MOX-Report No. 51/2025

Latent feedback control of distributed systems in multiple scenarios through deep learning-based reduced order models

Tomasetto, M.; Braghin, F., Manzoni, A.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it https://mox.polimi.it

Latent feedback control of distributed systems in multiple scenarios through deep learning-based reduced order models

Matteo Tomasetto[a], Francesco Braghin[a], Andrea Manzoni[b]

[a] *Department of Mechanical Engineering, Politecnico di Milano, Via privata Giuseppe La Masa 1, 20156, Milano, Italy*
[b] *MOX – Department of Mathematics, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133, Milano, Italy*

Abstract

Continuous monitoring and real-time control of high-dimensional distributed systems are often crucial in applications to ensure a desired physical behavior, without degrading stability and system performances. Traditional feedback control design that relies on full-order models, such as high-dimensional state-space representations or partial differential equations, fails to meet these requirements due to the delay in the control computation, which requires multiple expensive simulations of the physical system. The computational bottleneck is even more severe when considering parametrized systems, as new strategies have to be determined for every new scenario. To address these challenges, we propose a real-time closed-loop control strategy enhanced by nonlinear non-intrusive Deep Learning-based Reduced Order Models (DL-ROMs). Specifically, in the offline phase, *(i)* full-order state-control pairs are generated for different scenarios through the adjoint method, *(ii)* the essential features relevant for control design are extracted from the snapshots through a combination of Proper Orthogonal Decomposition (POD) and deep autoencoders, and *(iii)* the low-dimensional policy bridging latent control and state spaces is approximated with a feedforward neural network. After data generation and neural networks training, the optimal control actions are retrieved in real-time for any observed state and scenario. In addition, the dynamics may be approximated through a cheap surrogate model in order to close the loop at the latent level, thus continuously controlling the system in real-time even when full-order state measurements are missing. The effectiveness of the proposed method, in terms of computational speed, accuracy, and robustness against noisy data, is finally assessed on two different high-dimensional optimal transport problems, one of which also involving an underlying fluid flow.

Keywords: feedback control, PDE-constrained optimization, parametrized systems, reduced order modeling, deep learning, optimal transport

1. Introduction

Monitoring and controlling dynamical systems play a central role in Applied Sciences and Engineering. A control strategy enables to steer and influence the behaviour of a physical phenomenon in order to achieve desired outcomes. Fascinating examples in this direction are given by self-driving cars navigating traffic, spacecraft maneuvering through space, and emission reduction of industrial processes. Distributed dynamical systems are often described by high-dimensional state-space representations or Partial Differential Equations (PDEs). The space-time evolution of the state variable may be retrieved through full-order models, such as, e.g., finite differences, finite elements or finite volumes, yielding to a (possibly nonlinear) system of equations to be solved. This task is computationally demanding, especially when dealing with complex and stiff phenomena, since a large number of degrees of freedom are required to discretize and capture the state dynamics. The computational bottleneck becomes even more severe when *(i)* considering systems parametrized by a vector of scenario parameters $\boldsymbol{\mu}_s$, as different independent resolutions are required for every scenario of interest, and when *(ii)* dealing with Optimal Control Problems (OCPs). Indeed, optimal control values are typically determined through multiple full-order resolutions of the physical model within an optimization procedure minimizing a problem-specific *loss* or *cost functional* [50], making real-time control unfeasible. However, delayed control strategies may degrade the overall optimality, stability and system performances, which are often vital in applications. Our main goal is to overcome the computational barrier of traditional full-order solvers by devising an adaptive and instantaneous feedback control strategy in multiple scenarios for high-dimensional parametrized PDEs.

Feedback control design is traditionally addressed with Dynamic Programming Principle [10] and Hamilton-Jacobi-Bellman equation (HJB) [9]. Despite attempts to circumvent and reduce the problem complexity in this framework – see, e.g., [44, 4, 64, 3, 2] – numerical schemes suffer from the curse of dimensionality and become computationally intractable as the state and control dimensions increase, thus limiting its applicability in real-time control scenarios.

Instead of directly solving the HJB equation, the feedback controller may be determined exploiting Reinforcement Learning [71] or Deep Reinforcement Learning (DRL) [28, 78, 60] algorithms. Specifically, the so-called policy or agent – that is modeled through a neural network in DRL – learns to predict the best possible control action from the current observed state. In practice, policy training is carried out through a continuous interaction with the dynamical system, maximizing the reward signal computed by looking at the system response to the applied control. This strategy has been explored in several applications, ranging from flow control [76, 20, 61, 74, 31, 77, 79], metamaterial design [65, 51, 62], chemistry [80], optimal navigation in turbulent flows [73, 13] and swarm systems control [36]. However, standard DRL approaches are limited to low-dimensional state and control spaces due to sample inefficiency and computationally demanding training phases, especially when dealing with parametrized problems. To overcome these drawbacks, efficient, robust and interpretable policies have been proposed by [81, 15] leveraging sparsity and dictionary learning. Instead, [48, 28] and [14] exploit, respectively, autoencoders and a priori knowledge to cope with high-dimensional observations. Similar state compressions are employed by [47] in the context of robust control, and by [37] to provide a fast linear quadratic regulator feedback controller. Moreover, distributed actions may be considered through Multi-Agent Reinforcement Learning (MARL) strategies [21, 75, 57], where the dynamics is controlled by means of multiple local agents relying solely on local state values. However, the locality assumption compromises a smooth and non-local coordination among agents, that is typically crucial when controlling physical systems [38].

Another advanced technique for closed-loop control of dynamical systems is Model Predictive Control (MPC) [22]. In the MPC context, multiple open-loop optimization problems are solved as the dynamics evolves in time, iteratively updating the information on the current state looking at the system evolution. Rather than full-order models describing the dynamics at hand, MPC usually considers (possibly nonlinear) fast-evaluable surrogate models to quickly predict the state evolution over a prediction horizon starting from the current observed state. This approximation speeds up the optimization procedure aiming at computing the optimal control sequence in time by minimizing a suitable cost functional. Several techniques have been exploited to identify (possibly low-dimensional) surrogate models for the system dynamics – such as neural networks [24, 1, 23, 12, 54, 8], POD [30, 5], Sparse Identification of Nonlinear Dynamics (SINDy) [39], Eigensystem Realization Algorithm (ERA) [35] – or to approximate the Koopman operator – such as Dynamic Mode Decomposition (DMD) [41, 55, 56, 40] – in the MPC framework. For further details on DMD, SINDy and Koopman theory see, e.g., [63, 18, 19, 16, 17]. However, MPC may be not suitable for real-time applications with strict timing requirements and fast dynamics due to the multiple optimization problems involved online. Moreover, since the control is regarded as model input, it would be challenging to consider distributed control variables with a remarkably high number of degrees of freedom.

In this work, we design a real-time feedback control strategy capable of steering the dynamics of distributed systems described in terms of parametrized PDEs in multiple scenarios. Differently from the DRL approach, we focus on an offline-online decomposition, in the direction of imitation learning or offline reinforcement learning [46], in order to reduce the dimensionality of distributed and boundary control actions, as well as high-dimensional state observations. As far as the dimensionality reduction is concerned, we consider nonlinear non-intrusive Deep Learning-based Reduced Order Models (DL-ROMs), where data are compressed through Proper Orthogonal Decomposition (POD) [33], deep autoencoders [26, 25], or a combination thereof [27], while a feedforward neural network approximates low-dimensional policy bridging state and control latent spaces. Nonlinearity and non-intrusiveness allow for greater flexibility and speed-up with respect to traditional reduced order modeling (ROM) techniques, such as the Reduced Basis (RB) method [34, 59, 16, 49]. Indeed, despite several applications in the context of OCPs [42, 45, 43, 53, 11, 7, 69], the

RB method is not efficient when dealing with nonlinear or transport-dominated problems.

The proposed approach extends our previous work on real-time open-loop control for parametrized PDEs [72], where we approximate the optimal state and control variables starting from the corresponding scenario parameters $\boldsymbol{\mu}_s$ only, disregarding possible additional state measurements collected online. Whenever the latter data are available, it is possible to continuously inform the controller about the current system behaviour through a feedback loop, thus broadening the control strategy to challenging applications where the scenario parameters do not describe the whole problem variability. For example, in the optimal transport test cases detailed in Section 4, the scenario parameters identify the target location, while the current configuration is captured directly from the observed state, as it typically happens in realistic settings. Moreover, as shown in the test cases taken into account, the dimensionality reduction and the feedback signal are also helpful to gain robustness against uncertainties and deal with noisy state data, paving the way for sensor-based applications.

Inspired by the MPC strategy, we propose a model closure at the latent level in order to control the dynamical system even when state data are not available online due to, e.g., sensor failures, delay in receiving the measurements or time-consuming simulations. Specifically, in our framework, the latent dynamics is approximated through a low-dimensional deep learning-based surrogate model, which quickly predicts the reduced state evolution starting from the current state and control information available at the latent level. This allows us to obtain a self-contained controller capable of dealing with applications where continuous monitoring is unfeasible, or where the computational burden of full-order simulations does not meet the real-time requirement.

The paper is organized as follows. Section 2 briefly reviews open and closed-loop optimal control problems constrained by parametrized PDEs. Section 3 presents the real-time deep learning-based reduced order feedback control, delving the steps required in the offline and online phases, as well as the latent feedback loop. Section 4 shows the performance of the proposed approach when dealing with two challenging optimal transport problems. Section 5 discusses some possible future developments and extensions of the presented methodology.

2. From open-loop to feedback control of parametrized systems

This section briefly introduces the class of optimization problems investigated throughout this work, explaining the role of the main variables and setting up the notation. We consider Optimal Control Problems (OCPs) for distributed dynamical systems, whose state evolutions can be described in terms of Partial Differential Equations (PDEs). The main goal in this context is to steer the physical system at hand towards a target configuration, influencing the state behaviour by optimally tuning a suitable control variable. Mathematically speaking, this is achieved through the following constrained optimization problem:

$$\text{Given } \mathbf{y}_0 \text{ and } \boldsymbol{\mu}_s, \quad \text{find} \quad J_h\left(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s\right) \to \min \quad \text{s.t.} \quad \begin{cases} \mathbf{G}_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) = \mathbf{0} & \text{in } (0, T] \\ \mathbf{y}_h(0) = \mathbf{y}_0 \end{cases} \tag{1}$$

where $\mathbf{y}_h(t) : [0,T] \to \mathbb{R}^{N_h^y}$ and $\mathbf{u}_h(t) : [0,T] \to \mathbb{R}^{N_h^u}$ denote, respectively, the semi-discrete state and control, obtained discretizing the PDE through numerical techniques, such as, e.g., the Finite Element Method (FEM) [58]. Specifically, after splitting the domain Ω – that is the region where the physical phenomenon takes place – in sub-elements, the (scalar, for the sake of simplicity) infinite-dimensional state $y(\mathbf{x}, t) : \Omega \times [0,T] \to \mathbb{R}$ and control $u(\mathbf{x}, t) : \Omega \times [0,T] \to \mathbb{R}$ are approximated in suitable finite-dimensional spaces $\mathcal{Y}_h$ and $\mathcal{U}_h$ through basis expansions, that is

$$y(\mathbf{x}, t) \approx \sum_{i=1}^{N_h^y} \mathbf{y}_{h,i}(t) \xi_i^y(\mathbf{x}); \qquad u(t) \approx \sum_{i=1}^{N_h^u} \mathbf{u}_{h,i}(t) \xi_i^u(\mathbf{x})$$

where $\{\mathbf{y}_{h,i}(t)\}_{i=0}^{N_h^y}$ and $\{\mathbf{u}_{h,i}(t)\}_{i=0}^{N_h^u}$ are the elements of the vectors $\mathbf{y}_h(t)$ and $\mathbf{u}_h(t)$, while $\{\xi_i^y(\mathbf{x})\}_{i=0}^{N_h^y}$ and $\{\xi_i^u(\mathbf{x})\}_{i=0}^{N_h^u}$ are the space-varying basis functions spanning $\mathcal{Y}_h$ and $\mathcal{U}_h$, respectively. The discretization parameter $h > 0$ corresponds to the characteristic dimension of the sub-elements: the smaller h, the better the FEM approximation, at the price of more degrees of freedom and more expensive computations. In the following, we assume complete access to the state values within the domain Ω. However, the proposed approach can be easily extended in order to deal with different observed quantities defined as

$$\mathbf{z}_h(t) = O_h(t)\mathbf{y}_h(t) \in \mathbb{R}^{N_h^z} \quad \text{where} \quad O_h(t) : \mathbb{R}^{N_h^y} \to \mathbb{R}^{N_h^z} \quad \forall t \in [0, T],$$

allowing for partially observable states or state-related markers of the actual physical configuration.

The real-valued *loss* or *cost functional* $J_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s)$ in Equation (1) evaluates the effectiveness of a specific state-control trajectory with respect to the target configuration: the lower the cost, the higher the performance of that control action. The optimal control trajectory, which brings the dynamical system as close to the target as possible, and the corresponding optimal state, can be thus computed by minimizing the loss functional J_h.

Remark. Whenever a target configuration $\mathbf{y}_d \in \mathbb{R}^{N_h^y}$ has to be reached at final time $t = T$, the following loss functional may be employed

$$J_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) = \frac{1}{2}||\mathbf{y}_h(T) - \mathbf{y}_d||^2 + \frac{\beta}{2}\int_0^T ||\mathbf{u}_h(t)||^2 dt$$

where $\|\cdot\|$ stands for the Euclidean norm. If, instead, a target trajectory $\mathbf{y}_d(t)$ must be followed throughout the time interval $[0, T]$, we may consider

$$J_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) = \frac{1}{2}\int_0^T ||\mathbf{y}_h(t) - \mathbf{y}_d(t)||^2 dt + \frac{\beta}{2}\int_0^T ||\mathbf{u}_h(t)||^2 dt.$$

Note that, as detailed in [50], regularization terms concerning the norm of the control and its gradient are generally considered to guarantee the well-posedness of the OCP, as well as to prevent unfeasible and overconsuming control actions. The parameter $\beta > 0$ has to be chosen in order to properly balance the terms in the cost functional: the lower β, the closer the optimal state to the target configuration, the bigger the optimal control norm, and the harder the optimization procedure.

The constraint in Equation (1) is crucial to guarantee the physical admissibility of the optimal state-control pair. The state and the control variables are indeed coupled by the dynamical system $\mathbf{G}_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) = \mathbf{0}$ with $\mathbf{G}_h \in \mathbb{R}^{N_h^y}$ – for the sake of simplicity, the state system is here considered directly in the semi-discrete formulation, which is regarded as the high-fidelity Full-Order Model (FOM) – governing the evolution of the state system within the domain Ω and in the time interval $(0, T]$. In this work, we take into account dynamics described in terms of (possibly nonlinear) PDEs parametrized by a vector of scenario parameters $\boldsymbol{\mu}_s \in \mathcal{P} \subset \mathbb{R}^p$, where $\mathcal{P}$ denotes the parameter space. The presence of parameters identifying the scenario variability to address increases the OCP complexity, as we aim to optimally control the dynamics in real-time for multiple scenarios. For example, in the applications detailed in Section 4, $\boldsymbol{\mu}_s$ represents the target endpoint of the state trajectory, while the velocity field driving the state is regarded as control. Therefore, changes in the scenario entail completely different state routes and optimal controls to apply. Note that, along with the initial condition $\mathbf{y}_h(0) = \mathbf{y}_0$, suitable boundary conditions must be set on $\partial\Omega \times (0, T]$ in order to guarantee the well-posedness of the PDE and the corresponding OCP [50]. Besides the physical constraint in Equation (1), additional constraints may be taken into account whenever the state and the control are subject to extra physical or practical requirements. For example, it is possible to properly define the set of admissible state and control values and require that

$$\mathbf{y}_h(t) \in \mathcal{Y}_{ad} \subset \mathbb{R}^{N_h^y}, \quad \mathbf{u}_h(t) \in \mathcal{U}_{ad} \subset \mathbb{R}^{N_h^u} \quad \forall t \in [0, T].$$

In the following, without loss of generality, no additional constraints are considered, that is $\mathcal{Y}_{ad} = \mathbb{R}^{N_h^y}$ and $\mathcal{U}_{ad} = \mathbb{R}^{N_h^u}$.

Remark. Consider the semi-discrete formulation of a nonlinear time-dependent PDE

$$\mathbf{G}_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) = M_h(\boldsymbol{\mu}_s)\frac{d\mathbf{y}_h(t)}{dt} + A_h(\boldsymbol{\mu}_s)\mathbf{y}_h(t) + C_h(\mathbf{y}_h(t); \boldsymbol{\mu}_s) - \mathbf{f}_h(t; \boldsymbol{\mu}_s) - B_h(\mathbf{u}_h(t); \boldsymbol{\mu}_s) = \mathbf{0}$$

where $M_h(\boldsymbol{\mu}_s)$ is the mass matrix, $A_h(\boldsymbol{\mu}_s)$ is the stiffness matrix, $B_h(\boldsymbol{\mu}_s)$ is the control matrix, $C_h(\boldsymbol{\mu}_s)$ is the nonlinear term, and $\mathbf{f}_h(t; \boldsymbol{\mu}_s)$ is the external source. Let $\{t_j\}_{j=0}^{N_t}$ an evenly spaced grid with step Δt

discretizing the interval $[0, T]$. After approximating the time derivative through, e.g., semi-implicit Euler method, the N_h^y basis expansion coefficients $\mathbf{y}_h(t_{j+1})$ for $j = 0, ..., N_t - 1$ can be computed by solving the associated system of N_h^y equations

$$[M_h(\boldsymbol{\mu}_s) + \Delta t A_h(\boldsymbol{\mu}_s)]\mathbf{y}_h(t_{j+1}) = M_h(\boldsymbol{\mu}_s)\mathbf{y}_h(t_j) - \Delta t C_h(\mathbf{y}_h(t_j); \boldsymbol{\mu}_s) + \Delta t \mathbf{f}_h(t_{j+1}, \boldsymbol{\mu}_s) + \Delta t B_h(\mathbf{u}_h(t_j); \boldsymbol{\mu}_s)$$

The OCP in Equation (1) provides, once solved, the open-loop optimal control trajectory related to the initial state $\mathbf{y}_0$ and the scenario $\boldsymbol{\mu}_s$ under investigation. See, e.g., [50] for a complete presentation and analysis of open-loop OCPs. Nevertheless, the open-loop solution is vulnerable to disturbances and uncertainties in the system, and may compromise system stability and performance. To address this drawback, feedback control strategies may be exploited, in which the considered OCP is solved repeatedly over the time horizon, continuously updating the initial state with the current observed state, as depicted in Figure 1 with reference to the first test case detailed in Section 4. In practice, the delay between the availability of the current state and the computation of the subsequent optimal action is unavoidable, making this strategy not suitable for applications with strict timing requirements. The time lag becomes even larger when online sensor data are lacking and synthetic state data must be simulated at each time step, especially in case of a large full-order dimension N_h^y. The computational bottleneck may be overcome by fast-evaluable surrogate models approximating the full-order dynamics or by merely considering low-dimensional variables, as often happens in the Model Predictive Control (MPC) context, as detailed in Section 1. However, in order to design a real-time feedback controller capable of dealing with high-dimensional variables, as well as complex nonlinear dynamics, we aim to avoid any optimization procedure online, concentrating the computationally expensive steps in an offline phase to be performed only once.

