Reinforcement Learning-Based Estimation for Partial Differential Equations

Anonymous authors
Paper under double-blind review

Abstract

In systems governed by nonlinear partial differential equations such as fluid flows, the design of state estimators such as Kalman filters relies on a reduced-order model (ROM) that projects the original high-dimensional dynamics onto a computationally tractable low-dimensional space. However, ROMs are prone to large errors, which negatively affects the performance of the estimator. Here, we introduce the reinforcement learning reduced-order estimator (RL-ROE), a ROM-based estimator in which the correction term that takes in the measurements is given by a nonlinear policy trained through reinforcement learning. The nonlinearity of the policy enables the RL-ROE to compensate efficiently for errors of the ROM, while still taking advantage of the imperfect knowledge of the dynamics. Using examples involving the Burgers and Navier-Stokes equations, we show that in the limit of very few sensors, the trained RL-ROE outperforms a Kalman filter designed using the same ROM. Moreover, it yields accurate high-dimensional state estimates for trajectories corresponding to various physical parameter values, without direct knowledge of the latter.

1 Introduction

Active control of turbulent flows has the potential to cut down emissions across a range of industries through drag reduction in aircrafts and ships or improved efficiency of heating and air-conditioning systems, among many other examples (Brunton & Noack, 2015). But real-time feedback control requires inferring the state of the system from sparse measurements using an algorithm called a state estimator, which typically relies on a model for the underlying dynamics (Simon, 2006). Among state estimators, the Kalman filter is by far the most well-known thanks to its optimality for linear systems, which has led to its widespread use in numerous applications (Kalman, 1960; Zarchan, 2005). However, continuous systems such as fluid flows are governed by partial differential equations (PDEs) which, when discretized, yield high-dimensional and oftentimes nonlinear dynamical models with hundreds or thousands of state variables. These high-dimensional models are too expensive to integrate with common state estimation techniques, especially in the context of embedded systems. Thus, state estimators for control are instead designed based on a reduced-order model (ROM) of the system, in which the underlying dynamics are projected to a low-dimensional subspace that is computationally tractable (Barbagallo et al., 2009; Rowley & Dawson, 2017).

A big challenge is that ROMs provide a simplified and imperfect description of the dynamics, which negatively affects the performance of the state estimator. One potential solution is to improve the accuracy of the ROM through the inclusion of additional closure terms (Ahmed et al., 2021). In this paper, we leave the ROM untouched and instead propose a new design paradigm for the estimator itself, which we call a reinforcement-learning reduced-order estimator (RL-ROE). The RL-ROE is constructed from the ROM in an analogous way to a Kalman filter, with the crucial difference that the linear filter gain function, which takes in the current measurement data, is replaced by a nonlinear policy trained through reinforcement learning (RL). The flexibility of the nonlinear policy, parameterized by a neural network, enables the RL-ROE to compensate for errors of the ROM while still taking advantage of the imperfect knowledge of the dynamics. Indeed, we show that in the limit of sparse measurements, the trained RL-ROE outperforms a Kalman filter designed using the same ROM and displays robust estimation performance across different dynamical regimes. To our knowledge, the RL-ROE is the first application of RL to state estimation of parametric PDEs.

2 General methodology

2.1 Problem formulation

Consider the parametric discrete-time nonlinear system given by

$$\boldsymbol{z}_k = \boldsymbol{f}(\boldsymbol{z}_{k-1}; \mu), \tag{1a}$$
$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{z}_k + \boldsymbol{n}_k, \tag{1b}$$

where $\boldsymbol{z}_k \in \mathbb{R}^n$ and $\boldsymbol{y}_k \in \mathbb{R}^p$ are respectively the state and measurement at time k, $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a time-invariant nonlinear map from previous to current state, $\boldsymbol{n}_k \in \mathbb{R}^p$ is observation noise (assumed zero unless stated otherwise), $\mu \in \mathbb{R}$ is a physical parameter, and $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ is a linear map from state to measurement. The measurements are typically acquired by a small number of sensors, hence they are sparse in the sense that $p \ll n$. Furthermore, we assume in this study that the dynamics given in (1) are obtained from a high-fidelity numerical discretization of a nonlinear partial differential equation (PDE) which requires a large number $n \gg 1$ of continuous state variables (on the order of at least a few hundreds), making the state $\boldsymbol{z}_k$ high-dimensional. Nonetheless, our work is applicable to any high-dimensional nonlinear system of the form (1). We do not account for exogenous control inputs to the system, which is left for future work.

Here, we will focus on the post-transient dynamics of (1); these are the observed dynamics once the transients associated with the initial condition have died down. In particular, we consider systems whose post-transient dynamics are described by an attractor that is either a steady state, a periodic or quasi-periodic limit cycle, or a chaotic attractor, which encompasses the behavior of a large class of physical systems. The nature of the attractor is independent of the initial condition but depends on the value of μ, considered to be in a range $[\mu_1, \mu_2]$.

The purpose of the present work is to combine reduced-order modeling (ROM) and reinforcement learning (RL) to construct a state estimator that solves the following problem: given at every time k the history of measurements $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\}$ from a post-transient trajectory of (1), compute an online (real-time) estimate $\hat{\boldsymbol{z}}_k$ of the hidden state $\boldsymbol{z}_k$ without knowing the parameter value μ or the initial state $\boldsymbol{z}_0$. We emphasize that this is a different setting from the forecasting problem solved by operator learning methods, where μ and $\boldsymbol{z}_0$ are both known (Lu et al., 2021; Kovachki et al., 2023). In general, such a state estimator takes the form

$$\hat{\boldsymbol{z}}_k = \mathcal{E}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k), \tag{2}$$

where $\mathcal{E}$ is the map from measurements to state estimate that we seek. During the offline formulation and training of the estimator $\mathcal{E}$, we assume that we have access to a training dataset $\boldsymbol{Z}_{\text{train}} = \{\boldsymbol{Z}^\mu, \boldsymbol{Y}^\mu\}_{\mu\in S}$ for various values of μ belonging to a finite set $S \subset [\mu_1, \mu_2]$. Specifically, for each $\mu \in S$, $\boldsymbol{Z}^\mu = \{\boldsymbol{z}_0^\mu, \ldots, \boldsymbol{z}_K^\mu\}$ is a trajectory of (1a) and $\boldsymbol{Y}^\mu = \{\boldsymbol{y}_0^\mu, \ldots, \boldsymbol{y}_K^\mu\}$ contains the corresponding measurements given by (1b). Once trained, the estimator $\mathcal{E}$ can be deployed online to produce state estimates $\hat{\boldsymbol{z}}_k$ from sensor measurements $\boldsymbol{y}_k$, *without knowledge of the true initial state $\boldsymbol{z}_0$ or the parameter value μ.*

We propose in this paper to construct and train the estimator $\mathcal{E}$ by combining reduced-order modeling (ROM) with reinforcement learning (RL). The ROM procedure, which follows standard practices, is described in Section 2.2. The formulation of $\mathcal{E}$ based on the ROM and its training using RL, which constitutes the main novelty of the paper, is described in Section 2.3.

Note that the supervised setting considered here, in which a dataset of states and observations is available, is common in data-driven estimation (Greenberg et al., 2024). Such data is indeed available in many applications, either through offline simulations of the high-dimensional system (1) or by advanced visualization techniques such as particle image velocimetry in fluid mechanics (Adrian & Westerweel, 2011). Once the data is acquired, the offline formulation and training of $\mathcal{E}$ does not require knowledge of the system (1).

2.2 Reduced-order model

Since the high dimensionality of (1) renders online estimation impractical, state estimators $\mathcal{E}$ are typically formulated based on a reduced-order model (ROM) of the dynamics (Rowley & Dawson, 2017). A popular

approach to construct a ROM is first to choose a suitable linearly independent set of modes $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$, where $\boldsymbol{u}_i \in \mathbb{R}^n$, defining an r-dimensional subspace of $\mathbb{R}^n$ within which most of the states $\boldsymbol{z}_k$ are assumed to belong. Stacking these modes as columns of a matrix $\boldsymbol{U} \in \mathbb{R}^{n \times r}$, one can then approximate the high-dimensional state as $\boldsymbol{z}_k \simeq \boldsymbol{U}\boldsymbol{x}_k$, where the reduced-order state $\boldsymbol{x}_k \in \mathbb{R}^r$ represents the coordinates of $\boldsymbol{z}_k$ in the subspace. Finally, one finds a ROM for the dynamics of $\boldsymbol{x}_k$, which is vastly cheaper to evolve than (1) when $r \ll n$.

There exist various ways to find an appropriate set of modes $\boldsymbol{U}$ and corresponding ROM for the dynamics of $\boldsymbol{x}_k$ (Taira et al., 2017). In this work, we employ the Dynamic Mode Decomposition (DMD), a purely data-driven algorithm that has found numerous applications in fields ranging from fluid dynamics to neuroscience (Schmid, 2010; Kutz et al., 2016). Importantly, we seek a single ROM to describe dynamics corresponding to various parameter values $\mu \in [\mu_1, \mu_2]$ since the state estimator that we will later construct based on this ROM does not have knowledge of μ.

We apply DMD to the state trajectories $\{\boldsymbol{Z}^\mu\}_{\mu \in S}$ contained in the training dataset $\boldsymbol{Z}_{\text{train}}$. The DMD seeks a best-fit linear model of the dynamics in the form of a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ such that $\boldsymbol{z}^\mu_{k+1} \simeq \boldsymbol{A}\boldsymbol{z}^\mu_k$ for all k and μ, and computes the modes $\boldsymbol{U}$ as the r leading principal component analysis (PCA) modes of $\boldsymbol{Z}_{\text{train}}$. The transformation $\boldsymbol{z}_k = \boldsymbol{U}\boldsymbol{x}_k$ and the orthogonality of $\boldsymbol{U}$ then yield the linear discrete-time ROM

$$\boldsymbol{x}_k = \boldsymbol{A}_r \boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}, \tag{3a}$$
$$\boldsymbol{y}_k = \boldsymbol{C}_r \boldsymbol{x}_k + \boldsymbol{v}_k, \tag{3b}$$
$$\boldsymbol{z}_k = \boldsymbol{U}\boldsymbol{x}_k, \tag{3c}$$

where $\boldsymbol{A}_r = \boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U} \in \mathbb{R}^{r \times r}$ and $\boldsymbol{C}_r = \boldsymbol{C}\boldsymbol{U} \in \mathbb{R}^{p \times r}$ are the reduced-order state-transition and observation models, respectively. The (unknown) non-Gaussian process noise $\boldsymbol{w}_k$ and observation noise $\boldsymbol{v}_k$ account for the neglected PCA modes of $\boldsymbol{Z}_{\text{train}}$ in $\boldsymbol{U}$, as well as the error incurred by the linear approximation and effective averaging of the dynamics over a range of μ. Additional details regarding the calculation of $\boldsymbol{A}_r$ and $\boldsymbol{U}$ are provided in Appendix A of the supplementary materials.

2.3 Reinforcement learning-based reduced-order estimator

Using the ROM (3), we can now formulate the state estimator $\mathcal{E}$ defined in Section 2.1. The reduced-order estimator (ROE) that we propose takes the recursive form

$$\hat{\boldsymbol{x}}_k = \boldsymbol{A}_r \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{a}_k, \tag{4a}$$
$$\boldsymbol{a}_k \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}(\,\cdot\,|\boldsymbol{y}_k, \hat{\boldsymbol{x}}_{k-1}), \tag{4b}$$
$$\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k, \tag{4c}$$

where $\hat{\boldsymbol{x}}_k$ is an estimate of the reduced-order state $\boldsymbol{x}_k$ and $\boldsymbol{a}_k \in \mathbb{R}^r$ is an action sampled from a nonlinear and stochastic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, which takes as input the current measurement $\boldsymbol{y}_k$ and the previous state estimate $\hat{\boldsymbol{x}}_{k-1}$. The subscript $\boldsymbol{\theta}$ denotes the set of parameters that define the policy, whose goal is to use the sparse measurements $\boldsymbol{y}_k$ to act on the dynamics of $\hat{\boldsymbol{x}}_k$ in (4a) so that the high-dimensional state estimate $\hat{\boldsymbol{z}}_k$ converges (online) towards the hidden true state $\boldsymbol{z}_k$ from any initial estimate $\hat{\boldsymbol{x}}_0$. Note that designing state estimators, also called state observers, by correcting the dynamics model with a measurement-dependent term is a standard approach in control theory (Korovin & Fomichev, 2009; Besançon, 2007). We will consider two different versions of the ROE; one in which the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is parameterized by a multi-layer perceptron (MLP) feedforward network, and one in which it is parameterized by a long-short term memory (LSTM) recurrent network. The difference between the two lies in the internal memory of the LSTM, which allows the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ to depend implicitly on the entire history of past measurements $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k-1}\}$ in addition to the current measurement $\boldsymbol{y}_k$.

A Kalman filter is a special case of such an estimator, for which the action in (4b) is given by

$$\boldsymbol{a}_k = \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{C}_r \boldsymbol{A}_r \hat{\boldsymbol{x}}_{k-1}), \tag{5}$$

with $\boldsymbol{K}_k \in \mathbb{R}^{r \times p}$ the optimal Kalman gain. Although the Kalman filter is the optimal *linear* filter (Julier & Uhlmann, 2004; Simon, 2006), its performance suffers in the presence of unmodeled dynamics and parameter

uncertainty, both of which are present in our case. Thus, this motivates the adoption of the more general form (4b), which retains the dependence of $\boldsymbol{a}_k$ on $\boldsymbol{y}_k$ and $\hat{\boldsymbol{x}}_{k-1}$ but is more flexible thanks to the nonlinearity of the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$.

We first train the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ in an offline phase, using deep RL to solve the optimization problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}\left[\lim_{K\to\infty} \frac{1}{K}\sum_{k=1}^{K} \|\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k\|^2 + \lambda\|\boldsymbol{a}_k\|^2\right] \text{ subject to (1) and (4)}, \tag{6}$$

where the expectation is taken over initial estimates $\hat{\boldsymbol{x}}_0$, initial true states $\boldsymbol{z}_0$, parameters μ, trajectories of state estimates $\{\hat{\boldsymbol{z}}_1, \hat{\boldsymbol{z}}_2, \dots\}$ induced by $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ through (4), and trajectories of true states $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots\}$ and corresponding measurements $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots\}$ induced by (1). In practice, the optimization problem is solved offline using the parameters μ and true trajectories contained in the training dataset $\boldsymbol{Z}_{\text{train}}$. The first squared term in (6) penalizes the error between the high-dimensional estimate $\hat{\boldsymbol{z}}_k$ and the true state $\boldsymbol{z}_k$. The second squared term favors smaller values for the action $\boldsymbol{a}_k$, which acts as a regularization mechanism. Unless indicated otherwise, we will consider $\lambda = 0$. By considering different values of μ during training, a strategy called domain randomization (Peng et al., 2018b), we ensure robustness of the policy with respect to μ during online deployment of the estimator. Note that the stochasticity of $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ lets the RL algorithm explore different actions during the training process, but is turned off during online deployment.

We call the estimator constructed and trained through this process an RL-trained ROE, or RL-ROE for short. Finally, an interpretation of the estimator dynamics (4) and the optimization problem (6) in the context of Bayesian inference is presented in Appendix B.

