MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Reinforcement Learning-Based Estimation for Spatio-Temporal Systems

Mowlavi, Saviz; Benosman, Mouhacine

TR2024-134 October 02, 2024

Abstract

State estimators such as Kalman filters compute an estimate of the instantaneous state of a dynamical system from sparse sensor measurements. For spatio-temporal systems, whose dynamics are governed by partial differential equations (PDEs), state estimators are typically designed based on a reduced-order model (ROM) that projects the original high-dimensional PDE 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 with parametric uncertainties, we show that in the limit of very few sensors, the trained RL-ROE outperforms a Kalman filter designed using the same ROM and yields accurate instantaneous estimates of high-dimensional states corresponding to unknown initial conditions and physical parameter values. The RL-ROE opens the door to lightweight real-time sensing of systems governed by parametric PDEs.

Nature Scientific Reports 2024

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Reinforcement Learning-Based Estimation for Spatio-Temporal Systems

Saviz Mowlavi[1,*] **and Mouhacine Benosman**[1,*]

[1]Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA
[*]mowlavi@merl.com, benosman@merl.com

ABSTRACT

State estimators such as Kalman filters compute an estimate of the instantaneous state of a dynamical system from sparse sensor measurements. For spatio-temporal systems, whose dynamics are governed by partial differential equations (PDEs), state estimators are typically designed based on a reduced-order model (ROM) that projects the original high-dimensional PDE 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 with parametric uncertainties, we show that in the limit of very few sensors, the trained RL-ROE outperforms a Kalman filter designed using the same ROM and yields accurate instantaneous estimates of high-dimensional states corresponding to unknown initial conditions and physical parameter values. The RL-ROE opens the door to lightweight real-time sensing of systems governed by parametric PDEs.

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[1]. But real-time feedback control requires inferring the instantaneous state of the system from sparse measurements using a state estimation algorithm, which typically relies on a model of the underlying dynamics[2,3]. Among state estimators, the Kalman filter and its nonlinear variants such as the extended or unscented Kalman filters have seen widespread use in numerous applications[4,5]. For continuous spatio-temporal systems such as fluid flows, however, the governing partial differential equations (PDEs) yield high-dimensional discretized models that are too expensive to integrate with common model-based state estimation techniques, especially in the context of embedded systems. Thus, a common practice is to design state estimators using a reduced-order model (ROM) of the system, in which the underlying dynamics are projected to a low-dimensional subspace that is computationally tractable[6–8]. For example, several recent studies have demonstrated the potential of constructing a Kalman filter based on a data-driven ROM to estimate unsteady fluid flows using sparse measurements[9–11].

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[12] or nonlinear subspaces[13]. 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 based on 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) using a supervised dataset. 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 hybrid model-based and data-driven state estimator for high-dimensional parametric PDEs.

General methodology

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. We assume that the dynamics given in (1) are obtained from a high-fidelity numerical discretization of a nonlinear partial differential equation (PDE), which yields a high-dimensional state $\boldsymbol{z}_k$ with $n \gg 1$. Furthermore, the measurements are typically acquired by a small number of sensors, hence they are sparse in the sense that $p \ll n$.

The purpose of the present work is 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 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 setting is different from the forecasting problem solved by operator learning methods, where μ and $\boldsymbol{z}_0$ are both known[14,15]. In general, such a state estimator takes the form

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

where $\mathscr{E}$ is the map from measurements to state estimate that we seek.

In this paper, the estimator $\mathscr{E}$ is formulated and trained in an initial offline phase using a training dataset $\boldsymbol{Z}_{\text{train}} = \{\boldsymbol{Z}^\mu, \boldsymbol{Y}^\mu\}_{\mu\in S}$ of trajectories of (1) for various 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). It is typical in data-driven state estimation to assume that such data is available[16], either through offline simulations of the high-dimensional system (1) or by advanced visualization techniques such as particle image velocimetry in fluid mechanics[17]. Once trained, the estimator $\mathscr{E}$ can be deployed online to produce state estimates $\hat{\boldsymbol{z}}_k$ in real time, using sensor measurements $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\}$ from trajectories of (1) corresponding to unknown and previously unobserved initial states $\boldsymbol{z}_0$ and parameter values μ.

Overview of the methodology

The proposed methodology to design $\mathscr{E}$ offline, using the dataset $\boldsymbol{Z}_{\text{train}}$, consists of two steps illustrated in Figure 1. In a first step, we construct a data-driven reduced-order model (ROM) of the high-dimensional dynamics (1). The construction of the ROM, which follows standard practices, is described in the "Reduced-order model" section. In a second step, we formulate $\mathscr{E}$ as a reduced-order estimator (ROE) based on the ROM constructed in the first step, which we then train with RL using the trajectories contained in $\boldsymbol{Z}_{\text{train}}$. The formulation of $\mathscr{E}$ based on the ROM and its training using RL, which constitute the main novelty of the paper, are described in the "Reinforcement learning-based reduced-order estimator" section. Note that besides the availability of the training dataset $\boldsymbol{Z}_{\text{train}}$, the formulation and training of $\mathscr{E}$ do not require knowledge of the system (1).

