REINFORCEMENT LEARNING STATE ESTIMATION FOR HIGH-DIMENSIONAL NONLINEAR SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

High-dimensional nonlinear systems such as atmospheric or oceanic flows present a computational challenge for data assimilation (DA) algorithms such as Kalman filters. A potential solution is to rely on a reduced-order model (ROM) of the dynamics. However, ROMs are prone to large errors, which negatively affects the accuracy of the resulting forecast. Here, we introduce the reinforcement learning reduced-order estimator (RL-ROE), a ROM-based data assimilation algorithm in which the correction term that takes in the measurement data 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

A key challenge in Earth sciences, particularly in meteorology and oceanography, is to predict the spatio-temporal evolution of complex processes using physical knowledge as well as observational data (Kalnay, 2003; Dueben & Bauer, 2018). A popular approach for solving such data assimilation problems are state estimation algorithms, such as the Kalman filter and its numerous extensions, that use sparse sensor measurements to correct the state predicted by a forward dynamical model every time new data becomes available (Zarchan & Musoff, 2015; Houtekamer & Zhang, 2016; Carrassi et al., 2018). However, many processes arising in Earth sciences such as atmospheric or oceanic flows are governed by partial differential equations (PDEs) which, when discretized, yield high-dimensional and nonlinear dynamical models with thousands or more of state variables. Since integrating these high-dimensional models with Kalman filter-based DA techniques is computationally expensive, dimensionality reduction approaches in which a reduced-order model (ROM) of the dynamics is used instead of the full model have recently become popular (Ballabrera-Poy et al., 2001; Cao et al., 2007; Fang et al., 2009; Ştefănescu et al., 2015).

A fundamental issue is that ROMs provide a simplified and imperfect description of the dynamics (Rowley & Dawson, 2017), which negatively affects the accuracy of the forecast. One potential solution is to improve the accuracy of the ROM through the inclusion of additional closure terms (Ahmed et al., 2021). In this paper, we leave the ROM untouched and instead propose a new design paradigm for the DA component, resulting in a novel estimation algorithm 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. Our approach follows recent trends to use machine learning to improve DA frameworks (Kashinath et al., 2021; Chattopadhyay et al., 2021; Farchi et al., 2021; Grooms, 2021; Hatfield et al., 2021). 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.

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. Thus, the state $\boldsymbol{z}_k$ will hereafter be referred to as the high-dimensional state.

2.2 REDUCED-ORDER MODEL

Because the high dimensionality of (1) makes integration with DA techniques computationally expensive, an alternative approach is 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) and is related to the linear inverse modeling (LIM) method from climate science (Tu et al., 2014). 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. We call the estimator constructed and trained through this process an RL-trained ROE, or RL-ROE for short.

2.4 Summary of the proposed methodology

In summary, the methodology we propose consists of the following three steps. The first two are carried out offline using full knowledge of $\{\boldsymbol{z}_0, \cdots, \boldsymbol{z}_k\}$ from several trajectories of (1), and the third is performed online using sole knowledge of measurements $\{\boldsymbol{y}_0, \cdots, \boldsymbol{y}_k\}$ from a previously unseen trajectory of (1).

1. **Construction of an RL-ROE**. A ROM of the form (2) is obtained by applying the DMD to high-dimensional state snapshots $\boldsymbol{z}_k$ from a single trajectory of (1). An RL-ROE (3) is then designed based on this ROM.
2. **Training of the RL-ROE**. The policy $\boldsymbol{\pi_\theta}$ of the RL-ROE is trained using high-dimensional snapshots $\boldsymbol{z}_k$ from multiple reference trajectories of (1).
3. **Deployment of the RL-ROE**. Using measurements $\boldsymbol{y}_k$ from a previously unseen reference trajectory of (1), the trained RL-ROE returns an estimate $\hat{\boldsymbol{z}}_k = \boldsymbol{U}\hat{\boldsymbol{x}}_k$ for the corresponding (unobserved) high-dimensional state $\boldsymbol{z}_k$.

We note that previous studies (Morimoto & Doya, 2007; Hu et al., 2020) have already proposed designing state estimators using policies trained through reinforcement learning. In appendix B, we detail the novelties of our proposed methodology with respect to these studies.