2.4 Summary of the proposed methodology

In summary, the methodology we propose consists of the following three steps, illustrated in Figure 1. The first two steps are carried out offline using the training dataset $\boldsymbol{Z}_{\text{train}}$ containing snapshots of the state $\boldsymbol{z}_k$ and measurement $\boldsymbol{y}_k$ from multiple solution trajectories of (1) for $\mu \in S$. The third takes place online using sole knowledge of measurements $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_k\}$ from a trajectory of (1) with unknown μ, not necessarily belonging to S, and unknown initial state $\boldsymbol{z}_0$.

1. **Construction of the ROM (offline)**. A ROM of the form (3) is obtained by applying the DMD to the training dataset $\boldsymbol{Z}_{\text{train}}$.
2. **Training of the RL-ROE (offline)**. An RL-ROE of the form (4) is designed based upon the ROM constructed in Step 1, and its policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is trained using the ground-truth trajectories contained in $\boldsymbol{Z}_{\text{train}}$ (see Section 3 for details on the training procedure).
3. **Deployment of the RL-ROE (online)**. Using measurements $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_k\}$ from a trajectory of (1) corresponding to unknown μ and unknown initial state $\boldsymbol{z}_0$, the trained RL-ROE computes a real-time estimate $\hat{\boldsymbol{z}}_k$ of the hidden state $\boldsymbol{z}_k$.

The RL-ROE, which combines a ROM with a nonlinear policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ trained in an RL framework to solve the estimation problem, constitutes the main contribution of the present paper. The training procedure for $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, which involves a non-trivial reformulation of the time-varying estimation problem into a stationary Markov decision process, is described in the next section.

3 Offline training methodology

In order to train $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ with reinforcement learning, we need to formulate the optimization problem (6) as a stationary Markov decision process (MDP). However, this is no trivial task given that the aim of the policy is to minimize the error between the state estimate $\hat{\boldsymbol{z}}_k$ and a time-dependent ground-truth state $\boldsymbol{z}_k$. At first sight, such trajectory tracking problem requires a time-dependent reward function and, therefore, a non-stationary MDP (Puterman, 2014; Lecarpentier & Rachelson, 2019). To be able to use off-the-shelf RL

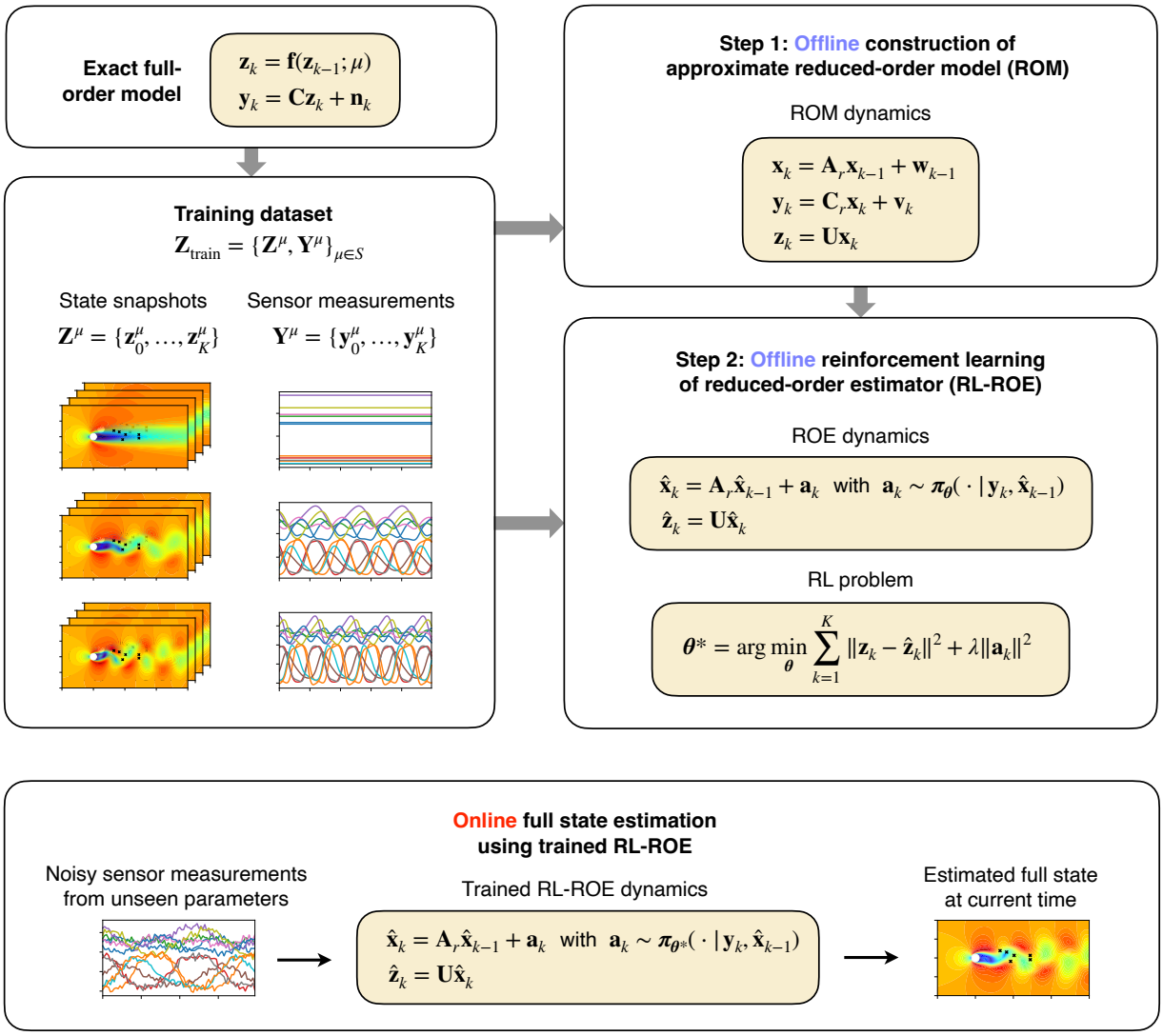


Figure 1: Overview of the proposed RL-ROE methodology.

algorithms, we introduce a trick to translate this non-stationary MDP to an equivalent stationary MDP based on an extended state. Specifically, we show hereafter that the problem can be framed as a stationary MDP whose state has been enhanced with $\boldsymbol{z}_k$ and μ, removing the time dependence from the reward function.

Letting $\boldsymbol{s}_k = (\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{z}_k, \mu) \in \mathbb{R}^{r+n+1}$ denote an augmented state at time k, we can define an MDP consisting of the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S} = \mathbb{R}^{r+n+1}$ is the augmented state space, $\mathcal{A} \subset \mathbb{R}^r$ is the action space, $\mathcal{P}(\cdot|\boldsymbol{s}_k, \boldsymbol{a}_k)$ is a transition probability, and $\mathcal{R}(\boldsymbol{s}_k, \boldsymbol{a}_k, \boldsymbol{s}_{k+1})$ is a reward function. At each time step k, the agent selects an action $\boldsymbol{a}_k \in \mathcal{A}$ according to the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ defined in (4b), which can be expressed as

$$\boldsymbol{a}_k \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}(\cdot \,|\boldsymbol{o}_k), \tag{7}$$

where $\boldsymbol{o}_k = (\boldsymbol{y}_k, \hat{\boldsymbol{x}}_{k-1})$ is a partial observation of the current state $\boldsymbol{s}_k$, since $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{z}_k$. The state $\boldsymbol{s}_{k+1} = (\hat{\boldsymbol{x}}_k, \boldsymbol{z}_{k+1}, \mu)$ at the next time step is then obtained from equations (1a) and (4a) as

$$\boldsymbol{s}_{k+1} = (\boldsymbol{A}_r \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{a}_k, \boldsymbol{f}(\boldsymbol{z}_k; \mu), \mu), \tag{8}$$

which defines the transition model $\boldsymbol{s}_{k+1} \sim \mathcal{P}(\cdot|\boldsymbol{s}_k, \boldsymbol{a}_k)$. Finally, the agent receives the reward

$$r_k = \mathcal{R}(\boldsymbol{s}_k, \boldsymbol{a}_k, \boldsymbol{s}_{k+1}) = -\|\boldsymbol{z}_k - \boldsymbol{U}\hat{\boldsymbol{x}}_k\|^2 - \lambda \|\boldsymbol{a}_k\|^2, \tag{9}$$

which is minus the term to be minimized at each step in (6). Thanks to the incorporation of $\boldsymbol{z}_k$ into $\boldsymbol{s}_k$, the reward function (9) has no explicit time dependence and the MDP is therefore stationary.

The RL training process then finds the optimal policy parameters

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\tau\sim\boldsymbol{\pi}_{\boldsymbol{\theta}}}[R(\tau)], \tag{10}$$

where the expectation is over trajectories $\tau = (\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{s}_2, \boldsymbol{a}_2, \dots)$, and $R(\tau) = \sum_{k=1}^{K_{\text{train}}} r_k$ is the finite-horizon undiscounted return, with K_{train} the length of each trajectory. Thus, the optimization problem (10) solved by RL is equivalent to that stated in (6). Formally, at the beginning of every episode, the environment is reset according to the distributions

$$\hat{\boldsymbol{x}}_0 \sim p_{\hat{\boldsymbol{x}}_0}(\cdot), \quad \boldsymbol{z}_0 \sim p_{\boldsymbol{z}_0}(\cdot), \quad \mu \sim p_{\mu}(\cdot), \tag{11}$$

leading to the augmented state $\boldsymbol{s}_1 = (\hat{\boldsymbol{x}}_0, \boldsymbol{z}_1, \mu) = (\hat{\boldsymbol{x}}_0, \boldsymbol{f}(\boldsymbol{z}_0; \mu), \mu)$, which constitutes the start of the agent-environment interactions. The distribution $p_{\hat{\boldsymbol{x}}_0}(\cdot)$ bestows robustness of the learned policy with respect to the initial state estimate $\hat{\boldsymbol{x}}_0$, while the distributions $p_{\boldsymbol{z}_0}(\cdot)$ and $p_{\mu}(\cdot)$ enable the same policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ to learn from multiple ground-truth trajectories of (1) corresponding to various parameters μ and initial true states $\boldsymbol{z}_0$. In practice, when advancing the MDP state from $\boldsymbol{s}_k$ to $\boldsymbol{s}_{k+1}$ during training rollouts, we do not use the high-dimensional dynamics (1) to compute $\boldsymbol{z}_{k+1}$ as indicated in (8), since that would be prohibitively expensive. Instead, we utilize the ground-truth trajectories $\boldsymbol{Z}^{\mu} = \{\boldsymbol{z}_0^{\mu}, \dots, \boldsymbol{z}_K^{\mu}\}$ contained in the training dataset $\boldsymbol{Z}_{\text{train}}$. Thus, at the beginning of every episode, we reset the environment by drawing a random μ from S and we initialize $\boldsymbol{z}_0$ by randomly choosing a state among $\{\boldsymbol{z}_0^{\mu}, \dots, \boldsymbol{z}_{K_{\text{start}}}^{\mu}\}$, where K_{start} is a user-defined hyperparameter that needs to satisfy $K_{\text{start}} + K_{\text{train}} \leq K$. In this way, we can simply take $\boldsymbol{z}_{k+1}$ from $\boldsymbol{Z}^{\mu}$ when advancing the MDP state from $\boldsymbol{s}_k$ to $\boldsymbol{s}_{k+1}$ during training rollouts. Finally, we set the initial estimate $\hat{\boldsymbol{x}}_0$ to $\boldsymbol{0}$, which we will also do during online deployment of the trained RL-ROE.

Since the policy (7) is conditioned on a partial observation $\boldsymbol{o}_k$ of the state $\boldsymbol{s}_k$, the stationary MDP that we have defined is, in fact, a partially observable MDP (POMDP). In this case, it is known that the globally optimal policy depends on a summary of the history of past observations and actions, $\boldsymbol{h}_k = \{\boldsymbol{o}_1, \boldsymbol{a}_1, \dots, \boldsymbol{o}_k\}$, rather than just the current observation $\boldsymbol{o}_k$ (Kaelbling et al., 1998; Ni et al., 2022). Nonetheless, policies formulated based on an incomplete summary of $\boldsymbol{h}_k$ are also common in practice and still achieve good results (Sutton & Barto, 2018). We pursue both approaches in the present paper through two alternative parameterizations of the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, using either an MLP or LSTM. In the MLP case, the policy simply depends on the current observation $\boldsymbol{o}_k = (\boldsymbol{y}_k, \hat{\boldsymbol{x}}_{k-1})$, while in the LSTM case, the policy depends on $\boldsymbol{o}_k$ along with an internal state vector that summarizes the entire history $\boldsymbol{h}_k$.

To learn $\boldsymbol{\theta}^*$, we employ the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017), which belongs to the class of policy gradient methods (Sutton et al., 2000). These methods do not require the Markov property of the state (that is, conditional independence of future states on past states given the present state) and can therefore be readily applied to the POMDP setting. For our problem, this guarantees that the PPO algorithm will converge to a locally optimum policy. The MLP and LSTM parameterizations of the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, implementation details and training hyperparameters are discussed in Appendix D.

4 Results

We evaluate the state estimation performance of the RL-ROE for systems governed by the Burgers equation and Navier-Stokes equations. For each system, we first compute various solution trajectories corresponding to different physical parameter values, which we use to construct the ROM and train the RL-ROE following the procedure outlined in Section 2.4. The trained RL-ROE is finally deployed online and compared against a time-dependent Kalman filter constructed from the same ROM, which we refer to as KF-ROE. The KF-ROE is given by equations (4a) and (5), with the calculation of the time-varying Kalman gain detailed in Appendix C of the supplementary materials.

Before proceeding to the results, we discuss our choice of baseline. The ensemble Kalman filter and 4D-Var are two estimation techniques for high-dimensional systems such as those governed by PDEs (Lorenc, 2003).

Although they are commonly employed for data assimilation in numerical weather prediction, they require large computational resources since they involve repeated solutions of the high-dimensional dynamics (1). Thus, they are not applicable in the context of embedded control systems, whose limited resources call for an inexpensive model such as the ROM (3). Since the ROM that we consider has linear dynamics, extensions of the Kalman filter for nonlinear dynamics such as the extended or unscented Kalman filters (Wan & Van Der Merwe, 2000; Julier & Uhlmann, 2004) are not relevant, and the vanilla Kalman filter remains the best choice of baseline.

4.1 Burgers equation

The forced Burgers equation is a prototypical nonlinear hyperbolic PDE that takes the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = f(x,t), \tag{12}$$

where $u(x,t)$ is the velocity at position $x \in [0, L]$ and time t, $f(x,t)$ is a distributed time-dependent forcing, and the scalar ν acts like a viscosity. Here, we choose a forcing of the form

$$f(x,t) = 2\sin(\omega t - kx) + 2\sin(3\omega t - kx) + 2\sin(5\omega t - kx), \tag{13}$$

where $k = 2\pi/L$, and we let ν and ω be related through a scalar parameter $\mu \in [0, 1]$ as follows:

$$\nu = \nu_1 + (\nu_2 - \nu_1)\mu, \qquad \omega = \omega_1 + (\omega_2 - \omega_1)\mu. \tag{14a}$$

Thus, μ can be regarded as a physical parameter that affects the dynamics of the forced Burgers equation through both ν and ω. We consider periodic boundary conditions and choose $L = 1$, $\nu_1 = 0.01$, $\nu_2 = 0.1$, $\omega_1 = 0.2\pi$, $\omega_1 = 0.4\pi$. Finally, we define a small number p of equally-spaced sensors along $[0, L]$ that measure the value of u at the corresponding locations.