Reduced-order model

A popular approach to construct a ROM is first to choose a suitable set of orthonormal 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 define a reduced-order state $\boldsymbol{x}_k = \boldsymbol{U}^\mathsf{T}\boldsymbol{z}_k \in \mathbb{R}^r$ representing the subspace coordinates of $\boldsymbol{z}_k$, which can itself be reconstructed as $\boldsymbol{z}_k \simeq \boldsymbol{U}\boldsymbol{x}_k$. Finally, one identifies a model 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 dynamics model for $\boldsymbol{x}_k$[18]. 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[19,20]. Importantly, we seek a single ROM to describe dynamics corresponding to various parameter values $\mu \in [\mu_1, \mu_2]$ by applying the DMD to all 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 $\boldsymbol{z}_{k+1}^\mu \simeq \boldsymbol{A}\boldsymbol{z}_k^\mu$ for all k and μ, and computes the modes $\boldsymbol{U}$ as the r leading principal component analysis (PCA) modes of $\{\boldsymbol{Z}^\mu\}_{\mu\in S}$. The transformation $\boldsymbol{z}_k = \boldsymbol{U}\boldsymbol{x}_k$ and the orthogonality of $\boldsymbol{U}$ then yield the linear discrete-time ROM

$$\boldsymbol{x}_k = \boldsymbol{A}_r\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}, \tag{3a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}_r\boldsymbol{x}_k + \boldsymbol{v}_k, \tag{3b}$$

$$\boldsymbol{z}_k = \boldsymbol{U}\boldsymbol{x}_k, \tag{3c}$$

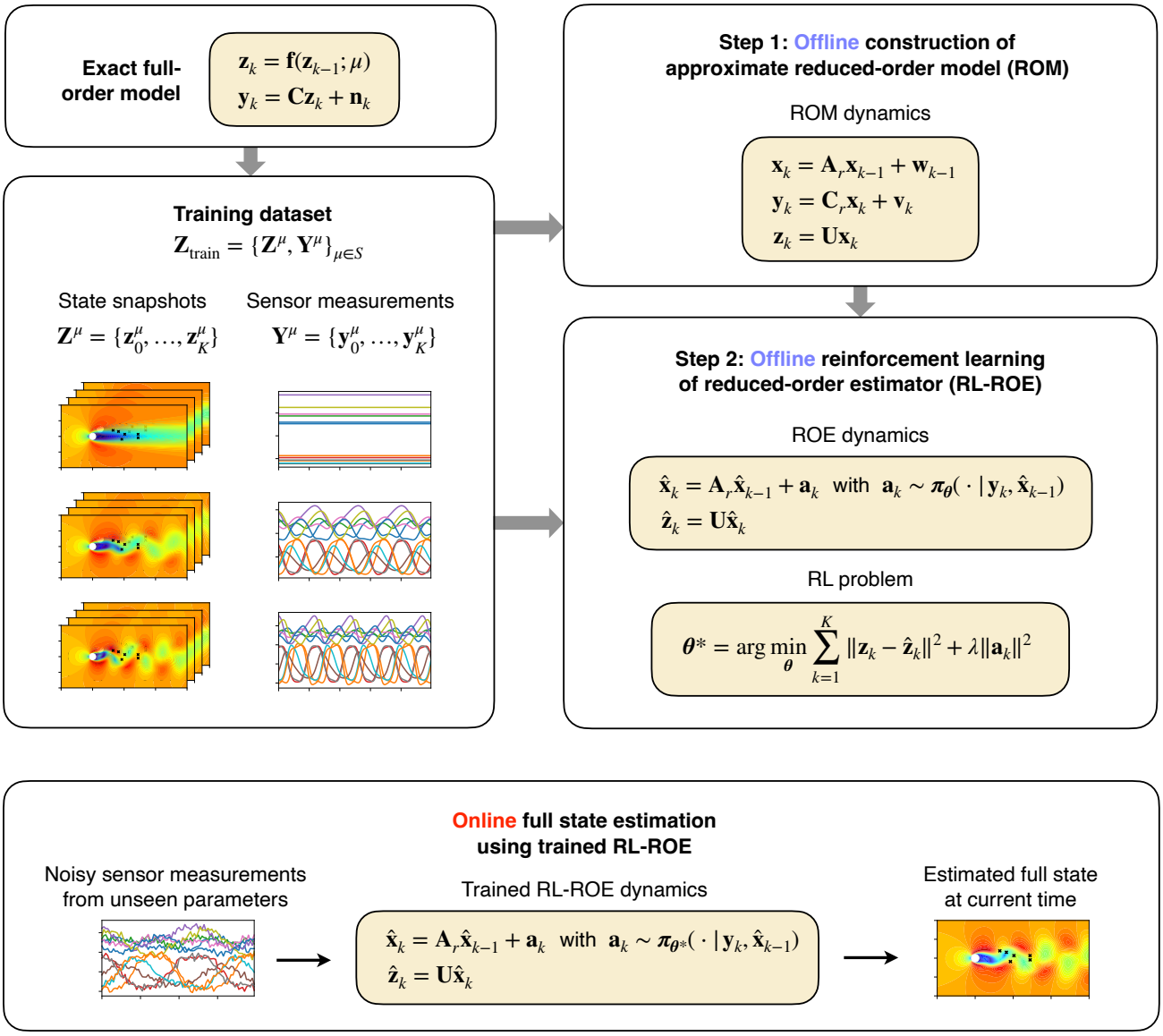

Figure 1. Overview of the proposed RL-ROE methodology.

where $\boldsymbol{A}_r = \boldsymbol{U}^\top \boldsymbol{A}\boldsymbol{U} \in \mathbb{R}^{r\times r}$ 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 the "Dynamic Mode Decomposition" section of the Methods.