3 Framing the problem as a stationary MDP

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 stationary MDP based on an extended state. 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

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

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.

Thanks to the stationarity of the MDP, one can use any deep RL algorithm to train the policy $\boldsymbol{\pi_\theta}$ using high-dimensional snapshots $\boldsymbol{z}_k$ from multiple reference trajectories of (1). Here, we use the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017), which belongs to the class of policy gradient methods (Sutton et al., 2000). The setup of the RL training process, the parameterization of the policy $\boldsymbol{\pi_\theta}$, and implementation details are discussed in Appendix C.

Remark. Since the policy (5) is conditioned on a partial observation $\boldsymbol{o}_k$ of the state $\boldsymbol{s}_k$, the MDP we have defined in this section is, in fact, a partially observable MDP (POMDP). In this case, it is known that the optimal policy depends on a summary of the history of past observations and actions, $\boldsymbol{h}_k = \{\boldsymbol{o}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{o}_k\}$, rather than just the current observation $\boldsymbol{o}_k$ (Kaelbling et al., 1998). However, policies formulated based on an incomplete summary of $\boldsymbol{h}_k$ are common in practice and still achieve good results (Sutton & Barto, 2018). We therefore pursue this approach in the present paper, and leave for future work testing the generalization of our policy input to a more complete summary of $\boldsymbol{h}_k$.

4 RESULTS

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

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0, \tag{8}$$

where $u(x,t)$ is the velocity at position $x \in [0, L]$ and time $t \in [0, T]$, 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 (8) using a spectral method with $n = 256$ Fourier modes and a fifth-order Runge-Kutta time integration scheme, and we save snapshots every $\Delta t = 0.05$ time units. The resulting discrete system takes the form (1) where the state vector $\boldsymbol{z}_k \in \mathbb{R}^n$ comprises the values of u at the collocation points $x_i = iL/n$ and time $t = k\Delta t$, and the measurement vector $\boldsymbol{y}_k \in \mathbb{R}^p$ comprises the values of u at $p = 8$ sensor locations $\bar{x}_i = iL/p$.

We then follow the procedure outlined in Section 2.4 to construct an RL-ROE, which we train offline using PPO following the methodology presented in Section 3. The trained RL-ROE is finally deployed online and compared against a Kalman filter under various reference trajectories as well as initial state estimates. To ensure a proper comparison, the Kalman filter is constructed from the same ROM on which the RL-ROE is based, and will therefore be referred to as KF-ROE. More details regarding these various steps can be found in Appendix D.

The performances of the trained RL-ROE and KF-ROE are now compared. Figures 1(a,b,c) show the L2 error of the RL-ROE and KF-ROE for specific reference trajectories of the Burgers equation, initialized from (15a) using $\alpha = 0.5$, 1, 2 where α is the amplitude of the initial condition (15a). Importantly, we note that the ROM was computed from the trajectory initialized with $\alpha = 1$, and is therefore very accurate (i.e., the model error is low) for this case. For each reference trajectory, we consider 20 different initial state estimates sampled from (15b). The curves and shaded area reported in Figures 1(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. Figures 1(d,e,f) show the trajectories of the reference,