We solve the forced Burgers equation using a spectral method with $n = 256$ Fourier modes and a fifth-order Runge-Kutta time integration scheme. We define the discrete-time state vector $\boldsymbol{z}_k \in \mathbb{R}^n$ that contains the values of u at n equally-spaced collocation points and at discrete time steps $t = k\Delta t$, where $\Delta t = 0.05$. To generate the training dataset $\boldsymbol{Z}_{\text{train}} = \{\boldsymbol{Z}^\mu, \boldsymbol{Y}^\mu\}_{\mu \in S}$ used for constructing the ROM and training the RL-ROE, we compute solutions of the Burgers equation corresponding to $\mu \in S = \{0, 0.1, 0.2, \ldots, 1\}$. For each μ, we discard the transient portion of the dynamics and save 401 snapshots $\boldsymbol{Z}^\mu = \{\boldsymbol{z}_0^\mu, \ldots, \boldsymbol{z}_{400}^\mu\}$ in the post-transient regime, as well as corresponding measurements $\boldsymbol{Y}^\mu = \{\boldsymbol{y}_0^\mu, \ldots, \boldsymbol{y}_{400}^\mu\}$ obtained from the p sensors (Appendix E describes the corresponding $\boldsymbol{C}$). We retain $r = 10$ modes when constructing the ROM, corresponding to an-order-of-magnitude reduction in the dimensionality of the system. We train the RL-ROE using $K_{\text{start}} = 200$ and episodes of length $K_{\text{train}} = 200$ steps to make full use of the trajectories stored in $\boldsymbol{Z}_{\text{train}}$, and we end the training process when the return no longer increases on average. The RL hyperparameters and learning curve displaying the performance improvement of the RL-ROE during the training process are reported in Appendix D.

The trained RL-ROE and the KF-ROE are now compared based on their ability to track trajectories of $\boldsymbol{z}_k$ corresponding to various unknown μ using sparse measurements from the p equally-spaced sensors. For each μ, the evaluation is carried out using 5 ground-truth trajectories corresponding to randomly-sampled initial conditions $\boldsymbol{z}_0$ in the post-transient regime, and the reduced estimate is always initialized as $\hat{\boldsymbol{x}}_0 = \boldsymbol{0}$. Beginning with $p = 4$ sensors, Figure 2 reports the mean (lines) and standard deviation (shaded areas) of the normalized L_2 error for 3 values of μ not present in the training dataset $\boldsymbol{Z}_{\text{train}}$. The normalized L_2 error is defined as $\|\hat{\boldsymbol{z}}_k - \boldsymbol{z}_k\|/\|\boldsymbol{z}_k\|$, where $\boldsymbol{z}_k$ is the (hidden) true state and $\hat{\boldsymbol{z}}_k$ is the corresponding estimate given by the RL-ROE or KF-ROE. The error of the RL-ROE, using either the MLP or LSTM policy, rapidly decreases to values close to the lower bound, which is the error incurred by projecting the true state $\boldsymbol{z}_k$ to the modes $\boldsymbol{U}$, i.e. the lowest possible error achievable by an ROE based on $\boldsymbol{U}$. Spatio-temporal contours of the ground-truth solutions for the same 3 values of μ and the corresponding RL-ROE and KF-ROE estimates are shown in Figure 3. The RL-ROE using either policy vastly outperforms the KF-ROE, which demonstrates the superiority of a nonlinear correction to the estimator dynamics (4a). We emphasize that the noise covariances in the KF-ROE were tuned using line search to obtain the best possible results (see Appendix C for details on the tuning process).

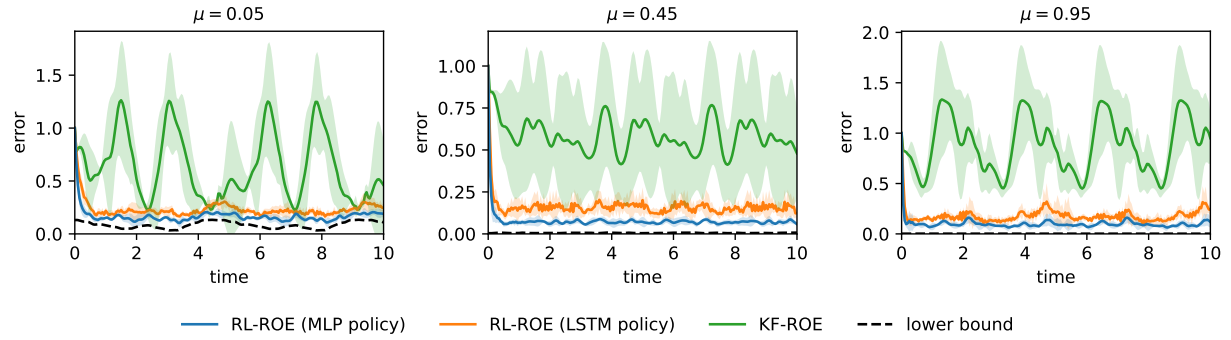


Figure 2: Burgers equation with $p = 4$ sensors. Normalized L_2 error of the RL-ROE and KF-ROE for the estimation of trajectories corresponding to values of μ not seen during training. The error is averaged over 5 trajectories with randomly-sampled initial true states $\boldsymbol{z}_0$. The shaded areas denote the standard deviation.

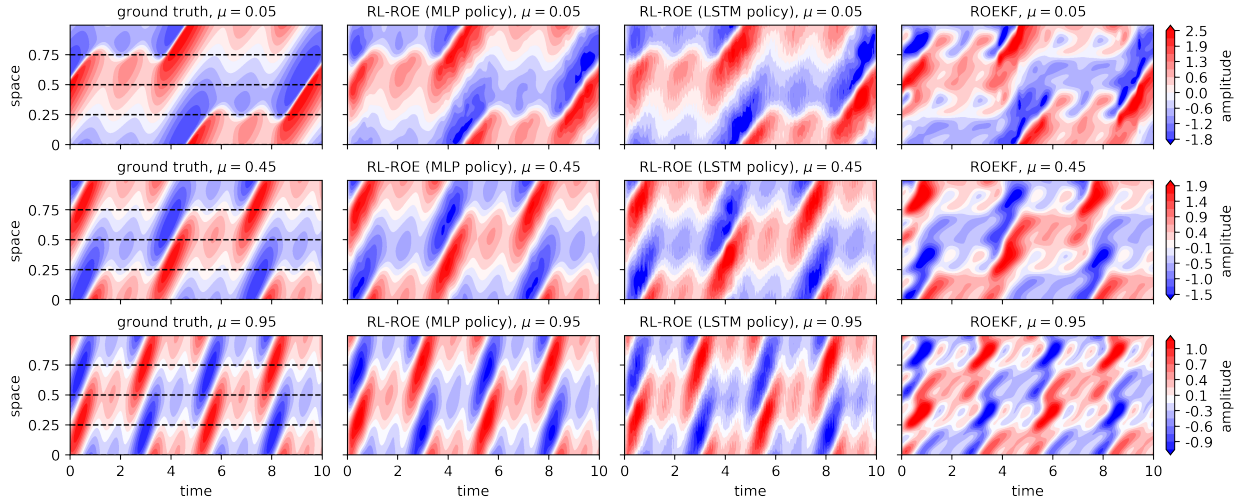


Figure 3: Burgers equation with $p = 4$ sensors. Ground-truth trajectories for values of μ not seen during training and corresponding RL-ROE and KF-ROE estimates. The dashed lines on the ground-truth trajectory plots indicate the sensor data seen by the RL-ROE and KF-ROE.

Figure 4 (left) reports the time average of the normalized L_2 error as a function of μ for $p = 4$. The RL-ROE exhibits robust performance across the entire parameter range $\mu \in [0, 1]$, including when estimating trajectories corresponding to previously unseen parameter values. Finally, Figure 4 (right) displays the average over time and over μ of the normalized L_2 error for varying number p of sensors. Note that each value of p corresponds to a separately trained RL-ROE. As the number of sensors increases, the KF-ROE performs better and better until its accuracy overtakes that of the RL-ROE. We hypothesize that the accuracy of the RL-ROE is limited by the inability of the RL training process to find an optimal policy, due to both the non-convexity of the optimization landscape as well as shortcomings inherent to current deep RL algorithms. This might also explain the slightly lower performance of the RL-ROE using the LSTM policy, which is more complex to train than the MLP policy. Even so, the strength of the nonlinear policy of the RL-ROE becomes very clear in the very sparse sensing regime; its performance remains remarkably robust as the number of sensors reduces to 2 or even 1. In Appendix F, spatio-temporal contours (similar as in Figure 3) of the ground-truth solution and corresponding estimates for $p = 2$ and 12 illustrate that the slight advantage held by the KF-ROE for $p = 12$ is reversed into clear superiority of the RL-ROE for $p = 4$.

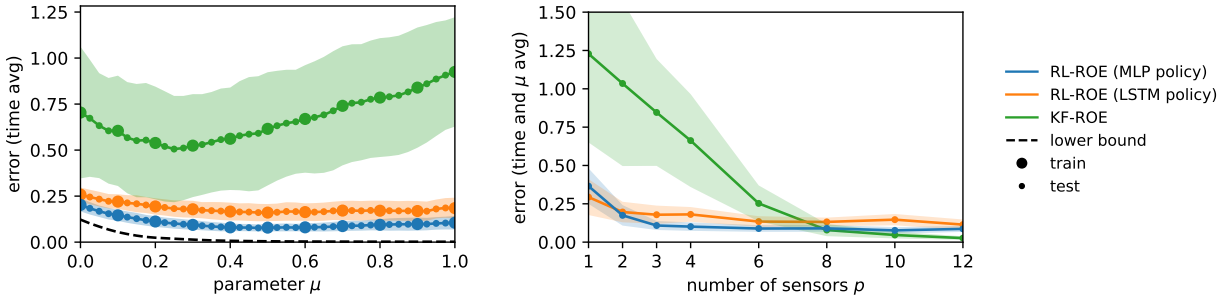


Figure 4: Burgers equation. Left: Normalized L_2 error, averaged over time, versus μ when using $p = 4$ sensors. Values of μ belonging to the training set S are shown by large circles while the testing values are displayed by small circles. The error is averaged over 5 trajectories with randomly-sampled initial true states $\boldsymbol{z}_0$. The shaded areas denote the standard deviation. Right: Normalized L_2 error, averaged over time and over the testing values of μ, versus number of sensors p.

4.2 Navier-Stokes equations: flow past a cylinder

The Navier-Stokes equations are a set of nonlinear PDEs that describe the motion of fluids flows. For incompressible fluids, the Navier-Stokes equations take the form

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \frac{1}{Re}\Delta \boldsymbol{u}, \tag{15a}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{15b}$$

where $\boldsymbol{u}(\boldsymbol{x}, t)$ and $p(\boldsymbol{x}, t)$ are the velocity vector and pressure at position $\boldsymbol{x}$ and time t, and the scalar Re is the Reynolds number. In this example, we consider the classical problem of a flow past a cylinder in a 2D domain, which is well known to exhibit a Hopf bifurcation from a steady wake to periodic vortex shedding at a critical Reynolds number $Re_c \sim 40$ (Jackson, 1987). For our study, we focus on the range $Re \in [10, 110]$, which makes the estimation problem very challenging since this range includes the bifurcation and therefore comprises solution trajectories with very different dynamics – steady for $Re < Re_c$, periodic limit cycle for $Re > Re_c$. Furthermore, the shedding frequency and spacing between consecutive vortices in the limit cycle regime both vary with Re. For the estimation problem, we define a sparse sensor array measuring the two components of $\boldsymbol{u}$ at p randomly-distributed locations in a box of size 10×5 cylinder diameters, situated downstream of the cylinder at a distance of 5 cylinder diameters.

We solve the Navier-Stokes equations with the open source finite volume code OpenFOAM using a mesh consisting of 18840 nodes and a second-order implicit scheme with time step 0.05. The discrete-time state vector $\boldsymbol{z}_k \in \mathbb{R}^{37680}$ contains the two velocity components of $\boldsymbol{u}$ at discrete time steps $t = k\Delta t$, where we choose $\Delta t = 0.25$. To generate the training dataset $\boldsymbol{Z}_{\text{train}} = \{\boldsymbol{Z}^{Re}, \boldsymbol{Y}^{Re}\}_{Re \in S}$ for constructing the ROM and training the RL-ROE, we run simulations of the Navier-Stokes equations for $Re \in S = \{10, 20, 30, \ldots, 110\}$. We discard the transient portion of the dynamics (for the cases $Re > Re_c$) and save 201 snapshots $\boldsymbol{Z}^{Re} = \{\boldsymbol{z}_0^{Re}, \ldots, \boldsymbol{z}_{200}^{Re}\}$ in the post-transient regime, as well as corresponding measurements $\boldsymbol{Y}^{Re} = \{\boldsymbol{y}_0^{Re}, \ldots, \boldsymbol{y}_{200}^{Re}\}$ obtained from the p sensors (Appendix E describes the corresponding $\boldsymbol{C}$). We retain $r = 20$ modes when constructing the ROM, corresponding to a three-orders-of-magnitude reduction in the dimensionality of the system. These modes are shown in Appendix G. We train the RL-ROE using $K_{\text{start}} = 200$ and episodes of length $K = 200$ steps to make full use of the trajectories stored in $\boldsymbol{Z}_{\text{train}}$. We end the training process when the return no longer increases on average. The RL hyperparameters and learning curve displaying the performance improvement of the RL-ROE during the training process are reported in Appendix D.

The trained RL-ROE and the KF-ROE are now compared based on their ability to track trajectories of $\boldsymbol{z}_k$ corresponding to various unknown Re using sparse measurements from the p sensors randomly distributed in the wake of the cylinder. For each Re, the evaluation is carried out using 5 ground-truth trajectories corresponding to randomly-sampled initial conditions $\boldsymbol{z}_0$ in the post-transient regime, and the reduced estimate is always initialized as $\hat{\boldsymbol{x}}_0 = \boldsymbol{0}$. Beginning with $p = 3$ sensors, Figure 5 reports the mean (lines) and standard

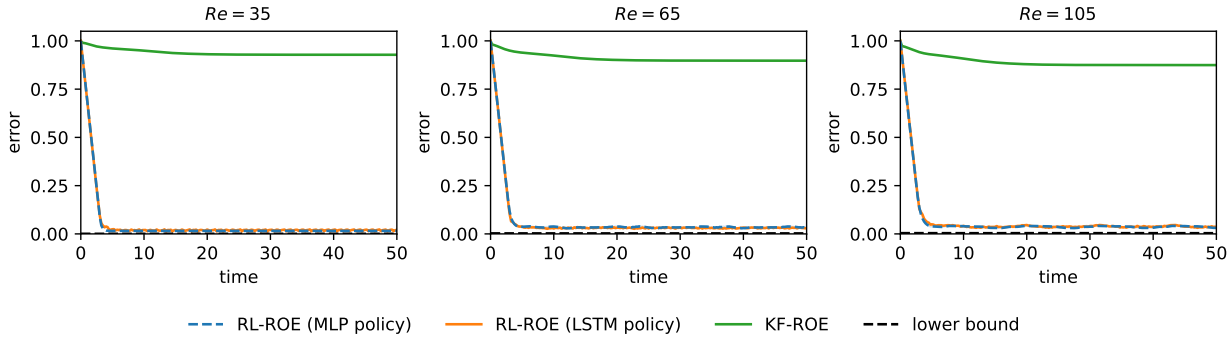


Figure 5: Flow past a cylinder with $p = 3$ sensors. Normalized L_2 error of the RL-ROE and KF-ROE for the estimation of trajectories corresponding to values of Re not seen during training. The error is averaged over 5 trajectories with randomly-sampled initial true states $\boldsymbol{z}_0$. The shaded areas denote the standard deviation.