Reinforcement learning-based reduced-order estimator

Using the ROM (3), we can now formulate the state estimator $\mathscr{E}$ defined in the "Problem formulation" section. 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 correct 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 $\boldsymbol{z}_k$, starting 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 estimation and control[2,21,22]. 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 $\{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1}\}$ 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[2,23], 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 note that unlike the Kalman filter, the ROE dynamics (4) does not output the covariance of the state estimate. However, one can still evaluate the accuracy of the state estimate by computing the magnitude of the measurement residual (also called the innovation) $\boldsymbol{y}_k - \boldsymbol{C}_r\hat{\boldsymbol{x}}_k$, which quantifies the difference between the observed and the predicted measurements. Finally, previous studies[24–26] have also proposed hybrid state estimators that combine model-based and data-driven components, but these studies were focused on low-dimensional systems.

In an initial offline phase, we use deep RL to train the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ by solving the optimization problem

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

which minimizes the mean square error between the high-dimensional estimate $\hat{\boldsymbol{z}}_k$ and the true state $\boldsymbol{z}_k$. 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 using the trajectories contained in the training dataset $\boldsymbol{Z}_{\text{train}}$, which span several parameter values μ and initial conditions $\boldsymbol{z}_0$. By considering different values of μ during training, a strategy called domain randomization[27], we ensure robustness of the policy with respect to μ during online deployment of the estimator. 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. The "Offline training methodology" section in Methods describes the RL training procedure for $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, which involves a non-trivial reformulation of the problem into a stationary Markov decision process. We call the estimator constructed and trained through this process an RL-trained ROE, or RL-ROE for short.

The RL-ROE can be interpreted through a Bayesian lens by noting that the optimization problem (6) seeks to minimize the mean square error (MSE) of the estimate $\boldsymbol{z}_k$. Such minimum mean square error (MMSE) estimate is equal to the mean of the posterior distribution $p(\boldsymbol{z}_k|\boldsymbol{y}_{1:k})$ of the hidden true state $\boldsymbol{z}_k$ conditioned on past measurements $\boldsymbol{y}_{1:k} = \{\boldsymbol{y}_1,\dots,\boldsymbol{y}_k\}$[3,28]. However, instead of directly solving for the full posterior distribution $p(\boldsymbol{z}_k|\boldsymbol{y}_{1:k})$, the RL-ROE is trained during the offline phase to directly minimize the MSE within the class of recursive estimators parameterized by (4). Finally, note 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 implicitly learns their distributions during the offline training phase from the trajectories of states and measurements contained in the training dataset $\boldsymbol{Z}_{\text{train}}$.

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. The trained RL-ROE is finally deployed online to estimate trajectories corresponding to unseen initial conditions and parameter values. The state estimates are compared against a time-dependent Kalman filter constructed from the same ROM, referred to as KF-ROE. The KF-ROE is given by equations (4a, 5, 4c), with the calculation of the time-varying Kalman gain detailed in the "Kalman filter" section of Methods. The noise covariances in the KF-ROE were tuned using line search to obtain the best possible results[16].

It is worth mentioning that the ensemble Kalman filter and 4D-Var are two estimation techniques for high-dimensional systems such as those governed by PDEs[29]. 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 require estimators

designed based on 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[23,30] are not relevant, and the vanilla Kalman filter remains the best choice of baseline.

Burgers' equation

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

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = f(x,t), \tag{7}$$

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

$$f(x,t) = 2\sin(\omega t - kx) + 2\sin(3\omega t - kx) + 2\sin(5\omega t - kx), \tag{8}$$

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

$$\nu = \nu_1 + (\nu_2 - \nu_1)\mu, \qquad \omega = \omega_1 + (\omega_2 - \omega_1)\mu. \tag{9a}$$

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 (the construction of the corresponding observation matrix $\boldsymbol{C}$ is detailed in the section "Observation matrix" of Methods). 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 algorithm and hyperparameters are reported in the "RL algorithm and hyperparameters" section of Methods.

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 testing values of μ not present in the training dataset $\boldsymbol{Z}_{\text{train}}$, 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**a** reports the mean (lines) and standard deviation (shaded areas) of the normalized L_2 error for 3 testing values of μ. 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 $\boldsymbol{U}$, that is, $\|\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{z}_k - \boldsymbol{z}_k\|/\|\boldsymbol{z}_k\|$. Spatio-temporal contours of the ground-truth state trajectories for the same 3 values of μ and the corresponding RL-ROE and KF-ROE estimates are shown in Figure 2**b**. The RL-ROE using either policy vastly outperforms the KF-ROE, which demonstrates the superiority of a nonlinear correction to the estimator dynamics (4a).

Figure 2**c** (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 2**c** (right) displays the average over time and over μ of the normalized L_2 error for varying numbers of sensors p, where 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. Indeed, spatio-temporal contours of the ground-truth state and corresponding estimates for $p = 2$ and 12 in Figure A1 of the Supplementary Materials show that the slight advantage held by the KF-ROE for $p = 12$ is reversed into clear superiority of the RL-ROE for $p = 2$.