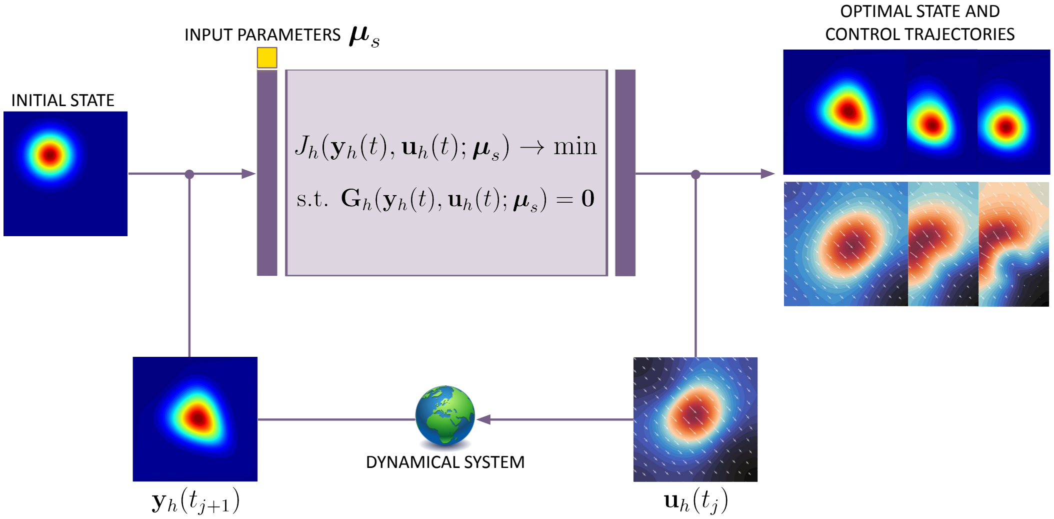
INPUT PARAMETERS $\boldsymbol{\mu}_s$
OPTIMAL STATE AND CONTROL TRAJECTORIES
INITIAL STATE
$J_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) \to \min$
s.t. $\mathbf{G}_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) = \mathbf{0}$
DYNAMICAL SYSTEM
$\mathbf{y}_h(t_{j+1})$
$\mathbf{u}_h(t_j)$

Figure 1. Feedback control scheme considering multiple optimal control problem resolutions and a high-fidelity full-order model of the dynamical system. Given the current state and the scenario parameters $\boldsymbol{\mu}_s$, the open-loop optimal control and state trajectories are retrieved by solving the optimal control problem. The feedback signal is then recovered by measuring or simulating the state at the next time step.

Similarly to the Reinforcement Learning (RL) framework, it is convenient to rephrase the problem focusing on the approximation of the so-called policy function

$$\pi_h : \mathbb{R}^{N_h^y} \times \mathcal{P} \to \mathbb{R}^{N_h^u}, \quad \mathbf{u}_h = \pi_h(\mathbf{y}_h, \boldsymbol{\mu}_s) \quad \text{where} \quad \mathbf{u}_h = \arg\min J_h(\mathbf{y}_h(\mathbf{u}_h), \mathbf{u}_h; \boldsymbol{\mu}_s) \tag{2}$$

where, for the sake of compactness, the reduced cost functional $J_h(\mathbf{y}_h(\mathbf{u}_h), \mathbf{u}_h; \boldsymbol{\mu}_s)$ directly considers the state-control dependence $\mathbf{y}_h(\mathbf{u}_h)$ given implicitly by the PDE $\mathbf{G}_h(\mathbf{y}_h, \mathbf{u}_h; \boldsymbol{\mu}_s) = \mathbf{0}$. The policy function corresponds to the one time-step solution map of the OCP in Equation (1): indeed, starting from the current observed state $\mathbf{y}_h$ and the scenario parameters $\boldsymbol{\mu}_s$, it returns the corresponding subsequent optimal control minimizing the loss J_h. Note that, since the policy returns only the current optimal control in place of the entire trajectory over time, the time t is not made explicit anymore. To recover the optimal control at every time instant, multiple policy evaluations are required with a continuous state update. As detailed in the next section, the policy input-output dimensions may be high-dimensional – especially when dealing with distributed state and control variables showing large N_h^y and N_h^u full-order dimensions – and model order reduction techniques are crucial to retrieve faster and lighter map.

3. Deep learning-based reduced order feedback control

This section presents the non-intrusive deep learning-based ROM strategy aiming at providing the optimal control action starting from an observed state and a scenario encoded in a vector of input parameters $\boldsymbol{\mu}_s$. Our approach focuses on an offline-online decomposition, where *(i)* the offline phase is concerned with the computationally expensive steps needed to build and calibrate a low-dimensional fast-evaluable surrogate model for the policy function, such as data generation, dimensionality reduction and neural networks training, while *(ii)* real-time optimal control actions may be inferred by evaluating the constructed policy in the online phase. This pipeline is inspired by the non-intrusive ROMs proposed by [33, 26, 27] – namely POD-NN, DL-ROM and POD-DL-ROM – and adapted to a feedback control framework. The proposed offline-online decomposition is crucial in a twofold manner: first of all, optimal control snapshots are generated and properly reduced offline, thus enabling us to easily handle distributed or boundary controls. This would not be easily implemented within, e.g., RL strategies, where the learning requires a continuous interaction between the agent and the dynamical system. Moreover, in contrast with the MPC framework, the optimization procedures are performed during data generation (offline), thus making it possible to retrieve the optimal control actions (online) through fast policy evaluations. The next sections explore in details the different steps required to build and employ the proposed method.

3.1. Offline phase

The rationale of the proposed approach is to approximate the policy function in Equation (2) in a supervised manner through a feedforward neural network. To do so, *(i)* state-control pairs have to be generated exploring different initial conditions $\mathbf{y}_0$ and different scenarios $\boldsymbol{\mu}_s$ in the parameter space $\mathcal{P}$, and *(ii)* neural network training must be performed to synthesize a surrogate policy that accurately predicts the optimal control starting from the corresponding observed state. Since the full-order state and control dimensions N_h^y and N_h^u may be remarkably high, an intermediate step is taken into account to compress the snapshots dimensionality and sharply reduce the input-output policy layers.

Data generation. The first preparatory procedure to build a policy approximation through supervised learning strategies is the generation of policy input-output pairs, that are optimal state and control trajectories. In order to acquire a robust policy capable of controlling the dynamical system across a wide variety of configurations, it is crucial to extensively explore different initial states and scenarios. While sampling strategies are usually enough to explore the (typically low-dimensional) parameter space $\mathcal{P}$, smart choices are required to sample from the (possibly high-dimensional) state space $\mathbb{R}^{N_h^y}$. For instance, in the applications detailed in Section 4, we parametrize the initial state field, so that state variability is reduced, while still covering all the possible relevant cases of interest that may occur. Every optimal state and control trajectory can be computed by solving the OCP in Equation (1) while considering $\mathbf{y}_0 = \mathbf{y}_0^{(i)}$ and $\boldsymbol{\mu}_s = \boldsymbol{\mu}_s^{(i)}$ for $i = 1, ..., N_s$. To this aim, it is possible to solve the system of Karush-Kuhn-Tucker (KKT) optimality conditions derived from Equation (1) via Lagrange multipliers' method, that is

$$\begin{cases} \nabla_y J_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) + (\partial_y \mathbf{G}_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s))^\top \mathbf{p}_h(t) = \mathbf{0} & \text{(adjoint equation)} \\ \mathbf{p}_h(T) = \nabla_y J_h(\mathbf{y}_h(T), \mathbf{u}_h(T); \boldsymbol{\mu}_s) & \text{(final condition)} \\ \nabla_u J_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s) + (\partial_u \mathbf{G}_h(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s))^\top \mathbf{p}_h(t) = \mathbf{0} & \text{(optimality condition)} \\ \mathbf{G}_h\left(\mathbf{y}_h(t), \mathbf{u}_h(t); \boldsymbol{\mu}_s\right) = \mathbf{0} & \text{(state equation).} \\ \mathbf{y}_h(0) = \mathbf{y}_0 & \text{(initial condition)} \end{cases} \tag{3}$$

where $\mathbf{p}_h(t) \in \mathbb{R}^{N_h^p}$ for $t \in [0, T]$ is the semi-discrete adjoint vector. For a complete presentation of the OCP solving methods see, e.g., [50].

Remark. In case of final target observation, which corresponds to the first example presented in Remark 2, the final condition for the adjoint equation ends up being

$$\mathbf{p}_h(T) = \mathbf{y}_h(T) - \mathbf{y}_d.$$

Instead, whenever the target state observation is distributed in $[0, T]$ as in the second example of Remark 2, the final condition becomes

$$\mathbf{p}_h(T) = \mathbf{0}.$$

The system of optimality conditions in Equation (3) returns – after being discretized through FEM and solved via, e.g., the Newton method – the optimal state and control trajectories

$$\{\mathbf{y}_h^{(i)}(t_0), \mathbf{y}_h^{(i)}(t_1), ..., \mathbf{y}_h^{(i)}(t_{N_t-1}), \mathbf{y}_h^{(i)}(t_{N_t})\}_{i=1,...,N_s}, \qquad \{\mathbf{u}_h^{(i)}(t_0), \mathbf{u}_h^{(i)}(t_1), ..., \mathbf{u}_h^{(i)}(t_{N_t-2}), \mathbf{u}_h^{(i)}(t_{N_t-1})\}_{i=1,...,N_s}$$

where $\{t_j\}_{j=0}^{N_t}$ introduces a uniform partition of the time horizon $[0, T]$. Starting from the generated trajectories, it is then possible to assemble the $N_t N_s$ input-output pairs useful for training and testing the policy surrogate model, that is

$$\{(\mathbf{y}_h^{(i)}(t_j), \boldsymbol{\mu}_s^{(i)}), \mathbf{u}_h^{(i)}(t_j)\}_{\substack{i=1,...,N_s \\ j=0,...,N_t-1}} \qquad \text{where} \qquad (\mathbf{y}_h^{(i)}(t_j), \boldsymbol{\mu}_s^{(i)}) \in \mathbb{R}^{N_h^y} \times \mathcal{P}, \quad \mathbf{u}_h^{(i)}(t_j) \in R^{N_h^u}.$$

In the following, without loss of generality, whenever the dependence on time is not made explicit, the snapshots are reordered and denoted by

$$\{\mathbf{y}_h^{(k)}, \boldsymbol{\mu}_s^{(k)}, \mathbf{u}_h^{(k)}\}_{k=1,...,N_t N_s}.$$

Dimensionality reduction. Reduced Order Models (ROMs) are very helpful when dealing with parametrized or many-query problems, as the number of degrees of freedom into play may be dramatically reduced by extracting the (few) essential features relevant for the problem, ending up with faster (but still very accurate) methods. The dimensionality reduction is typically performed by projecting the available snapshots onto lower-dimensional subspaces, which may be linear – such as in the Proper Orthogonal Decomposition (POD) framework – or nonlinear – as when autoencoders (AEs) are employed.

- **Proper Orthogonal Decomposition**: full-order state and control snapshots may be projected onto linear subspaces of dimensions, respectively, $N_y \ll N_h^y$ and $N_u \ll N_h^u$ by POD [70]. The N_y and N_u basis elements (also known as POD modes) spanning the two latent subspaces are computed by the Singular Values Decomposition (SVD) of the state and control snapshots matrices, and are then collected column-wise in the matrices $\mathbb{V}_y \in \mathbb{R}^{N_h^y \times N_y}$ and $\mathbb{V}_u \in \mathbb{R}^{N_h^u \times N_u}$. The resulting projections thus read as follows:

  $$\mathbf{y}_N = \mathbb{V}_y^\top \mathbf{y}_h, \qquad \mathbf{y}_{h,\text{rec}} = \mathbb{V}_y \mathbf{y}_N$$
  $$\mathbf{u}_N = \mathbb{V}_u^\top \mathbf{u}_h, \qquad \mathbf{u}_{h,\text{rec}} = \mathbb{V}_u \mathbf{u}_N$$

  where $\mathbf{y}_{h,\text{rec}} \in \mathbb{R}^{N_h^y}$ and $\mathbf{u}_{h,\text{rec}} \in \mathbb{R}^{N_h^u}$ are the full-order state and control projections approximating $\mathbf{y}_h$ and $\mathbf{u}_h$ up to a reconstruction error. Therefore, instead of dealing with the full-order state and control variables, it is possible to focus on a set of low-dimensional features extracted from their basis expansions coefficients $\mathbf{y}_N \in \mathbb{R}^{N_y}$ and $\mathbf{u}_N \in \mathbb{R}^{N_u}$, respectively. Note that the latent dimensions N_y and N_u are usually selected by a trade-off between having small latent subspaces and low reconstruction errors [34, 59].

- **Autoencoders**: linear ROMs may require a remarkably high number of POD modes in complex settings characterized by, e.g., involved scenario-state or scenario-control dependencies, as well as nonlinear or transport-dominated PDEs, compromising the speed up of the method. To recover more effective embeddings, projections onto nonlinear subspaces may be exploited, under the form

  $$\mathbf{y}_N = \varphi_E^y(\mathbf{y}_h), \qquad \mathbf{y}_{h,\text{rec}} = \varphi_D^y(\mathbf{y}_N)$$
  $$\mathbf{u}_N = \varphi_E^u(\mathbf{u}_h), \qquad \mathbf{u}_{h,\text{rec}} = \varphi_D^u(\mathbf{u}_N)$$

where $\varphi_E^y : \mathbb{R}^{N_h^y} \to \mathbb{R}^{N_y}$, $\varphi_E^u : \mathbb{R}^{N_h^u} \to \mathbb{R}^{N_u}$, $\varphi_D^y : \mathbb{R}^{N_y} \to \mathbb{R}^{N_h^y}$ and $\varphi_D^u : \mathbb{R}^{N_u} \to \mathbb{R}^{N_h^u}$ are nonlinear functions compressing high-dimensional snapshots into latent representations and viceversa. Throughout this work, we model encoding and decoding mappings through the so-called autoencoders, which are neural networks showing a bottleneck architecture. The nonlinearity feature is due to the activation functions exploited within the autoencoders, such as the leaky ReLU function, as taken into account in the following. Note that, as proposed by [26, 25] in the framework of Deep Learning-based Reduced Order Models (DL-ROMs), convolutional autoencoders may be employed for a remarkable dimensionality reduction.

- **POD+AE**: the speed of POD and the power of autoencoders can also be combined, as initially proposed by [27] in the POD-DL-ROM framework. In this context – here referred to as POD+AE – autoencoders are trained to further compress the dimensionality of snapshots that has been preliminarily reduced by POD, that is,

  $$\begin{aligned} \mathbf{y}_N &= \varphi_E^y(\mathbb{V}_y^\top \mathbf{y}_h), \quad & \mathbf{y}_{h,\text{rec}} &= \mathbb{V}_y \varphi_D^y(\mathbf{y}_N) \\ \mathbf{u}_N &= \varphi_E^u(\mathbb{V}_u^\top \mathbf{u}_h), \quad & \mathbf{u}_{h,\text{rec}} &= \mathbb{V}_u \varphi_D^u(\mathbf{u}_N). \end{aligned}$$

  In this way we can obtain lighter NNs architecture and faster trainings, without compromising the overall accuracy.

Low-dimensional policy. After selecting a suitable compression technique to reduce state and control data, it is possible to define a feedforward neural network approximating the policy function at the latent level, that is,

$$\pi_N : \mathbb{R}^{N_y} \times \mathcal{P} \to \mathbb{R}^{N_u}, \quad \tilde{\mathbf{u}}_N = \pi_N\left(\mathbf{y}_N, \boldsymbol{\mu}_s\right),$$

where $\tilde{\mathbf{u}}_N \in \mathbb{R}^{N_u}$ is the approximation of the control embedding $\mathbf{u}_N \in \mathbb{R}^{N_u}$ given by the surrogate policy π_N. Differently from the full-order policy function π_h in Equation (2), the input-output spaces are now reduced. Lighter neural network architectures are crucial to speed up both training and evaluations, especially when dealing with real-time applications involving high-dimensional variables.

Neural network training. The weights and biases of the aforementioned neural networks – namely, the autoencoders and the policy function – are trained all at once within a single optimization procedure. Doing so, an implicit link is established between the state and control reductions and the low-dimensional policy, obtaining latent coordinates that are meaningful in view of the policy-based control. In particular, the following cumulative loss function is taken into account

$$J_{\text{NN}} = \lambda_1 J_{\text{rec}}^y + \lambda_2 J_{\text{rec}}^u + J_{\pi_N}.$$

The reconstruction errors J_{rec}^y and J_{rec}^u are given by

$$J_{\text{rec}}^y = \begin{cases} 0 & \text{in case a POD-NN is used,} \\ J_{\text{AE}}^y & \text{in case a DL-ROM is used,} \\ J_{\text{POD+AE}}^y & \text{in case a POD-DL-ROM is used;} \end{cases} \qquad J_{\text{rec}}^u = \begin{cases} 0 & \text{in case a POD-NN is used,} \\ J_{\text{AE}}^u & \text{in case a DL-ROM is used,} \\ J_{\text{POD+AE}}^u & \text{in case a POD-DL-ROM is used,} \end{cases} \tag{4}$$

where the mean square reconstruction errors

$$\begin{aligned} J_{\text{AE}}^y &= \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbf{y}_h^{(k)} - \mathbf{y}_{h,\text{rec}}^{(k)} \right\|^2 = \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbf{y}_h^{(k)} - \varphi_D^y(\varphi_E^y(\mathbf{y}_h^{(k)})) \right\|^2 \\ J_{\text{AE}}^u &= \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbf{u}_h^{(k)} - \mathbf{u}_{h,\text{rec}}^{(k)} \right\|^2 = \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbf{u}_h^{(k)} - \varphi_D^u(\varphi_E^u(\mathbf{u}_h^{(k)})) \right\|^2 \\ J_{\text{POD+AE}}^y &= \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbb{V}_y^\top \mathbf{y}_h^{(k)} - \mathbb{V}_y^\top \mathbf{y}_{h,\text{rec}}^{(k)} \right\|^2 = \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbb{V}_y^\top \mathbf{y}_h^{(k)} - \varphi_D^y(\varphi_E^y(\mathbb{V}_y^\top \mathbf{y}_h^{(k)})) \right\|^2 \\ J_{\text{POD+AE}}^u &= \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbb{V}_u^\top \mathbf{u}_h^{(k)} - \mathbb{V}_u^\top \mathbf{u}_{h,\text{rec}}^{(k)} \right\|^2 = \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbb{V}_u^\top \mathbf{u}_h^{(k)} - \varphi_D^u(\varphi_E^u(\mathbb{V}_u^\top \mathbf{u}_h^{(k)})) \right\|^2 \end{aligned}$$

are computed on a set of training snapshots with indices $I_{\text{train}} \subset \{1, ..., N_t N_s\}$ with, typically, $|I_{\text{train}}| \approx 0.8 N_t N_s$. Moreover, the mean square prediction error entailed by the policy on training data is computed as

$$\begin{aligned} J_{\pi_N} &= \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \underbrace{\left\| \mathbf{u}_N^{(k)} - \tilde{\mathbf{u}}_N^{(k)} \right\|^2}_{\text{Latent space error}} + \lambda_3 \underbrace{\left\| \varphi_D^u(\mathbf{u}_N^{(k)}) - \varphi_D^u(\tilde{\mathbf{u}}_N^{(k)}) \right\|^2}_{\text{After-decoding error}} \\ &= \frac{1}{|I_{\text{train}}|} \sum_{k \in I_{\text{train}}} \left\| \mathbf{u}_N^{(k)} - \pi_N(\mathbf{y}_N^{(k)}, \boldsymbol{\mu}_s^{(k)}) \right\|^2 + \lambda_3 \left\| \varphi_D^u(\mathbf{u}_N^{(k)}) - \varphi_D^u(\pi_N(\mathbf{y}_N^{(k)}, \boldsymbol{\mu}_s^{(k)})) \right\|^2 . \end{aligned} \tag{5}$$

The after-decoding loss term in Equation (5) – which is considered whenever the control is reduced through AE or POD+AE – is helpful in training the control decoder in accordance with the surrogate policy, thus achieving more accurate decodings of the policy predictions and better results at the full-order levels. The hyperparameters $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$ must be chosen properly to balance the magnitudes of the different terms within J_{NN}. A summary of the offline phase is available in Figure 2 when considering the snapshots of the first test case detailed in Section 4 and when taking into account POD+AE as reduction strategy both for the state and the control.