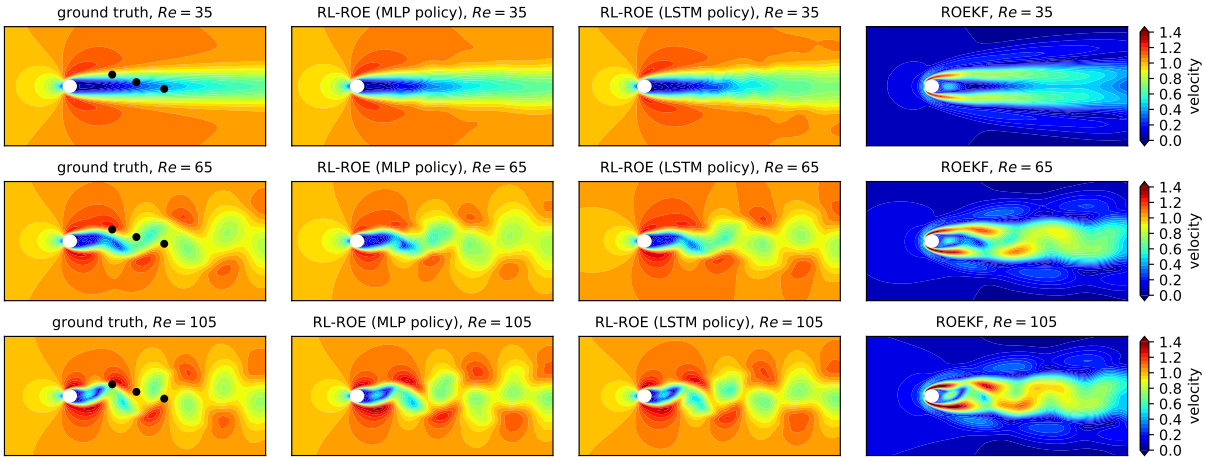


Figure 6: Flow past a cylinder with $p = 3$ sensors. Velocity magnitude at $t = 50$ of the ground-truth trajectories for values of Re not seen during training and corresponding RL-ROE and KF-ROE estimates. The black crosses in the contours of the reference solutions indicate the sensor locations.

deviation (shaded areas) of the normalized L_2 error for 3 values of Re not present in the training dataset $\boldsymbol{Z}_{\text{train}}$. The error of the RL-ROE, using either the MLP or LSTM policy, rapidly reaches a value close to the lower bound, while that of the KF-ROE remains around 1. The ground-truth velocity magnitude at $t = 50$ for the same 3 values of Re and the corresponding RL-ROE and KF-ROE estimates are shown in Figure 6. Remarkably, the RL-ROE with either policy manages to estimate very precisely the entire flow field across different dynamical regimes, with the steady wake at $Re = 35$ being reproduced equally well as the wake of vortices at $Re = 65$ and 105. The KF-ROE, on the other hand, struggles to estimate the flow fields for all 3 Reynolds numbers, and instead predicts a velocity field that is almost everywhere zero except in the wake. Again, the superiority of the RL-ROE is granted by the nonlinearity of its policy – in fact, bifurcations such as the one exhibited by this flow are inherently nonlinear phenomena. Appendix H shows corresponding results in the presence of non-zero observation noise.

Figure 7 (left) reports the time average of the normalized L_2 error as a function of Re for $p = 3$. The RL-ROE exhibits robust performance across the entire range of Reynolds numbers, including in the vicinity of the bifurcation at $Re_c \sim 40$ and for values of Re not seen during training. Figure 7 (right) displays the average over time and over Re of the normalized L_2 error for varying number p of sensors. Although the KF-ROE

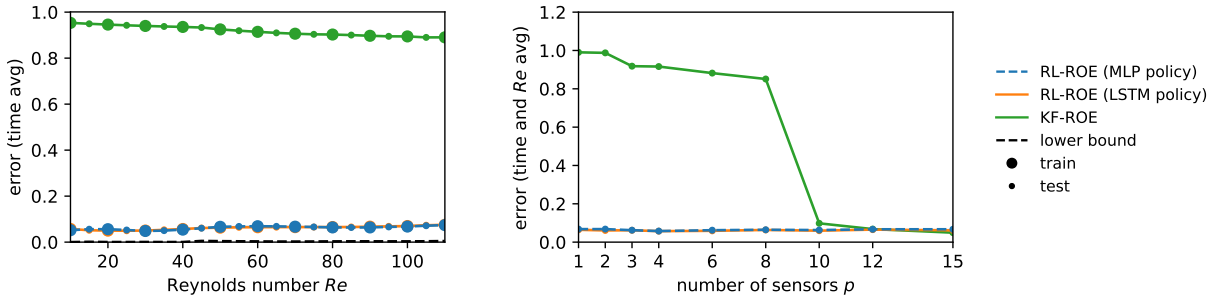


Figure 7: Flow past a cylinder. Left: Normalized L_2 error, averaged over time, versus Re when using $p = 3$ sensors. Values of Re belonging to the training set S are shown by large circles while the testing values are displayed by small circles. The error is averaged over 5 trajectories with randomly-sampled initial true states $\boldsymbol{z}_0$. The shaded areas denote the standard deviation. Right: Normalized L_2 error, averaged over time and over testing values of Re, versus number of sensors p.

eventually becomes very accurate in the presence of a large number p of sensors, the accuracy of the RL-ROE remains remarkably stable as p decreases, allowing it to vastly outperforms the KF-ROE as soon as $p < 12$. This again showcases the benefits of using a nonlinear policy to correct the estimator dynamics. Finally, note that similar to the Burgers example, the LSTM policy does not yield better estimates for the RL-ROE compared to the MLP policy, suggesting that an internal memory is not needed to achieve optimality in this example.

4.3 Navier-Stokes equations: chaotic vorticity flow

An equivalent formulation of the Navier-Stokes equations for 2D problems is given by the vorticity form

$$\frac{\partial \omega}{\partial t} + \boldsymbol{u} \cdot \nabla \omega = \frac{1}{Re} \Delta \omega, \tag{16a}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{16b}$$

where $\boldsymbol{u}(\boldsymbol{x}, t)$ and $\omega(\boldsymbol{x}, t) = \nabla \times \boldsymbol{u}$ are the velocity vector and vorticity at position $\boldsymbol{x} = (x, y)$ and time t, and the scalar Re is the Reynolds number. In this example, we borrow the setup considered in Li et al. (2020) where the vorticity field in a periodic domain $\Omega = [0, 1] \times [0, 1]$ is forced by a term $f(\boldsymbol{x}) = 0.1 \sin(2\pi(x + y)) + 0.1 \cos(2\pi(x + y))$ on the right-hand side of equation 16a with $Re = 1000$, giving rise to chaotic dynamics. Here, instead of different parameter values, we consider various trajectories arising from different vorticity initial conditions sampled from a Gaussian random field; see details in Li et al. (2020). For the estimation problem, we define a sparse sensor array measuring the vorticity ω at p randomly-distributed locations in Ω.

We utilize the publicly-available dataset from Li et al. (2020), which contains 1000 solution trajectories of the vorticity discretized on a 256×256 grid and saved at 50 time steps with sampling time $\Delta t = 1$. For our training dataset $\boldsymbol{Z}_{\text{train}} = \{\boldsymbol{Z}^i, \boldsymbol{Y}^i\}_{i=1}^{50}$, we keep 50 solution trajectories and downsample the vorticity to a 64×64 grid, yielding a state vector $\boldsymbol{z}_k \in \mathbb{R}^{4086}$. For each trajectory, we discard the first 9 snapshots and retain 41 consecutive snapshots $\boldsymbol{Z}^i = \{\boldsymbol{z}_0^i, \ldots, \boldsymbol{z}_{40}^i\}$ and we calculate the corresponding measurements $\boldsymbol{Y}^i = \{\boldsymbol{y}_0^i, \ldots, \boldsymbol{y}_{40}^i\}$ obtained from the p sensors (Appendix E describes the corresponding $\boldsymbol{C}$). We retain $r = 20$ modes when constructing the ROM. These modes are shown in Appendix G. We train the RL-ROE using $K_{\text{start}} = 10$ and episodes of length $K = 30$ steps to make full use of the trajectories stored in $\boldsymbol{Z}_{\text{train}}$. We end the training process when the return no longer increases on average. The RL hyperparameters are reported in Appendix D.

The trained RL-ROE and the KF-ROE are now compared based on their ability to track various unseen trajectories of $\boldsymbol{z}_k$ using sparse measurements from the p randomly distributed sensors. The evaluation is carried out starting from 5 randomly-sampled initial conditions $\boldsymbol{z}_0$ along each trajectory, and the reduced estimate is always initialized as $\hat{\boldsymbol{x}}_0 = \boldsymbol{0}$. Beginning with $p = 3$ sensors, Figure 8 reports the mean (lines)

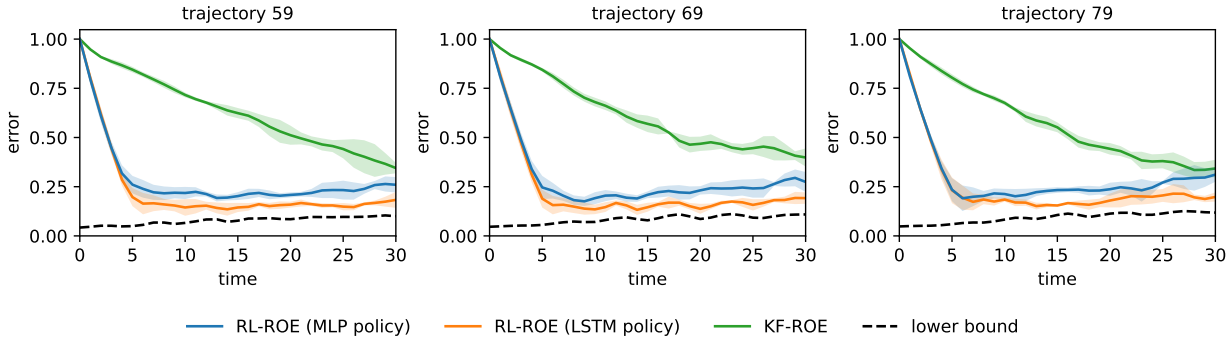


Figure 8: Chaotic vorticity flow with $p = 3$ sensors. Normalized L_2 error of the RL-ROE and KF-ROE for the estimation of trajectories not seen during training. The error is averaged over 5 randomly-sampled initial true states $\boldsymbol{z}_0$. The shaded areas denote the standard deviation.

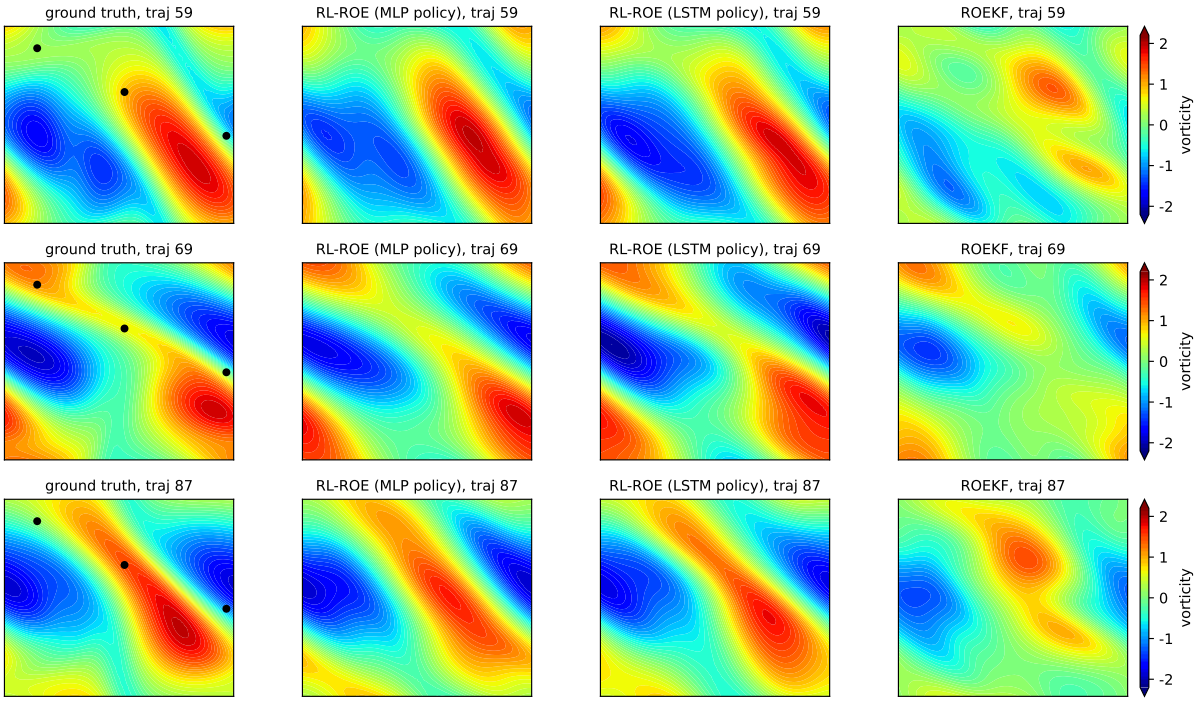


Figure 9: Chaotic vorticity flow with $p = 3$ sensors. Velocity magnitude at $t = 15$ of the ground-truth trajectories not seen during training and corresponding RL-ROE and KF-ROE estimates. The black crosses in the contours of the reference solutions indicate the sensor locations.

and standard deviation (shaded areas) of the normalized L_2 error for 3 unseen trajectories. The error of the RL-ROE, using either the MLP or LSTM policy, rapidly decreases close to the lower bound, while the error of the KF-ROE decreases much more slowly. The ground-truth vorticity at $t = 15$ for the same 3 trajectories and the corresponding RL-ROE and KF-ROE estimates are shown in Figure 6. As observed in the previous two examples, the RL-ROE with either policy once again produce much more accurate estimates than the KF-ROE. Here, however, the LSTM policy leads to lower error than the MLP policy, indicating that the internal memory of the LSTM is helpful in this chaotic case.