Figure A2 of the Supplementay Materials present additional results analyzing the performance of the RL-ROE verus KF-ROE in the absence of parametric uncertainties. When considering a single value of μ, both the RL-ROE and KF-ROE

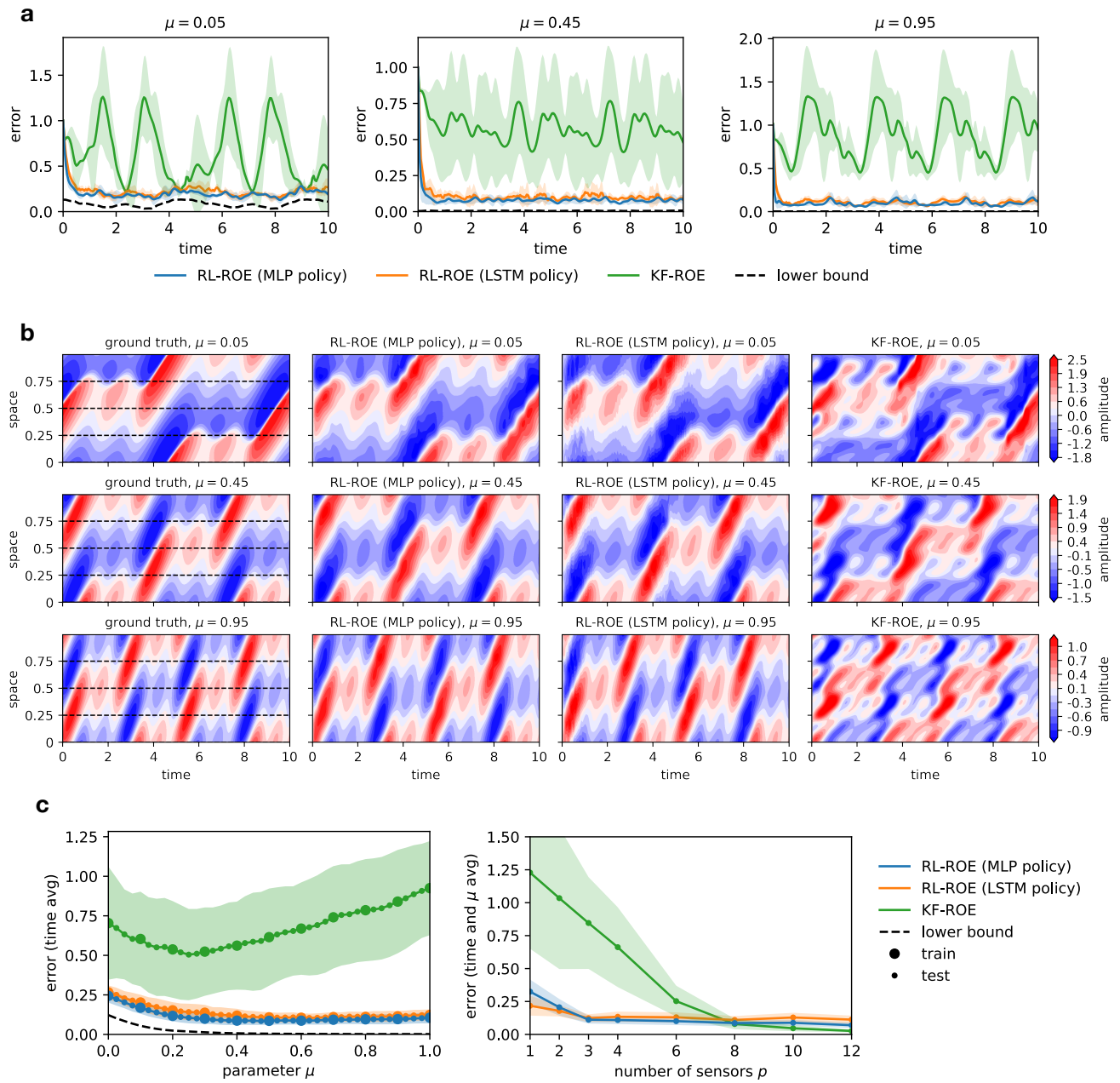

Figure 2. Burgers' equation. **a**, Normalized L_2 error of the RL-ROE and KF-ROE state estimates for values of μ not seen during training, using $p = 4$ sensors. **b**, Trajectories of the ground-truth state for values of μ not seen during training and corresponding RL-ROE and KF-ROE estimates using $p = 4$ sensors. The dashed lines on the ground-truth trajectory plots indicate the sensor data seen by the RL-ROE and KF-ROE. **c**, 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. Right: Normalized L_2 error, averaged over time and over the testing values of μ, versus number of sensors p. In **a** and **c**, the error metrics are averaged over 5 trajectories with randomly-sampled initial true states $\boldsymbol{z}_0$, and the shaded areas denote the standard deviation.

yield more accurate estimates than when μ varies and is unknown. However, the nonlinearity of the correction term in the RL-ROE and the data-driven nature of the RL training process lets the RL-ROE compensate for modeling errors in the ROM, ultimately resulting in better estimation performance than the KF-ROE. Finally, Figure A3 of the Supplementay Materials present additional results analyzing the performance obtained when the RL-ROE dynamics (4a) is formulated without the ROM dynamics term $\boldsymbol{A}_r\hat{\boldsymbol{x}}_{k-1}$. Not only are the estimates generated by the RL-ROE without ROM dynamics less accurate, the training process also takes on the order of 10 times longer to converge. This illustrates the benefits of informing the RL-ROE estimator dynamics with prior knowledge on the system dynamics through the ROM.

