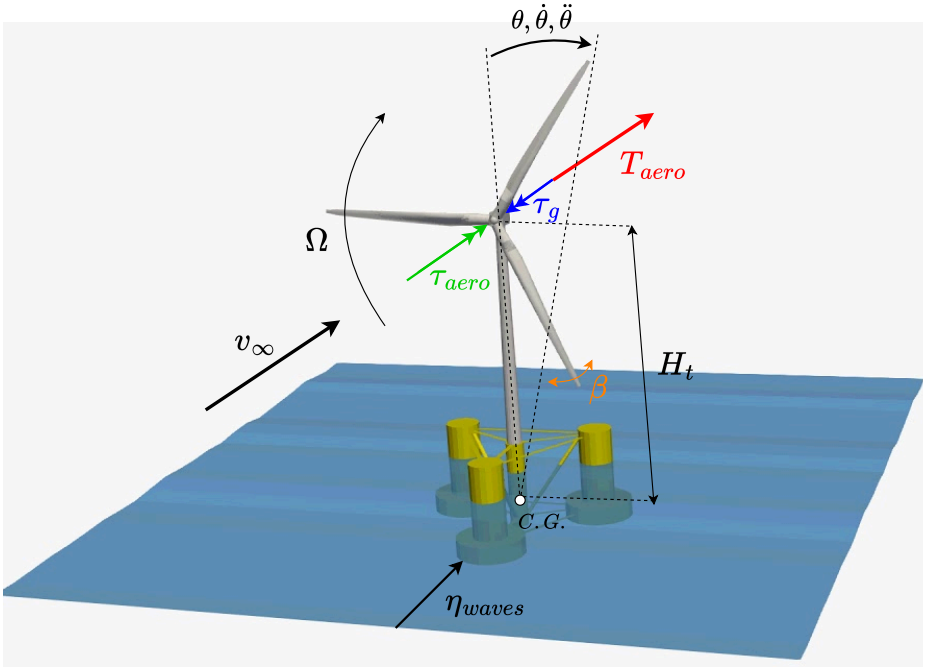

Figure 5: Configuration of interest and relevant variables.

The pitch dynamics of the floating platform are modeled as a second–order nonlinear equation:

$$J_{\mathrm{p}}\ddot{\theta} + D\dot{\theta} + D_v|\dot{\theta}|\dot{\theta} + K\theta = H_t T_{\mathrm{aero}} + \tau_{\mathrm{waves}}, \tag{48}$$

where J_{p} denotes the total rotational inertia about the pitch axis (including platform, turbine and added hydrodynamic inertia), D and D_v represent linear and quadratic hydrodynamic damping, and K is the hydrostatic restoring stiffness accounting for buoyancy and mooring effects. The aerodynamic thrust generates a pitching moment through the lever arm H_t, while τ_{waves} represents the external excitation due to waves.

Neglecting drivetrain dynamics and blade aeroelasticity, the rotor speed evolves according to

$$\dot{\Omega} = \frac{1}{J_{\mathrm{rot}}}\left(\tau_{\mathrm{aero}} - \tau_g\right), \tag{49}$$

with J_{rot} the rotor inertia.

The aerodynamic torque is modeled using the power coefficient C_p as

$$\tau_{\mathrm{aero}} = \frac{1}{2}\rho A \frac{C_p(\xi,\beta)}{\Omega} v^3, \tag{50}$$

where ρ is the air density, $A = \pi R^2$ is the rotor swept are, $\xi = \Omega R/v$ is the tip-speed ratio, and β is the blade pitch angle. Similarly, the aerodynamic thrust is obtained from the thrust coefficient C_T as:

$$T_{\mathrm{aero}} = \frac{1}{2}\rho A C_T(\xi,\beta) v^2. \tag{51}$$

For a floating turbine, the effective wind speed seen by the rotor is modified by the platform motion:

$$v = v_\infty - H_t\dot{\theta}. \tag{52}$$

This kinematic coupling introduces an additional feedback between the aerodynamic and hydrodynamic subsystems and is the key mechanism responsible for potential negative damping. Both C_p and C_T are typically obtained using Blade Element Momentum Theory (BEMT). The ones used in this tutorial are shown in Figure 6. These were obtained with the `CCBlade` solver (Abbas et al., 2022) on a grid of values of β and ξ. The data on the grid were used to fit a surrogate model using ridge regression combining Gaussian radial basis functions (RBFs) in ξ and a polynomial basis in β as described in (Randino et al., 2026). This surrogate model is faster to evaluate than a local interpolator.

