

Improving the prediction stability of reservoir computing by means of dynamic structural bias

Jieyi Liu¹, Kaiwen Jiang¹, Jingfang Fan², and Yong Zou^{1,*}

¹*School of Physics, East China Normal University, Shanghai 200241, China*

²*School of Systems Science, Beijing Normal University, Beijing 100875, China*



(Received 15 November 2025; accepted 2 April 2026; published 4 May 2026)

Optimization of computational efficiency and improving predictive stability remain key challenges in reservoir computing (RC) research. Building on the principle of Poincaré recurrence theorem, we propose a dynamic structural bias RC algorithm, significantly improving the prediction stability. Our framework refines the conventional training process by introducing a two-step mechanism. First, a reference state library is constructed from the training time series. Second, this library is used in conjunction with a dot-product similarity calculation to generate a structural, dynamic bias term. This dynamic bias introduces high-fidelity historical information into the reservoir's state update equation. The benefit is that the bias in the prediction phase acts as a dynamic constraint guiding the reservoir evolution toward the true known trajectory. We prove that our algorithm enhances the model's ability to maintain generalized synchronization with the target system's attractor in the autonomous prediction phase. Consequently, our method achieves two key advantages: predictive stability is dramatically improved, and the model's sensitivity to network hyperparameters is reduced. We demonstrate the effectiveness of our method through the reconstruction and long-term prediction of various low-dimensional chaotic systems, showing a substantially increased prediction horizon and a more stable, faithful reproduction of the attractor's structural geometry.

DOI: [10.1103/5nbr-bqxb](https://doi.org/10.1103/5nbr-bqxb)

I. INTRODUCTION

Reservoir computing (RC) is a recurrent neural network model renowned for its effectiveness in processing temporal data, particularly in predicting time series arising from chaotic and complex dynamical systems [1–3]. At the algorithmic level, RC improvements have primarily focused on optimizing its structure, training methods, input/output processing, and feedback mechanisms to enhance its expressive power, stability, and generalization ability for time-series modeling. From early fully randomized connection matrices, RC has evolved to include various structured designs, such as small-world networks [4], brain networks [5], and parallel RC [6] based on real-world networks. These advances have substantially improved the network's information propagation and nonlinear memory capacity. The optimization of the output layer has been continually refined. Traditional ridge regression has been complemented by methods such as sparse regularization (L_1 norm) and linearized multinoise training to adapt to different application scenarios [7]. Furthermore, the introduction of higher-order reservoir matrices as concatenation terms [8] or the use of shallow neural networks as nonlinear readout layers [9] has also enhanced expressive power. By introducing

signal transformations [10] at the input layer, researchers have improved RC's data utilization and feature extraction capabilities. The introduction of feedback mechanisms [11], such as control parameters and data updates, has made multifunctional RC [12] and long-term stable prediction possible [13].

In terms of applications, RC has expanded from its early use in predicting chaotic systems [3] to areas such as climate modeling [14,15], speech recognition [16], financial analysis [17], and biological neural signal processing [5]. In addition, RC has been proposed to replace missing or unobservable nodes within complex systems while maintaining overall dynamic stability [12]. With the fusion of physical implementations and dynamic algorithms, RC is accelerating its transition into practical applications in fields such as engineering [18], communications [19], and neuromorphic computing.

Despite its advantages, conventional RC faces challenges that limit its practical application. Primarily, it is highly sensitive to hyperparameters, such as the spectral radius and the sparsity of the reservoir matrix. Their optimal selection is often a tedious process that relies on extensive empirical tuning rather than systematic methods. Second, due to the inherent randomness of its internal structure, RC's prediction performance can be unstable, with results varying widely across different realizations. Obtaining a reliable output often requires computationally expensive strategies, such as multiple repeated experiments or ensemble methods, posing a notable challenge for real-world deployment.

The sensitivity of RC to hyperparameters is due to its inherent random internal structure and the dynamically unconstrained nature in the autonomous prediction step, which differs from the constructive role of noise that has been used

*Contact author: yzou@phy.ecnu.edu.cn

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

in traditional machine learning algorithms to improve generalization ability [20]. For example, incorporating noise during the training phase is a widely used strategy to improve generalization, robustness, and stability of the model against input perturbations [21]. In some cases, adding noise to the input data is equivalent to incorporating a regularization term into the loss function, thus reducing overfitting [22]. Similar applications exist in RC, where studies have shown that adding noise to input data during training can lead to improved attractor reconstruction [20]. However, noise primarily enhances the model's generalization ability, making its performance more robust when faced with diverse input perturbations or tasks.

As a non-Markovian process, a reservoir's evolution is largely influenced by its historical memory, a capacity that is directly related to its predictive performance. To enhance this, several RC algorithms have been improved by leveraging past information through methods such as time-delayed input [7], virtual nodes [18], or convolutional layers. However, these approaches, while addressing memory capacity, have not successfully resolved the issues of hyperparameter sensitivity and result instability that continue to plague reservoir computing. This highlights the need for frameworks that can provide both enhanced performance and greater practical robustness.

In artificial neural networks, bias refers to a constant term added to a neuron's node, also known as node offset. Without a bias term, the function of a subsequent node would be severely constrained by its connected nodes. Bias provides the function with an "offset," moving it away from the origin and thus enhancing the model's ability to fit data. In the RC framework, bias manifests itself primarily at three levels: input layer, reservoir state updates, and output layer.

At the input layer, a common practice is to set up a parameter-aware channel. This involves adding a one-dimensional control parameter to the original N -dimensional input, resulting in an $(N + 1)$ -dimensional input. The choice of this control parameter is quite varied; it can be a constant, a specific parameter describing the dynamic system (such as a bifurcation parameter or coupling strength), or a part of the original data. The use of a special parameter as a bias in reservoirs has been applied to predict dynamic changes such as transient chaos [23] and amplitude death [24,25]. In contrast, methods that use a portion of the original data—such as a component from a specific dimension of the predicted data—as an input bias are more dynamic and possess a stronger long-term predictive capability [13,26–28]. A constant bias is more conducive to controlling the switch of the system between different datasets [12,29]. Recent research has shown that this method is more effective in allowing the model to distinguish between different systems in the phase space [30].

At the reservoir state update level, there are generally two types of bias that can be added to the update equation: a constant bias [31,32] and a special parameter bias [12]. The former is typically optimized as a hyperparameter. Using a system such as this, different nodes can produce diverse responses to the same input, enriching the dynamic characteristics of the reservoir. However, in existing RC research, the constant bias term in the state update equation is not always explicitly used to improve prediction performance. In the latter case, the special parameter bias is more often

treated as a tuning parameter, used to manually control the reservoir's operating state. For example, in multifunctional RC, different types of input data are often accompanied by different constant bias values to adjust the reservoir's dynamic response range [12].

In the training process, the regularization parameter β of the ridge regression is used in the least-squares objective, which prevents the output weights from becoming excessively large and therefore avoids overfitting. This regularization term enhances the model's stability and generalization performance under conditions of limited data.

In this work, we propose an approach to improve the performance of reservoir computing. Using a dot-product calculation, we combine the reservoir state containing past time-series information from the training process with the current reservoir state. This is then introduced into the computation as a dynamic structural bias for the model's evolution. This allows the fitted high-dimensional function to have richer expressive power within the same sample space, thereby reducing the reservoir's dependence on network structure, and thus on hyperparameters. In addition, in the autonomous prediction phase, this mechanism enhances the model's ability to maintain generalized synchronization with the target system's attractor, yielding stable predictions for the reservoir.

The structure of the paper is as follows. In Sec. II, we present our RC algorithm with a dynamic structural bias term, showing the collection of reference library from the training process and the way to include the maximum similarity in the update rule of the reservoir. In Sec. IV, we show that the numerical results demonstrate an increased prediction horizon and stable prediction when random hyperparameters are chosen. The mechanism for successful prediction from the generalized synchronization perspective is explained in Sec. V, which is followed by the conclusion in Sec. VIII.

II. DYNAMIC STRUCTURAL BIAS RESERVOIR COMPUTING

A. Traditional reservoir computing

Traditional RC consists of three layers: the input layer $\mathbf{W}_{\text{in}}$, the hidden layer of the reservoir $\mathbf{W}_r$, and the output layer $\mathbf{W}_{\text{out}}$, which is schematically shown in Figs. 1(a) and 1(b).

(1) The input layer maps an N -dimensional input signal $\mathbf{u}^t \in \mathbb{R}^N$ into an M -dimensional reservoir space by the weighted input matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{M \times N}$, where the elements of $\mathbf{W}_{\text{in}}$ are drawn randomly from $[-\sigma, \sigma]$, where $\sigma \in (0, 1)$ is a hyperparameter specifying the connectivity strength of a reservoir node to input.

(2) The hidden layer is a reservoir consisting of M network nodes that are connected by an $M \times M$ weighted adjacency matrix $\mathbf{W}_r$ with elements that are randomly chosen from a normal distribution. There are two hyperparameters that must be properly chosen for $\mathbf{W}_r$, namely, the average connectivity d and the spectral radius ρ . The matrix $\mathbf{W}_r$ is normalized by its largest absolute eigenvalue and subsequently rescaled to a target spectral radius, ensuring consistent reservoir dynamics across different realizations. The reservoir states are represented by the vector $\mathbf{r}^t \in \mathbb{R}^M$, that is, $\mathbf{r}^t = [r_1^t, r_2^t, \dots, r_M^t]^T$, where r_i^t represents the state of the i th node in the reservoir at

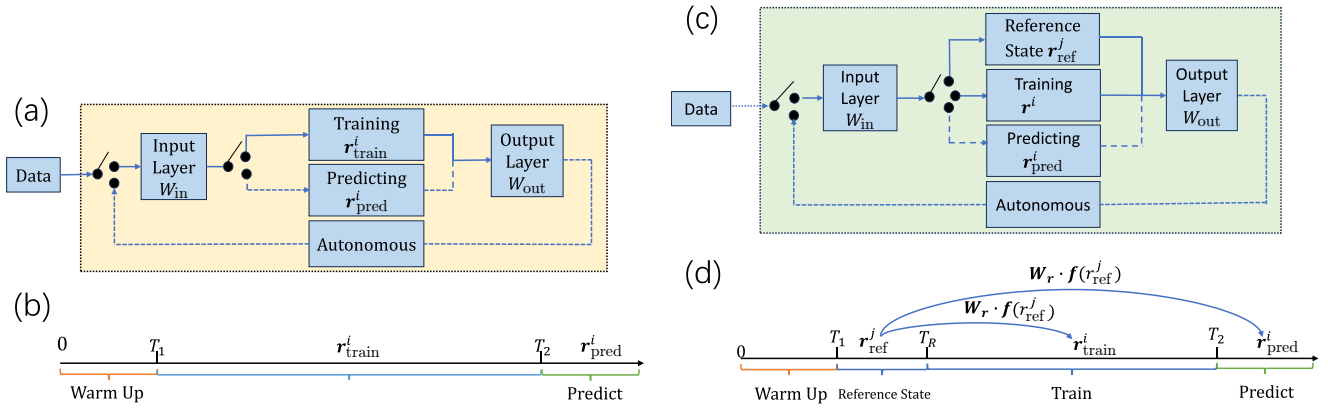


FIG. 1. Schematic illustration for reservoir computing: (a), (b) the traditional framework and (c), (d) the reservoir with dynamic structural bias $\mathbf{W}_r f(\mathbf{r}_{\text{ref}}^j)$, which is motivated by Poincaré recurrence. The period $[0, T_1]$ is used to discard reservoir's initial states, which are used for both TR-RC and DB-RC. In panels (c) and (d), reservoir states in period $[T_1, T_R]$ are used to construct reference library, which will be used in the training and prediction processes.

time t . The update rule of the reservoir is expressed by

$$\mathbf{r}^{t+1} = (1 - \alpha)\mathbf{r}^t + \alpha \tanh[\mathbf{W}_r \mathbf{r}^t + \mathbf{W}_{\text{in}} \mathbf{u}^t], \quad (1)$$

where $\alpha \in (0, 1)$ is a hyperparameter that controls the speed of updating.