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

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla p + \frac{1}{Re}\Delta\boldsymbol{u}, \tag{10a}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{10b}$$

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 $Re_c \sim 40$[31]. 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 (the construction of the corresponding observation matrix $\boldsymbol{C}$ is detailed in the section "Observation matrix" of Methods). 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 Figure A2 of the Supplementary Materials. 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 algorithm and hyperparameters are reported in the "RL algorithm and hyperparameters" section of Methods.

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 values of Re not present in the training dataset $\boldsymbol{Z}_{\text{train}}$, 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 3**a** reports the mean (lines) and standard deviation (shaded areas) of the normalized L_2 error for 3 testing values of Re. 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 large. 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 3**b**. 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. Figures A4 and A5 of the Supplementary Materials shows corresponding results in the presence of non-zero observation noise.

Figure 3**c** (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 3**c** (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 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

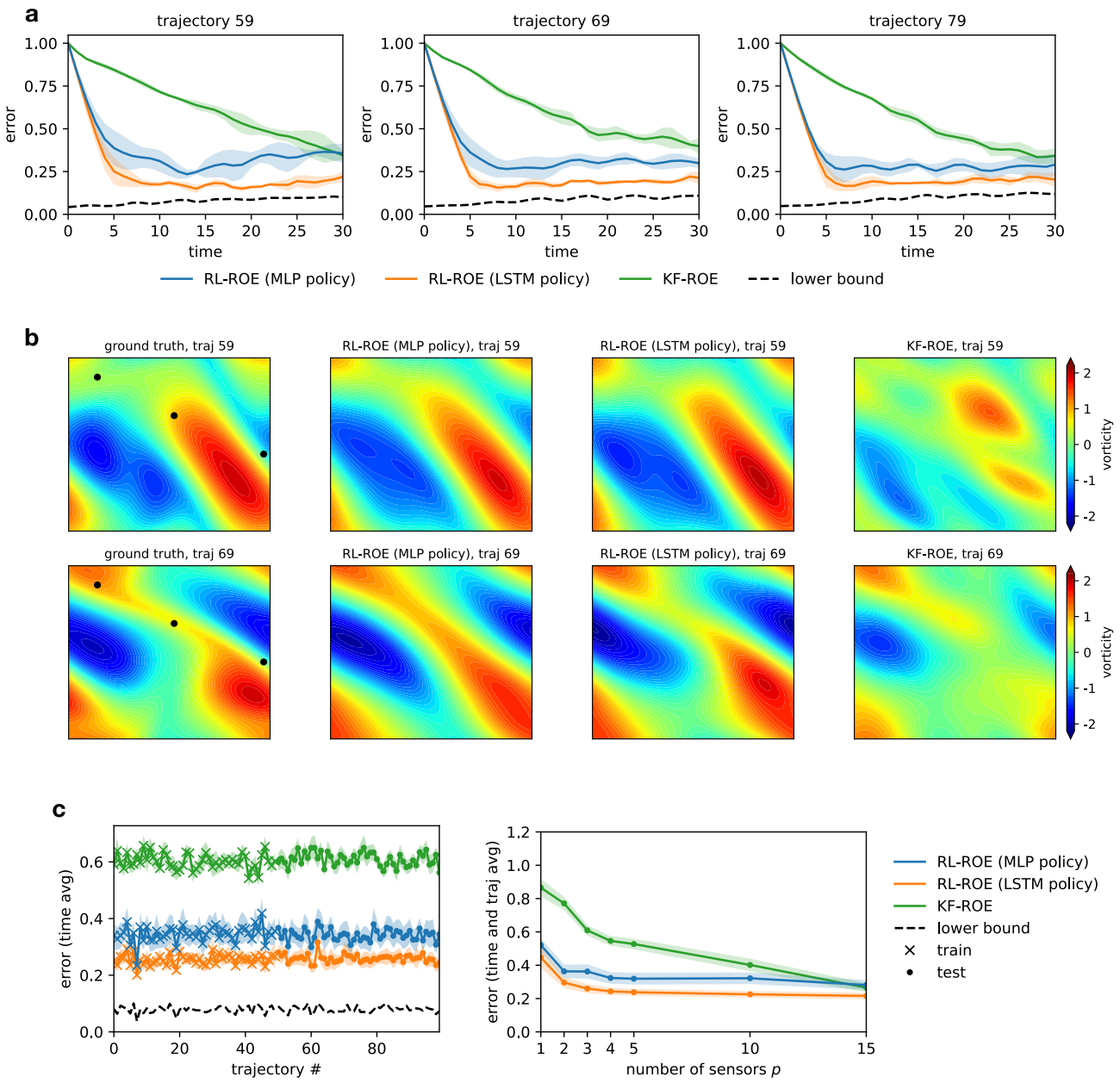

Figure 3. Flow past a cylinder. **a**, Normalized L_2 error of the RL-ROE and KF-ROE state estimates for values of Re not seen during training, using $p = 3$ sensors. **b**, Ground-truth velocity magnitude at $t = 50$ for values of Re not seen during training and corresponding RL-ROE and KF-ROE estimates using $p = 3$ sensors. The black crosses in the contours of the reference solutions indicate the sensor locations. **c**, Left: Normalized L_2 error, averaged over time, versus μ when using $p = 3$ sensors. Values of μ belonging to the training set S are shown by large circles while the testing values are displayed by small circles. Right: Normalized L_2 error, averaged over time and over the testing values of Re, versus number of sensors p. In **a** and **c**, the error metrics are averaged over 5 trajectories with randomly-sampled initial true states $\mathbf{z}_0$, and the shaded areas denote the standard deviation.

