Reinforcement Learning State Estimation for High-Dimensional Nonlinear Systems

Anonymous authors
Paper under double-blind review

Abstract

In high-dimensional nonlinear systems such as fluid flows, the design of state estimators such as Kalman filters relies on a reduced-order model (ROM) of the dynamics. 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 data assimilation feedback term is given by a nonlinear stochastic policy trained through reinforcement learning. The flexibility of the nonlinear policy enables the RL-ROE to compensate for errors of the ROM, while still taking advantage of the imperfect knowledge of the dynamics. We show that the trained RL-ROE is able to outperform a Kalman filter designed using the same ROM, and displays robust estimation performance with respect to different reference trajectories and initial state estimates.

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 & Musoff, 2015). 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, including the Kalman filter or its numerous extensions. Thus, state estimators are instead designed based on a reduced-order model (ROM) of the underlying dynamics (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 itself 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 is replaced by a nonlinear stochastic policy trained through reinforcement learning (RL). The flexibility of the nonlinear policy enables the RL-ROE to compensate for errors of the ROM, while still taking advantage of the imperfect knowledge of the dynamics. We describe how we frame the problem as a stationary Markov decision process in order to enable RL training, which is non-trivial since the RL-ROE must be able to estimate time-varying states. Finally, we show that the trained RL-ROE is able to outperform a Kalman filter designed using the same ROM, and displays robust estimation performance with respect to different reference trajectories and initial state estimates. The RL-ROE is the first application of reinforcement learning to state estimation for high-dimensional systems.

2 Problem formulation

2.1 Setup

Consider the discrete-time nonlinear system given by

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

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 current to next state, and $\boldsymbol{C} \in \mathbb{R}^{p \times n}$ is a linear map from state to measurement. In this study, we assume that the dynamics given in (1) are obtained from the numerical discretization of a nonlinear partial differential equation (PDE), which typically requires a large number n of state dimensions. Note that we do not account for exogenous control inputs to the system, which will be studied in future extensions of the present work.

2.2 Reduced-order model

Because the high dimensionality of (1) makes online prediction and control impractical, it is instead customary to formulate a reduced-order model (ROM) of the dynamics (Rowley & Dawson, 2017). First, one chooses 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 $\mathbb{R}^n$ in which most of the dynamics is assumed to take place. Stacking these modes as columns of a matrix $\boldsymbol{U} \in \mathbb{R}^{n \times r}$, one can then express $\boldsymbol{z}_k \simeq \boldsymbol{U}\boldsymbol{x}_k$, 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 wide applications in fields ranging from fluid dynamics to neuroscience (Schmid, 2010; Kutz et al., 2016). Starting with a collection of snapshots $\boldsymbol{Z} = \{\boldsymbol{z}_0, \ldots, \boldsymbol{z}_m\}$ collected along a trajectory of (1a), 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 $\boldsymbol{z}_{k+1} \simeq \boldsymbol{A}\boldsymbol{z}_k$, and computes the modes $\boldsymbol{U}$ as the r leading principal component analysis (PCA) modes of $\boldsymbol{Z}$. The transformation $\boldsymbol{z}_k \simeq \boldsymbol{U}\boldsymbol{x}_k$ and the orthogonality of $\boldsymbol{U}$ then yield a linear discrete-time ROM of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_r\boldsymbol{x}_k + \boldsymbol{w}_k, \tag{2a}$$
$$\boldsymbol{y}_k = \boldsymbol{C}_r\boldsymbol{x}_k + \boldsymbol{v}_k, \tag{2b}$$

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. In order to account for the neglected PCA modes of $\boldsymbol{Z}$ as well as the unmodeled dynamics incurred by the linear approximation $\boldsymbol{z}_{k+1} \simeq \boldsymbol{A}\boldsymbol{z}_k$, we add (unknown) non-Gaussian process noise $\boldsymbol{w}_k$ and observation noise $\boldsymbol{v}_k$. Additional details regarding the calculation of $\boldsymbol{A}_r$ and $\boldsymbol{U}$ are provided in Appendix A.

2.3 Reduced-order estimator

This paper uses reinforcement learning (RL) to solve the following estimation problem: given a sequence of measurements $\{\boldsymbol{y}_0, \cdots, \boldsymbol{y}_k\}$ from a reference trajectory $\{\boldsymbol{z}_0, \cdots, \boldsymbol{z}_k\}$ of (1) and knowing the ROM (2) defined by $\boldsymbol{A}_r$, $\boldsymbol{C}_r$ and $\boldsymbol{U}$, we want to estimate the high-dimensional state $\boldsymbol{z}_k$ at current time k. To this effect, we design a reduced-order estimator (ROE) of the form

$$\hat{\boldsymbol{x}}_k = \boldsymbol{A}_r\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{a}_k, \tag{3a}$$
$$\boldsymbol{a}_k \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}(\,\cdot\,|\boldsymbol{y}_k, \hat{\boldsymbol{x}}_{k-1}), \tag{3b}$$

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 stochastic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ which depends on 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 defines the stochastic policy, whose goal is to minimize the mean square error $\mathbb{E}[\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k]$ over a range of reference trajectories and initial reduced-order state estimates. Here, $\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k$ denotes the high-dimensional state estimate reconstructed from $\hat{\boldsymbol{x}}_k$. A Kalman filter is a special case of such an estimator, for which the action in (3b) is given by

$$\boldsymbol{a}_k = \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{C}_r\boldsymbol{A}_r\hat{\boldsymbol{x}}_{k-1}), \tag{4}$$

with $\boldsymbol{K}_k \in \mathbb{R}^{r \times p}$ the optimal Kalman gain. Although the Kalman filter is optimal when the state-transition and observation models are known exactly, its performance suffers in the presence of unmodeled dynamics. In our case, such model errors are unavoidable due to the ROM (2) being an inherent approximation of the high-dimensional dynamics (1), which motivates our adoption of the more general form (3b). This form 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 stochastic policy $\boldsymbol{\pi_\theta}$, which we train with deep RL in an offline stage. The stochasticity of $\boldsymbol{\pi_\theta}$ forces the RL algorithm to explore different actions during the training process, in order to find eventually an optimal $\boldsymbol{\theta}^*$ such that $\mathbb{E}[\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k]$ is minimized for various reference trajectories and initial estimates. We call the estimator constructed and trained through this process an RL-trained ROE, or RL-ROE for short.

Thus, the methodology we propose consists of two steps. In a first offline stage, a ROM of the form (2) is obtained using high-dimensional snapshots $\boldsymbol{z}_k$ from a single trajectory of (1). The RL-ROE (3) is then constructed based on this ROM, and its policy $\boldsymbol{\pi_\theta}$ is trained using high-dimensional snapshots $\boldsymbol{z}_k$ from multiple reference trajectories of (1). Finally, the trained RL-ROE is deployed online to track a reference trajectory of (1). In the online stage, the RL-ROE only requires measurements $\boldsymbol{y}_k$ from the reference trajectory, and gives an estimate $\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k$ for the high-dimensional state.

In summary, our contributions in this paper are two-fold:

1. We propose RL-ROE, a reduced-order state estimator for high-dimensional nonlinear systems. The RL-ROE takes the form (3), which combines two unique features: first, the state transition model $\boldsymbol{A}_r$ is a ROM of the high-dimensional dynamics; second, the term $\boldsymbol{a}_k$ that assimilates measurements is sampled from a stochastic policy $\boldsymbol{\pi_\theta}$ trained with RL. The training procedure for $\boldsymbol{\pi_\theta}$, which involves a non-trivial reformulation of the time-varying tracking problem as a stationary Markov decision process, is described in Section 4.
2. The performance of the RL-ROE is compared in Section 5 with that of KF-ROE, a Kalman filter constructed from the same ROM. The comparison is performed in the context of the Burgers equation using a range of reference trajectories and initial state estimates.

3 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. Their reward function, however, was calculated using the measurement error instead of the state estimate error, potentially limiting the performance of the trained estimator. 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 reference state $\boldsymbol{x}_k$ limits its practical application.

A major difference between these past studies and our work is that they do not construct a ROM of the dynamics and only consider low-dimensional systems with four state variables at most, in comparison with the hundred or more state dimensions that our RL-ROE can handle. Therefore, RL-ROE represents the first application of reinforcement learning to state estimation for high-dimensional systems, which makes it applicable to systems governed by PDEs such as fluid flows.

4 Training methodology

In this section, we describe the offline training process for the policy $\boldsymbol{\pi_\theta}$ in the RL-ROE (3). In order to train $\boldsymbol{\pi_\theta}$ with reinforcement learning, we need to formulate the problem 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 $\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k$ and a time-dependent reference state $\boldsymbol{z}_k$. At first sight, such trajectory tracking problem requires a time-dependent reward function and, therefore, a time-varying MDP.