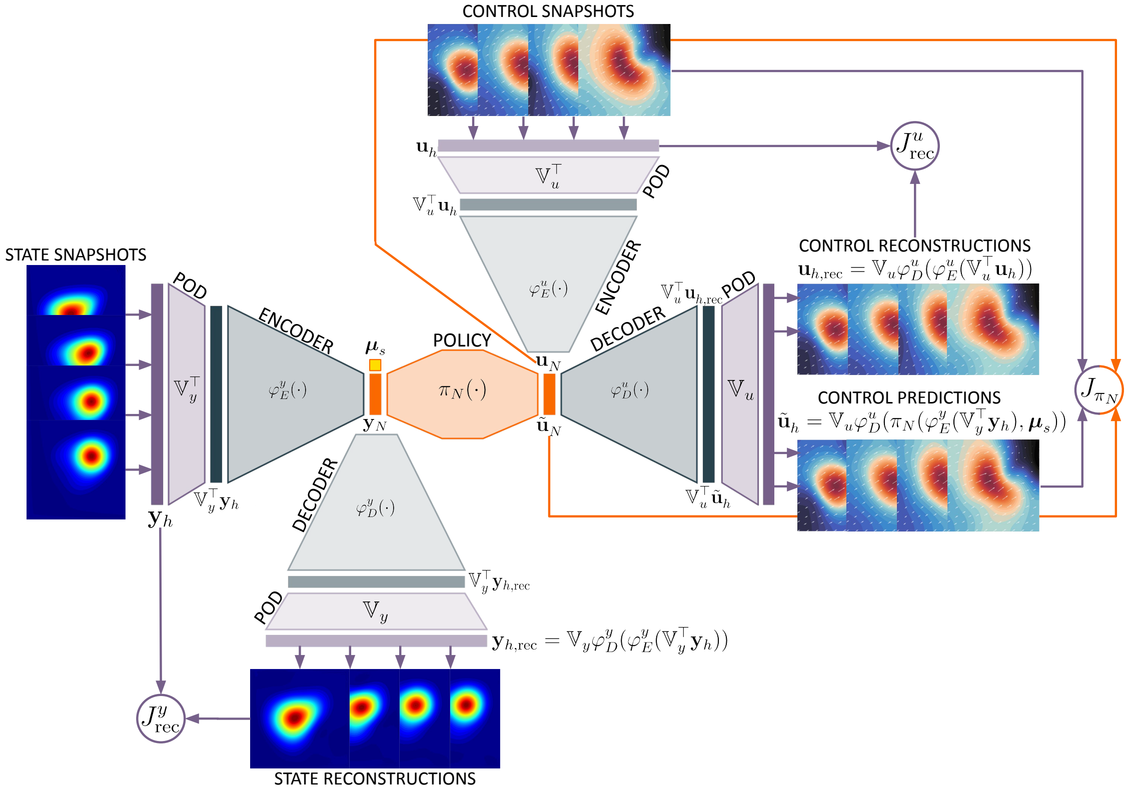
CONTROL SNAPSHOTS
$\mathbf{u}_h$
$\mathbb{V}_u^\top$
POD
$\mathbb{V}_u^\top \mathbf{u}_h$
J_{rec}^u
$\varphi_E^u(\cdot)$
ENCODER
CONTROL RECONSTRUCTIONS
$\mathbf{u}_{h,\text{rec}} = \mathbb{V}_u \varphi_D^u(\varphi_E^u(\mathbb{V}_u^\top \mathbf{u}_h))$
STATE SNAPSHOTS
POD
ENCODER
$\boldsymbol{\mu}_s$
POLICY
$\mathbb{V}_u^\top \mathbf{u}_{h,\text{rec}}$
POD
DECODER
$\mathbb{V}_y^\top$
$\varphi_E^y(\cdot)$
$\pi_N(\cdot)$
$\mathbf{u}_N$
$\varphi_D^u(\cdot)$
$\mathbb{V}_u$
$\mathbf{y}_N$
$\tilde{\mathbf{u}}_N$
CONTROL PREDICTIONS
$\tilde{\mathbf{u}}_h = \mathbb{V}_u \varphi_D^u(\pi_N(\varphi_E^y(\mathbb{V}_y^\top \mathbf{y}_h), \boldsymbol{\mu}_s))$
J_{π_N}
$\mathbb{V}_y^\top \mathbf{y}_h$
$\mathbf{y}_h$
$\mathbb{V}_u^\top \tilde{\mathbf{u}}_h$
DECODER
$\varphi_D^y(\cdot)$
$\mathbb{V}_y^\top \mathbf{y}_{h,\text{rec}}$
POD
$\mathbb{V}_y$
$\mathbf{y}_{h,\text{rec}} = \mathbb{V}_y \varphi_D^y(\varphi_E^y(\mathbb{V}_y^\top \mathbf{y}_h))$
J_{rec}^y
STATE RECONSTRUCTIONS

Figure 2. Offline phase of the deep learning-based reduced order feedback controller. After generating optimal state and control snapshots through the adjoint method, the state and control autoencoders, namely $\varphi_D^y(\varphi_E^y(\cdot))$ and $\varphi_D^u(\varphi_E^u(\cdot))$, and the surrogate policy π_N are trained minimizing the cumulative loss function $J_{\text{NN}} = \lambda_1 J_{\text{rec}}^y + \lambda_2 J_{\text{rec}}^u + J_{\pi_N}$.

3.2. Online phase

Whenever a new state $\mathbf{y}_h^{\text{new}}$ is observed in a scenario $\boldsymbol{\mu}_s^{\text{new}}$ unseen during training, the corresponding optimal control action is inferred through a forward pass of the low-dimensional policy and the encoding-decoding maps, that is

$$\tilde{\mathbf{u}}_h^{\text{new}} = \phi_D^u(\tilde{\mathbf{u}}_N^{\text{new}}) = \phi_D^u(\pi_N(\phi_E^y(\mathbf{y}_h^{\text{new}}), \boldsymbol{\mu}_s^{\text{new}}))$$

where

$$
\begin{aligned}
\phi_E^y(\mathbf{y}_h^{\text{new}}) &= \begin{cases} \mathbb{V}_y^\top \mathbf{y}_h^{\text{new}} & \text{in case a POD-NN is used,} \\ \varphi_E^y(\mathbf{y}_h^{\text{new}}) & \text{in case a DL-ROM is used,} \\ \varphi_E^y(\mathbb{V}_y^\top \mathbf{y}_h^{\text{new}}) & \text{in case a POD-DL-ROM is used;} \end{cases} \\
\phi_D^u(\tilde{\mathbf{u}}_N^{\text{new}}) &= \begin{cases} \mathbb{V}_u \tilde{\mathbf{u}}_N^{\text{new}} & \text{in case a POD-NN is used,} \\ \varphi_D^u(\tilde{\mathbf{u}}_N^{\text{new}}) & \text{in case a DL-ROM is used,} \\ \mathbb{V}_u \varphi_D^u(\tilde{\mathbf{u}}_N^{\text{new}}) & \text{in case a POD-DL-ROM is used.} \end{cases}
\end{aligned}
\tag{6}
$$

In practice, starting from the observed high-dimensional state directly measured from the dynamical system at hand, we predict the distributed control responsible for steering the system towards the optimal configuration. To this aim, state reduction and control decoding are crucial to move back and forth from the full-order dimensions to the latent level where the surrogate policy is employed. New control predictions can be then retrieved in loop whenever new state measurements are available. A sketch of the proposed feedback loop is depicted in Figure 3 when considering the snapshots of the first test case detailed in Section 4 and when taking into account POD+AE as reduction strategy both for the state and the control. Differently from intrusive techniques – such as, e.g., the Reduced Basis method [34, 59] where the parameter-to-solution map is retrieved by properly projecting and solving the system of optimality conditions in Equation (3) – the proposed deep-learning based reduced order modeling strategy is non-intrusive, resulting in a very general and flexible tool capable of retrieving the optimal control action in real-time for a wide range of control problems.

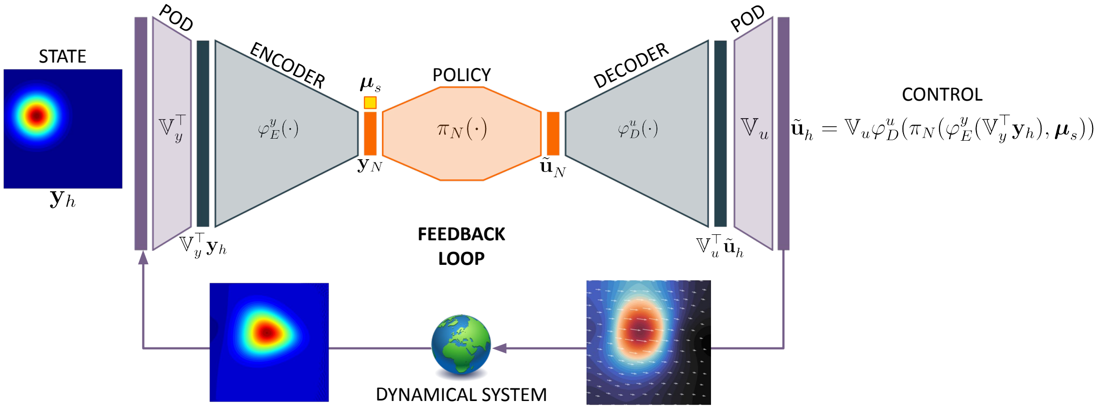

Figure 3. Online phase of the deep learning-based reduced order feedback controller. The optimal full-order control action corresponding to the observed state $\mathbf{y}_h$ in a scenario described by input parameters $\boldsymbol{\mu}_s$ is inferred online through forward passes of π_N and the encoding-decoding mappings.

3.3. Latent feedback loop

The deep-learning based feedback control strategy introduced so far relies on the real-time availability of full-order state information in the online phase. Indeed, when dealing with feedback controllers, the optimal control action is inferred by looking at quantities observed from the dynamical system at hand. Specifically, the input of the surrogate policy π_N is the current full-order state variable in its reduced form, which provides a comprehensive overview of the configuration and evolution of the system. However, continuous monitoring of the dynamical system may be unfeasible, especially when dealing with high-dimensional data. Indeed, the computational burden to generate synthetic state data online through high-fidelity FOM simulations – as taken into account throughout the test cases in Section 4 – may not meet strict timing requirements, thereby compromising the feedback controller performances. Instead, whenever the state snapshot is measured through sensors widespread in the domain, continuous monitoring may be compromised by, e.g., sensor malfunctions, delays in data processing or temporary absence of signal for communications. To overcome these limitations, here we propose a feedback loop closure at the latent level that allows the model to rapidly predict the control action even in absence of state measurements online. Specifically, we aim to train a neural network to approximate the forward model (i.e. the time advancing scheme of the dynamical system) at the latent level, that is

$$\varphi_N : \mathbb{R}^{N_y} \times \mathbb{R}^{N_u} \times \mathcal{P} \to \mathbb{R}^{N_y}, \quad \tilde{\mathbf{y}}_N(t_j) = \varphi_N\left(\mathbf{y}_N(t_{j-1}), \mathbf{u}_N(t_{j-1}), \boldsymbol{\mu}_s\right) \qquad \forall j = 1, ..., N_t.$$

As previously discussed, to preserve a better consistency between the dimensionality reduction and the maps defined in the latent spaces, we employ a single optimization step to train all the neural networks into play, that are the autoencoders, the policy π_N and the forward map φ_N. Specifically, the cumulative loss function now becomes

$$J_{\text{NN}} = \lambda_1 J^y_{\text{rec}} + \lambda_2 J^u_{\text{rec}} + J_{\pi_N} + J_{\varphi_N}$$

where

$$J_{\varphi_N} = \frac{1}{|I'_{\text{train}}|} \sum_{(i,j)\in I'_{\text{train}}} \lambda_4 \underbrace{\left\| \mathbf{y}^{(i)}_N(t_j) - \tilde{\mathbf{y}}^{(i)}_N(t_j) \right\|^2}_{\text{Prediction-from-data error}} + \lambda_5 \underbrace{\left\| \mathbf{y}^{(i)}_N(t_j) - \hat{\mathbf{y}}^{(i)}_n(t_j) \right\|^2}_{\text{Prediction-from-policy error}} + \lambda_6 \underbrace{\left\| \varphi^y_D(\mathbf{y}^{(i)}_N(t_j)) - \varphi^y_D(\tilde{\mathbf{y}}^{(i)}_N(t_j)) \right\|^2}_{\text{After-decoding error}} \tag{7}$$

while $J^y_{\text{rec}}, J^u_{\text{rec}}$ and J_{π_N} are defined in Equation (4) and Equation (5), respectively. The state predictions appearing in Equation (7) are the approximations of future state values starting from, respectively, the available training control data and policy-based control predictions, that are

$$\tilde{\mathbf{y}}^{(i)}_N(t_j) = \varphi_N(\mathbf{y}^{(i)}_N(t_{j-1}), \mathbf{u}^{(i)}_N(t_{j-1}), \boldsymbol{\mu}^{(i)}_s), \qquad \hat{\mathbf{y}}^{(i)}_N(t_j) = \varphi_N(\mathbf{y}^{(i)}_N(t_{j-1}), \pi_N(\mathbf{y}^{(i)}_N(t_{j-1}), \boldsymbol{\mu}^{(i)}_s), \boldsymbol{\mu}^{(i)}_s) \qquad \forall j = 1, ..., N_t,$$

while $I'_{\text{train}} \subset \{1, ..., N_s\} \times \{1, ..., N_t\}$ denotes the set of training indices. The prediction-from-data and prediction-from-policy error terms in Equation (7) are useful to achieve accurate forward-in-time states at the latent level, both starting from control data and policy outputs. Instead, the after-decoding error term is helpful in obtaining acceptable results also at higher dimensions whenever AE or POD+AE reduction strategies are exploited. Here, three additional hyperparameters $\lambda_4, \lambda_5, \lambda_6 \in \mathbb{R}$ are considered to balance the terms in the cumulative loss function.

The proposed feedback latent loop is depicted in Figure 4 when considering the snapshots of the first test case detailed in Section 4 and when taking into account POD+AE as reduction strategy both for the state and the control. Whenever full-order state data are available online, either in the form of sensor measurements or synthetic data simulated via a high-fidelity FOM, these may be exploited to infer the related optimal control action to be applied on the dynamical system, that is

$$\tilde{\mathbf{u}}_h(t_j) = \phi^u_D(\pi_N(\mathbf{y}_N(t_j), \boldsymbol{\mu}_s)) \quad \text{where} \quad \mathbf{y}_N(t_j) = \phi^y_E(\mathbf{y}_h(t_j)) \qquad \forall j = 0, ..., N_t - 1$$

with ϕ^u_D and ϕ^y_E defined in Equation (6). Whether the full-order state snapshot is missing due to, e.g., latency concerns or sensors failures, state predictions at the latent level are provided by a forward pass of φ_N, and optimal control can still be predicted up to an approximation error through

$$\tilde{\mathbf{u}}_h(t_j) = \phi^u_D(\pi_N(\tilde{\mathbf{y}}_N(t_j), \boldsymbol{\mu}_s)) \quad \text{where} \quad \tilde{\mathbf{y}}_N(t_j) = \varphi_N(\mathbf{y}_N(t_{j-1}), \tilde{\mathbf{u}}_N(t_{j-1}), \boldsymbol{\mu}_s) \qquad \forall j = 1, ..., N_t - 1,$$

resulting in a continuous control of the dynamical system.

4. Numerical results

This section details the numerical tests performed to assess the effectiveness of the proposed approach. Specifically, we focus on two different time-dependent optimal transport problems in two spatial dimensions where, by optimally tuning the velocity field responsible for the state movement over space and time, the state has to be steered from its starting configuration to the desired final destination. In this context, our aim is to synthesize a feedback controller capable of dealing with both different initial settings and different target locations to be chosen online. To do so, we explore the initial state variability, as well as different target locations, in the data generation procedure. To better visualize the problem, note that the state variable may represent the density of particles systems described through a mean-field model, such as

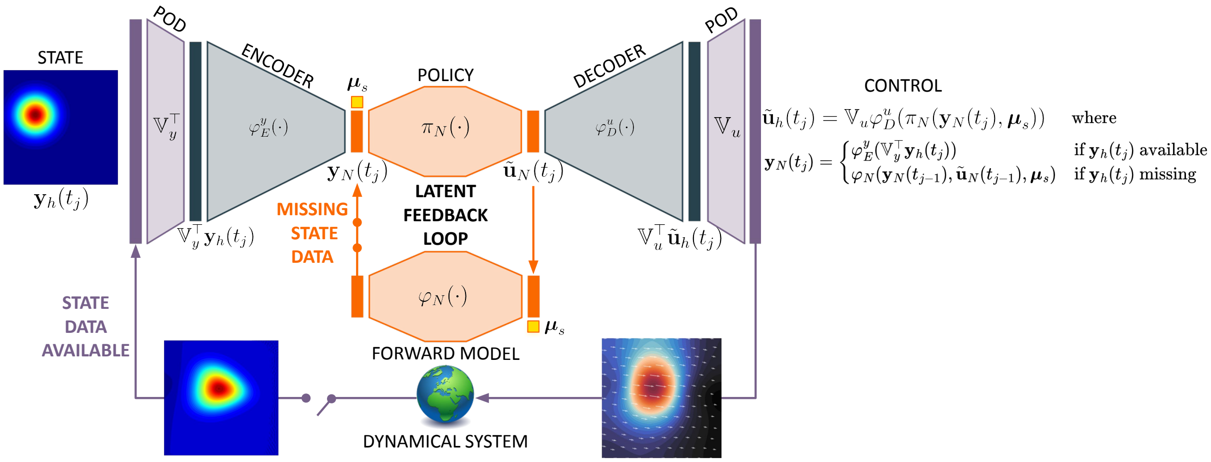
STATE
$\mathbf{y}_h(t_j)$
POD
$\mathbb{V}_y^\top$
$\mathbb{V}_y^\top \mathbf{y}_h(t_j)$
ENCODER
$\varphi_E^y(\cdot)$
$\boldsymbol{\mu}_s$
$\mathbf{y}_N(t_j)$
POLICY
$\pi_N(\cdot)$
$\tilde{\mathbf{u}}_N(t_j)$
LATENT FEEDBACK LOOP
MISSING STATE DATA
$\varphi_N(\cdot)$
$\boldsymbol{\mu}_s$
FORWARD MODEL
DECODER
$\varphi_D^u(\cdot)$
POD
$\mathbb{V}_u$
$\mathbb{V}_u^\top \tilde{\mathbf{u}}_h(t_j)$
CONTROL
$\tilde{\mathbf{u}}_h(t_j) = \mathbb{V}_u \varphi_D^u(\pi_N(\mathbf{y}_N(t_j), \boldsymbol{\mu}_s))$ where
$\mathbf{y}_N(t_j) = \begin{cases} \varphi_E^y(\mathbb{V}_y^\top \mathbf{y}_h(t_j)) & \text{if } \mathbf{y}_h(t_j) \text{ available} \\ \varphi_N(\mathbf{y}_N(t_{j-1}), \tilde{\mathbf{u}}_N(t_{j-1}), \boldsymbol{\mu}_s) & \text{if } \mathbf{y}_h(t_j) \text{ missing} \end{cases}$
STATE DATA AVAILABLE
DYNAMICAL SYSTEM

Figure 4. Online phase of the deep learning-based reduced order feedback controller with latent feedback loop. The optimal full-order control action corresponding to the observed state $\mathbf{y}_h$ in a scenario described by input parameters $\boldsymbol{\mu}_s$ is inferred online through forward-passes of π_N and the encoding-decoding mappings. Whenever full-order state data $\mathbf{y}_h$ are not available online, the trained deep learning-based forward model φ_N is exploited to predict the state evolution, allowing for a continuous prediction of the control action.

swarms of autonomous robots [67, 68, 66] delivering goods to the target location, while avoiding collisions with boundaries and obstacles along the route.