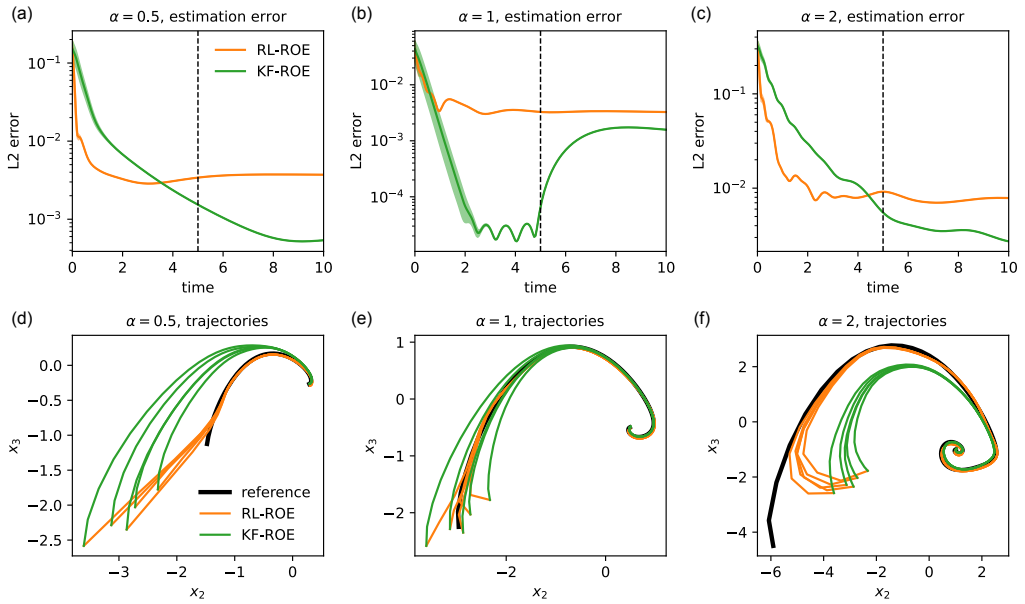

Figure 1: Accuracy of the RL-ROE. (a,b,c) L2 error of the RL-ROE and KF-ROE with respect to specific reference trajectories of the Burgers equation, initialized from (15a) using $\alpha = 0.5, 1, 2$. The RL-ROE and KF-ROE are evaluated using 20 different initial estimates sampled from (15b). The curves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

RL-ROE and KF-ROE solutions in a two-dimensional slice of phase space spanned by the second and third columns[1] of $\boldsymbol{U}$.

A few important observations emerge from Figure 1. First, the RL-ROE outperforms the KF-ROE when the ROM suffers from large model errors, as is the case for $\alpha = 0.5$ and $\alpha = 2$. 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 1(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 $t = 5$.

5 Conclusions

In this paper, we have introduced the reinforcement learning reduced-order estimator (RL-ROE), a new methodology for forecasting the state of a high-dimensional nonlinear dynamical system using sparse observations. Our approach follows the recent trend 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 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] 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.

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.

Joaquim Ballabrera-Poy, Antonio J Busalacchi, and Ragu Murtugudde. Application of a reduced-order kalman filter to initialize a coupled atmosphere–ocean model: Impact on the prediction of el nino. *Journal of climate*, 14(8):1720–1737, 2001.

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

Yanhua Cao, Jiang Zhu, I Michael Navon, and Zhendong Luo. A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 53(10):1571–1583, 2007.

Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

Ashesh Chattopadhyay, Mustafa Mustafa, Pedram Hassanzadeh, Eviatar Bach, and Karthik Kashinath. Towards physically consistent data-driven weather forecasting: Integrating data assimilation with equivariance-preserving spatial transformers in a case study with era5. *Geoscientific Model Development Discussions*, pp. 1–23, 2021.

Peter D Dueben and Peter Bauer. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10):3999–4009, 2018.

F Fang, CC Pain, IM Navon, MD Piggott, GJ Gorman, PE Farrell, PA Allison, and AJH Goddard. A pod reduced-order 4d-var adaptive mesh ocean modelling approach. *International Journal for Numerical Methods in Fluids*, 60(7):709–732, 2009.

Alban Farchi, Patrick Laloyaux, Massimo Bonavita, and Marc Bocquet. Using machine learning to correct model error in data assimilation and forecast applications. *Quarterly Journal of the Royal Meteorological Society*, 147(739):3067–3084, 2021.

Ian Grooms. Analog ensemble data assimilation and a method for constructing analogs with variational autoencoders. *Quarterly Journal of the Royal Meteorological Society*, 147(734):139–149, 2021.

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.

Sam Hatfield, Matthew Chantry, Peter Dueben, Philippe Lopez, Alan Geer, and Tim Palmer. Building tangent-linear and adjoint models for data assimilation with neural networks. *Journal of Advances in Modeling Earth Systems*, 13(9):e2021MS002521, 2021.

Peter L Houtekamer and Fuqing Zhang. Review of the ensemble kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 144(12):4489–4532, 2016.

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

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

Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

K Kashinath, M Mustafa, A Albert, JL Wu, C Jiang, S Esmaeilzadeh, K Azizzadenesheli, R Wang, A Chattopadhyay, A Singh, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

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.