Figure 10 (left) reports the time average of the normalized L_2 error for each trajectory for $p = 3$. Figure 10 (right) displays the average over time and over all testing trajectories of the normalized L_2 error for

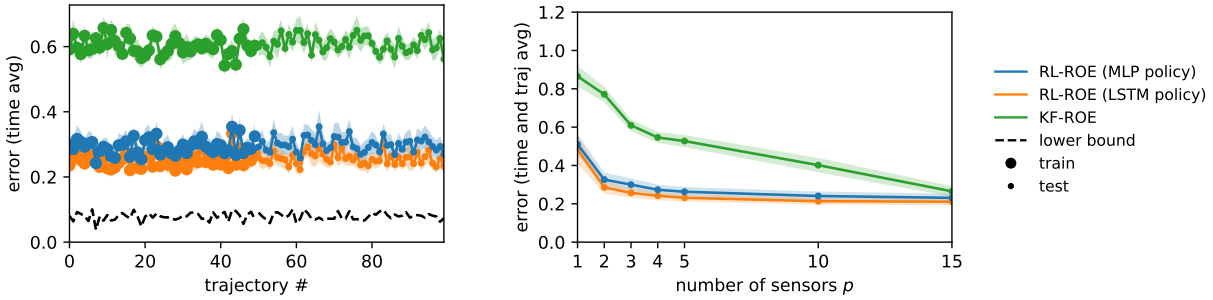


Figure 10: Chaotic vorticity flow. Left: Normalized L_2 error, averaged over time, versus trajectory number when using $p = 3$ sensors. Trajectories belonging to the training set are shown by large circles while the testing trajectories are displayed by small circles. The error is averaged over 5 randomly-sampled initial true states $\boldsymbol{z}_0$. The shaded areas denote the standard deviation. Right: Normalized L_2 error, averaged over time and over testing trajectories, versus number of sensors p.

different numbers p of sensors. We again observe the superiority of the RL-ROE over the KF-ROE, and in this example the LSTM policy consistently produces slightly more accurate estimates than the MLP policy. This confirms that, for this chaotic system, the optimal policy $\boldsymbol{\pi_\theta}$ needs to depend on the entire history of observations, rather than just the current observation and the previous estimate.

5 Related work

Previous studies have already proposed designing state estimators using policies trained through reinforcement learning. Morimoto & Doya (2007) introduced an estimator of the form $\hat{\boldsymbol{x}}_k = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{L}(\hat{\boldsymbol{x}}_{k-1})(\boldsymbol{y}_{k-1} - \boldsymbol{C}\hat{\boldsymbol{x}}_{k-1})$, where $\boldsymbol{f}(\cdot)$ is the state-transition model of the system, and the state-dependent filter gain matrix $\boldsymbol{L}(\hat{\boldsymbol{x}}_{k-1})$ is defined using Gaussian basis functions whose parameters are learned through a variant of vanilla policy gradient. Hu et al. (2020) proposed an estimator of the form $\hat{\boldsymbol{x}}_k = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{L}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)(\boldsymbol{y}_k - \boldsymbol{C}\boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}))$, where $\boldsymbol{L}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)$ is approximated by neural networks trained with a modified Soft-Actor Critic algorithm (Haarnoja et al., 2018). Although they derived convergence properties for the estimate error, the dependence of the filter gain $\boldsymbol{L}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)$ on the ground-truth state $\boldsymbol{x}_k$ limits its practical application. Furthermore, a major difference between these past studies and our work is that they only consider low-dimensional systems with four state variables at most. Our RL-ROE, on the other hand, handles parametric PDEs described by tens of thousands of state variables as shown in the previous section.

In another line of works, reinforcement learning has been applied to learn control policies for joint torques that enable simulated characters to imitate given reference motions consisting of a sequence of target poses; see e.g. Peng et al. (2018a) or Lee et al. (2019). These are essentially trajectory tracking problems, as the policy learns to drive the character's pose towards that defined by the reference motion. Similar to us, these studies also propose to transform the problem into a stationary MDP by augmenting the state; but they do so by appending a scalar phase variable $\phi \in [0, 1]$ that represents the normalized time elapsed in the reference motion. The reward can then be formulated in terms of $q(\phi)$, where $q(\cdot)$ describes the reference motion, and no explicit time dependence appears. However, this approach would not work for our purpose since we do not wish to restrict the RL-ROE to estimating a single ground-truth (reference) trajectory for each value of μ. Many dynamical systems indeed admit multiple post-transient trajectories for given parameter values (Cross & Hohenberg, 1993). Augmenting the MDP's state with the entire snapshot from the reference trajectory instead of just time ensures that the policy $\boldsymbol{\pi_\theta}$ can learn any number of reference trajectories for each μ.

6 Conclusions

In this paper, we have introduced the reinforcement learning reduced-order estimator (RL-ROE), a new state estimation methodology for parametric PDEs. Our approach relies on the construction of a computationally

inexpensive reduced-order model (ROM) to approximate the dynamics of the system. The novelty of our contribution lies in the design, based on this ROM, of a reduced-order estimator (ROE) in which the filter correction term is given by a nonlinear stochastic policy trained offline through reinforcement learning. We introduce a trick to translate the time-dependent trajectory tracking problem in the offline training phase to an equivalent stationary MDP, enabling the use of off-the-shelf RL algorithms. We demonstrate using simulations of the Burgers and Navier-Stokes equations that in the limit of very few sensors, the trained RL-ROE vastly outperforms a Kalman filter designed using the same ROM, which is attributable to the nonlinearity of its policy (see Appendix I for a quantification of this nonlinearity). The RL-ROE yields accurate high-dimensional state estimates for parameter values and initial conditions that are unseen during offline training and unknown during online estimation.

References

Ronald J Adrian and Jerry Westerweel. *Particle image velocimetry*. Number 30. Cambridge university press, 2011.

Shady E Ahmed, Suraj Pawar, Omer San, Adil Rasheed, Traian Iliescu, and Bernd R Noack. On closures for reduced order models—a spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, 33 (9):091301, 2021.

Alexandre Barbagallo, Denis Sipp, and Peter J Schmid. Closed-loop control of an open cavity flow using reduced-order models. *Journal of Fluid Mechanics*, 641:1–50, 2009.

Gildas Besançon. *Nonlinear observers and applications*, volume 363. Springer, 2007.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

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

Mark C Cross and Pierre C Hohenberg. Pattern formation outside of equilibrium. *Reviews of modern physics*, 65(3):851, 1993.

Fred Daum and Jim Huang. Curse of dimensionality and particle filters. In *2003 IEEE Aerospace Conference Proceedings (Cat. No. 03TH8652)*, volume 4, pp. 4_1979–4_1993. IEEE, 2003.

Ido Greenberg, Netanel Yannay, and Shie Mannor. Optimization or architecture: How to hack kalman filtering. *Advances in Neural Information Processing Systems*, 36, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Liang Hu, Chengwei Wu, and Wei Pan. Lyapunov-based reinforcement learning state estimator. *arXiv preprint arXiv:2010.13529*, 2020.

CP Jackson. A finite-element study of the onset of vortex shedding in flow past variously shaped bodies. *Journal of fluid Mechanics*, 182:23–45, 1987.

Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Steven M Kay. *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., 1993.

Sergey K Korovin and Vasily V Fomichev. State observers for linear systems with uncertainty. In *State Observers for Linear Systems with Uncertainty*. de Gruyter, 2009.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

Erwan Lecarpentier and Emmanuel Rachelson. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)*, 38(4):1–13, 2019.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020.

Andrew C Lorenc. The potential of the ensemble kalman filter for nwp—a comparison with 4d-var. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 129(595):3183–3203, 2003.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Jun Morimoto and Kenji Doya. Reinforcement learning state estimator. *Neural computation*, 19(3):730–756, 2007.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. In *International Conference on Machine Learning*, pp. 16691–16723. PMLR, 2022.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4): 1–14, 2018a.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018b.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`, 2019.

Clarence W Rowley and Scott TM Dawson. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

Simo Särkkä. *Bayesian filtering and smoothing*. Cambridge university press, 2013.

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

Chris Snyder, Thomas Bengtsson, Peter Bickel, and Jeff Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Kunihiko Taira, Steven L Brunton, Scott TM Dawson, Clarence W Rowley, Tim Colonius, Beverley J McKeon, Oliver T Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S Ukeiley. Modal analysis of fluid flows: An overview. *AIAA Journal*, 55(12):4013–4041, 2017.

Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On Dynamic Mode Decomposition: theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pp. 153–158. Ieee, 2000.

Paul Zarchan. *Progress in astronautics and aeronautics: fundamentals of Kalman filtering: a practical approach*, volume 208. Aiaa, 2005.

A Dynamic Mode Decomposition

In this appendix, we describe the DMD algorithm (Schmid, 2010; Tu et al., 2014), which is a popular data-driven method to extract spatial modes and low-dimensional dynamics from a dataset of high-dimensional snapshots. Here, we use the DMD to construct a ROM of the form (3) given an observation model $\boldsymbol{C}$ and a concatenated collection of snapshots $\boldsymbol{Z}_{\text{train}} = \{\boldsymbol{Z}^{\mu}\}_{\mu \in S}$, where each $\boldsymbol{Z}^{\mu} = \{\boldsymbol{z}_0^{\mu}, \dots, \boldsymbol{z}_m^{\mu}\}$ contains snapshots from a trajectory of (1a) for a specific value μ.

Fundamentally, the DMD seeks a best-fit linear model of the dynamics in the form of a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ such that $\boldsymbol{z}_{k+1} \simeq \boldsymbol{A}\boldsymbol{z}_k$. First, arrange the snapshots into two time-shifted matrices

$$\boldsymbol{X} = \{\boldsymbol{z}_0^{\mu_1}, \dots, \boldsymbol{z}_{m-1}^{\mu_1}, \dots, \boldsymbol{z}_0^{\mu_q}, \dots, \boldsymbol{z}_{m-1}^{\mu_q}\}, \tag{17a}$$

$$\boldsymbol{Y} = \{\boldsymbol{z}_1^{\mu_1}, \dots, \boldsymbol{z}_m^{\mu_1}, \dots, \boldsymbol{z}_1^{\mu_q}, \dots, \boldsymbol{z}_m^{\mu_q}\}, \tag{17b}$$

where q denotes the number of elements in S. The best-fit linear model is then given by $\boldsymbol{A} = \boldsymbol{Y}\boldsymbol{X}^{\dagger}$, where $\boldsymbol{X}^{\dagger}$ is the pseudoinverse of $\boldsymbol{X}$. The ROM is then obtained by projecting the matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ onto a basis $\boldsymbol{U}$ consisting of the r leading left singular vectors of $\boldsymbol{X}$, which approximate the r leading PCA modes of $\boldsymbol{Z}$. Using the truncated singular value decomposition

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} \tag{18}$$

where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{n \times r}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$, the resulting reduced-order state-transition and observation models are given by

$$\boldsymbol{A}_r = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U} = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}, \tag{19a}$$

$$\boldsymbol{C}_r = \boldsymbol{C}\boldsymbol{U}. \tag{19b}$$

Conveniently, the ROM matrices $\boldsymbol{A}_r$ and $\boldsymbol{C}_r$ can be calculated directly from the truncated SVD of $\boldsymbol{X}$, which avoids forming the large $n \times n$ matrix $\boldsymbol{A}$.

B Bayesian interpretation

Here, we evaluate the meaningfulness of the RL-ROE design by framing the estimator dynamics (4) and the optimization problem (6) in the context of Bayesian inference. Unless stated otherwise, we consider the estimation problem in terms of the reduced state $\boldsymbol{x}_k$, governed by the ROM (3).

B.1 Bayesian optimal filter

From a Bayesian perspective, the goal at each time $k = 1, \dots, K$ is to calculate $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$, the posterior probability density function (pdf) measuring our belief in the true state $\boldsymbol{x}_k$ given the measurement data $\boldsymbol{y}_{1:k} = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_k\}$. It is assumed that we know the transition model $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ describing the dynamics of the system, the observation model $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ relating state and measurement data, and the initial pdf $p(\boldsymbol{x}_0)$. Then, the posterior pdf $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ can formally be obtained recursively by alternating between prediction and update steps (Särkkä, 2013).

Prediction step. Starting from the posterior pdf $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ at $k-1$, one first uses the dynamics of the system to compute the prior pdf at k via the Chapman-Kolmogorov equation

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1}) d\boldsymbol{x}_{k-1}, \tag{20}$$

where the Markovian property of the dynamics has been used.

Update step. The prior is then updated using the new measurement $\boldsymbol{y}_k$ via Bayes' rule, yielding the posterior pdf

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})}, \tag{21}$$

where the normalizing constant is $p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{y}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) d\boldsymbol{x}_k$.

These equations generally do not admit analytical solutions, except in the linear and Gaussian setting which results in the Kalman filter (Särkkä, 2013). Particle filters provide an approximate solution in the general setting but they are computationally expensive, their required ensemble size scaling exponentially with the state dimension (Daum & Huang, 2003; Snyder et al., 2008).

B.2 RL-ROE from a Bayesian perspective

To frame the RL-ROE in the context of Bayesian inference, we first need to relate the state estimate $\hat{\boldsymbol{x}}_k$ to the posterior $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$. Specifically, we define $\hat{\boldsymbol{x}}_k$ as the mean of the posterior,

$$\hat{\boldsymbol{x}}_k = \mathbb{E}[\boldsymbol{x}_k|\boldsymbol{y}_{1:k}] = \int \boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) d\boldsymbol{x}_k, \tag{22}$$

which, assuming that $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ is known, is commonly referred to as the minimum mean square error (MMSE) estimator (Kay, 1993). Using this definition, we can now interpret the estimator dynamics (4) and the optimization problem (6) from the perspective of the Bayesian filter equations, with the transition model $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ and the observation model $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ given by (3a) and (3b).

Prediction step. Starting from the estimate $\hat{\boldsymbol{x}}_{k-1} = \mathbb{E}[\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1}]$ at $k-1$, we take the expectation of (20) to obtain the 'prior' estimate at k,

$$\hat{\boldsymbol{x}}_k^- = \mathbb{E}[\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}] = \boldsymbol{A}_r \hat{\boldsymbol{x}}_{k-1}, \tag{23}$$

where we have made use of the linearity of the transition model (3a), and we have assumed that the process noise $\boldsymbol{w}_k$ is zero-mean (which is empirically observed in our examples; see Appendix J).

Update step. Formally, the estimate $\hat{\boldsymbol{x}}_k$ is given by (22) using the posterior pdf $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ from the Bayes update (21). However, (21) cannot be solved explicitly since the RL-ROE does not evolve the full prior $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$. Instead, we make use of the well-known fact that the MMSE estimator is equivalent to

minimizing the mean square error, or variance, of the posterior estimate (see, e.g., chapters 10 and 11 in Kay (1993) or chapter 2 in Särkkä (2013)). This can be seen as follows:

$$\begin{aligned}\hat{\boldsymbol{x}}_k &= \mathbb{E}[\boldsymbol{x}_k|\boldsymbol{y}_{1:k}] \\ &= \int \boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) d\boldsymbol{x}_k \\ &= \arg\min_{\hat{\boldsymbol{x}}_k} \int \|\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k\|^2 p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) d\boldsymbol{x}_k \\ &= \arg\min_{\hat{\boldsymbol{x}}_k} \mathbb{E}\left[\|\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k\|^2 | \boldsymbol{y}_{1:k}\right]. \end{aligned} \tag{24}$$

We relax the above minimization problem by restricting $\hat{\boldsymbol{x}}_k$ to a specific class of functions, as is done when deriving linear MMSE estimators. In the latter case, the optimal solution is

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k^-), \tag{25}$$

where the prior estimate $\hat{\boldsymbol{x}}_k^-$ is given by (23), and $\boldsymbol{K}_k$ is obtained from the closed-form solution of (24) (Kay, 1993). We generalize this approach by considering the nonlinear form

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{\mu}_{\boldsymbol{\theta}_k}(\boldsymbol{y}_k, \hat{\boldsymbol{x}}_k^-), \tag{26}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}_k}$ is a nonlinear function parameterized by $\boldsymbol{\theta}_k$, whose role is to update the prior $\hat{\boldsymbol{x}}_k^-$ using $\boldsymbol{y}_k$ in such a way to minimize the posterior variance. Then, the minimization problem (24) becomes

$$\boldsymbol{\theta}_k^* = \arg\min_{\boldsymbol{\theta}_k} \mathbb{E}\left[\|\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k\|^2\right], \tag{27}$$

where it is implicitly assumed that the expectation is conditioned on $\boldsymbol{y}_{1:k}$.