Throughout this section, the following mean relative errors are employed to fairly evaluate the prediction accuracy of the proposed approach:

$$\varepsilon_{\text{rel}}^y = \frac{1}{|I_{\text{test}}|} \sum_{k \in I_{\text{test}}} \frac{\|\mathbf{y}_h^{(k)} - \tilde{\mathbf{y}}_h^{(k)}\|}{\|\mathbf{y}_h^{(k)}\|}, \qquad \varepsilon_{\text{rel}}^u = \frac{1}{|I_{\text{test}}|} \sum_{k \in I_{\text{test}}} \frac{\|\mathbf{u}_h^{(k)} - \tilde{\mathbf{u}}_h^{(k)}\|}{\|\mathbf{u}_h^{(k)}\|}$$

where $I_{\text{test}} = \{1, ..., N_t N_s\} \setminus I_{\text{train}}$ is the set containing the indices of test data, that are the snapshots exploited only for evaluation purposes. Note that, when evaluating the reconstruction capabilities of the chosen reduction method, $\mathbf{y}_{h,\text{rec}}^{(k)}$ and $\mathbf{u}_{h,\text{rec}}^{(k)}$ are taken into account in place of $\tilde{\mathbf{y}}_h^{(k)}$ and $\tilde{\mathbf{u}}_h^{(k)}$, respectively.

4.1. Optimal transport in a vacuum

This section is devoted to the application of the proposed real-time feedback controller to an optimal transport problem in a vacuum – that is, for the sake of simplicity, we neglect the effects of surrounding fluid, such as air or water, in the space-time domain $\Omega \times (0, T]$, with final time $T > 0$. The state dynamics is described by the Fokker-Planck equation (also known as Kolmogorov forward equation)

$$\begin{cases} \dfrac{\partial y}{\partial t} + \nabla \cdot (-\nu \nabla y + \mathbf{u} y) = 0 & \text{in } \Omega \times (0, T] \\ (-\nu \nabla y + \mathbf{u} y) \cdot \mathbf{n} = 0 & \text{on } \partial\Omega \times (0, T] \\ y(0) = y_0(\mu_1^0, \mu_2^0) & \text{in } \Omega \times \{t = 0\} \end{cases} \tag{8}$$

where $\Omega = (-1, 1)^2$ is the 2D domain with space coordinates denoted by x_1 and x_2, ν is the diffusion coefficient – here set equal to 0.001 in order to focus mainly on the transport effect – and $\mathbf{n}$ is the outward normal versor to the boundary $\partial\Omega$. Starting from a Gaussian density with variance equal to 0.05 and centered at (μ_1^0, μ_2^0), that is

$$y_0(\mu_1^0, \mu_2^0) = \frac{10}{\pi} \exp\left(-10(x_1 - \mu_1^0)^2 - 10(x_2 - \mu_2^0)^2\right),$$

we aim to steer the state $y : \Omega \times [0, T] \to \mathbb{R}$ toward a final target destination exploiting the velocity field $\mathbf{u} : \Omega \times [0, T] \to \mathbb{R}^2$ as control action. Note that the parametrization of the initial state $y_0 = y_0(\mu_1^0, \mu_2^0)$ is crucial to reduce the state variability while exploring the state space in the data generation. By doing so, we focus on a specific but meaningful set of starting configurations, which correspond to steady particles systems at rest awaiting instructions.

Remark. Let $M(t) = \int_\Omega y(t)d\Omega$ be the total mass obtained by integrating the state variable in the domain of interest. As shown by, e.g., [67], Equation (8) entails mass conservation, as required when considering robotic swarms moving in $\Omega \times [0, T]$. Indeed, thanks to the divergence theorem and to the boundary condition selected, we obtain

$$\frac{d}{dt}M(t) = \int_\Omega \frac{\partial y(t)}{\partial t} d\Omega = \int_\Omega -\nabla \cdot (-\nu \nabla y + \mathbf{u}y) d\Omega = \int_{\partial\Omega} -(-\nu\nabla y + \mathbf{u}y) \cdot \mathbf{n} d\Gamma = 0$$

The domain Ω – which does not show inner obstacles, for the sake of simplicity – is discretized through `gmsh` utilities [29] yielding a conformal mesh with triangular elements and 7569 nodes. The semi-discrete formulation of Equation (8) is derived by FEM, taking into account continuous and piecewise linear finite element basis functions both for the state and the control, ending up with remarkably high number of degrees of freedom $N_h^y = 7569$ and $N_h^u = 15138$. To solve the high-fidelity FOM, time discretization is made over a uniform grid spanning $[0, T]$ with time step $\Delta t = 0.25$, where the final time T is set equal to 1.

The optimal transport field capable of steering the state towards a target location in Ω can be found through an optimization procedure. In particular, among all the physically-admissible state-control pairs satisfying Equation (8), we aim to find the minimizer of the cost functional

$$J(y, \mathbf{u}; \boldsymbol{\mu}_s) = \frac{1}{2}\int_0^T \int_\Omega (y - y_d)^2 d\Omega dt + \int_0^T \int_{\partial\Omega} y^2 d\Gamma dt + \frac{\beta}{2}\int_0^T \int_\Omega ||\mathbf{u}||^2 d\Omega dt + \frac{\beta_g}{2}\int_0^T \int_\Omega ||\nabla \mathbf{u}||^2 d\Omega dt \quad (9)$$

measuring the discrepancy between the current observed state and the target state $y_d(\mu_1^d, \mu_2^d)$, which is the Gaussian density

$$y_d(\mu_1^d, \mu_2^d) = \frac{10}{\pi} \exp\left(-10(x_1 - \mu_1^d)^2 - 10(x_2 - \mu_2^d)^2\right)$$

centered at (μ_1^d, μ_2^d) and showing the same covariance as the initial configuration y_0. Since we aim to rapidly control the state dynamics considering different destinations online, the mean coordinates of the target location are regarded as scenario parameters, that is $\boldsymbol{\mu}_s = (\mu_1^d, \mu_2^d)$. Instead, since the state information is directly captured within the feedback loop, the mean coordinates of the initial state (μ_1^0, μ_2^0) are not regarded as scenario parameters, so that their knowledge is not required in the online phase. The boundary integral in Equation (9) is useful to avoid collisions with the domain boundary, as necessary when dealing with particles systems, while the regularizing terms concerning the norms of the control and its gradient penalize overconsuming and irregular control actions, which may be unfeasible in practice. To properly balance the magnitudes of the different terms appearing in the loss functional, we set $\beta = \beta_g = 0.2$. Note that the discrete cost functional $J_h = J_h(\mathbf{y}_h, \mathbf{u}_h; \boldsymbol{\mu}_s)$ introduced in Equation (1) can be easily recovered starting from Equation (9) thanks to discretization techniques such as FEM. To further visualize the problem setting, Figure 5 shows an example of optimal state trajectory related to $(\mu_1^0, \mu_2^0) = (-0.45, 0.21)$ and $\boldsymbol{\mu}_s = (0.29, -0.24)$, along with the corresponding optimal control actions applied. Specifically, it is possible to assess that the optimal transport steers the state density towards the target destination over time, that is

$$||\mathbf{y}_h(t) - \mathbf{y}_d|| \xrightarrow[t \to T]{} 0$$

As described in Section 3, the starting point in building the deep learning-based reduced order feedback controller is data generation. In particular, we consider 100 random scenarios sampled in the parameter space $\mathcal{P} = (0.0, 0.5) \times (-0.5, 0.5)$, i.e. we take into account different endpoints placed on the right-hand side of the domain, avoiding positions too close to the boundaries. Moreover, to account for the initial state variability, we sample the starting position coordinates on the left-hand side of Ω, that is $(\mu_1^0, \mu_2^0) \in (-0.5, 0.0) \times (-0.5, 0.5)$. For every combination of starting-final state positions, we thus compute the optimal state and control trajectory at $N_t = \frac{T}{\Delta t} = 4$ time instants through `dolfin-adjoint` [52], which provides an OCP solver in `fenics` [6] exploiting FEM and the adjoint method, with an average computational time equal to 15 minutes per scenario. In particular, we select L-BFGS-B as optimization algorithm, while considering a tolerance and maximum number of iterations equal to, respectively, 10^{-6} and 500. Note that, leveraging the horizontal symmetry of the problem, it is possible to double the dataset cardinality for free, ending up with a total of $N_s = 200$ scenarios and $N_s N_t = 800$ state-scenario-control triplets investigated. Out of all the available trajectories, training and test sets are obtained through a $80 : 20$ split. In particular, while

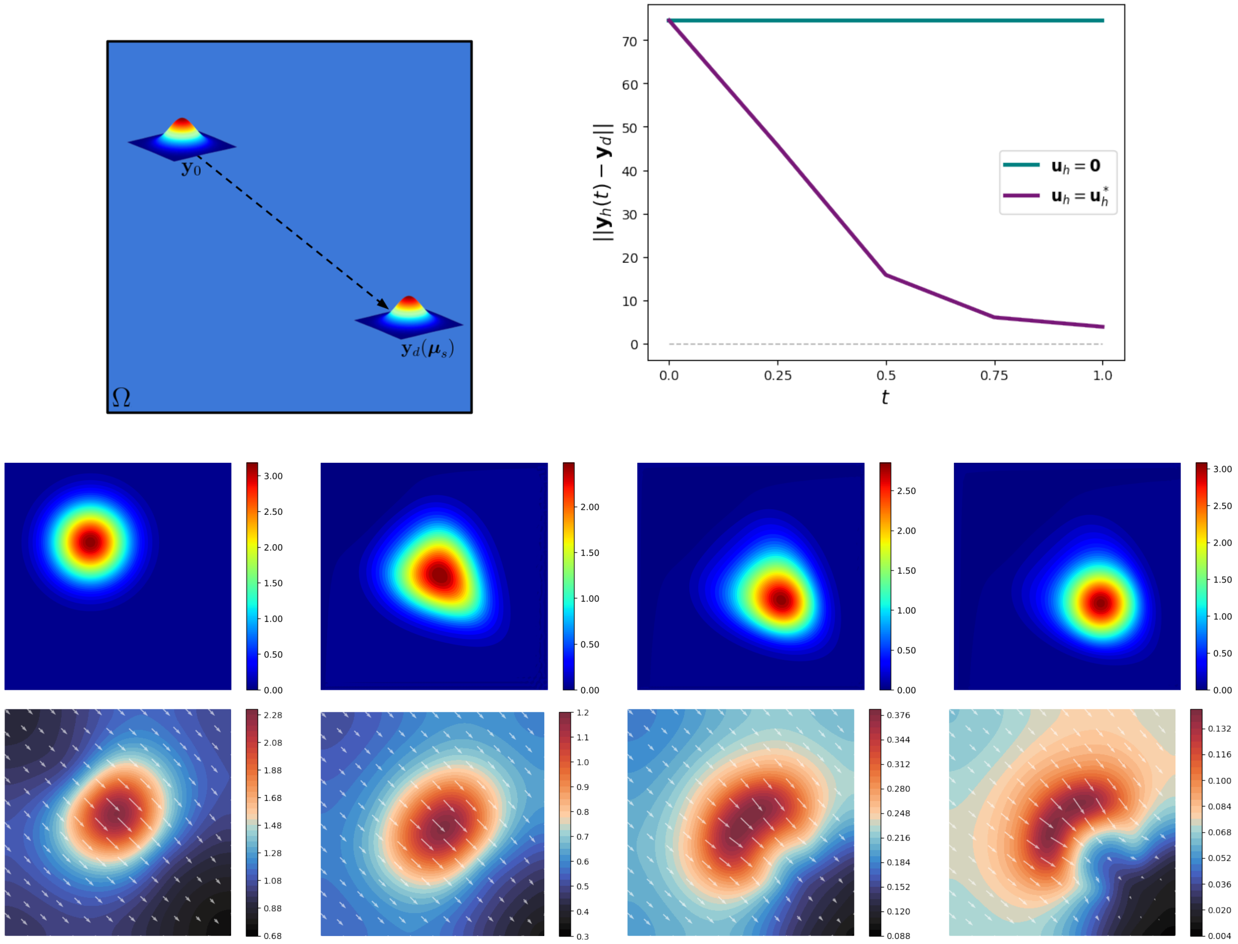

Figure 5. *Test 1.1.* Optimal transport in a vacuum. Top left: representation of an optimal state trajectory in a vacuum within the domain Ω, where $\mathbf{y}_0$ stands for the initial density centered at $(\mu_1^0, \mu_2^0) = (-0.45, 0.21)$, while $\mathbf{y}_d(\boldsymbol{\mu}_s)$ represents the target configuration centered at $(\mu_1^d, \mu_2^d) = (0.29, -0.24)$. Top right: discrepancy between current state $\mathbf{y}_h(t)$ and target configuration $\mathbf{y}_d$ centered at $(\mu_1^d, \mu_2^d) = (0.29, -0.24)$ at different time instants in the uncontrolled ($\mathbf{u}_h = \mathbf{0}$) and optimal ($\mathbf{u}_h = \mathbf{u}_h^*$) settings. Other panels: space-varying optimal state and control at $t = 0, 0.25, 0.5, 0.75$ related to the scenario parameters $\boldsymbol{\mu}_s = (0.29, -0.24)$. The control velocity fields on Ω are depicted through vector fields, with the underlying colours corresponding to their magnitude.

training data are exploited for reduction and neural networks training purposes, test data are used only to evaluate the generalization capabilities of our controller.

The second necessary step to construct our feedback controller is dimensionality reduction. First of all, we apply a linear projection through POD, looking at the singular values decays and the reconstruction errors in order to select the number of modes to retain. In particular, by selecting 150 and 160 modes for state and control, respectively, we end up with $\varepsilon_{\text{rel}}^y = 0.21\%$ and $\varepsilon_{\text{rel}}^u = 0.36\%$. Note that, being the control variable a vector field, POD is applied component-wise, thus taking into account 80 POD modes for each spatial component. Figure 6 displays the singular values decays and the two most energetic POD modes related to state and control reductions. In particular, due to the significant transport effect, the singular values show slow polynomial decays, thus requiring a remarkably high number of POD modes in order to correctly reconstruct the state and control fields.

The state and control POD coefficients can be further compressed through an autoencoder-based projection onto nonlinear subspaces, resulting in the POD+AE reduction strategy introduced in Section 3. As far as the state autoencoder architecture is concerned, the latent dimension is set equal to $N_y = 10$, while the encoder and decoder consist, respectively, of 1 and 2 hidden layers with 100 neurons each and with leaky Relu as activation function. A similar structure is taken into account for the control reduction, where the

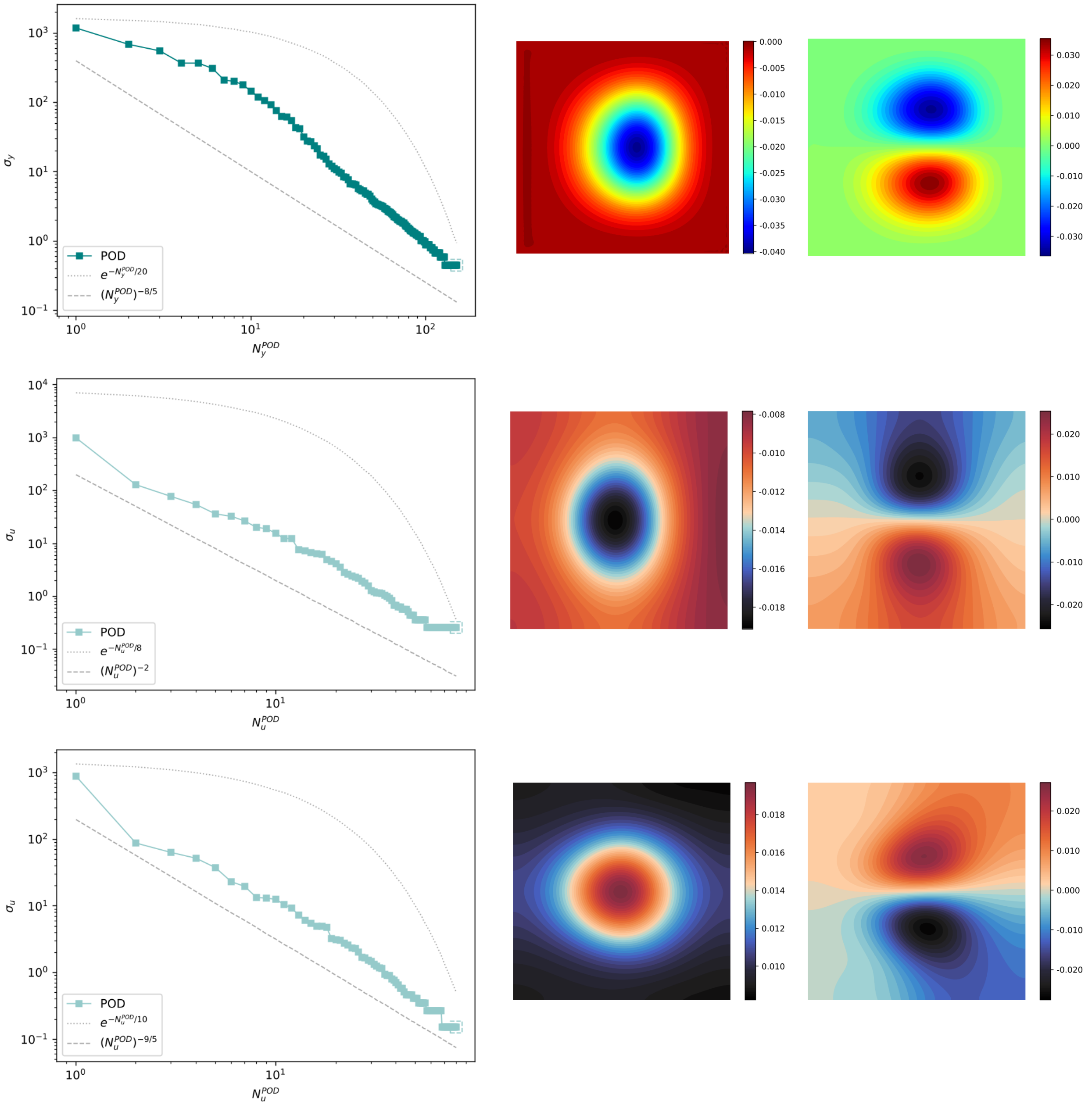

Figure 6. *Test 1.1.* Optimal transport in a vacuum. Singular values decay in log-log scale along with the two most energetic POD modes related to the state (top), x_1 component (center) and x_2 component (bottom) of the control.

latent dimension is increased to $N_u = 18$ and the number of neurons per layer in the decoder is doubled. In order to predict the optimal control action starting from an observed state configuration, a policy surrogate model π_N is required. In particular, to this aim, we take into account a deep feedforward neural network showing 3 hidden layers with 50 neurons each and leaky Relu as activation function. Note that no activation functions are considered in the output layer of the networks to avoid restrictions on output values. After initializing the weights through the strategy proposed by [32], we train the networks in 1 hour and 11 minutes minimizing the cumulative loss function J_{NN} introduced in Section 3 through the L-BFGS optimization algorithm, while considering $\lambda_1 = \lambda_2 = \lambda_3 = 0.01$. The POD+AE reconstruction errors entailed by the use of the state and control autoencoders end up being, respectively, $\varepsilon_{\text{rel}}^y = 3.20\%$ and $\varepsilon_{\text{rel}}^u = 5.04\%$. Instead, the relative prediction error of the policy surrogate model on the test data at the latent level – that is, the discrepancy between $\mathbf{u}_N$ and $\tilde{\mathbf{u}}_N = \pi_N(\mathbf{y}_N, \boldsymbol{\mu}_s)$ – is equal to 4.28%, while it increases to 7.09% after decoding through POD+AE – that is, the error between $\mathbf{u}_h$ and $\tilde{\mathbf{u}}_h = \mathbb{V}_u \varphi_D^u(\tilde{\mathbf{u}}_N)$. Figure 7 displays a control test trajectory reconstructed by POD+AE (second row), that is

$$\mathbf{u}_{h,\text{rec}} = \mathbb{V}_u \varphi_D^u(\varphi_E^u(\mathbb{V}_u^\top \mathbf{u}_h))$$

and the corresponding reconstruction errors (fourth row). In particular, we highlight that the latent coordinates, whose dimensionality is 841 times smaller than N_h^u, perfectly capture the full-order features of the control velocity. By a visual inspection of the third row of Figure 7, along with the reported errors in the last row of the same figure, it is also possible to assess the high accuracy of the optimal control predictions

$$\tilde{\mathbf{u}}_h = \mathbb{V}_u \varphi_D^u(\pi_N(\mathbf{y}_N, \boldsymbol{\mu}_s))$$

provided by the low-dimensional policy.

