Neural networks in feedback for flow analysis and control

Tarcísio C. Déda* and William R. Wolf
University of Campinas, Campinas, São Paulo, Brazil

Scott T. M. Dawson
Illinois Institute of Technology, Chicago, Illinois, USA

(Received 15 December 2023; accepted 6 May 2024; published 12 June 2024)

This work presents a methodology for analysis and control of nonlinear fluid systems using neural networks. The approach is demonstrated in four different study cases: the Lorenz system, a modified version of the Kuramoto-Sivashinsky equation, a streamwise-periodic two-dimensional channel flow, and a confined cylinder flow. Neural networks are trained as models to capture the complex system dynamics and estimate equilibrium points through a Newton method, enabled by back-propagation. These neural network surrogate models (NNSMs) are leveraged to train a second neural network, which is designed to act as a stabilizing closed-loop controller. The training process employs a recurrent approach, whereby the NNSM and the neural network controller are chained in closed loop along a finite time horizon. By cycling through phases of combined random open-loop actuation and closed-loop control, an iterative training process is introduced to overcome the lack of data near equilibrium points. This approach improves the accuracy of the models in the most critical region for achieving stabilization. Through the use of L1 regularization within loss functions, the NNSMs can also guide optimal sensor placement, reducing the number of sensors from an initial candidate set. The data sets produced during the iterative training process are also leveraged for conducting a linear stability analysis through a modified dynamic mode decomposition approach. The results demonstrate the effectiveness of computationally inexpensive neural networks in modeling, controlling, and enabling stability analysis of nonlinear systems, providing insights into the system behavior and offering potential for stabilization of complex fluid systems.

DOI: 10.1103/PhysRevFluids.9.063904

I. INTRODUCTION

Neural networks (NNs) are powerful tools for data-driven mathematical modeling, finding a range of applications in the field of fluid mechanics such as flow modeling, analysis, optimization, and control [1–5]. Recent advances in experimental data sampling, the accessibility of data from increasingly complex simulations, and the widespread availability of libraries that facilitate the design and training of NNs has encouraged their rapid development. In this context, NNs have been applied to data compression, model order reduction, flow analysis, turbulence modeling, and feature extraction, all leveraging their power as general nonlinear function approximators. In the field of turbulence modeling, deep neural networks have been used to improve the prediction capabilities of Reynolds-averaged Navier-Stokes (RANS) closure models [6,7]. In the area of flow estimation and reconstruction, Milano and Koumoutsakos [8] presented one of the first applications of neural networks in fluid mechanics for reconstructing near-wall turbulent flow fields. More recently,

*tarcisio.deda@gmail.com

for example, Fukami *et al.* [9] employed convolutional neural networks (CNNs) to accurately reconstruct turbulent flows from coarse flowfield images.

Considering reduced-order modeling, Lui and Wolf [10] combined deep neural networks and proper orthogonal decomposition (POD) to build accurate models of unsteady flows involving transient features, rewriting the Navier-Stokes equations as a set of nonlinear ordinary differential equations. More recently, Miotto and Wolf [11] employed vision transformers and CNNs to extract quantities of interest such as aerodynamic coefficients and skin friction from flow-field visualizations. These authors demonstrated that neural network models based on input images were effective in interpolating and extrapolating between flow regimes, offering a promising alternative for sensors in experimental campaigns and for building surrogate models of complex unsteady flows. Representations of complex systems were studied by Page *et al.* [12], who employed deep convolutional networks combined with frequency domain analyses in latent spaces to obtain a simplified phase space in a Kolmogorov flow. Regarding prediction modeling with limited data sensing, flow reconstruction from a sparse distribution of probes was performed by Deng *et al.* [13]. In this study, the authors employed POD methods based on long short-term memory (LSTM) networks, being able to accurately predict flow fields. Further studies exploring this idea were conducted by Manohar *et al.* [14] by employing flow reconstruction from sparse pressure probes using LSTM-based networks to more complex two-dimensional (2D) flows, which resulted in more accurate reconstructions when compared to POD-based low-dimensional subspaces.

In the field of flow control, different techniques based on training NNs are found in the literature to either build a model for enabling control design [15,16] or to directly infer a suitable control law for a given task. Training of NN-based models for control design was conducted by Bieker *et al.* [16], who applied limited sensor data supported by delay coordinates, and by Morton *et al.* [15], in an approach that modeled the dynamics of a cylinder flow with models trained to learn state mappings to span a Koopman invariant subspace. Regarding model-free approaches, reinforcement learning (RL) can be highlighted as a promising tool for automatic training of control strategies with applications to drag reduction, for example. In their work, Rabault *et al.* [17], Ren *et al.* [18], and Castellanos *et al.* [19] developed RL strategies to train controllers able to reduce drag in a confined cylinder flow through a set of measurements from velocity probes. Their sensor and actuation setups are employed to test the techniques proposed in the current work. Similarly, Li and Zhang [20] considered RL strategies to suppress vortex shedding utilizing stability analyses and unstable equilibrium computation of the underlying system to obtain improved results. Approaches based on RL have also been successfully applied to complex turbulent flows in both experimental [21] and computational [22] setups, being able, in some cases, to stabilize the flows [23] or reduce drag [24,25]. Reinforcement learning has also been applied to gliding control in different flow setups [26,27] and to open-loop setups where high complexity of the operating parameter space makes it justifiable to employ machine learning optimization instead of brute force searches [28]. Comprehensive reviews of RL for flow control can be found in Refs. [29,30].

Models based on sparse identification of nonlinear dynamics (SINDy) [31] have been combined with order-reduction techniques, such as POD, as an alternative to black-box NN models, being able to find simpler and more intuitive models based on a library of interpretable functions. The approach can be applied for building nonlinear models with control inputs, which can be leveraged for closed-loop control applications [32]. On the topic of flow optimization through open-loop control, genetic algorithms can be a candidate tool for parameter search. In such cases, studies have been performed to find the best active control setup to improve a fitness function that aims to reduce drag in a bluff body [33], to maximize the thrust vectoring angle in a supersonic jet [34], and to find the best open-loop rotation values in order to reduce drag and symmetrize the wake of the flow past a cluster of cylinders [35].

Traditional feedback control design approaches often rely on well-established control theory, for which the literature provides techniques that can ensure optimality, robustness, adaptability, or the achievement of specific performance characteristics such as settling (convergence) time. Such linear control methods have been applied for the control of a variety of flow problems [36], such as cavities

[37,38], bluff-body wakes [39,40], and unsteady aerodynamic systems [41–43]. However, the broad application of such approaches in fluid mechanics is limited by the fact that they typically rely on linear approximations of the dynamics near desired equilibrium points. Furthermore, building data-driven linear models of flows is also hindered by the fact that signals sampled from fluid flows can be governed by nonlinear phenomena such as chaos, limit cycles, multiple equilibria, switching, and hysteresis. In the case of unstable flows (more specifically globally unstable flows), data sets containing measurements at (approximately) linear regions near equilibrium can be totally absent. Alternatively, nonlinear control techniques, which include sliding-mode control [44,45], feedback linearization [46], backstepping control [47], extremum seeking control [48], and switched control [49], can be applied to the task of controlling nonlinear systems, although stabilization relative to a natural equilibrium point may still require a suitable model with some level of accuracy near equilibrium. Differentiable nonlinear models built from data can be leveraged for model predictive control (MPC), which employs real-time optimization to a cost function within a finite horizon. Although MPC still presents similar limitations near the linear regions for some flows—such as difficulties in stabilizing the system—it has been successfully applied to different flow control problems [15,16,50,51]. For example, near stabilization of the flow past a cluster of cylinders was achieved by Maceda *et al.* [52].

The determination of optimal or advantageous locations to place sensors to enable full state reconstruction has been explored through a variety of methods, including via gappy proper orthogonal decomposition [53], a pivoted QR decomposition of an identified set of basis functions [54], data-driven dynamical models with Kalman filters [55,56], and shallow decoder neural networks [57]. A machine learning control strategy with the appropriate setup of loss functions can at the same time seek the control objective and a sparse sensor selection from an initial candidate set, as sought in the present work. An example of this type of approach is presented by Paris *et al.* [58], where an L0 regularization scheme is proposed to find the best probing locations. As well as state reconstruction, the problem of sensor placement arises in flow control problems, where the placement of both sensors and actuators is important, and can be optimized and analyzed using linear control theory [59–61]. The sensor placement methodology employed in the present work differs from these previous methods and utilizes a modified neural network loss function that incorporates the L1 norm of node (and thus physical location) weights. The use of L1-norm-based loss functions has been applied to find sparse solutions in a range of flow analysis problems, such as to find a reduced set of dynamic mode decomposition (DMD) modes to represent flow dynamics [62], and to find spatially localized resolvent modes [63,64].

In the present work, we present an approach that enables nonlinear control design by utilizing neural network architectures to model both the system and control law. To achieve stabilization of complex systems exhibiting nonlinear dynamics, a neural network controller (NNC) is trained in closed loop with a neural network surrogate model (NNSM), which is built from data obtained from a real plant [51]. The methodology is proposed as a candidate solution to overcome the aforementioned lack of measurements near the linear region of unstable nonlinear systems. The training framework also allows for optimal sensor placement by automatic selection from a set of candidate probes. Here, through penalization of model inputs, a sparse configuration is aimed that exclude the less relevant sensors. Furthermore, a by-product consisting of data sets that contain measurements from approximately linear regions of the state space is leveraged for the data-driven estimation of equilibrium points—which are used as control setpoints—and for conducting a data-driven linear stability analysis. This novelty allows for approaching both problems without the need to modify the solver and enables the construction of linearized flow models involving complex geometries, as well as other nonlinear systems. Moreover, it potentially enables equilibrium estimation and stability analysis through data gathered from experimental applications.

Explicit training of models through the present supervised machine learning approach presents several advantages over reinforcement learning for direct control training, although the setup involves additional steps such as building a model for the plant. These advantages include the ability to obtain insightful linear models from the black-box neural networks, as well as other procedures that

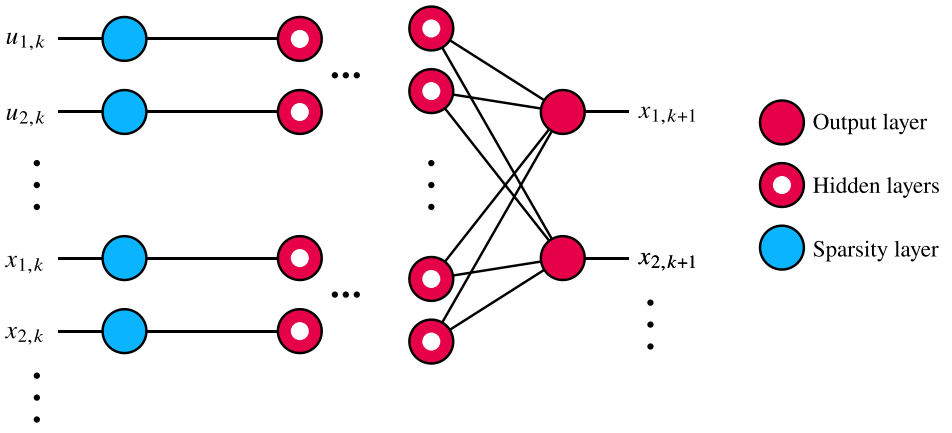

FIG. 1. Schematic of the NNSM where the red (blue) nodes are related to weights penalized by the L2 (L1) regularization. A ReLU function is employed in the nodes with a white mark, while a linear function is used for the other nodes.

can be enabled with it, such as linear stability analysis and equilibrium estimation. Another relevant benefit emerges at the moment data are already sampled and available: any number of different models and controllers can be trained with different hyperparameters and objective functions, without the need to run expensive simulations or experiments again—as opposed to reinforcement learning, which requires online interactions between the environment and the actors being trained. The capabilities of the proposed methodology are demonstrated through analysis and control of four different plants, namely, the Lorenz system, a modified version of the Kuramoto-Sivashinsky equation, a streamwise-periodic 2D channel flow, and a confined cylinder flow. The remainder of this work is organized as follows: Sec. II describes the tools and techniques introduced and employed along the work, while Sec. III describes in detail the proposed iterative approach for training the NNs. In Sec. IV, we describe the different systems and/or plants to which the techniques are applied. Section V then presents the results for each choice of plant. Finally, the benefits and limitations of the approaches presented are discussed in Sec. VI.

II. METHODOLOGY

A. Neural network surrogate models

In this work, NNSMs are employed to represent the dynamics of complex systems, such as fluid flows. Given a fixed time step Δt, we are interested in discrete systems of the form

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k), \tag{1}$$

where $\mathbf{x}$ is the vector containing the system states, $\mathbf{u}$ is the vector of control inputs, and the subscripts represent the discrete time steps such that time $t = k\Delta t$. The state vector $\mathbf{x}$ consists of a set of sensor measurements from which the dynamics can be inferred using data-driven methods. This set of of measurements consists of a subset of the true plant full set of states.

From a data set containing measurements of both $\mathbf{x}$ and $\mathbf{u}$ from the plant, for multiple time steps, a NNSM is trained to approximate $F(\mathbf{x}, \mathbf{u})$. A schematic of the neural network, with its inputs and outputs, is presented in Fig. 1. Here, the symbol $\tilde{F}$ is used to denote the function represented by the NNSM, and thus a good fit would imply $\tilde{F} \approx F$. The network is composed of fully connected hidden layers which employ a rectified linear unit (ReLU) activation function given by $f(s_i) = max(0, s_i)$, whereas a linear function $f(s_i) = s_i$ is used for the output layer. Here, s_i can be understood as each element of a given array $\mathbf{s} = \mathbf{W}\mathbf{r} + \mathbf{h}$, where $\mathbf{W}$ and $\mathbf{h}$ are composed of the trainable weights and

biases and $\mathbf{r}$ is the input fed to the layer, which is also the output from the previous layer. In this context, $\tilde{F}$ is a nonlinear function composed of a combination of activation functions parametrized by the trained weights and biases [65]. The data for training and prediction are normalized so the average for each input and output is zero and the standard deviation is 1. We also point out that the present NNs have low complexity, being kept at a maximum of two hidden layers for the cases studied in this work (with further details provided in Appendix A).

Another idea explored and applied in the current work is related to the problem of sensor selection. We propose an approach that uses an initial complete set of candidate sensors that could be reduced by somehow penalizing their count during the neural network training process. There are several reasons to incorporate such an approach into the proposed methodology. First, using a smaller number of sensors reduces the total number of training parameters in the NNSM, which in turn reduces the amount of training data required. Second, real-world flow control applications typically use limited real-time measurements, and selecting a small number of sensors that are optimal for control has practical advantages. Finally, the incorporation of sensor selection into the NN training process allows for optimal sensor locations to be identified without relying on any prior knowledge or intuition of the system dynamics. Note that, here, optimality refers to an adequate sensor configuration that is at the same time reduced in number of probes and able to keep the NNSM capable of estimating the dynamics of the plant (which is more precisely described in this section by a cost function).

The approach proposed here consists of searching for a subset of relevant sensors among an initial set of candidates. To do so, we implement a layer of trainable parameters that are used to weight the NNSM inputs. These weights are penalized using an L1 regularization. The main objective is the production of a sparse layer excluding unnecessary measurements. Hence, after training, the weights below a given threshold (in absolute values) are truncated to zero. For the initialization, each weight value is set to 1.

To avoid these weights getting too low by the cost of growing hidden layer weights, an L2 regularization is applied to the hidden layers. This ensures that the input weights can only decrease if they are irrelevant to estimate the output, which is still the full set of states in a future time instant. Figure 1 illustrates such configuration. The loss function

$$\mathcal{L} = \frac{\mathbf{w_x}}{nn_s} \cdot \sum_{i=1}^{n_s \leqslant N_s} \mathbf{g}_i \odot \mathbf{g}_i + r_2 \|\mathbf{w}_\mathrm{h}\|_2 + r_1 \|\mathbf{w}_\mathrm{s}\|_1, \tag{2}$$

$$\mathbf{g}_i = \{[\mathbf{x}_{k+1}]_i - [\tilde{F}(\mathbf{x}_k, \mathbf{u}_k)]_i\} \oslash \mathbf{s_x}, \tag{3}$$

is used for evaluation of the network convergence, where $\mathbf{g}$ is the array of differences between the training data and the NNSM output normalized by the standard deviation $\mathbf{s_x}$ of the states sampled for the training data. The index i refers to each of the n_s sampled measurements. If more than an upper bound of N_s samples are available, we (randomly) subsample N_s data points in order to reduce computational costs of the iterative procedure that will be explained in the next section. One should note that the present notation employs the Hadamard (elementwise) product $\odot$ and division $\oslash$, as well as the L1 and L2 vector norms, $\|\cdot\|_1$ and $\|\cdot\|_2$, respectively. Furthermore, $\mathbf{w_x} = [1, \ldots, 1]_{(1\times n)}$ is always used, $\mathbf{w}_\mathrm{h}$ is the stacked weights array for the hidden layers, $\mathbf{w}_\mathrm{s}$ is the array of weights for the sparsity layer, n is the total number of candidate sensors, and r_1 and r_2 are the L1 and L2 regularization factors. In practice, r_1 can be adjusted such that larger values tend to provide sparser configurations at the cost of possibly reducing the accuracy of the model. The parameter r_2 can be increased to avoid the hidden layer weights to grow indefinitely larger to compensate for the sparsity layer weights becoming small. It is important to mention that, since the network weights and biases of the hidden layers are initialized randomly, the process of selecting sensors is stochastic and, therefore, we should expect that different runs for the same problem could result in different sensor configurations.