RL-ROE. To obtain the estimator dynamics (4) and the optimization problem (6), we further consider that the function $\boldsymbol{\mu}_{\boldsymbol{\theta}_k}$ is stochastic and independent of time; it is therefore expressed by a stationary policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. Then, combining the posterior estimate (26) with the prior estimate (23) gives the recursion

$$\hat{\boldsymbol{x}}_k = \boldsymbol{A}_r \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{a}_k, \tag{28a}$$

$$\boldsymbol{a}_k \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}(\,\cdot\,|\boldsymbol{y}_k, \hat{\boldsymbol{x}}_{k-1}), \tag{28b}$$

which is the estimator dynamics (4). The parameters $\boldsymbol{\theta}$ of the stationary policy are found by extending the minimization problem (27) to all time steps, yielding

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}\left[\sum_{k=1}^{K} \|\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k\|^2\right]. \tag{29}$$

In the RL setting, this optimization problem is solved in a data-driven manner by sampling system trajectories. Thus, the expectation is taken over initial estimates $\hat{\boldsymbol{x}}_0$, initial true states $\boldsymbol{x}_0$, parameters μ, estimate trajectories $\{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \dots\}$ induced by $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ through (4), and true trajectories of states $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$ and measurements $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots\}$. Finally, we express the posterior variance through the reconstructed high-dimensional estimate $\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k$ and the high-dimensional true state $\boldsymbol{z}_k$, and we add a regularization term that penalizes the magnitude of the action $\boldsymbol{a}_k$. This gives

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}\left[\sum_{k=1}^{K} \|\boldsymbol{z}_k - \boldsymbol{U}\hat{\boldsymbol{x}}_k\|^2 + \lambda\|\boldsymbol{a}_k\|^2\right], \tag{30}$$

which is the optimization problem (6).

We conclude this analysis with a couple remarks. First, it demonstrates that the cost being minimized in (30) derives from the variance minimization principle underlying the definition of the MMSE estimator. Second,

an important difference between the RL-ROE and the Bayesian optimal filter is that the RL-ROE does not require knowledge of the distribution of the process noise $\boldsymbol{w}_k$ and observation noise $\boldsymbol{v}_k$. Rather, it leverages knowledge of the true trajectories and corresponding measurements in the offline training phase to find the policy $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ that yields the minimum mean square error. Third, the RL solves the optimization problem (30) in a batch approach during the offline training phase, sampling entire trajectories of the estimate between each update of the policy parameters. The trained RL-ROE is then applied online in a recursive manner.

C Kalman filter

The time-dependent Kalman filter that we use as a benchmark in this paper, KF-ROE, is based on the same ROM (3) as the RL-ROE, with identical matrices $\boldsymbol{A}_r$, $\boldsymbol{C}_r$ and $\boldsymbol{U}$. Similarly to the RL-ROE, the reduced-order estimate $\hat{\boldsymbol{x}}_k$ is given by equation (4a), from which the high-dimensional estimate is reconstructed as $\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k$. However, the KF-ROE differs from the RL-ROE in its definition of the action $\boldsymbol{a}_k$ in (4a), which is instead given by the linear feedback term (5). The calculation of the optimal Kalman gain $\boldsymbol{K}_k$ in (5) requires the following operations at each time step:

$$\boldsymbol{P}_k^- = \boldsymbol{A}_r \boldsymbol{P}_{k-1} \boldsymbol{A}_r^\mathsf{T} + \boldsymbol{Q}_k, \tag{31}$$

$$\boldsymbol{S}_k = \boldsymbol{C}_r \boldsymbol{P}_k^- \boldsymbol{C}_r^\mathsf{T} + \boldsymbol{R}_k, \tag{32}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}_r^\mathsf{T} \boldsymbol{S}_k^{-1}, \tag{33}$$

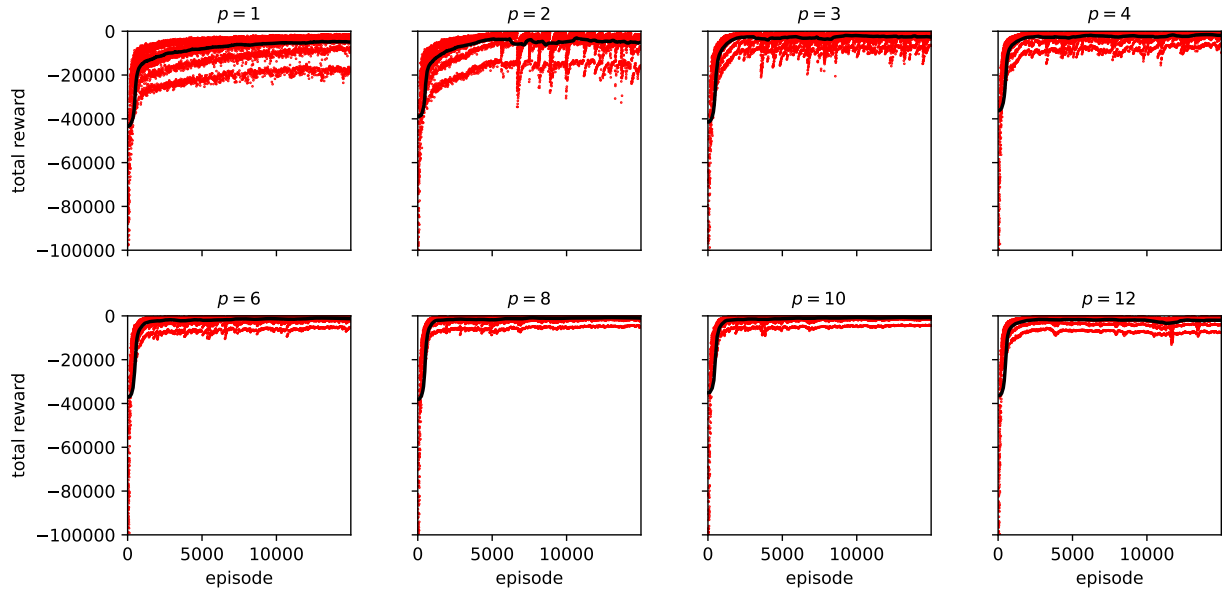
$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}_r) \boldsymbol{P}_k^-, \tag{34}$$

where $\boldsymbol{P}_k^-$ and $\boldsymbol{P}_k$ are respectively the a priori and a posteriori estimate covariance matrices, $\boldsymbol{S}_k$ is the innovation covariance, and $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ are respectively the covariance matrices of the process noise $\boldsymbol{w}_k$ and observation noise $\boldsymbol{v}_k$ in the ROM (3). Following a standard procedure, we tune these noise covariance matrices to yield the best possible results (Simon, 2006). We assume that $\boldsymbol{Q}_k = \beta_Q \boldsymbol{I}$ and $\boldsymbol{R}_k = \beta_R \boldsymbol{I}$ and perform a line search to find the values of β_Q and β_R that yield the best performance. This resulted in $\beta_Q = 10^3$ and $\beta_R = 1$ for the Burgers example, and $\beta_Q = 10^9$ and $\beta_R = 1$ for the Navier-Stokes example. At time step $k = 0$, the a posteriori estimate covariance is initialized as $\boldsymbol{P}_0 = \mathrm{cov}(\boldsymbol{U}^\mathsf{T}\boldsymbol{z}_0 - \hat{\boldsymbol{x}}_0)$, which can be calculated from the distributions (11).

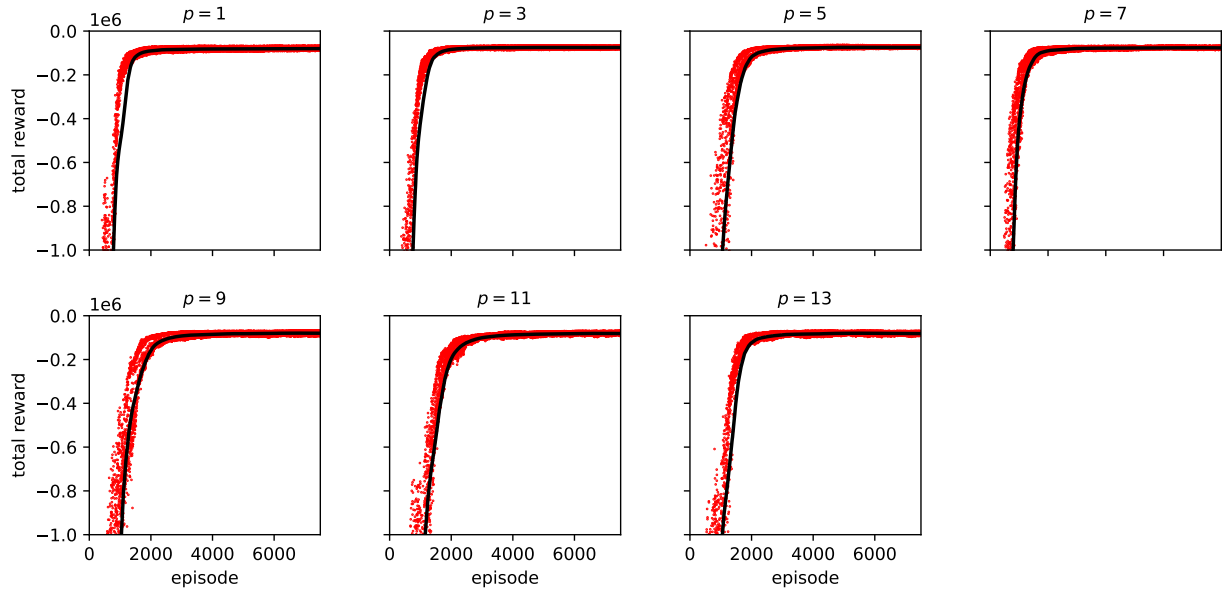
D RL algorithm, hyperparameters and learning curves

We employ the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) to find the optimal policy parameters $\boldsymbol{\theta}^*$. PPO is a policy gradient algorithm that alternates between sampling data by computing a set of trajectories $\{\tau_1, \tau_2, \tau_3, \dots\}$ using the most recent version of the policy, and updating the policy parameters $\boldsymbol{\theta}$ in a way that increases the probability of actions that led to higher rewards during the sampling phase. The policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ encodes a diagonal Gaussian distribution described by an MLP or LSTM network that maps from observation to mean action, $\boldsymbol{\mu}_{\boldsymbol{\theta}'}(\boldsymbol{o}_k)$, together with a vector of standard deviations $\boldsymbol{\sigma}$, so that $\boldsymbol{\theta} = \{\boldsymbol{\theta}', \boldsymbol{\sigma}\}$. We utilize the Stable Baselines3 (SB3) implementation of PPO (Raffin et al., 2019) and define our MDP as a custom environment in OpenAI Gym (Brockman et al., 2016).

For all examples, the stochastic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is trained with PPO using the default hyperparameters from Stable Baselines3, except for the discount factor γ set to 0.9 and the clipping parameter set to 0.05. The mean output of the stochastic policy and the value function are approximated by two separate neural networks. In the MLP case, each network contains two hidden feedforward layers with 64 neurons and tanh activation functions. In the LSTM case, each network contains one LSTM layer with 256 hidden units. The input to the policy is normalized using a running average and standard deviation during the training process, which alternates between sampling data for 10 trajectories (of length 200 timesteps each) and updating the policy. Each policy update consists of multiple gradients steps through the most recent data using 10 epochs, a minibatch size of 64 and a learning rate of 0.0003. The policy is trained for a total of two to three million timesteps, which depending on the dimensionality of the ROM takes between 15 min and an hour on a Core i7-12700K CPU. Figure 11 reports the learning curves corresponding to the unforced and forced cases. During training, the policy is tested (with stochasticity switched off) after each update using 20 separate



(a) Burgers



(b) Navier-Stokes

Figure 11: Learning curves for the stochastic policy of the RL-ROE for the (a) Burgers and (b) Navier-Stokes cases. Each plot corresponds to a different number p of sensor measurements. The line and shaded area show the mean and standard deviation of the results over 10 runs, each one smoothed with a moving average of size 100 episodes.

test trajectories, and is saved if it outperforms the previous best policy. Finally, the RL-ROE is assigned the latest saved policy upon ending of the training process, and the stochasticity of the policy is switched off during subsequent evaluation of the RL-ROE.

E Construction of the observation matrix

We describe how we construct the observation matrix $\boldsymbol{C}$ in the Burgers and Navier-Stokes examples, once the number, type and locations of the sensors have been chosen.

In the Burgers example, the state vector $\boldsymbol{z}_k \in \mathbb{R}^n$ contains the values of u at n collocation points, and the measurements $\boldsymbol{y}_k \in \mathbb{R}^p$ consist of the values of u at p equally-spaced sensors, as described in Section 4.1. Let us introduce the indices $\{j_1, \ldots, j_p\}$ of the entries in $\boldsymbol{z}_k$ corresponding to the measurements $\boldsymbol{y}_k$. Then, $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ can be related by $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{z}_k$, where the matrix $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ contains ones at the entries indexed $\{(1, j_1), \ldots, (p, j_p)\}$, and zeros everywhere else.

In the flow past a cylinder Navier-Stokes example, the state vector $\boldsymbol{z}_k \in \mathbb{R}^{2n}$ contains the horizontal and vertical components of velocity $\boldsymbol{u}$ at n collocation points, and the measurements $\boldsymbol{y}_k \in \mathbb{R}^{2p}$ consist of the components of $\boldsymbol{u}$ at p randomly distributed sensors in a box in the cylinder wake, as described in Section 4.2. Let us introduce the indices $\{j_1, \ldots, j_{2p}\}$ of the entries in $\boldsymbol{z}_k$ corresponding to the measurements $\boldsymbol{y}_k$. Then, $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ can be related by $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{z}_k$, where the matrix $\boldsymbol{C} \in \mathbb{R}^{2p\times n}$ contains ones at the entries indexed $\{(1, j_1), \ldots, (2p, j_{2p})\}$, and zeros everywhere else.

In the chaotic vorticity Navier-Stokes example, the state vector $\boldsymbol{z}_k \in \mathbb{R}^n$ contains the values of ω at n collocation points, and the measurements $\boldsymbol{y}_k \in \mathbb{R}^p$ consist of the values of ω at p randomly distributed sensors in the domain, as described in Section 4.3. Let us introduce the indices $\{j_1, \ldots, j_p\}$ of the entries in $\boldsymbol{z}_k$ corresponding to the measurements $\boldsymbol{y}_k$. Then, $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ can be related by $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{z}_k$, where the matrix $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ contains ones at the entries indexed $\{(1, j_1), \ldots, (p, j_p)\}$, and zeros everywhere else.

F Additional results for the Burgers equation

Figure 12 shows spatio-temporal contours of the ground-truth solution and corresponding estimates, for $p = 2$ and 12. The RL-ROE vastly outperforms the KF-ROE for $p = 2$, while for $p = 12$ the KF-ROE is slightly more accurate.

G Modes for the two Navier-Stokes examples

The first 18 modes (i.e. columns of $\boldsymbol{U}$) of the 20-dimensional ROM for the flow past a cylinder example of Section 4.2 are displayed in Figure 13, while the 20 modes of the 20-dimensional ROM for the chaotic flow example of Section 4.3 are displayed in Figure 14.