For the purposes of this exercise, we assume that the surrogate model for the C_p and C_T are accurate, that is the turbine dynamics is perfectly known.

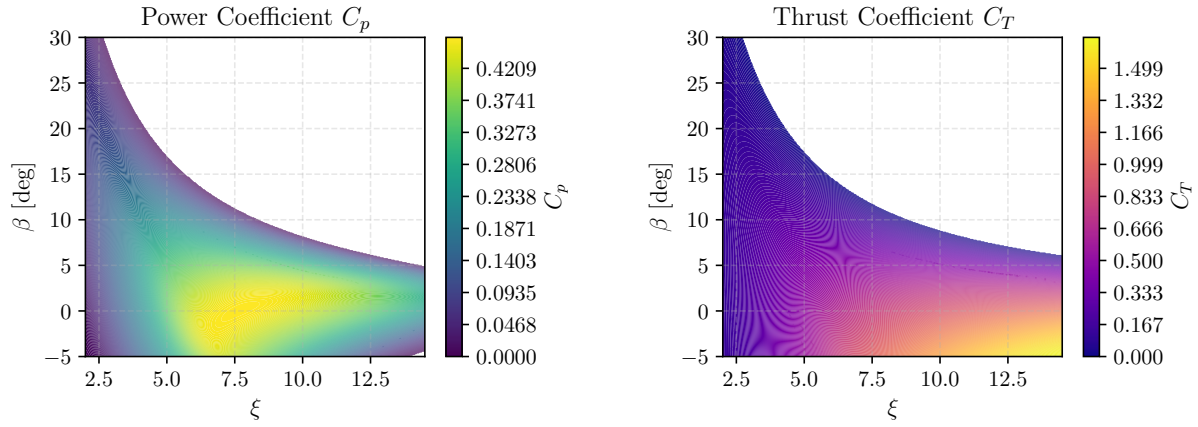

Figure 6: Contour plots of the power coefficient $C_p(\lambda, \beta)$ and the thrust coefficient $C_T(\lambda, \beta)$ for the turbine considered.

The actuator dynamics for blade pitch and generator torque are modelled as first-order systems, such that the commanded inputs $(\beta_c, \tau_{g,c})$ are not applied instantaneously. Specifically,

$$\dot{\beta} = \frac{1}{\tau_\beta}\big(\beta_c - \beta\big), \qquad \dot{\tau}_g = \frac{1}{\tau_{\tau_g}}\big(\tau_{g,c} - \tau_g\big), \tag{53}$$

where τ_β and τ_{τ_g} are time constants representing the bandwidth of the pitch and torque actuators.

Both the inflow velocity and the wave excitation signals are modeled as Gaussian processes (GPs), such that

$$x_i(t) \sim \mathcal{GP}\big(\mu_i(t), \kappa_i(t, t')\big), \tag{54}$$

where $\mu_i(t)$ denotes the mean function and $\kappa_i(t, t')$ is the covariance kernel defined above, whose white-noise contribution $\sigma_{n,i}^2 \delta(t - t')$ ensures that sampled realizations are noisy. This follows the framework introduced in Tutorials 2 and 3 of Lecture 5. The stochastic structure of both processes relies on a common two-scale covariance kernel, which captures variability occurring at two distinct temporal scales.

The two-scale kernel is defined as

$$\kappa_i^{\text{2-scales}}(t, t') = \sigma_{i,1}^2 \exp\left(-\frac{(t - t')^2}{2\,\ell_{i,1}^2}\right) + \sigma_{i,2}^2 \exp\left(-\frac{(t - t')^2}{2\,\ell_{i,2}^2}\right) + \sigma_{n,i}^2\,\delta(t - t'), \qquad i \in \{w, \eta\}, \tag{55}$$

where $\sigma_{i,1}$ and $\sigma_{i,2}$ denote the amplitude parameters associated with the two characteristic time scales $\ell_{i,1}$ and $\ell_{i,2}$, respectively, and $\sigma_{n,i}^2$ represents the variance of the white-noise component. The term $\delta(\cdot)$ denotes the Dirac delta function. The index $i = w$ refers to the wind generation case, whereas $i = \eta$ corresponds to the wave excitation case.

While the wind generator only rely on a two-scale kernel, the wave case adds the contribution of an harmonic kernel, defined as

$$\kappa_\eta(t,t') = \kappa_\eta^{\text{2-scales}}(t,t') + \sigma_h^2 \exp\left(-\frac{2}{\ell_H^2}\sin^2\left(\frac{\pi(t-t')}{T_\eta}\right)\right), \tag{56}$$

where σ_h controls the energy of the oscillatory component, T_w denotes the dominant wave period, and ℓ_H characterizes the temporal coherence of the wave train.