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.

Navier-Stokes equations: chaotic vorticity flow

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

$$\frac{\partial \omega}{\partial t} + \boldsymbol{u} \cdot \nabla \omega = \frac{1}{Re} \Delta \omega, \tag{11a}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{11b}$$

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[32] 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 11a 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[32]. 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[32], 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 describes the corresponding $\boldsymbol{C}$). We retain $r = 20$ modes when constructing the ROM. These modes are shown in Figure A3 of the Supplementary Materials. 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 algorithm and hyperparameters are reported in the "RL algorithm and hyperparameters" section of Methods.

The trained RL-ROE and the KF-ROE are now compared based on their ability to track various trajectories of $\boldsymbol{z}_k$ not present in the training dataset, 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 4**a** reports the mean (lines) and standard deviation (shaded areas) of the normalized L_2 error for 3 testing 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 4**b**. 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 4**c** (left) reports the time average of the normalized L_2 error for each trajectory for $p = 3$. Figure 4**c** (right) displays the average over time and over all testing trajectories of the normalized L_2 error for 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.

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 Figure A6 in the Supplementary Materials 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.

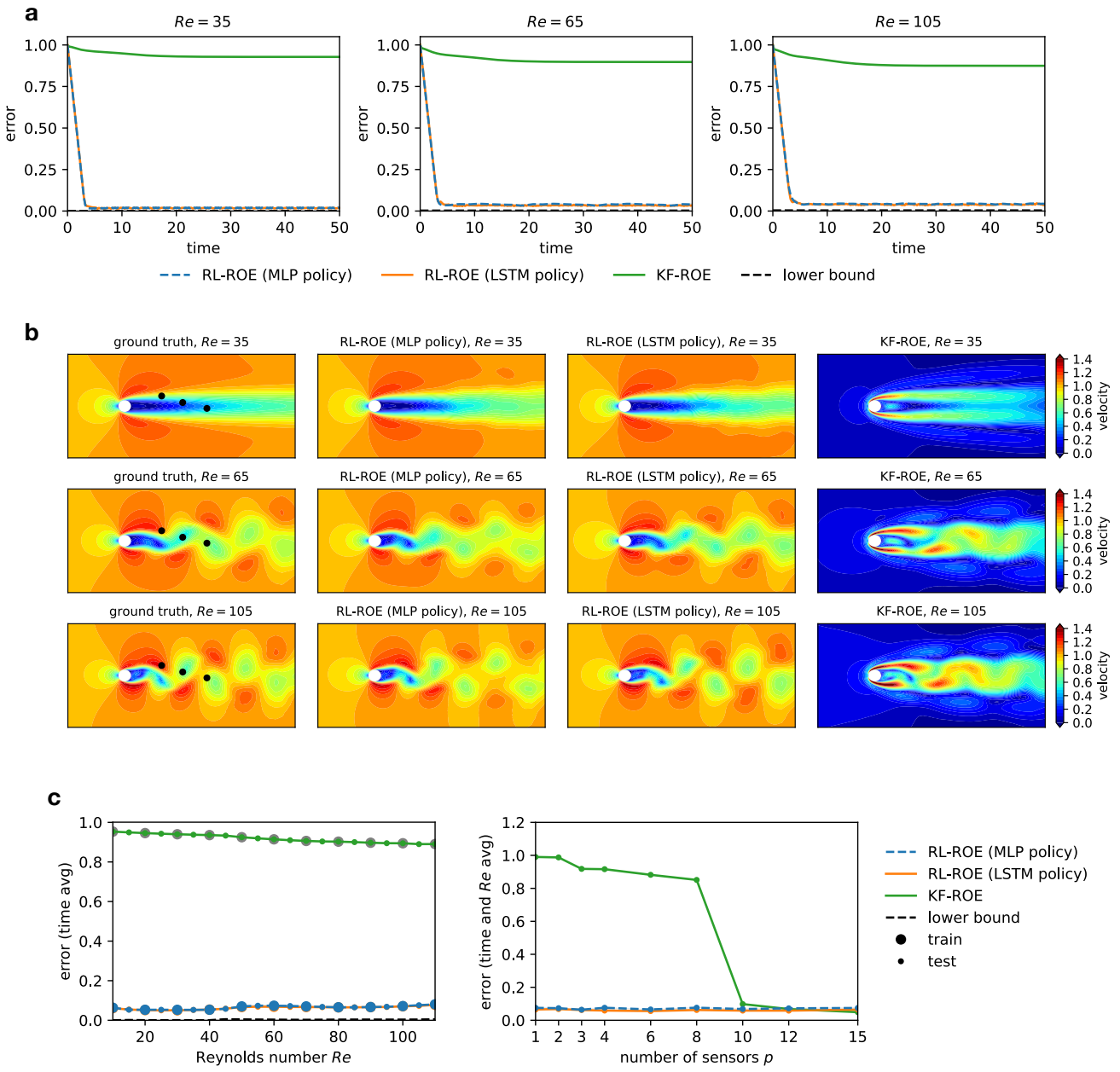

Figure 4. Chaotic vorticity flow. **a**, Normalized L_2 error of the RL-ROE and KF-ROE state estimates for trajectories not seen during training, using $p = 3$ sensors. **b**, Ground-truth velocity magnitude at $t = 15$ for trajectories not seen during training and corresponding RL-ROE and KF-ROE estimates using $p = 3$ sensors. The black crosses in the contours of the ground truth plots indicate the sensor locations. **c**, Left: Normalized L_2 error, averaged over time, versus trajectory number when using $p = 3$ sensors. Trajectories belonging to the training set S are shown by large circles while the testing trajectories are displayed by small circles. Right: Normalized L_2 error, averaged over time and over testing trajectories, versus number of sensors p. In **a** and **c**, the error metrics are averaged over 5 randomly-sampled initial true states $\boldsymbol{z}_0$ along each trajectory, and the shaded areas denote the standard deviation.