In case continuous monitoring of the state density is unfeasible or the computational burden to simulate synthetic data does not meet strict timing requirements, we can exploit the latent feedback loop introduced in Section 3.3. In particular, the surrogate model φ_N for the dynamics at the latent level is retrieved by a deep feedforward neural network that has 3 hidden layers with 50 neurons each and takes into account leaky Relu as activation function. We train the neural networks – that are the state and control autoencoders, the policy π_N and the forward model φ_N – in 1 hour and 20 minutes by minimizing the cumulative loss function introduced in Section 3.3 with $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_6 = 0.001$ and $\lambda_4 = \lambda_5 = 1$, exploiting L-BFGS as optimization algorithm. Note that, while φ_N is initialized through the strategy proposed in [32], we exploit the previously trained networks as initializations for the autoencoders and the policy. The POD+AE reconstruction errors committed by the state and control autoencoders are now equal to $\varepsilon_{\text{rel}}^y = 3.96\%$ and $\varepsilon_{\text{rel}}^u = 4.61\%$, while the policy prediction error after-decoding is equal to 6.98%. Instead, the prediction-from-data and the prediction-from-policy relative errors entailed by φ_N on test data are equal to, respectively, 2.09% and 1.37% at the latent level, while they increase to 7.49% and 5.45% after POD+AE decoding. Figure 8 shows a state test trajectory reconstructed by POD+AE (second row), that is

$$\mathbf{y}_{h,\text{rec}} = \mathbb{V}_y \varphi_D^y(\varphi_E^y(\mathbb{V}_y^\top \mathbf{y}_h))$$

and the corresponding reconstruction errors (fourth row). In particular, despite a dimensionality reduction of 757 times with respect to N_h^y, the low-dimensional features are able to reconstruct accurately the full-order snapshots. Moreover, Figure 8 displays the forward model predictions (third row)

$$\tilde{\mathbf{y}}_h(t_j) = \mathbb{V}_y \varphi_D^y(\varphi_N(\mathbf{y}_N(t_{j-1}), \mathbf{u}_N(t_{j-1}), \boldsymbol{\mu}_s)) \quad \forall j = 1, ..., N_t$$

and the corresponding prediction errors (last row) related to the same test scenario.

After data generation, dimensionality reduction and neural networks training in the offline phase, we can now control the dynamical system taking into account new initial configurations and scenarios, which have not been seen during training. Figure 9 displays the state evolution and the corresponding control actions related to a test case with initial state position $(\mu_1^0, \mu_2^0) = (-0.24, -0.14)$ and scenario parameters $\boldsymbol{\mu}_s = (0.48, -0.03)$. Specifically, both the feedback loops at the full-order and latent levels effectively steer the state density towards the desired target, with a decreasing discrepancy $||\mathbf{y}_h(t) - \mathbf{y}_d||$ over time. As far as computational times are concerned, the proposed controller with model closure at full-order level requires 0.81 seconds to provide control actions and simulate states for all the N_t time steps in the test setting considered, while it reduces to 0.028 seconds in the case of latent feedback loop, with a remarkably high speed-up with respect to full-order methods (1000× for the loop closure at the full-order level, 32000× for the latent feedback loop). Note that both proposed control strategies remain effective even when considering different (sufficiently small) time steps Δt and different final times T, thus allowing control actions to be applied to the system at different instants (according to the delay in the state computation) and an arbitrary number of times.

4.2. Optimal transport in a fluid

In this application, we focus on a more challenging optimal transport problem where *(i)* a rounded obstacle is added in the middle of the square $(-1, 1)^2$ – that is, the domain now becomes $\Omega = (-1, 1)^2 \setminus \mathcal{B}_{0.15}(0, 0)$

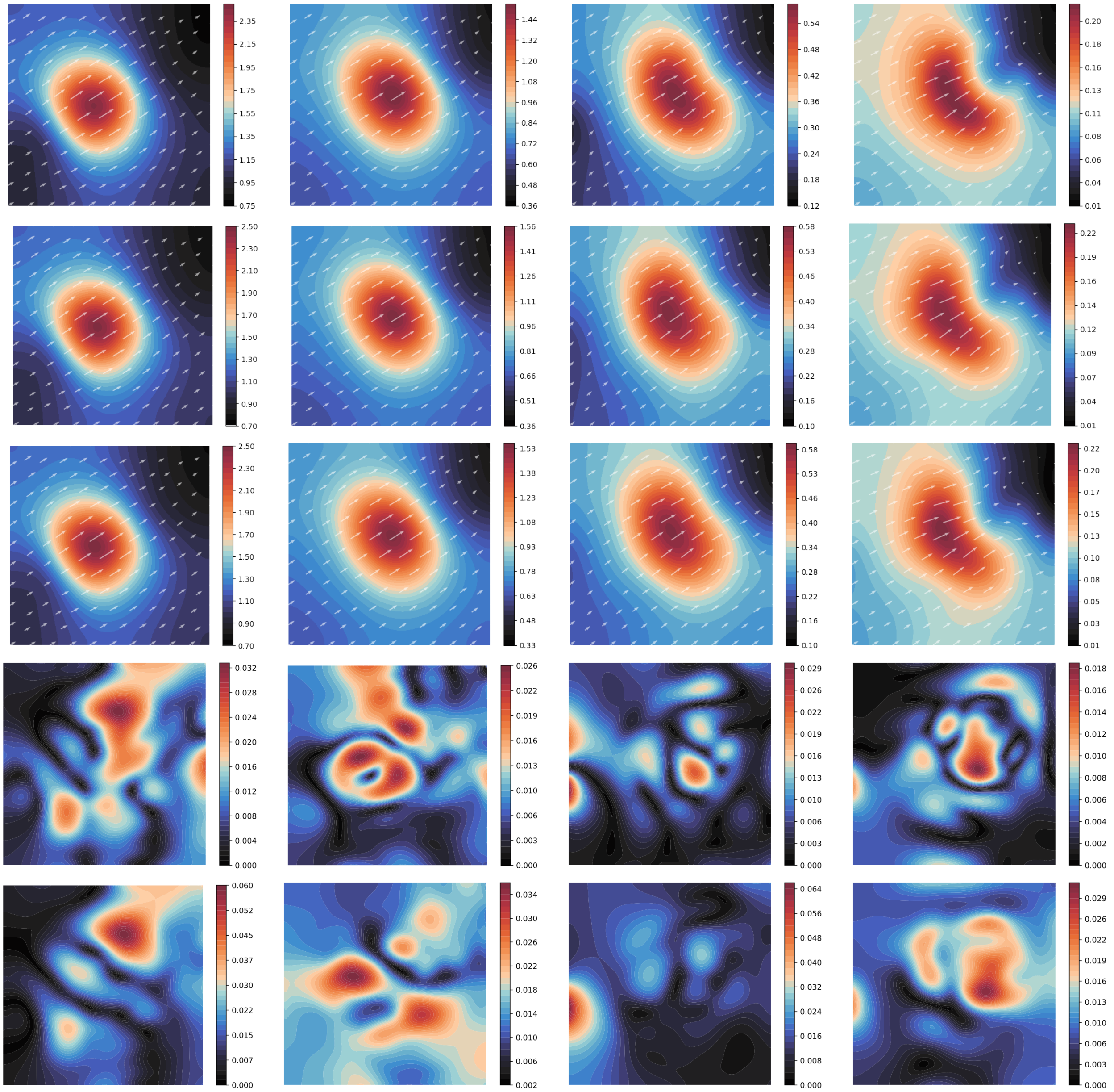

Figure 7. *Test 1.1.* Optimal transport in a vacuum. High-fidelity optimal control trajectory (first row), POD+AE reconstructions (second row), policy predictions (third row), POD+AE reconstruction errors (fourth row) and policy prediction errors (fifth row) at $t = 0, 0.25, 0.5, 0.75$ related to the test scenario parameters $\boldsymbol{\mu}_s = (0.31, 0.27)$. The control velocity fields on Ω are depicted through vector fields, with the underlying colours corresponding to their magnitude.

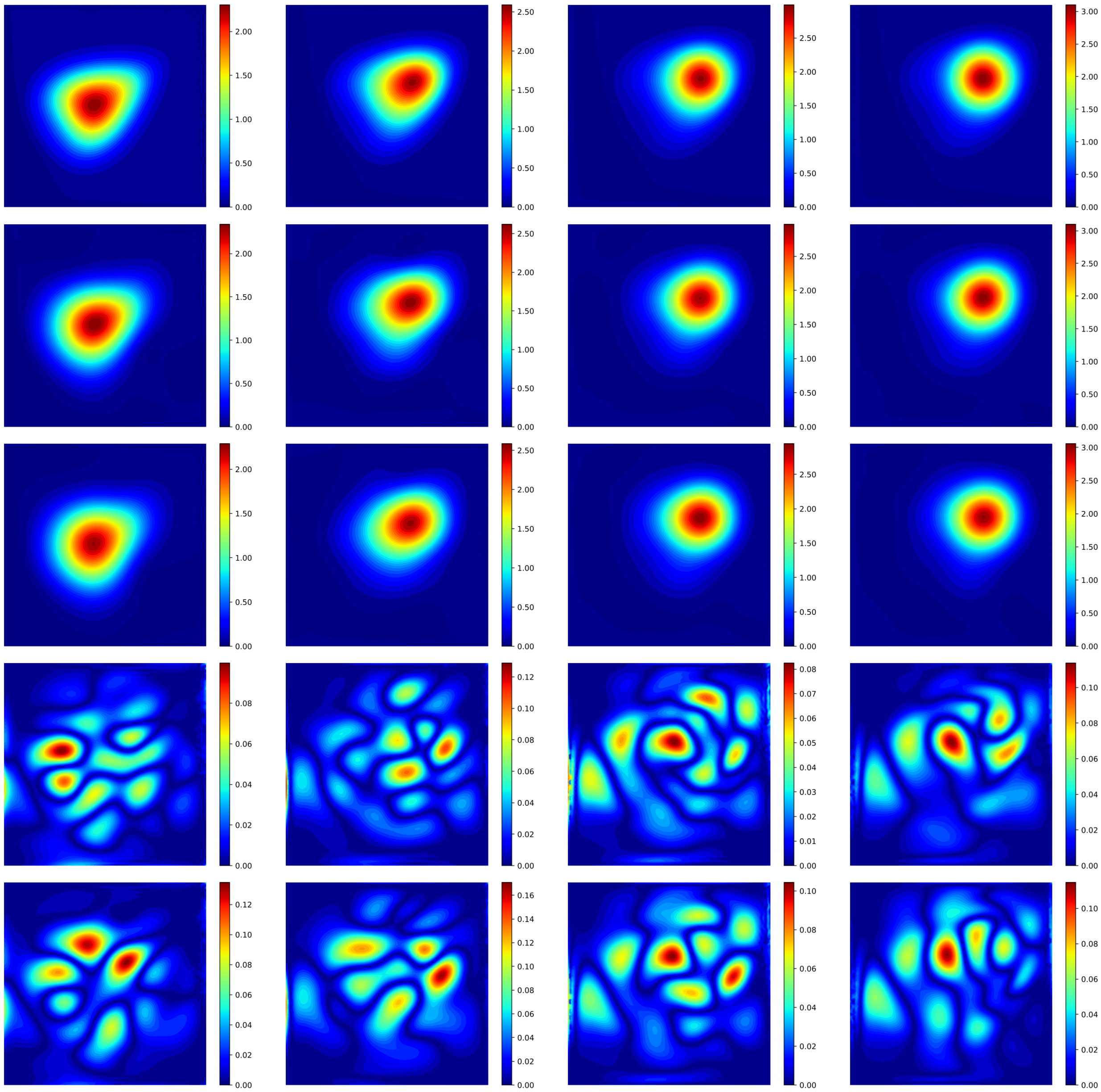

Figure 8. *Test 1.1.* Optimal transport in a vacuum. High-fidelity optimal state trajectory (first row), POD+AE reconstructions (second row), forward model predictions (third row), POD+AE reconstruction errors (fourth row) and forward model prediction errors (fifth row) at $t = 0.25, 0.5, 0.75, 1.0$ related to the test scenario parameters $\boldsymbol{\mu}_s = (0.31, 0.27)$.

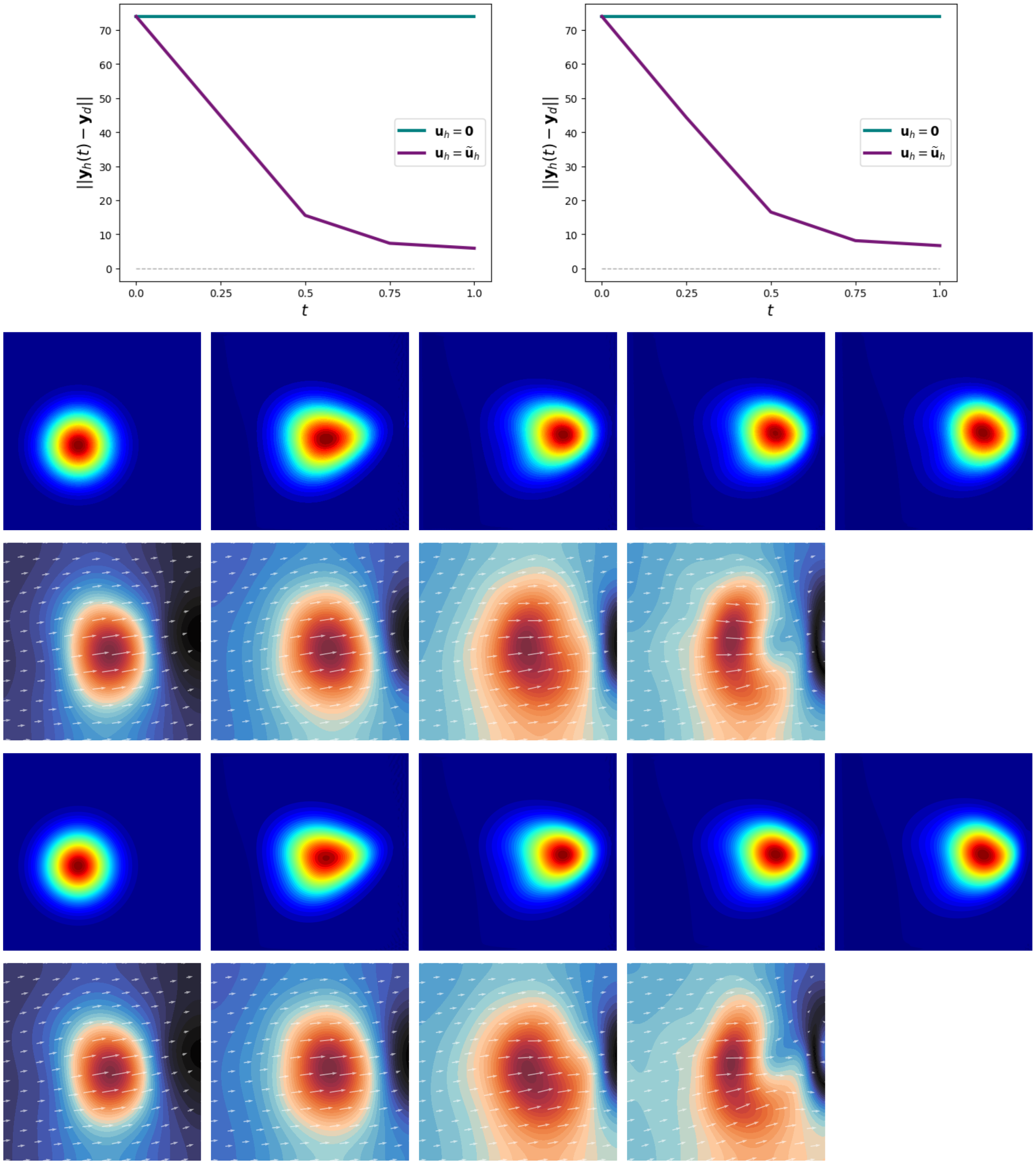

Figure 9. *Test 1.1.* Optimal transport in a vacuum. First row: discrepancy between the target configuration $\mathbf{y}_d$ centered at $(\mu_1^d, \mu_2^d) = (0.48, -0.03)$ and the state $\mathbf{y}_h(t)$ considering the deep learning-based reduced order feedback controller (left) and the latent feedback loop (right) at different time instants in the uncontrolled setting ($\mathbf{u}_h = 0$) and when exploiting the optimal control predicted by the policy ($\mathbf{u}_h = \tilde{\mathbf{u}}_h$). Other panels: system evolution driven by policy-based controls related to an initial state centered at $(\mu_1^0, \mu_2^0) = (-0.24, -0.14)$ and scenario parameters $\boldsymbol{\mu}_s = (0.48, -0.03)$, while exploiting the loop closure at the full-order (second and third rows) and latent (fourth and fifth rows) level. The control velocity fields on Ω are depicted through vector fields, with the underlying colours corresponding to their magnitude.

where $\mathcal{B}_{0.15}(0,0)$ is the circle centered at $(0,0)$ with radius 0.15 – and *(ii)* an underlying fluid flow brings the state upwards out of the domain. In the context of robotic swarms, this external fluid may represent a wind field (for aerial vehicles) or a water current (for underwater applications), affecting particle movement. The state system is now described in terms of the following Fokker-Plank equation

$$\begin{cases} \dfrac{\partial y}{\partial t} + \nabla \cdot (-\nu \nabla y + \mathbf{u} y + \mathbf{v} y) = 0 & \text{in } \Omega \times (0,T] \\ (-\nu \nabla y + \mathbf{u} y + \mathbf{v} y) \cdot \mathbf{n} = 0 & \text{on } \partial\Omega \times (0,T] \\ y(0) = y_0(\mu_1^0, \mu_2^0) & \text{in } \Omega \times \{t=0\} \end{cases} \tag{10}$$

where, differently from Equation (8), an additional transport term with velocity $\mathbf{v} : \Omega \to \mathbb{R}^2$ is considered in order to describe the fluid flow surrounding the density. Note that, while $\mathbf{v}$ steers the density upwards out of the domain complicating the control task, the state is passive with respect to this transport effect. Specifically, the velocity field $\mathbf{v}$ in Ω is modeled through (steady, for the sake of simplicity) Navier-Stokes equations, that are

$$\begin{cases} -\mu \Delta \mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v} + \nabla p = 0 & \text{in } \Omega \\ \text{div } \mathbf{v} = 0 & \text{in } \Omega \\ \mathbf{v} = \mathbf{0} & \text{on } \Gamma_{\text{obs}} \\ \mathbf{v} = \mathbf{v}_{\text{in}}(\gamma_{\text{in}}, \alpha_{\text{in}}) & \text{on } \Gamma_{\text{in}} \\ \mathbf{v} \cdot \mathbf{n} = 0 & \text{on } \Gamma_{\text{walls}} \\ (\mu \nabla \mathbf{v} - p)\mathbf{n} \cdot \mathbf{t} = 0 & \text{on } \Gamma_{\text{walls}} \\ (\mu \nabla \mathbf{v} - p)\mathbf{n} = 0 & \text{on } \Gamma_{\text{out}} \end{cases} \tag{11}$$

where $\mu = 0.01$ is the kinematic viscosity, $p : \Omega \to \mathbb{R}$ is the pressure field and $\mathbf{t}$ is the tangential versor to the boundary $\partial\Omega$. The underlying fluid enters the domain $\Omega = (-1,1)^2 \setminus \mathcal{B}_{0.15}(0,0)$ from the bottom side (inflow) $\Gamma_{\text{in}} = \partial\Omega \cap \{x_2 = -1\}$ with a Dirichlet boundary datum

$$\mathbf{v}_{\text{in}}(\gamma_{\text{in}}, \alpha_{\text{in}}) = \begin{bmatrix} (x_1+1)(1-x_1)\gamma_{\text{in}} \sin(\alpha_{\text{in}}) \\ \gamma_{\text{in}} \cos(\alpha_{\text{in}}) \end{bmatrix}$$

where γ_{in} and α_{in} are, respectively, the inflow velocity intensity and the angle of attack, while the parabolic factor $(x_1+1)(1-x_1)$ is helpful to avoid singularities at the domain corners. On the outflow, that is the top side $\Gamma_{\text{out}} = \partial\Omega \cap \{x_2 = 1\}$ where the flow exits the domain, homogeneous Neumann boundary conditions are considered. Moreover, we employ free-slip condition on the sidewalls $\Gamma_{\text{wall}} = \partial\Omega \cap \{x_1 = \pm 1\}$ and no-slip conditions on the obstacle boundary Γ_{obs}.