To be able to use off-the-shelf RL algorithms, we introduce a trick to translate this time-varying MDP to an equivalent, extended stationary MDP. Indeed, we show hereafter that the problem can be framed as a stationary MDP by including $\boldsymbol{z}_k$ into our definition of the MDP's state. Letting $\boldsymbol{s}_k = (\boldsymbol{z}_k, \hat{\boldsymbol{x}}_{k-1}) \in \mathbb{R}^{n+r}$ 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 $\mathcal{S} = \mathbb{R}^{n+r}$ 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 $\boldsymbol{\pi_\theta}$ defined in (3b), which can be expressed as

$$\boldsymbol{a}_k \sim \boldsymbol{\pi_\theta}(\,\cdot\,|\boldsymbol{o}_k), \tag{5}$$

where $\boldsymbol{o}_k = (\boldsymbol{y}_k, \hat{\boldsymbol{x}}_{k-1}) = (\boldsymbol{C}\boldsymbol{z}_k, \hat{\boldsymbol{x}}_{k-1})$ is a partial observation of the current state $\boldsymbol{s}_k$. The state $\boldsymbol{s}_{k+1} = (\boldsymbol{z}_{k+1}, \hat{\boldsymbol{x}}_k)$ at the next time step is then obtained from equations (1a) and (3a) as

$$\boldsymbol{s}_{k+1} = (\boldsymbol{f}(\boldsymbol{z}_k), \boldsymbol{A}_r\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{a}_k), \tag{6}$$

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

$$r_k = \mathcal{R}(\boldsymbol{s}_k, \boldsymbol{a}_k, \boldsymbol{s}_{k+1}) = -(\boldsymbol{z}_k - \boldsymbol{U}\hat{\boldsymbol{x}}_k)^\intercal \boldsymbol{Q}(\boldsymbol{z}_k - \boldsymbol{U}\hat{\boldsymbol{x}}_k) - \boldsymbol{a}_k^\intercal \boldsymbol{R}\boldsymbol{a}_k, \tag{7}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R} \in \mathbb{R}^{r\times r}$ are positive semidefinite and positive definite matrices, respectively. The first term in the reward function $\mathcal{R}$ penalizes the difference between the high-dimensional state estimate $\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k$ and the reference $\boldsymbol{z}_k$, which is only partially observed by the agent. The second term favors smaller values for the action $\boldsymbol{a}_k$; such regularization leads to more robust estimation performance in the presence of noise during online deployment of the RL-ROE, as we will see later. Unless indicated otherwise, we will consider $\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R} = \boldsymbol{I}$. Thanks to the incorporation of $\boldsymbol{z}_k$ into $\boldsymbol{s}_k$, the reward function (7) has no explicit time dependence and the MDP is therefore stationary.

The goal of the RL training process is then to find the optimal policy parameters

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\tau\sim\boldsymbol{\pi_\theta}} [R(\tau)], \tag{8}$$

where the expectation is over trajectories $\tau = (\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{s}_2, \boldsymbol{a}_2, \dots)$, and $R(\tau)$ is the finite-horizon undiscounted return

$$R(\tau) = \sum_{k=1}^{K} r_k, \tag{9}$$

with the integer K denoting the length of each training trajectory. Contrary to conventional RL notation, trajectories here start at time $k = 1$. Indeed, the environment is initialized at time $k = 0$ according to the distributions

$$\boldsymbol{z}_0 \sim \boldsymbol{p}_{\boldsymbol{z}_0}(\cdot), \tag{10a}$$

$$\hat{\boldsymbol{x}}_0 \sim \boldsymbol{p}_{\hat{\boldsymbol{x}}_0}(\cdot), \tag{10b}$$

from which the augmented state $\boldsymbol{s}_1 = (\boldsymbol{z}_1, \hat{\boldsymbol{x}}_0) = (\boldsymbol{f}(\boldsymbol{z}_0), \hat{\boldsymbol{x}}_0)$ follows immediately. Thus, $\boldsymbol{s}_1$ constitutes the start of the trajectory of agent-environment interactions. This sequence of operations mirrors that of a Kalman filter: the calculation of the current state estimate uses the previous state estimate as well as the current measurement, and therefore begins from the second time step, which is $k = 1$ in our case.

To find the optimal policy parameters $\boldsymbol{\theta}^*$, 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). PPO 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 $\boldsymbol{\pi_\theta}$ encodes a diagonal Gaussian distribution described by a neural 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). Besides the discount factor γ, all results to follow are obtained with the default PPO hyperparameters in SB3, which demonstrates the robustness of our approach with respect to the RL hyperparameters.

Remark 1. The offline RL training phase described in this section (to find the optimal parameters $\boldsymbol{\theta}^*$) requires knowledge of the high-dimensional state $\boldsymbol{z}_k$ from several reference trajectories of (1).

During online deployment, however, the RL-ROE (3) only relies on measurements $\boldsymbol{y}_k$, since the trained policy $\boldsymbol{\pi}_{\boldsymbol{\theta}^*}$ is conditioned on $\boldsymbol{y}_k$ and the previous reduced state estimate $\hat{\boldsymbol{x}}_{k-1}$.

Remark 2. Similar to any learning method, the trained RL-ROE is expected to perform well for reference trajectories and initial estimates sampled from the same distributions used during the offline training phase. Thus, the choice of the distributions for $\boldsymbol{z}_0$ and $\hat{\boldsymbol{x}}_0$ in (10) is critical since it defines the range of initial reference states and initial estimates that the trained RL-ROE will be able to handle during online deployment. This is similar to the state of the art in model-based observer theory, which only guarantees convergence of nonlinear observers locally, i.e., near the true initial states (Benosman & Borggaard, 2021).

5 Results

We evaluate our proposed RL-ROE using simulations of the Burgers equation, a prototypical nonlinear hyperbolic PDE which takes the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = f(x,t), \tag{11}$$

where $u(x,t)$ is the velocity at position $x \in [0,L]$ and time $t \in [0,T]$, $f(x,t)$ is a distributed time-dependent forcing, and the scalar parameter ν acts like a viscosity. The boundary conditions are periodic and the initial condition $u(x,0) = u_0(x)$ will be specified later. We choose $L = 1$, $T = 10$, and $\nu = 0.01$.

We discretize the Burgers equation (11) with a spectral method using $n = 256$ Fourier modes, and integrate forward in time using a fifth-order Runge-Kutta method. Solution snapshots are saved every $\Delta t = 0.05$ time unit, yielding the discrete high-dimensional state

$$\boldsymbol{z}_k = [u(x_0, k\Delta t), u(x_1, k\Delta t), \ldots, u(x_{n-1}, k\Delta t)]^\mathsf{T} \in \mathbb{R}^n, \tag{12}$$

where $x_i = iL/n$ are the collocation points, and $k = 0, \ldots, T/\Delta t$ is the time index. Further, we place $p = 8$ sensors at equidistant locations, providing the sparse measurement vector

$$\boldsymbol{y}_k = [u(\bar{x}_0, k\Delta t), u(\bar{x}_1, k\Delta t), \ldots, u(\bar{x}_{p-1}, k\Delta t)]^\mathsf{T} \in \mathbb{R}^p, \tag{13}$$

where $\bar{x}_i = iL/p$ are the sensor locations. Thus, the state $\boldsymbol{z}_k$ and measurement $\boldsymbol{y}_k$ are governed by a high-dimensional discrete-time nonlinear system of the form given in equation (1).

We then follow the procedure outlined in Section 2 to construct offline a ROM and corresponding ROE, which we train offline using PPO following the methodology presented in Section 4. Finally, the trained RL-ROE is deployed online and compared against a Kalman filter under various initial conditions for the reference trajectory as well as the state estimate. Specifically, we adopt the following steps:

1. **Construction of the ROM.** Starting from the pulse-shaped initial condition

$$u_0(x) = \frac{1}{\cosh(20(x - L/2))}, \tag{14}$$

we calculate one solution trajectory of (1) for $t \in [0, T/2] = [0, 5]$, and we denote as $\boldsymbol{Z}^{\text{DMD}} = \{\boldsymbol{z}_0^{\text{DMD}}, \ldots, \boldsymbol{z}_m^{\text{DMD}}\}$ the resulting solution snapshots at times $k = 0, \ldots, m = T/2\Delta t$. The DMD is then applied to these snapshots, yielding a ROM of the form (2) defined by matrices $\boldsymbol{A}_r$, $\boldsymbol{C}_r$ and $\boldsymbol{U}$. The ROM governs the evolution of a reduced-order state $\boldsymbol{x}_k \simeq \boldsymbol{U}^\mathsf{T}\boldsymbol{z}_k \in \mathbb{R}^r$. We pick $r = 15$ for the dimensionality of the reduced-order subspace (which corresponds in this case to 99.99% of the energy of the snapshots $\boldsymbol{Z}^{\text{DMD}}$ being included in the modes $\boldsymbol{U}$), giving the ROM a significant computational advantage compared with the high-dimensional system (1) of size $n = 256$.

2. **Training of the RL-ROE.** We train the stochastic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ of the RL-ROE (3) using PPO, as described in Section 4. In order for the resulting estimator to perform well under various reference trajectories and initial estimates, we initialize each trajectory of the MDP during the offline training process with

$$\boldsymbol{z}_0 = \alpha\boldsymbol{z}_0^{\text{DMD}}, \tag{15a}$$

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{U}^\mathsf{T}\boldsymbol{z}_0^{\text{DMD}} + \boldsymbol{\beta}, \tag{15b}$$

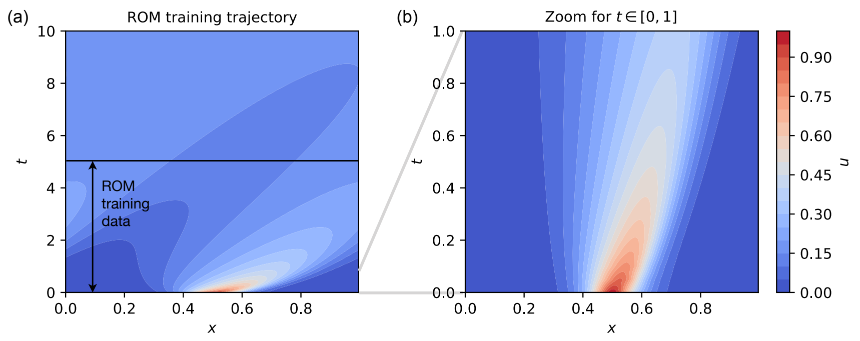

Figure 1: (a) Training trajectory of the unforced Burgers equation used for the construction of the ROM with DMD. The discrete-time snapshots in $t \in [0, 5]$ constitute the training data fed to the DMD. The tracking performance of the RL-ROE will be evaluated for $t \in [0, 10]$ on trajectories with similar dynamical behavior as this one. (b) Zooming into $t \in [0, 1]$ reveals the fast decay of the pulse-shaped initial condition (14).

where $\alpha \sim \mathcal{U}(0.5, 2) \in \mathbb{R}$ and $\boldsymbol{\beta} \sim \mathcal{N}(0, 0.1\mathbf{I}) \in \mathbb{R}^r$, which defines the distributions given in (10). In other words, the reference trajectories are initialized as a randomly scaled up or scaled down version of the pulse defined in (14), and the reduced-order state estimate is initialized as a reduced-order projection of that pulse, polluted with additive Gaussian noise. During training, we limit each trajectory to the same time window $t \in [0, T/2]$ that was used in constructing the ROM – that is, we pick $K = T/2\Delta t$ in the finite-horizon return (9). We end the training when the return no longer increases on average.