In practice, the final model can be viewed as a function of a reduced vector $\mathbf{x_r}$ containing $n_r \leqslant n$ values, corresponding to the nonzero components of the sparsity weights, $\mathbf{w}_\mathrm{s}$. Although this is not

done in the current work, pruning could also be conducted to reduce the complexity of the neural network by excluding unnecessary connections [66]. Regardless of pruning, it is now possible to evaluate the NNSM as a function with the following form:

$$\mathbf{x}_{k+1} = \tilde{F}_r(\mathbf{x}_{\mathbf{r}k}, \mathbf{u}_k), \tag{4}$$

so the full state vector is reconstructed from the reduced set of measurements. Similarly, we can define a function that predicts the components of the state corresponding to identified sensor locations,

$$\mathbf{x}_{\mathbf{r}k+1} = \tilde{F}_{rr}(\mathbf{x}_{\mathbf{r}k}, \mathbf{u}_k). \tag{5}$$

In short, $\tilde{F}_r$ and $\tilde{F}_{rr}$ are evaluated with the reduced set of states, but the former outputs the full set of states while the latter outputs the reduced set. Both $\tilde{F}_r$ and $\tilde{F}_{rr}$ are used for different goals in the present work, as will be made clear in the following sections.

B. Estimation of equilibrium state

To tackle equilibrium estimation, a linearization of the NNSM is obtained through back-propagation. In the first step, the reduced states matrix $\mathbf{A}_{\mathbf{rr}(n_r \times n_r)}$ is obtained for the time-invariant system of the form

$$\mathbf{d}_{\mathbf{r}k+1} = \mathbf{A}_{\mathbf{rr}}\mathbf{d}_{\mathbf{r}k} + \mathbf{B}_{\mathbf{r}}\mathbf{u}_k, \tag{6}$$

$$\mathbf{d}_{\mathbf{r}k} = \mathbf{x}_{\mathbf{r}k} - \mathbf{x}_{\mathbf{r}}{}^*, \tag{7}$$

where $\mathbf{A}_{\mathbf{rr}(n_r \times n_r)}$ and $\mathbf{B}_{\mathbf{r}(n_r \times m)}$ are constant matrices obtained from the Jacobian of $\tilde{F}_{rr}$ evaluated at a given equilibrium operating condition $\mathbf{x}_{\mathbf{r}}{}^*$ and $\mathbf{u}^*$ as follows:

$$\nabla \tilde{F}_{rr} = [\mathbf{A}_{\mathbf{rr}}\ \mathbf{B}_{\mathbf{r}}] = \begin{bmatrix} \frac{\partial \tilde{F}_{rr}}{\partial \mathbf{x}} & \frac{\partial \tilde{F}_{rr}}{\partial \mathbf{u}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \tilde{f}_1}{\partial x_1} & \cdots & \frac{\partial \tilde{f}_1}{\partial x_{n_r}} & \frac{\partial \tilde{f}_1}{\partial u_1} & \cdots & \frac{\partial \tilde{f}_1}{\partial u_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \tilde{f}_{n_r}}{\partial x_1} & \cdots & \frac{\partial \tilde{f}_{n_r}}{\partial x_{n_r}} & \frac{\partial \tilde{f}_{n_r}}{\partial u_1} & \cdots & \frac{\partial \tilde{f}_{n_r}}{\partial u_m} \end{bmatrix}. \tag{8}$$

Here, $\tilde{f}_i$ is the ith element in the output of $\tilde{F}$. The elements of $\nabla \tilde{F}_{rr}$ are obtained by automatic differentiation through back-propagation of the neural network models, allowing for constructing matrices $\mathbf{A}_{\mathbf{rr}}$ and $\mathbf{B}_{\mathbf{r}}$.

In this work, $\mathbf{u}^* = 0$ is considered for calculation of a natural equilibrium point of the system, i.e., the equilibrium point related to the uncontrolled system. To compute $\nabla \tilde{F}_{rr}$, an initial guess $\mathbf{x}^*_{\mathbf{r}0}$ is used. The first estimate for the matrix $\mathbf{A}_{\mathbf{rr}}$ is obtained by taking the first n_r columns of $\nabla \tilde{F}_{rr}$, while the remaining columns compose $\mathbf{B}_{\mathbf{r}}$ [see Eq. (8)]. The Newton method is employed to search for an equilibrium point for $\tilde{F}_{rr}$.

Since $\mathbf{u}^* = 0$, the system of the form

$$\mathbf{x}_{\mathbf{r}k+1} = \mathbf{A}_{\mathbf{rr}}\mathbf{x}_{\mathbf{r}k} + \mathbf{b}_{\mathbf{r}} \tag{9}$$

is considered, where $\mathbf{b}_{\mathbf{r}}$ is an invariant vector. In this context, $\mathbf{b}_{\mathbf{r}}$ is introduced to add a bias that shifts equilibrium from the origin. For a given equilibrium point $\mathbf{x}_{\mathbf{r}}{}^*$,

$$\mathbf{x}_{\mathbf{r}}{}^* = \mathbf{A}_{\mathbf{rr}}\mathbf{x}_{\mathbf{r}}{}^* + \mathbf{b}_{\mathbf{r}}, \tag{10}$$

which gives

$$\mathbf{x}_{\mathbf{r}}{}^* = (I - \mathbf{A}_{\mathbf{rr}})^{-1}\mathbf{b}_{\mathbf{r}}. \tag{11}$$

The first guess for $\mathbf{b}_{\mathbf{r}0}$ is obtained by evaluating

$$\mathbf{b}_{\mathbf{r}i} = \tilde{F}_{rr}(\mathbf{x}^*_{\mathbf{r}i}, \mathbf{0}) - \mathbf{A}_{\mathbf{rr}}\mathbf{x}^*_{\mathbf{r}i}, \tag{12}$$

ALGORITHM 1. Equilibrium estimation using linearization of the NNSM.

Obtain reduced states matrix $\mathbf{A_{rr}}$ and $\mathbf{B_r}$ using Eq. (8)
$\mathbf{b}_{\mathbf{r}0} \leftarrow \tilde{F}_{rr}(\mathbf{x}^*_{\mathbf{r}0}, \mathbf{0}) - \mathbf{A_{rr}}\mathbf{x}^*_{\mathbf{r}0}$ ▷ Compute $\mathbf{b}_{\mathbf{r}0}$ using Eq. (12)
$\mathbf{x}^*_{\mathbf{r}1} \leftarrow (I - \mathbf{A_{rr}})^{-1}\mathbf{b}_{\mathbf{r}0}$ ▷ Update $\mathbf{x_r}^*$ using Eq. (11)
for $i = 1$ to $N - 1$ **do** ▷ Iterative process
  $\mathbf{b}_{\mathbf{r}i} \leftarrow \tilde{F}_{rr}(\mathbf{x}^*_{\mathbf{r}i}, \mathbf{0}) - \mathbf{A_{rr}}\mathbf{x}^*_{\mathbf{r}i}$ ▷ Update $\mathbf{b_r}$ using Eq. (12)
  $\mathbf{x}^*_{\mathbf{r}i+1} \leftarrow (I - \mathbf{A_{rr}})^{-1}\mathbf{b}_{\mathbf{r}i}$ ▷ Update $\mathbf{x_r}^*$ using Eq. (11)
  Update reduced states matrix $\mathbf{A_{rr}}$ and $\mathbf{B_r}$ at updated equilibrium point
end for
$\mathbf{x}^* \leftarrow \tilde{F}_r(\mathbf{x}^*_{\mathbf{r}N}, \mathbf{u}_k)$ ▷ Map to full set of sensors
return $\mathbf{x}^*$

with $i = 0$, which comes from the linear approximation of $\tilde{F}_{rr}$. With $\mathbf{b_r}$, $\mathbf{x_r}^*$ can be updated by evaluating Eq. (11), which can subsequently be used to update the reduced states matrix $\mathbf{A_{rr}}$ and $\mathbf{B_r}$ via Eq. (8), and then $\mathbf{b_r}$ via Eq. (12). The iterative process can be repeated to obtain an estimate of the equilibrium point. If desired, the equilibrium point can be mapped to the full set of sensors by evaluating

$$\mathbf{x}^* = \tilde{F}_r(\mathbf{x_r}^*, 0). \tag{13}$$

The complete procedure is synthesized in Algorithm 1. Rather than iterating for a fixed number of steps, one could alternatively employ a stopping criterion based on the size of the update, or estimated error. It is important to mention that this approach is better suitable when the spectrum of $\mathbf{A_{rr}}$ does not contain eigenvalues at $1 + 0\mathrm{j}$ (integrator poles). In these cases, there is a continuous space of possible equilibrium points, any of which can be found by applying the Newton method. An alternative method for refining the equilibrium estimate, which is based on dynamic mode decomposition with control (DMDc) and which we observe can give improved results, will be described in Sec. II D.

C. Neural network controller

NNCs are employed as a candidate for controlling complex nonlinear systems. A nonlinear control law of the form

$$\mathbf{u}_k = K(\mathbf{x}_{\mathbf{r}k}) \tag{14}$$

is implemented as a machine learning model, where the reduced set of states $\mathbf{x}_{\mathbf{r}k}$ are real-time measurements used to evaluate the control signal. A recurrent network is built for the training task as shown in Fig. 2. The model is trained such that a set of initial conditions is brought closer to the equilibrium point $\mathbf{x}^*_{\mathbf{r}0}$ along iterations within a fixed discrete horizon n_h.

The training data consist of a set of measurements $\mathbf{X}_0 = [\mathbf{x}_{\mathbf{r}0,1}\ \mathbf{x}_{\mathbf{r}0,2}\ \ldots\ \mathbf{x}_{\mathbf{r}0,p}]$ containing p samples of the n_r-state system at an initial time $k = 0$. Note that the subscript 0 refers to the initial condition in the context of the finite-time closed-loop recurrent network, not to the simulation initial condition. The following loss function is targeted for minimization:

$$\mathcal{L} = \frac{1}{p\,n_h}\left(\frac{\mathbf{w}_e}{n}\cdot\sum_{j=1}^{p}\sum_{i=1}^{n_h}\mathbf{e}_{i,j}\odot\mathbf{e}_{i,j} + \frac{\mathbf{w}_u}{m}\cdot\sum_{j=1}^{p}\sum_{i=0}^{n_h-1}\mathbf{u}_{i,j}\odot\mathbf{u}_{i,j}\right), \tag{15}$$

$$\mathbf{e}_{i,j} = \mathbf{w}_f \odot (\tilde{\mathbf{x}}_{i,j} - \mathbf{x}^*), \tag{16}$$

where p is the size of the training data set. Here, $\mathbf{w}_f$ is an array consisting of the inverse of the standard deviations of the data sampled for training. As in the NNSM case, this is done in order to avoid states with larger numerical amplitudes from being more important in the cost function.

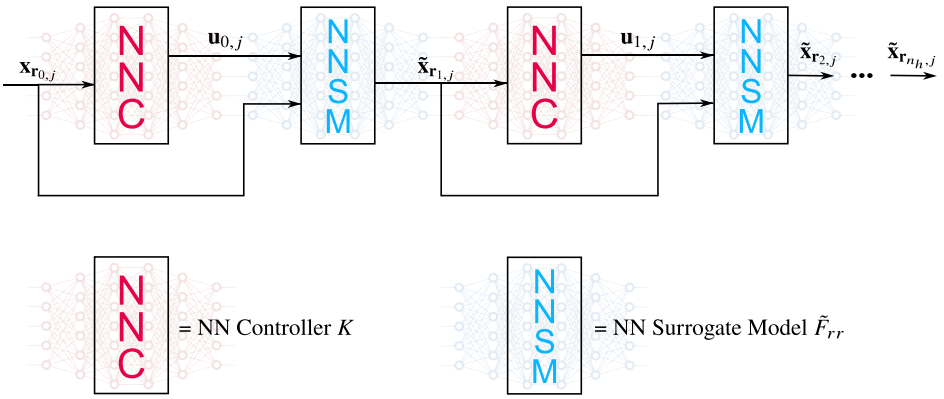

FIG. 2. Schematic of the NNC training where the initial data set is propagated through recurrent evaluations of the NNSM and NNC. Only the NNC weights and biases are updated during training in order to bring the states $\mathbf{x}_{\mathbf{r}1\leqslant i\leqslant n_h,j}$ closer to $\mathbf{x}_{\mathbf{r}}{}^*$.

Furthermore, the weight vector $\mathbf{w}_e$ has all elements equal to $1/n_r$, which averages the result of the sum using an equal weighting of the states. Similarly, $\mathbf{w}_u$ has all elements equal to w_u, a scalar hyperparameter used to penalize the control inputs. Within the scope of this recurrent approach, the states $\tilde{\mathbf{x}}_{i,j}$ are the ones found through forward propagation of an input $\mathbf{x}_{0,j}$, where index i corresponds to the ith time step within the discrete loop and index j is related to the jth state vector in the data set $\mathbf{X}_0$. In short, this loss function is composed of two competing terms: the first one is for penalizing the controlled error of the states relative to the target equilibrium point along the finite horizon, and the second one penalizes the control input energy, also along that same horizon. The parameter w_u can be tuned to prioritize one term over the other. This approach is similar to those used in classical optimal control methods, such as the linear quadratic regulator (LQR).

The hidden NNC layers are implemented with ReLU activation, except for the output layer that has a sigmoid activation, which is normalized to limit the control effort within the desired range. Once the training process is finished, the NNC block is used independently of the training loop to evaluate the control law in real time through Eq. (14). In the present work, all of the NNCs are implemented with a single hidden layer containing eight nodes, which means they are computationally inexpensive as required for real-time applications. Appendix A further describes the hyperparameters employed. General guidelines for the implementation of the proposed NN framework are presented in Appendix B.

D. Linear stability analysis

Linear stability theory can be employed to elucidate mechanisms underlying the generation and amplification of flow instabilities. Here, we aim to employ the proposed neural network models to conduct linear stability analyses of unsteady flows, thereby illuminating physical mechanisms associated with unstable equilibrium points. In particular, with such an approach, it will be possible to compute the frequencies and growth rates of the most unstable modes and their associated eigenfunctions.

In an unsteady flow, a Reynolds decomposition allows the splitting of the flow states $\boldsymbol{x}(\boldsymbol{x}_c, t)$ in a base flow $\boldsymbol{x}^*(\boldsymbol{x}_c)$, here obtained at the equilibrium point, plus a time-dependent fluctuation component $\boldsymbol{x}'(\boldsymbol{x}_c, t)$. In the present notation, $\boldsymbol{x}_c$ represents the spatial coordinates. If the fluctuations are sufficiently small, the equations can be linearized about the base flow and the Navier-Stokes equations can then be written in the following linear form:

$$\frac{\partial \boldsymbol{x}'}{\partial t} = \mathcal{A}\boldsymbol{x}', \tag{17}$$

where the matrix $\mathcal{A} = \mathcal{A}(\boldsymbol{x}^*)$ is a linear operator. For a system that is homogeneous in the streamwise (x_c) and spanwise (z_c) directions, the evolution of linear disturbances by Eq. (17) can be investigated by a direct analysis of the operator $\mathcal{A}$ with the transformation

$$\boldsymbol{x}'(x_c, y_c, z_c, t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{\boldsymbol{x}}(\alpha, y_c, \beta, \omega) e^{\mathrm{j}(\alpha x_c + \beta z_c - \omega t)}\, d\alpha\, d\beta\, d\omega, \tag{18}$$

where both the streamwise and spanwise wavenumbers $\alpha, \beta \in \mathbb{R}$, and the frequency $\omega \in \mathbb{C}$.

In discrete form, and under the transformation of Eq. (18), Eq. (17) can be written as

$$-\mathrm{j}\omega\hat{\boldsymbol{v}} = \mathbf{A}\hat{\boldsymbol{v}}. \tag{19}$$

In this case, the linear operator becomes $\mathbf{A} = \mathbf{A}(\boldsymbol{x}^*, \alpha, \beta)$, and it can be analyzed for each separate wavenumber pair (α, β) as an eigenvalue problem:

$$\mathbf{V}\boldsymbol{\Lambda} = \mathbf{A}\mathbf{V}. \tag{20}$$

In this equation, the columns $\hat{\boldsymbol{v}}_j$ of $\mathbf{V}$ are the eigenvectors of $\mathbf{A}$, and the eigenvalues $\lambda_j = -\mathrm{j}\omega_j$ are the corresponding diagonal entries of $\boldsymbol{\Lambda}$. Here the frequency and growth rate are the real and imaginary parts of ω_j, respectively.