Similarly to the optimal transport problem in a vacuum detailed in Section 4.1, by minimizing the loss function in Equation (9) with $\beta = \beta_g = 0.2$, we aim to find the optimal control action that brings the state density from an initial configuration y_0 to the target destination y_d, that are

$$\begin{aligned} y_0(-0.5, \mu_2^0) &= \frac{10}{\pi} \exp\left(-10(x_1+0.5)^2 - 10(x_2 - \mu_2^0)^2\right) \\ y_d(0.5, \mu_2^d) &= \frac{10}{\pi} \exp\left(-10(x_1-0.5)^2 - 10(x_2 - \mu_2^d)^2\right) \end{aligned}$$

Note that, due to the presence of the obstacle, we set the mean x_1-coordinates of y_0 and y_d equal to -0.5 and 0.5, respectively. Note also that the integral on $\partial\Omega$ in the loss functional J helps to avoid collisions both with domain boundaries and the circular obstacle, as typically crucial in safety critical applications related to robotic swarms. In this setting, we consider the vector of scenario parameters $\boldsymbol{\mu}_s = (\mu_2^d, \gamma_{\text{in}}, \alpha_{\text{in}})$ in order to deal with both different final coordinates and different underlying flows. Figure 10 shows the test case setting, along with a comparison between the uncontrolled and the optimal state trajectories.

As far as the finite element discretization is concerned, we consider a mesh discretizing the domain Ω with 7681 nodes and step size $h = 0.05$. While $\mathbb{P}_1$ finite elements are taken into account when discretizing state and control in Equation (10), Taylor-Hood $\mathbb{P}_2$-$\mathbb{P}_1$ elements are considered in Equation (11) to guarantee the inf-sup stability and the well-posedness of the problem [58]. The degrees of freedom of the discrete state $\mathbf{y}_h$, control $\mathbf{u}_h$, velocity $\mathbf{v}_h$ and pressure $\mathbf{p}_h$ variables end up being, respectively, $N_h^y = 7681$, $N_h^u = 15362$,

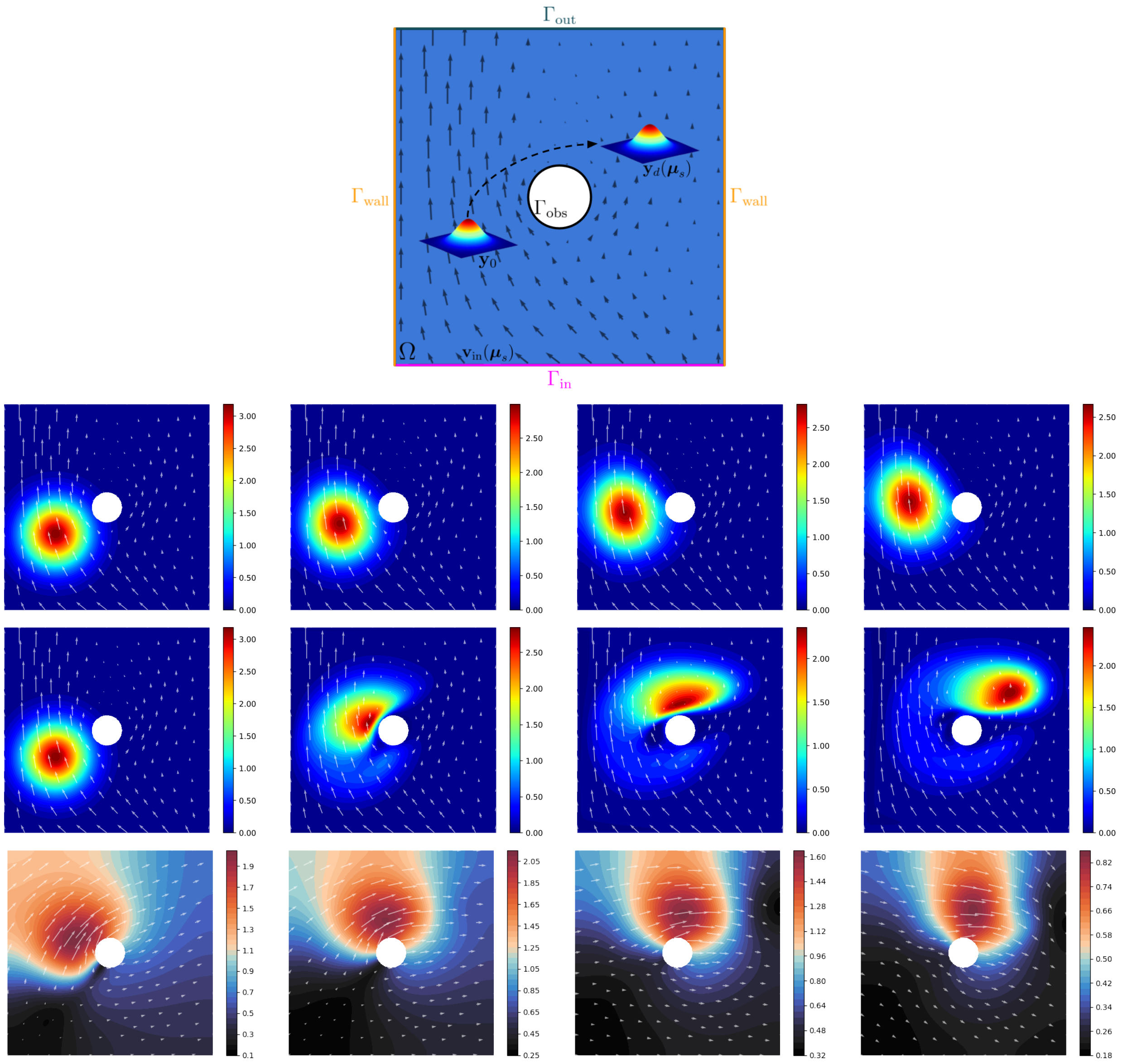

Figure 10. *Test 1.2.* Optimal transport in a fluid. Top: representation of an optimal state trajectory in a fluid within the domain Ω, where $\mathbf{y}_0$ stands for the initial density centered at $(\mu_1^0, \mu_2^0) = (-0.5, -0.25)$, $\mathbf{y}_d(\boldsymbol{\mu}_s)$ represents the target configuration centered at $(\mu_1^d, \mu_2^d) = (0.5, 0.39)$ and $\mathbf{v}_{\text{in}}(\boldsymbol{\mu}_s)$ is the inlet velocity with inflow intensity $\gamma_{\text{in}} = 0.40$ and angle of attack $\alpha_{\text{in}} = -0.92$, that is $\boldsymbol{\mu}_s = (0.39, 0.40, -0.92)$. Other panels: space-varying uncontrolled state, optimal state and control at $t = 0, 0.25, 0.5, 0.75$ related to the scenario parameters $\boldsymbol{\mu}_s = (0.39, 0.40, -0.92)$. The underlying fluid velocity vector field $\mathbf{v}_h$ on Ω is depicted together with the state. The control velocity fields on Ω are depicted through vector fields, with the underlying colours corresponding to their magnitude.

$N_h^v = 60732$ and $N_h^p = 7681$. Note that, to keep the Péclet number always smaller than 1, thus avoiding instabilities while solving Equation (10), an artificial diffusion equal to $0.5h\gamma_{\text{in}}$ is added to the diffusion coefficient ν [58]. Instead, regarding the time discretization, we consider an evenly spaced time grid spanning $[0, T]$ with time step $\Delta t = 0.25$, where the final time T is set equal to 1.5 and $N_t = 6$.

In the data generation step, we simulate $N_s = 150$ trajectories through the adjoint method implemented in `dolfin-adjoint`, exploiting L-BFGS-B as optimization algorithm, a tolerance equal to 10^{-6} and 500 as maximum number of iterations. Every full-order optimal trajectory – which requires, on average, 22 minutes to be computed – is related to scenario parameters and initial x_2-coordinate μ_2^0 randomly sampled, respectively, in the parameter space $\mathcal{P} = (-0.5, 0.5) \times (0.1, 1.0) \times (-1.0, 1.0)$ and in the interval $(-0.5, 0.5)$. Note that, since $\gamma_{\text{in}} \in (0.1, 1.0)$, the Reynolds numbers taken into account range from 20 to 200. The $N_s N_t = 900$ snapshots are then split into training and test set with a $90 : 10$ ratio.

The state and control snapshots are then reduced through POD, looking at the singular values decays and the reconstruction errors in order to select the number of modes to retain. Specifically, 200 and 160 modes are considered for state and control, respectively, ending up with reduction errors equal to $\varepsilon_{\text{rel}}^y = 0.29\%$ and $\varepsilon_{\text{rel}}^u = 0.29\%$. As already highlighted in the previous section, a linear projection is not enough in this context to achieve low-dimensional latent spaces for both state and control, as confirmed by the polynomial singular values decays in Figure 11.

To achieve lower-dimensional subspaces, allowing for a lighter policy that is faster to train and evaluate online, we employ the POD+AE reduction strategy proposed by [27]. In particular, thanks to the nonlinear projection provided by the autoencoders with leaky Relu as activation function, we take into account latent spaces of dimension $N_y = N_u = 14$. Both the state and control encoders are made of 1 hidden layer having 100 neurons, while the corresponding decoders consist of 2 hidden layers with 100 neurons each. The low-dimensional policy π_N is instead modeled through a deep feedforward neural network with 3 hidden layers of 50 neurons each and leaky Relu as activation function. Note that, together with the latent state coordinates and the scenario parameters, the input of π_N is enriched by considering meaningful problem-driven quantities, such as $\tan(\mu_2^d)$, $\tan(\alpha_{\text{in}})$, $\gamma_{\text{in}} \sin(\alpha_{\text{in}})$ and $\gamma_{\text{in}} \cos(\alpha_{\text{in}})$. Starting from the initialization of the weights proposed by [32], we train the networks in 1 hour and 25 minutes minimizing the cumulative loss function J_{NN} with $\lambda_1 = \lambda_2 = \lambda_3 = 0.001$ through the L-BFGS optimization algorithm. After training, it is possible to accurately compress and reconstruct the state and control trajectories, with reconstruction errors equal to $\varepsilon_{\text{rel}}^y = 4.39\%$ and $\varepsilon_{\text{rel}}^u = 4.43\%$. Similarly, the low-dimensional policy π_N is now able to correctly predict the optimal control actions starting from state data in the test set, with prediction error equal to 3.62% at the latent level and 7.08% after POD+AE decoding. To visually assess the accuracy and the generalization capabilities of the trained networks on test scenarios, Figure 12 displays a control test trajectory along with the corresponding POD+AE reconstruction and policy prediction.

Whenever only a few lagging state data are available online – in the worst-case scenario, only the initial state is known – the latent feedback loop can be trained and exploited to continuously control the system anyway. This is possible if we take into account a low-dimensional surrogate model for the time-advancing scheme of the system at the latent level, that is we model φ_N with a deep feedforward neural network having 3 hidden layers of 50 neurons each and leaky Relu as activation function. While φ_N is initialized following the procedure in [32], the autoencoders and the policy inherit the initial weights from the previous training. As detailed in Section 3.3, the cumulative loss function J_{NN} with $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_6 = 0.0001$ and $\lambda_4 = \lambda_5 = 0.01$ is minimized through the L-BFGS optimization algorithm in 1 hour and 31 minutes. The POD+AE reconstruction errors on test data are equal to $\varepsilon_{\text{rel}}^y = 3.71\%$ and $\varepsilon_{\text{rel}}^u = 4.40\%$, while the policy prediction error after-decoding is 7.11%. Instead, the prediction-from-data and the prediction-from-policy of the forward model φ_N are accurate up to an error equal to, respectively, 2.32% and 2.48% at the latent level, while they increase to 7.29% and 7.63% after POD+AE decoding. The POD+AE state reconstructions and the forward model predictions related to a trajectory in the test set are visualized in Figure 13.

Thanks to the trained deep-learning based reduced order feedback controller, we can now move the state for new initial configurations and scenarios, which have not been seen during training. Figure 14 presents the state evolutions related to two different test cases with same initial state position $(\mu_1^0, \mu_2^0) = (-0.5, 0.0)$, target destination $(\mu_1^d, \mu_2^d) = (0.5, 0.0)$ and inflow velocity intensity $\gamma_{\text{in}} = 0.5$, but different angles of attack ($\alpha_{\text{in}} = 0.5$ for the first test case, while $\alpha_{\text{in}} = -0.5$ in the second setting). In the first scenario, we assume that we can continuously monitor the system, that is we have access to high-dimensional state data online. Instead, in the second test case, the latent feedback loop is necessary since only the initial configuration

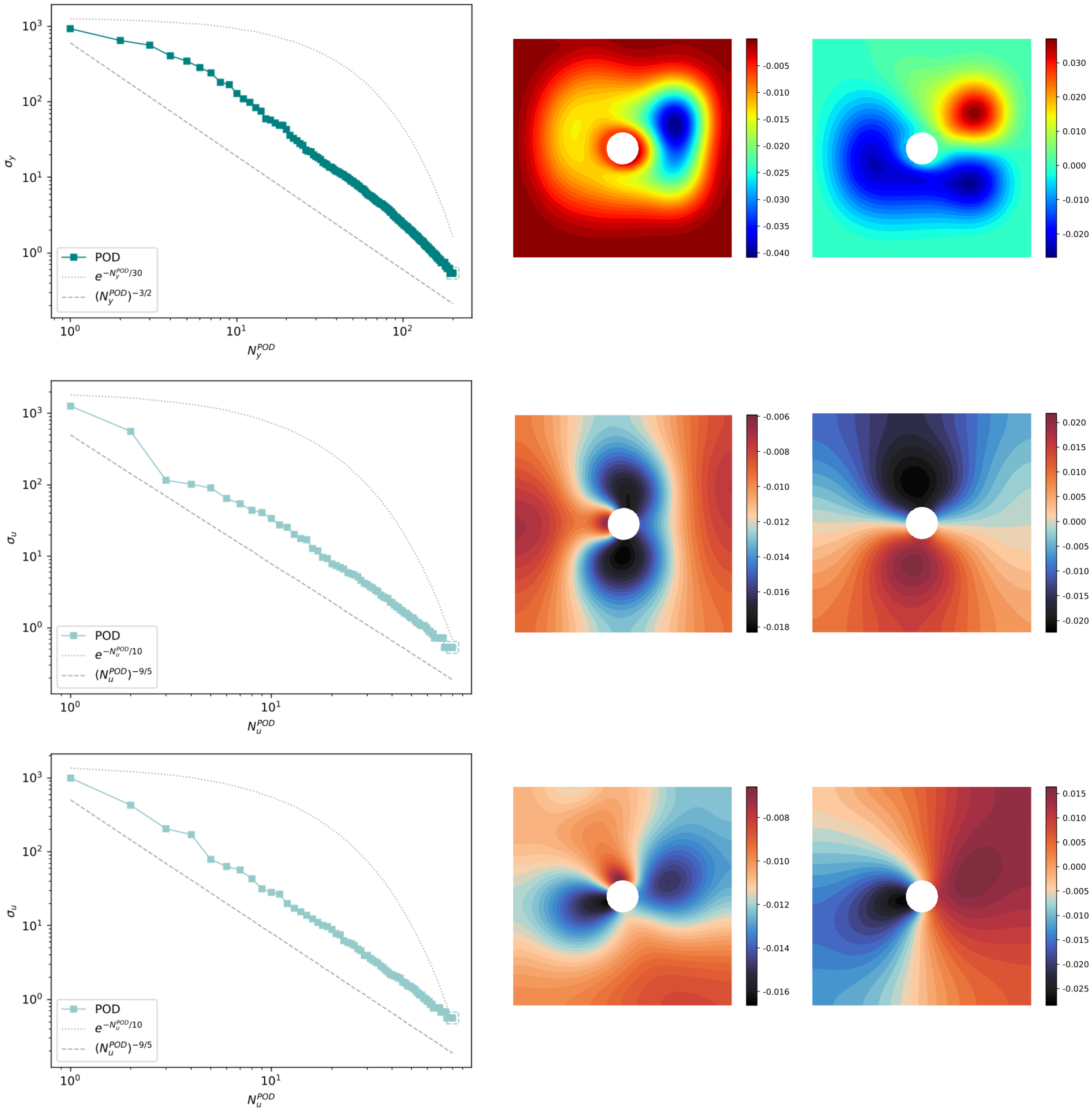

Figure 11. *Test 1.2.* Optimal transport in a fluid. Singular values decay in log-log scale along with the two most energetic POD modes related to the state (top), x_1 component (center) and x_2 component (bottom) of the control.

is known. By a visual inspection, it is possible to assess that in both cases the state density is moved towards the target destination. In particular, the reduced order feedback controller is capable of exploiting the underlying fluid flow in order to steer the system with the less expensive and energetic control action, as required by the regularization terms in the loss function J. Indeed, when the angle of attack is oriented to the right-hand side of the domain, the state transition occurs below the obstacle, taking advantage of the underlying current to reach the target location. In contrast, when the angle of attack is left-oriented, the controller exploits very different control velocity fields, splitting the state density into two clusters, where the bigger one moves above the obstacle, thus mainly avoiding countercurrent routes that would require more expensive control actions. Moreover, it is possible to note that, once the state reaches the target position, the optimal control is concentrated in the top-right region of Ω in order to balance the upward thrust provided by