3. **Evaluation of the RL-ROE.** We evaluate the trained RL-ROE 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 (3a) and (4), with the calculation of the time-varying Kalman gain detailed in Appendix B. The RL-ROE and KF-ROE are compared online based on three specific reference trajectories initialized from (15a) using $\alpha = 0.5$, 1, and 2. For each reference trajectory, we consider 20 different initial state estimates sampled from (15b), and feed measurements $\boldsymbol{y}_k$ to the RL-ROE and KF-ROE. Their tracking performance of the reference state $\boldsymbol{z}_k$ is then evaluated and compared over the full time window $t \in [0, T]$.

We carry out the above procedure for two different choices of the forcing $f(x, t)$, each leading to qualitatively different dynamics: first, the unforced case $f(x, t) = 0$, and second, a sinusoidal forcing of the form $f(x, t) = \sin(\omega t - kx)$ with $\omega = \pi$ and $k = 2\pi/L$. We emphasize that the RL-ROE is not (yet) designed to account for different exogenous control inputs to the system. Here, the forcing $f(x, t)$ is simply considered as a way to generate more complex solutions by altering the dynamics of the PDE. Thus, the two choices for $f(x, t)$ will be treated separately in the results below, and we will construct and train a different RL-ROE in each case.

5.1 UNFORCED CASE

Beginning with the unforced case $f(x, t) = 0$, we first construct the ROM on which the RL-ROE and KF-ROE will be based. Figure 1 depicts the solution of the discretized Burgers equation to the initial condition (14). The solution snapshots belonging to the time window $t \in [0, T/2] = [0, 5]$ constitute the training data $\boldsymbol{Z}^{\mathrm{DMD}}$ fed to the DMD algorithm, which yields the ROM (2). Further, this figure illustrates the type of dynamics exhibited by the Burgers equation in the unforced regime: a quickly decaying and widening pulse for $t \leq 2$, followed by a slow convergence to a uniform steady state for $t > 2$.

Before training and evaluating the RL-ROE, we quantify the accuracy of the ROM itself – that is, given knowledge of the true initial condition. For this purpose, we consider three initial conditions defined by (15a) with $\alpha = 0.5$, 1, 2, and we compute the corresponding reference solution (using the discretized Burgers equation) as well as the ROM solution. The results, reported in Appendix C, show that the model error is very low for the initial condition $\alpha = 1$ and time window $t \in [0, 5]$,

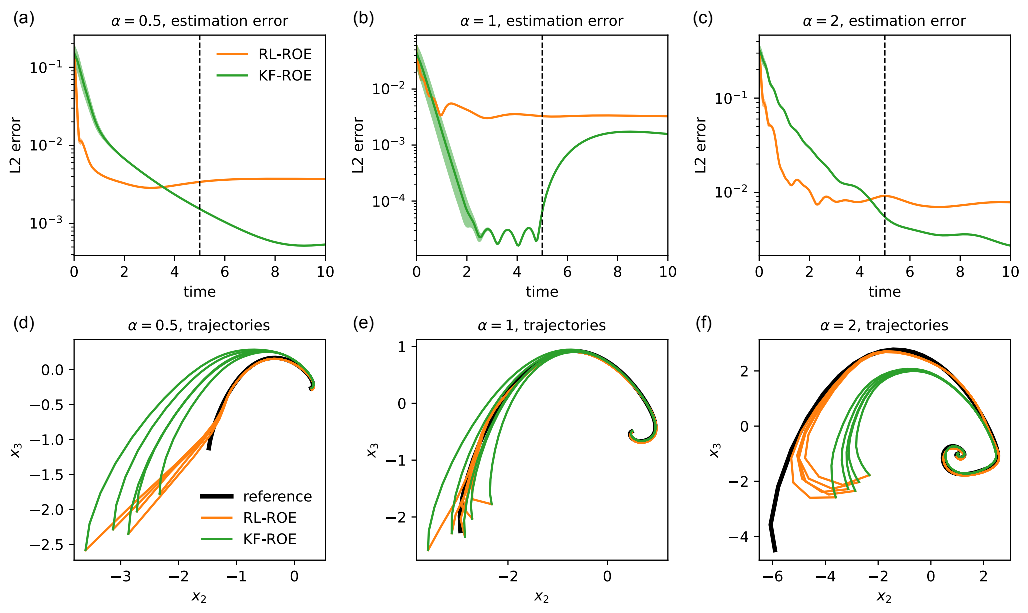

Figure 2: Accuracy of the RL-ROE for the unforced case. (a,b,c) L2 error of the RL-ROE and KF-ROE with respect to specific reference trajectories of the Burgers equation, initialized from (15a) using $\alpha = 0.5, 1, 2$. The RL-ROE and KF-ROE are evaluated using 20 different initial estimates sampled from (15b). The curves and shaded area indicate the mean and standard deviation of the error, respectively. (Appendix E shows the same data for $t \in [0, 2]$.) (d,e,f) Phase space trajectories of the RL-ROE and KF-ROE predictions for 5 initial estimates, and the reference solution.

since this case corresponds to the same solution trajectory used for constructing the ROM. On the other hand, the model error increases for larger times or other values of α.

The RL-ROE is then trained using PPO following the methodology outlined in Section 4, together with the distributions (15) for trajectory initialization. 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 is now compared with the KF-ROE, a Kalman filter constructed from the same ROM. Figures 2(a,b,c) show the L2 error of the RL-ROE and KF-ROE with respect to specific reference trajectories of the Burgers equation, initialized from (15a) using $\alpha = 0.5$, 1, 2. For each reference trajectory, we consider 20 different initial state estimates sampled from (15b). The curves and shaded area reported in Figures 2(a,b,c) indicate the mean and standard deviation of the error, defined at time step k by $|\boldsymbol{U}\hat{\boldsymbol{x}}_k - \boldsymbol{z}_k|$, where $\hat{\boldsymbol{x}}_k$ is the reduced-order estimate given by the RL-ROE or KF-ROE, and $\boldsymbol{z}_k$ is the high-dimensional reference solution. (Appendix E shows the same data for $t \in [0, 2]$.) Figures 2(d,e,f) show the trajectories of the reference, RL-ROE and KF-ROE solutions in a two-dimensional slice of phase space spanned by the second and third columns[1] of $\boldsymbol{U}$ – in other words, the time history of the second and third components of $\boldsymbol{U}^\mathsf{T}\boldsymbol{z}_k$ and $\hat{\boldsymbol{x}}_k$.

A few important observations emerge from Figure 2. First, when the ROM suffers from large model errors due to initial conditions deviations, as is the case for $\alpha = 0.5$ and $\alpha = 2$, the RL-ROE is able to outperform the KF-ROE. In the time window $t \leq 2$ during which most of the transient dynamics take place, the RL-ROE displays up to an order of magnitude lower error than the KF-ROE. Second, when the ROM is very accurate, as is the case for $\alpha = 1$, the KF-ROE gives lower error for most of the time duration. Even then, however, Figure 2(e) shows that the RL-ROE converges faster to the reference trajectory. Last, the RL-ROE manages to keep the error at a low level in the time window $t \in [5, 10]$, despite the fact that it was trained using trajectories that end at $t = 5$.

[1]Since the columns of $\boldsymbol{U}$ approximate the PCA modes of the training snapshots $\boldsymbol{Z}^{\text{DMD}}$ without centering, the first column will be dominated by the mean of the data. Thus, we display the trajectory coordinates associated with the second and third columns, which capture the largest amount of variance within the data.

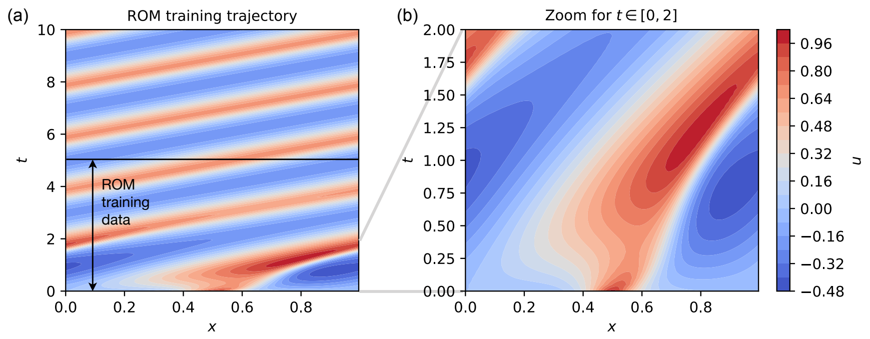

Figure 3: (a) Training trajectory of the forced Burgers equation used for the construction of the ROM with DMD. The discrete-time snapshots in $t \in [0, 5]$ constitute the training data fed to the DMD. The tracking performance of the RL-ROE will be evaluated for $t \in [0, 10]$ on trajectories with similar dynamical behavior as this one. (b) Zooming into $t \in [0, 1]$ reveals the fast decay of the pulse-shaped initial condition (14).

The Burgers equation converges to a uniform steady state in this unforced case, which can explain why both the RL-ROE and KF-ROE display low errors at larger times. We now investigate a fundamentally different dynamical regime in the next section by adding nonzero forcing to the Burgers equation, which prevents the solution from settling on a steady state.

5.2 Forced case

We now consider the forced case $f(x, t) = \sin(\omega t - kx)$, with $\omega = \pi$ and $k = 2\pi/L$. Repeating the steps of the previous section, we first construct the ROM on which the RL-ROE and KF-ROE will be based. Figure 3 depicts the solution of the discretized Burgers equation with forcing to the initial condition (14). The solution snapshots belonging to the time window $t \in [0, T/2] = [0, 5]$ constitute the training data $\boldsymbol{Z}^{\mathrm{DMD}}$ fed to the DMD algorithm for constructing the ROM (2). This figure illustrates the type of dynamics exhibited by the Burgers equation in the forced regime: a transient phase with for $t \leq 2$ during which the pulse changes shape into a skewed wave, followed by a limit cycle regime with the wave traveling at constant velocity for $t > 2$.