There are several ways in which the identified NNSMs and NNCs can be leveraged to conduct stability analyses of the underlying dynamical systems. Most directly, we could utilize the matrix $\mathbf{A_{rr}}$ arising from the linearization of the NNSM obtained through back-propagation [Eq. ((8). Rather than linearizing the global nonlinear model, we can alternatively identify a linear model directly from data taken near the equilibrium point, which is the method used in the present work. This approach amounts to performing a linear regression inspired in DMDc [67]. The idea comes from the fact that the controlled system can be slightly perturbed, producing a rich set of data near the equilibrium point, where the dynamics of the system tends to be approximately linear.

Given a data set $\mathbf{X} = [\mathbf{d}_0 \dots \mathbf{d}_{p-1}]$ containing p consecutive full state measurements, a data set $\mathbf{U} = [\mathbf{u}_0 \dots \mathbf{u}_{p-1}]$ containing the control inputs history, and a data set $\mathbf{X}' = [\mathbf{d}_1 \dots \mathbf{d}_p]$ containing full state measurements with a unit shift, matrices $\mathbf{A}$ and $\mathbf{B}$ can be inferred as follows:

$$\mathbf{G} = \mathbf{X}'\boldsymbol{\Omega}^{\dagger}, \tag{21}$$

$$\mathbf{G} \approx \begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}, \tag{22}$$

$$\boldsymbol{\Omega} = \begin{bmatrix}\mathbf{X} \\ \mathbf{U}\end{bmatrix}. \tag{23}$$

As shown, matrix $\boldsymbol{\Omega}$ is built from $\mathbf{X}$ and $\mathbf{U}$. By calculating its Moore-Penrose inverse $\boldsymbol{\Omega}^{\dagger}$, it is possible to compute $\mathbf{G}$, from which approximations for $\mathbf{A}$ and $\mathbf{B}$ can be extracted.

Note that $\mathbf{X}$ is composed of the deviation of the states from their estimated equilibrium values, as shown in Eq. (7). Admitting that the estimation of this equilibrium $\mathbf{x}^*$ is not exact, the proposed linear regression can be contaminated. To account for such imperfect estimation, a modification to this method is proposed, where a matrix $\mathbf{O} = [1 \dots 1]_{(1\times p)}$ containing 1s is employed. This allows for the modified procedure:

$$\mathbf{H} = \mathbf{X}'\boldsymbol{\Psi}^{\dagger}, \tag{24}$$

$$\mathbf{H} \approx \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{c}\end{bmatrix}, \tag{25}$$

$$\boldsymbol{\Psi} = \begin{bmatrix}\mathbf{X} \\ \mathbf{U} \\ \mathbf{O}\end{bmatrix}. \tag{26}$$

For systems with no integrator poles, this modification allows for the estimation of a system of the form

$$\mathbf{d}_{k+1} = \mathbf{A}\mathbf{d}_k + \mathbf{B}\mathbf{u}_k + \mathbf{c}, \tag{27}$$

where $\mathbf{c}$ is a constant vector that displaces the equilibrium point of the new states $\mathbf{d}$ from zero. Beyond providing a better estimation of $\mathbf{A}$ to conduct stability analysis, the addition of the constant $\mathbf{c}$ allows for the correction of the previous estimate of $\mathbf{x}^*$. Although this is not done in the current work, it could also be used, for example, to compensate control loop measurements to improve NNC results. Since the NNC is trained to control the NNSM and, furthermore, $\mathbf{x}^*$ is an equilibrium point for the NNSM, $\mathbf{c}$ could be used to adapt the NNC loop for the real plant.

To reduce computational costs, two different sets of matrices are used based on the sparse configuration obtained as described in Sec. II A:

(i) The first is obtained by using the reduced states data set $\mathbf{X_r} = [\mathbf{d}_{\mathbf{r}0} \dots \mathbf{d}_{\mathbf{r}p-1}]$. Replacing $\mathbf{X}$ by $\mathbf{X_r}$, the algorithm produces a matrix $\mathbf{A}_{\mathbf{r}(n\times n_r)}$ which maps the reduced set of sensors to the full set at the next time step, as well as $\mathbf{B}$ and $\mathbf{c}$.

(ii) The second set of matrices is obtained by also using $\mathbf{X_r}$ with the further substitution of $\mathbf{X}'$ by a reduced matrix $\mathbf{X_r}'$. This produces the same matrices $\mathbf{A_{rr}}_{(n_r\times n_r)}$ and $\mathbf{B}_{\mathbf{r}(n_r\times m)}$ shown in Sec. II B—with potentially better accuracy—as well as the reduced bias vector $\mathbf{c_r}$.

A stability analysis can be conducted by finding the eigenvectors $\mathbf{V}_{\mathbf{r}i}$ and eigenvalues λ_i from the square matrix $\mathbf{A_{rr}}$, where $i = 1, 2, \dots, n_r$. Since each vector $\mathbf{V}_{\mathbf{r}i}$ is an eigenvector of $\mathbf{A_{rr}}$, each $\mathbf{A_{rr}}\mathbf{V}_{\mathbf{r}i}$ is also an eigenvector of $\mathbf{A_{rr}}$, which allows the mapping

$$\mathbf{V}_{i(n\times 1)} = \mathbf{A}_{\mathbf{r}(n\times n_r)}\mathbf{V}_{\mathbf{r}i(n_r\times 1)} \tag{28}$$

to estimate the eigenvectors across all sensor states. This mapping is employed in this work for visualization of modes projected over the full set of candidate sensors.

III. ITERATIVE TRAINING

A key aspect of the proposed methodology is the development of a training methodology that ensures that both the NNSM and NNC are trained with sufficient quantities of data, particularly in the regions of state space where model accuracy is important for control effectiveness. To train and improve the neural network models employed in this work, an iterative training algorithm is proposed. Four different modes of data sampling are considered for training the NNSM and the NNC. Each mode produces data that are compiled for training both networks. They are as follows:

(1) Uncontrolled sweep mode: the system is perturbed by an open-loop control signal with no closed-loop control.

(2) Controlled sweep mode: the system is controlled in closed loop at the same time that it is perturbed by an open-loop signal.

(3) Release mode: the plant runs with no control and no open-loop perturbation, allowing for better visualization of the next mode and to test whether the system can be successfully controlled from an initially uncontrolled state.

(4) Control mode: the system is controlled in closed loop with no open-loop perturbation.

All open-loop perturbations applied in this work are random staircase signals with constant step length. This choice is inspired in classical approaches for modeling linear systems that take advantage of square wave perturbations, which are simple to implement and at the same time able to excite a range of frequencies of the dynamical systems. In the case of linear systems, a fixed amplitude disturbance is sufficient, since a different amplitude would roughly provide the same response multiplied by a constant factor. In this work, a random staircase is used to account for multiple amplitudes, provided the applications involve nonlinear systems. It is important to mention that, although it is expected that this approach should provide suitable sweeping of the state space, there is no guarantee that specific regions are reached, for example, if the states get stuck in regions presenting unstable limit cycles or hysteresis [51].

The *uncontrolled sweep* is the first mode employed so as to produce data with control input history. A number of samples are gathered from the perturbed plant. These first samples are used to compute the means and standard deviations used for data normalization of the neural network's inputs and outputs. These data are also used for training the NNSM, estimating its equilibrium point

ALGORITHM 2. Data sampling algorithm.

```
for i = 1 to n do
    if i==1 then
        Run uncontrolled sweep mode to gather data
        Compute parameters for normalization
    else
        Reduce the amplitude of open-loop perturbations by a factor α
        Run controlled sweep mode to gather data around the estimated equilibrium
    end if
    Update the database with the new data
    Train NNSM with the updated database
    Truncate negligible weights in the sparsity layer
    Estimate equilibrium from NNSM
    Train NNC with the updated database
    Run release mode to allow the system to reach its natural behavior
    Gather data in release mode
    Run control mode to drive the system to a new behavior and test control effectiveness
    Gather data in control mode
end for
```

$\mathbf{x_r}^*$, and training the NNC, all for the first time. The NNSM sparsity layer has its negligible weights truncated according to a threshold (further described in Appendix A), which accounts for identifying and updating the relevant sensor locations.

The second mode to be run is the *release* mode, which allows the system to achieve its natural (uncontrolled) behavior, i.e., a limit cycle or a chaotic attractor, for example. The *control* mode is applied after the system completely or partially returns to its natural operation. When the NNC is turned on, the dynamics of the controlled system drives the plant states to work under a new behavior, which can tend to a new limit cycle, to a new chaotic attractor, or even to an equilibrium point—which can either be or not be desirably close to the estimated equilibrium $\mathbf{x}^*$.

Data are gathered in each of these three steps, building a database that grows along the iterative process. After they are run, the process is repeated for a number of iterations, but instead of doing an uncontrolled sweep, the controlled sweep mode is applied in every iteration beyond the first one. In each NNSM training step, a set of n_s samples is taken, including measurements from the current and the previous iterations. We use the total number of measurements available, unless that number is greater than a threshold N_s. In this case, we randomly sample N_s measurements from the entire available data set. Similarly, in each NNC control step, we randomly sample p points from the entire data set to build $\mathbf{X}_0$. The idea is to leverage the fact that the controlled system should have a tendency to get closer to the equilibrium point, and by perturbing the controlled system appropriately, a larger set of data is obtained around this equilibrium. When a new iteration begins, the amplitudes of the open-loop perturbations are reduced by a constant factor $0 < \alpha \leqslant 1$, so that data are gathered closer to the actual plant equilibrium. At each iteration, the NNSM and the NNC are retrained with new data collected from the full database and the equilibrium point is updated using the retrained NNSM. Algorithm 2 synthesizes the steps described, and a schematic showing the phase portrait for the Lorenz system is also presented in Fig. 3. Each iteration in this figure depicts 1000 measurements sampled during the sweep modes (uncontrolled for iteration 1, controlled for subsequent iterations). As the iterations advance, the closed-loop control tends to bring states closer to the origin, which is an equilibrium point for the Lorenz system. This enhances the estimate of the equilibrium point, and it can be seen in the smaller limits of the axes, as shown for the higher iterations. Also, beginning in iteration 7, since the perturbations are reduced between iterations, they become unable to shoot the controlled states out of the main stable orbit around the now-stabilized equilibrium.

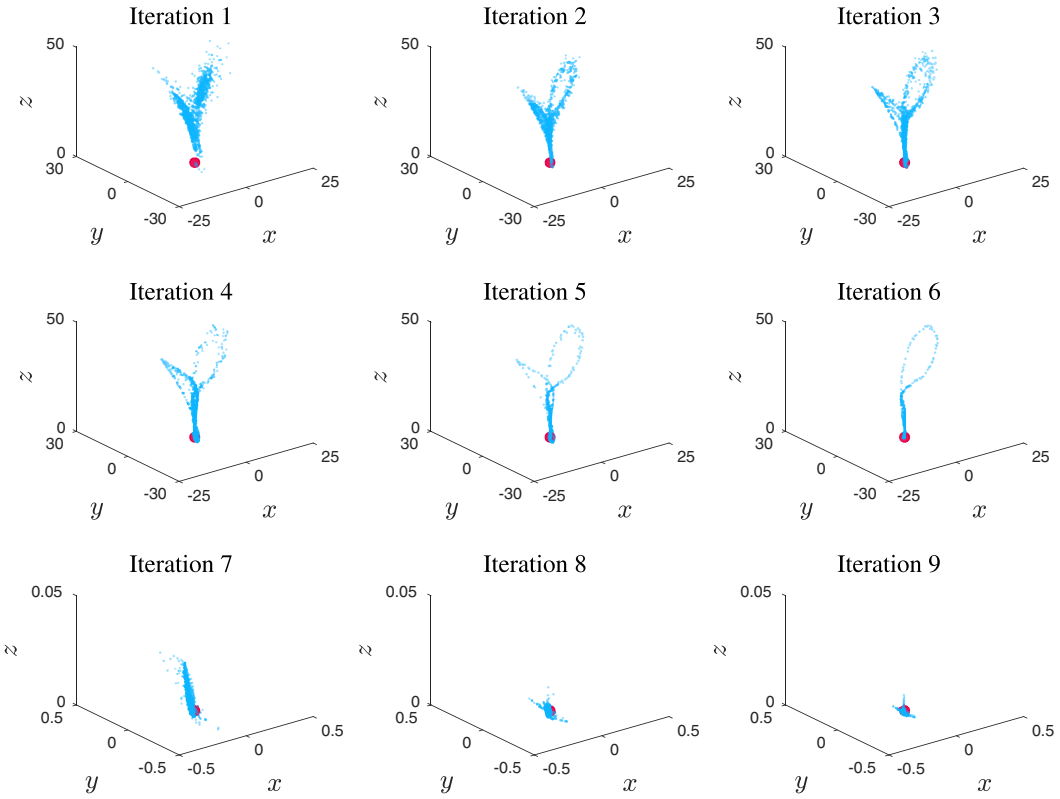

FIG. 3. Phase portraits of the Lorenz system for data sampled during the sweep modes at different iterations. Note that for iterations 7–9 the axes limits are considerably smaller. The red dot indicates the origin, which is an equilibrium point for the Lorenz system.

For the stability analysis, only the data produced by the last sweep mode is used since it may contain more accurate data near the equilibrium point, hopefully ruled by approximately linear dynamics. One of the advantages of using such a type of data is enabling DMD-based techniques to identify structures corresponding to eigenvectors of the equilibrium-linearized system, even for systems exhibiting strong nonlinearities.

IV. STUDY CASES

This section outlines the four systems that will be used to test and validate the proposed methodology. For each of these cases, the iterative training process to identify the NNSM and NNC is conducted by using the sampled training data and minimizing the loss functions presented in Eqs. (2) and (15). In some test cases, we utilize this framework to conduct stability analysis and/or sensor placement tasks.

A. Lorenz equations

The first study case for this work is the Lorenz system, whose dynamics are described by the following set of nonlinear ordinary differential equations:

$$\dot{x} = \sigma(y - x), \tag{29}$$

$$\dot{y} = \rho x - y - xz + u, \tag{30}$$

$$\dot{z} = -\beta z + xy, \tag{31}$$

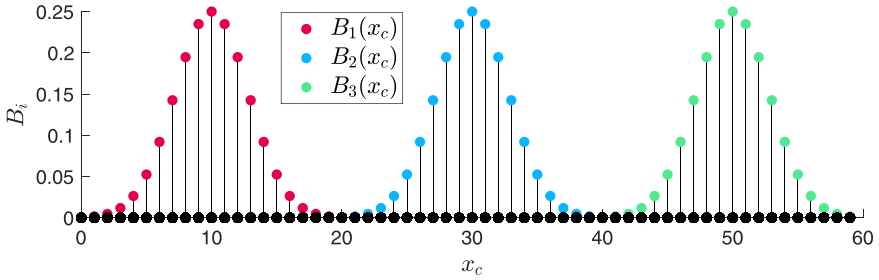

FIG. 4. Actuation and sensor setup for controlling the modified KS equation. The colored dots represent the control input profiles B_i at each of the 60 black dots measured initially (candidate sensors).

where $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ are chosen for the present analysis. With this set of parameters, the uncontrolled system $[u(t) = 0]$ behaves as a chaotic attractor. The goal is to apply the techniques proposed in this work to find an equilibrium point, i.e., to stabilize the system bringing the states towards this point through an NNC law $u_k = K(x_k, y_k, z_k)$. We also compute a linear model to obtain the system poles. Since the Lorenz system with the chosen parameters presents three distinct unstable equilibrium points with $u(t) = 0$, the equilibrium point search can lead to different results that strongly depend on the choice of the initial guess $\mathbf{x}^*_{\mathbf{r}0}$. Here, we choose $\mathbf{x}^*_{\mathbf{r}0} = [0.1, 0.1, 0.1]$, so convergence to [0, 0, 0] is expected.

B. Modified Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky (KS) equation is given by the following nonlinear partial differential equation:

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x_c} = -\frac{1}{R}\left(P\frac{\partial^2 v}{\partial x_c^2} + \frac{\partial^4 v}{\partial x_c^4}\right), \tag{32}$$

where R is equivalent to the Reynolds number in a fluid flow, and P represents a balance between energy production and dissipation. This system has been used to emulate, for example, combustion instabilities in flame fronts [68,69] and hydrodynamic instabilities in shear flows [70]. In this context, x_c is the spatial coordinate and $v(x_c)$ is the variable of interest.

