

How more data can hurt: Instability and regularization in next-generation reservoir computing

Cite as: Chaos 35, 073102 (2025); doi: 10.1063/5.0262977

Submitted: 3 February 2025 · Accepted: 2 June 2025 ·

Published Online: 1 July 2025



View Online



Export Citation



CrossMark

Yuanzhao Zhang,^{1,a)} Edmilson Roque dos Santos,^{2,3,b)} Huixin Zhang,^{4,5,c)} and Sean P. Cornelius^{4,d)}

AFFILIATIONS

¹Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, New Mexico 87501, USA

²Department of Electrical and Computer Engineering, Clarkson University, Potsdam, New York 13699, USA

³Clarkson Center for Complex Systems Science, Clarkson University, Potsdam, New York 13699, USA

⁴Department of Physics, Toronto Metropolitan University, Toronto, Ontario M5B 2K3, Canada

⁵Department of Automation, Shanghai University, Shanghai 200444, China

Note: This paper is part of the Special Topic on Nonlinear Dynamics of Reservoir Computing: Theory, Application and Realization.

^{a)}Author to whom correspondence should be addressed: y Zhang@santafe.edu

^{b)}Electronic mail: edmilson.roque.usp@gmail.com

^{c)}Electronic mail: huixinzhang@shu.edu.cn

^{d)}Electronic mail: cornelius@torontomu.ca

ABSTRACT

It has been found recently that more data can, counter-intuitively, hurt the performance of deep neural networks. Here, we show that a more extreme version of the phenomenon occurs in data-driven models of dynamical systems. To elucidate the underlying mechanism, we focus on next-generation reservoir computing (NGRC)—a popular framework for learning dynamics from data. We find that, despite learning a better representation of the flow map with more training data, NGRC can adopt an ill-conditioned “integrator” and lose stability. We link this data-induced instability to the auxiliary dimensions created by the delayed states in NGRC. Based on these findings, we propose simple strategies to mitigate the instability, either by increasing regularization strength in tandem with data size, or by carefully introducing noise during training. Our results highlight the importance of proper regularization in data-driven modeling of dynamical systems.

Published under an exclusive license by AIP Publishing. <https://doi.org/10.1063/5.0262977>

It is not well understood how the amount and quality of data might affect the stability of machine-learning models in the context of learning unknown dynamics. This is different from the typical tasks considered by the machine-learning community such as regression or classification. Indeed, for forecasting tasks, the model often needs to be run in an auto-regressive fashion, which introduces the problem of stability (i.e., the prediction can be accurate in the short term but then fails catastrophically as errors accumulate over time). Here, we show that, surprisingly, more data (or higher quality data) can often hurt the performance of machine-learning models by inducing instability in their autonomous prediction mode. We perform detailed analyses of this phenomenon using next-generation reservoir computing (NGRC)—a popular framework introduced recently for data-driven modeling of dynamical systems. Using techniques

from dynamical systems and numerical analysis, we pinpoint the mechanism behind data-induced instability to the auxiliary dimensions created by the delayed states in NGRC. Based on this finding, we propose two simple strategies to mitigate the instability. One is to increase the regularization strength in tandem with the amount of data, while the other is to intentionally decrease the quality of the data by a careful introduction of noise. This approach has the advantage that a fixed noise level can work for any amount of data.

I. INTRODUCTION

Machine learning of dynamical systems (MLDS) has received increasing attention in recent years due to both its theoretical

interest^{1–4} and potential for wide-ranging applications from creating digital twins⁵ to modeling climate⁶ and controlling biological systems.⁷ Specifically, recent advances in MLDS have opened new possibilities in inferring effective brain connections,⁸ predicting tipping points,⁹ discovering pattern-forming dynamics,¹⁰ reconstructing chaotic attractors,¹¹ and anticipating synchronization transitions.¹²

MLDS frameworks span the full spectrum from domain-agnostic deep-learning models to physics-informed models emphasizing interpretability and generalizability.¹³ Domain-agnostic models are usually based on artificial neural networks or other “universal approximator” functions. Popular examples include neural ordinary differential equations (ODEs),¹⁴ neural operators,¹⁵ and reservoir computing.¹⁶ In contrast, physics-based models adopt strong inductive biases, potentially yielding greater accuracy at the expense of expressivity. Some common paradigms in this space include symbolic regression¹⁷ and Sparse Identification of Nonlinear Dynamics (SINDy).¹⁸ Finally, there are also hybrid models^{19–21} that try to find the optimal balance between model scale and domain knowledge.

It is natural to believe that—as in other machine-learning tasks—more data are generally better in MLDS. (A notable exception is the double-descent phenomenon,^{22,23} which we discuss more in Sec. III.) Indeed, one major challenge faced by the MLDS methods in many applications is the paucity of high-quality data (for example, due to noisy, sparse, and/or partial measurements). Here, we show that, sometimes, too much high-quality data can also be a problem if one is not careful. Specifically, we demonstrate that adding more (noise-free) data can hurt the performance of machine-learning models without commensurate regularization. This counter-intuitive phenomenon stems from the potential long-term instability one must contend within forecasting tasks. In contrast to more “static” tasks such as image classification, simply fitting the model to a high-dimensional surface or distribution can be insufficient for data-driven models of dynamical systems—we also need to ensure the long-term stability of the model when run autonomously.

We demonstrate the main phenomenon using next-generation reservoir computing (NGRC),^{24,25} which draws inspiration from the reservoir computing paradigm^{26,27} but is more closely related to statistical forecasting methods such as nonlinear vector-autoregression (NVAR).^{28,29} NGRC has been shown to perform well in many challenging forecasting and control tasks,^{30–33} especially if appropriate nonlinear features are available.³⁴ The relatively simple structure of NGRC models allows us to systematically probe the mechanism behind data-induced instability.

II. RESULTS

A. Next-generation reservoir computing

Consider a dynamical system whose n -dimensional state \mathbf{x} obeys a set of n autonomous differential equations of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}). \quad (1)$$

NGRC aims to learn a representation of \mathbf{f} through a discrete map of the form

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{W} \cdot \mathbf{g}_t, \quad (2)$$

where the index t runs over a set of discrete times separated by Δt , which represents the time resolution of the training data. \mathbf{W} is the $n \times m$ readout matrix of trainable weights and

$$\mathbf{g}_t = \mathbf{g}(\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_{t-k+1}) \quad (3)$$

is an m -dimensional vector of (non)linear features, calculated from the current state and $k - 1$ past states. Here, $k \geq 1$ is a hyperparameter that governs the amount of memory in the NGRC model. The features generally include a constant (bias) term, nk linear terms from the k states, plus a number of user-specified nonlinear features.

During training, given a time series $\{\mathbf{x}_t\}_{t=1, \dots, N_{\text{train}}}$, we seek a \mathbf{W} that minimizes the least-square error between \mathbf{y}_t and $\mathbf{W} \cdot \mathbf{g}_t$, where $\mathbf{y}_t = \mathbf{x}_{t+1} - \mathbf{x}_t$. This is typically achieved via Ridge regression with Tikhonov regularization—a convex optimization problem with a unique solution given by

$$\mathbf{W} = \mathbf{Y} \mathbf{G}^T (\mathbf{G} \mathbf{G}^T + \lambda \mathbb{I})^{-1}. \quad (4)$$

Here, \mathbf{Y} (\mathbf{G}) is a matrix whose columns are \mathbf{y}_t (\mathbf{g}_t) and $\lambda \geq 0$ is the regularization coefficient. Note that training can be performed simultaneously on $N_{\text{traj}} > 1$ trajectories. In this case, one simply concatenates the regressors (\mathbf{g}_t) and regressands (\mathbf{y}_t) obtained from each trajectory.

During the prediction phase, we are given an initial condition \mathbf{x}_0 and $k - 1$ previous states \mathbf{x}_{-1} to \mathbf{x}_{-k} , with which we can iterate Eqs. (2) and (3) as an autonomous dynamical system (each output becomes part of the model’s input at the next time step). All simulations in this study are performed in Julia, using a custom implementation of NGRC.

B. Data-induced instability

We first demonstrate the main phenomenon by applying NGRC to a representative nonlinear system—the magnetic pendulum.³⁵ The system consists of a ferromagnetic bob suspended above the origin of the (x, y) plane. There, three identical point magnets are placed at the vertices of an equilateral triangle with unit side length (Fig. 1). The pendulum bob moves under the influence of magnetic forces as well as gravity and frictional damping, resulting in the following equations of motion:

$$\ddot{x} = -\omega_0^2 x - a\dot{x} - \sum_{i=1}^3 \frac{x - \tilde{x}_i}{\left((x - \tilde{x}_i)^2 + (y - \tilde{y}_i)^2 + h^2 \right)^{\frac{3}{2}}}, \quad (5)$$

$$\ddot{y} = -\omega_0^2 y - a\dot{y} - \sum_{i=1}^3 \frac{y - \tilde{y}_i}{\left((x - \tilde{x}_i)^2 + (y - \tilde{y}_i)^2 + h^2 \right)^{\frac{3}{2}}}. \quad (6)$$

Here, $(\tilde{x}_i, \tilde{y}_i)$ are the coordinates of the i th magnet, ω_0 is the pendulum’s natural frequency, a is the damping coefficient, and h is the bob’s height above the plane. We focus on parameter values for which the system is multistable, with a total of three fixed-point attractors (one for each magnet).