As in the unforced case, we first quantify the accuracy of the ROM itself. For this purpose, we consider three initial conditions defined by (15a) with $\alpha = 0.5$, 1, 2, and we compute the corresponding reference solution (using the discretized Burgers equation with forcing) as well as the ROM solution. This time, the results reported in Appendix C show that the model error is higher for all cases, which reflects the more complicated nature of the dynamics in this forced regime.

The RL-ROE is then trained using PPO following the methodology outlined in Section 4, together with the distributions (15) for trajectory initialization. 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 is now compared with the KF-ROE, a Kalman filter constructed from the same ROM. Figures 4(a,b,c) show the L2 error of the RL-ROE and KF-ROE with respect to specific reference trajectories of the Burgers equation, initialized from (15a) using $\alpha = 0.5$, 1, 2. As before, we consider 20 different initial state estimates sampled from (15b) for each reference trajectory. The curves and the surrounding shade reported in Figures 4(a,b,c) indicate the mean and standard deviation of the resulting error, respectively. (Appendix E shows the same data for $t \in [0, 2]$.) Figures 4(d,e,f) show the trajectories of the reference, RL-ROE and KF-ROE solutions in a two-dimensional slice of phase space spanned by the second and third columns of $\boldsymbol{U}$ – in other words, the time history of the second and third components of $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{z}_k$ and $\hat{\boldsymbol{x}}_k$.

The RL-ROE outperforms the KF-ROE in both the initial transient phase as well as the later limit cycle regime, for all three reference trajectories. This is consistent with our previous observation that the RL-ROE has an advantage over the KF-ROE when the ROM suffers from large model errors. Moreover, the estimation performance of the RL-ROE remains stable in the time window $t \in [5, 10]$, even though trajectories stop at $t = 5$ during the training process. All together, these

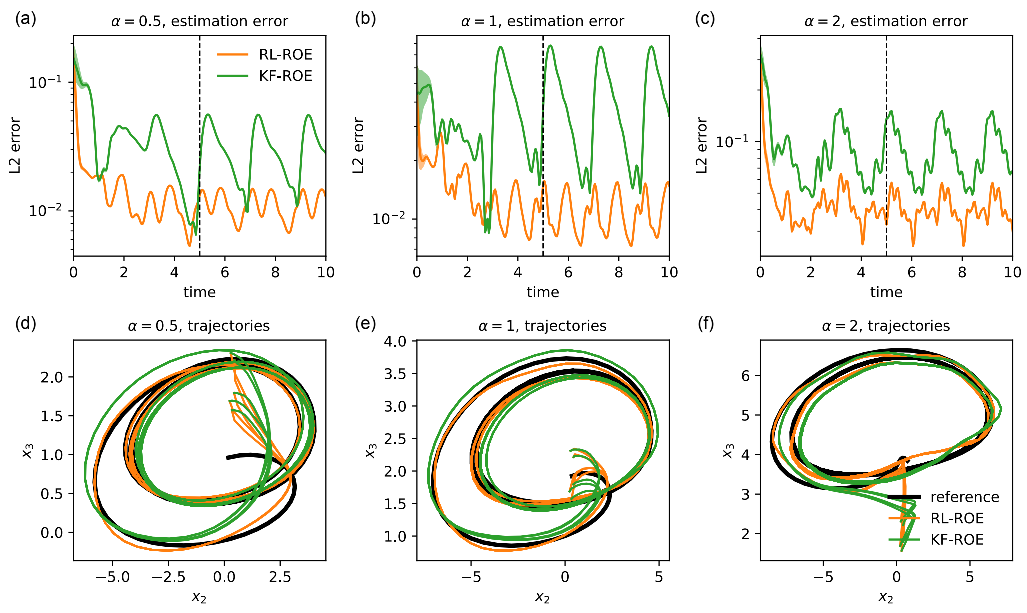

Figure 4: Accuracy of the RL-ROE for the forced case. (a,b,c) L2 error of the RL-ROE and KF-ROE with respect to specific reference trajectories of the Burgers equation, initialized from (15a) using $\alpha = 0.5, 1, 2$. The RL-ROE and KF-ROE are evaluated using 20 different initial estimates sampled from (15b). The curves and the surrounding shade indicate the mean and standard deviation of the error, respectively. (Appendix E shows the same data for $t \in [0, 2]$.) (d,e,f) Phase space trajectories of the RL-ROE and KF-ROE predictions for 5 initial estimates, and the reference solution.

results demonstrate the estimation performance of the RL-ROE, and its robustness with respect to model errors.

We include a number of additional results as appendices. In Appendix F, we show that the regularization term in the reward function (7) leads to better estimation performance in the presence of observation noise. In Appendix G, we compare our RL-ROE approach with two alternative data-driven estimators formulated as standard supervised learning problems, and we observe that the RL-ROE is more robust to observation noise and gives smoother predictions.

6 Conclusions

In this paper, we have introduced the reinforcement learning reduced-order estimator (RL-ROE), a new methodology for estimating the state of a high-dimensional nonlinear dynamical system. Our approach follows the standard practice of constructing 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 feedback correction term is given by a nonlinear stochastic policy trained through reinforcement learning. To be able to use off-the-shelf RL algorithms, we introduce a trick to translate this trajectory tracking problem, i.e., time-varying MDP, to an equivalent stationary MDP based on an augmented state. We show using simulations of the Burgers equation in two very different dynamical regimes that the trained RL-ROE is able to outperform a Kalman filter designed using the same ROM and displays robust estimation performance with respect to different reference trajectories and initial state estimates.

This work opens the door to a number of potential future directions. A logical next step is to evaluate the performance of the RL-ROE on other physical systems, for instance the Navier-Stokes equations governing the motion of fluid flows. Different types of ROM could also be considered, potentially leading to improved performance of the RL-ROE.

REFERENCES

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.

Mouhacine Benosman and Jeff Borggaard. Robust nonlinear state estimation for a class of infinite-dimensional systems using reduced-order models. *International Journal of Control*, 94(5), 2021.

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.

Petr Chaupa, Jakub Novák, and Peter Januška. Model predictive control using different state observers,. In *Recent Advances in Automatic Control, Information and Communications, pages=191–196, year=2013*.

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.

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

J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

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

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.

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.

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.

Paul Zarchan and Howard Musoff. *Fundamentals of Kalman filtering: a practical approach*. Aiaa, 2015.

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 (2) given a snapshot sequence $\boldsymbol{Z} = \{\boldsymbol{z}_0, \ldots, \boldsymbol{z}_m\}$ collected along a trajectory of (1a) and an observation model $\boldsymbol{C}$.

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 $\boldsymbol{z}_{k+1} \simeq \boldsymbol{A}\boldsymbol{z}_k$. Arranging the snapshots into two time-shifted matrices

$$\boldsymbol{X} = \{\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{m-1}\}, \quad \boldsymbol{Y} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}, \tag{16}$$

the best-fit linear model is 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

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} \tag{17}$$

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

$$\boldsymbol{A}_r = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U} = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}, \tag{18a}$$
$$\boldsymbol{C}_r = \boldsymbol{C}\boldsymbol{U}. \tag{18b}$$

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 Kalman filter

The time-dependent Kalman filter that we use as a benchmark in this paper, KF-ROE, is based on the same ROM (2) 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 (3a), 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 (3a), which is instead given by the linear feedback term (4). The calculation of the optimal Kalman gain $\boldsymbol{K}_k$ in (4) requires the following operations at each time step:

$$\boldsymbol{P}_k^- = \boldsymbol{A}_r\boldsymbol{P}_{k-1}\boldsymbol{A}_r^{\mathsf{T}} + \boldsymbol{Q}_k, \tag{19}$$
$$\boldsymbol{S}_k = \boldsymbol{C}_r\boldsymbol{P}_k^-\boldsymbol{C}_r^{\mathsf{T}} + \boldsymbol{R}_k, \tag{20}$$
$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{C}_r^{\mathsf{T}}\boldsymbol{S}_k^{-1}, \tag{21}$$
$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{C}_r)\boldsymbol{P}_k^-, \tag{22}$$

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 (2). Since these noise covariance matrices are unknown, we choose $\boldsymbol{Q}_k = \boldsymbol{R}_k = \boldsymbol{I}$ for all k after verifying empirically that these values yield the best possible results. 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 initial reference and estimated state distributions (15).

Remark 3. We are seeking a Kalman-type observer, which in ‘filters domain’ is also known as an infinite impulse response filter (IIR). These observers are to be contrasted with the finite impulse filters (FIR). Indeed, the later are well known to be based on a mapping between n previous samples of input/output and the desired observed state at the current instant (e.g., Chaupa et al., equations 2,3, and 12), and lead to exact convergence in finite-time, in the noiseless setting. On the other hand the IIR observer is well known to be based on the last measurement of the output/input only, e.g., (Chaupa et al., equation 8), and lead to an average asymptotic performance, e.g.,maximum likelihood estimate in the our case.