It is clear that, for any constant c, $v(x_c) = c$ corresponds to an equilibrium point, i.e., $\partial v/\partial x_c = 0$. Since we are interested—among other goals—in computing an equilibrium point for this system, we propose a modified version of the KS equation given by

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x_c} = -\frac{1}{R}\left(P\frac{\partial^2 v}{\partial x_c^2} + \frac{\partial^4 v}{\partial x_c^4}\right) - \frac{Q}{L}\int_0^L (v(x_c) - V)\,dx_c + \sum_{i=0}^{m} B_i(x_c)u_i. \tag{33}$$

The new term $-Q/L\int_0^L (v(x_c) - V)\,dx_c$ corresponds to a spatial average of the difference between $v(x_c)$ and a chosen equilibrium V. We make use of this modification to ensure that the equilibrium point $v(x_c) = V$ is unique for $u_i(t) = 0$. A globally unstable configuration is set by choosing $R = 0.25$, $P = 0.05$, $Q = 0.0005$, $V = 0.2$, and $L = 60$ with periodic boundary conditions. The control term $\sum_{i=0}^{m} B_i(x_c)u_i$ comes similarly to the approach proposed by Fabbiane *et al.* [70], where m is the number of control inputs, $B_i(x_c)$ is a window function for each actuator, and u_i represents the control signals.

The equation is discretized in the present simulation with $\Delta x_c = 1$ and $\Delta t = 0.025$, producing the state vector $\mathbf{x}_{(60\times 1)}$. Sampling for control is made each 400 time steps, thus giving a control time step $\Delta t_c = 10$. Three actuators are employed by using Gaussian window functions centered at $x_c = 10$, $x_c = 30$, and $x_c = 50$. Sampling of v is conducted at each of the 60 discrete grid points. Figure 4 shows a sensor and actuation schematic. In this context, the goal is to train an

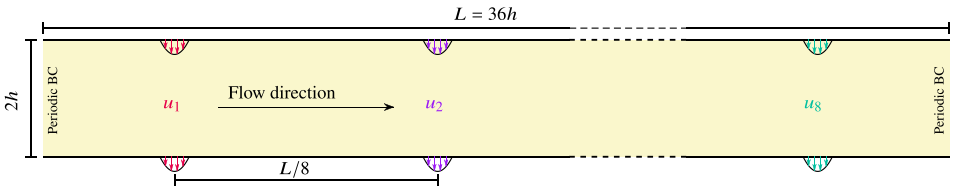

FIG. 5. Actuation setup for controlling the 2D channel flow. Each pair of actuators works in opposition, ensuring zero-net mass flux.

NNC to allow for the evaluation of the control law $\mathbf{u}_{(3\times1)} = K(\mathbf{x}_{\mathbf{r}(n_r\times1)})$, where $n_r \leqslant 60$. In this test case, we conduct equilibrium estimation, sensor placement, control, and stability analysis. For the equilibrium estimation, we choose an initial guess equal to 0.1 for each point ($\mathbf{x}^*_{\mathbf{r}0} = [0.1, \ldots, 0.1]$).

C. Periodic 2D channel flow

Another problem we consider is a streamwise-periodic channel flow. We employ numerical simulations using the open-source Nek5000 computational fluid dynamics (CFD) code previously applied for RL control [20] to solve the incompressible Navier-Stokes equations for a Reynolds number $\mathrm{Re} = 8000$, based on the half-channel height h and the velocity at the channel center line, U. The channel length is set to $L = 36h$ so that two unstable modes appear, at streamwise wavelengths L/5 and L/6. A total of $m = 8$ control inputs $u_1, \ldots, u_8$, which forms the control input vector $\mathbf{u}_{(8\times1)}$, modulates the velocities of eight pairs of actuators, which are implemented as Dirichlet boundary conditions imposing a specified jet velocity. Each pair works in opposition to ensure a zero-net mass flux, being weighted by a parabolic window function as depicted in Fig. 5.

The distribution of candidate sensors is presented in Fig. 6. The chosen locations are concentrated near the wall to capture velocity fluctuations from hydrodynamic instabilities. A vertical line of 50 sensors is also employed for comparison with stability analysis computed from the solution of the Orr-Sommerfeld equation. In total, 332 candidate sensor positions are chosen, where horizontal and vertical velocities $v_x(x_c, y_c)$ and $v_y(x_c, y_c)$ are sampled, totaling 664 signals that compose the state vector $\mathbf{x}_{(664\times1)}$, which can be used to evaluate $\mathbf{u}_{(8\times1)} = K(\mathbf{x}_{\mathbf{r}(n_r\times1)})$, where $n_r \leqslant 664$. The initial guess $\mathbf{x}^*_{\mathbf{r}0}$ for the equilibrium point is 1 for v_x probes and 0 for v_y probes. Results presented in the next section show that for this test case it is not required to have a more sophisticated guess (such as the mean flow). For the present channel flow, we are interested in equilibrium estimation, sensor placement, control, and stability analysis.

D. Confined cylinder flow

Finally, the study of a 2D confined flow past a cylinder is presented. For this task, a Nek5000 setup [20] is employed considering a Reynolds number $\mathrm{Re} = 150$ based on the cylinder diameter D

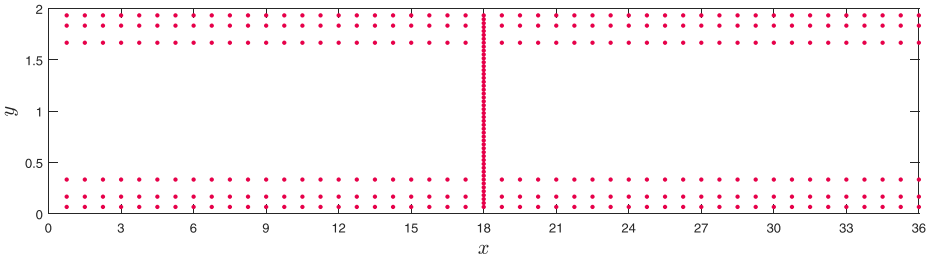

FIG. 6. Initial candidate sensor positions for the channel flow. The dimensions are distorted for better visualization.

FIG. 7. Initial candidate sensor positions for the confined cylinder flow. The box represents the spatial domain used in the simulation.

and maximum velocity U of the inflow profile. The initial configuration of sensors is equivalent to the one presented by Rabault *et al.* [17], depicted in Fig. 7, where 153 locations are chosen, totaling 306 velocity measurements for u and v velocity components [leading to the state vector $\mathbf{x}_{(306\times1)}$]. No-slip wall boundary conditions are applied at the top and bottom limits of the domain.

The actuation scheme is presented in Fig. 8. A single control input u modulates minijets in opposition, thus providing zero-net mass flux. The actuation here is implemented using Dirichlet boundary conditions, which set the normal velocities at the wall. For this problem, we are interested in finding an equilibrium point (only to estimate a control setpoint), sensor placement, and flow control through the NNC law $u = K(\mathbf{x}_{(n_r\times1)})$, where $n_r \leqslant 306$. We use the same strategy as in the channel flow case to make an initial guess for the equilibrium point, which is choosing 1 for v_x probes and 0 for v_y probes.

V. RESULTS

A. Lorenz equations

To assess the performance of the proposed techniques, results for the low-order Lorenz system are presented. Starting with the problem of estimating an equilibrium point for this system, Fig. 9 presents the values of each state at each iteration. Two plots compare this computation using both the back-propagation-based linearization of the neural network (as described in Sec. II B) and the alternative version employing the modified DMDc method proposed in Sec. II D. In both cases, a convergence towards equilibrium at $x = y = z = 0$ is seen. It is clear that, for the initial iterations, better approximations are obtained using back-propagation [Fig. 9(a)]. The fact that DMD-based techniques rely on data sampled from approximately linear systems compromises their performance when the plant is strongly nonlinear. In some cases, the linearization of a nonlinear model at the point of interest can provide better results when nonlinearities cannot be neglected, as shown in Ref. [51]. On the other hand, for the later iterations, the DMD-based approach is able to provide more accurate results, since the closed-loop control is able to maintain states near equilibrium, i.e., operating almost as a linear system [Fig. 9(b)].

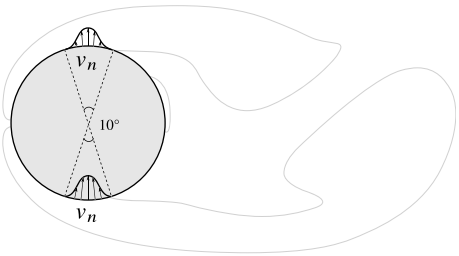

FIG. 8. Actuation scheme applied to the cylinder flow. Blowing and suction devices in opposition are modulated by a single control input.

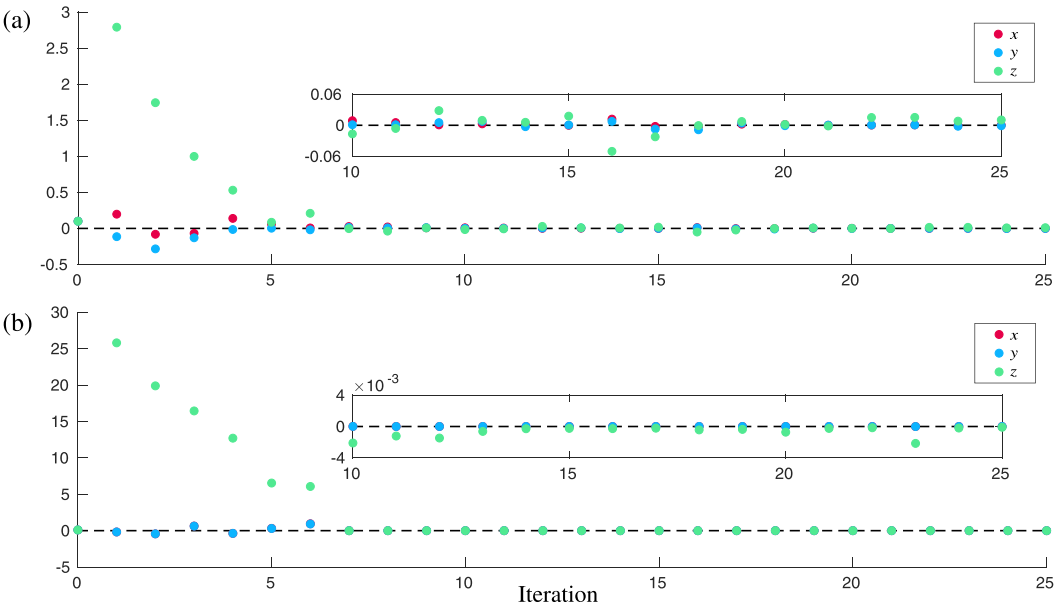

FIG. 9. Equilibrium point estimation along iterations. (a) The estimation through back-propagation, where more accurate estimates are made at first iterations. (b) The fixed estimation through the DMDc variant, where better approximations are found in the last iterations. In this case, the red dots are hidden behind the blue ones.

The stabilization of the controlled Lorenz system is depicted in Fig. 10, where signals are presented against time for iterations 1 and 4. For the former case, depicted in Fig. 10(a), it is still not possible to obtain a stabilizing controller. This might be due to poor estimation of the dynamics near the equilibrium point obtained for such iteration. On the other hand, at iteration 4 shown in Fig. 10(b), the trained NNC is able to fully stabilize the plant, bringing the states close to $x = y = z = 0$. This case was studied with longer periods in which the control was kept turned on (not shown here for brevity), revealing that the system was indeed stabilized, being able to return to and remain at equilibrium even after transient perturbations. As seen in Fig. 9(a), the equilibrium point for iteration 4 is estimated at $x \approx 0.14$, $y \approx -0.01$, and $z \approx 0.53$, which is close enough to allow for the NNC to stabilize the plant (though subsequent iterations have much more accurate estimates).

Finally, a modal stability analysis is conducted, where poles are computed for the controlled Lorenz system. The modified DMDc approach is applied to data obtained from a single iteration of the sweep mode. Similarly to what is observed in Fig. 9(b), later iterations tend to provide better approximations of equilibria. Figure 11 presents results for iterations 4 and 9. For comparison, the Lorenz system equations are linearized analytically, and the obtained poles s_i are discretized so that $z_i = e^{s_i \Delta t}$. These poles are presented as "ground truth" in Fig. 11. As also shown in Fig. 3, at iteration 4, the perturbed controlled system is driven to nonlinear orbits. At iteration 9, sampled states are concentrated much closer to equilibrium, thus providing better approximations through linear regression (see Fig. 3). It is important to remember that these are not the linearizations employed for estimating the equilibrium points used to train the NNC. For that, we use the back-propagation approach, also presented for iteration 4 in Fig. 11. This explains why, even though the linearization is poorly fit to the actual system, the system is still stabilized. Further studies involving the proposed Lorenz system are presented in Appendix C, where we briefly discuss some aspects regarding the robustness of the designed control systems.

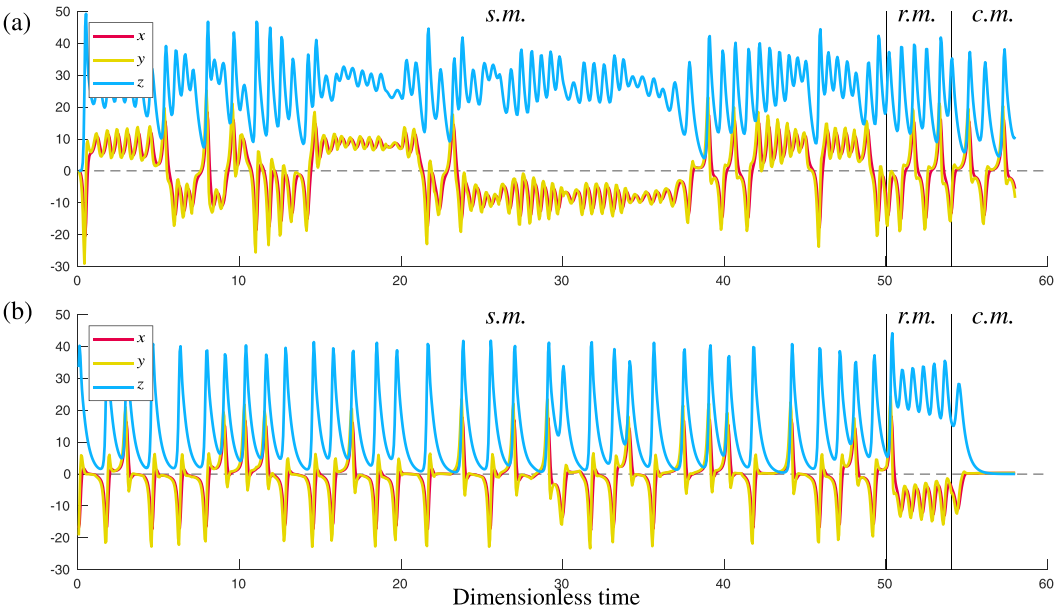

FIG. 10. Sweep, release, and control modes (s.m., r.m., and c.m., respectively) for iterations (a) 1 and (b) 4. The trained NNC is unable to stabilize the system in iteration 1, in contrast to the trained control system in iteration 4, where stabilization is achieved. This can be seen by comparing both control modes.

B. Modified Kuramoto-Sivashinsky equation

For the modified KS equation, two different training approaches are presented. They only differ in terms of application of the sparsity layer. One of the cases does not include the sparsity layer on the neural network (i.e., the full set of sensors is employed), whereas the other case includes a sparsity layer to reduce the number of sensors required. In this second case, the total number of sensors used to infer a model for the system dynamics, as well as to conduct closed-loop control, is reduced from 60 to 9, as represented in Fig. 12. In both cases, the trained process is conducted along ten iterations of sweep, release, and control. We observe in Fig. 12 that the sparse sensors are selected to be approximately evenly distributed throughout the domain, with three sensors located

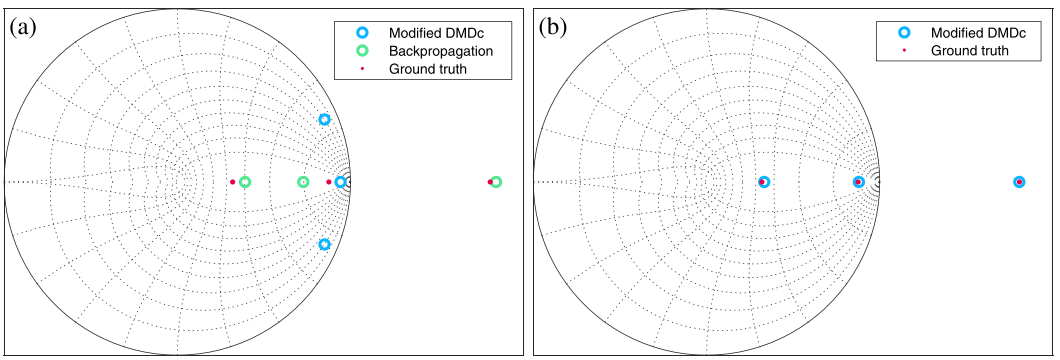

FIG. 11. Poles of the discrete-time Lorenz system obtained by linearization of the controlled system compared to ground truth positions. (a) At iteration 4, the approximation is still not satisfactory, which is explained by the nonlinear dynamics present in the data employed. (b) At iteration 9, the estimated poles are found considerably closer to the ground truth solution.