To learn the dynamics of the magnetic pendulum, we train NGRC on varying numbers of trajectories (N_{traj}) obtained from the initial conditions with the bob at rest in different positions (x_0, y_0) . We then ask the trained model to reconstruct the basins of attraction

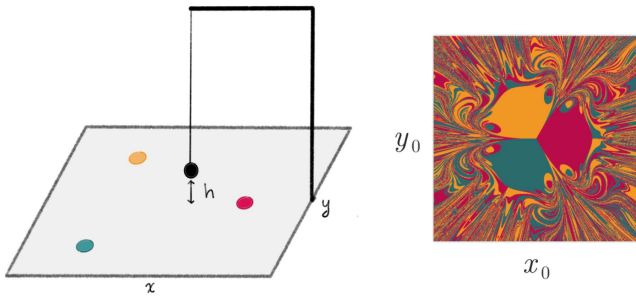


FIG. 1. Magnetic pendulum with three fixed-point attractors and the corresponding basins of attraction. (Left) Schematic of a magnetic pendulum system with three magnets. We take the coordinates of the magnets to be $(\frac{1}{\sqrt{3}}, 0)$, $(\frac{-1}{2\sqrt{3}}, \frac{-1}{2})$, and $(\frac{-1}{2\sqrt{3}}, \frac{1}{2})$ in dimensionless units. The (x, y) coordinates of the pendulum bob and the corresponding velocities (\dot{x}, \dot{y}) fully specify the system's state. (Right) Simulated basins of attraction for the region of initial conditions under study, namely, the states of zero initial velocity with $-1.5 \leq x_0, y_0 \leq 1.5$.

on the (x_0, y_0) plane, again with $\dot{x}_0 = \dot{y}_0 = 0$, that is, to accurately predict the final attractor for a wide range of different initial conditions. This is a challenging task that is crucial in many applications of multistable dynamical systems.^{36–39} We equip NGRC with x - and y -components of the magnetic force terms as its nonlinear features. In this case, NGRC is capable of learning the dynamics well enough to accurately reconstruct the basins.³⁴ Later, we will show that our results also hold for more generic nonlinearities, such as radial basis functions (RBFs).

Figure 2 shows our first main result. We see that as more training trajectories are included, NGRC models capture the intricate basins better and better, as expected. However, at some point, the trained model becomes unstable—all forecasted trajectories diverge to infinity instead of converging to one of the three fixed-point attractors. At any fixed regularization strength (λ), this instability transition is attributable solely to the amount of training data. Indeed, once the number of training trajectories crosses a threshold, the NGRC model—perfectly stable when trained with less data—blows up. This sudden breakdown occurs for a wide range of regularization coefficients λ , with more data needed to induce instability for more aggressive regularization. This is surprising in light of the usual expectation that in machine learning, more data usually translate into better performance with less overfitting, especially when the data are independent and noise-free, as here. Interestingly, the same phenomenon afflicts forecasting chaos in the Lorenz system, where larger regularization is needed to ensure model stability as N_{traj} increases (supplementary material, Fig. S1).

So what is causing this data-induced instability? We will offer two complementary explanations in the sections that follow. However, first, we demonstrate what is *not* the cause, which happens to be most people's first guess.

C. It is not the flow map!

From Eq. (2), we see that NGRC learns the dynamics from data by constructing a map from \mathbf{x}_t to \mathbf{x}_{t+1} . In dynamical systems theory,

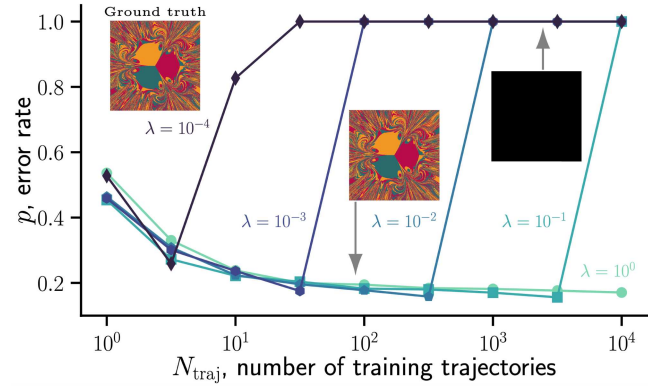


FIG. 2. More (noise-free) data can induce instability in NGRC. As we increase the number of trajectories N_{traj} used to train NGRC (with pendulum force terms as nonlinear features), the model can undergo a sudden transition from accurately capturing the three fractal-like basins to losing stability and sending all trajectories to infinity. This happens for a wide range of regularization coefficients λ , although larger λ can delay the onset of instability. We measure the performance of a trained NGRC model using the error rate p , which is the percentage of initial conditions for which the model predicts the wrong attractor. Each data point is obtained by averaging p over ten independent trials. The ground-truth basins and representative NGRC predictions are shown as insets. Black denotes trajectories that diverged to infinity. Here, the pendulum parameters are set to $\omega_0 = 0.5$, $a = 0.2$, and $h = 0.2$, and the NGRC hyperparameters used are $k = 2$, $\Delta t = 0.01$, and $N_{\text{train}} = 3000$.

a flow map predicts the state of the system Δt time units later based on the current state. For example, the magnetic pendulum system induces a flow map $\Phi_{\Delta t}$ from \mathbb{R}^4 to \mathbb{R}^4 ,

$$\Phi_{\Delta t}(x_t, y_t, \dot{x}_t, \dot{y}_t) = (x_{t+\Delta t}, y_{t+\Delta t}, \dot{x}_{t+\Delta t}, \dot{y}_{t+\Delta t}). \quad (7)$$

Equivalently, we can encode the information in the flow surface,

$$\phi_{\Delta t}(\mathbf{x}_t) = \Phi_{\Delta t}(\mathbf{x}_t) - \mathbf{x}_t = (\Delta x_t, \Delta y_t, \Delta \dot{x}_t, \Delta \dot{y}_t). \quad (8)$$

Once an accurate flow surface has been constructed, NGRC can, in principle, make reliable forecasts from any initial conditions (at least in parts of the state space where there were enough training data). In this sense, the goal of NGRC is not dissimilar to many other machine-learning frameworks—find an accurate fit of the target high-dimensional manifold and use it to make predictions.

Of course, in practice, we only have a finite amount of data, and like many other machine-learning tasks, one needs to worry about overfitting. Could it be that as NGRC tries to fit more and more data points on the flow surface, it generates a pathological surface that contorts itself to fit the training data, diverging from the true flow surface in the gaps between?

Figure 3 shows that this hypothesis, however natural, fails to explain the data-induced instability. There, we characterize the mismatch between the model-predicted and true flow surfaces on a 2D sub-manifold $(x, y, 0, 0) \rightarrow \Delta \dot{x}$ via the fitting error (e) between the real/NGRC flow maps for the x -velocity, namely,

$$e = \|\Delta \dot{x}^{(\text{real})} - \Delta \dot{x}^{(\text{NGRC})}\|. \quad (9)$$

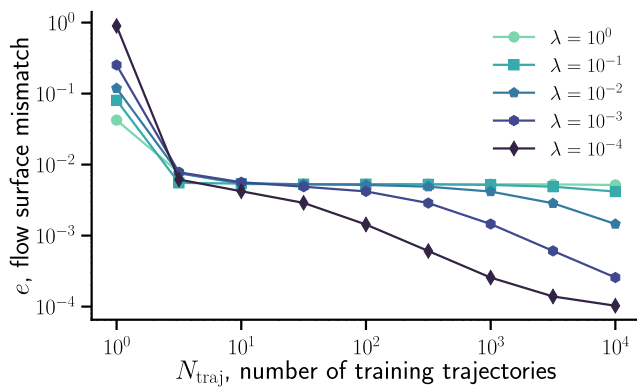


FIG. 3. NGRC does not overfit the flow surface. Using the same setup as in Fig. 2, we plot the fitting error [e, Eq. (9)] against the number of training trajectories N_{traj} . The fitting error monotonically decreases for all values of the regularization coefficient λ considered, giving no indication of any over-fitting that might lead to instability.

We observe no signs of overfitting. In fact, as more and more training trajectories are included, e keeps decreasing for all values of regularization coefficient λ considered. We emphasize that e is evaluated on a uniform and dense grid independent of the training data points. These results remain qualitatively the same if we examine the fitting error over other 2D sub-manifolds of the flow surface, such as $(0, 0, \dot{x}, \dot{y}) \rightarrow \Delta y$.

Figure 4 further rules out overfitting as the cause of data-induced instability. This time, we replace the pendulum force terms with 1000 radial basis functions (RBF)⁴⁰ of the same form considered in Ref. 34, so as to demonstrate that the same instability occurs for NGRC models with generic nonlinearities. We see that NGRC is stable when trained with 100 trajectories and predicts the basins well, despite noticeable deviations in its fit to the flow surface. When trained with 1000 trajectories, NGRC reconstructs a much better flow surface. However, its predictions now diverge to infinity for all initial conditions tested. If it is not the flow map, then what is the cause of this instability?