This formulation ensures that both processes share a common multi-scale stochastic backbone, while the wave model additionally captures the narrow-band periodic structure inherent to ocean wave dynamics.

The selected Gaussian-process model enables faster generation of stochastic incoming disturbances than more realistic sea-state descriptions based on wave spectra (e.g. the JONSWAP spectrum in Mazzaretto et al. (2022)).

The kernel (56) is used to produce signals of wave elevation $\eta_w(t)$. From these, the wave torque applied to the platform can be estimated by convolving $\eta_w(t)$ with the hydrodynamic wave-excitation impulse response, typically computed for the floater using potential-flow BEM solvers (Robertson et al., 2014). For the purposes of this exercise, we omit these details and assume that the controller can be informed by the incoming waves and compute from these the associated moments τ_{waves} forcing the platform in (48).

Selected Conditions The selected wind turbine for this exercise is the NREL 5 MW reference model (Jonkman et al., 2009), characterized by a rotor radius of $R = 63$ m and a rotor inertia of $J_{\text{rot}} = 1.16 \times 10^5$ kg m^2. The system reaches its rated operating point at a wind speed of $v_\infty^{(r)} = 11.4$ m/s, which corresponds to a rated low-speed-shaft (LSS) rotor speed of $\Omega^{(r)} = 1.27$ rad/s. The generator torque τ_g is also expressed on the same shaft, allowing the elimination of the gearbox ratio from the system of equations.

The reduced one-degree-of-freedom pitch model of the platform, as described by Eq. (48), is parameterized with values representative of an OC4-type semi-submersible (Robertson et al., 2014). These coefficients capture the total inertia, combined hydrodynamic damping, and hydrostatic restoring properties of the system:

$$\begin{aligned} J &= 1.5 \times 10^7 \text{ kg m}^2, \\ D &= 2.0 \times 10^6 \text{ N m s/rad}, \\ D_v &= 5.0 \times 10^5 \text{ N m s}^2/\text{rad}^2, \\ K &= 1.2 \times 10^7 \text{ N m/rad}. \end{aligned}$$

The actuator dynamics for blade pitch and generator torque in (53) are bounded by physical rate limits and magnitude saturation. For the system considerd in this exercise, these limits are

$$\begin{aligned} \beta &\in [0°, 90°], & |\dot{\beta}| &\leq 8°/\text{s}, \\ \tau_g &\in [0, 43.1 \text{ kNm}], & |\dot{\tau}_g| &\leq 15 \text{ kNm/s}, \end{aligned}$$

The turbulent wind field excitation signal ($i = w$) is generated using the kernel parameters $[\sigma_{w,1}, \sigma_{w,2}, \sigma_{n,w}] = [1, 0.5, 0.1]$ m^2/s^2, $[\ell_{w,1}, \ell_{w,2}] = [20, 0.5]$ s. The signal has a mean of $\mu_w = 12$ m/s, ensuring to be above the rated wind speed $v_\infty^{(r)}$.

For the wave case ($i = \eta$), the kernel is defined by $[\sigma_{\eta,1}, \sigma_{\eta,2}, \sigma_{n,\eta}, \sigma_h] = [0.5, 0.45, 0.03, 2]$ m^2/s^2, $[\ell_{\eta,1}, \ell_{\eta,2}, \ell_H] = [2, 1.5, 2]$ s and $T_w = 10$ s. The mean signal value for the wave is set to $\mu_\eta = 0$.

An example signal for the inflow velocity, wave elevation and associated induced moment are shown in Figure 7.