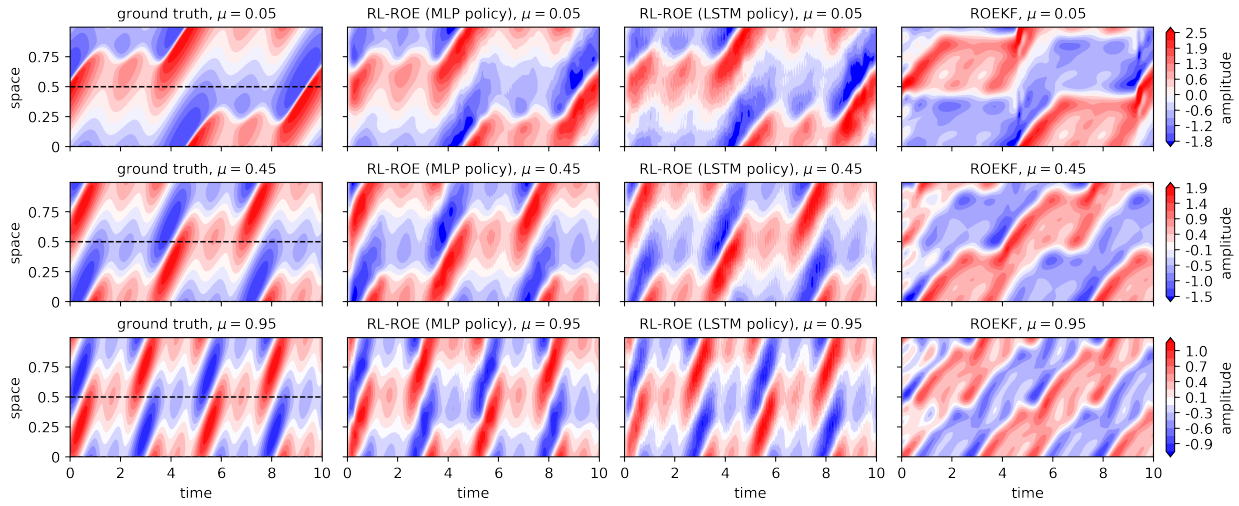
H Effect of observation noise for the Navier-Stokes equations

In this appendix, we evaluate the estimation accuracy of the RL-ROE in the presence of non-zero observation noise $\boldsymbol{n}_k$ polluting the sensor measurements $\boldsymbol{y}_k$ in (1). Specifically, we consider that $\boldsymbol{n}_k$ is Gaussian white noise of standard deviation $\sigma = 0.1$. For the case of $p = 3$ sensors, Figure 15 shows the time series of the noise-free measurements contained in $\boldsymbol{y}_k$ for various values of Re, together with their polluted counterpart (recall each sensor measures two components of velocity, as detailed in Appendix E). Using the noisy measurements, we then repeat the same experiments carried out in the main text; the results are shown in Figures 16 and 17, which are the counterparts of Figures 5 and 6. We observe excellent robustness of the RL-ROE in the presence of noise, with the estimate maintaining high accuracy.

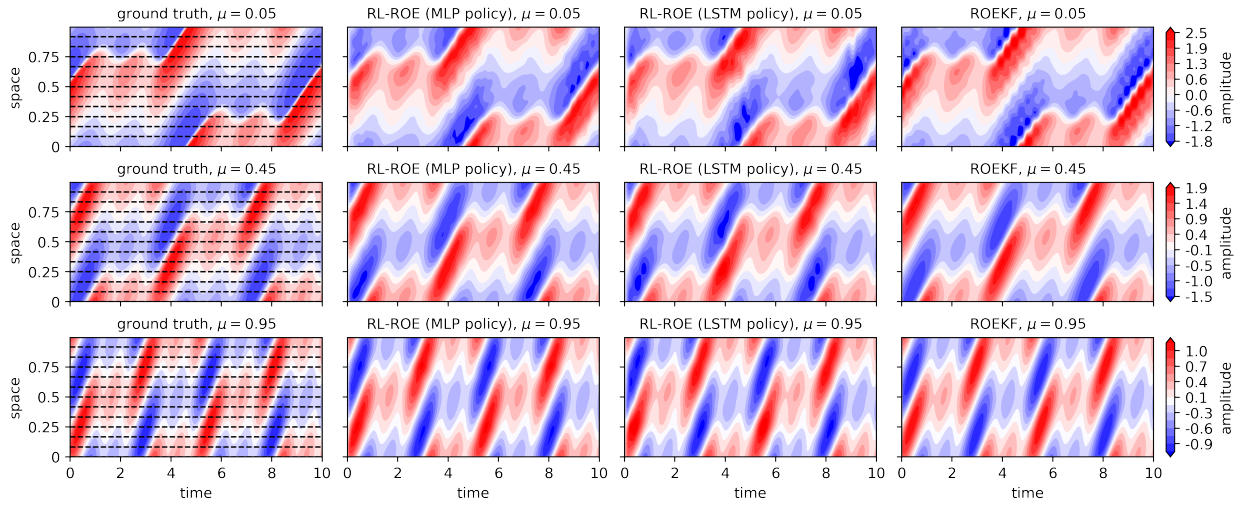
I Nonlinearity of the trained policy

In this appendix, we evaluate the degree of nonlinearity of the RL-trained policies $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ obtained in our Burgers and Navier-Stokes examples. Since the stochasticity of $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is switched off during online deployment, $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ can be expressed by its mean function $\boldsymbol{\mu}_{\boldsymbol{\theta}'}$ (see Appendix D) so that the action in (4b) becomes

$$\boldsymbol{a}_k = \boldsymbol{\mu}_{\boldsymbol{\theta}'}(\boldsymbol{y}_k, \hat{\boldsymbol{x}}_{k-1}). \tag{35}$$



(a) $p = 2$



(b) $p = 12$

Figure 12: Burgers equation with (a) $p = 2$ and (b) $p = 12$ sensors. Ground-truth (reference) trajectories for values of μ not seen during training and corresponding RL-ROE and KF-ROE estimates. The dashed lines on the reference trajectory plots indicate the sensor data seen by the RL-ROE and KF-ROE.

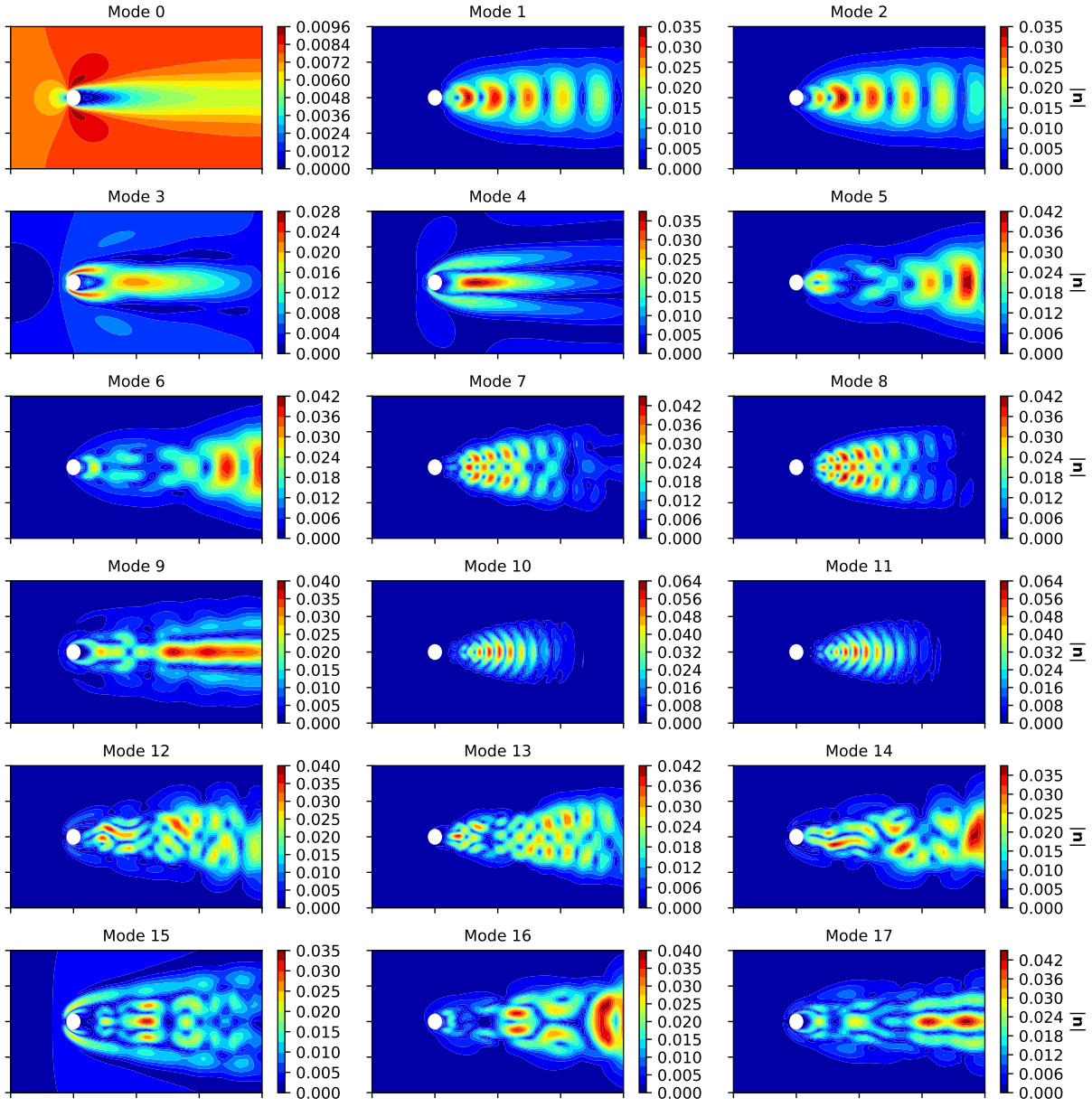


Figure 13: First 18 modes of the 20-dimensional ROM for the flow past a cylinder example.

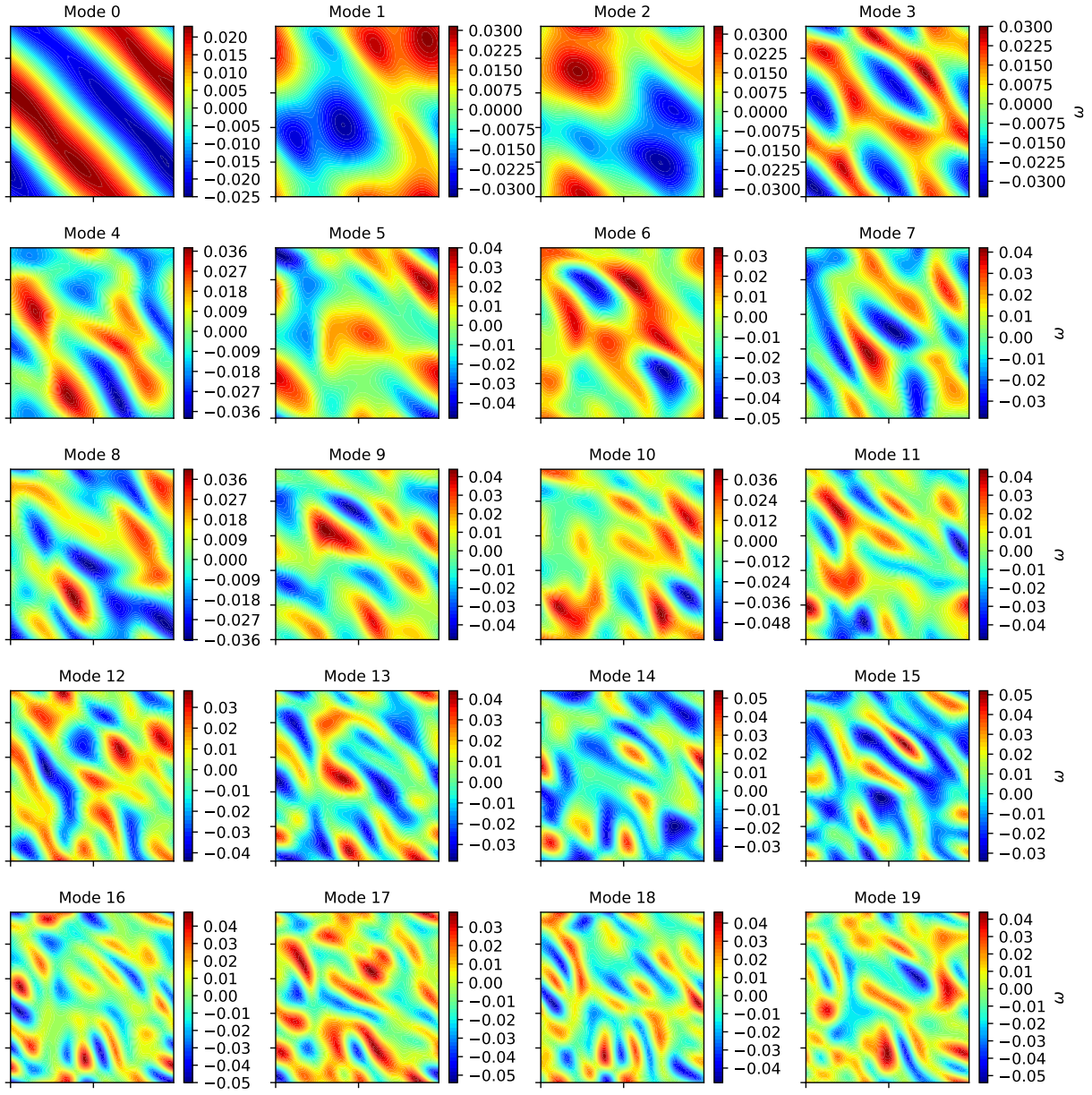


Figure 14: Modes of the 20-dimensional ROM for the chaotic flow example.

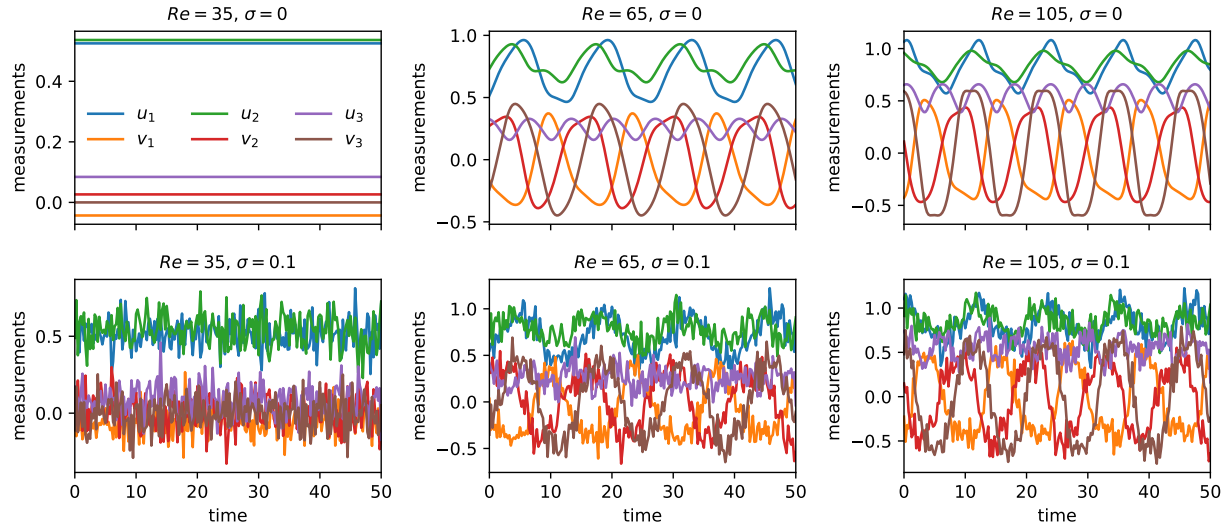


Figure 15: Navier-Stokes equations with $p = 3$ sensors and observation noise of std σ. Time series of measurements for different values of Re for $\sigma = 0$ (top row) and $\sigma = 0.1$ (bottom row).