Methods

Dynamic Mode Decomposition

The DMD algorithm[19,33] 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 $\{\boldsymbol{Z}^{\mu}\}_{\mu\in S}$, 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

$$\boldsymbol{X}=\{\boldsymbol{z}_0^{\mu_1},\dots,\boldsymbol{z}_{m-1}^{\mu_1},\dots,\boldsymbol{z}_0^{\mu_q},\dots,\boldsymbol{z}_{m-1}^{\mu_q}\}, \tag{12a}$$

$$\boldsymbol{Y}=\{\boldsymbol{z}_1^{\mu_1},\dots,\boldsymbol{z}_m^{\mu_1},\dots,\boldsymbol{z}_1^{\mu_q},\dots,\boldsymbol{z}_m^{\mu_q}\}, \tag{12b}$$

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

$$\boldsymbol{X}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top} \tag{13}$$

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

$$\boldsymbol{A}_r=\boldsymbol{U}^{\top}\boldsymbol{A}\boldsymbol{U}=\boldsymbol{U}^{\top}\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}, \tag{14a}$$

$$\boldsymbol{C}_r=\boldsymbol{C}\boldsymbol{U}. \tag{14b}$$

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

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 [34,35]. To be able to use off-the-shelf RL 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 $(\mathscr{S},\mathscr{A},\mathscr{P},\mathscr{R})$, where $\mathscr{S}=\mathbb{R}^{r+n+1}$ is the augmented state space, $\mathscr{A}\subset\mathbb{R}^r$ is the action space, $\mathscr{P}(\cdot|\boldsymbol{s}_k,\boldsymbol{a}_k)$ is a transition probability, and $\mathscr{R}(\boldsymbol{s}_k,\boldsymbol{a}_k,\boldsymbol{s}_{k+1})$ is a reward function. At each time step k, the agent selects an action $\boldsymbol{a}_k\in\mathscr{A}$ according to the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ defined in (4b), which can be expressed as

$$\boldsymbol{a}_k\sim\boldsymbol{\pi}_{\boldsymbol{\theta}}(\cdot\,|\boldsymbol{o}_k), \tag{15}$$

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

$$\boldsymbol{s}_{k+1}=(\boldsymbol{A}_r\hat{\boldsymbol{x}}_{k-1}+\boldsymbol{a}_k,\boldsymbol{f}(\boldsymbol{z}_k;\mu),\mu), \tag{16}$$

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

$$r_k=\mathscr{R}(\boldsymbol{s}_k,\boldsymbol{a}_k,\boldsymbol{s}_{k+1})=-\|\boldsymbol{z}_k-\boldsymbol{U}\hat{\boldsymbol{x}}_k\|^2, \tag{17}$$

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 (17) has no explicit time dependence and the MDP is therefore stationary.

The RL training process then finds the optimal policy parameters

$$\boldsymbol{\theta}^*=\arg\max_{\boldsymbol{\theta}}\mathop{\mathbb{E}}_{\tau\sim\boldsymbol{\pi}_{\boldsymbol{\theta}}}[R(\tau)], \tag{18}$$

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 (18) 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

$$\hat{\boldsymbol{x}}_0 \sim p_{\hat{\boldsymbol{x}}_0}(\cdot), \quad \boldsymbol{z}_0 \sim p_{\boldsymbol{z}_0}(\cdot), \quad \mu \sim p_\mu(\cdot), \tag{19}$$

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 (16), 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 satisfies $K_{\text{start}} + K_{\text{train}} \leq K$. 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 (15) 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 $\boldsymbol{o}_k$[36,37]. Nonetheless, policies formulated based on an incomplete summary of $\boldsymbol{h}_k$ are also common in practice and still achieve good results[38]. 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 $\boldsymbol{h}_k$.

RL algorithm and hyperparameters

To learn the optimal policy parameters $\boldsymbol{\theta}^*$, we employ the Proximal Policy Optimization (PPO) algorithm[39], which belongs to the class of policy gradient methods[40]. 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 our POMDP setting. For our problem, this guarantees that the PPO algorithm will converge to a locally optimum policy.

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 $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ 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[41] and define our MDP as a custom environment in OpenAI Gym[42].

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. During training, the policy is tested (with stochasticity switched off) after each update using 20 separate 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.

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:

$$\boldsymbol{P}_k^- = \boldsymbol{A}_r \boldsymbol{P}_{k-1} \boldsymbol{A}_r^\mathsf{T} + \boldsymbol{Q}_k, \tag{20}$$

$$\boldsymbol{S}_k = \boldsymbol{C}_r \boldsymbol{P}_k^- \boldsymbol{C}_r^\mathsf{T} + \boldsymbol{R}_k, \tag{21}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}_r^\mathsf{T} \boldsymbol{S}_k^{-1}, \tag{22}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}_r) \boldsymbol{P}_k^-, \tag{23}$$

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 [2]. 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 (19).