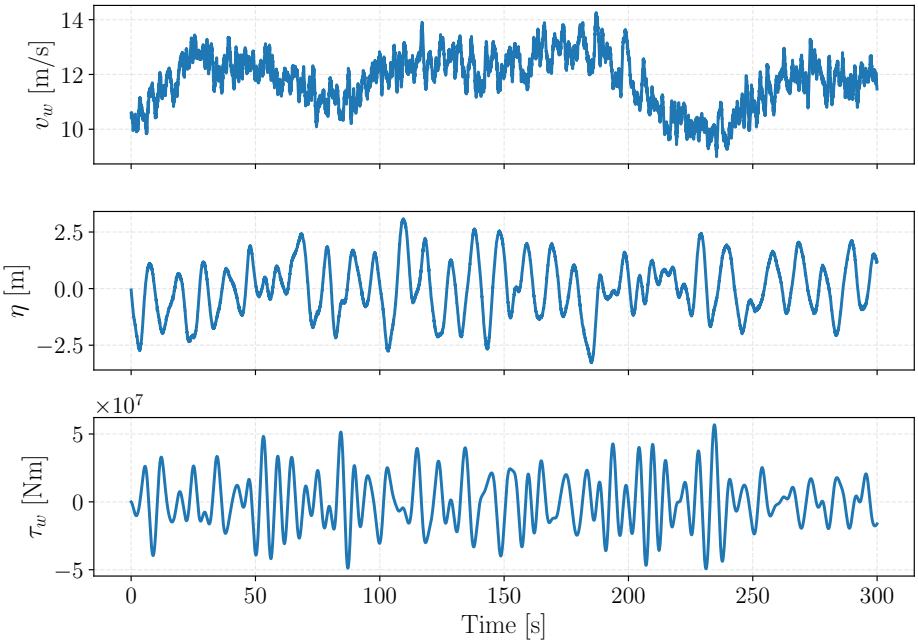

Figure 7: Example realization of the disturbance signals z obtained from the corresponding Gaussian processes. From top to bottom: the wind inflow velocity v_w [m/s], the wave elevation η [m], and the resulting wave-induced torque τ_w [Nm].

Episode Initialization

At the beginning of each episode, the initial state $\boldsymbol{s}_0$ is sampled from a multivariate Gaussian distribution centered at the equilibrium values:

$$\boldsymbol{s}_0 \sim \mathcal{N}\Big(\bar{\boldsymbol{s}}(v_0, \tau_{w,0}), \boldsymbol{\Sigma}\Big) \,, \tag{57}$$

where the reference state $\bar{\boldsymbol{s}} = [\theta^\star, \dot{\theta}^\star, \Omega^\star]^\top$ is determined by finding the steady-state fixed point of the system under the sampled environmental conditions v_0 and $\tau_{w,0}$, and the covariance matrix $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_\theta^2, \sigma_{\dot{\theta}}^2, \sigma_\Omega^2)$ define the allowed variability with respect to the mean values. We here consider $\sigma_\theta = 2.5°$, $\sigma_{\dot{\theta}} = 0.1$ rad/s, drawn from the expected operational dynamics of the OC4 (Robertson et al., 2014), and $\sigma_\Omega = 0.2 \cdot \Omega^{(r)}$. For the above-rated conditions concerning this exercise, we enforce $\Omega^\star = \Omega^{(r)}$ and $\dot{\theta}^\star = 0$. The remaining turbine pitch degree of freedom is found as follows. First, the steady state pitch angle $\beta^\star$ is obtained by solving the torque balance in Eq. (49) enforcing $\dot{\Omega} = 0$, such that $\tau_{\mathrm{aero}}(v_0, \beta^\star, \Omega^{(r)}) = N_g \tau_g$. Given the equilibrium pitch $\beta^\star$, the steady state platform

tilt $\theta^\star$ is finally computed balancing the aerodynamic thrust moment and external wave disturbances against the hydrostatic restoring stiffness K. That is,

$$\theta^* = \frac{T(v_0, \beta^*, \Omega^{(r)})H_t + \tau_{w,0}}{K} \,, \tag{58}$$

where T is the rotor thrust and H_t is the hub height.

Baseline Controller: the $K\Omega^2$ Strategy As a reference strategy, we adopt the classical $K\Omega^2$ torque controller introduced in Lecture 5 and widely used in variable-speed wind turbines (Pao and Johnson, 2009; Bianchi et al., 2007; Pao et al., 2024). The generator torque is prescribed as $tau_g = K\,\Omega^2$, where the gain K is selected such that, under steady inflow conditions, aerodynamic and generator torques balance at a prescribed optimal tip–speed ratio $\xi_\star$. This ensures operation near maximum power capture for constant wind velocity and absence of waves. While effective for land-based turbines, this strategy does not explicitly account for platform dynamics. In floating configurations, the coupling $v = v_\infty - H_t\omega$ introduces additional feedback, and the classical $K\Omega^2$ law may contribute to negative damping and amplified platform oscillations under wave excitation. Figure 8 illustrate the response of the floating wind turbine to an example of inputs for wind and wave disturbances.