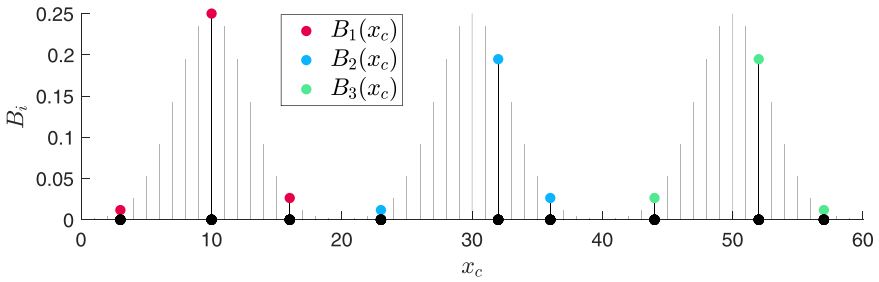

FIG. 12. Sparse configuration obtained in training for the case with sensor placement, where nine sensors are kept for the KS equation.

across each of the three Gaussian functions defining the actuators, and with one sensor at or slightly downstream of each of the actuator peaks.

Figure 13 shows the calculation of the equilibrium points along x_c found in each case using Newton's method and the modified DMDc. It is possible to observe in Fig. 13(a) that the modified DMDc is able to enhance the accuracy of the equilibrium point when the full set of sensors is employed. However, as a trade-off for reducing the number of sensors, Fig. 13(b) shows that the estimate of the equilibrium is worsened with sparse sensors. Despite this, the trained NNC is still able to stabilize the system, as exemplified in Fig. 14, where the waves are controlled until convergence. We further observe that the system remains stable as long as the controller is turned on.

For the modal stability analysis of the modified KS equation, the computed poles are shown in Fig. 15. The ground truth poles are obtained by linearizing Eq. (33) with a discretization performed using fourth-order centered finite difference schemes for all spatial derivatives. The poles are found by computing the eigenvalues s_i of the matrix built from the approximation (with periodic boundary conditions) and transformed to their discrete version $z_i = e^{s_i \Delta t}$. The map obtained with the proposed technique is fairly accurate, with good preservation of frequency and growth rates of the eigenvalues closest to the unit circle. Since the sparse case is composed of only nine sensors, only nine eigenvalues are found.

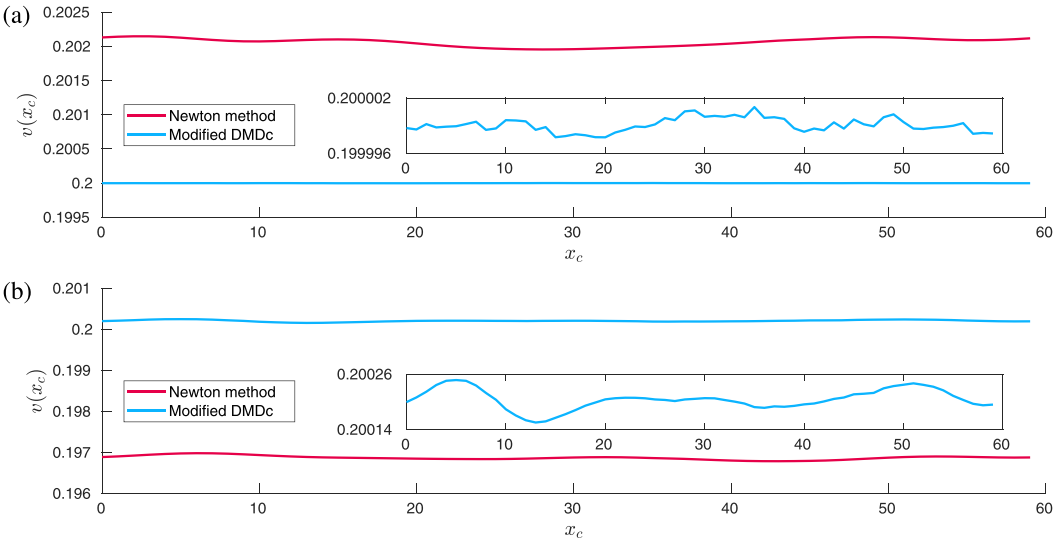

FIG. 13. Estimated equilibrium points of the KS equation found with the (a) full and (b) sparse set of sensors. The true equillibrium is $v(x_c) = 0.2$.

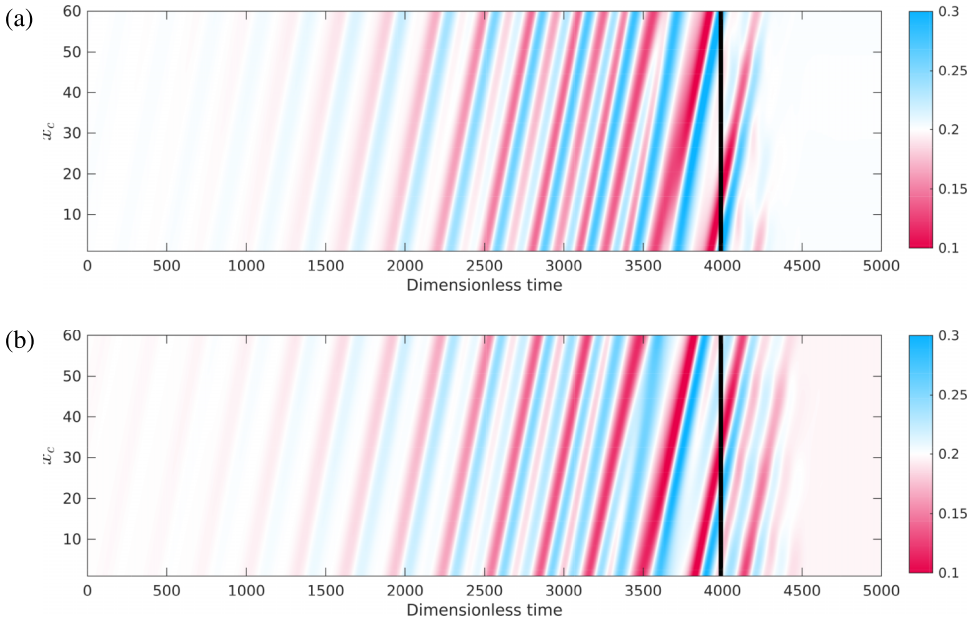

FIG. 14. Space-time map showing the growth (decay) of disturbances in the KS equation before (after) turning the controller for the (a) full and (b) sparse set of sensors. The black vertical line highlights the instant when the control is turned on.

C. Periodic 2D channel flow

The 2D channel flow case is studied with dynamic sensor placement. The final distribution of probes after 19 iterations of training (with sweep, release, and control modes) is presented in Fig. 16. It is possible to notice the preference for horizontal velocity probes (u) rather than vertical (v) ones. The sensor placement is able to achieve a reduction from 664 to 181 signals; 148 and 33 correspond to u and v velocity probes, respectively. The spatial distribution of these chosen sensors shows that almost all of the near-wall u sensors are included, possibly highlighting the importance

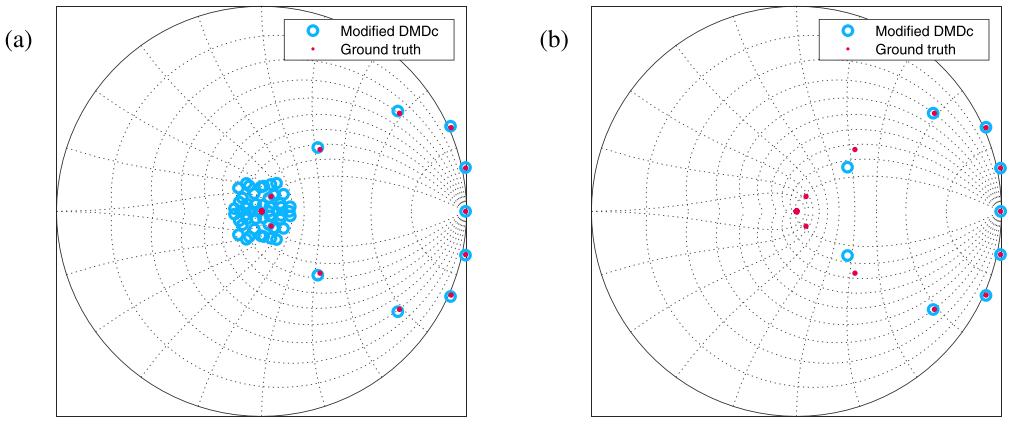

FIG. 15. Poles of the discrete-time modified KS system obtained by linearization of the controlled system compared to the ground truth solution. The (a) full and (b) sparse sensor approaches are compared. Most of the ground truth poles are concentrated close to the origin, making them harder to visualize.

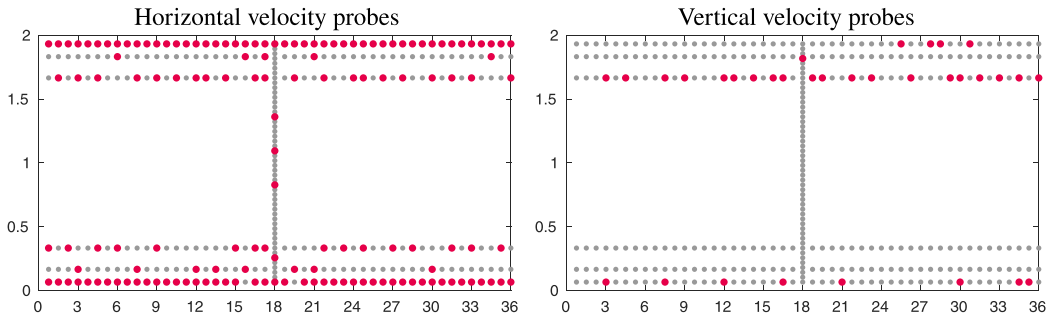

FIG. 16. Probe locations employed with sparse sensing, where the number of probes is reduced from a total of 664 to 181, from which 148 are used for the horizontal velocity component, while 33 are used for the vertical one. The grey dots are deactivated by the sparsity layer.

of near-wall information for estimating and controlling (with wall-based actuation) the system dynamics.

With feedback of the signals probed at the locations obtained through training, the NNC is able to completely stabilize the flow. A comparison of the uncontrolled and controlled flows is presented in Fig. 17, where v-velocity contours are shown. As expected, the actuation and the flow field do not converge to the exact equilibrium, since its estimate is not perfect, especially considering that, for control, we employ the approximation obtained through Newton's method (see Sec. II D). The controlled flow is presented in two ways: (1) using the same contour range of the uncontrolled flow, and (2) with a more saturated contour range that allows one to notice some small residual velocity fluctuations. When employing the same scale as that of the uncontrolled case, the bias becomes nearly imperceptible. For the uncontrolled case, the snapshot represents a frame captured from an unsteady flow, where the flow structures are transported from left to right, while the controlled case is simply a steady state.

The present flow includes two unstable modes. These are computed through the eigendecomposition of the associated Orr-Sommerfeld operator, which is discretized using a pseudospectral method with Chebyshev polynomials utilizing the toolbox by Weideman and Reddy [71]. The wavelengths of the unstable modes obtained by the Orr-Sommerfeld equation correspond to 1/6 (mode 1) and 1/5 (mode 2) of the channel length. This is in agreement with the wavelengths of the dominant structures observed in simulations of the uncontrolled nonlinear system, as shown in Fig. 17(a).

By applying the modified DMDc approach to the data obtained near equilibrium, two unstable pairs of modes are found, corresponding to the two unstable modes of the true linearized system. For

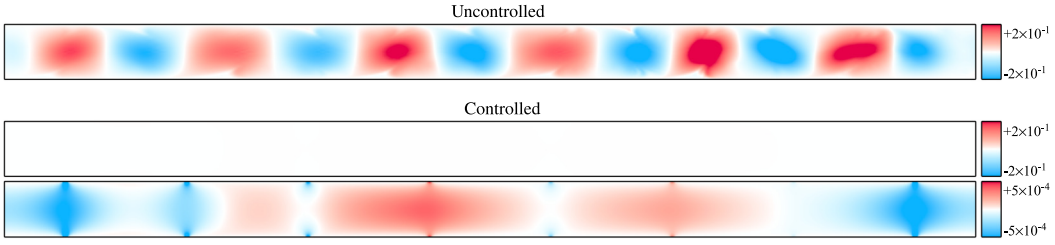

FIG. 17. Comparison of uncontrolled and controlled flows through visualization of v-velocity contours (see Movie 1, Supplemental Material [74]). For the controlled case, a more saturated contour range is also shown in the bottom plot, and one can see that the actuation does not converge to zero due to an imperfect estimation of the equilibrium.

TABLE I. Eigenvalues for unstable modes 1 and 2 for 2D channel flow. The real and imaginary parts correspond to the growth rates and frequencies, respectively.

	Mode 1	Mode 2
Ground truth	$1.8601 \times 10^{-3} \pm \mathrm{j}2.6403 \times 10^{-1}$	$6.8185 \times 10^{-4} \pm \mathrm{j}2.0206 \times 10^{-1}$
Modified DMDc	$2.0545 \times 10^{-3} \pm \mathrm{j}2.6396 \times 10^{-1}$	$7.4956 \times 10^{-4} \pm \mathrm{j}2.0214 \times 10^{-1}$

this case, the discrete poles z_i found with the modified DMDc are converted to a continuous version $s_i = \ln(z_i)/\Delta t$, so that one can directly infer the corresponding growth rates and frequencies. A comparison between the true and estimated unstable eigenvalues is shown in Table I, where close agreement is observed. In these results, the imaginary part corresponds to the oscillation frequency while the real component provides the growth rate. The higher relative error found for the growth rate when compared to the frequency can be explained by the transient timescales being too slow compared to the discretization time Δt. Since Δt needs to be small enough to capture the signals without significant aliasing—i.e., the sampling rate needs to be at least large enough to capture the most important frequencies present in the measured signals—and the timescale of the oscillations is considerably faster than the growth rate, the discrete time horizon to capture the long-term growth becomes too large. In this context, we ensure that the sampling time Δt is chosen so that the sampling rate $1/\Delta t$ is always larger than twice the highest frequency present in the signal, thus satisfying the Shannon-Nyquist sampling rate required to resolve this frequency. The value chosen here is $\Delta t = 1.05$ units of time, which corresponds to an angular frequency of $2\pi/\Delta t \approx 6$. This value is larger, for example, than twice the frequencies found in Table I. This sampling rate is also three orders of magnitude larger than the growth rates of the unstable modes.

The present approach is also able to provide the associated eigenvectors. In Fig. 18, the absolute values of the u- and v-velocity eigenvectors related to each pair of eigenvalues are plotted. Here, these eigenvectors are mapped to the vertical line of sensors for comparing with the ground truth values. Despite displaying a slight asymmetry for the proposed method, which could likely be improved by adding more sensor points, the comparisons are in good agreement. The eigenmode shapes shown in Fig. 18 can potentially provide insight into the sensor locations chosen (Fig. 16). In particular, the fact that near-wall streamwise velocity sensors are chosen is consistent with the leading eigenmodes having large amplitude near the wall. Conversely, the fact that the selected wall-normal velocity sensor locations are farther from the wall could be related to the fact that this component of the eigenmodes increases in amplitude farther away from the wall.

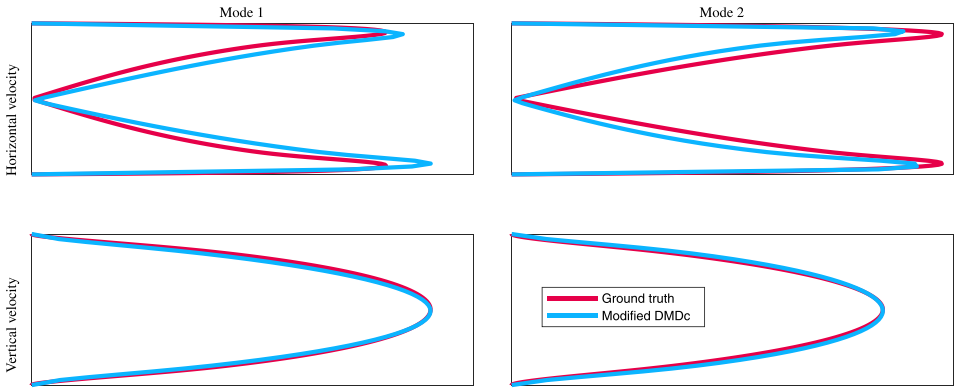

FIG. 18. Unstable modes for the 2D channel flow. Ground truth and estimated results are compared in terms of absolute values for the eigenvectors mapped to the vertical line of sensors.