(3) The output layer transforms the high-dimensional reservoir states $\mathbf{r}^t$ into the output signal $\mathbf{v}^t$ via $\mathbf{W}_{\text{out}}$ as

$$\mathbf{v}^t = \mathbf{W}_{\text{out}} \mathbf{r}^t. \quad (2)$$

The process of using a traditional RC for time-series prediction has three distinct stages [Figs. 1(a) and 1(b)]: warm-up, training, and prediction. (1) Warm-up $[0, T_1]$: This is the initial period in which the input data force the reservoir to discard its random initial state. (2) Training $[T_1, T_2]$: In this stage, the goal is to find optimal output weights $\mathbf{W}_{\text{out}}$ that map the high-dimensional states of the reservoir to the desired target output. (3) Prediction $[T_2, \infty)$: This stage uses the model's own output as feedback, replacing the input signal $\mathbf{u}^t$ and forecasting the time series autonomously.

When training the reservoir, the estimation of $\mathbf{W}_{\text{out}}$ is achieved by minimizing the loss function

$$L = \sum_{t=T_1}^{T_2} \|\mathbf{v}_g^t - \mathbf{W}_{\text{out}} \mathbf{r}^t\|^2 + \beta \|\mathbf{W}_{\text{out}}\|^2. \quad (3)$$

Note that data before $t < T_1$ are excluded from the loss function to eliminate the effects of initial conditions. Therefore, $\mathbf{W}_{\text{out}}$ is often obtained by ridge regression as follows:

$$\mathbf{W}_{\text{out}} = \mathbf{V}_g \mathbf{R}^T (\mathbf{R} \mathbf{R}^T + \beta \mathbb{I})^{-1}, \quad (4)$$

where $\mathbf{V}_g$ is the desired target signal, $\mathbf{R}$ is the reservoir states, $\mathbb{I}$ is the identity matrix, and β is the Tikhonov regularization parameter.

In the prediction phase $[T_2, \infty)$, the reservoir runs autonomously,

$$\mathbf{r}^{t+1} = (1 - \alpha)\mathbf{r}^t + \alpha \tanh[\mathbf{W}_r \mathbf{r}^t + \mathbf{W}_{\text{in}} \mathbf{v}_{\text{out}}^t], \quad (5)$$

$$\mathbf{v}_{\text{out}}^{t+1} = \mathbf{W}_{\text{out}} \mathbf{r}^{t+1}. \quad (6)$$

B. Dynamic biased reservoir computing

We divide the training into two parts, which is shown in Figs. 1(c) and 1(d). (1) In $[T_1, T_R]$, we collect reservoir states to form a reference matrix, that is, $\mathbf{R}_{\text{ref}} = [\mathbf{r}^{T_1}, \mathbf{r}^{T_1+1}, \dots, \mathbf{r}^{T_R-1}]$. There are $T_R - T_1$ reservoir states in $\mathbf{R}_{\text{ref}}$, which provides a reference window for the next steps. We find that it is sufficient for the reference window $T_R - T_1$ to cover several pronounced oscillations for a time series, which will be further validated by numerical results. (2) In the training phase $[T_R, T_2]$, we introduce the update rule with a dynamic bias term as follows:

$$\mathbf{r}^{t+1} = (1 - \alpha)\mathbf{r}^t + \alpha \tanh[\mathbf{W}_r \mathbf{r}^t + \mathbf{W}_{\text{in}} \mathbf{u}^t + \mathbf{W}_r f(\mathbf{r}_{\text{ref}}^j)], \quad (7)$$

$$f(\mathbf{r}_{\text{ref}}^j) = \mathbf{r}_{\text{ref}}^j - \mathbf{r}^t, \quad (8)$$

$$j = \operatorname{argmax}_j \langle (\mathbf{r}^t)^\top, \mathbf{R}_{\text{ref}} \rangle. \quad (9)$$

In Eq. (7), a dynamic bias term $\mathbf{W}_r f(\mathbf{r}_{\text{ref}}^j)$ compares the current state $\mathbf{r}^t$ with a reference state $\mathbf{r}_{\text{ref}}^j$ that shares the highest similarity in history $[T_1, T_R]$. More specifically, $f(\mathbf{r}_{\text{ref}}^j)$ is determined by the difference between the current state $\mathbf{r}^t$ and the most similar reference state $\mathbf{r}_{\text{ref}}^j$. The dot product $\langle \cdot \rangle$ in Eq. (9) is effective and useful in finding the highest similar state in the reference matrix. In Eq. (9), $(\mathbf{r}^t)^\top$ is the transpose of the reservoir states at time t . Since input dynamics is encoded by both the magnitude and orientation of the state vectors of a reservoir, normalization would discard critical amplitude information and diminish the model's feature selectivity. We show in the following that Eq. (9) prioritizes dynamical relevance over simple geometric alignment, maintaining that the full vector magnitude is necessary to capture the most representative historical states and ensure a robust structural bias.

Traditionally, a constant bias is often introduced to shift nonlinear activation, such as $\tanh(\mathbf{W}_r \mathbf{r} + \mathbf{W}_{\text{in}} \mathbf{u} + \mathbf{b})$, where $\mathbf{b}$ is a vector of M dimension whose values are chosen from $(-a, a)$. Here, we propose to use a structural and dynamic bias $\mathbf{W}_r f(\mathbf{r}_{\text{ref}}^j)$ that injects information about the true trajectory of

a reference library into the reservoir at each time step. This is equivalent to applying a nonlocal guiding force within the state space that pulls the current state toward the trajectory of the highest similarity in history, making it more closely approximate the real phase-space evolution. In addition, $\mathbf{W}_{\text{out}}$ in the output layer can still be explicitly obtained as Eq. (4).

In the prediction step $[T_2, \infty)$, the reservoir runs autonomously in a closed loop as follows:

$$\mathbf{r}^{t+1} = (1 - \alpha)\mathbf{r}^t + \alpha \tanh[\mathbf{W}_r \mathbf{r}^t + \mathbf{W}_{\text{in}} \mathbf{v}_{\text{out}}^t + \mathbf{W}_r f(\mathbf{r}_{\text{ref}}^j)], \quad (10)$$

$$f(\mathbf{r}_{\text{ref}}^j) = \mathbf{r}_{\text{ref}}^j - \mathbf{r}^t, \quad (11)$$

$$j = \operatorname{argmax}_j \langle (\mathbf{r}^t)', \mathbf{R}_{\text{ref}} \rangle, \quad (12)$$

$$\mathbf{v}_{\text{out}}^{t+1} = \mathbf{W}_{\text{out}} \mathbf{r}^t. \quad (13)$$

In RC, the reservoir state can be seen as the discretized result of a time-series trajectory in a high-dimensional space. According to the Poincaré recurrence theorem in dynamical systems, a time series embedded in a reconstructed phase space, any “state” (or pattern) that has occurred in the past will, with probability one, recur arbitrarily closely in the future. Therefore, recurrence represents “naturally occurrence analogous” in the history that has been used for weather forecasts [33,34]. Within the framework of the reference state library, the reservoir selects the historical state most similar to the current state by calculating the similarity between trajectories at different moments, which is achieved by the dot product. In addition, the difference between the current and the most similar state in history is used for subsequent computations. We will demonstrate how this method can effectively capture the dynamic features of the input data during model training and enhance the model’s predictive ability by introducing historical information from the reservoir.

Generally speaking, inductive bias is crucial to a model’s fitting capability. If the bias is too weak, it may lead to overfitting, weakening the model’s generalizability, and resulting in poorer performance on unseen data. Conversely, if the bias is too strong, the model may be faulty and perform poorly on both the training and the test data. To incorporate this bias, we can make specific structural adjustments to the model that go beyond simple training.

In the update rule [Eqs. (7) and (10)], the term $\mathbf{W}_r f(\mathbf{r}_{\text{ref}}^j)$ is a dynamic bias, which can be interpreted as a perturbation applied to the reservoir function, analogous to a standard bias term $\mathbf{b}$. However, a key distinction is that the bias term proposed here is dynamically computed from a previous state of the reservoir, which makes it time-varying. This dynamic bias incorporates temporal and structural information from the input data, facilitating more effective dynamic adjustments to the function’s evolution. Consequently, the mapping range of the function is tailored to cover the dataset’s distribution more precisely. In the next section, we will prove that the term $\mathbf{W}_r f(\mathbf{r}_{\text{ref}}^j)$ keeps the model evolving along the generalized synchronization manifold between the reservoir and the input signal in a stable manner.

The search for the highest similar state from the reference library plays an important role in our algorithm. We propose using a dot product for this purpose because of its high computational efficiency. Figure 2 shows the computation of the dot product [Eqs. (9) and (12)]. More specifically, the calculation of the dot-product similarity by Eqs. (9) and (12) only involves a single matrix-vector multiplication, with a complexity of approximately $O(M \times (T_R - T_1))$, where M is the reservoir dimension and $(T_R - T_1)$ is the number of reference states. This allows for the use of highly efficient vectorized operations.

In contrast, the L_2 norm similarity (Euclidean distance) has been widely used to explore recurrence in phase space, which requires more computation for each candidate state than a simple dot product. It also cannot be directly computed as a single matrix multiplication like the dot product, leading to higher memory access.