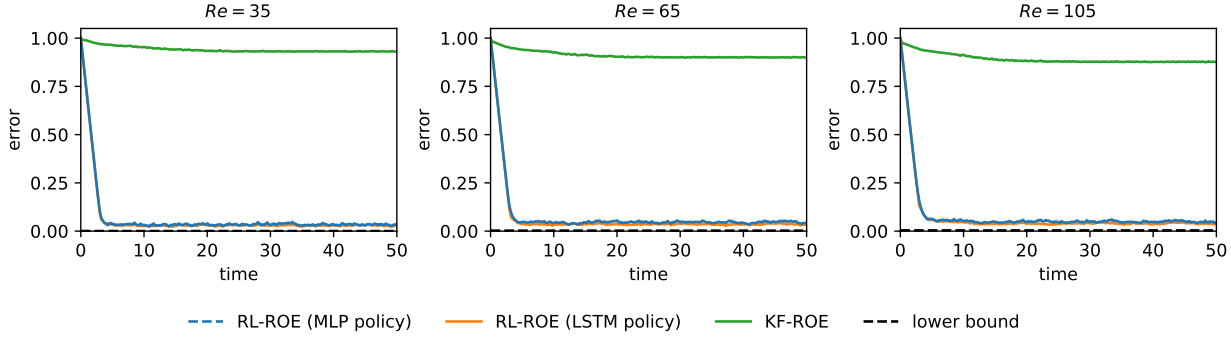


Figure 16: Navier-Stokes equations with $p = 3$ sensors and observation noise of std $\sigma = 0.1$. Normalized L_2 error of the RL-ROE and KF-ROE for the estimation of trajectories corresponding to values of Re not seen during training.

We therefore quantify the nonlinearity of the policy by evaluating the Jacobian of the function $\boldsymbol{\mu}_{\boldsymbol{\theta}'}$ with respect to its two arguments $\boldsymbol{y}_k$ and $\hat{\boldsymbol{x}}_{k-1}$. The Jacobian is a matrix whose components are the first-order derivatives $\partial\mu_i/\partial y_j$ and $\partial\mu_i/\partial\hat{x}_j$, where (i, j) refers to the indices of the vectors entries in $\boldsymbol{\mu}$ and $\boldsymbol{y}_k$ or $\hat{\boldsymbol{x}}_{k-1}$, respectively. Instead of looking at individual components, we consider the Frobenius norm of the Jacobian, defined as

$$\left\| \frac{\partial \boldsymbol{\mu}_{\boldsymbol{\theta}'}}{\partial(\boldsymbol{y}, \hat{\boldsymbol{x}})} \right\|_F^2 = \sum_{i=1}^{r} \left[\sum_{j=1}^{p} \left(\frac{\partial \mu_i}{\partial y_j} \right)^2 + \sum_{j=1}^{r} \left(\frac{\partial \mu_i}{\partial \hat{x}_j} \right)^2 \right]. \tag{36}$$

The Jacobian (and its norm) of a linear policy will be independent of the input values, while the Jacobian (and its norm) of a nonlinear policy will change with the input values. Figure 18 show the distribution of the norm of the Jacobian of the trained policies obtained in the Burgers and Navier-Stokes examples for various values of p. The distributions are obtained by calculating the Jacobian along a solution trajectory of the RL-ROE corresponding to the results shown in Sections 4.1 and 4.2. The wide spread of the distributions demonstrates that the mean function $\boldsymbol{\mu}_{\boldsymbol{\theta}'}$ trained by the RL process is highly nonlinear, as opposed to the linear correction term (5) in the KF-ROE.

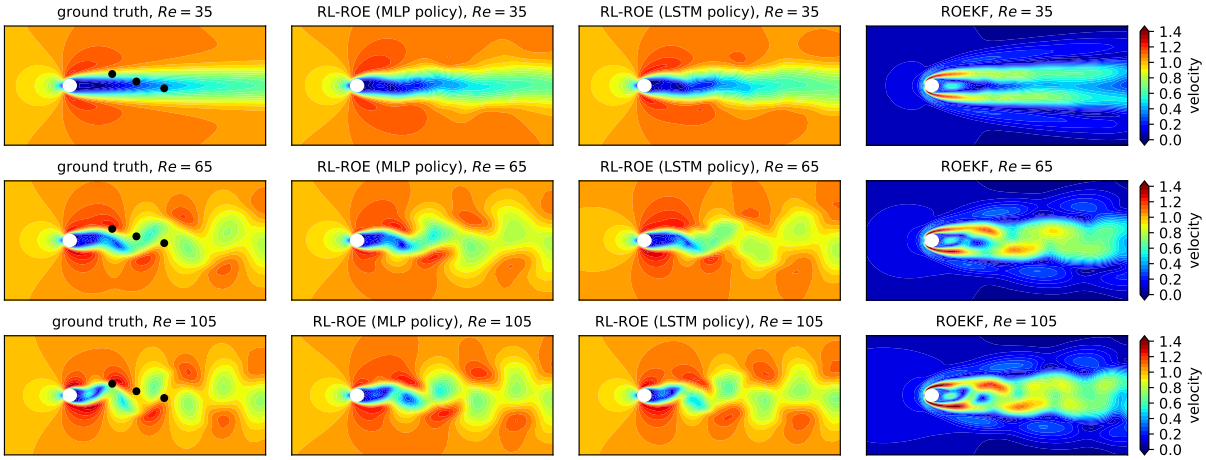
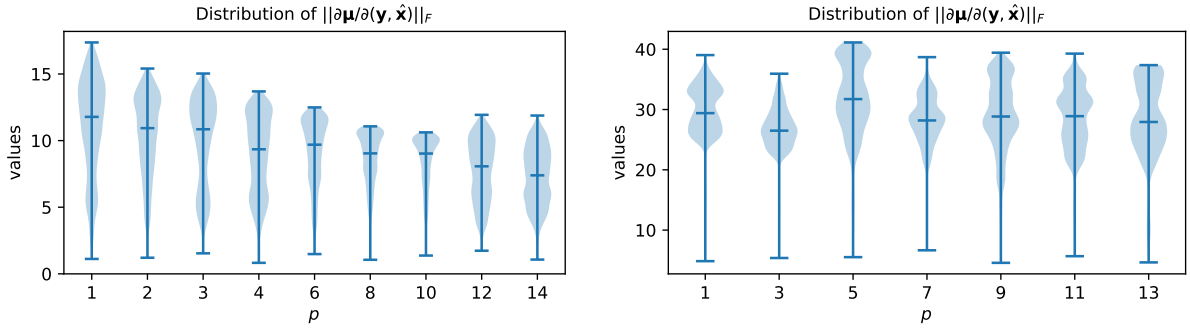


Figure 17: Navier-Stokes equations with $p = 3$ sensors and observation noise of std $\sigma = 0.1$. Velocity magnitude at $t = 50$ of the ground-truth (reference) trajectories for values of Re not seen during training and corresponding RL-ROE and KF-ROE estimates. The black crosses in the contours of the reference solutions indicate the sensor locations.



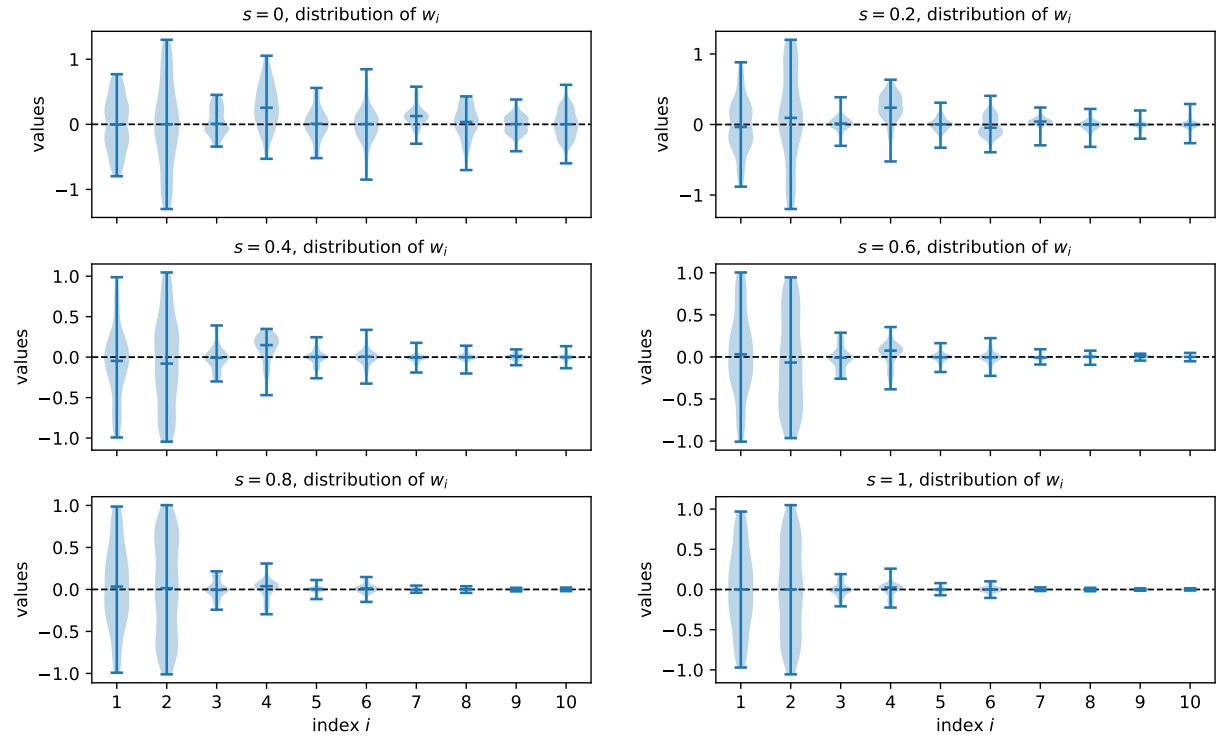
(a) Burgers equation

(b) Navier-Stokes equations

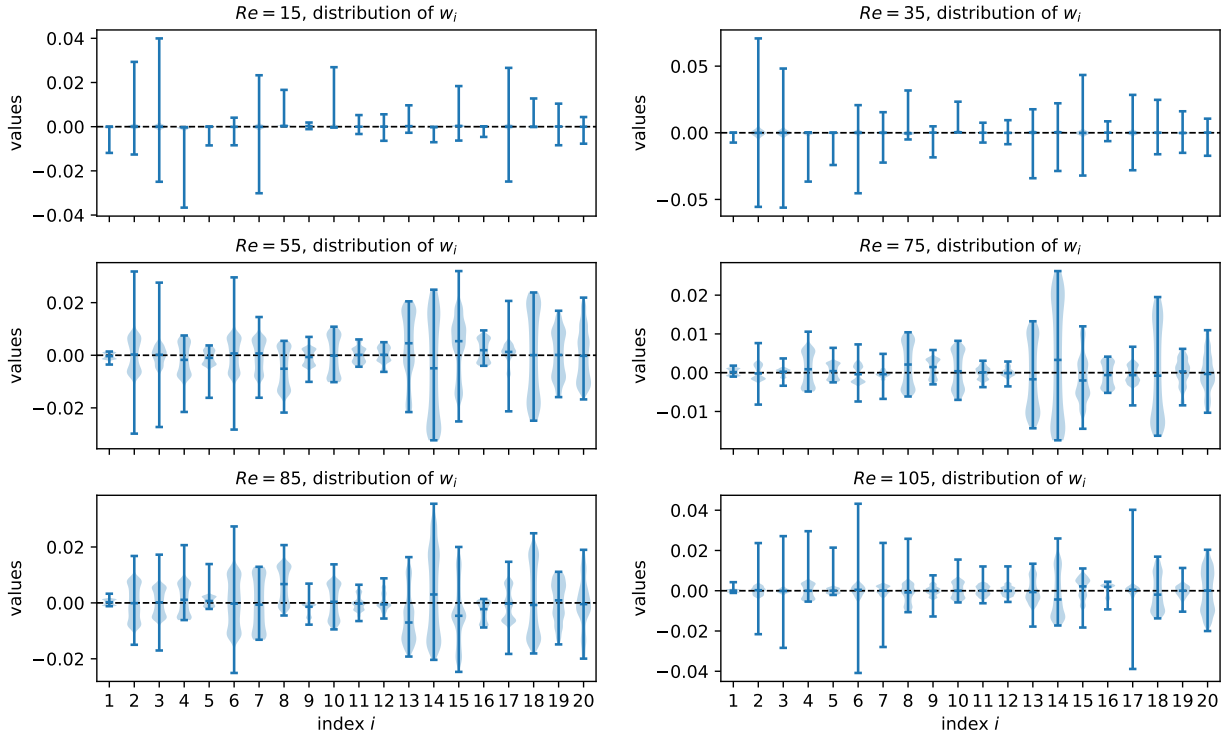
Figure 18: Nonlinearity of the trained policies obtained in the (a) Burgers and (b) Navier-Stokes examples for various p. Distribution of the Frobenius norm of the Jacobian of the trained mean policy $\boldsymbol{\mu}_{\boldsymbol{\theta}'}$, sampled along solution trajectories of the RL-ROE.

J Distribution of process noise

We evaluate empirically the distribution of the process noise $\boldsymbol{w}_k$ in the ROM dynamics (3a). First, note that the vector $\boldsymbol{w}_k$ can be evaluated as $\boldsymbol{w}_k = \boldsymbol{A}_r \boldsymbol{x}_{k-1} - \boldsymbol{x}_k$, where $\boldsymbol{x}_{k-1} = \boldsymbol{U}^\mathsf{T} \boldsymbol{z}_{k-1}$ and $\boldsymbol{x}_k = \boldsymbol{U}^\mathsf{T} \boldsymbol{z}_k$ are the reduced-order projections of two consecutive states $\boldsymbol{z}_{k-1}$ and $\boldsymbol{z}_k$ solving the high-dimensional dynamics (1). For a given ROM, we can then sample the process noise by evaluating values of $\boldsymbol{w}_k$ along trajectories of the high-dimensional dynamics (1). We show in Figure 19 the distributions of the components of $\boldsymbol{w}_k$ for the Burgers and Navier-Stokes examples, using the ROM constructed in Sections 4.1 and 4.2. The sampling is done along trajectories of (1) corresponding to different parameter values (the parameter being μ for Burgers, Re for Navier-Stokes), and the corresponding distributions of process noise are shown separately for each parameter value. (Note that the same ROM is shared across all parameter values.) The distributions reveal that the process noise is non-Gaussian but approximately zero-mean.



(a) Burgers equation



(b) Navier-Stokes equations

Figure 19: Distribution of process noise in the (a) Burgers and (b) Navier-Stokes examples, sampled along solution trajectories of the high-dimensional dynamics (1). The distributions of each component of $\boldsymbol{w}_k$ obtained for each parameter value are shown individually.