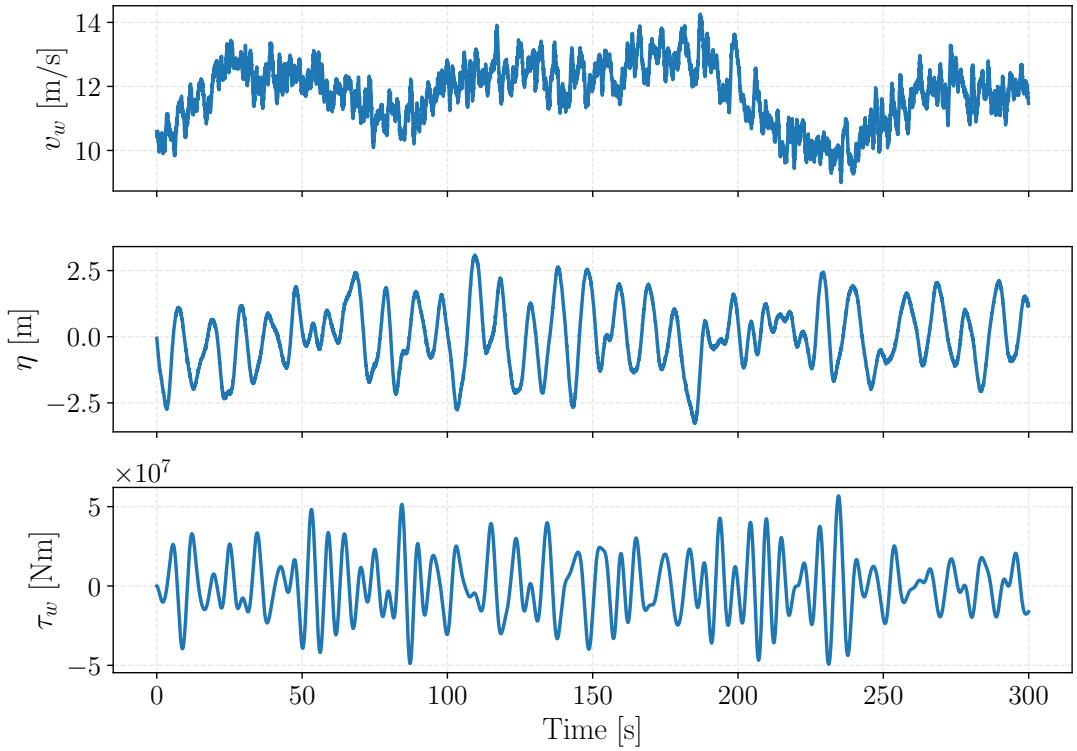

Figure 8: Example realization of the disturbance signals z obtained from the corresponding Gaussian processes. From top to bottom: the wind inflow velocity v_w [m/s], the wave elevation η [m], and the resulting wave-induced torque τ_w [Nm].

Control Policies and Rewards The controller has two goals: keep electrical power near its rated value and reduce oscillations from wind turbulence and wave–induced platform motion. Because the tutorial treats above–rated operation, the reference power is fixed at $P_{\text{ref}} = P^{(r)}$. The instantaneous electrical power is $P = \tau_g \Omega$. To ensure steady production and dynamic stability, the stage reward is the negative of a quadratic penalty on power deviation, short–term power fluctuations, and a term related to platform motion:

$$r_k = -\left[w_P \left(\frac{P_k - P_{\text{ref}}}{P_{\text{ref}}} \right)^2 + w_{\Delta P} \left(\frac{P_k - P_{k-1}}{P_{\text{ref}}} \right)^2 + w_\omega \omega_k^2 \right], \tag{59}$$

where w_P, $w_{\Delta P}$, and w_ω are positive weights. The first term enforces regulation around rated power. The second term penalizes rapid variations of power, discouraging high-frequency oscillations. The third term directly penalizes platform angular velocity. This contribution acts on the physical mechanism responsible for negative aerodynamic damping: wave-induced torque excites pitch motion, which modifies the effective inflow velocity $v = v_\infty - H_t \omega$ and propagates into aerodynamic torque and power fluctuations. By discouraging large ω, the controller reduces the transmission of wave energy into the aerodynamic subsystem and enhances overall stability. Since actuator dynamics already incorporate bandwidth limitations and rate constraints, no additional control-effort penalty is introduced; physical saturation bounds implicitly regularize excessive actuation.