The current state $\mathbf{r}^t$ is ranked in descending order of similarity of the dot product to all reference states $\mathbf{r}_{\text{ref}}$. Based on this, we recommend using the maximum similarity as a criterion for bias selection, as this yields the maximum valid prediction time. In fact, subsequent discussions will demonstrate that uniformly selecting a fixed similarity rank (such as consistently choosing the second-largest or third-largest similarity value, etc.) for bias construction does not degrade the prediction performance.

Furthermore, if we were to not just select the particular similarity rank (i.e., the reference state with the highest dot product), but instead calculate the similarities of all reference states and perform a weighted average, we would first need a proper normalization as used in softmax, requiring a full-scale weighting operation at each time step. In a typical reservoir computing setup, this direct weighting would multiply the computation time per time step, which is detrimental to online prediction of long time series.

Therefore, after considering the trade-off between computational cost and accuracy, we select the single reference state with the largest dot product for bias calculation. This ensures that the introduction of reference information does not considerably increase the time complexity of the prediction process. In the following discussion, we will further analyze the impact of different state selection strategies, that is, the second or third highest dot product and the total length of the reference sequence on model performance.

C. Dynamic structural bias keeps the synchronization manifold stable

In Sec. IV F, we will numerically compare the advantage of using structural bias $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$ over a direct difference bias as in $\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t$. Here, we first show that $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$ helps the reservoir to evolve along the generalized synchronization manifold in a stable manner. For a well-trained reservoir, a generalized synchronization is achieved between the reservoir states and input signals.

Using a notation $\delta \mathbf{r}^t = \mathbf{r}_{\text{ref}}^j - \mathbf{r}^t$ as a small perturbation to the synchronization manifold, we rewrite the update rule of the reservoir in terms of the perturbation $\delta \mathbf{r}^t$ in the following [leave the subscripts of training and prediction from Eqs. (7)

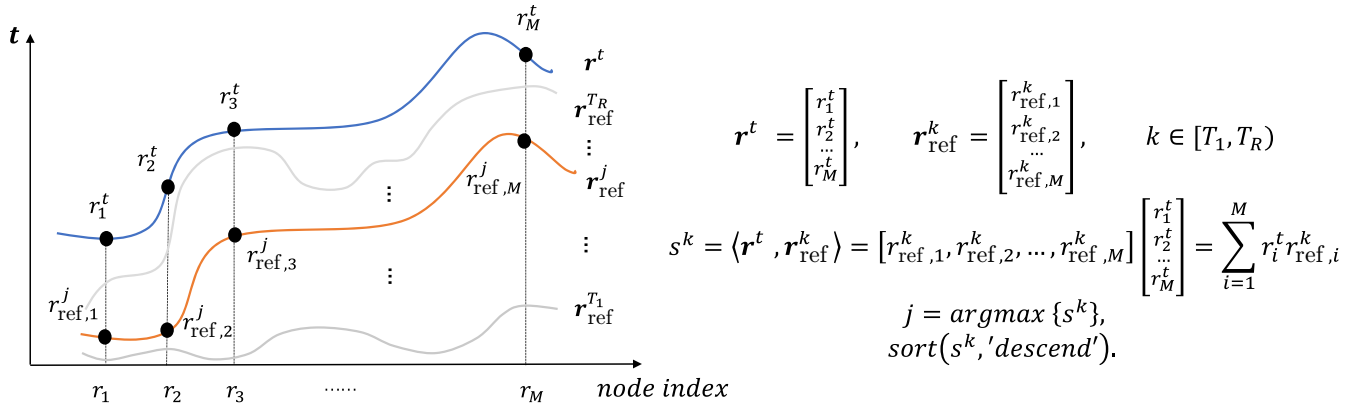


FIG. 2. Searching for the highest similarity state for the current state $\mathbf{r}^t$ from the reference library $\mathbf{R}_{\text{ref}}$ by dot product. The reference states are denoted as $\mathbf{r}_{\text{ref}}^k$, where $k \in [T_1, T_R)$. The similarity values are ranked in descending order and $\mathbf{r}_{\text{ref}}^j$ corresponds to the maximum similar state to $\mathbf{r}^t$.

and (10)]:

$$\mathbf{r}^{t+1} = (1 - \alpha)(\mathbf{r}_{\text{ref}}^j - \delta \mathbf{r}^t) + \alpha \tanh[\mathbf{W}_r \mathbf{r}_{\text{ref}}^j + \mathbf{W}_{\text{in}} \mathbf{u}^t + (\mathbf{G} - \mathbf{W}_r) \delta \mathbf{r}^t], \quad (14)$$

where $\mathbf{G}$ is a general notation: (1) $\mathbf{G} = \mathbf{I}$ means that the bias is determined by the difference between the reference state and the current state and (2) $\mathbf{G} = \mathbf{W}_r$ represents the structural bias.

For a small perturbation $\delta \mathbf{r}^t$, the Taylor expansion of $\tanh(\cdot)$ at $\mathbf{x}_0 = \mathbf{W}_r \mathbf{r}_{\text{ref}}^j + \mathbf{W}_{\text{in}} \mathbf{u}^t$ is

$$\begin{aligned} \tanh(\mathbf{x}_0 + (\mathbf{G} - \mathbf{W}_r) \delta \mathbf{r}^t) \\ \approx \tanh(\mathbf{x}_0) + \mathbf{J}_{\tanh}(\mathbf{x}_0)(\mathbf{G} - \mathbf{W}_r) \delta \mathbf{r}^t, \end{aligned} \quad (15)$$

where $\mathbf{J}_{\tanh}(\mathbf{x}_0^i) = \text{diag}[\text{sech}^2(\mathbf{x}_0^i)]$, $i = 1, \dots, n$.

We define $\delta \mathbf{r}^{t+1} = \mathbf{r}_{\text{ref}}^j - \mathbf{r}^{t+1}$ and then the perturbation at time $t + 1$ is

$$\delta \mathbf{r}^{t+1} = \alpha(\mathbf{r}_{\text{ref}}^j - \tanh(\mathbf{x}_0)) + \mathbf{M}_j^t \cdot \delta \mathbf{r}^t, \quad (16)$$

where $\mathbf{M}_j^t$ is given by

$$\mathbf{M}_j^t = (1 - \alpha)\mathbf{I} - \alpha \mathbf{J}_{\tanh}(\mathbf{x}_0) \cdot (\mathbf{G} - \mathbf{W}_r) \quad (17)$$

and $\mathbf{I}$ is the identity matrix. The small perturbation $\delta \mathbf{r}^{t+1}$ converges to zero if the eigenvalues of $\mathbf{M}_j^t$ are smaller than 1.

More specifically, in our approach using dynamic structural bias, that is, inserting $\mathbf{G} = \mathbf{W}_r$ into Eq. (16), the perturbation iterates

$$\delta \mathbf{r}^{t+1} = \alpha(\mathbf{r}_{\text{ref}}^j - \tanh(\mathbf{x}_0)) + (1 - \alpha) \cdot \delta \mathbf{r}^t. \quad (18)$$

Therefore, small perturbations in the reservoir state exhibit exponential decay, primarily driven by the contractive nature of the leaky integrator of $\alpha < 1$. Additionally, for this perturbation to converge to zero, the current state $\mathbf{r}^t$ should remain close enough to the most similar state $\mathbf{r}_{\text{ref}}^j$ from the

reference library, effectively satisfying the condition $\mathbf{r}_{\text{ref}}^j - \tanh(\mathbf{x}_0) \approx 0$. In the limit, this convergence of perturbation is equivalent to the system operating within the regime of generalized synchronization, where the reservoir dynamics is uniquely determined by the input-driven manifold.

The above convergence of a perturbation holds only for a fixed reference state $\mathbf{r}_{\text{ref}}^j$. We expand the above discussion to a more general case of a different reference state $\mathbf{r}_{\text{ref}}^k$ as the reservoir evolves and show that $\delta \mathbf{r}^{t+1}$ remains bounded. In addition, we show in Sec. V A that the selected reference state $\mathbf{r}_{\text{ref}}^j$ typically remains unchanged for a short period and only switches to a reference state $\mathbf{r}_{\text{ref}}^k$ at specific moments. In this case, the perturbation at time $t + 1$ is $\delta \mathbf{r}^{t+1} = \mathbf{r}_{\text{ref}}^k - \mathbf{r}^{t+1} = \mathbf{r}_{\text{ref}}^j - \mathbf{r}^{t+1} + \Delta \mathbf{r}_{\text{ref}}$, where $\Delta \mathbf{r}_{\text{ref}} = \mathbf{r}_{\text{ref}}^k - \mathbf{r}_{\text{ref}}^j$. Then, the perturbation of Eq. (16) is

$$\delta \mathbf{r}^{t+1} = \alpha(\mathbf{r}_{\text{ref}}^j - \tanh(\mathbf{x}_0)) + \mathbf{M}_j^t \cdot \delta \mathbf{r}^t + \Delta \mathbf{r}_{\text{ref}}. \quad (19)$$

In addition, Eq. (18) becomes the following:

$$\delta \mathbf{r}^{t+1} = \alpha(\mathbf{r}_{\text{ref}}^j - \tanh(\mathbf{x}_0)) + (1 - \alpha) \cdot \delta \mathbf{r}^t + \Delta \mathbf{r}_{\text{ref}}. \quad (20)$$

Due to $0 < \alpha < 1$, we further show that the perturbation [Eq. (20)] does not grow in time, since the contribution of $\Delta \mathbf{r}_{\text{ref}}$ to the perturbation remains bounded.

In summary, the choice of $\mathbf{G} = \mathbf{W}_r$ guarantees a contraction in the linear part of $\mathbf{M}_j^t$ [Eq. (17)]. The perturbation converges to zero if the same reference state is chosen for time t and $t + 1$, or it is restricted to a small neighborhood if different reference states are identified. In other words, the consistent topology between the bias term and the reservoir connectivity helps the reservoir to evolve close to the generalized synchronization manifold between the trained reservoir and input time series, leading to stable and long-term prediction. In contrast, direct use of difference $\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t$, that is, $\mathbf{G} = \mathbf{I}$, does not have this property.

D. Assessment of prediction skills

1. NRMSE

We use normalized root mean square errors (NRMSEs) to quantify the training quality of the reservoir. The NRMSE evaluates the relative predictive power of a model, allowing

for a fair comparison of model performance across different datasets or models. More specifically, NRMSEs focus on the error between the predicted trajectory $\mathbf{v}_{\text{pr}}$ and the true trajectory $\mathbf{v}_{\text{true}}$, which is calculated in the following:

$$\text{NRMSE} = \sqrt{\left\langle \frac{(\mathbf{v}^t - \mathbf{v}_{\text{true}}^t)^2}{\gamma^2} \right\rangle}, \quad (21)$$

where $\mathbf{v}^t$ is the predicted value at time t , $\mathbf{v}_{\text{true}}^t$ is the corresponding ground truth, γ is the standard deviation, and $\langle \cdot \rangle$ is the average over the time domain.