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 . 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 . 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 . 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.

Code and data availability

The code and data used to generate the results in this study are available from S.M. upon reasonable request.

References

1. Brunton, S. L. & Noack, B. R. Closed-loop turbulence control: Progress and challenges. *Appl. Mech. Rev.* **67** (2015).
2. Simon, D. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches* (John Wiley & Sons, 2006).
3. Särkkä, S. *Bayesian filtering and smoothing* (Cambridge university press, 2013).
4. Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.* **82**, 35–45 (1960).
5. Zarchan, P. *Progress in astronautics and aeronautics: fundamentals of Kalman filtering: a practical approach*, vol. 208 (Aiaa, 2005).
6. Benner, P., Gugercin, S. & Willcox, K. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review* **57**, 483–531 (2015).
7. Rowley, C. W. & Dawson, S. T. Model reduction for flow analysis and control. *Annu. Rev. Fluid Mech.* **49**, 387–417 (2017).
8. Brunton, S. L. & Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge, Cambridge University Press, 2019).
9. Gomez, D. F. *et al.* Data-driven estimation of the unsteady flowfield near an actuated airfoil. *J. Guid. Control. Dyn.* **42**, 2279–2287 (2019).

10. Tsolovikos, A., Bakolas, E., Suryanarayanan, S. & Goldstein, D. Estimation and control of fluid flows using sparsity-promoting dynamic mode decomposition. *IEEE Control. Syst. Lett.* **5**, 1145–1150 (2020).

11. Mowlavi, S. & Benosman, M. Dual parametric and state estimation for partial differential equations. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, 8156–8161 (IEEE, 2023).

12. Ahmed, S. E. *et al.* On closures for reduced order models—a spectrum of first-principle to machine-learned avenues. *Phys. Fluids* **33**, 091301 (2021).

13. Lee, K. & Carlberg, K. T. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.* **404**, 108973 (2020).

14. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. machine intelligence* **3**, 218–229 (2021).

15. Kovachki, N. *et al.* Neural operator: Learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.* **24**, 1–97 (2023).

16. Greenberg, I., Yannay, N. & Mannor, S. Optimization or architecture: How to hack kalman filtering. *Adv. Neural Inf. Process. Syst.* **36** (2024).

17. Adrian, R. J. & Westerweel, J. *Particle image velocimetry*. 30 (Cambridge university press, 2011).

18. Taira, K. *et al.* Modal analysis of fluid flows: An overview. *AIAA J.* **55**, 4013–4041 (2017).

19. Schmid, P. J. Dynamic mode decomposition of numerical and experimental data. *J. fluid mechanics* **656**, 5–28 (2010).

20. Kutz, J. N., Brunton, S. L., Brunton, B. W. & Proctor, J. L. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (SIAM, 2016).

21. Korovin, S. K. & Fomichev, V. V. State observers for linear systems with uncertainty. In *State Observers for Linear Systems with Uncertainty* (de Gruyter, 2009).

22. Besançon, G. *Nonlinear observers and applications*, vol. 363 (Springer, 2007).

23. Julier, S. J. & Uhlmann, J. K. Unscented filtering and nonlinear estimation. *Proc. IEEE* **92**, 401–422 (2004).

24. Morimoto, J. & Doya, K. Reinforcement learning state estimator. *Neural computation* **19**, 730–756 (2007).

25. Hu, L., Wu, C. & Pan, W. Lyapunov-based reinforcement learning state estimator. *arXiv preprint arXiv:2010.13529* (2020).

26. Revach, G. *et al.* Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Process.* **70**, 1532–1547 (2022).

27. Peng, X. B., Andrychowicz, M., Zaremba, W. & Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, 3803–3810 (IEEE, 2018).

28. Kay, S. M. *Fundamentals of statistical signal processing: estimation theory* (Prentice-Hall, Inc., 1993).

29. Lorenc, A. C. The potential of the ensemble kalman filter for nwp—a comparison with 4d-var. *Q. J. Royal Meteorol. Soc. A journal atmospheric sciences, applied meteorology physical oceanography* **129**, 3183–3203 (2003).

30. Wan, E. A. & Van Der Merwe, R. 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)*, 153–158 (Ieee, 2000).

31. Jackson, C. A finite-element study of the onset of vortex shedding in flow past variously shaped bodies. *J. fluid Mech.* **182**, 23–45 (1987).

32. Li, Z. *et al.* Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations* (2020).

33. Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L. & Kutz, J. N. On Dynamic Mode Decomposition: theory and applications. *J. Comput. Dyn.* **1**, 391–421 (2014).

34. Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming* (John Wiley & Sons, 2014).

35. Lecarpentier, E. & Rachelson, E. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. *Adv. neural information processing systems* **32** (2019).

36. Kaelbling, L. P., Littman, M. L. & Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artif. intelligence* **101**, 99–134 (1998).

37. Ni, T., Eysenbach, B. & Salakhutdinov, R. Recurrent model-free rl can be a strong baseline for many pomdps. In *International Conference on Machine Learning*, 16691–16723 (PMLR, 2022).

38. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).

39. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

40. Sutton, R. S., McAllester, D. A., Singh, S. P. & Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063 (2000).

41. Raffin, A. *et al.* Stable baselines3. https://github.com/DLR-RM/stable-baselines3 (2019).

42. Brockman, G. *et al.* Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

Author contributions statement

S.M. and M.B. designed research, performed research, and wrote the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

This article contains supplementary materials.