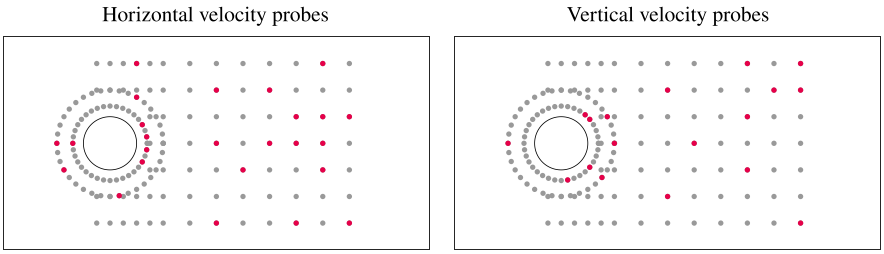

FIG. 19. Demonstration of the application of the sparsity layer for the confined cylinder flow. The total number of measurements is reduced from 306 to 47 (25 and 18 probes for the horizontal and vertical velocity components, respectively). The grey dots are deactivated by the present L1 regularization.

D. Confined cylinder flow

As was the case in Sec. V C, this system is modeled and controlled while also incorporating a sparsity layer in the NNSM to select sensors. In doing so, the NNSM training process for the confined cylinder flow selects a total of 47 sensors from the candidate set of 306 sensors: 25 and 18 points for u and v velocity components, respectively. While the number of sensors is significantly reduced, as presented in Fig. 19, the NNC approach with iterative training is able to successfully stabilize this flow. To provide an overview of the training process, the lift coefficient over the cylinder is presented in Fig. 20 for the first three iterations. In the first iteration, the NNSM and the NNC are trained using only open-loop data. For this iteration, the lift fluctuations are only slightly reduced when flow control is turned on. For the following iterations, the models are trained using both open- and closed-loop data, which improves the results, as more data are available near the equilibrium point, where the system is approximately linear.

While the stabilization is not completely achieved by the end of the second iteration, a reduction in the main oscillations can be observed. By the third iteration, complete stabilization is achieved through control.

Important features observed in the controlled flow are shown in Fig. 21, which shows the convergence history of the drag and lift coefficients, as well as of the control input, during the control stage of the third iteration. The present flow control approach is able to suppress both the

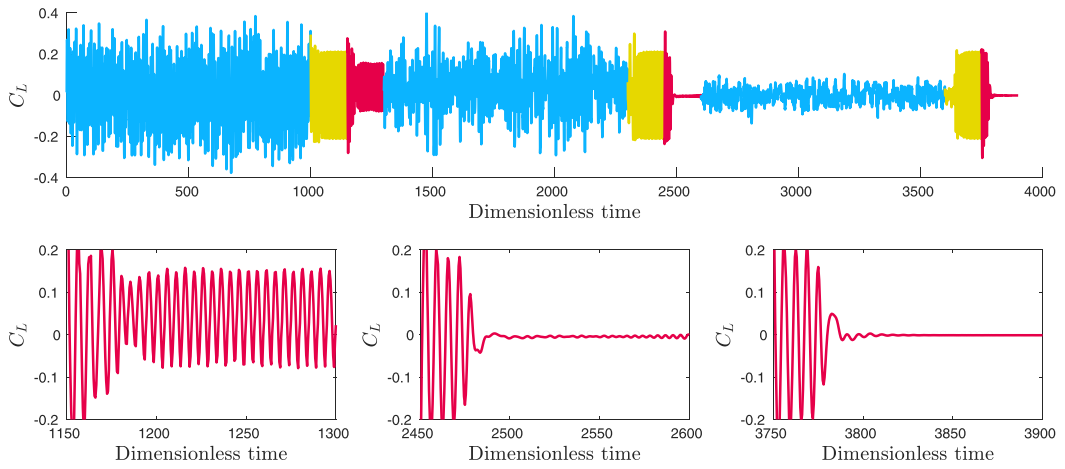

FIG. 20. Evolution of lift coefficient for three iterations of the sweep (blue), release (yellow), and control (red) stages for the confined cylinder flow.

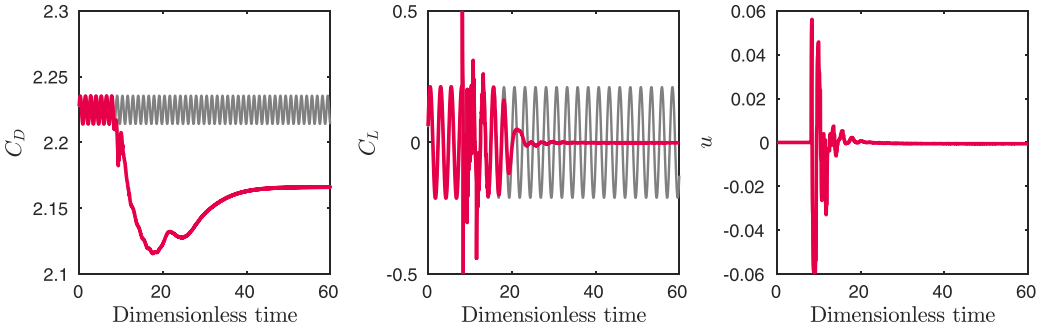

FIG. 21. History of drag and lift coefficients, and control effort for the third iteration in the control mode. Gray lines show uncontrolled solution.

drag and lift coefficient fluctuations. At the same time, the mean drag is also reduced. The complete stabilization also ensures that only a minimum effort is required to keep the system operating with small oscillations and drag losses. Residual efforts are due to imperfect estimation of equilibrium, and may also be required for compensating eventual perturbations. Figure 22 shows u-velocity contours for the uncontrolled and controlled cases. In the former, vortex shedding develops along the wake, inside the plane channel. On the other hand, a steady solution is obtained for the controlled case.

To contextualize these observations, we now discuss prior control results for this confined cylinder configuration. The most direct comparison can be made with the work of Li and Zhang [20], who also attempted to stabilize the flow. Since their chosen flow and goals are very similar to those considered in the present work, they provide a good basis for comparison. The previous authors proposed an approach for the stabilization problem using reinforcement learning to train neural networks capable of minimizing the vortex shedding energy, which is translated to a reward function that takes the velocity fluctuations at selected probe locations. Such calculations could be done by either subtracting the mean flow or the equilibrium flow found through selective frequency damping (SFD) from the measured velocity probes. By using the former approach, convergence was not observed, whereas using the equilibrium states from SFD enabled stabilization. The results, however, required manual positioning of probes through a heuristic approach informed by considering the wavemaker region obtained from the overlap regions of leading direct and adjoint eigenmodes. Furthermore, the prior computation of the equilibrium through SFD is a dedicated step that

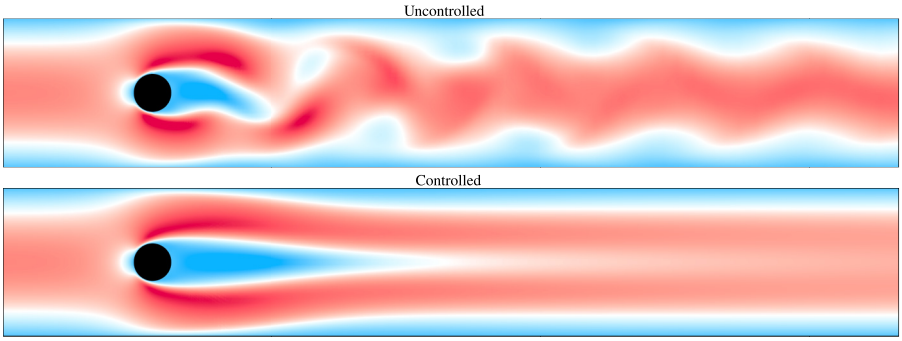

FIG. 22. Comparison of uncontrolled and controlled u-velocity fields (see Movie 2, Supplemental Material [74]). The former is represented by a snapshot taken at the flow natural limit cycle. For the controlled case, the snapshot is captured after stabilization is achieved.

requires additional simulation-intrusive computations. By contrast, our approach enables automatic equilibrium (and eigenmode) computations, which do not require the use of modified numerical solvers. Li and Zhang [20] found that successful control of the confined cylinder flow resulted in a small, nonzero mean control signal (i.e., control jet flow rate), which we also observe here. In our case, this is on the order of $\bar{u} \approx -6 \times 10^{-4}$, which is too small to observe on the right-hand plot of Fig. 21.

One of the clearest advantages of using reinforcement learning, as opposed to the techniques proposed in the present work, is that a model is not needed for the plant to be controlled. By querying the system through a series of episodes, the RL training process refines the control strategy over time, getting closer to the objective in a model-free approach. The absence of an auxiliary model, however, makes it difficult to redesign the control law without the need to run the experiments again with a different reward function, which might be potentially expensive for flow problems. Our approach has the advantage of being able to retrain models by leveraging a previously sampled data set. Similarly, retraining the controller with new hyperparameters by leveraging a previously trained model is more straightforward. Typically, closed-loop control offers a degree of resilience to variations in the plant. This means that models do not need to be absolutely precise; the controller should function effectively with the actual plant, provided the model adequately reflects the core dynamics of the system. On the other hand, for the stabilization problem, an adequate fit near the equilibrium points requires data sampling in these regions, which can be a tricky feat to accomplish. In this work, the iterative approach is proposed to solve this issue, and the production of a rich data set of points at both quasilinear regions and strongly nonlinear ones is shown to be valuable for building the NNSMs. Furthermore, we mention again the ability of inquiring the black-box NNSMs with the goal of obtaining interpretable features such as a linearized model and the equilibrium points.

It is more difficult to make a direct comparison with other prior control studies of this system, primarily due to the different control objectives considered. The current work aims for stabilization, whereas other prior investigations considering this confined cylinder flow [17,72] focused on drag minimization. As we holistically approach the control problem proposing methods that also provide estimates of the equilibrium, the stabilization task is directly enabled through NNC training. For this confined cylinder flow, a drag reduction is seen through stabilization; the lift and drag oscillations are suppressed, and the control input required to keep the controlled system at the unstable equilibrium approaches zero. While we find a drag reduction of 2.6% for the fully stabilized flow, larger drag reduction—5.7% at $\mathrm{Re} = 100$ and 21.6% at $\mathrm{Re} = 200$—are reported by Varela *et al.* [72]. This suggests that the optimal solution for drag reduction is not the flow at its equilibrium. This can be verified, for example, in the strong overshoot seen in C_D (Fig. 21), where the maximum drag reduction is reached before the flow is stabilized. In fact, in iteration 1, the control system produces an average drag reduction of 3.4%, lower than the stabilized flow at iteration 3 of 2.6%.

VI. CONCLUSIONS

In the present work, neural networks are applied to perform as surrogate models of nonlinear dynamical systems, and also as controllers for stabilizing such systems. The proposed approach is first tested with the Lorenz set of ordinary differential equations, as well as with a modified version of the Kuramoto-Sivashinsky partial differential equation. Then, flow control is demonstrated for a periodic 2D channel flow, for which a data-driven linear stability analysis is also conducted and, finally, for a more complex case of a confined cylinder flow. The trained neural networks provide adequate surrogate models that achieve all the present results with a single hidden layer, except for the confined cylinder case, which employs two layers. In the same fashion, the neural networks obtained to control the investigated systems are also single-layer models with only a few neurons. Despite the simple individual neural network architectures, through a recurrent training strategy the present framework is able to obtain effective controllers. Results also support the efficacy of neural networks for learning complex system dynamics, as well as to obtain important flow features, such

as equilibrium points, and leading eigenmodes of the linearized system about these fixed points. Although these data-driven models are trained as black boxes, relevant information can be extracted by using back-propagation.

In this work, several different nonlinear unstable systems are investigated, and their dynamics do not naturally converge to equilibrium points. This behavior hinders data sampling that represents the plant dynamics close to such points, where linear approximations should be able to inform the main eigenvalues and linear modes. As a solution for this problem, we propose the application of NNC with an iterative training of the models. As well as achieving stabilization, this method proves to be an efficient approach for equilibrium computation, sensor placement, and modal stability analysis. As demonstrated by results, the iterative approach is key for bringing the system closer to a progressively better estimation of the equilibrium point. In all the cases presented, the first iteration is unable to provide accurate results, especially for equilibrium computation and modal analysis. Even for the control task, stabilization is not achieved, for example, in the cylinder flow case. The improvement achieved with the iterative process puts the presented NN approach as an efficient and accurate candidate for flow stabilization and data-driven flow analysis, with the possibility of enabling limited sensor allocation.

The first plant studied with the proposed methodology is the Lorenz system. With the chosen numerical parameters, its dynamics results in a chaotic attractor featuring three different equilibrium points. While not explicitly shown, it was found that the initial guess for the equilibrium point has an important role in the iterative process, which means in general that, if initial guesses are too far from an equilibrium of interest, the algorithm may converge to another point, which might not be desirable. In the case presented in the Results section, the initial choice at $x = y = z = 0.1$ converges near the origin after a few iterations. Results from this first test indicate that the iterative training of the NNSM and NNC successfully overcomes the lack of data sampled near the equilibrium. As the control strategy improves and the perturbations are reduced in amplitude, data sets containing more measurements near equilibrium are built, improving the quality of linear approximations and the control design aiming stabilization. Linear regression through DMD-based algorithms is also enabled for a considerably broader range of plants, taking advantage of data produced by controlled nonlinear systems that are brought closer to linear operation through feedback control. The modification to the DMDc method used in this work also helps extend the analyses to systems whose equilibrium points are unknown or poorly estimated, allowing for a model fitting that also enhances estimates of equilibrium bias.

The present methodology is next applied to control a more complex problem, a modified version of the Kuramoto-Sivashinsky equation. This modified equation is proposed in order to force a single equilibrium point instead of a continuous space of possibilities. This test case allows the study of the effectiveness of sensor placement through L1 regularization of the inputs. Again, equilibrium estimation and feedback control obtain satisfactory results, even with a reduction in the number of sensors from 60 to 9. Although this reduction slightly worsens the estimation of equilibrium and, therefore, adds further errors to control convergence, tuning the L1 regularization through the choice of a single hyperparameter leads to a configuration with an intermediate number of sensors at reduced accuracy cost. Even so, the application of Newton's method with linearization of the NNSM through back-propagation is able to provide an equilibrium point estimation that allows for stabilization and, therefore, to obtain the approximately linear data set subsequently used to identify the dominant eigenmodes of the linearized system.

Application of the NNC is then applied to the Navier-Stokes equations. As a first case, stabilization of a 2D streamwise-periodic channel flow is sought, where the capability of conducting a linear stability analysis is also explored for a flow configuration with two unstable modes. The L1 regularization is applied in this case and, from a total of 664 measured signals, training is able to reduce sensing to 181 probes. Stabilization is achieved with eight pairs of actuators in opposition through NNC, which enables sampling of data close to equilibrium, i.e., of signals described by approximately linear dynamics. The data are used for performing a linear regression through a modified DMDc. The unstable eigenvalues and eigenmodes computed by this technique show good

agreement with those obtained directly from the solution of the linear Orr-Sommerfeld operator. The different magnitudes of the growth rate and frequency of the unstable modes result in different levels of accuracy for their estimation, with the growth rates being close to zero. In this sense, the frequencies present lower relative errors than the growth rates due to the large discrete time intervals associated with the latter for the chosen time step. The natural growth rate corresponds to a very small component of the measured signal compared to the effect of the uncontrolled frequency.

Finally, control of a confined cylinder flow is also presented. The setup chosen with actuators in opposition is the same proposed in different studies found in the literature [17,20]. Here, we assess the ability of the neural network model to stabilize a complex system through the iterative training framework. Even for this more complex case, the NNC is built from a single layer containing only eight nodes. Results show that flow control suppresses the lift and drag oscillations from vortex shedding and also reduces the mean drag by bringing the flow to its equilibrium point. Not only is the system successfully controlled, but additionally the low complexity of the NNC suggests that there is room for applying the technique to considerably more complex fluid systems without prohibitive costs. For this confined cylinder configuration, prior work [20] found that RL control performance could be improved through equilibrium computation and linear stability analysis in the vicinity of this equilibrium. Here, we achieve similarly successful control results without needing to perform these auxiliary computations explicitly, but where the results of such computations are available as a by-product of our modeling and control methodology.

In each of the examples featuring partial differential equations across a spatial domain, we have utilized a sparsity layer in the NNSM to reduce the number of sensor measurements (and thus model inputs). While the resulting sensor locations are chosen to optimize a given cost function, they are not necessarily unique solutions, in the sense that alternative sensor locations may be obtained for different realizations (with different random input signals and randomly initialized NN weights) of the training procedures. This is clear from observing that the identified sensors do not follow the same symmetry properties of the flow configurations. That being said, the chosen sensor locations do reveal some aspects of the underlying physics. For the modified KS equation, the sensors are distributed throughout the domain, in accordance to the translation invariance of the problem (aside from the actuator locations). For the 2D channel flow, the sensor locations indicate the importance of near-wall streamwise velocity measurements, with a sparser set of measurements away from the wall. The asymmetry in the chosen locations of the wall-normal velocity probe locations is perhaps due to the fact that the instabilities in this system consist of mode shapes where the vertical velocity component extends across the whole domain, so taking measurements at both the upper and lower walls may be unnecessary. In the confined cylinder flow, we find that sensors both close to the cylinder and downstream in the wake are chosen, indicating the importance of measurements in both regions for suppressing the natural vortex shedding behavior. While beyond the scope of the present work, it could be interesting to compare the identified sensor locations with those obtained from alternative methods [54–57].