D. Integrator perspective

In this subsection, inspired by the original paper,²⁴ we view NGRC models as integrators, and we show that NGRC instability can be understood by using the techniques from numerical analysis. For the purpose of demonstration, it is easiest to consider NGRC models with pendulum force terms as nonlinear features. In this case, all terms in the original ODE [Eqs. (5) and (6)] are available in NGRC’s feature library. This allows direct comparison of NGRC models with traditional integration schemes.

The integrators that most resemble the NGRC architecture are the linear multistep methods, defined by

$$\sum_{j=0}^s a_j \mathbf{x}_{i+j} = \Delta t \sum_{j=0}^s b_j \mathbf{f}(t_{i+j}, \mathbf{x}_{i+j}), \quad (10)$$

which uses the states and derivatives from s previous steps to predict \mathbf{x}_{i+s} and integrate the equation $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ forward. (For implicit methods, such as Adams–Moulton, $b_s \neq 0$ and the integrator also uses information about the derivative at the future step.) Because the NGRC models make updates using only the current and past states [cf. Eq. (2)], it is most analogous to the popular Adams–Bashforth methods, which are explicit methods with $a_{s-1} = -1, a_{s-2} = \dots = a_0 = 0$, and $b_s = 0$. The other b_j ($0 \leq j < s$) are chosen such that the method has order s [i.e., the local truncation error of the integrator is of order $\mathcal{O}(\Delta t^{s+1})$], which determines b_j uniquely. We stress, however, that an NGRC model with exact nonlinearities is not necessarily a special case of Adams–Bashforth or similar methods. This is because it works directly with different scalar features (instead of the vector field \mathbf{f} as a whole). Accordingly, NGRC models can assign different weights to individual terms in its features library, unconstrained by the right-hand side of the ODEs.

For $k = 2$, the natural integrator to compare the NGRC models with is the two-step Adams–Bashforth method,

$$\mathbf{x}_{i+2} = \mathbf{x}_{i+1} + \Delta t \left(\frac{3}{2} \mathbf{f}(t_{i+1}, \mathbf{x}_{i+1}) - \frac{1}{2} \mathbf{f}(t_i, \mathbf{x}_i) \right). \quad (11)$$

Do NGRC models approximately rediscover the two-step Adams–Bashforth method on their own? In Table I, we compare the readout matrix of NGRC models with the two-step Adams–Bashforth method. For this purpose, it is helpful to rewrite Eqs. (5) and (6) as first-order ODEs,

$$\dot{x} = v_x, \quad (12)$$

$$\dot{y} = v_y, \quad (13)$$

$$\dot{v}_x = -\omega_0^2 x - a v_x + \sum_{i=1}^3 F_{i,x}, \quad (14)$$

$$\dot{v}_y = -\omega_0^2 y - a v_y + \sum_{i=1}^3 F_{i,y}, \quad (15)$$

where we use F_i as the x - and y - components of the force from the i th magnet. From Eqs. (12) to (15), we can easily infer what two-step Adams–Bashforth thinks each entry in the readout matrix should be, which are summarized in the top table (for $\omega_0 = 0.5$ and $a = 0.2$). Now looking at the trained NGRC models, we see they are quite different from the two-step Adams–Bashforth method. For example, an inspection of the weights for $N_{\text{traj}} = 10$ reveals that instead of splitting the coefficients between the current state and the delayed state $\frac{3}{2}$ to $-\frac{1}{2}$ (optimal, as in Adams–Bashforth), NGRC splits the coefficients equally for many of the linear terms (compare the terms highlighted in orange). The weights do capture the $\frac{3}{2}$ to $-\frac{1}{2}$ split for the nonlinear terms. However, the model is often confused about which way the split should go, routinely assigning the $-\frac{1}{2}$ factor to the current state instead of the delayed state (compare the terms highlighted in green).

The most striking departure from Adams–Bashforth, however, lies in the coefficients for v_x and v_y in relation to Eqs. (14) and (15), which are highlighted in red and blue, respectively. There, instead of the optimal factor pairs $\{\frac{3}{2}, -\frac{1}{2}\}$, the NGRC models assign factors

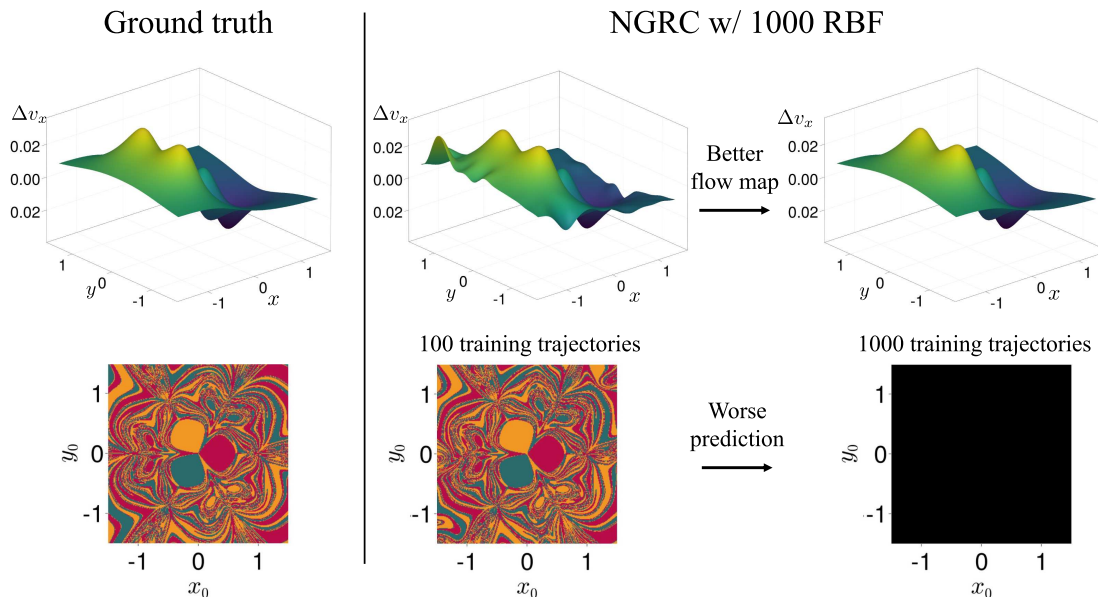


FIG. 4. Instability in NGRC cannot be explained by errors in fitting the flow surface. Here, we demonstrate this point using the magnetic pendulum system and NGRC with 1000 radial basis functions (RBF) as nonlinear features. Each RBF has a center chosen randomly and uniformly from $(x_0, y_0) \in [-1.5, 1.5]^2$. When trained with $N_{\text{traj}} = 100$ trajectories, NGRC achieves a passable (but far from perfect) fit of the flow surface and does a good job in capturing the basins (error rate $p \approx 0.2$). When we increased N_{traj} to 1000, the model learned a near-perfect flow map, but it became unstable and all predictions blew up. The pendulum parameters are set to $\omega_0 = 0.5$, $a = 0.2$, and $h = 0.4$, and the NGRC parameters used are $k = 2$, $\lambda = 10^{-5}$, $\Delta t = 0.01$, and $N_{\text{train}} = 5000$.

far away from zero to both the current state and the delayed state. These factors are similar in magnitude but opposite in signs and, when combined together, roughly recover the corresponding coefficient on the right-hand side of the ODE (i.e., the sum of the factors is close to 1). These magnitudes grow rapidly with N_{traj} , eventually making the NGRC model unstable for $N_{\text{traj}} = 1000$.

At this point, one might say, wait a moment, is not a flow map all you need to advance a dynamical system in time? Where does the integrator come in? If NGRC has a perfect fit of the flow surface, how can it still be unstable? In Sec. II E, we address these questions by identifying a source of data-induced instability: the ill-conditioning of matrix \mathbf{G}^T in Eq. (4). Each row of \mathbf{G}^T represents the evaluation of the various candidate features using a set of k consecutive states from a training trajectory. For small Δt , when the same features are evaluated at successive time points, the corresponding column vectors in \mathbf{G}^T are nearly linearly dependent, leading to a nearly rank-deficient \mathbf{G}^T . This ill-conditioning explains the uncontrolled growth of the NGRC weights with N_{traj} observed in Table I.