To quantify short-term prediction, we use Lyapunov times t_λ to evaluate the prediction horizon for chaotic time series, which provides us with a characteristic temporal scale for the reservoir. In particular, for each coordinate, we calculate that the maximal time NRMSE is below the threshold; that is, the valid predict time (VPT) is thus defined as

$$\text{VPT} = \frac{1}{\lambda} \arg\max(t | \text{NRMSE}_{t_i} < \varepsilon, \forall t < t_i), \quad (22)$$

where λ is the maximal Lyapunov exponent of the underlying system. The prediction fails when the NRMSE is larger than the error tolerance ε for the first time; that is, $\varepsilon = 0.12$ is chosen throughout. The prediction horizon $\mathbf{v}$ is then defined as the minimal time across all coordinates during which the NRMSE is below the threshold.

2. Dynamic time warping

NRMSE is often not suitable for time series showing spikes, for example, time series from the Hindmarsh-Rose (HR) neuron model. A minor timing or amplitude mismatch appears at the spike of sudden change, resulting in a large, localized error. In the case of prediction of spike time series, we use dynamic time warping (DTW) as a similarity measure with a high tolerance to spike variability in both magnitude and time [35]. DTW is an algorithm that measures the similarity between two time series by finding the optimal alignment between them. Unlike NRMSE, DTW is elastic, which means that it can stretch or compress sections of one time series to match the other, which makes it particularly effective for spike time series.

There are mainly three steps before defining the VPT based on DTW method [35]. (1) We need to build a grid matrix $\mathbf{D}$: The two time series $\mathbf{v}_{\text{pr}}$ and $\mathbf{v}_{\text{tr}}$ are arranged on $\mathbf{D}$. Each point D_{ij} represents a possible pairing between a point from the $\mathbf{v}_{\text{pr}}^i$ series and a point from $\mathbf{v}_{\text{tr}}^j$, for example, by $D_{ij} = \|\mathbf{v}_{\text{pr}}^i - \mathbf{v}_{\text{tr}}^j\|$, where $\|\cdot\|$ is the Euclidean distance. (2) We search for the best route $\mathbf{A}$: DTW algorithm then finds the path from start to finish through $\mathbf{D}$ that has the lowest total cost. This path can move horizontally, vertically, or diagonally, allowing it to “warp” the time axis to find the optimal match. (3) We compute the cumulative cost along this best path, that is, DTW error. A low score means that the time series are very similar, whereas a high score means that they are very different. In analogy to NRMSE, the VPT is then defined as

$$\text{VPT}_t = \frac{1}{\lambda} \arg\max \left(\min_{A \in A_{t_i, t_i}} \frac{\langle A, D \rangle}{t_i} < \varepsilon, \forall t < t_i \right). \quad (23)$$

Using a spike time series of the HR neuron model as an example, Fig. 3(a) shows the VPT defined by DTW error with the tolerance $\varepsilon = 0.12$. In this case, the cumulative DTW error suggests that the VPT is $8t_\lambda$. Note that the DTW error is computed along the best warping path, which is not exactly a linear progression in time. However, the NRMSE suggests a successful prediction at 2 Lyapunov time [Fig. 3(b)], although the time series show high similarity with the true series for much longer time.

III. MODELS FOR CHAOTIC TIME SERIES

We consider four representative chaotic systems, Lorenz [33], Rössler [36], Chen [37], and Hindmarsh-Rose neuron model [38].

$$\begin{aligned} \text{Lorenz:} & \begin{cases} \frac{dx}{dt} = 10(y - x), \\ \frac{dy}{dt} = -xz + 28z - y, \\ \frac{dz}{dt} = -xy - 8z/3, \end{cases} \\ \text{Rössler:} & \begin{cases} \frac{dx}{dt} = -y - z, \\ \frac{dy}{dt} = x + 0.2y, \\ \frac{dz}{dt} = 0.2 + z(x - 5.7), \end{cases} \end{aligned} \quad (24)$$

$$\text{Chen:} \begin{cases} \frac{dx}{dt} = 40(y - x), \\ \frac{dy}{dt} = -12x - xz + 28y, \\ \frac{dz}{dt} = xy - 3z. \end{cases}$$

$$\text{HR neuron:} \begin{cases} \frac{dx}{dt} = y - x^3 + 3x^2 - z + 3.1, \\ \frac{dy}{dt} = 1 - 5x^2 - y, \\ \frac{dz}{dt} = 0.06[4.2(x + 1.56) - z]. \end{cases} \quad (25)$$

The maximal Lyapunov exponents and Lyapunov time t_λ for the four systems are, respectively, summarized in Table I. All equations of the four systems are integrated by a fourth-order Runge-Kutta method with step size $h = 0.01$. The transient time T_1 is removed from initialization. Due to the different chaotic nature, we sample the time series for Rössler and HR, leading to a different sampling time dt . The length of the reference library and the training are summarized in Table I. We present the time axis in Lyapunov units in all relevant figures below to have a better consistency.

In the following, we choose the reservoir size $M = 200$ and the results do not change qualitatively for other values of M . There are five hyperparameters in our model, namely, the input scaling factor σ for the connection strength of the input matrix $\mathbf{W}_{\text{in}}$, the spectral radius ρ , and the average connectivity d of the reservoir matrix $\mathbf{W}_r$, the leakage rate of the reservoir α , and the Tikhonov regularization parameter β . Bayesian optimization is one of the most powerful and efficient methods for tuning hyperparameters with the least number of computationally expensive evaluations. Therefore, it has been widely suggested to find the optimal hyperparameters of RC since only five hyperparameters are involved in the training process. In our case, the hyperparameters of four chaotic systems are summarized in Table I, which are guided by Bayesian optimization.

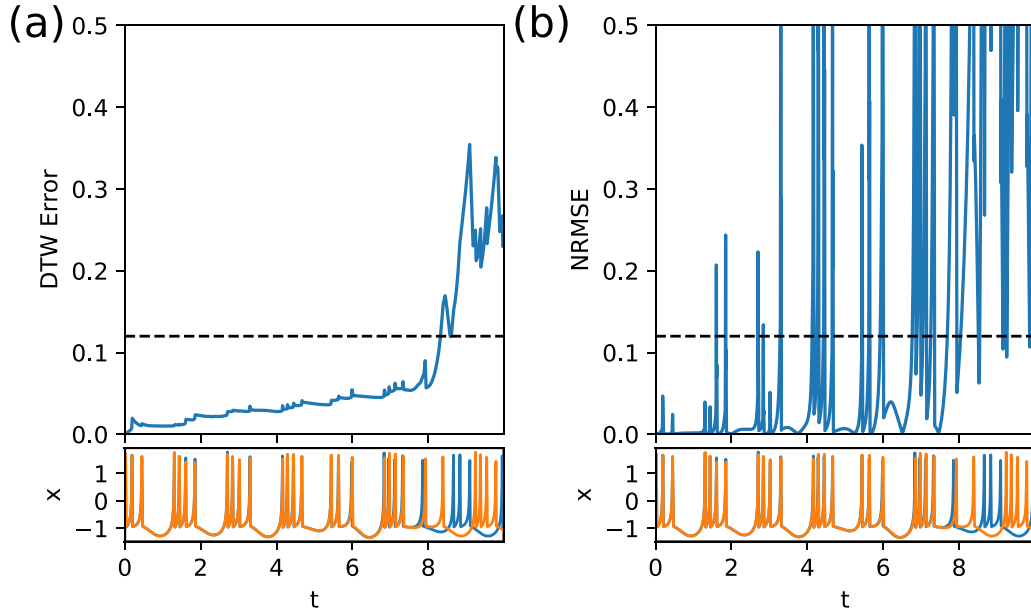


FIG. 3. Valid prediction time for spike time series using (a) DTW error and (b) NRMSE. In our numerical results below, we apply DTW to the HR.

IV. RESULTS

We compare the results of our dynamic structural bias RC (DB-RC) with those of traditional RC (TR-RC).

A. DB-RC improves the prediction horizon

Let us start by showing that DB-RC improves chaotic time-series prediction horizons. Figure 4 shows the comparisons between the predicted time series of the dynamical states and the actual ones. The vertical dashed lines represent the position of the VPT. The components of the variables predicted by the RC are plotted by blue lines, while the true time series is represented by black lines. In each subplot, the left column represents the prediction of the DB-RC, while the right column is obtained by the TR-RC. The prediction horizons by the DB-RC are 2–3 Lyapunov times longer than those of the TR-RC. More specifically, DB-RC (TR-RC) shows that the VPT in terms of Lyapunov time t_λ is 6–8 (5–6) for Lorenz, 8–9 (3–8) for Rössler, 6–10 (3–6) for Chen, and 7–15 (5–10) for the HR model.

B. DB-RC leads to stable predictions for optimal hyperparameters

Next, we assess the stability of the prediction power of a reservoir, which refers to its ability to consistently produce

accurate predictions across multiple runs, despite the random nature of its internal connections. This is crucial because high average performance can be misleading if the results of individual runs are highly variable. We run the same experiment multiple times (e.g., 5000 times in our work) with different random initializations of the reservoir connection matrices $\mathbf{W}_r$. For each run, we record the Lyapunov time of the predicted trajectory. Then, the shape of the distribution provides a visual representation of stability. A narrow single-peaked distribution centered around a large Lyapunov time is ideal. A wide or bimodal distribution (where performance clusters into “good” and “bad” runs) indicates low stability (for instance, a problem with the hyperparameter choice).

Figures 5(a)–5(d) show the frequency plot for all four chaotic systems. Out of 5000 realizations, we focus on the center of the prediction length in the frequency plot. In particular, Figs. 5(a)–5(d) show that the prediction length of the DB-RC is centered around $6 - 8t_\lambda$, while that of the TR-RC is around $5 - 6t_\lambda$ for the Lorenz system. Similarly, for the DB-RC method we get $8t_\lambda$ in the Rössler, $6 - 10t_\lambda$ in Chen, and $5 - 10t_\lambda$ in HR model. On the other hand, less prediction stability is observed for the TR-RC method, $3 - 8t_\lambda$ in Rössler, $0 - 8t_\lambda$ in Chen, and $0 - 5t_\lambda$ in the HR.

We define the success rate as the ratio of prediction horizons greater than $5t_\lambda$. More specifically, when hyperpa-

TABLE I. Maximal Lyapunov exponent λ and time t_λ , sampling time dt , reference length, training length, and choices for hyperparameters obtained by Bayesian optimization.