The modeling and control framework utilized here is similar in principle to that used in classical linear control theory, with separate models for the plant and controller connected in feedback. While nonlinear control is substantially more complex, here we show that relatively simple neural networks can be used to develop effective nonlinear models for both the plant and controller, when trained with a method that promotes the generation of large quantities of near-equilibrium data. Future work could also utilize linearization of the plant and/or controller models for linear control design, with better authority over the plant behavior through well-studied control design techniques that can ensure robustness, optimality, or specific pole placement.

While it was demonstrated that the proposed methodology could be used to identify modal linear amplification mechanisms, future work could extend these methods to study nonmodal amplification, such as transient growth and resolvent analysis. Data-driven implementations of such analyses have been implemented previously [73], though for non-normal systems it may be more difficult to obtain accurate results without adjoint operators and/or data, which may require further modifications to the methodology presented here.

We finally emphasize that the methods and results are obtained from nonintrusive methods in the sense that only the nonlinear system (with appropriate control inputs) has been used. This means that the methodology could in principle be applied (for both control and flow physics analysis) in an experimental setting. This, as well as the application to systems presenting a higher degree of complexity, remains the subject of future work.

ACKNOWLEDGMENTS

The authors acknowledge the financial support received from Fundação de Amparo à Pesquisa do Estado de São Paulo, FAPESP, under Grants No. 2013/08293-7 and No. 2021/06448-0. The first author is supported by FAPESP Ph.D. scholarships No. 2019/19179-7 and No. 2022/00469-8, which are also acknowledged. S.T.M.D. acknowledges support from NSF Award No. 2238770. The computational resources used in this work were provided by CENAPAD-SP (Project No. 551), and by LNCC via the SDumont cluster (Project SimTurb).

APPENDIX A: HYPERPARAMETERS

In this Appendix, the chosen values for the hyperparameters in each case are presented. They are organized in Table II. Regarding the iterative training, the "number of iterations" corresponds to how many times the proposed training steps are conducted. It is expected that at each iteration the perturbed plant controlled by NNC tends to produce more data near equilibrium. The "number of steps in sweep," "number of steps in release," and "number of steps in control" correspond to the number of measurements sampled in each mode at each iteration.

The open-loop control signal is parametrized by a "control input saturation value," which defines the possible uniform interval of random values it can assume. The staircase signal has a step width given by the "number of time steps for each stair step (open-loop signal)". Also, the "maximum amplitude decay at each iteration" parameter makes sure the open-loop signal is reduced in amplitude at each training iteration.

From each mode (sweep, release, and control), data are sampled for training, but the data-set size is limited to a "maximum number of points sampled for training NNSM." This ensures that the training cost does not increase beyond a certain point. An "NNSM learning rate" is defined for the ADAM optimization algorithm, and the model is updated through a "number of epochs at each training iteration for the NNSM." The "number of neurons at NNSM hidden layers" hyperparameter defines the number of layers, which is given by the number of elements in the array of numbers, and the number of neurons in each layer, which is the value at each position. Note that, except for the cylinder case in Table II, all NNSM models are trained with a single layer. For assessment of possible overfitting, an "NNSM training set ratio" as a fraction of sampled data is also chosen, which allows for the comparison of losses between a training set and a testing set of samples. The "hidden layers L2 regularization" parameter is the r_2 variable defined in this work. "Use sparsity layer" is a Boolean that determines if sparse sensor placement is used. If so, we can set the "sparsity layer L1 regularization" parameter (r_1) and the "sparsity layer truncation tolerance" that determines the absolute value below which the input layer weights are truncated to zero. The input layer is organized according to the "number of control inputs" and the "number of states" measured.

Similarly a "maximum number of points sampled for training NNC" is also defined to avoid excessive complexity, as well as the "NNC learning rate" for ADAM and the "number of epochs at each training step for NNC." The "training finite horizon length" in number of steps (n_h) is also presented. For the NNC, an "NNC training set ratio" as a fraction of sampled data is also chosen. The "number of neurons at NNC hidden layers" is always the same in all cases, which consists of a single layer containing eight neurons. "NNC control saturation" stands for the limits of control effort that NNC can provide. It is implemented as an output layer consisting of weighted sigmoid functions. The "control input weight in loss function" corresponds to $\mathbf{w_u}$.

TABLE II. Hyperparameters for each case studied.

Description	Lorenz	KS, full sensing	KS, sparse sensing	Channel flow	Cylinder flow
Number of training iterations	25	10	10	19	3
Number of steps in sweep	1000	2000	2000	1800	2000
Number of steps in release	80	400	400	300	300
Number of steps in control	80	100	100	900	300
Control input saturation value ($\pm$)	$4.5 \times 10^{+1}$	3.0×10^{-1}	3.0×10^{-1}	1.6×10^{-1}	6.0×10^{-2}
Number of time steps for each stair step (open-loop signal)	20	10	10	10	10
Stair signal maximum amplitude	$4.5 \times 10^{+1}$	3.0×10^{-1}	3.0×10^{-1}	1.6×10^{-1}	6.0×10^{-2}
Maximum amplitude decay at each iteration	6.5×10^{-1}	8.0×10^{-1}	8.0×10^{-1}	8.0×10^{-1}	9.0×10^{-1}
Maximum number of points sampled for training NNSM	4000	4000	4000	4000	4000
NNSM learning rate	6.0×10^{-3}	6.0×10^{-3}	6.0×10^{-3}	5.0×10^{-3}	1.0×10^{-2}
Number of epochs at each training iteration for NNSM	1000	1000	1000	2000	2000
Number of neurons at NNSM hidden layers	[18]	[18]	[18]	[80]	[100, 80]
NNSM training set ratio (fraction of sampled data)	7.0×10^{-1}	7.0×10^{-1}	7.0×10^{-1}	7.0×10^{-1}	7.0×10^{-1}
Hidden layers L2 regularization	1.0×10^{-5}	1.0×10^{-5}	1.0×10^{-5}	1.0×10^{-5}	1.0×10^{-5}
Use sparsity layer	False	False	True	True	True
Sparsity layer L1 regularization			1.0×10^{-4}	1.0×10^{-5}	1.0×10^{-4}
Sparsity layer truncation tolerance			3.0×10^{-3}	3.0×10^{-3}	3.0×10^{-3}
Number of control inputs	1	3	3	8	1
Number of states	3	60	60	664	306
Maximum number of points sampled for training NNC	1000	1000	1000	1000	1000
NNC learning rate	4.0×10^{-3}	1.0×10^{-2}	1.0×10^{-2}	1.0×10^{-2}	1.0×10^{-2}
Number of epochs at each training step for NNC	100	200	200	300	300
Training finite horizon length	10	50	50	140	140
NNC training set ratio (fraction of sampled data)	9.0×10^{-1}	9.0×10^{-1}	9.0×10^{-1}	9.0×10^{-1}	9.0×10^{-1}
Number of neurons at NNC hidden layers	[8]	[8]	[8]	[8]	[8]
NNC control saturation	$4.5 \times 10^{+1}$	3.0×10^{-1}	3.0×10^{-1}	1.6×10^{-1}	6.0×10^{-2}
Control input weight in loss function	5.0×10^{-3}	5.0×10^{-3}	5.0×10^{-3}	1.0×10^{-1}	1.0×10^{-1}
Initial guess for equilibrium point	$[1.0 \times 10^{-1}$, 1.0×10^{-1}, $1.0 \times 10^{-1}]$	$[1.0 \times 10^{-1}, \ldots, 1.0 \times 10^{-1}]$	$[1.0 \times 10^{-1}, \ldots, 1.0 \times 10^{-1}]$	1.0 for x-velocity, 0.0 for y-velocity	1.0 for x-velocity, 0.0 for y-velocity
Number of steps for Newton method (equilibrium estimation)	10	10	10	10	10

Finally, regarding equilibrium computation, the "initial guess for equilibrium point" is presented for each case, corresponding to the first guess used in the first training iteration. At each of these iterations, the estimation of equilibrium is updated along a "number of steps for the Newton method."

APPENDIX B: IMPLEMENTATION OVERVIEW AND GENERAL GUIDELINES

The NNSMs have demonstrated the capability to model the present nonlinear systems, while the NNC proved to be capable of stabilizing such systems. However, the convergence and functionality of the control system depends on the selection of hyperparameters, whose choices can make a considerable difference in the performance. In this section, we provide general guidelines for setting up the training process. Although the quality of the models and control systems are affected by the hyperparameters, it is important to mention that different values of such parameters could present equally satisfactory results in terms of control performance. Hence, the guidelines provided here are not necessarily the only possible path. Also, the comparison between two different hyperparameter setups will only make sense within a same study case.

As mentioned in the main text, the first step consists of gathering data through open-loop sweeps. These data will be used in the first iteration to train the NNSM. It is recommended to select a number of steps for the sweep step that is large enough to properly sweep the state space. This roughly means that systems with more complex dynamics should require a larger number of time steps in the data set. One can conclude, for example, that the number of data points used in the Lorenz case could be smaller by comparing with the other cases (since it has only three states). In fact, we sample way more points than what is actually needed, which is not a problem since simulating the Lorenz system is computationally inexpensive. For the control step, we recommend utilizing a window that could encompass the control convergence time. Increasing the window beyond this could lead to the sampling of redundant data at the equilibrium point. The release mode is mainly used for visualization and testing, so it is not the most important parameter of concern. However, it is important to consider that very large release windows would make the fit more prone to prioritize the dynamics of the uncontrolled plant in the competition to reduce the loss function. The open-loop signal is set by trial and error, by analyzing the produced signals and trajectories to see if they are drifting away from their natural responses. The same amplitude is set for saturating the controller, although some extrapolation could also be tried. The number of time steps for each staircase step in the control input is set to values that are not too large, avoiding the trajectories to stick to fixed orbits or forced equilibrium points.

Now, the complexity of the networks, which include the number of layers and the number of nodes in each layer, can be chosen. For the NNSM, in general, our recommendation is to set the smaller values that would make the loss function converge satisfactorily. This helps avoid both excessive computation and overfitting. For the cylinder case, it was noticed that a single layer did not lead to a proper reduction of the loss function during training. Hence, the model complexity was increased by adding one extra hidden layer and more nodes until the loss function converged to satisfactory values. For the NNC, the procedure is similar: starting from a very simple controller, increase the complexity until the NNC loss function converges properly. For both the NNSM and the NNC, the balance between the number of training epochs and learning rate should be such that the number of epochs is the smallest possible by increasing the learning rate without making the training process unstable. Very high learning rates can make the loss function oscillate during training, hindering convergence.

The open-loop signal decay is set together with the global number of training iterations. If the overall process becomes too (computationally) expensive, the total number of iterations could be reduced and the decay intensified in an attempt to obtain data near equilibrium earlier. The maximum number of points used to train both networks is chosen only based on computational capabilities. Fixed values for the L2 regularization factor are recommended to avoid indefinite growth of the

hidden layer weights. The L1 factor, on the other hand, can be adjusted for balancing the NNSM accuracy and the number of sensors. Since the contribution of the L2 regularization factor to the loss function represents a quadratic growth with an increase of the weights, modest changes to the L1 factor do not require a change in the L2 regularization.

The fixed horizon length for the NNC training is adjusted according to the control convergence time. For example, if a large control input weight is chosen, a slower convergence might occur, which could require an increase in the horizon length. If the latter is too small, the controller might not be well trained for regions near equilibrium. One important thing to mention is that, since the NNC training process relies on a recurrent setup, increasing horizon lengths can quickly make the algorithm computationally prohibitive. Finally, regarding the Newton method, we tested different initial guesses for the Lorenz case. As expected, this leads to convergence to different equilibrium points, since the present Lorenz setup depicts three different equilibrium points. For the flow configurations, we simply used the reference values of velocity as initial guess, i.e., 1 for horizontal and 0 for vertical velocity components. This choice was shown to be reasonable for the tested cases. Another option would be to use estimates based on the mean flow.

APPENDIX C: REMARKS ON ROBUSTNESS

In this Appendix, we present a brief discussion about the robustness of the closed NNC loop. We choose the Lorenz system due to the possibility of running computationally inexpensive tests. We show the results by employing the controller trained at iteration 4—discussed in Sec. V A—for different types of undesirable phenomena that could make the control not work properly. In all cases, the controller is employed as designed, but in some cases, we also propose modifications to the methodology that could further increase the robustness of the control system. Selected from a series of trial-and-error tests, we present the threshold conditions beyond which the controller saturates or fails to keep the Lorenz system in the regions near equilibrium.

First, in Figs. 23(a) and 23(b) we show the results of modifying the control input from $u = K(x, y, z)$ to $u = K(x, y, z) + \delta_0 \sin(\omega_d k)$. We choose a low frequency ω_d and increase δ_0 to more than one-third of the saturation value (45 in absolute value) chosen for u (see Table II). With $\delta_0 = 17$, the states move away from the region near $[0, 0, 0]$. This happens when u saturates and, therefore, increasing the saturation levels should enhance the ability to keep the states closer to equilibrium. Furthermore, allowing the controller to output higher efforts also tends to increase the robustness to disturbances.

The effects of quantization in the digitization of the signals are also tested. In Figs. 23(c) and 23(d), it is shown that the states move away from near the equilibrium when a resolution of four levels (equivalent to two-bit quantization) is used. With eight levels (three-bit quantization), the states are kept in the region close to equilibrium. As expected, a low resolution in quantization worsens the performance of control and this needs to be taken into account in real-world applications.

In this work, a standard fourth-order Runge-Kutta scheme is used to simulate the Lorenz system. We verified that the maximum time step for the simulation to stay stable is $\Delta t = 0.1$. In the simulations shown in the present work, we use $\Delta t = 0.05$. Exclusively for the cases shown in Figs. 23(e) and 23(f) we use a time step of 0.03 for the Runge-Kutta method and sample the states every six steps ($\Delta t = 0.18$) or every seven steps ($\Delta t = 0.21$). Since the control law $u_k = K(x_k, y_k, z_k)$ is a static nonlinear function, i.e., it does not depend on past measurements, it is generalizable for other values of Δt beyond that used for training. However, when Δt is too large, the effects of aliasing make it impossible for the control loop to keep working properly. This is a limitation of digital control, and a minimum sample rate is required independently of the control technique itself.

Another test that is presented encompasses the addition of noise to the measured signals. Independent Gaussian noise is applied to each of the three states, with a mean value of zero and standard deviations of $\sigma_s = 0.5$ and $\sigma_s = 0.6$, as shown in Figs. 23(g) and 23(h). In each case, the same σ_s is used for x, y, and z. The reason that makes the states move away from near the equilibrium

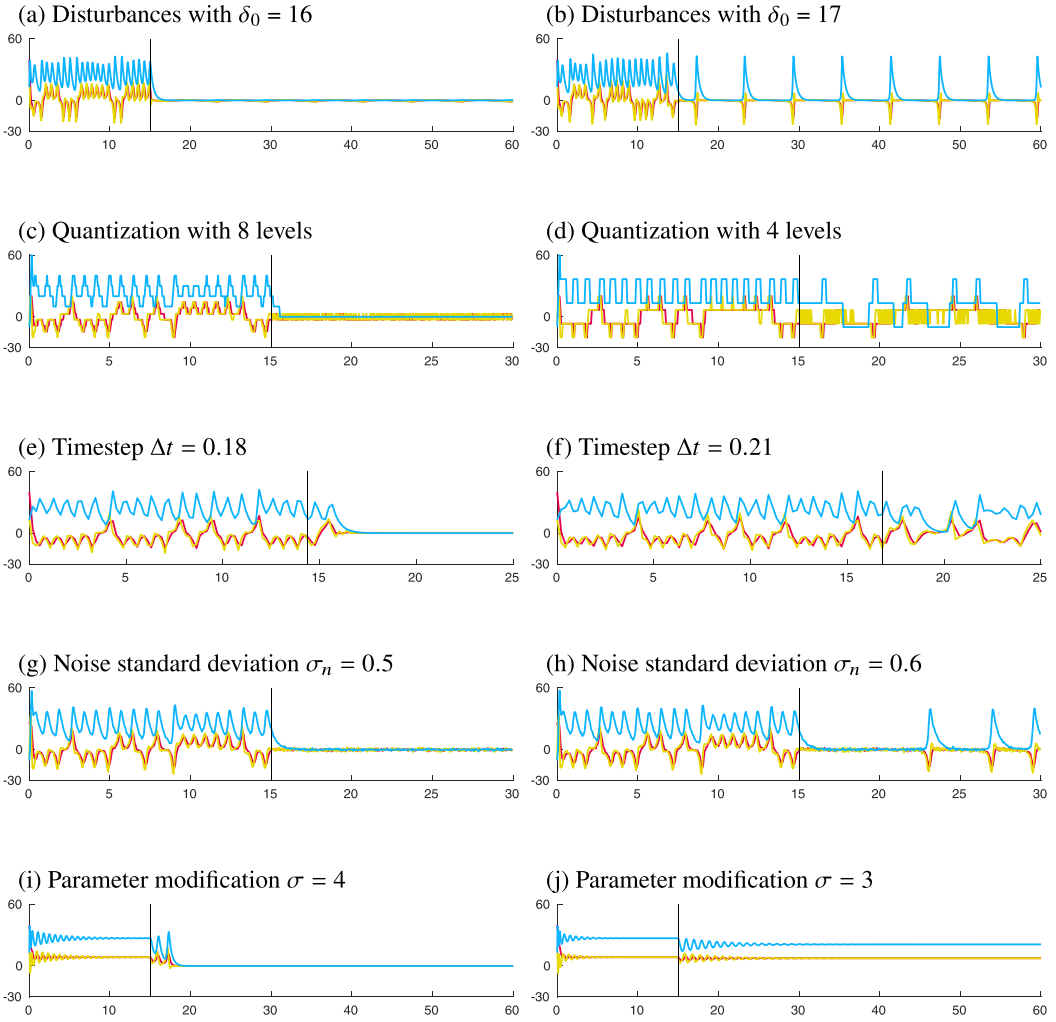

FIG. 23. Controlled Lorenz system subject to different undesirable phenomena. The vertical black line depicts the moment when the control is turned on. The colored curves represent the measured states seen by the controller.

point is the high amplification of noise produced by the controller that makes the control input saturate, rendering the loop unable to hold the states. One possible solution to this issue within the scope of NNC is the implementation of a state estimator. Since the NNSM can be linearized in different regions of the input space, the implementation of an extended Kalman filter should be a candidate approach to solve this issue, allowing for the controller to receive cleaner signals and thus avoiding the amplification of noise.