E. Ill-conditioned \mathbf{G}^T is a source of data-induced instability

To better understand the source of the ill-conditioning of \mathbf{G}^T , we use $k = 2$ as an example and, hence, split \mathbf{G}^T into two submatrices,

$$\mathbf{G}^T = [\mathbf{A}, \mathbf{B}], \tag{16}$$

where \mathbf{A} is the $N_{\text{train}}N_{\text{traj}} \times l$ matrix with feature terms evaluated at the current time, and \mathbf{B} is the $N_{\text{train}}N_{\text{traj}} \times m$ matrix collecting all remaining terms, which include the bias term and the features evaluated at delayed states, as illustrated in the left panel of Fig. 5. For the magnetic pendulum, the original equations of motion comprise $l = 10$ features in \mathbf{A} , while \mathbf{B} includes the constant feature and delayed counterparts resulting in $m = 11$. As shown in the middle panel in Fig. 5, the ill-conditioning of \mathbf{G}^T cannot be attributed to either submatrix \mathbf{A} or \mathbf{B} individually. We compute the condition numbers $\kappa(\mathbf{G}^T)$, $\kappa(\mathbf{A})$, and $\kappa(\mathbf{B})$ from one trajectory, and the histogram represents the distribution of these condition numbers over different trajectories. Matrix \mathbf{G}^T is ill-conditioned when compared to its submatrices, with $\kappa(\mathbf{G}^T)$ typically around $\mathcal{O}(10^{10})$.

What happens when more data are added? Adding different trajectories into matrix \mathbf{G}^T expands the space spanned by its column vectors. For example, due to the high sensitivity of the magnetic pendulum to the initial conditions, those trajectories sampled uniformly during training are likely to be distinct. While enlarging \mathbf{G}^T would theoretically improve its conditioning, this is not observed numerically. Notably, the ill-conditioning persists as more training data are added. The right panel in Fig. 5 shows that the linear dependence between the column space of \mathbf{B} , $\mathcal{R}(\mathbf{B})$, and \mathbf{A} , $\mathcal{R}(\mathbf{A})$, quantified by principal angles between subspaces (PABS),⁴¹ remains even with a hundredfold increase of the number of trajectories. A cluster of principal angles $\theta_{\text{AB}}^i \in [0, \frac{\pi}{2}]$ for $i = 1, \dots, \min\{l, m\}$ near zero indicates that at least one direction in $\mathcal{R}(\mathbf{A})$ aligns with $\mathcal{R}(\mathbf{B})$, regardless of

TABLE I. NGRC learns an increasingly unstable integrator as more data are included in the training. Here, we compare NGRC models with the closest numerical integrator—two-step Adams–Bashforth from the linear multistep methods family. We use the same setup to train the NGRC models as in Fig. 2, with $\lambda = 0.01$ and N_{traj} ranging from 10 to 1000. The NGRC models are stable for $N_{\text{traj}} = 10$ and 100 (error rate $p = 0.23$ and 0.18, respectively), but it becomes unstable for $N_{\text{traj}} = 1000$ ($p = 1$). To save space, we denote the delayed states with a hat (e.g., \hat{x}) and the x - and y -components of the nonlinear terms with $F_{i,\cdot}$. The key observation is that, as N_{traj} is increased, we have an increasingly severe imbalance between the weights assigned to the current states and the delayed states. For the two-step Adams–Bashforth method, the current state is assigned a weight that is $\frac{3}{2}$ times the corresponding coefficient on the right-hand side of the ODE, whereas the delayed state has a factor of $-\frac{1}{2}$. This assignment is optimal in the sense that the factors $\{\frac{3}{2}, -\frac{1}{2}\}$ are the only pair that makes the method an order-two integrator [i.e., its local truncation error is $\mathcal{O}(\Delta t^3)$]. In contrast, the learned weights are significantly larger and become increasingly unbalanced as more training data are used. For example, see the pairs highlighted in red and blue for the models trained with different N_{traj} . The factors are almost equal numbers with opposite signs and, despite being orders of magnitude larger, their sum largely cancels each other and roughly match the corresponding coefficient on the right-hand side of the ODE. This imbalance is directly reflected in the increasingly large condition number κ of the NGRC readout matrix and is what eventually leads to the data-induced instability.

	x	y	v_x	v_y	\hat{x}	\hat{y}	\hat{v}_x	\hat{v}_y	$F_{1,x}$	$F_{1,y}$	$F_{2,x}$	$F_{2,y}$	$F_{3,x}$	$F_{3,y}$	$F_{1,\hat{x}}$	$F_{1,\hat{y}}$	$F_{2,\hat{x}}$	$F_{2,\hat{y}}$	$F_{3,\hat{x}}$	$F_{3,\hat{y}}$
Two-step Adams–Bashforth																				
$\Delta x/\Delta t$	0.00	0.00	1.50	0.00	0.00	0.00	-0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\Delta y/\Delta t$	0.00	0.00	0.00	1.50	0.00	0.00	0.00	-0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\Delta v_x/\Delta t$	-0.375	0.00	-0.30	0.00	0.125	0.00	0.10	0.00	1.50	0.00	1.50	0.00	1.50	0.00	-0.50	0.00	-0.50	0.00	-0.50	0.00
$\Delta v_y/\Delta t$	0.00	-0.375	0.00	-0.30	0.00	0.125	0.00	0.10	0.00	1.50	0.00	1.50	0.00	1.50	0.00	-0.50	0.00	-0.50	0.00	-0.50
$N_{\text{traj}} = 10, \kappa = 5, p = 0.23$																				
$\Delta x/\Delta t$	0.00	0.00	0.50	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00
$\Delta y/\Delta t$	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01
$\Delta v_x/\Delta t$	-0.13	0.00	1.06	-0.12	-0.13	0.00	-1.26	0.12	-0.49	0.00	1.50	0.00	-0.49	0.00	1.50	0.00	-0.49	0.00	1.50	0.00
$\Delta v_y/\Delta t$	0.00	-0.13	-0.11	1.30	0.00	-0.13	0.11	-1.50	0.00	-0.49	0.00	1.50	0.00	-0.49	0.00	1.50	0.00	-0.49	0.00	1.50
$N_{\text{traj}} = 100, \kappa = 27, p = 0.18$																				
$\Delta x/\Delta t$	0.00	0.00	0.53	0.00	0.00	0.00	0.47	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00
$\Delta y/\Delta t$	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.47	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01
$\Delta v_x/\Delta t$	-0.14	0.00	13.77	-0.39	-0.14	0.00	-14.00	0.39	-0.43	0.00	1.56	0.00	-0.43	0.00	1.56	0.00	-0.43	0.00	1.56	0.00
$\Delta v_y/\Delta t$	0.00	-0.14	-0.40	13.93	0.00	-0.15	0.40	-14.16	0.00	-0.43	0.00	1.56	0.00	-0.43	0.00	1.56	0.00	-0.43	0.00	1.56
$N_{\text{traj}} = 1000, \kappa = 215, p = 1$																				
$\Delta x/\Delta t$	0.00	0.00	0.72	0.00	0.00	0.00	0.28	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00
$\Delta y/\Delta t$	0.00	0.00	0.00	0.72	0.00	0.00	0.00	0.28	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01
$\Delta v_x/\Delta t$	-0.26	0.00	108.68	0.51	-0.26	0.00	-109.11	-0.51	0.05	0.00	2.03	0.00	0.05	0.00	2.03	0.00	0.05	0.00	2.03	0.00
$\Delta v_y/\Delta t$	0.00	-0.26	0.42	109.56	0.00	-0.26	-0.42	-109.98	0.00	0.05	0.00	2.04	0.00	0.05	0.00	2.04	0.00	0.05	0.00	2.04

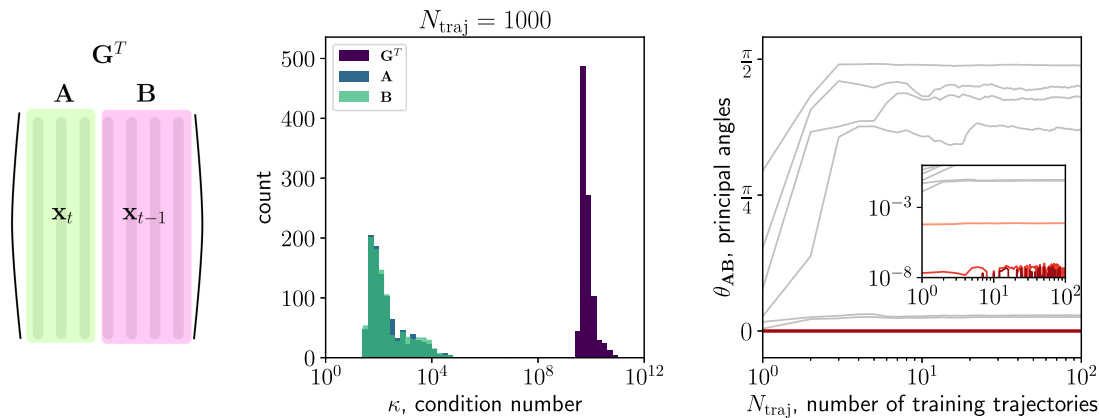


FIG. 5. Ill-conditioning arises from the linear dependence between the current and delayed states. The left panel illustrates that for $k = 2$, matrix \mathbf{G}^T can be partitioned into two submatrices corresponding to the features evaluated at the current and delayed states along the time series. For the magnetic pendulum, \mathbf{A} and \mathbf{B} are $N_{\text{train}}N_{\text{traj}} \times 10$ and $N_{\text{train}}N_{\text{traj}} \times 11$ matrices, respectively. The middle panel displays the histogram of the condition number of each matrix \mathbf{G}^T , \mathbf{A} , and \mathbf{B} for $N_{\text{traj}} = 1000$ trajectories. While \mathbf{A} and \mathbf{B} are well-conditioned, with overlapping histograms around $\mathcal{O}(10^3)$, \mathbf{G}^T is ill-conditioned, exhibiting a condition number, orders of magnitude larger than its submatrices. The right panel shows the ten principal angles between the column spaces of \mathbf{A} and \mathbf{B} , as the number of trajectories is increased. Even a 100-fold increase in trajectories has no impact on the group of principal angles near zero, indicating linear dependence between the column spaces $\mathcal{R}(\mathbf{A})$ and $\mathcal{R}(\mathbf{B})$. This is corroborated by the inset panel, which highlights a few principal angles below 10^{-1} on a logarithmic scale. Principal angles are color-coded: those below 10^{-3} appear in red, while larger angles are shown in gray. All settings and parameters are the same as in Fig. 2.