System	λ	t_λ	dt	Reference length	Training length	Input scaling σ	Spectral radius ρ	Average conn. d	Leak rate α	Tikhonov reg. β
Lorenz	0.91	1.01	0.01	500	4000	0.06	1.09	0.05	0.13	10^{-7}
Rössler	0.07	14	0.01	280	2800	0.10	1.20	0.01	0.10	10^{-8}
Chen	2.33	0.43	0.05	100	4000	0.11	0.08	0.14	0.21	10^{-7}
HR	0.01	88	0.10	2000	17 600	0.50	0.13	0.08	0.14	10^{-5}

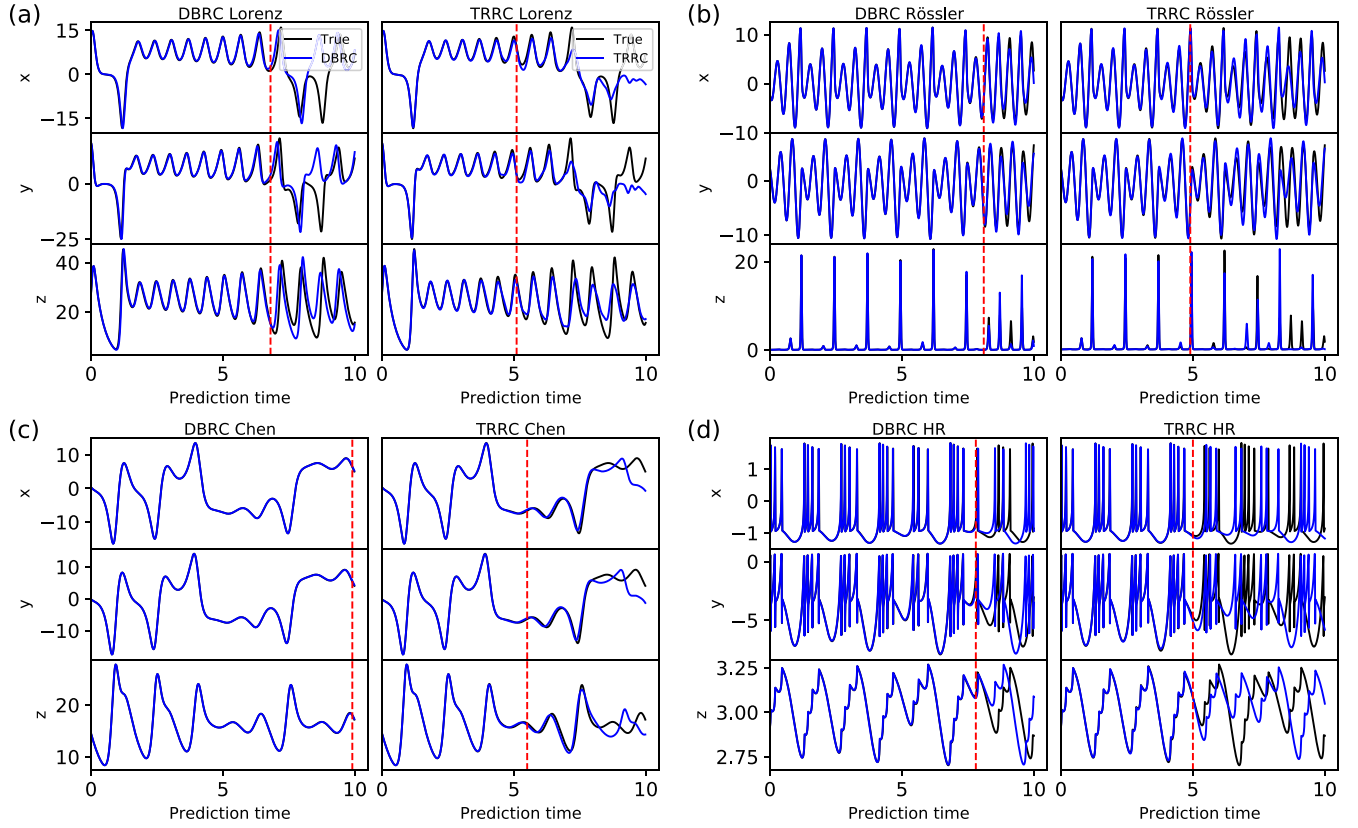


FIG. 4. Comparison of DB-RC and TR-RC in terms of the prediction Lyapunov time. The vertical dashed lines correspond to the valid prediction time when error is smaller than 0.2. (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR.

rameters are fixed, DB-RC (TR-RC) shows that stability in terms of success rate is 99.2% (94.4%) for Lorenz, 98.9% (47.2%) for Rössler, 92.2% (55%) for Chen, and 84.7% (74.4%) for the HR neuron model.

We further evaluate the stability of the reservoir by random choice for hyperparameters of spectral radius $\rho \in (0, 1)$ and

average connectivity density $d \in (0, 1)$, while other values of σ , α , and β are fixed as shown in Table I. Again, of 5000 realizations, Figs. 5(e)–5(h) show that the success rate of prediction horizons greater than $5t_\lambda$ for DB-RC (TR-RC) is 98.1% (30.4%) for Lorenz, 99.8% (5.3%) for Rössler, 82.5% (35.4%) for Chen, and 65.5% (26.9%) for HR.

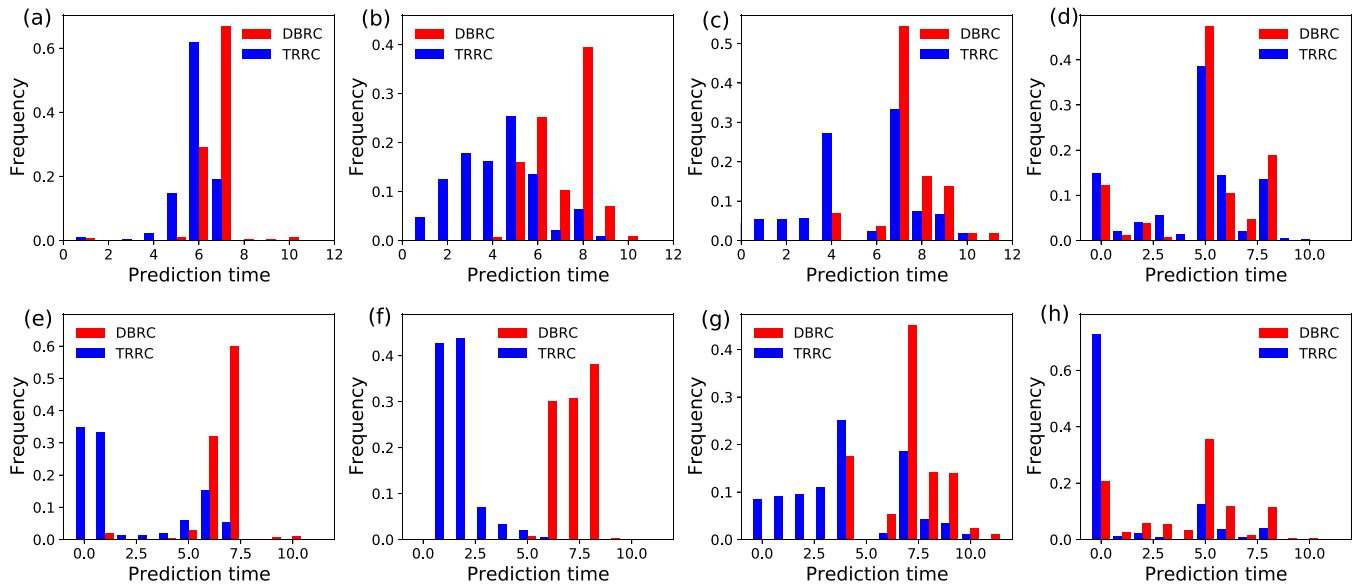


FIG. 5. Frequency plots for the prediction time of 5000 random realizations. (a)–(d) Hyperparameters are optimized with Bayesian method, while (e)–(h) hyperparameters are randomly chosen. (a), (e) Lorenz; (b), (f) Rössler; (c), (g) Chen; and (d), (h) HR.

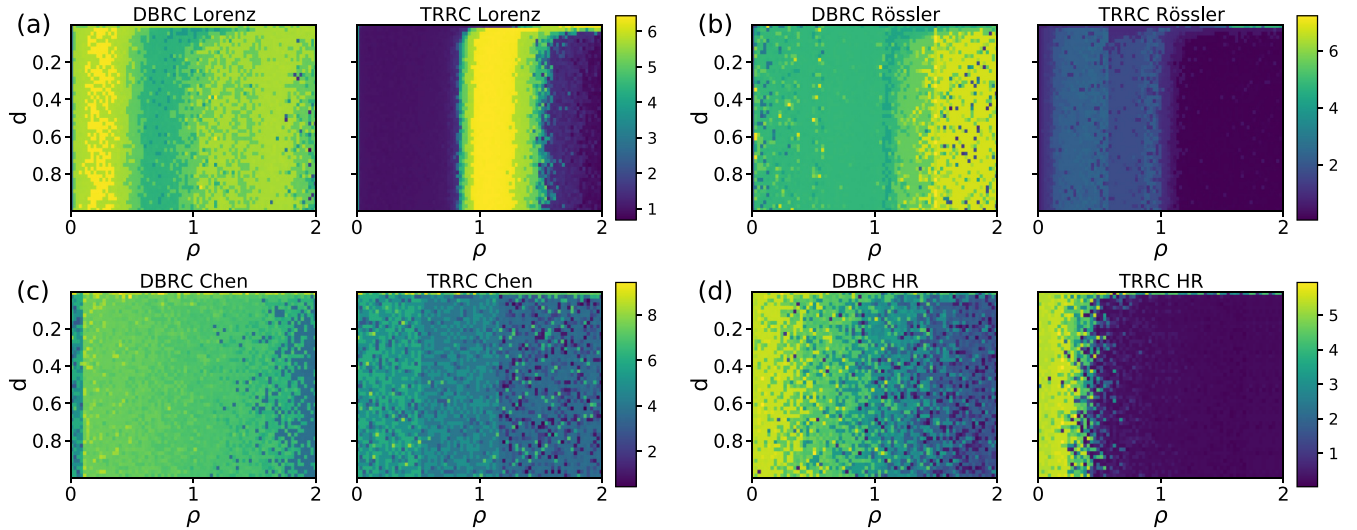


FIG. 6. The hyperparameter space (ρ, d) color coded by the predicted Lyapunov time t_λ . (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR neuron model.

Therefore, a truly stable DB-RC provides a consistent level of performance, which is a key requirement for real-world deployment.