Răzvan Ştefănescu, Adrian Sandu, and Ionel Michael Navon. Pod/deim reduced-order strategies for efficient four dimensional variational data assimilation. *Journal of Computational Physics*, 295: 569–595, 2015.

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

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

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

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

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 $\boldsymbol{Z} = \{\boldsymbol{z}_0, \dots, \boldsymbol{z}_m\}$ 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

$$\boldsymbol{X} = \{\boldsymbol{z}_0, \dots, \boldsymbol{z}_{m-1}\}, \quad \boldsymbol{Y} = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_m\}, \tag{9}$$

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

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T} \tag{10}$$

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

$$\boldsymbol{A}_r = \boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U} = \boldsymbol{U}^\mathsf{T}\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}, \tag{11a}$$

$$\boldsymbol{C}_r = \boldsymbol{C}\boldsymbol{U}. \tag{11b}$$

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 Related work

In this appendix, we compare our work with two previous studies that 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.

C Setup of the RL training process

In this appendix, we describe the setup of the RL training process as well as implementation details. The goal of the training is to find the optimal policy parameters

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\tau \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}} [R(\tau)], \tag{12}$$

where the expectation is over trajectories $\tau = (\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{s}_2, \boldsymbol{a}_2, \dots)$, and $R(\tau) = \sum_{k=1}^{K} r_k$ is the finite-horizon undiscounted return, 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

$$\boldsymbol{z}_0 \sim \boldsymbol{p}_{\boldsymbol{z}_0}(\cdot), \tag{13a}$$

$$\hat{\boldsymbol{x}}_0 \sim \boldsymbol{p}_{\hat{\boldsymbol{x}}_0}(\cdot), \tag{13b}$$

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 $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ 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.

D Protocol for the results on the Burgers equation

We adopted the following steps when obtaining the results shown in Section 4:

1. **Construction of an RL-ROE.** 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 $\boldsymbol{Z}^{\text{DMD}} = \{\boldsymbol{z}_0^{\text{DMD}}, \dots, \boldsymbol{z}_m^{\text{DMD}}\}$ the resulting solution snapshots at times $k = 0, \dots, m = T/2\Delta t$. 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 $n = 256$. An RL-ROE (3) is then designed based on this ROM.
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 3. 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
$$\boldsymbol{z}_0 = \alpha \boldsymbol{z}_0^{\text{DMD}}, \tag{15a}$$
$$\hat{\boldsymbol{x}}_0 = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{z}_0^{\text{DMD}} + \boldsymbol{\beta}, \tag{15b}$$
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 (13). 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. We end the training when the return no longer increases on average. The training hyperparameters and learning curves are presented in Appendix E.
3. **Deployment and 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 F. 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 $t \in [0, T]$.

E 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 2 reports the learning curves. 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. 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.

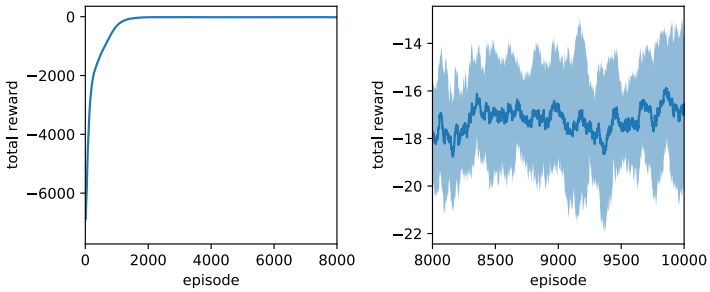

Figure 2: Learning curves for the stochastic policy. The line and shaded area show the mean and standard deviation of the results over 10 runs, each one smoothed with a moving average of size 100 episodes.

F 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:

$$\boldsymbol{P}_k^- = \boldsymbol{A}_r \boldsymbol{P}_{k-1} \boldsymbol{A}_r^\mathsf{T} + \boldsymbol{Q}_k, \tag{16}$$

$$\boldsymbol{S}_k = \boldsymbol{C}_r \boldsymbol{P}_k^- \boldsymbol{C}_r^\mathsf{T} + \boldsymbol{R}_k, \tag{17}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}_r^\mathsf{T} \boldsymbol{S}_k^{-1}, \tag{18}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}_r) \boldsymbol{P}_k^-, \tag{19}$$

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).