the amount of training data. This near-linear dependence persists because features are evaluated at consecutive times. Consequently, adding data only increases the size of \mathbf{G}^T used in the Ridge regression [Eq. (4)], leading to an increase in the norm of \mathbf{W} , or equivalently, the condition number $\kappa(\mathbf{W})$, which tends to grow as shown in Fig. 6.

These numerical observations can be made more precisely by analyzing in more detail the least-square problem [$\lambda = 0$ in Eq. (4)].

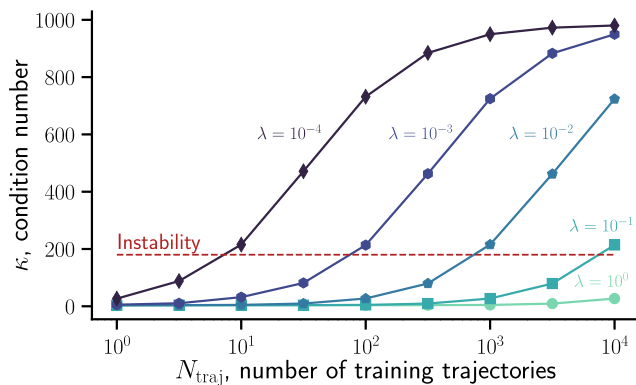


FIG. 6. Condition number as an indicator for the instability in NGRC. Using the same setup as in Fig. 2, we plot the condition number $\kappa(\mathbf{W})$ of the readout matrix against the number of training trajectories N_{traj} . The condition number monotonically increases as more training trajectories are included. Once a threshold has been crossed ($\kappa(\mathbf{W}) \approx 180$), NGRC becomes unstable (cf. Fig. 2). We note, however, that this threshold based on the condition number is likely system-specific.

The least-square problem approximates \mathbf{Y} , induced by the flow surface $\phi_{\Delta t}$ in Eq. (8), with a vector lying in the column space of \mathbf{G}^T . Since \mathbf{G}^T is a partitioned matrix, the column space of \mathbf{A} corresponds to the directions associated with feature terms in the original equations of motion Eqs. (12)–(15) and, thus, the flow surface itself. In contrast, the transverse directions of the flow surface are spanned by vectors lying in $\mathcal{R}(\mathbf{B})$. Therefore, it follows that

Result II.1. For $\lambda = 0$, whenever the minimum principal angle $\theta_{\mathbf{AB}}$ is close to zero, the solution of Eq. (4), \mathbf{W} , has nonzero entries (corresponding to the transverse directions) that grow with N_{traj} .

The idea is to adapt Ref. 42 in the context of the NGRC model, using sensitivity analysis of the exact least-square regression problem [$\lambda = 0$ in Eq. (4)]. We refer the reader to SM.IV in the supplementary material for details. The ill-conditioning in data matrix \mathbf{G} and increasing N_{traj} inevitably leads to growing weights (entries of \mathbf{W}) corresponding to the transverse directions in the solution of the least-square problem.⁴³ The result also extends to a small regularization coefficient λ , where \mathbf{W} is computed using the Ridge regression in Eq. (4). These growing coefficient entries of \mathbf{W} in the transverse direction potentially lead to an unstable NGRC model. Without a mechanism to constrain the NGRC dynamics, NGRC does not learn a stable integration scheme, and the autonomous dynamics consequently diverge to infinity after a handful of iterations.

Geometrically, the key insight is the following: for the magnetic pendulum system, flow map $\Phi_{\Delta t}$ is defined on $\mathbb{R}^4 \rightarrow \mathbb{R}^4$; for NGRC, because of the inclusion of delayed states, map Ψ is actually defined on $\mathbb{R}^{4k} \rightarrow \mathbb{R}^4$. Thus, for $k > 1$, NGRC is learning a higher dimensional map than the true flow map. During training, because

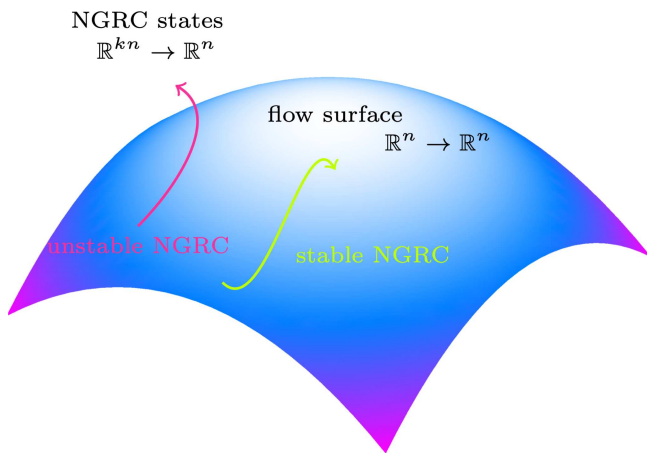


FIG. 7. Stable NGRCs stay close to the flow surface whereas unstable NGRCs move away from the flow surface. During the prediction phase, a stable NGRC model would stay close to the low-dimensional sub-manifold where the $\mathbb{R}^n \rightarrow \mathbb{R}^n$ flow surface is defined. This is also where all training data come from (and, thus, where meaningful fitting happens). An unstable NGRC model, in contrast, would move rapidly away from the flow surface due to transverse instability created by the delayed states, which typically results in the predictions blowing up.

all data come from the real system, NGRC only sees the $\mathbb{R}^4 \rightarrow \mathbb{R}^4$ sub-manifold of Ψ , on which $\Phi_{\Delta t}$ is defined. It is, thus, no surprise that, as NGRC tries to fit more and more data on the $\mathbb{R}^4 \rightarrow \mathbb{R}^4$ sub-manifold, it inevitably creates instability in the other $4(k - 1)$ directions transverse to the sub-manifold. When such transverse instabilities exist, NGRC trajectories will move away exponentially from that sub-manifold (so as long as we do not start exactly on the subspace where the flow map is defined). In this case, the fitting error on the flow surface is meaningless, no matter how good the fit is. The instability of NGRC (as an integrator) we characterized in Sec. II D is hidden and not reflected in the flow surfaces visualized in Sec. II C because we are using the ground truth for the delayed states. Instead, the instability only becomes apparent when looking at the full $\mathbb{R}^{4k} \rightarrow \mathbb{R}^4$ space (in particular, at directions transverse to the low-dimensional flow surface). We illustrate this geometric origin of NGRC instability in Fig. 7 and show the transverse instability explicitly in Fig. 8.

The results above have a few interesting corollaries. First, we can make \mathbf{G}^T better-conditioned (and the corresponding NGRC model more stable) simply by setting $k = 1$ (i.e., no delayed states). The nearly linear dependence between the current and delayed states is absent, making the entries of \mathbf{W} to not grow with more training data. Indeed, we no longer observe any instability for NGRC in this case. However, we pay a price by eschewing the delayed states: although stable, NGRC becomes much less expressive—it cannot fit the flow surface well and the prediction performance is poor unless the sampling rate is unrealistically high (i.e., $\Delta t \rightarrow 0$). Alternatively, \mathbf{G}^T becomes less ill-conditioned if the time skip (delay) between the current state and the delayed state is increased (see SM.III A in the supplementary material for details). However, increasing the

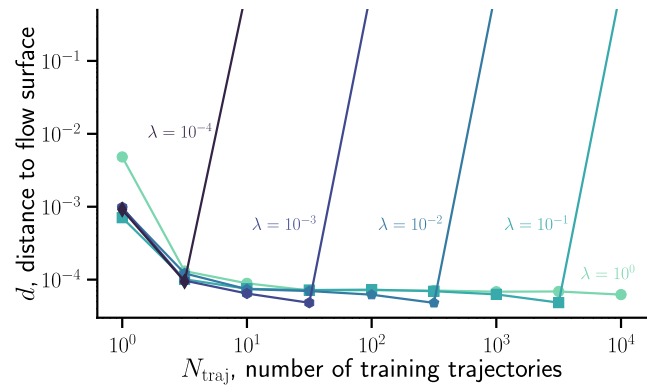


FIG. 8. More data can create transverse instability on the flow surface. Using the same setup as in Fig. 2, we plot the average distance to the flow surface against the number of training trajectories. The average distance d is obtained by averaging the distance to the flow surface over the NGRC-predicted trajectory for 100 time units. For a given regularization coefficient λ , transverse instability develops beyond a certain number of training trajectories, and the NGRC predictions diverge exponentially away from the low-dimensional flow surface (d jumps from around 10^{-4} to numerical infinity). We note that the transverse instability sets in exactly when NGRC becomes unstable (cf. Fig. 2).

time skip demands more training data in addition to creating a new hyperparameter to optimize, which can be challenging in practical applications. Finally, the NGRC model can also be stabilized by reducing $\kappa(\mathbf{W})$. For example, for a fixed N_{traj} , if $\kappa(\mathbf{W})$ is above the instability threshold, we can stabilize NGRC by increasing the regularization coefficient λ . Operationally, more aggressive regularization helps us to dampen the pseudo-inverse of \mathbf{G}^T , filtering components corresponding to small singular values and mitigating the spurious linearly dependent directions between the current and delayed states.