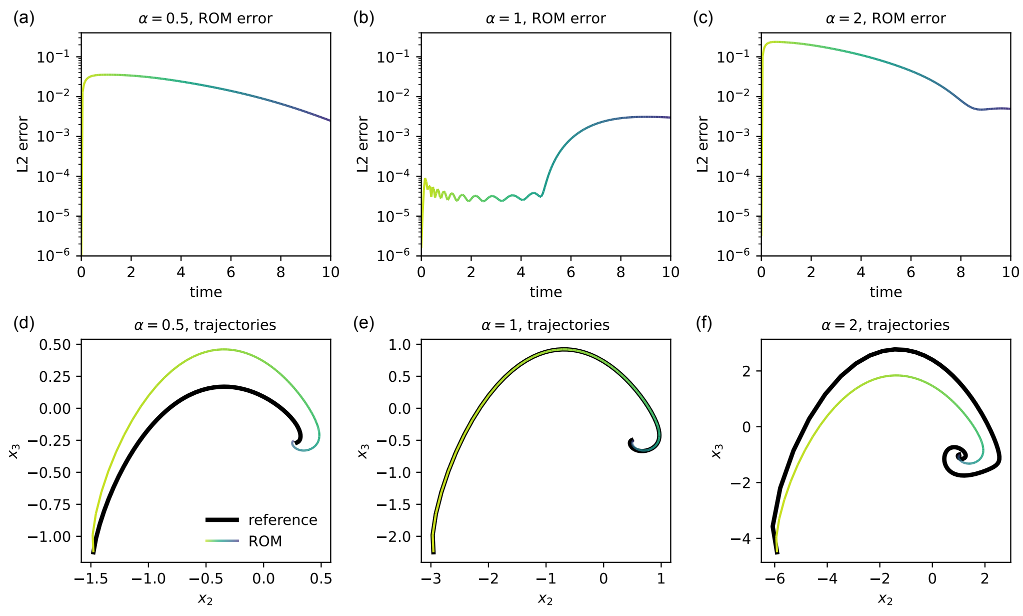

Figure 5: Accuracy of the ROM for the unforced case. (a,b,c) L2 error of the ROM prediction with respect to a reference numerical solution of the Burgers equation, both initialized from (15a) using $\alpha = 0.5, 1, 2$. The case $\alpha = 1$ corresponds to the solution trajectory whose snapshots in $t \in [0, 5]$ were used to construct the ROM. (d,e,f) Phase space trajectories of the ROM prediction and the reference solution.

C Model error

In this appendix, we quantify the accuracy of the two ROMs utilized in Sections 5.1 and 5.2 for the unforced and forced Burgers equations, respectively. For this purpose, we consider three initial conditions defined by (15a) with $\alpha = 0.5, 1, 2$, and we compute the corresponding reference solution (using the discretized Burgers equation) as well as the ROM solution. Results pertaining to the unforced and forced cases are shown in Figures 5 and 6, respectively.

Figures 5(a,b,c) and 6(a,b,c) show the resulting L2 error of the ROM solution, defined at each time step k by $|\boldsymbol{U}\boldsymbol{x}_k - \boldsymbol{z}_k|$, where $\boldsymbol{x}_k$ is the reduced-order state given by the ROM and $\boldsymbol{z}_k$ is the high-dimensional reference solution. Figures 5(d,e,f) and 6(d,e,f) show the trajectories of the reference and ROM solutions in a two-dimensional slice of phase space spanned by the second and third columns of $\boldsymbol{U}$ – in other words, the time history of the second and third components of $\boldsymbol{U}^\top \boldsymbol{z}_k$ and $\boldsymbol{x}_k$. The L2 error curves and ROM trajectory curves are colored according to the value of time using the same color scheme, which confirms that most of the dynamics take place during the first two time units, as already observed in Figure 1 for $\alpha = 1$.

D Training hyperparameters and learning curves

The stochastic policy $\boldsymbol{\pi_\theta}$ is trained with PPO using the default hyperparameters from Stable Baselines3, except for the discount factor γ which we choose as 0.75. The mean output of the stochastic policy and the value function are approximated by two neural networks, each containing two hidden layers with 64 neurons and tanh activation functions. The training process alternates between sampling data for 20 trajectories (of length 100 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 one million timesteps, corresponding to 10000 trajectories. Figure 7 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 10 separate test trajectories, and is saved if it outperforms the previous best policy.

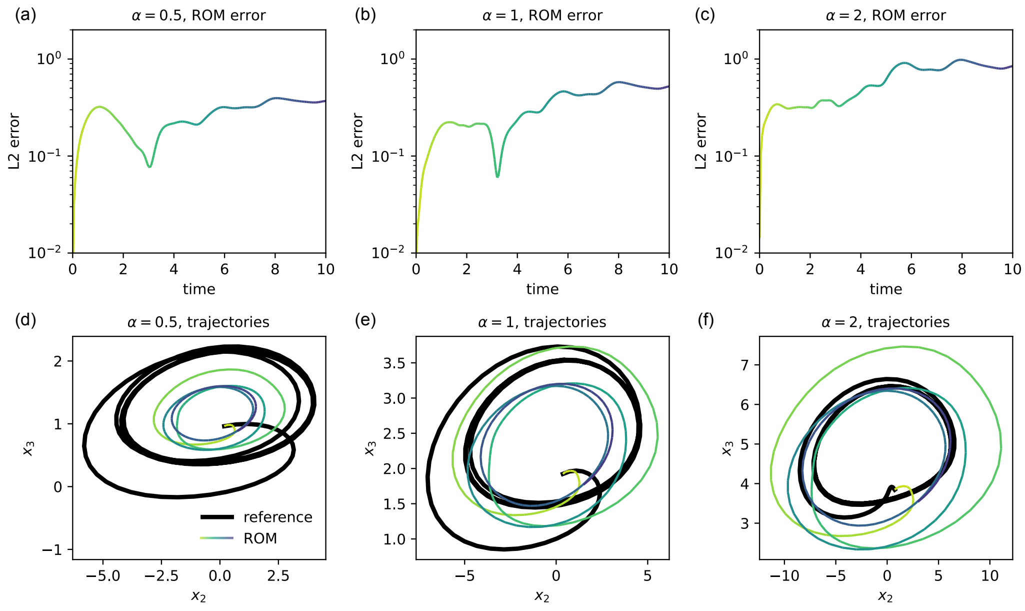

Figure 6: Accuracy of the ROM for the forced case. (a,b,c) L2 error of the ROM prediction with respect to a reference numerical solution of the Burgers equation, both initialized from (15a) using $\alpha = 0.5, 1, 2$. The case $\alpha = 1$ corresponds to the solution trajectory whose snapshots in $t \in [0, 5]$ were used to construct the ROM. (d,e,f) Phase space trajectories of the ROM prediction and the reference solution.

Finally, the RL-ROE is defined using 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.

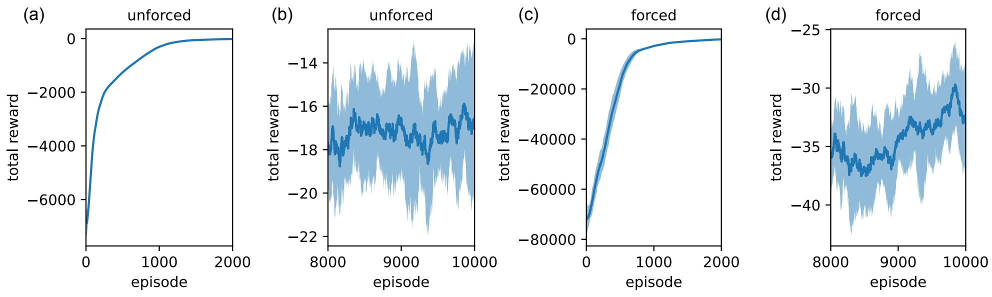

Figure 7: Learning curves for the stochastic policy in the (a,b) unforced and (c,d) forced cases. The line and shaded area show the mean and standard deviation of the results over 10 runs, each one smoothed with a moving average of size 100 episodes.

E Estimation error for short times

Figures 8 and 9 show the same data as the first row in Figures 2 and 4, but for $t \in [0, 2]$. Focusing on this initial transient phase, it becomes clear that the variance of the RL-ROE estimate decays much faster than the KF-ROE estimate.

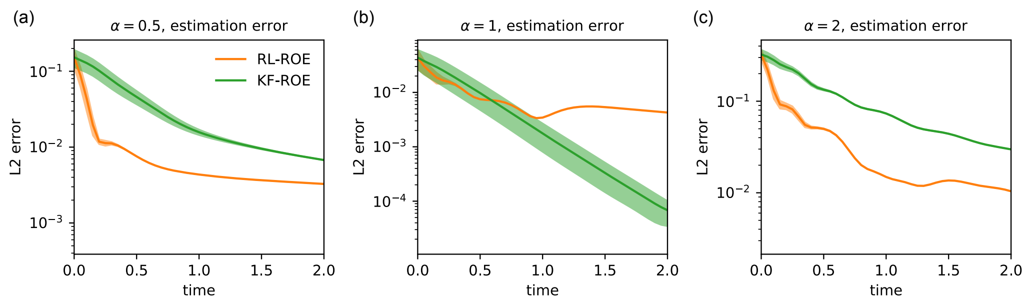

Figure 8: Same as the first row of Figure 2, but shown for $t \in [0, 2]$.

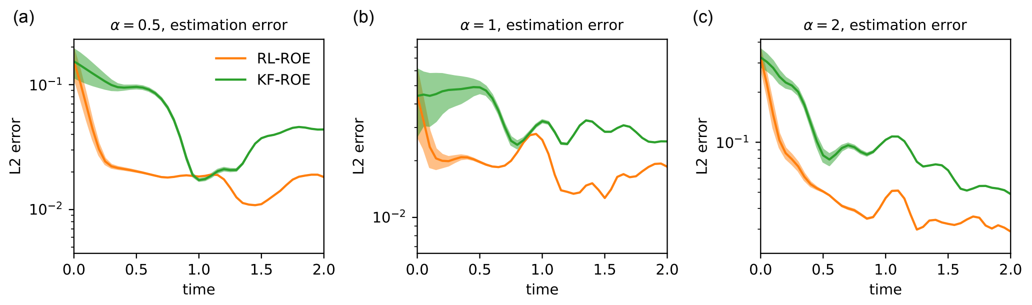

Figure 9: Same as the first row of Figure 4, but shown for $t \in [0, 2]$.

F Effect of penalizing the action magnitude in the reward

We investigate the effect of penalizing the magnitude of the action $\boldsymbol{a}_k$ in the reward function (7). To this effect, we repeat the experiments of Sections 5.1 and 5.2, this time training the RL-ROE using different magnitudes R for the matrix $\boldsymbol{R} = R\boldsymbol{I}$. When evaluating the performance of the trained RL-ROE, we consider different amounts of Gaussian observation noise added to the measurements $\boldsymbol{y}_k$. The results for the unforced case are shown in Figures 10, 11, and 12 for observation noise of standard deviation $\sigma = 0$, 0.1, and 0.3, respectively. The results for the forced case are shown in Figures 13, 14, and 15 for observation noise of standard deviation $\sigma = 0$, 0.1, and 0.3, respectively. In absence of noise, the highest estimation accuracy is obtained for $R = 0$, and decreases as R increases. However, in the presence of noise, Figures 11, 12, and 14, 15, the estimation accuracy is generally highest for $R = 10$. This confirms that penalizing the magnitude of the action $\boldsymbol{a}_k$ in the reward function acts as a regularization that allows the RL-ROE to perform better on noisy measurement data.