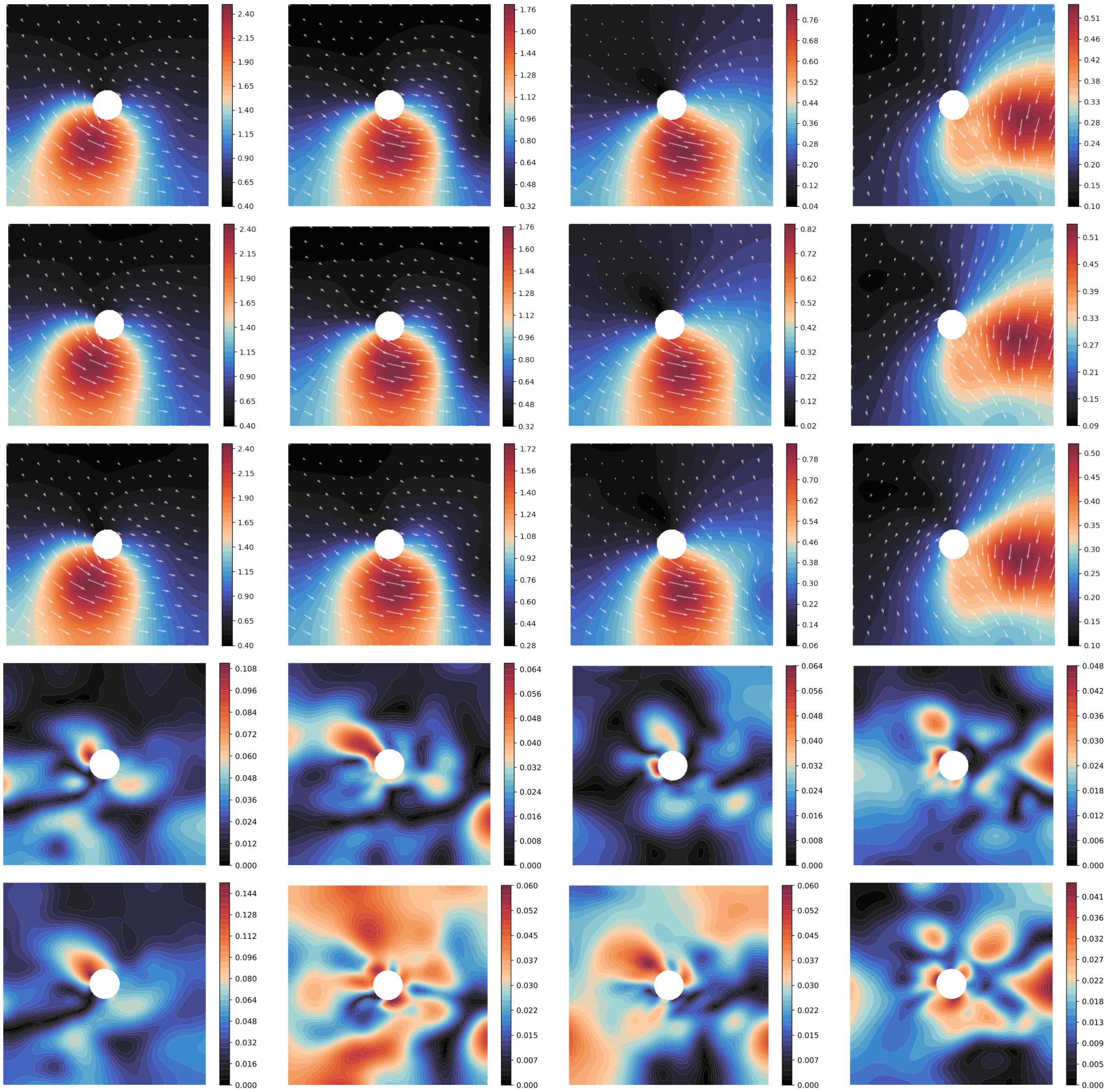

Figure 12. *Test 1.2.* Optimal transport in a fluid. High-fidelity optimal control trajectory (first row), POD+AE reconstructions (second row), policy predictions (third row), POD+AE reconstruction errors (fourth row) and policy prediction errors (fifth row) at $t = 0, 0.25, 0.5, 0.75$ related to the test scenario parameters $\boldsymbol{\mu}_s = (-0.30, 0.49, 0.59)$. The control velocity fields on Ω are depicted through vector fields, with the underlying colours corresponding to their magnitude.

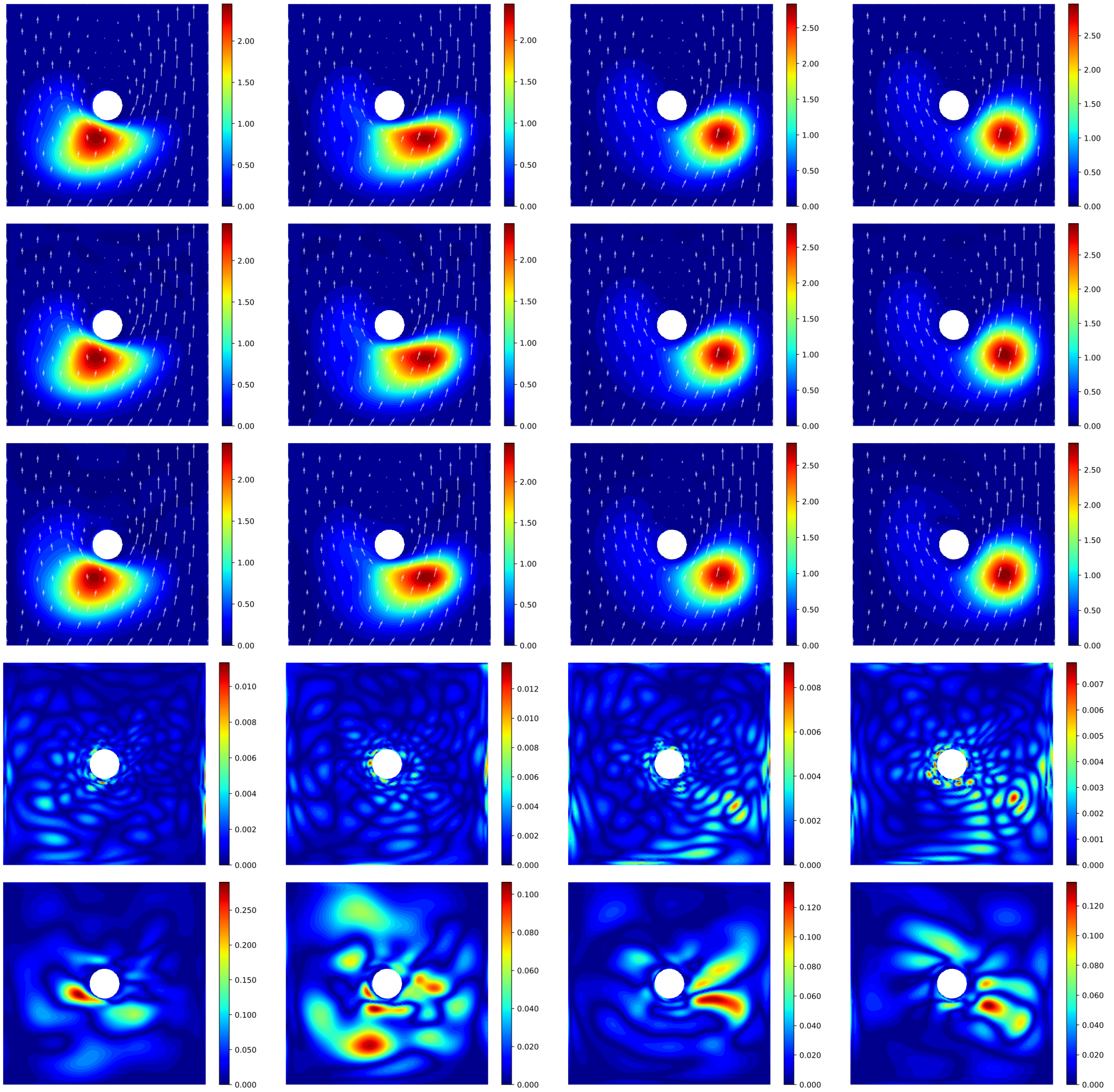

Figure 13. *Test 1.2.* Optimal transport in a fluid. High-fidelity optimal state trajectory (first row), POD+AE reconstructions (second row), forward model predictions (third row), POD+AE reconstruction errors (fourth row) and forward model prediction errors (fifth row) at $t = 0.25, 0.5, 0.75, 1.0$ related to the test scenario parameters $\boldsymbol{\mu}_s = (-0.30, 0.49, 0.59)$. The underlying fluid velocity vector field $\mathbf{v}_h$ on Ω is depicted together with the state.

$\mathbf{v}_h$. As far as computational times are concerned, the deep learning-based reduced order feedback controller with model closure at the full-order level requires 1.21 seconds to provide the control actions and simulate the states for all the N_t time steps in the test case considered, while it boils down to 0.05 seconds in the case of latent feedback loop, with a remarkably high speed-up with respect to full-order methods (1100× for the loop closure at the full-order level, 26500× for the latent feedback loop). As already highlighted in the previous test case, the proposed controllers remain effective even when taking into account different (possibly inhomogeneous) time discretizations than the one exploited offline.

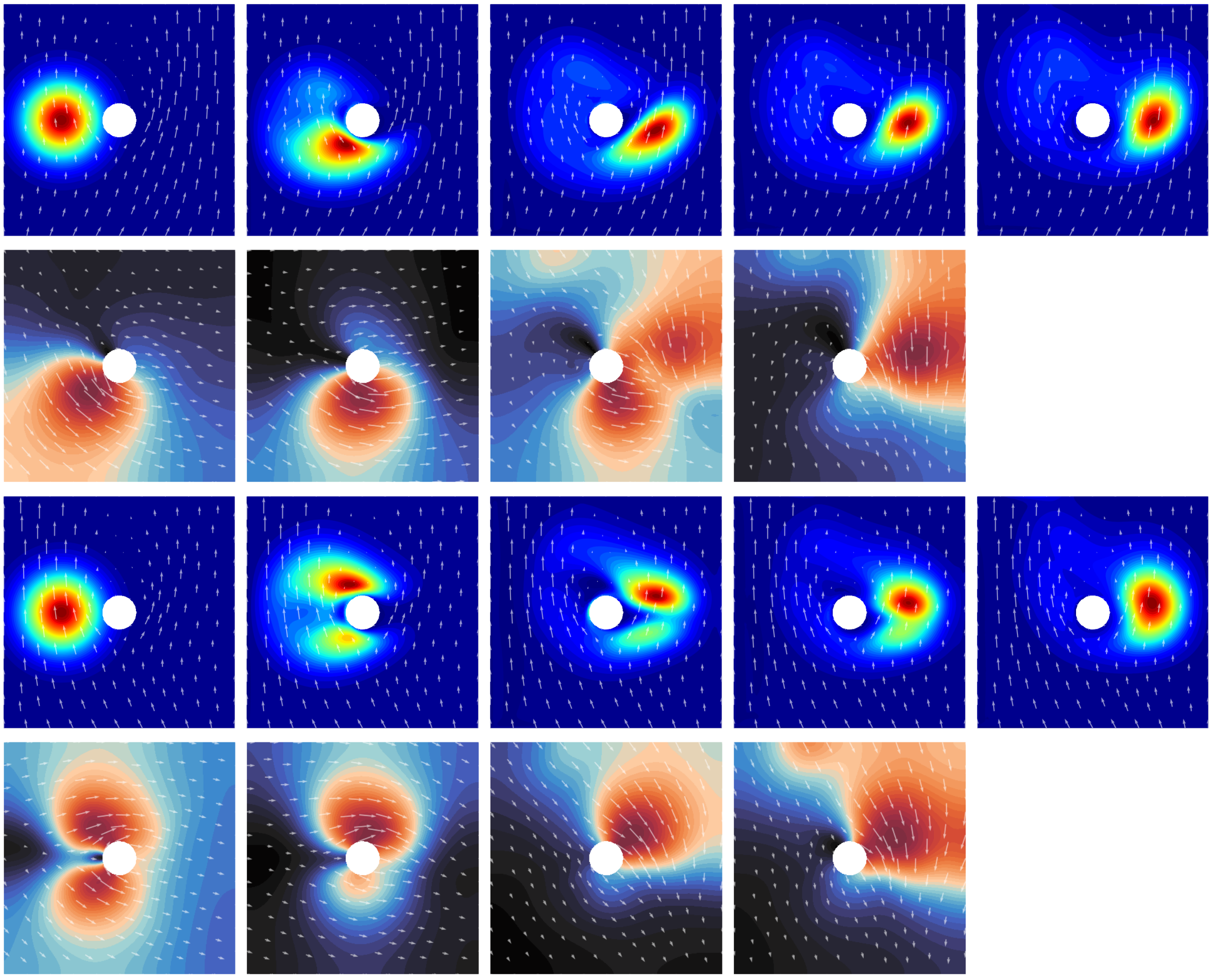

Figure 14. *Test 1.2.* Optimal transport in a fluid. First and second rows: system evolution driven by policy-based controls at $t = 0, 0.25, 0.75, 1.0, 1.5$ related to an initial state centered at $(\mu_1^0, \mu_2^0) = (-0.5, 0.0)$ and a vector of scenario parameters $\boldsymbol{\mu}_s = (0.0, 0.5, 0.5)$, while exploiting the loop closure at the full-order level. Third and fourth rows: system evolution driven by policy-based controls at $t = 0, 0.25, 0.75, 1.0, 1.5$ related to an initial state centered at $(\mu_1^0, \mu_2^0) = (-0.5, 0.0)$ and a vector of scenario parameters $\boldsymbol{\mu}_s = (0.0, 0.5, -0.5)$, while exploiting the latent feedback loop. The underlying fluid velocity vector fields $\mathbf{v}_h$ on Ω is depicted together with the state. The control velocity fields on Ω are depicted through vector fields, with the underlying colours corresponding to their magnitude.

In this work, state information in the online phase is generated through full-order, high-fidelity simulations of the underlying dynamics, which is assumed to be known. However, in several practical settings, the state snapshots are captured through sensors, which may be affected by noise. To assess the generalization capabilities of our controller against disturbances, we try to optimally transport the state density in a parametric fluid while considering online state information corrupted by noise. We recall that noisy data are exploited here only in the online evaluation phase, while the training is entirely performed with the aforementioned noise-free snapshots. In particular, the noise is modeled through a Gaussian distribution, that is $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, being $\mathbf{0}$ the N_h^y-dimensional zero vector and I the $N_h^y \times N_h^y$ identity matrix, where

5 different noise levels are taken into account, namely the standard deviations $\sigma = 0.03, 0.075, 0.15, 0.3, 0.6$ (which correspond approximately to, respectively, $1\%, 2.5\%, 5\%, 10\%, 20\%$ of the range of state values). Figure 15 compares the performances of the deep learning-based reduced order feedback controllers in 100 random scenarios across different noise levels, along with the noise-free case. In particular, for every test case, we compute the probability of arrival

$$\mathbb{P}(Y \in \mathcal{B}_{0.5}(\mu_1^d, \mu_2^d)) \approx \int_{\mathcal{B}_{0.5}(\mu_1^d, \mu_2^d)} y(T) d\Omega$$

where Y is a random variable with probability density function equal to $y(T)$, that is the final state obtained by exploiting our controllers, extended on the whole $\mathbb{R}^2$. Instead, the circle centered at (μ_1^d, μ_2^d) with radius 0.5, namely $\mathcal{B}_{0.5}(\mu_1^d, \mu_2^d)$, corresponds approximately to the 90%-confidence region of the target density, that is $\mathbb{P}(Y_d \in \mathcal{B}_{0.5}(\mu_1^d, \mu_2^d)) \approx 0.9$, where Y_d is a random variable having y_d as probability density function. Despite the noise added to the state data, it is possible to assess that both the feedback loop at the full-order and latent level are capable of steering the state density towards the final target, with results comparable to the setting without noise. This is mainly due to the POD+AE reduction, which is able to extract the relevant features for control design, while discarding erratic and high-frequency disturbances. When the noise level is remarkably high, the accuracy of the feedback latent loop is lower since its optimal transport predictions are entirely based on a single noisy state snapshot at $t = 0$. Instead, the full-order counterpart is still able to steer the state towards the final destination taking advantage of the multiple corrupted state data received at every time step. Note that our reduced order feedback controllers perform all the 600 simulations considered in this analysis in 720 and 30 seconds, respectively, while a full-order solver based on, e.g., FEM would require approximately 8 days of computations. An example of a state trajectory driven by our controllers when taking into account a standard deviation σ equal to 0.3, a random initial setting and a random scenario is available in Figure 15.

5. Conclusions

In this work, we present a deep learning-based reduced order feedback controller capable of steering dynamical systems very rapidly. Unlike several control strategies available in the literature, the proposed framework is capable of dealing with both *(i)* complex and parametrized dynamics modeled via (possibly nonlinear) time-dependent PDEs, *(ii)* high-dimensional state observations and *(iii)* distributed control actions. This allows us to rapidly control complicated systems in multiple scenarios unseen during training, as often required in applications, amortizing the offline cost due to data generation and networks training. Although only full-order synthetic data have been exploited in this work, the proposed framework can be easily extended to (even low- and high-dimensional) sensor data or videos capturing the dynamics, paving the way for cheap and portable control devices. Indeed, as shown in Section 4.2 when dealing with the optimal transport problem in a fluid, the proposed architectures are capable of dealing with noisy data, thanks to the reduction carried out and the feedback signal considered. To handle the high-dimensionality of the data, we extract low-dimensional features relevant for control design through very accurate and efficient non-intrusive reduced order models, such as POD, AE and POD+AE. Note that we consider a unified framework with different reduction strategies in order to exploit the most effective one for each problem at hand, extending the current state-of-the-art on non-intrusive ROMs to closed-loop control problems. Thanks to data compression, it is now possible to learn a low-dimensional surrogate model for the policy bridging state and control latent spaces, which is faster to train and evaluate online.

Inspired by the MPC approach, we consider a model closure at the latent level, here referred to as latent feedback loop, embedding a surrogate model of the reduced order dynamics in the controller. Our controller is therefore able to continuously control the dynamics of interest even in the absence of online state data, avoiding losses of optimality, performance or stability, while overcoming the necessity of continuous monitoring of the dynamical systems.