F. How to fix this?

The game we have to play is controlling the ill-conditioning of \mathbf{G}^T to avoid the unstable NGRC dynamics. So are there simple ways to mitigate data-induced instabilities? Here, we discuss two simple strategies to overcome the ill-conditioned \mathbf{G}^T : data size-dependent regularization coefficient and training with noise.

Data size-dependent regularization coefficient: To make progress, it is helpful to recall the loss function for Ridge regression,

$$\ell(\mathbf{W}) = \sum_{i=1}^{N_{\text{traj}}} \|\mathbf{Y}_i - \mathbf{W} \cdot \mathbf{G}_i\|^2 + \lambda \|\mathbf{W}\|^2. \quad (17)$$

Here, $\|\cdot\|$ is the Frobenius norm (elementwise, the L_2 norm). \mathbf{Y}_i (\mathbf{G}_i) is a matrix whose columns are the states \mathbf{y}_t (features \mathbf{g}_t) from the i th training trajectory. Assuming that each training trajectory gives rise to a similar fitting error, we can see that the first term in Eq. (17) is proportional to the number of training trajectories N_{traj} . Thus, as the training data increases, the fitting error in Eq. (17) becomes more and more dominant, effectively decreasing the regularization strength λ and, eventually, leading to an under-regularized model.

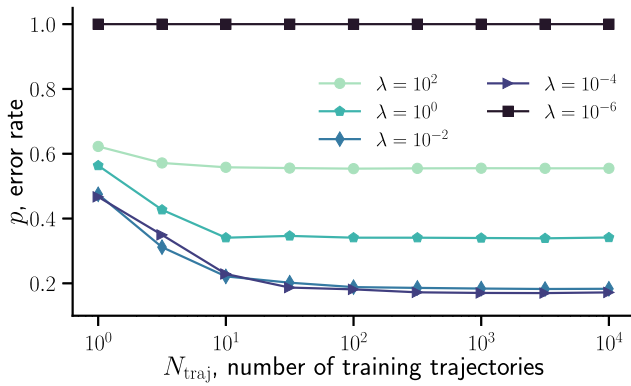


FIG. 9. Proper scaling of the regularization strength fixes data-induced instability in NGRC. This is the analog of Fig. 2 but with the regularization coefficient λ scaled linearly by the amount of training data [cf. Eq. (18)]. Now, as we increase the number of training trajectories N_{traj} , the trained NGRC no longer undergoes instability transitions, and the error rate p stabilizes for large N_{traj} . If the regularization is insufficient, e.g., $\lambda = 10^{-6}$, then the trained NGRC is unstable for all N_{traj} . If the regularization is too large, e.g., $\lambda = 10^2$, it will negatively impact the prediction accuracy. All settings and parameters other than regularization are the same as in Fig. 2.

In order to maintain a roughly constant model complexity (measured here by $\|\mathbf{W}\|$), the regularization coefficient λ should scale linearly with the amount of training data.

Next, we revisit the magnetic pendulum system using the new loss function defined below, which increases regularization accordingly as more training data are introduced,

$$\ell(\mathbf{W}) = \sum_{i=1}^{N_{\text{traj}}} \|\mathbf{Y}_i - \mathbf{W} \cdot \mathbf{G}_i\|^2 + \lambda N_{\text{traj}} \|\mathbf{W}\|^2. \quad (18)$$

Figure 9 shows the basin prediction error rate p as a function of N_{traj} . Compared to Fig. 2, we see that there is no longer an instability transition at large N_{traj} , showing that the scaled regularization in Eq. (18) can successfully suppress the data-induced instability. We have also confirmed that the condition number κ becomes roughly independent of N_{traj} , indicating a nearly constant model complexity $\|\mathbf{W}\|$ regardless of the amount of training data. As expected, there is a sweet spot of intermediate regularization that optimizes accuracy while ensuring stability. In the case of Fig. 9, it is $\lambda \in (10^{-4}, 10^{-1})$.

Figure 10 tells a similar story: after an initial decrease in the fitting error of the flow surface, it approaches a constant value as N_{traj} is further increased. Smaller λ leads to smaller e , but a λ that is too small (e.g., $\lambda = 10^{-6}$) triggers instability transverse to the flow surface and destabilizes the NGRC model.

Training with noise: Aside from properly scaling the Ridge regression regularization strength λ , a possible alternative strategy to mitigate instability is adding noise to the training data. Such noise regularization has been shown to improve the performance and robustness of reservoir computers.⁴⁴ There are also intimate connections between noise regularization and Ridge regression when i.i.d. noise is added directly to the regressors.⁴⁵ One potential advantage of regularizing with noise is that a constant level of noise may suffice

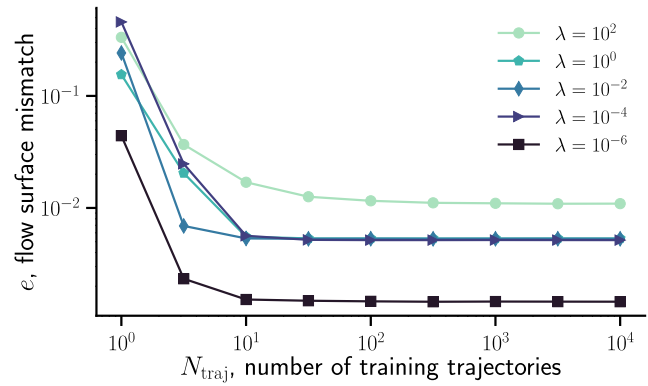


FIG. 10. Under scaled regularization, fitting error to the flow surface saturates as more training data are included. This is the analog of Fig. 3 but with the regularization coefficient λ scaled linearly by the amount of training data [cf. Eq. (18)]. Instead of the fitting error e continuously decreasing with N_{traj} , it stabilizes quickly as the optimization achieves a balance between the error term and the regularization term. All settings and parameters other than regularization are the same as in Fig. 3.

to stabilize the trained model independent of the amount of training data used.

We have tested this possibility by applying additive noise to the training time series before calculating the features \mathbf{g}_t [cf. Eq. (3)]. At each time step t , we draw noise uniformly and independently from $[-\sigma, \sigma]$ for each of the $n \cdot k$ components in $\{\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_{t-k+1}\}$. Here, σ is a parameter that controls the noise strength. We stress that only the training inputs are modified; the next-step targets ($\mathbf{y}_t = \mathbf{x}_{t+1} - \mathbf{x}_t$) are still determined based on the noise-free data. Noise breaks the linear dependence between current and delayed states because random matrices whose entries are i.i.d. contain almost surely a set of linear independent random vectors. Intuitively, this forces the NGRC model to learn a contraction map that flows back to the flow surface in its vicinity.

Figure 11 shows the basin prediction error p vs the number of training trajectories N_{traj} for varying noise strengths σ . For this analysis, we set the (unscaled) Tikhonov regularization $\lambda = 10^{-4}$, for which model instability appeared at $N_{\text{traj}} \approx 10$ sans noise regularization (cf. Fig. 2). We see that the effects of noise are a mixed blessing. With sufficient noise strength, it does mitigate the harmful effects of more data. In particular, for a given noise strength σ , the basin prediction error (p) is asymptotically independent of N_{traj} . However, the precise value of p one attains as $N_{\text{traj}} \rightarrow \infty$ is exquisitely sensitive to the noise strength. When σ is too low (e.g., 8×10^{-5}), noise fails to stabilize; we still see a transition to $p = 100\%$ at a finite value of N_{traj} . Increasing σ only slightly to 10^{-4} stabilizes the model regardless of N_{traj} and, in fact, achieves the lowest asymptotic error rate observed. Unfortunately, increasing σ any further from here rapidly degrades model performance (Fig. 11). All told, there is a narrow range of σ —less than an order of magnitude—in which noise is strong enough to be an effective regularizer but not so strong that it hinders the learning of the flow map.