The control inputs are the commanded blade pitch angle β_c and generator torque $\tau_{g,c}$. We adopt a static state–feedback policy of the form

$$\begin{bmatrix} \beta_c \\ \tau_{g,c} \end{bmatrix} = \boldsymbol{a}_{\text{mid}} + \boldsymbol{a}_{\text{amp}} \ \tanh\Big(\mathbf{K}\, \boldsymbol{\phi}_k \Big), \tag{60}$$

where $\tanh(\cdot)$ acts componentwise, $\mathbf{K} \in \mathbb{R}^{2 \times n_\phi}$ is the matrix of policy parameters, and the feature vector is defined as

$$\boldsymbol{\phi}_k = \begin{bmatrix} 1 & \dfrac{P_k - P_{\text{ref}}}{P_{\text{ref}}} & \dfrac{\Omega_k}{\Omega^{(r)}} & \omega_k \end{bmatrix}^\top \tag{61}$$

The constant term lets the policy shift the equilibrium operating point without knowing the exact steady state. The normalized power error enforces rated-power tracking. The rotor speed Ω represents drivetrain kinetic energy and affects aerodynamic torque via the tip–speed ratio. The platform pitch rate ω_k is included as the dynamical pathway through which wave excitation alters inflow velocity and can cause negative damping. This term lets the controller counteract that effect directly, instead of responding only to its impact on the power signal.

The vectors $\boldsymbol{a}_{\text{mid}}$ and $\boldsymbol{a}_{\text{amp}}$ enforce physical actuation bounds,

$$\boldsymbol{a}_{\text{mid}} = \begin{bmatrix} 45^\circ \\ 21.55 \text{ kNm} \end{bmatrix}, \qquad \boldsymbol{a}_{\text{amp}} = \begin{bmatrix} 45^\circ \\ 21.55 \text{ kNm} \end{bmatrix}, \tag{62}$$

to preserve the actuation bounds defined previously. The hyperbolic tangent ensures smooth saturation while preserving differentiability, guaranteeing feasibility of the commanded inputs before they are filtered by the actuator dynamics in Eq. (53).

The turbulent wind field is generated using TurbSim with mean wind speed $v_\infty = 11$ m/s and turbulence intensity $TI = 16\%$, oscillating around rated conditions. Wave excitation can be prescribed either as a sinusoidal input or through a JONSWAP spectrum. We consider significant wave heights ranging from moderate to severe conditions ($H_s = 1.18$–3.44 m) with associated peak periods between $T_s = 7.27$ and 9.32 s.

Each simulation runs over an observation window of $T_o = 600$ s, corresponding to approximately four times the natural pitch period of the platform. The system is assumed to be instrumented at a sampling frequency of $f_s = 4$ Hz.

References

Abarbanel, H. D. I., Rozdeba, P. J., and Shirman, S. (2018). Machine learning in data assimilation for chaotic dynamical systems. *Neural Computation*, 30(7):1811–1862.

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.

Adler, E. J. and Martins, J. R. (2023). Hydrogen-powered aircraft: Fundamental concepts, key technologies, and environmental impacts. *Progress in Aerospace Sciences*, 141:100922.

Anderson, B. D. and Johnstone, R. M. (1985). Global adaptive pole positioning. *IEEE Transactions on Automatic Control*, 30(1):11–22.

Ariyur, K. B. and Krstic, M. (2003). *Real-Time Optimization by Extremum-Seeking Control*. John Wiley & Sons.

Asch, M., Bocquet, M., and Nodet, M. (2016). *Data Assimilation: Methods, Algorithms, and Applications*. SIAM.

Åström, K. J. and Wittenmark, B. (1995). *Adaptive Control*. Addison-Wesley, 2nd edition.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Badwe, A. S., Gudi, R. D., Patwardhan, S. C., and Shah, S. L. (2010). Quantifying the impact of model-plant mismatch on controller performance. *Journal of Process Control*, 20(4):408–425.

Bansal, S., Calandra, R., Xiao, T., and Levine, S. (2017). Mbmf: Model-based priors for model-free reinforcement learning. In *CoRL*.

Bardi, M. and Capuzzo-Dolcetta, I. (1997). *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhäuser.

Bernhardsson, B. (1989). Dual control of a first-order system with two possible gains. *International Journal of Adaptive Control and Signal Processing*, 3(1):15–22.

Bertsekas, D. P. (2005). Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC*. *European Journal of Control*, 11(4-5):310–334.

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

Blanke, M., Kinnaert, M., Lunze, J., and Staroswiecki, M. (2016). *Diagnosis and Fault-Tolerant Control*. Springer, 3rd edition.

Buizza, C., Quilodrán Casas, C., Nadler, P., Mack, J., Marrone, S., Titus, Z., Le Cornec, C., Heylen, E., Dur, T., Baca Ruiz, L., Heaney, C., Díaz Lopez, J. A., Kumar, K. S., and Arcucci, R. (2022). Data learning: Integrating data assimilation and machine learning. *Journal of Computational Science*, 58:101525.

Carnarius, A., Thiele, F., Özkaya, E., and Gauger, N. R. (2011). Optimal control of the cylinder wake in the laminar regime by trust-region evolution strategies. *Journal of Computational Physics*, 230(9):3348–3364.