C. DB-RC leads to flexible choices for hyperparameters

In addition to a random pair of hyperparameters ρ and d as above, here we search the entire parameter space $\rho \in (0, 2)$ and $d \in (0, 1)$ with step sizes $\Delta\rho = 0.02$ and $\Delta d = 0.02$. For each pair (ρ, d) , we run 500 random realizations and record the maximal prediction time. Figure 6 shows that the DB-RC method exhibits strong predictive capabilities over an exceptionally broad range for the parameter space, compared to the traditional RC.

For a TR-RC, when we search the space (ρ, d) , we expect to see the best prediction performance in a narrow and specific region of this space. In particular, in the low region of $\rho < 1$, the reservoir dynamics is stable and rapidly fades. The system has very short memory and cannot remember past inputs well, leading to poor prediction performance. On the other hand, in the high region of $\rho > 1$, the dynamics of the reservoir is unstable and often diverges to chaotic states. The internal reservoir states become dominated by their own dynamics rather than the input signal, leading to a loss of predictability. In the optimal region $\rho \approx 1$, the reservoir has a long, but fading, memory. It is rich enough to capture complex dynamics from the input signal without becoming unstable. Here, we find the best prediction accuracy.

For the average connectivity density d , we expect the following results. In the low region of d , the connections are too sparse for the input signal to propagate effectively through the reservoir. The internal dynamics are simplistic, limiting the reservoir's ability to create a diverse set of states, leading to poor prediction. On the other hand, in the high region of d , a highly connected reservoir is computationally expensive and can suffer from redundant connections that do not add information. The dynamics can become uniform and lose the complexity needed to separate input patterns, often leading

to performance similar to or worse than an optimal sparse network. Therefore, in the optimal d region, a moderate level of connectivity is required to ensure rich and complex internal dynamics. A highly sparse, yet well-connected reservoir network provides a good balance between network diversity and computational efficiency.

Due to the above considerations, we typically find a valley of optimal prediction performance where both ρ is close to 1 and d is at a moderate, optimal level, as shown by the traditional RC method in Fig. 6. However, this argument varies among the four systems under study. One property of traditional RC is that a good predictive horizon can be obtained in a narrow range for the hyperparameters.

In contrast, the DB-RC method leads to a much broader range of (ρ, d) showing a high predictive horizon.

D. The highest similarity rank yields the best prediction performance

In the proposed DB-RC algorithm, both the selection for the bias construction and the length of the reference state are critical factors that influence performance. The following discussion will be conducted under two conditions: a fixed set of reservoir hyperparameters and randomly chosen reservoir hyperparameters. Each single data point is calculated from 100 repeated realizations.

In the DB-RC algorithm, we first perform a dot product between the current reservoir state and the reference library to select the highest similar reference. Then, we use this reference state to interact with a network matrix, $\mathbf{W}_r$, to construct a bias. This bias method offers two advantages over the constant bias used in traditional reservoir computing: (1) It is dynamic and can change dynamically over time. (2) It is computationally efficient since it selects only the single reference state with the highest dot-product similarity.

Next, we choose a fixed similarity rank from the first to the tenth most similar reference states: We sequentially select the reference reservoir state with the k th largest dot-product

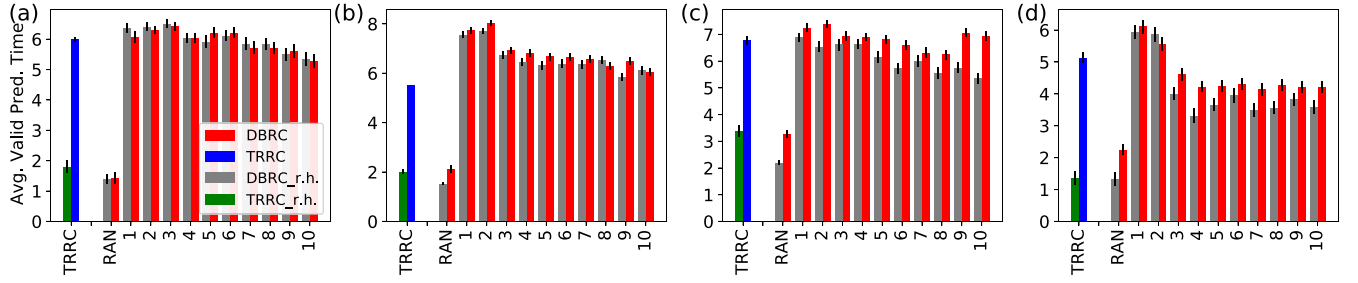


FIG. 7. Average valid prediction time when different similarity ranks are used. The highest similarity yields the best prediction performance. (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR.

similarity to the current reservoir state for bias construction. We show that the average valid prediction time decreases slightly when a reference state less similar to the current state is included. A random choice of a state from the reference library fails to achieve prediction performance.

In the traditional RC algorithm, we use a constant bias value that is often determined by Bayesian optimization. We compare the prediction performance under fixed reservoir hyperparameters with that under randomly selected hyperparameters of spectral radius ρ and connectivity d . All other experimental parameters are kept consistent between the different bias methods.

The computational results are shown in Fig. 7: (1) When hyperparameters (ρ, d) are fixed, both random bias and random choosing of a reference state lead to the worst prediction performance. The average valid prediction time reaches maximum at the highest similarity, which then shows a decreasing trend when the similarity ranks are changed to the tenth-largest similarity value. The highest and second-highest similar reference states yield improved prediction performance compared to the bias used in traditional RC.

(2) when (ρ, d) are randomly chosen, the prediction horizon of $6t_\lambda$ suggests that the stability of the performance is greatly improved when the highest and second-highest reference states are included in the bias. In contrast, the bias in the traditional RC shows only one t_λ , showing much less stability robustness to the random choice for (ρ, d) . In summary, the best performance appears when the highest similar reference state is included in the bias.

In general, as shown in Fig. 7, the average prediction time decreases when a less similar reference state is chosen. However, the general downward trend varies across the four systems. When the similarity rank varies from 1 to 20 (note that similarities are ranked in descending order), the effective prediction horizon by the DB-RC method for the four systems decreases by 12.28% (15.79%), 22.77% (31%), 18.76% (23.1%), and 27% (51.3%), respectively. The values in brackets are for randomly chosen hyperparameters.

Furthermore, using the results of the traditional RC under fixed hyperparameters as a benchmark, we summarize the threshold to select the similarity rank in the DB-RC method in Table II. When the similarity rank of the reference state is higher than this threshold (less similar reference states are chosen), the valid prediction horizon will fall below the benchmark set by the traditional RC.

E. Effects of the length of the reference state

Traditionally speaking, the requirement for the length of the training data is that it should be long enough to sufficiently sample the entire chaotic attractor in the phase space. The length is often measured not in data points but in terms of the characteristic timescales of the systems, such as the Lyapunov time t_λ . The necessary length depends heavily on the complexity of the chaotic system and the size of the reservoir. For most common low-dimensional chaotic systems (such as Lorenz or Rössler), this usually means that training data should span at least $10\text{--}50 t_\lambda$ [3].

Rather similarly, we find that the length of the data required for constructing the reference library also varies between different chaotic time series, as shown in Fig. 8. It is observed that the DB-RC algorithm achieves its best prediction performance for the Lorenz system when the reference length is constructed using data around $5t_\lambda$. For the Rössler system, the optimal length is $2t_\lambda$, the Chen system performs best using $4t_\lambda$, and the HR system achieves peak performance around $3t_\lambda$. In general, when the length of the reference state is within $5t_\lambda$, the average valid prediction time of the models shows an increasing trend with the length of the reference state, and a maximum superior to traditional RC appears within this range. When the reference is too short, the state information it contains may be insufficient to fully characterize the system's dynamic features, which limits the model's learning capability. As the length increases, the region of the phase space covered by the reference states gradually becomes richer, enabling the model to learn the system's characteristics more comprehensively, and thus the prediction performance improves.

However, once the length of the reference library exceeds $5t_\lambda$, the model performance begins to decline as the length continues to increase, although minor rebounds may still occur in localized ranges. While an excessively long reference state provides more state samples, the redundant or even noisy information it contains can interfere with the model's

TABLE II. The threshold for selecting the similarity rank in DB-RC method.

	Lorenz	Rössler	Chen	HR
Fixed hyperparameters	8	44	9	3
Random hyperparameters	7	38	5	3

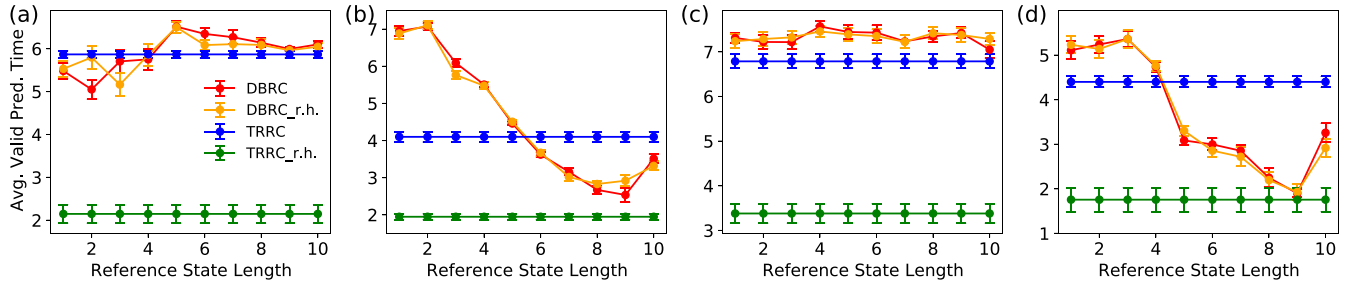


FIG. 8. Effects of the length of reference state on the average valid prediction time. The horizontal lines are benchmark results obtained by traditional RC. (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR.

extraction of key dynamic features, thereby weakening its predictive ability. From a computational cost perspective, once the length of the reference library exceeds $10t_\lambda$, the duration of a single computation becomes several times longer than that of the traditional algorithm. Therefore, we do not suggest choosing the reference state for more than $10t_\lambda$.

Around these optimal reference state lengths, the system's data points in phase space form a relatively complete attractor contour, which effectively captures the system's typical dynamic modes (such as the lobe switching of the Lorenz system or the spiking behavior of the HR system). This, in turn, provides sufficient informational support for both the construction of the bias term and the subsequent prediction.

Finally, the reference state length can also be treated as a hyperparameter. When independently optimizing the reference length using Bayesian optimization, its optimal value is often found within a relatively large interval [for example, when the search range is set to $(0, 100)$, it frequently converges around 90 Lyapunov time]. In many cases, such a large value is undesirable due to the computation limit. Therefore, we suggest choosing $5t_\lambda$ for the reference state length.