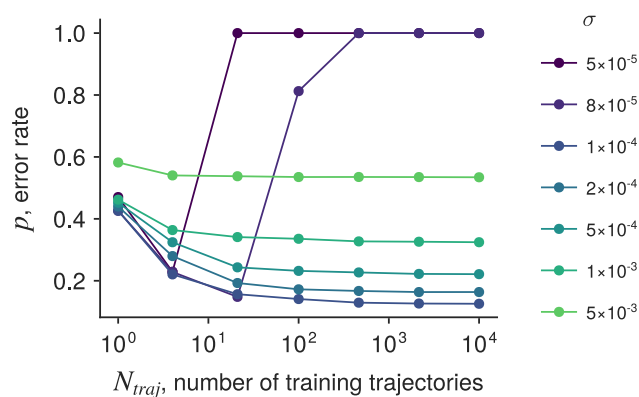


FIG. 11. Noise-based regularization can stabilize NGRC but is sensitive to the choice of noise strength. Unlike Tikhonov regularization in Fig. 2, noise-based regularization with a fixed strength can stabilize NGRC models for all data sizes. However, noise strength σ needs to be chosen carefully to ensure stability without sacrificing accuracy. Each data point represents the average over ten independent trials. Here, we set $\lambda = 10^{-4}$. All other hyperparameters are the same as in Fig. 2.

III. DISCUSSION

Here, we have shown how more data can induce instability when learning unknown dynamical systems. We focused on next-generation reservoir computing, a simple but powerful framework recently introduced to complement the standard reservoir computing paradigm.²⁴ We showed that the data-induced instability observed in NGRC does not come from overfitting the flow map. Instead, it has to do with the instabilities along auxiliary dimensions transverse to the flow map. Finally, we linked such instabilities to the fact that training with increasing amounts of data at fixed regularization can lead to under-regularized models. Motivated by this observation, we proposed simple fixes either by increasing the regularization strength in proportion to data size or by introducing the right amount of noise.

Similar instabilities can arise in other high-dimensional machine-learning models that involve solving a linear regression with an ill-conditioned feature matrix. For instance, in network reconstruction problems, adding “unnecessary” features can result in ill-conditioning.⁴² Traditional reservoir computers (e.g., echo-state networks)^{46–63} also exhibit ill-conditioned matrices⁵⁴ that affect dynamical stability. For example,⁴⁷ reports a failure to reproduce the correct Lyapunov exponents without regularization. However, the source of ill-conditioning in RCs differs from that in NGRC. Traditional RCs do not explicitly use past states of the system as features; instead, the dynamics emerge from the evolution of the reservoir states. The stability of these dynamics is typically controlled by the spectral radius of the reservoir’s internal weight matrix. Setting the spectral radius less than one ensures a contraction in the reservoir state space, which, for some classes of systems, guarantees the echo-state property.⁶⁵ Moreover, hyperparameters like the spectral radius and the regularizer parameter are often tuned in an optimization method to maintain stability during testing, hiding such data-induced instability. This is different from NGRC, where the

readout matrix needs to “deal by itself” with potential instabilities in the NGRC dynamics. Thus, the benefit of NGRC in reducing the computational cost of optimizing in higher dimensional hyper-parameter space requires a proper regularization scaling.

The deleterious effect of more training data is also a crucial aspect of the “double-descent” phenomenon,^{22,66} in which model performance depends non-monotonically on the ratio of parameters/data. Double descent has been demonstrated in simple regression tasks⁶⁷ as well as deep neural networks such as ResNet and Transformers.²³ It has been shown that regularization plays a key role in the double-descent phenomenon.^{68–72} In particular, double descent mostly appears in under-regularized models, and optimal regularization can often completely suppress the test error peak at the interpolation threshold (i.e., model performance improves monotonically with model size and data size).⁷³ These similarities suggest a tantalizing connection between the results here and the double-descent phenomenon. However, we have not observed a second descent in the context of basin prediction with NGRC (e.g., adding more data does not eventually stabilize NGRC). In the future, it would be interesting to explore different model sizes and look for double-descent curves as more features are included in NGRC or bigger reservoirs are used in RC.⁷⁴

SUPPLEMENTARY MATERIAL

See the [supplementary material](#) for details about the theory behind data-induced instability and additional simulations.

ACKNOWLEDGMENTS

We thank Dan Gauthier and David Wolpert for pointing out the importance of normalizing λ with data size and for providing references. We also thank Michelle Girvan, William Gilpin, Brian Hunt, Zachary Nicolaou, and Anastasia Bizyaeva for insightful discussions. Y.Z. acknowledges support from the Omidyar Fellowship and the National Science Foundation (NSF) under Grant No. DMS-2436231. E.R.S. acknowledges support from the National Institute on Alcohol Abuse and Alcoholism under Grant No. R01AA029926. S.P.C. acknowledges support from the Natural Science and Engineering Research Council (NSERC, RGPIN-2020-05015) and the Digital Research Alliance of Canada.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Y.Z. and S.P.C. designed the research and performed the initial study. E.R.S. developed the theory for data-induced instability in NGRC dynamics. H.Z. performed the numerical simulations of the forecasting task for the Lorenz system. All authors analyzed the data and wrote the paper.

Yuanzhao Zhang: Conceptualization (lead); Investigation (equal); Methodology (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal).

Edmilson Roque dos Santos: Formal analysis (lead); Investigation (equal); Methodology (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Huixin Zhang:** Investigation (equal); Software (equal); Validation (equal); Visualization (equal). **Sean P. Cornelius:** Conceptualization (lead); Investigation (equal); Methodology (equal); Software (lead); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data and code that support the findings of this study are available on GitHub at <https://github.com/spcornelius/RCBasins>, Ref. 75.