G Comparison with two supervised learning approaches

In this appendix, we compare our proposed RL-ROE with two alternative estimation techniques trained offline in a supervised learning setting.

The first approach is purely data-driven. Here, the estimator takes the form

$$\hat{\boldsymbol{z}}_k = \boldsymbol{U} f_{\mathrm{DNN}}(\boldsymbol{y}_k, \ldots, \boldsymbol{y}_{k-q}), \tag{23}$$

where the nonlinear function $f_{\mathrm{DNN}} : \mathbb{R}^{p \times q} \to \mathbb{R}^r$ is a feed-forward deep neural network (DNN) taking a given number q of past observations as input. The DNN is then trained by minimizing the loss given for each data point by the squared estimation error $||\hat{\boldsymbol{z}}_k - \boldsymbol{z}_k||^2$, where $\boldsymbol{z}_k$ is the ground-truth high-dimensional state. The entire data set consists of the same high-dimensional reference trajectories used for training the RL-ROE – that is, they are initialized from the distribution (15a) and are limited to the time window $t \in [0, T/2] = [0, 5]$. This results in a total of 14646 data points for all trajectories and time indices. The DNN consists of 4 hidden layers of 50 neurons each, and

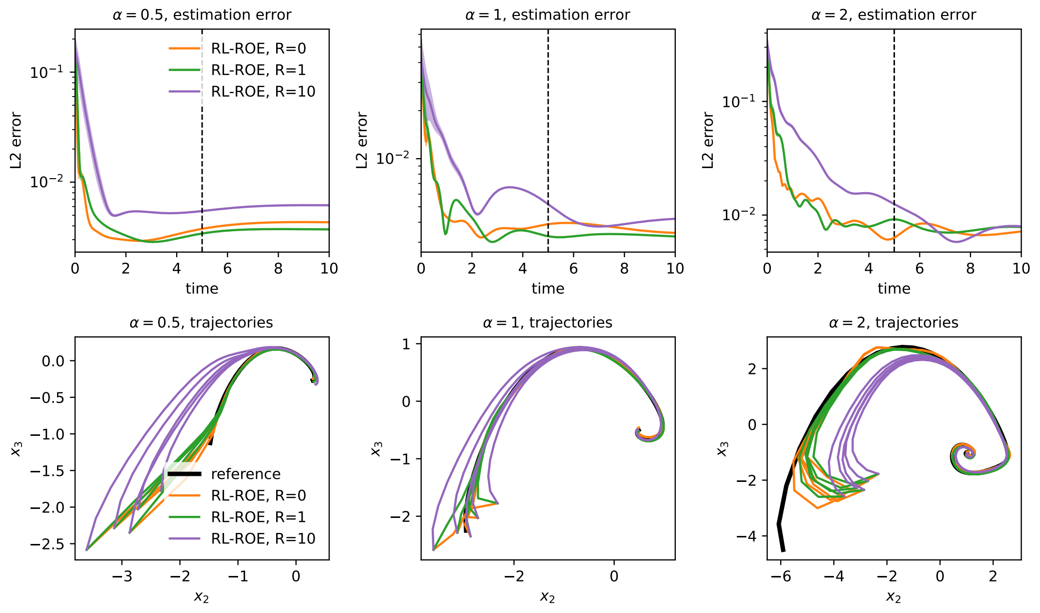

Figure 10: Effect of $\boldsymbol{R}$ on the accuracy of the RL-ROE for the unforced case. Same as Figure 2, but comparing several RL-ROEs trained using different magnitudes R for the matrix $\boldsymbol{R} = R\boldsymbol{I}$ in the reward function (7).

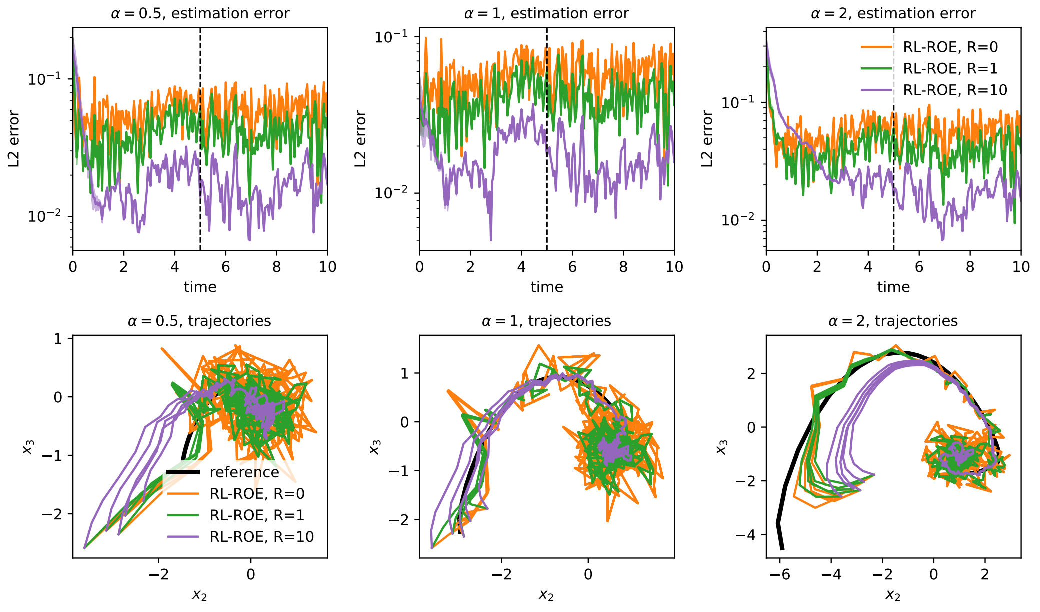

Figure 11: Effect of $\boldsymbol{R}$ on the accuracy of the RL-ROE for the unforced case with moderate noise. Same as Figure 2, but comparing several RL-ROEs trained using different magnitudes R for the matrix $\boldsymbol{R} = R\boldsymbol{I}$ in the reward function (7), and with independent Gaussian observation noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$.

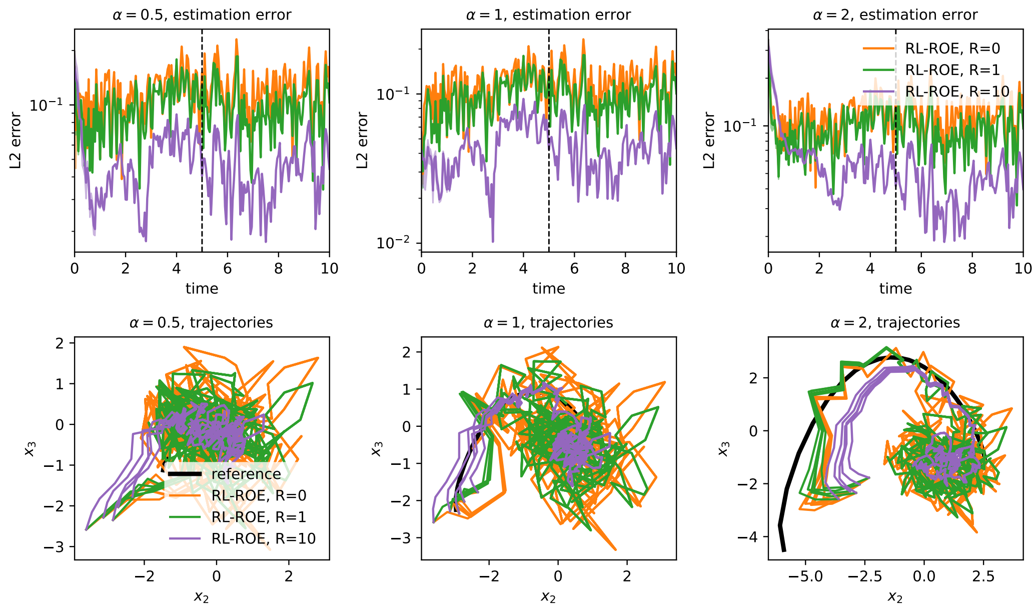

Figure 12: Effect of $\boldsymbol{R}$ on the accuracy of the RL-ROE for the unforced case with large noise. Same as Figure 2, but comparing several RL-ROEs trained using different magnitudes R for the matrix $\boldsymbol{R} = R\boldsymbol{I}$ in the reward function (7), and with independent Gaussian observation noise of standard deviation $\sigma = 0.3$ added to the measurements $\boldsymbol{y}_k$.

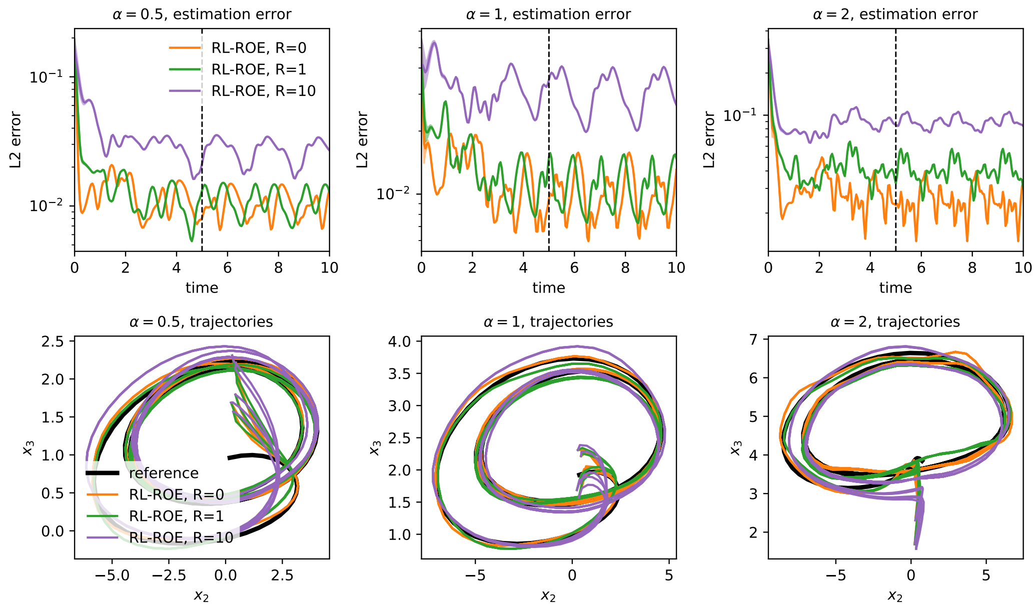

Figure 13: Effect of $\boldsymbol{R}$ on the accuracy of the RL-ROE for the forced case. Same as Figure 4, but comparing several RL-ROEs trained using different magnitudes R for the matrix $\boldsymbol{R} = R\boldsymbol{I}$ in the reward function (7).