F. Dynamic structural bias outperforms other bias

Our DB-RC allows the reservoir to use historical reference states to timely reduce error accumulation, leading to stable prediction performance. In this section, we compare this bias setting with other choices. The bias term of our approach not only possesses time-varying characteristics, but also exhibits two notable differences in its construction: (1) the bias incorporates information from the difference between a historical reference state and the current state; (2) the bias interacts with the reservoir network $\mathbf{W}_r$, creating heterogeneous structural biasing for the reservoir node. More specifically, the bias is set as $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$, considering the difference between the current state and the highest similar state.

The following four bias construction methods are included for comparison purposes.

- (1) The highest similar reference state $\mathbf{r}_{\text{ref}}^j$.
- (2) The difference between the reference state and the current state $\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t$.
- (3) Structural bias using the highest similar reference state $\mathbf{W}_r \mathbf{r}_{\text{ref}}^j$.
- (4) Structural bias using the difference between the reference state and the current state $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$.

We explain the advantage of (4) over (2) in Sec. II C from the perspective of the stability of the generalized synchronization manifold between the reservoir and the target system.

Figure 9 shows that under fixed hyperparameters, the prediction performance of both bias models (i and ii) that do not incorporate the reservoir matrix $\mathbf{W}_r$ is inferior to the traditional algorithm. The differential bias model (ii) performs better than model (i) using only the reference state bias, indicating that the feedback information provided by the differential term positively contributes to improving prediction performance. After introducing the network matrix $\mathbf{W}_r$, the performance improves markedly. The structural bias model can slightly exceed the traditional RC algorithm, while the differential structural bias model further enhances the prediction accuracy on this basis. The results under random hyperparameter selections are largely consistent with those under fixed hyperparameters, further validating the aforementioned results.

Traditional RC shows poor prediction performance during the prediction phase, which is mainly due to two factors: error accumulation and insufficient expressive power. In the training phase, if a large error occurs in one step, it is corrected in a timely manner by the input data. However, in the prediction phase, any minor error will accumulate and amplify in subsequent computations due to the recursive nature, eventually causing the computation to diverge. Similarly, if the reservoir's fit to the system function is poor, it is also unable to make stable predictions.

In our DB-RC, the reference state presents a form of dynamic regularization. Instead of relying solely on data from the previous time step during iteration, it further searches for similar states from the historical reference trajectory $\mathbf{r}_{\text{ref}}$. Using a reference state as a bias introduces a timely perturbation to the reservoir evolution, allowing the system to stay closer to the known true trajectory and reducing error accumulation.

In this way, the prediction performance of the reservoir no longer depends solely on the evolution of the high-dimensional function learned during training. Instead, it is guided by a reference library built during the reference state construction phase. This allows the model to better fit the system's local structure and improves the stability of its predictions.

Furthermore, the performance of traditional RC is dependent on the structural setup of the reservoir, i.e., hyperparameters. The reservoir network generated by a selected set of parameters often corresponds to that system's dynamic characteristics, which makes it easier to simulate a function

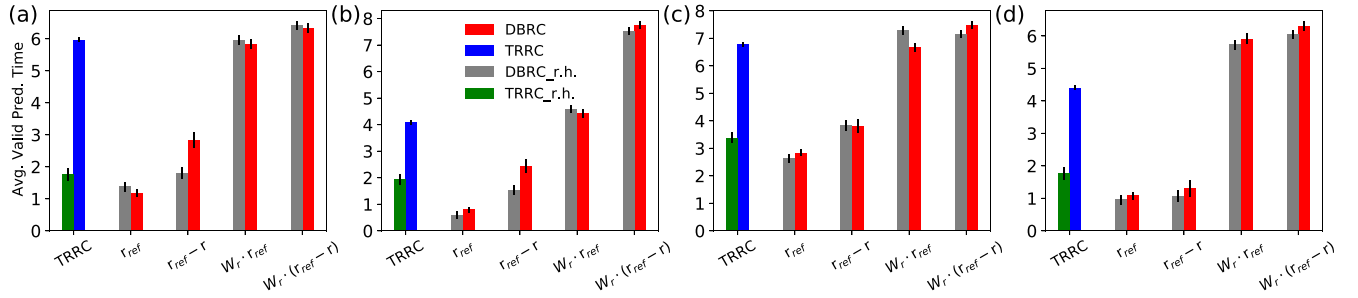


FIG. 9. Average valid prediction time versus four bias models (i–iv) and the constant bias in the traditional RC. In each bias model, the left bar corresponds to random hyperparameters of ρ , d and the right bar is for optimized hyperparameters. (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR.

that matches its dynamics. However, once the reservoir parameters and the temporal features of the input data change, the reservoir fitting ability decreases rapidly.

In contrast, our proposed DB-RC with historical references learns more than just a high-dimensional dynamical system function; it also learns a mapping from the input to a specific state structure. This means that even if the initial function fit is not good enough, the model can still be adjusted through a reference-state-driven computation, providing greater adaptability.

G. Dynamic bias yields long-term prediction stability

In this section, we utilize the fractional dimension D_2 to characterize the stability of the long-term prediction in reconstructing the overall geometry of the attractors.

The climate of an RC is often assessed by the length of the prediction horizon over short-term and long-term scales. The short-term prediction climate is characterized by high accuracy and low error accumulation, as we presented in the above sections. In this case, the prediction is considered quantitatively reliable for a finite period, i.e., the Lyapunov time, known as the prediction horizon.

The long-term prediction climate is usually characterized by the model's ability to maintain qualitative fidelity to the system's attractor [3]. In this case, the model's prediction no longer tracks the specific true trajectory. The prediction error accumulates exponentially because of the sensitive dependence on the initial conditions inherent to chaotic systems. However, for a well-trained RC, the predicted trajectory still remains confined to the correct phase-space geometry of the attractor. The RC fails in quantitative prediction, but succeeds in qualitative reconstruction. Therefore, it is suggested to combine the Lyapunov time and the fractional dimension to evaluate a well-trained reservoir [31,39,40].

We compute the correlation dimension D_2 using the Grassberger-Procaccia algorithm [41,42]. We assess qualitative success by comparing D_2 of the predicted time series with that of the original ground truth time series. We run 1000 independent realizations and the results are plotted in Fig. 10. Note that the D_2 values for the original systems are the following: Lorenz 2.05 ± 0.01 , Rössler 1.87 ± 0.02 , Chen 2.03 ± 0.01 , and HR 1.78 ± 0.02 [42]. Under fixed hyperparameters [Fig. 10(a)], the D_2 values of the reconstructed Lorenz attractors obtained by both algorithms are concentrated near the ground truth. However, under ran-

dom hyperparameters [Fig. 10(e)], the DB-RC is still able to maintain stable attractor reconstruction performance, with calculated D_2 values consistent with the true system. In contrast, traditional RC exhibits instability, showing a large number of instances where the prediction diverges $D_2 \approx 0$ or tends toward quasiperiodic dynamics $D_2 \approx 1$. Qualitatively similar results are obtained for other systems of Rössler, Chen, and HR. Therefore, we conclude that the RC with dynamic bias successfully reconstructs the topology of chaotic attractors, showing high stability and fidelity of an RC's long-term prediction climate.

V. MECHANISM OF DYNAMIC BIAS

Compared to traditional RC algorithms, the DB-RC proposed in this work does not introduce additional complexity at the network structure level. The reservoir topology, number of nodes, and nonlinear update rule remain unchanged. The improvement of the algorithm is realized by introducing reference state information during the state update process, which indirectly alters the evolution trajectory of the reservoir state in the high-dimensional phase space. The performance enhancement of our method does not involve structural expansion, but is derived from the optimization of state dynamics. The crucial property of our algorithm is that it enhances the reservoir's ability to maintain generalized synchronization with the target system's attractor in the autonomous prediction phase. In the following, we attempt to analyze the mechanisms of the DB-RC algorithm from the following two perspectives: (1) repeatedly chosen similar states of the reference library and (2) changes in the reservoir state.

A. Frequently used reference states capture the recurrences

The length of the reference library state is much shorter than that of the training period (summarized in Table I). Next, we demonstrate that the states within the reference library that are repeatedly identified by the reservoir as the most similar actually capture the recurrence properties of the system. To this end, we aggregate the reference states selected at each time step during the training phase and plot their distribution characteristics within the reference library.

Taking the Lorenz system as an example, the input time series of length 500 points (top panel in Fig. 11(a), $t \in [0, 5]$ and $dt = 0.01$, the initial transients are removed) is used to collect the reference library in the reservoir space. Then,

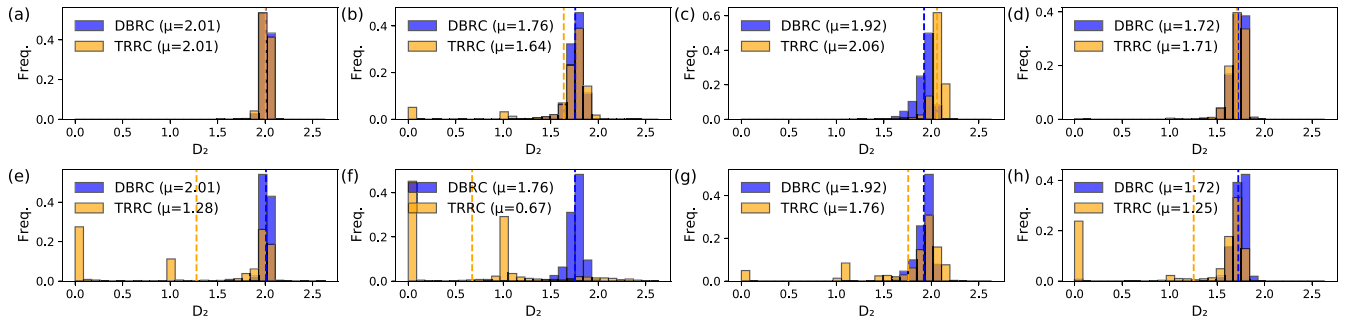


FIG. 10. Frequency plot of the correlation dimension D_2 estimated by an ensemble of 1000 realizations. Panels (a)–(d) are for fixed hyperparameters and panels (e)–(h) are for random hyperparameters. (a), (e) Lorenz; (b), (f) Rössler; (c), (g) Chen; and (d), (h) HR. The vertical dashed lines are the averaged values of the ensemble, which are indicated by the legends.

for each reference state, we count the frequency with which this reference state is identified as the most similar state in the following 4000 training steps. From the frequency distribution perspective [middle and bottom panels in Fig. 11(a)], the repeatedly selected reference states geometrically correspond to extreme state points of the input signal, reflecting the quasiperiodic recurrence characteristics of the

dynamical system. The multimodal frequency distribution reveals the ability of reference states to distinguish the structural features of the attractor. More specifically, when hyperparameters are fixed, the middle panel of Fig. 11(a) shows that the first five peaks in the frequency plot correspond to the quasiperiodic circling motion of the Lorenz system on a single lobe, reflecting short-term recurrence events of the attractor