Chaaban, R. and C-P., F. (2014). Reducing blade fatigue and damping platform motions of floating wind turbines using model predictive control. *International Conference on Structural Dynamics*.

Chen, W.-H., Rhodes, C., and Liu, C. (2021). Dual Control for Exploitation and Exploration (DCEE) in Autonomous Search.

Chiuso, A. and Pillonetto, G. (2019). System identification: A machine learning perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:281–304.

Dehkordi, M. B., Forgione, M., and Piga, D. (2025). Uncertainty quantification in neural state-space models: Applications for experiment design and uncertainty-aware mpc. *European Journal of Control*, 85:101359.

Dixon, L. C. W. (1981). The adjoint method for sensitivity analysis of dynamic systems. *Numerical Optimization of Dynamic Systems*.

Feldbaum, A. A. (1960). Dual control theory. *Automation and Remote Control*, 21:874–880.

Feldbâum, A. A. (1963). Dual control theory problems. *IFAC Proceedings Volumes*, 1(2):541–550.

Ficili, I., Giacobbe, M., Tricomi, G., and Puliafito, A. (2025). From sensors to data intelligence: Leveraging iot, cloud, and edge computing with ai. *Sensors*, 25(6):1763.

Goodwin, G. C. and Payne, R. L. (1977). *Dynamic System Identification: Experiment Design and Data Analysis*. Academic Press.

Gratiet, L. L. (2013). Recursive co-kriging model for Design of Computer experiments with multiple levels of fidelity with an application to hydrodynamic. arXiv: 1210.0686 [math.ST].

Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. (2012). A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Gunzburger, M. D. (2002). *Perspectives in Flow Control and Optimization*. SIAM.

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

Isermann, R. (2006). *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Springer.

Johannink, T. et al. (2019). Residual reinforcement learning for robot control. In *ICRA*.

Jones, D., Snider, C., Nassehi, A., Yon, J., and Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52.

Jonkman, J., Butterfield, S., Musial, W., and Scott, G. (2009). Definition of a 5-mw reference wind turbine for offshore system development. Technical report, National Renewable Energy Laboratory (NREL), Golden, CO.

Jonkman, J. M. (2007). Dynamics modeling and loads analysis of an offshore floating wind turbine. In *Proceedings of the OMAE 2007 Conference*.

Kagermann, H., Wahlster, W., and Helbig, J. (2013). Recommendations for implementing the strategic initiative industrie 4.0. Final report of the industrie 4.0 working group, National Academy of Science and Engineering.

Kahn, G., Villaflor, A., Pong, V., Abbeel, P., and Levine, S. (2017). Uncertainty-Aware Reinforcement Learning for Collision Avoidance.

Karikomi, K. et al. (2015). Wind tunnel testing on negative-damped responses of a 7mw floating offshore wind turbine. In *EWEA 2015*.

Kennedy, M. and O'Hagan, A. (1998). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87.

Korenhof, P., Blok, V., and Kloppenburg, S. (2021). Steering representations—towards a critical understanding of digital twins. *Philosophy & Technology*, 34(4):1751–1773.

Krstić, M. (2000). Performance improvement and limitations in extremum seeking control. *Systems & Control Letters*, 39(5):313–326.

Kumar, V. and Michael, N. (2012). Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research*, 31(11):1279–1291.

Lackner, M. A. (2009). Controlling platform motions and reducing blade loads for floating wind turbines. *Wind Engineering*, 33(6):541–553.

Larsen, T. J. and Hanson, T. D. (2007). A method to avoid negative damped low frequent tower vibrations for a floating, pitch controlled wind turbine. In *Journal of Physics: Conference Series*, volume 75. IOP.

Lecomte, Y., Ratz, M., and Mendez, M. A. (2024). Reinforcement twinning for attitude control of multirotordrones: an experimental proof of concept. Technical report, von Karman Institute for Fluid Dynamics.

Lecomte, Y., Vardanyan, A., Schram, C., and Mendez, M. A. (2025). Reinforcement twinning for attitude control of multirotor drones: an experimental proof of concept. In *AIFluids conference*.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 363–369.

Lee, E. A. and Seshia, S. A. (2015). *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. MIT Press.

Leugering, G., Engell, S., Griewank, A., Hinze, M., Rannacher, R., Schulz, V., Ulbrich, M., and Ulbrich, S., editors (2011). *Constrained Optimization and Optimal Control for Partial Differential Equations*. SpringerLink. Springer Basel, Basel.

Li, X. and Gao, H. (2015). Load mitigation for a floating wind turbine via generalized h_∞ structural control. *IEEE Transactions on Industrial Electronics*, 63(1):332–342.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning.

Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall, 2nd edition.

Marques, P. A., Ahizi, S., and Mendez, M. A. (2024). Real-time data assimilation for the thermodynamic modeling of cryogenic storage tanks. *Energy*, 302:131739.

Mazzaretto, O. M., Menéndez, M., and Lobeto, H. (2022). A global evaluation of the jonswap spectra suitability on coastal areas. *Ocean Engineering*, 266:112756.

Mesbah, A. (2018). Stochastic model predictive control with active uncertainty learning: A Survey on dual control. *Annual Reviews in Control*, 45:107–117.

Moerland, T. M., Broekens, J., and Jonker, C. M. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118. (Preprint 2022).

Moore, J., Ryall, T., and Xia, L. (1989). Central tendency adaptive pole assignment. *IEEE Transactions on Automatic Control*, 34(3):363–367.

Nagabandi, A. et al. (2018). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*.

Namik, H., Rotea, M., and Lackner, M. (2013). Active structural control with actuator dynamics on a floating wind turbine. In *51st AIAA Aerospace Sciences Meeting*, page 455.

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.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Peng, S. (1992). Stochastic Hamilton-Jacobi-Bellman equations. *SIAM Journal on Control and Optimization*, 30(2):284–304.

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.

Poletti, R., Schena, L., Koloszar, L., Degroote, J., and Mendez, M. A. (2025). Reinforcement twinning for hybrid control of flapping-wing drones.

Polydoros, A. S. and Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86:153–173.

Randino, S., Schena, L., Coudou, N., Garone, E., and Mendez, M. A. (2026). Nonlinear system identification for model-based control of waked wind turbines. *Data-Centric Engineering*, 7.

Ravindran, S. S. (2000). Reduced-order modeling of approximations to the Navier-Stokes equations. *Computers & Mathematics with Applications*, 40(10-11):1319–1334.

Robertson, A., Jonkman, J., Masciola, M., Song, H., Goupee, A., Coulling, A., and Luan, C. (2014). Definition of the semisubmersible floating system for phase ii of oc4. Technical report, National Renewable Energy Laboratory (NREL), Golden, CO.

Rowley, C. W. and Dawson, S. T. M. (2017). Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417.

Ryall, T. and Moore, J. (1989). Central tendency minimum variance adaptive control. *IEEE Transactions on Automatic Control*, 34(3):367–371.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay.

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.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation. *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

Schwenzer, M., Ay, M., Bergs, T., and Abel, D. (2021). Review on model predictive control: an engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117:1327–1349.

Sicari, S., Rizzardi, A., Grieco, L. A., and Coen-Porisini, A. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164.

Sritharan, S. S., editor (1998). *Optimal Control of Viscous Flow*. SIAM.

Stengel, R. F. (1994). *Optimal Control and Estimation*. Dover Publications.

Stockhouse, D., Phadnis, M., Grant, E., Johnson, K., Damiani, R., and Pao, L. (2022). Control of a floating wind turbine on a novel actuated platform. In *2022 American Control Conference (ACC)*, pages 3532–3537.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT press.

Tao, F., Qi, Q., Liu, A., and Nee, A. Y. C. (2019). Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison. *Engineering*, 5(4):653–661.

Veers, P., Dykes, K., Lantz, E., Barth, S., Bottasso, C. L., Carlson, O., Clifton, A., Green, J., Green, P., Holttinen, H., Laird, D., Lehtomäki, V., Lundquist, J. K., Manwell, J., Marquis, M., Meneveau, C., Moriarty, P., Munduate, X., Muskulus, M., Naughton, J., Pao, L., Paquette, J., Peinke, J., Robertson, A., Sanz Rodrigo, J., Sempreviva, A. M., Smith, J. C., Tuohy, A., and Wiser, R. (2019). Grand challenges in the science of wind energy. *Science*, 366(6464).

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.

Werner, R. et al. (2023). Deep reinforcement learning for flow control exploits different physics for increasing Reynolds number regimes. *Fluids*.

Wittenmark, B. (1995). Adaptive Dual Control Methods: An Overview. *IFAC Proceedings Volumes*, 28(13):67–72.

Wittenmark, B. and Elevitch, C. (1985). An Adaptive Control Algorithm with Dual Features. *IFAC Proceedings Volumes*, 18(5):587–592.

Åström, K. J. and Helmersson, A. (1986). Dual control of an integrator with unknown gain. *Computers & Mathematics with Applications*, 12(6, Part A):653–662.

Åström, K. J. and Wittenmark, B. (2008). *Adaptive Control*. Dover Publications.