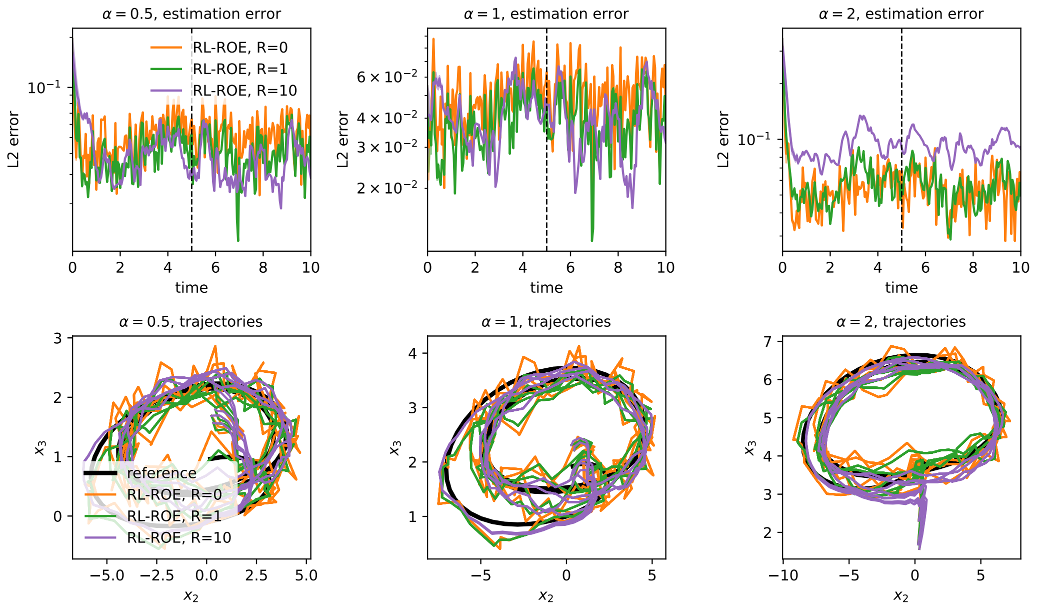

Figure 14: Effect of $\boldsymbol{R}$ on the accuracy of the RL-ROE for the forced case with moderate noise. Same as Figure 4, but comparing several RL-ROEs trained using different magnitudes R for the matrix $\boldsymbol{R} = R\boldsymbol{I}$ in the reward function (7), and with independent Gaussian observation noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$.

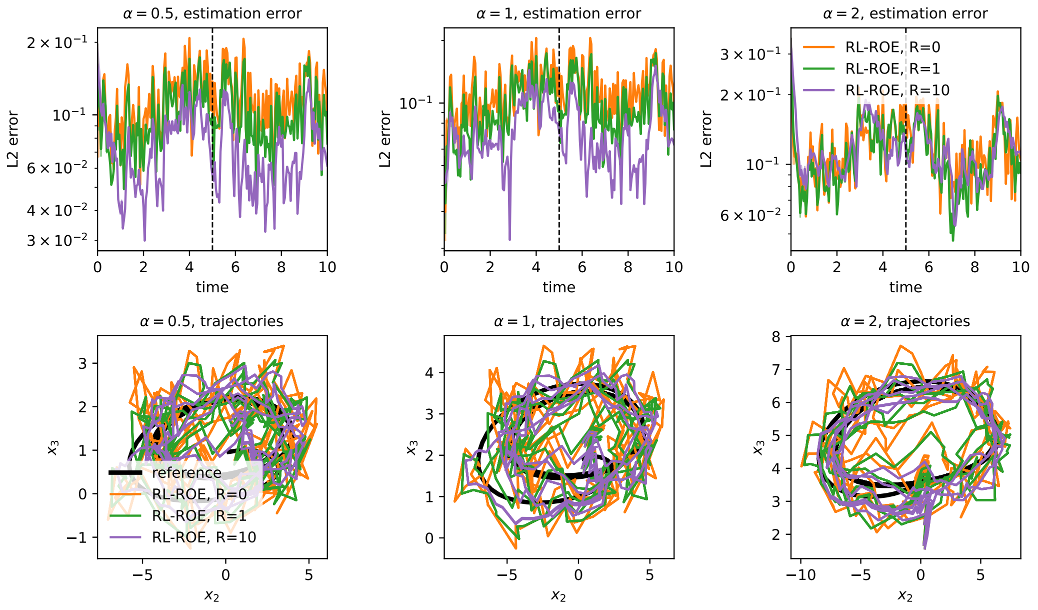

Figure 15: Effect of $\boldsymbol{R}$ on the accuracy of the RL-ROE for the forced case with large noise. Same as Figure 4, but comparing several RL-ROEs trained using different magnitudes R for the matrix $\boldsymbol{R} = R\boldsymbol{I}$ in the reward function (7), and with independent Gaussian observation noise of standard deviation $\sigma = 0.3$ added to the measurements $\boldsymbol{y}_k$.

is trained using the Adam optimizer over 1000 epochs with minibatch size of 256. The resulting estimator is denoted DNN in the following results.

The second approach is a hybrid ROM-based and data-driven approach, similarly to our proposed RL-ROE. Here, the estimator takes the form

$$\hat{\boldsymbol{z}}_k = \boldsymbol{U}(\hat{\boldsymbol{x}}_k^{\mathrm{ROM}} + f_{\mathrm{DNN}}(\hat{\boldsymbol{x}}_k^{\mathrm{ROM}}, \boldsymbol{y}_k)), \tag{24}$$

where $\hat{\boldsymbol{x}}_k^{\mathrm{ROM}}$ is the reduced state generated by the raw ROM (in our case, the DMD model), and the nonlinear function $f_{\mathrm{DNN}} : \mathbb{R}^{r+p} \to \mathbb{R}^r$ is a DNN acting as a nonlinear correction to the ROM prediction. The DNN is then trained by minimizing the loss given for each data point by the squared estimation error $||\hat{\boldsymbol{z}}_k - \boldsymbol{z}_k||^2$, where $\boldsymbol{z}_k$ is the ground-truth high-dimensional state. The entire data set consists of pairwise combinations of high-dimensional reference trajectories and reduced ROM trajectories. Similar to the training process for the RL-ROE, the reference and ROM trajectories are initialized from the distributions (15a) and (15b), and are limited to the time window $t \in [0, T/2] = [0, 5]$. This results in a total of 305020 data points for all pairs of trajectories and time indices. The DNN consists of 4 hidden layers of 80 neurons each, and is trained using the Adam optimizer over 500 epochs with minibatch size of 256. The resulting estimator is denoted ROM+DNN in the following results.

We compare in Figures 16 and 17 the estimation performance of the RL-ROE with that of the DNN estimator (23) using $q = 3$. Since the DNN estimator essentially acts as an interpolation function, it is more interesting to compare them in the presence of observation noise, which we model as Gaussian noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$. The RL-ROE produces more accurate estimates with smoother trajectories; such robustness to measurement noise is a consequence of the stochasticity inherent to the RL training process. Furthermore, a clear benefit of the RL-ROE design philosophy is that it yields an accurate dynamic model, since the imperfect ROM dynamics (in our case given by DMD) are corrected by the RL-trained nonlinear policy $\boldsymbol{a}_k \sim \boldsymbol{\pi}$, unlike the DNN estimator design which has no inherent dynamics. This will later allow the RL-ROE to be paired with model-based controllers, opening the door to a wide class of control strategies that are commonly used with Kalman filters.

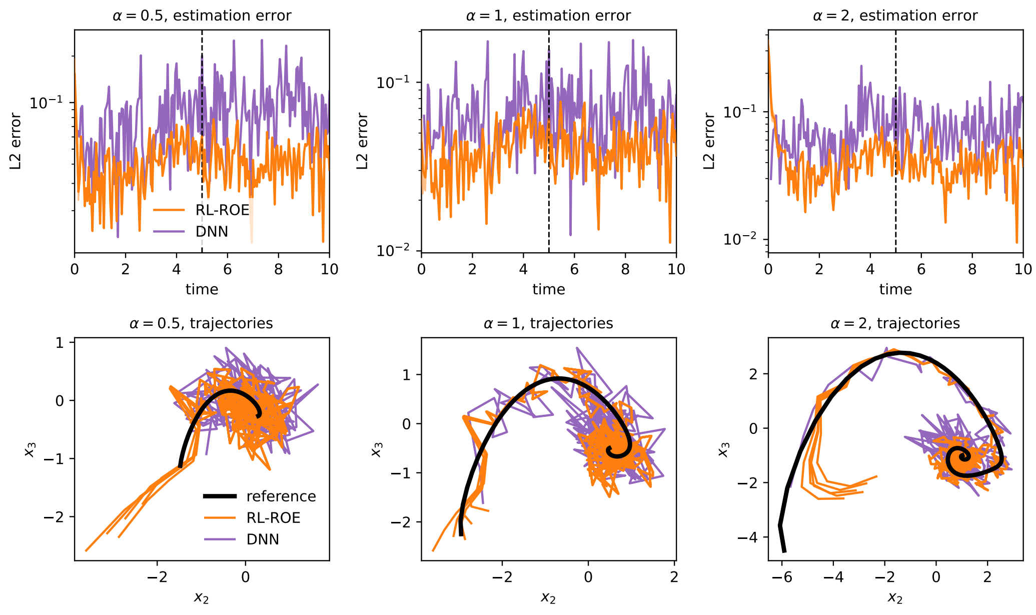

Figure 16: Accuracy of the DNN estimator for the unforced case with noise. Same as Figure 2, but comparing the RL-ROE with the DNN estimator, and with independent Gaussian observation noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$.