Finally, we check the capability of the NNC to control a modified version of the Lorenz equations. We keep the β and ρ values from the equations the same as the originals and modify the parameter σ from $\sigma = 10$ to $\sigma = 4$ and $\sigma = 3$, as shown in Figs. 23(i) and 23(j). The choice of reducing σ instead of increasing comes from the fact that making it larger tends to drive the simulation numerically unstable. When the modification reaches $\sigma = 3$, the controller loses its ability to maintain the system states close to $[0, 0, 0]$. In this scenario, the controller is permanently locked at saturation levels. One possible solution to improve the control functionality when considerable plant variations are

expected is to modify the discrete formulation from

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k)$$

to

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k, \mathbf{h}),$$

where $\mathbf{h}$ is a parameter space that could contain, for example, the Reynolds number of a flow. This approach would clearly suffer from requiring that $\mathbf{h}$ is somehow measured or estimated, although there can be some margin for inaccuracies, as it is expected in closed-loop control systems and shown in the example tested. In this scenario, the NNC could be trained as a control law $\mathbf{u} = \mathbf{K}(\mathbf{x_r}, \mathbf{h})$.

As a final comment regarding the proposed tests, we mention that we also tried to apply measurement noise and larger values of Δt to reconduct the iterative training process from scratch. Interestingly, the training provided controllers with similar performance as the original case, starting to present issues only when the limits $\sigma_s \approx 0.6$ or $\Delta t \approx 0.21$ were exceeded—values similar to those shown in Fig. 23. This means that adding noise to the measurements does not make it impossible to train adequate models, as long as overfitting is properly avoided. It also indicates that we can build discrete models (NNSMs) for the Lorenz system that can work at values of Δt that are considerably higher than the ones allowed by the Runge-Kutta scheme.

[1] M. Gad-el-Hak, Modern developments in flow control, Appl. Mech. Rev. **49**, 365 (1996).
[2] C. Lee, J. Kim, D. Babcock, and R. Goodman, Application of neural networks to turbulence control for drag reduction, Phys. Fluids **9**, 1740 (1997).
[3] J. Kutz, Deep learning in fluid dynamics, J. Fluid Mech. **814**, 1 (2017).
[4] M. P. Brenner, J. D. Eldredge, and J. B. Freund, Perspective on machine learning for advancing fluid mechanics, Phys. Rev. Fluids **4**, 100501 (2019).
[5] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. **52**, 477 (2020).
[6] J. Ling, A. Kurzawski, and J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. **807**, 155 (2016).
[7] C. Xie, X. Xiong, and J. Wang, Artificial neural network approach for turbulence models: A local framework, Phys. Rev. Fluids **6**, 084612 (2021).
[8] M. Milano and P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. **182**, 1 (2002).
[9] K. Fukami, K. Fukagata, and K. Taira, Super-resolution reconstruction of turbulent flows with machine learning, J. Fluid Mech. **870**, 106 (2019).
[10] H. F. S. Lui and W. R. Wolf, Construction of reduced-order models for fluid flows using deep feedforward neural networks, J. Fluid Mech. **872**, 963 (2019).
[11] R. F. Miotto and W. R. Wolf, Flow imaging as an alternative to non-intrusive measurements and surrogate models through vision transformers and convolutional neural networks, Phys. Fluids **35**, 045143 (2023).
[12] J. Page, M. P. Brenner, and R. R. Kerswell, Revealing the state space of turbulence using machine learning, Phys. Rev. Fluids **6**, 034402 (2021).
[13] Z. Deng, Y. Chen, Y. Liu, and K. C. Kim, Time-resolved turbulent velocity field reconstruction using a long short-term memory (LSTM)-based artificial intelligence framework, Phys. Fluids **31**, 075108 (2019).
[14] K. H. Manohar, C. Morton, and P. Ziadé, Sparse sensor-based cylinder flow estimation using artificial neural networks, Phys. Rev. Fluids **7**, 024707 (2022).
[15] J. Morton, A. Jameson, M. J. Kochenderfer, and F. D. Witherden, Deep dynamical modeling and control of unsteady fluid flows, Adv. Neural Info. Proc. Sys. **31** (2018).

[16] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz, Deep model predictive flow control with limited sensor data and online learning, Theor. Comput. Fluid Dyn. **34**, 577 (2020).
[17] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi, Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control, J. Fluid Mech. **865**, 281 (2019).
[18] F. Ren, J. Rabault, and H. Tang, Applying deep reinforcement learning to active flow control in weakly turbulent conditions, Phys. Fluids **33**, 037121 (2021).
[19] R. Castellanos, G. Cornejo Maceda, I. De La Fuente, B. Noack, A. Ianiro, and S. Discetti, Machine-learning flow control with few sensor feedback and measurement noise, Phys. Fluids **34**, 047118 (2022).
[20] J. Li and M. Zhang, Reinforcement-learning-based control of confined cylinder wakes with stability analyses, J. Fluid Mech. **932**, A44 (2022).
[21] D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, Reinforcement learning for bluff body active flow control in experiments and simulations, Proc. Natl. Acad. Sci. USA **117**, 26091 (2020).
[22] L. Guastoni, J. Rabault, P. Schlatter, H. Azizpour, and R. Vinuesa, Deep reinforcement learning for turbulent drag reduction in channel flows, Eur. Phys. J. E **46**, 27 (2023).
[23] T. Sonoda, Z. Liu, T. Itoh, and Y. Hasegawa, Reinforcement learning of control strategies for reducing skin friction drag in a fully developed turbulent channel flow, J. Fluid Mech. **960**, A30 (2023).
[24] T. Lee, J. Kim, and C. Lee, Turbulence control for drag reduction through deep reinforcement learning, Phys. Rev. Fluids **8**, 024604 (2023).
[25] O. A. Mahfoze, A. Moody, A. Wynn, R. D. Whalley, and S. Laizet, Reducing the skin-friction drag of a turbulent boundary-layer flow with low-amplitude wall-normal blowing within a Bayesian optimization framework, Phys. Rev. Fluids **4**, 094601 (2019).
[26] G. Novati, L. Mahadevan, and P. Koumoutsakos, Controlled gliding and perching through deep-reinforcement-learning, Phys. Rev. Fluids **4**, 093902 (2019).
[27] A. Jing, Z. Tang, J. Gao, and G. Pan, An improved ddpg reinforcement learning control of underwater gliders for energy optimization, in *2020 3rd International Conference on Unmanned Systems (ICUS)* (IEEE, New York, 2020), pp. 621–626.
[28] H. Ghraieb, J. Viquerat, A. Larcher, P. Meliga, and E. Hachem, Single-step deep reinforcement learning for open-loop control of laminar and turbulent flows, Phys. Rev. Fluids **6**, 053902 (2021).
[29] C. Vignon, J. Rabault, and R. Vinuesa, Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions, Phys. Fluids **35**, 031301 (2023).
[30] J. Viquerat, P. Meliga, A. Larcher, and E. Hachem, A review on deep reinforcement learning for fluid mechanics: An update, Phys. Fluids **34**, 111301 (2022).
[31] S. L. Brunton, J. L. Proctor, and N. J. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA **113**, 3932 (2016).
[32] E. Kaiser, J. N. Kutz, and S. L. Brunton, Sparse identification of nonlinear dynamics for model predictive control in the low-data limit, Proc. R. Soc. A **474**, 20180335 (2018).
[33] F. Zigunov, P. Sellappan, and F. Alvi, A bluff body flow control experiment with distributed actuation and genetic algorithm-based optimization, Exp. Fluids **63**, 23 (2022).
[34] F. Zigunov, M. Song, P. Sellappan, and F. S. Alvi, Multiaxis shock vectoring control of overexpanded supersonic jet using a genetic algorithm, J. Propul. Power **39**, 249 (2023).
[35] C. Raibaudo, P. Zhong, B. R. Noack, and R. J. Martinuzzi, Machine learning strategies applied to the control of a fluidic pinball, Phys. Fluids **32**, 015108(2020).
[36] D. Sipp and P. J. Schmid, Linear closed-loop control of fluid instabilities and noise-induced perturbations: A review of approaches and tools, Appl. Mech. Rev. **68**, 020801 (2016).
[37] C. W. Rowley and D. R. Williams, Dynamics and control of high-Reynolds-number flow over open cavities, Annu. Rev. Fluid Mech. **38**, 251 (2006).
[38] A. Barbagallo, D. Sipp, and P. J. Schmid, Closed-loop control of an open cavity flow using reduced-order models, J. Fluid Mech. **641**, 1 (2009).
[39] S. J. Illingworth, Model-based control of vortex shedding at low Reynolds numbers, Theor. Comput. Fluid Dyn. **30**, 429 (2016).

[40] T. L. Flinois and A. S. Morgans, Feedback control of unstable flows: A direct modelling approach using the eigensystem realisation algorithm, J. Fluid Mech. **793**, 41 (2016).
[41] S. L. Brunton, S. T. Dawson, and C. W. Rowley, State-space model identification and feedback control of unsteady aerodynamic forces, J. Fluids Struct. **50**, 253 (2014).
[42] B. Herrmann, S. L. Brunton, J. E. Pohl, and R. Semaan, Gust mitigation through closed-loop control. II. Feedforward and feedback control, Phys. Rev. Fluids **7**, 024706 (2022).
[43] G. Sedky, A. Gementzopoulos, F. D. Lagor, and A. R. Jones, Experimental mitigation of large-amplitude transverse gusts via closed-loop pitch control, Phys. Rev. Fluids **8**, 064701 (2023).
[44] S. Shi, S. Xu, J. Gu, and H. Min, Global high-order sliding mode controller design subject to mismatched terms: Application to buck converter, IEEE Trans. Circuits Syst. I **66**, 4840 (2019).
[45] S. Baek, J. Baek, and S. Han, An adaptive sliding mode control with effective switching gain tuning near the sliding surface, IEEE Access **7**, 15563 (2019).
[46] F. Alyoussef and I. Kaya, A review on nonlinear control approaches: Sliding mode control back-stepping control and feedback linearization control, in *International Engineering and Natural Sciences Conference (IENSC 2019)* (Science & Education Group (INESEG), Diyarbakır, Turkey, 2019), Vol. 2019, pp. 608–619.
[47] A. Zulu and S. John, A review of control algorithms for autonomous quadrotors, Open J. Appl. Sci. **4**, 547 (2014).
[48] T. C. Déda and W. R. Wolf, Extremum seeking control applied to airfoil trailing-edge noise suppression, AIAA J. **60**, 823 (2022).
[49] L. N. Egidio, G. S. Deaecto, J. P. Hespanha, and J. C. Geromel, Trajectory tracking for a class of switched nonlinear systems: Application to PMSM, Nonlinear Anal. Hybrid Syst. **44**, 101164 (2022).
[50] H. Arbabi, M. Korda, and I. Mezić, A data-driven Koopman model predictive control framework for nonlinear partial differential equations, in *2018 IEEE Conference on Decision and Control (CDC)* (IEEE, New York, 2018), pp. 6409–6414.
[51] T. Déda, W. R. Wolf, and S. T. M. Dawson, Backpropagation of neural network dynamical models applied to flow control, Theor. Comput. Fluid Dyn. **37**, 35 (2023).
[52] G. Y. C. Maceda, Y. Li, F. Lusseyran, M. Morzyński, and B. R. Noack, Stabilization of the fluidic pinball with gradient-enriched machine learning control, J. Fluid Mech. **917**, A42 (2021).
[53] K. Willcox, Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition, Comput. Fluids **35**, 208 (2006).
[54] K. Manohar, B. W. Brunton, J. N. Kutz, and S. L. Brunton, Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns, IEEE Control Syst. Mag. **38**, 63 (2018).
[55] P. Sashittal and D. J. Bodony, Data-driven sensor placement for fluid flows, Theor. Comput. Fluid Dyn. **35**, 709 (2021).
[56] J. Graff, A. Medina, and F. D. Lagor, Information-based sensor placement for data-driven estimation of unsteady flows, AIAAJ **61** (2023).
[57] J. Williams, O. Zahn, and J. N. Kutz, Data-driven sensor placement with shallow decoder networks, arXiv:2202.05330.
[58] R. Paris, S. Beneddine, and J. Dandois, Robust flow control and optimal sensor placement using deep reinforcement learning, J. Fluid Mech. **913**, A25 (2021).
[59] K. K. Chen and C. W. Rowley, H2 optimal actuator and sensor placement in the linearised complex Ginzburg–Landau system, J. Fluid Mech. **681**, 241 (2011).
[60] B. Jin, S. J. Illingworth, and R. D. Sandberg, Optimal sensor and actuator placement for feedback control of vortex shedding, J. Fluid Mech. **932**, A2 (2022).
[61] G. A. Freire, A. V. Cavalieri, F. J. Silvestre, A. Hanifi, and D. S. Henningson, Actuator and sensor placement for closed-loop control of convective instabilities, Theor. Comput. Fluid Dyn. **34**, 619 (2020).
[62] M. R. Jovanović, P. J. Schmid, and J. W. Nichols, Sparsity-promoting dynamic mode decomposition, Phys. Fluids **26**, 024103 (2014).
[63] C. S. Skene, C.-A. Yeh, P. J. Schmid, and K. Taira, Sparsifying the resolvent forcing mode via gradient-based optimisation, J. Fluid Mech. **944**, A52 (2022).

[64] B. Lopez-Doriga, E. Ballouz, H. J. Bae, and S. T. M. Dawson, A sparsity-promoting resolvent analysis for the identification of spatiotemporally-localized amplification mechanisms, in *AIAA SCITECH 2023 Forum* (AIAA, Reston, VA, 2023), p. 0677.

[65] K. Gurney, *An Introduction to Neural Networks* (CRC Press, Boca Raton, FL, 2018).

[66] O. Zahn, J. Bustamante, Jr., C. Switzer, T. L. Daniel, and J. N. Kutz, Pruning deep neural networks generates a sparse, bio-inspired nonlinear controller for insect flight, PLoS Comput. Biol. **18**, e1010512 (2022).

[67] J. L. Proctor, S. L. Brunton, and J. N. Kutz, Dynamic mode decomposition with control, SIAM J. Appl. Dyn. Syst. **15**, 142 (2016).

[68] G. I. Sivashinsky, Nonlinear analysis of hydrodynamic instability in laminar flames – I. Derivation of basic equations, Acta Astronaut. **4**, 1177 (1977).

[69] Y. Kuramoto, Diffusion-induced chaos in reaction systems, Prog. Theor. Phys. **64**, 346 (1978).

[70] N. Fabbiane, O. Semeraro, S. Bagheri, and D. S. Henningson, Adaptive and model-based control theory applied to convectively unstable flows, Appl. Mech. Rev. **66**, 060801(2014).

[71] J. A. Weideman and S. C. Reddy, A Matlab differentiation matrix suite, ACM Trans. Math. Software **26**, 465 (2000).

[72] P. Varela, P. Suárez, F. Alcántara-Ávila, A. Miró, J. Rabault, B. Font, L. M. García-Cuevas, O. Lehmkuhl, and R. Vinuesa, Deep reinforcement learning for flow control exploits different physics for increasing Reynolds number regimes, Actuators **11**, 359 (2022).

[73] B. Herrmann, P. J. Baddoo, R. Semaan, S. L. Brunton, and B. J. McKeon, Data-driven resolvent analysis, J. Fluid Mech. **918**, A10 (2021).

[74] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevFluids.9.063904 for animated visualizations of the flows stabilized by neural network controllers.