Throughout the optimal transport test cases presented, we demonstrate the accuracy of our approach in very challenging settings, characterized by high-dimensional variables, transport-dominated trajectories and complex parameters-to-solution dependencies. The speed-up of our controllers with respect to full-order high-fidelity models based on, e.g., FEM is remarkably high. Indeed, the proposed control strategies consist

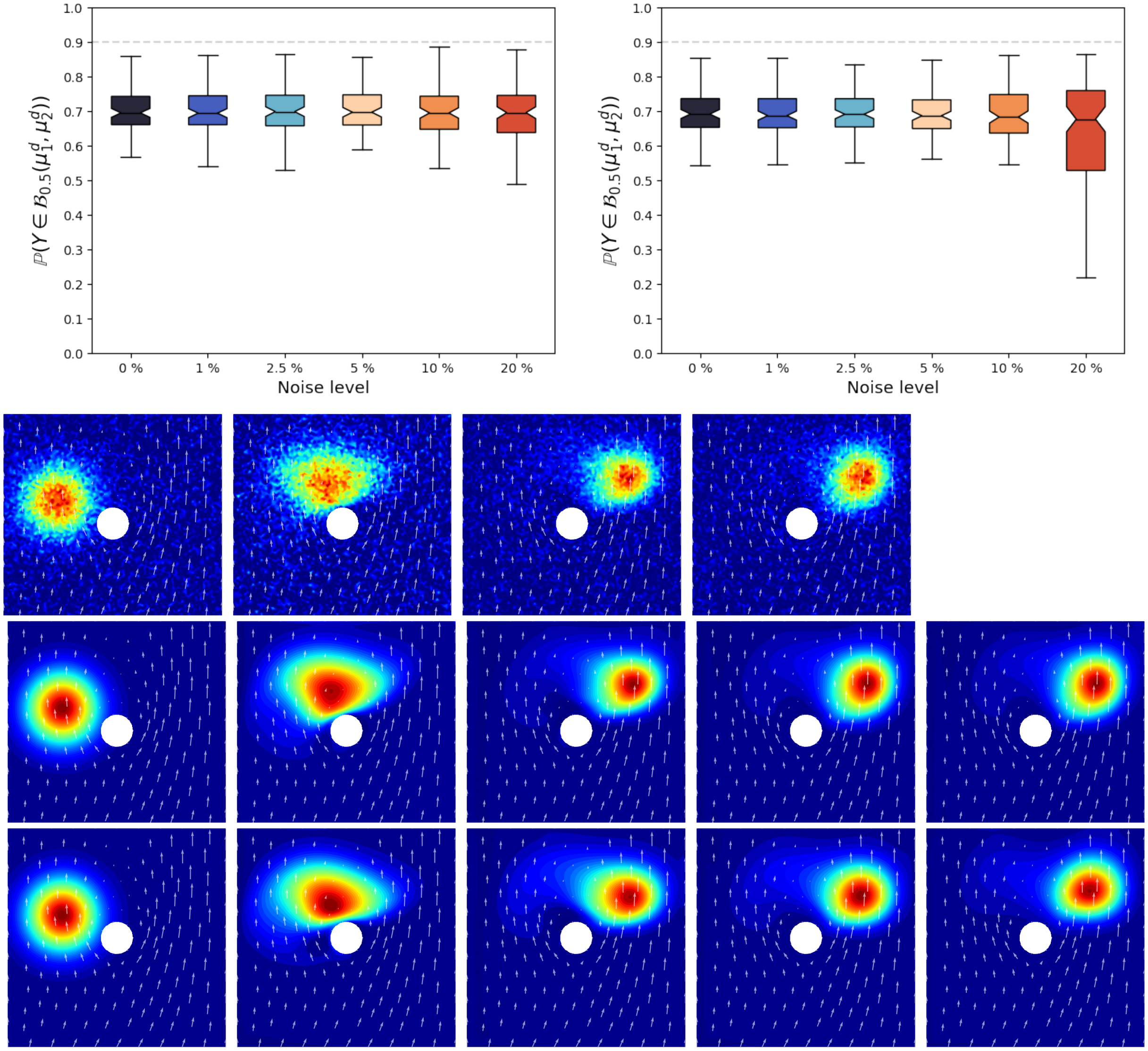

Figure 15. *Test 1.2.* Optimal transport in a fluid. First row: boxplots of the probabilities of arrival in 100 random scenarios for different noise levels when considering the deep learning-based reduced order feedback controller at full-order level (left) and the latent feedback loop (right). Second row: online state data at $t = 0, 0.25, 0.75, 1.0$ corrupted by Gaussian random noise with standard deviation $\sigma = 0.3$ related to an initial state centered at $(\mu_1^0, \mu_2^0) = (-0.5, 0.2)$ and a vector of scenario parameters $\boldsymbol{\mu}_s = (0.42, 0.39, 0.48)$. Other panels: system evolution at $t = 0, 0.25, 0.75, 1.0, 1.5$ driven by policy-based controls exploiting state data corrupted by Gaussian random noise with standard deviation $\sigma = 0.3$ related to an initial state centered at $(\mu_1^0, \mu_2^0) = (-0.5, 0.2)$ and a vector of scenario parameters $\boldsymbol{\mu}_s = (0.42, 0.39, 0.48)$, while exploiting the loop closure at the full-order (third row) and latent (fourth row) level. The underlying fluid velocity vector field $\mathbf{v}_h$ on Ω is depicted together with the state.

of efficient forward passes through the considered networks, which usually exploit light architectures due to the dimensionality reduction performed. Moreover, after training, the same architecture may be recycled to obtain rapidly different control actions related to different scenarios of interest.

The proposed controller may be extended in future works in multiple directions. For instance, as already mentioned, different data sources may be considered, such as sensors or cameras recording the system evolution, possibly corrupted by noise. In addition, multi-agent reinforcement learning strategies may be exploited, overcoming the necessity of a high-fidelity full-order OCP solver to generate training data, while still considering distributed parametrized systems. Another possible improvement may be dedicated to speed up the offline phase. For instance, both data-driven or physics-informed surrogate models for the state and adjoint equations may be introduced in order to rapidly augment the dataset of (possibly few) high-fidelity optimal snapshots with many low-fidelity data.

Acknowledgments

AM acknowledges the Project "Reduced Order Modeling and Deep Learning for the real- time approximation of PDEs (DREAM)" (Starting Grant No. FIS00003154), funded by the Italian Science Fund (FIS) - Ministero dell'Università e della Ricerca and the project FAIR (Future Artificial Intelligence Research), funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence). MT and AM are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

References

[1] E. Aggelogiannaki and H. Sarimveis. Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. *Computers and Chemical Engineering*, 32(6):1225–1237, 2008.

[2] G. Albi, S. Bicego, and D. Kalise. Control of high-dimensional collective dynamics by deep neural feedback laws and kinetic modelling. arXiv:2404.02825, 2024.

[3] A. Alla, B. Haasdonk, and A. Schmidt. Feedback control of parametrized PDEs via model order reduction and dynamic programming principle. *Advances in Computational Mathematics*, 46(9), 2020.

[4] A. Alla and M. Hinze. HJB-POD feedback control for navier-stokes equations. In *Progress in Industrial Mathematics at ECMI 2014*, pages 861–868, Cham, 2016. Springer International Publishing.

[5] A. Alla and S. Volkwein. Asymptotic stability of POD based model predictive control for a semilinear parabolic PDE. *Advances in Computational Mathematics*, 41(5):1073–1102, 2015.

[6] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. Rognes, and G. Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100):9–23, 2015.

[7] D. Amsallem, M. Zahr, Y. Choi, and C. Farhat. Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51(4):919–940, 2015.

[8] E. A. Antonelo, E. Camponogara, L. O. Seman, J. P. Jordanou, E. R. de Souza, and J. F. Hübner. Physics-informed neural nets for control of dynamical systems. *Neurocomputing*, 579:127419, 2024.

[9] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhäuser Boston, MA, 2009.

[10] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[11] P. Benner, E. Sachs, and S. Volkwein. Model order reduction for PDE constrained optimization. *International Series of Numerical Mathematics*, 165:303–326, 2014.

[12] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz. Deep model predictive flow control with limited sensor data and online learning. *Theoretical and Computational Fluid Dynamics*, 34(4):577–591, 2020.

[13] L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, and K. Gustavsson. Zermelo's problem: Optimal point-to-point navigation in 2D turbulent flows using reinforcement learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10):103138, 10 2019.

[14] N. Botteghi, K. Alaa, M. Poel, B. Sirmacek, C. Brune, A. Mersha, and S. Stramigioli. Low dimensional state representation learning with robotics priors in continuous action spaces. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 190–197, 2021.

[15] N. Botteghi and U. Fasel. Parametric PDE Control with Deep Reinforcement Learning and Differentiable L0-Sparse Polynomial Policies. arXiv:2403.15267, 2024.

[16] S. Brunton and J. N. Kutz. *Data-driven science and engineering: machine learning, dynamical systems, and control.* Cambridge University Press, 2019.

[17] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz. Modern Koopman Theory for Dynamical Systems. *SIAM Review*, 64(2):229–340, 2022.

[18] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 113(15):3932–3937, 2016.

[19] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Sparse Identification of Nonlinear Dynamics with Control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[20] M. A. Bucci, O. Semeraro, A. Allauzen, G. Wisniewski, L. Cordier, and L. Mathelin. Control of chaotic systems by deep reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2231):20190351, 2019.

[21] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[22] E. Camacho and C. Bordons. *Model Predictive Control.* Springer London, 2004.

[23] Y. Chen, Y. Shi, and B. Zhang. Optimal Control Via Neural Networks: A Convex Approach. arXiv:1805.11835, 2018.

[24] A. Draeger, S. Engell, and H. Ranke. Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15(5):61–66, 1995.

[25] N. R. Franco, A. Manzoni, and P. Zunino. A Deep Learning approach to Reduced Order Modeling of parameter dependent Partial Differential Equations. *Mathematics of Computation*, 92(340):483–524, 2023.

[26] S. Fresca, L. Dede', and A. Manzoni. A Comprehensive Deep Learning-Based Approach to Reduced Order Modeling of Nonlinear Time-Dependent Parametrized PDEs. *Journal of Scientific Computing*, 87(2):1–36, 2021.

[27] S. Fresca and A. Manzoni. POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388:114181, 2022.

[28] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, and E. Hachem. A review on deep reinforcement learning for fluid mechanics. *Computers and Fluids*, 225, 2021.

[29] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[30] J. Ghiglieri and S. Ulbrich. Optimal flow control based on POD and MPC and an application to the cancellation of Tollmien–Schlichting waves. *Optimization Methods and Software*, 29(5):1042–1074, 2014.

[31] L. Guastoni, J. Rabault, P. Schlatter, H. Azizpour, and R. Vinuesa. Deep reinforcement learning for turbulent drag reduction in channel flows. *The European Physical Journal E*, 46(4):27, 2023.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[33] J. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363, 2018.

[34] J. S. Hesthaven, G. Rozza, and B. Stamm. Certified Reduced Basis Methods for Parametrized Partial Differential Equations. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, pages 1–131, 2015.

[35] M. K. Hickner, U. Fasel, A. G. Nair, B. W. Brunton, and S. L. Brunton. Data-Driven Unsteady Aeroelastic Modeling for Control. *AIAA Journal*, 61(2):780–792, 2023.

[36] M. Hüttenrauch, A. Šošić, and G. Neumann. Deep reinforcement learning for swarm systems. *J. Mach. Learn. Res.*, 20(1):1966–1996, jan 2019.

[37] T. Ishize, H. Omichi, and K. Fukagata. Flow control by a hybrid use of machine learning and control theory. arXiv:2311.08624, 2023.

[38] J. Jeon, J. Rabault, J. Vasanth, F. Alcántara-Ávila, S. Baral, and R. Vinuesa. Advanced deep-reinforcement-learning methods for flow control: group-invariant and positional-encoding networks improve learning speed and quality. arXiv:2407.17822, 2024.

[39] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2219), 2018.

[40] S. Klus, F. Nüske, S. Peitz, J.-H. Niemann, C. Clementi, and C. Schütte. Data-driven approximation of the Koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

[41] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[42] K. Kunisch and S. Volkwein. Control of the Burgers Equation by a Reduced-Order Approach Using Proper Orthogonal Decomposition. *Journal of Optimization Theory and Applications*, 102(2):345–371, 1999.

[43] K. Kunisch and S. Volkwein. Proper orthogonal decomposition for optimality systems. *ESAIM: Modélisation mathématique et analyse numérique*, 42(1):1–23, 2008.

[44] K. Kunisch, S. Volkwein, and L. Xie. HJB-POD-Based Feedback Design for the Optimal Control of Evolution Problems. *SIAM Journal on Applied Dynamical Systems*, 3(4):701–722, 2004.

[45] F. Leibfritz and S. Volkwein. Reduced order output feedback control design for PDE systems using proper orthogonal decomposition and nonlinear semidefinite programming. *Linear Algebra and its Applications*, 415(2):542–575, 2006. Special Issue on Order Reduction of Large-Scale Systems.

[46] S. Levine, A. Kumar, G. Tucker, and F. Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. arXiv:2005.01643, 2020.

[47] D. Luo, T. O'Leary-Roseberry, P. Chen, and O. Ghattas. Efficient PDE-Constrained optimization under high-dimensional uncertainty using derivative-informed neural operators. arXiv:2305.20053, 2023.

[48] P. Ma, Y. Tian, Z. Pan, B. Ren, and D. Manocha. Fluid directed rigid body control using deep reinforcement learning. *ACM Transactions on Graphics*, 37(4), 2018.

[49] A. Manzoni and S. Pagani. A certified RB method for PDE-constrained parametric optimization problems. *Communications in Applied and Industrial Mathematics*, 10(1):123–152, 2019.

[50] A. Manzoni, A. Quarteroni, and S. Salsa. *Optimal Control of Partial Differential Equations*. Springer Cham, 2021.

[51] M. Mirzakhanloo, S. Esmaeilzadeh, and M.-r. Alam. Active cloaking in Stokes flows via reinforcement learning. *Journal of Fluid Mechanics*, 903(A34), 2020.

[52] S. K. Mitusch, S. W. Funke, and J. S. Dokken. dolfin-adjoint 2018.1: automated adjoints for fenics and firedrake. *Journal of Open Source Software*, 4(38):1292, 2019.

[53] F. Negri, G. Rozza, A. Manzoni, and A. Quarteroni. Reduced basis method for parametrized elliptic optimal control problems. *SIAM Journal on Scientific Computing*, 35(5):A2316–A2340, 2013.

[54] S. Peitz and K. Bieker. On the universal transformation of data-driven models to control systems. *Automatica*, 149:110840, 2023.

[55] S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of PDEs. *Automatica*, 106:184–191, 2019.

[56] S. Peitz and S. Klus. *Feedback Control of Nonlinear PDEs Using Data-Efficient Reduced Order Models Based on the Koopman Operator*, pages 257–282. Springer International Publishing, Cham, 2020.

[57] S. Peitz, J. Stenner, V. Chidananda, O. Wallscheid, S. L. Brunton, and K. Taira. Distributed control of partial differential equations using convolutional reinforcement learning. *Physica D: Nonlinear Phenomena*, 461:134096, 2024.

[58] A. Quarteroni. *Numerical Models for Differential Problems*. Springer Cham, 2017.

[59] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: An introduction*. Springer Cham, 2015.

[60] J. Rabault, F. Ren, W. Zhang, H. Tang, and H. Xu. Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization. *Journal of Hydrodynamics*, 32(2):234–246, 2020.

[61] F. Ren, C. Wang, and H. Tang. Bluff body uses deep-reinforcement-learning trained active flow control to achieve hydrodynamic stealth. *Physics of Fluids*, 33(9):093602, 2021.

[62] L. Rosafalco, J. M. De Ponti, L. Iorio, R. V. Craster, R. Ardito, and A. Corigliano. Reinforcement learning optimisation for graded metamaterial design using a physical-based constraint on the state representation and action space. *Scientific Reports*, 13(1):21836, 2023.

[63] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[64] A. Schmidt and B. Haasdonk. Data-driven surrogates of value functions and applications to feedback control for dynamical systems. *IFAC-PapersOnLine*, 51(2):307–312, 2018. 9th Vienna International Conference on Mathematical Modelling.

[65] T. Shah, L. Zhuo, P. Lai, A. De La Rosa-Moreno, F. Amirkulova, and P. Gerstoft. Reinforcement learning applied to metamaterial design. *The Journal of the Acoustical Society of America*, 150(1):321–338, 2021.

[66] C. Sinigaglia, F. Braghin, and S. Berman. Optimal Control of Velocity and Nonlocal Interactions in the Mean-Field Kuramoto Model. *Proceedings of the American Control Conference*, 2022-June:290–295, 2022.

[67] C. Sinigaglia, A. Manzoni, and F. Braghin. Density Control of Large-Scale Particles Swarm Through PDE-Constrained Optimization. *IEEE Transactions on Robotics*, 38(6):3530–3549, 2022.

[68] C. Sinigaglia, A. Manzoni, F. Braghin, and S. Berman. Robust optimal density control of robotic swarms. arXiv:2205.12592, 2022.

[69] C. Sinigaglia, D. E. Quadrelli, A. Manzoni, and F. Braghin. Fast active thermal cloaking through PDE-constrained optimization and reduced-order modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478(2258), 2022.

[70] L. Sirovich. Turbulence and the dynamics of coherent structures part i: Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987.

[71] R. S. Sutton and A. G. Barto. *Reinforcement Learning: an introduction*. The MIT Press, 2018.

[72] M. Tomasetto, A. Manzoni, and F. Braghin. Real-time optimal control of high-dimensional parametrized systems by deep learning-based reduced order models. arXiv:2409.05709, 2024.

[73] F. Tonti, J. Rabault, and R. Vinuesa. Navigation in a simplified urban flow through deep reinforcement learning. arXiv:2409.17922, 2024.

[74] P. Varela, P. Suárez, F. Alcántara-Ávila, A. Miró, J. Rabault, B. Font, L. M. García-Cuevas, O. Lehmkuhl, and R. Vinuesa. Deep reinforcement learning for flow control exploits different physics for increasing reynolds number regimes. *Actuators*, 11(12), 2022.

[75] J. Vasanth, J. Rabault, F. Alcántara-Ávila, M. Mortensen, and R. Vinuesa. Multi-agent reinforcement learning for the control of three-dimensional rayleigh-bénard convection. arXiv:2407.21565, 2024.

[76] S. Verma, G. Novati, and P. Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences of the United States of America*, 115(23):5849–5854, 2018.

[77] C. Vignon, J. Rabault, J. Vasanth, F. Alcántara-Ávila, M. Mortensen, and R. Vinuesa. Effective control of two-dimensional Rayleigh–Bénard convection: Invariant multi-agent reinforcement learning is all you need. *Physics of Fluids*, 35(6):065146, 2023.

[78] C. Vignon, J. Rabault, and R. Vinuesa. Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions. *Physics of Fluids*, 35(3), 2023.

[79] R. Vinuesa, O. Lehmkuhl, A. Lozano-Durán, and J. Rabault. Flow control in wings and discovery of novel approaches via deep reinforcement learning. *Fluids*, 7(2), 2022.

[80] Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley. Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports*, 9(1), 2019.

[81] N. Zolman, U. Fasel, J. N. Kutz, and S. L. Brunton. SINDy-RL: Interpretable and Efficient Model-Based Reinforcement Learning. arXiv:2403.09110, 2024.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

50/2025 Bonetti, S.; Botti, M.; Antonietti, P.F.
Conforming and discontinuous discretizations of non-isothermal Darcy–Forchheimer flows

49/2025 Zanin, A.; Pagani, S.; Corti, M.; Crepaldi, V.; Di Fede, G.; Antonietti, P.F.; the ADNI
Predicting Alzheimer's Disease Progression from Sparse Multimodal Data by NeuralODE Models

48/2025 Temellini, E.; Ballarin, F.; Chacon Rebollo, T.; Perotto, S.
On the inf-sup condition for Hierarchical Model reduction of the Stokes problem

47/2025 Gimenez Zapiola, A.; Consolo, A.; Amaldi, E.; Vantini, S.
Penalised Optimal Soft Trees for Functional Data

46/2025 Mirabella, S.; David, E.; Antona, A.; Stanghellini, C.; Ferro, N.; Matteucci, M.; Heuvelink, E.; Perotto, S.
On the Impact of Light Spectrum on Lettuce Biophysics: A Dynamic Growth Model for Vertical Farming

45/2025 Caliò, G.; Ragazzi, F.; Popoli, A.; Cristofolini, A.; Valdettaro, L; De Falco, C.; Barbante, F.
Hierarchical Multiscale Modeling of Positive Corona Discharges

44/2025 Brivio, S.; Fresca, S.; Manzoni, A.
Handling geometrical variability in nonlinear reduced order modeling through Continuous Geometry-Aware DL-ROM

43/2025 Tomasetto, M.; Manzoni, A.; Braghin, F.
Real-time optimal control of high-dimensional parametrized systems by deep-learning based reduced order models

42/2025 Franco, N. R.; Manzoni, A.; Zunino, P.; Hesthaven, J. S.
Deep orthogonal decomposition: a continuously adaptive neural network approach to model order reduction of parametrized partial differential equations

41/2025 Torzoni, M.; Maisto, D.; Manzoni, A.; Donnarumma, F.; Pezzulo, G.; Corigliano, A.
Active digital twins via active inference