We compare in Figures 18 and 19 the estimation performance of the RL-ROE with that of the ROM+DNN estimator (24). The comparison is also performed in the presence of Gaussian observation noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$. Once more, the RL-ROE

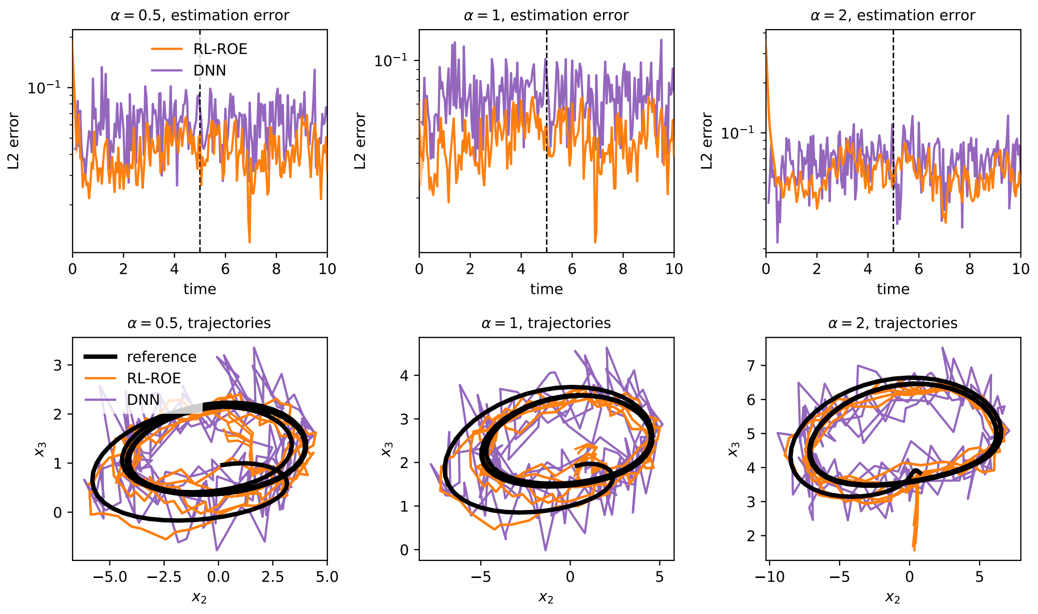

Figure 17: Accuracy of the DNN estimator for the forced case with noise. Same as Figure 4, but comparing the RL-ROE with the DNN estimator, and with independent Gaussian observation noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$.

produces more accurate estimates with smoother trajectories, especially in the forced case. Finally, we show in Figure 20 that in the absence of observation noise, the RL-ROE has better generalization abilities once the time goes beyond the window $t \in [0, 5]$ seen during training. These comparisons suggest that the RL-ROE design, in which the policy directly corrects the ROM dynamics and is trained in an RL setting that accounts for errors compounding over time, is more robust to data not seen during training and to measurement noise.

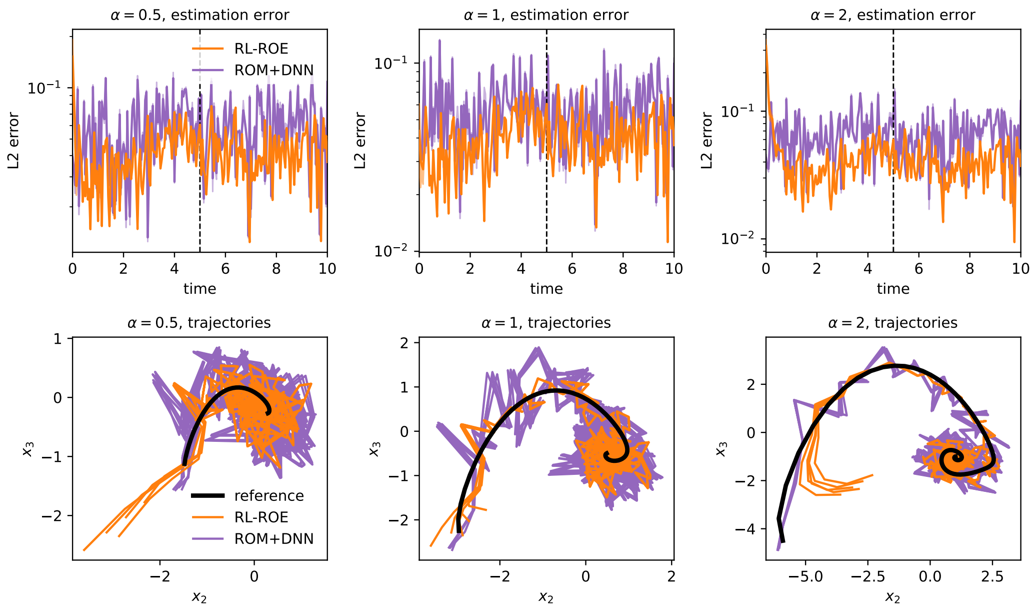

Figure 18: Accuracy of the ROM+DNN estimator for the unforced case with noise. Same as Figure 2, but comparing the RL-ROE with the ROM+DNN estimator, and with independent Gaussian observation noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$.

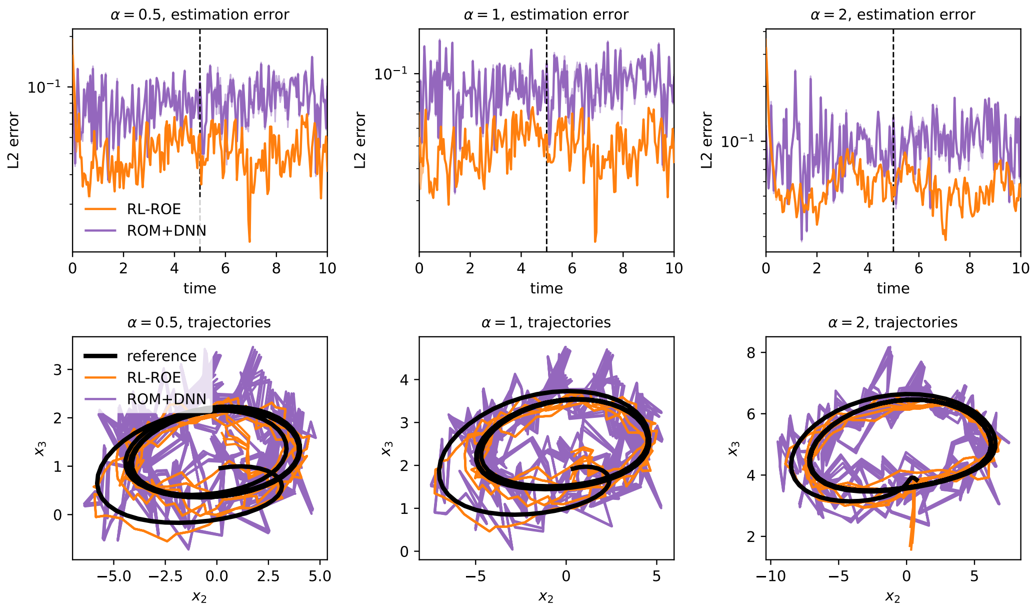

Figure 19: Accuracy of the ROM+DNN estimator for the forced case with noise. Same as Figure 4, but comparing the RL-ROE with the ROM+DNN estimator, and with independent Gaussian observation noise of standard deviation $\sigma = 0.1$ added to the measurements $\boldsymbol{y}_k$.

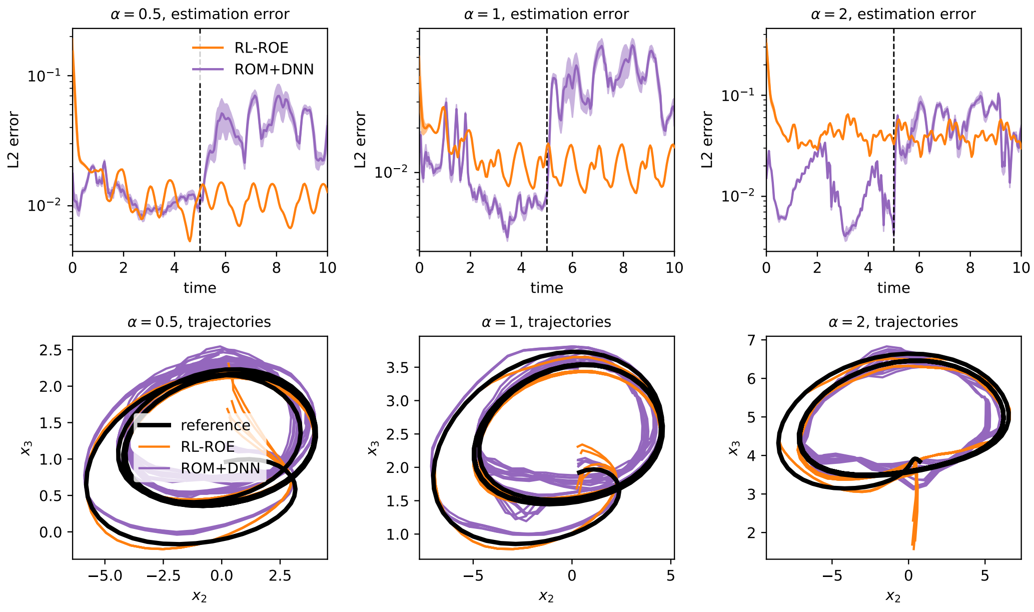

Figure 20: Accuracy of the ROM+DNN estimator for the forced case. Same as Figure 4, but comparing the RL-ROE with the ROM+DNN estimator.