REFERENCES

- E. Weinan, “A proposal on machine learning via dynamical systems,” *Commun. Math. Stat.* **5**, 1–11 (2017).
- M. Levine and A. Stuart, “A framework for machine learning of model error in dynamical systems,” *Commun. Am. Math. Soc.* **2**, 283–344 (2022).
- N. Göring, F. Hess, M. Brenner, Z. Monfared, and D. Durstewitz, “Out-of-domain generalization in dynamical systems reconstruction,” [arXiv:2402.18377](https://arxiv.org/abs/2402.18377) (2024).
- W. Gilpin, “Generative learning for nonlinear dynamics,” *Nat. Rev. Phys.* **6**, 194–206 (2024).
- S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, “Scaling digital twins from the artisanal to the industrial,” *Nat. Comput. Sci.* **1**, 313–320 (2021).
- Z. Li *et al.*, “Fourier neural operator for parametric partial differential equations,” [arXiv:2010.08895](https://arxiv.org/abs/2010.08895) (2020).
- W. Gilpin, Y. Huang, and D. B. Forger, “Learning dynamics from large biological data sets: Machine learning meets systems biology,” *Curr. Opin. Syst. Biol.* **22**, 1–7 (2020).
- R. Delabays, G. De Pasquale, F. Dörfler *et al.*, “Hypergraph reconstruction from dynamics,” *Nat. Commun.* **16**, 2691 (2025).
- L.-W. Kong, H.-W. Fan, C. Grebogi, and Y.-C. Lai, “Machine learning prediction of critical transition and system collapse,” *Phys. Rev. Res.* **3**, 013090 (2021).
- Z. G. Nicolaou, G. Huo, Y. Chen, S. L. Brunton, and J. N. Kutz, “Data-driven discovery and extrapolation of parameterized pattern-forming dynamics,” *Phys. Rev. Res.* **5**, L042017 (2023).
- W. Gilpin, “Deep reconstruction of strange attractors from time series,” *NeurIPS* **33**, 204–216 (2020).
- H. Fan, L.-W. Kong, Y.-C. Lai, and X. Wang, “Anticipating synchronization with machine learning,” *Phys. Rev. Res.* **3**, 023237 (2021).
- W. Gilpin, “Model scale versus domain knowledge in statistical forecasting of chaotic systems,” *Phys. Rev. Res.* **5**, 043252 (2023).
- R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” *NeurIPS* **31**, 1 (2018).
- K. Azizzadenesheli *et al.*, “Neural operators for accelerating scientific simulations and design,” *Nat. Rev. Phys.* **6**, 320–328 (2024).
- J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach,” *Phys. Rev. Lett.* **120**, 024102 (2018).
- M. Schmidt and H. Lipson, “Distilling free-form natural laws from experimental data,” *Science* **324**, 81–85 (2009).
- S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proc. Natl. Acad. Sci. U.S.A.* **113**, 3932–3937 (2016).
- J. Pathak *et al.*, “Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model,” *Chaos* **28**, 041101 (2018).
- G. E. Karniadakis *et al.*, “Physics-informed machine learning,” *Nat. Rev. Phys.* **3**, 422–440 (2021).
- R. Chepuri, D. Amzalag, T. Antonsen, and M. Girvan, “Hybridizing traditional and next-generation reservoir computing to accurately and efficiently forecast dynamical systems,” *Chaos* **34**, 063114 (2024).
- M. Belkin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine-learning practice and the classical bias–variance trade-off,” *Proc. Natl. Acad. Sci. U.S.A.* **116**, 15849–15854 (2019).
- P. Nakkiran *et al.*, “Deep double descent: Where bigger models and more data hurt,” *J. Stat. Mech.* **2021**, 124003 (2021).
- D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, “Next generation reservoir computing,” *Nat. Commun.* **12**, 5564 (2021).
- E. Bollt, “On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD,” *Chaos* **31**, 013108 (2021).
- W. Maass, T. Natschläger, and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations,” *Neural Comput.* **14**, 2531–2560 (2002).
- H. Jaeger and H. Haas, “Harnessing nonlinearly in wireless communication,” *Science* **304**, 78–80 (2004).
- S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains* (John Wiley & Sons, 2013).
- L. Jaurigue and K. Lüdge, “Connecting reservoir computing with statistical forecasting and deep neural networks,” *Nat. Commun.* **13**, 227 (2022).
- D. J. Gauthier, I. Fischer, and A. Röhm, “Learning unseen coexisting attractors,” *Chaos* **32**, 113107 (2022).
- W. A. Barbosa and D. J. Gauthier, “Learning spatiotemporal chaos using next-generation reservoir computing,” *Chaos* **32**, 093137 (2022).
- R. M. Kent, W. A. Barbosa, and D. J. Gauthier, “Controlling chaotic maps using next-generation reservoir computing,” *Chaos* **34**, 023102 (2024).
- R. M. Kent, W. A. Barbosa, and D. J. Gauthier, “Controlling chaos using edge computing hardware,” *Nat. Commun.* **15**, 3886 (2024).
- Y. Zhang and S. P. Cornelius, “Catch-22s of reservoir computing,” *Phys. Rev. Res.* **5**, 033213 (2023).
- A. E. Motter, M. Gruiz, G. Károlyi, and T. Tél, “Doubly transient chaos: Generic form of chaos in autonomous dissipative systems,” *Phys. Rev. Lett.* **111**, 194101 (2013).
- D. A. Wiley, S. H. Strogatz, and M. Girvan, “The size of the sync basin,” *Chaos* **16**, 015103 (2006).
- R. Delabays, M. Tylø, and P. Jacquod, “The size of the sync basin revisited,” *Chaos* **27**, 103109 (2017).
- Y. Zhang and S. H. Strogatz, “Basins with tentacles,” *Phys. Rev. Lett.* **127**, 194101 (2021).
- Y. Zhang, P. S. Skardal, F. Battiston, G. Petri, and M. Lucas, “Deeper but smaller: Higher-order interactions increase linear stability but shrink basins,” [arXiv:2309.16581](https://arxiv.org/abs/2309.16581) (2023).
- D. Lowe and D. Broomhead, “Multivariable functional interpolation and adaptive networks,” *Complex Syst.* **2**, 321–355 (1988).
- Å. Björck and G. H. Golub, “Numerical methods for computing angles between linear subspaces,” *Math. Comput.* **27**, 579–594 (1973).
- M. Novaes, E. Roque dos Santos, and T. Pereira, “Recovering sparse networks: Basis adaptation and stability under extensions,” *Physica D* **424**, 132895 (2021).
- G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. (The Johns Hopkins University Press, 1996).
- A. Wikner *et al.*, “Stabilizing machine learning prediction of dynamics: Novel noise-inspired regularization tested with reservoir computing,” *Neural Networks* **170**, 94–110 (2024).
- W. N. van Wieringen, “Lecture notes on ridge regression,” [arXiv:1509.09169](https://arxiv.org/abs/1509.09169) (2015).
- J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, “Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data,” *Chaos* **27**, 121102 (2017).
- Z. Lu, B. R. Hunt, and E. Ott, “Attractor reconstruction by machine learning,” *Chaos* **28**, 061104 (2018).
- L. Grigoryeva and J.-P. Ortega, “Echo state networks are universal,” *Neural Networks* **108**, 495–508 (2018).
- T. L. Carroll and L. M. Pecora, “Network structure effects in reservoir computers,” *Chaos* **29**, 083130 (2019).

- ⁵⁰T. L. Carroll, “Do reservoir computers work best at the edge of chaos?” *Chaos* **30**, 121109 (2020).
- ⁵¹D. Patel, D. Canaday, M. Girvan, A. Pomerance, and E. Ott, “Using machine learning to predict statistical properties of non-stationary dynamical processes: System climate, regime transitions, and the effect of stochasticity,” *Chaos* **31**, 033149 (2021).
- ⁵²A. Wikner *et al.*, “Using data assimilation to train a hybrid forecast system that combines machine-learning and knowledge-based components,” *Chaos* **31**, 053114 (2021).
- ⁵³A. Röhm, D. J. Gauthier, and I. Fischer, “Model-free inference of unseen attractors: Reconstructing phase space features from a single noisy trajectory using reservoir computing,” *Chaos* **31**, 103127 (2021).
- ⁵⁴A. Flynn, V. A. Tsachouridis, and A. Amann, “Multifunctionality in a reservoir computer,” *Chaos* **31**, 013125 (2021).
- ⁵⁵W. A. Barbosa *et al.*, “Symmetry-aware reservoir computing,” *Phys. Rev. E* **104**, 045307 (2021).
- ⁵⁶J. Z. Kim, Z. Lu, E. Nozari, G. J. Pappas, and D. S. Bassett, “Teaching recurrent neural networks to infer global temporal structure from local examples,” *Nat. Mach. Intell.* **3**, 316–323 (2021).
- ⁵⁷G. A. Gottwald and S. Reich, “Combining machine learning and data assimilation to forecast dynamical systems from noisy partial observations,” *Chaos* **31**, 101103 (2021).
- ⁵⁸H. Ma, D. Prosperino, and C. R ath, “A novel approach to minimal reservoir computing,” *Sci. Rep.* **13**, 12970 (2023).
- ⁵⁹L. Gonon, L. Grigoryeva, and J.-P. Ortega, “Approximation bounds for random neural networks and reservoir systems,” *Ann. Appl. Probab.* **33**, 28–69 (2023).
- ⁶⁰L.-W. Kong, Y. Weng, B. Glaz, M. Haile, and Y.-C. Lai, “Reservoir computing as digital twins for nonlinear dynamical systems,” *Chaos* **33**, 033111 (2023).
- ⁶¹L.-W. Kong, G. A. Brewer, and Y.-C. Lai, “Reservoir-computing based associative memory and itinerancy for complex dynamical attractors,” *Nat. Commun.* **15**, 4840 (2024).
- ⁶²X. Li *et al.*, “Higher-order Granger reservoir computing: Simultaneously achieving scalable complex structures inference and accurate dynamics prediction,” *Nat. Commun.* **15**, 2506 (2024).
- ⁶³M. Yan *et al.*, “Emerging opportunities and challenges for the future of reservoir computing,” *Nat. Commun.* **15**, 2056 (2024).
- ⁶⁴X. Dutoit *et al.*, “Pruning and regularization in reservoir computing,” *Neurocomputing* **72**, 1534–1546 (2009).
- ⁶⁵L. Grigoryeva and J.-P. Ortega, “Differentiable reservoir computing,” *J. Mach. Learn. Res.* **20**, 1–62 (2019).
- ⁶⁶M. Belkin, “Fit without fear: Remarkable mathematical phenomena of deep learning through the prism of interpolation,” *Acta Numer.* **30**, 203–248 (2021).
- ⁶⁷P. Nakkiran, “More data can hurt for linear regression: sample-wise double descent,” [arXiv:1912.07242](https://arxiv.org/abs/1912.07242) (2019).
- ⁶⁸M. Belkin, D. Hsu, and J. Xu, “Two models of double descent for weak features,” *SIAM J. Math. Data Sci.* **2**, 1167–1180 (2020).
- ⁶⁹S. d’Ascoli, M. Refinetti, G. Biroli, and F. Krzakala, “Double trouble in double descent: Bias and variance (s) in the lazy regime,” in *ICML (PMLR, 2020)*, pp. 2280–2290.
- ⁷⁰S. Mei and A. Montanari, “The generalization error of random features regression: Precise asymptotics and the double descent curve,” *Commun. Pure Appl. Math.* **75**, 667–766 (2022).
- ⁷¹R. Schaeffer *et al.*, “Double descent demystified: Identifying, interpreting & ablating the sources of a deep learning puzzle,” [arXiv:2303.14151](https://arxiv.org/abs/2303.14151) (2023).
- ⁷²X. Davies, L. Langosco, and D. Krueger, “Unifying grokking and double descent,” [arXiv:2303.06173](https://arxiv.org/abs/2303.06173) (2023).
- ⁷³P. Nakkiran, P. Venkat, S. Kakade, and T. Ma, “Optimal regularization can mitigate double descent,” [arXiv:2003.01897](https://arxiv.org/abs/2003.01897) (2020).
- ⁷⁴A. H. Ribeiro, J. N. Hendriks, A. G. Wills, and T. B. Sch on, “Beyond Occam’s razor in system identification: Double-descent when modeling dynamics,” *IFAC-PapersOnLine* **54**, 97–102 (2021).
- ⁷⁵Github. <https://github.com/spcornelius/RCBasins>