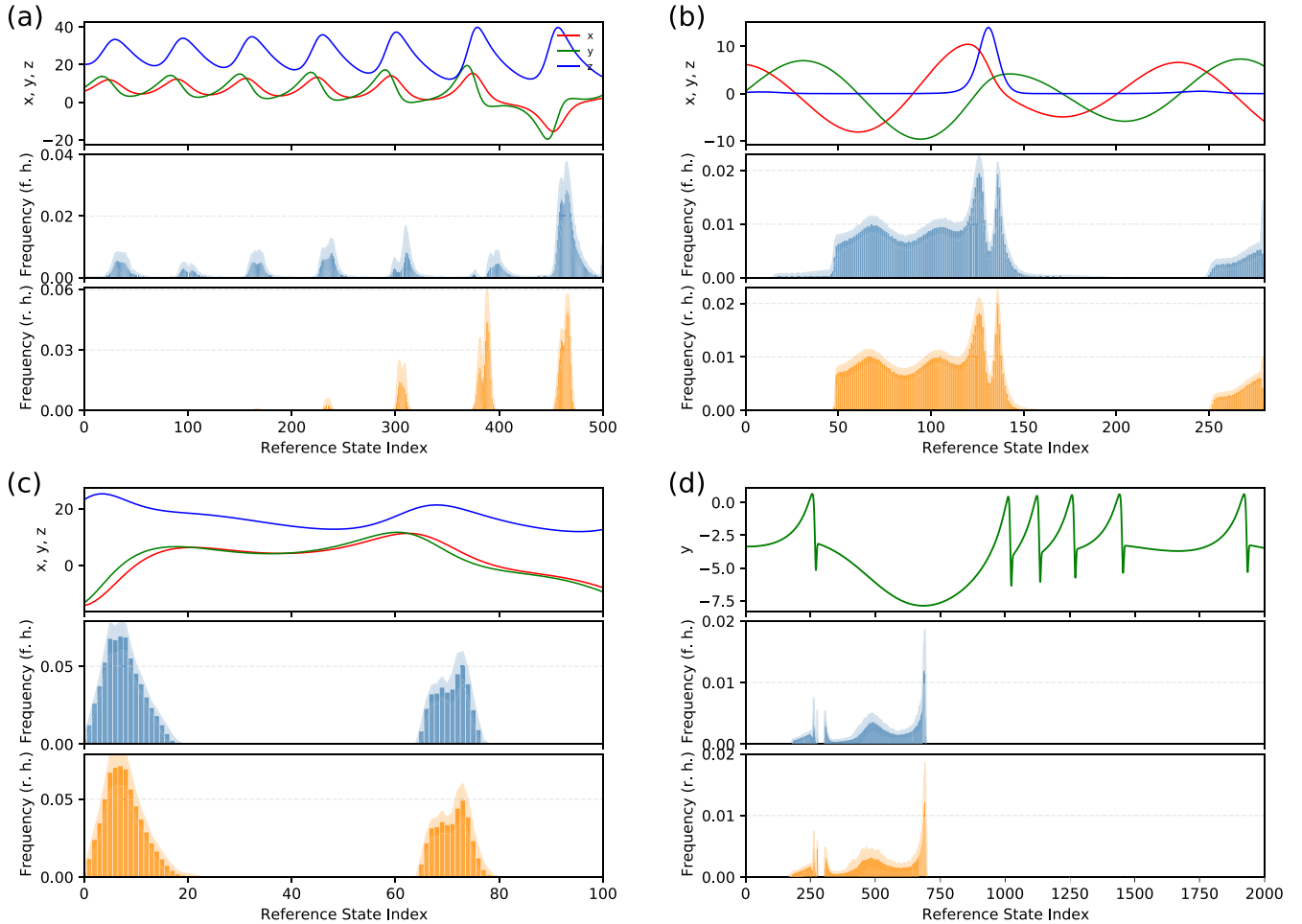


FIG. 11. In each subplot, top panel: the input time series for generating reference library; middle and bottom panels: frequency plot for each reference state and variance (middle panel is for fixed hyperparameters and bottom panel is for random hyperparameters). (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR.

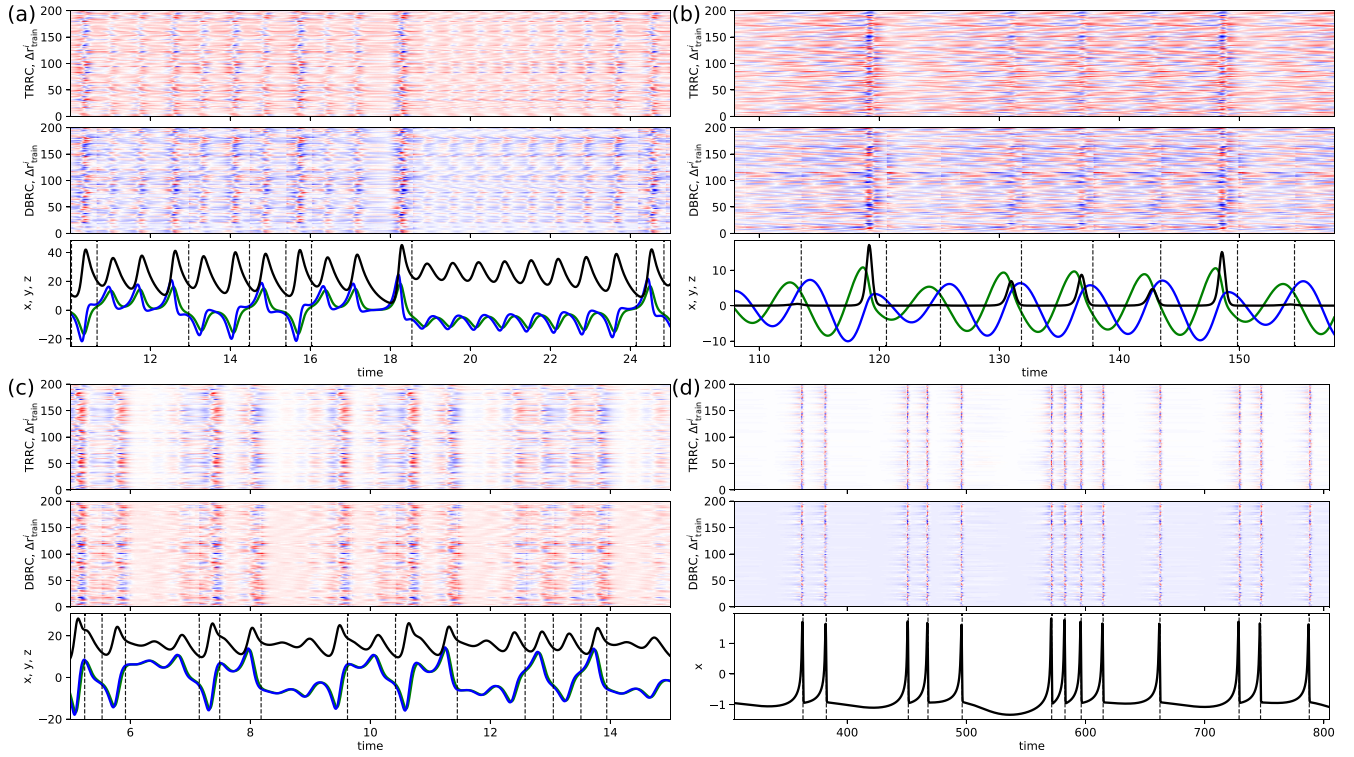


FIG. 12. Spatial-temporal plot of dynamic difference $\Delta \mathbf{r}^t = \mathbf{r}^t - \mathbf{r}^{t-1}$ for fixed hyperparameters in the training process. (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR. In each system, from the top to the bottom panel is, respectively, TR-RC, DB-RC, and the identified reference states of the highest similarity that are highlighted by vertical dashed lines superimposed on the training time series. The y axis in each plot is the index for each reservoir node.

within local regions. The last two peaks correspond to the transition process of the system between the two lobes, capturing recurrence characteristics when the trajectories pass near the central saddle point. When hyperparameters are randomly chosen, the last three peaks in the bottom panel of Fig. 11(a) show that the model is still able to capture the quasiperiodic recurrence properties within one lobe and the transition between two lobes.

In the Rössler system [Fig. 11(b)], the frequency plot of the reference states exhibits a clear bimodal structure. The first broader peak corresponds to the quasiperiodic motion before the system generates a spike, during which the attractor mainly evolves through rotational cycling in the x - y plane. The second narrower peak corresponds to the phase in which the system returns to the x - y plane after the termination of the spikes and resumes its quasiperiodic behavior. The distribution pattern remains largely consistent under random hyperparameters, indicating that the reference states exhibit a stable response to the recurrence patterns of the Rössler system.

In the Chen system [Fig. 11(c)], the reference state distribution also exhibits a bimodal characteristic. The two peaks correspond to the switching process of the attractor between lobes, reflecting the sensitivity of reference states to the system's transitions across different lobes. In the HR system [Fig. 11(d)], the peaks in the distribution correspond to regions where spike events occur, indicating that the reference states respond primarily to spike recurrences.

B. DB-RC captures changes of the reservoir states

Next, we focus on the dynamic difference between two consecutive time points of the reservoir states, i.e., $\Delta \mathbf{r}^t = \mathbf{r}^t - \mathbf{r}^{t-1}$ for TR-RC and DB-RC during the training process. In addition, we superimpose the dynamic evolution of $\Delta \mathbf{r}^t$ on the actual training time series, and the result is shown in Fig. 12.

When choosing fixed hyperparameters, the spatial-temporal plots for the reservoir states of Lorenz [Fig. 12(a)] show that noticeable changes in $\Delta \mathbf{r}^t$ fundamentally align with the lobe switching of the attractor. By introducing the reference state, the reservoir can thus better capture variations in the original dynamics, thus enhancing its predictive capability. The change in the reservoir state of the Rössler system [Fig. 12(b)] over time is similar to that of the Lorenz system. The DB-RC algorithm shows a noticeable enhanced response near the peaks of the z variable. Furthermore, a sudden change occurs near the $x = y$ plane, which corresponds to the spiking behavior in z . In the case of the Chen system [Fig. 12(c)], the change in the reservoir state dynamics primarily captures the lobe switching behavior. However, two types of lobe switching are captured: one is the switching near $x = y = 0$, which is similar to the Lorenz lobe switching and represents a smooth transition in the phase space; the other occurs near the maximum of x or y , where the data point enters one lobe and then performs a small loop before settling into the other. In the HR system [Fig. 12(d)], both algorithms show clear sudden changes at spikes. This is a particularly distinct dy-

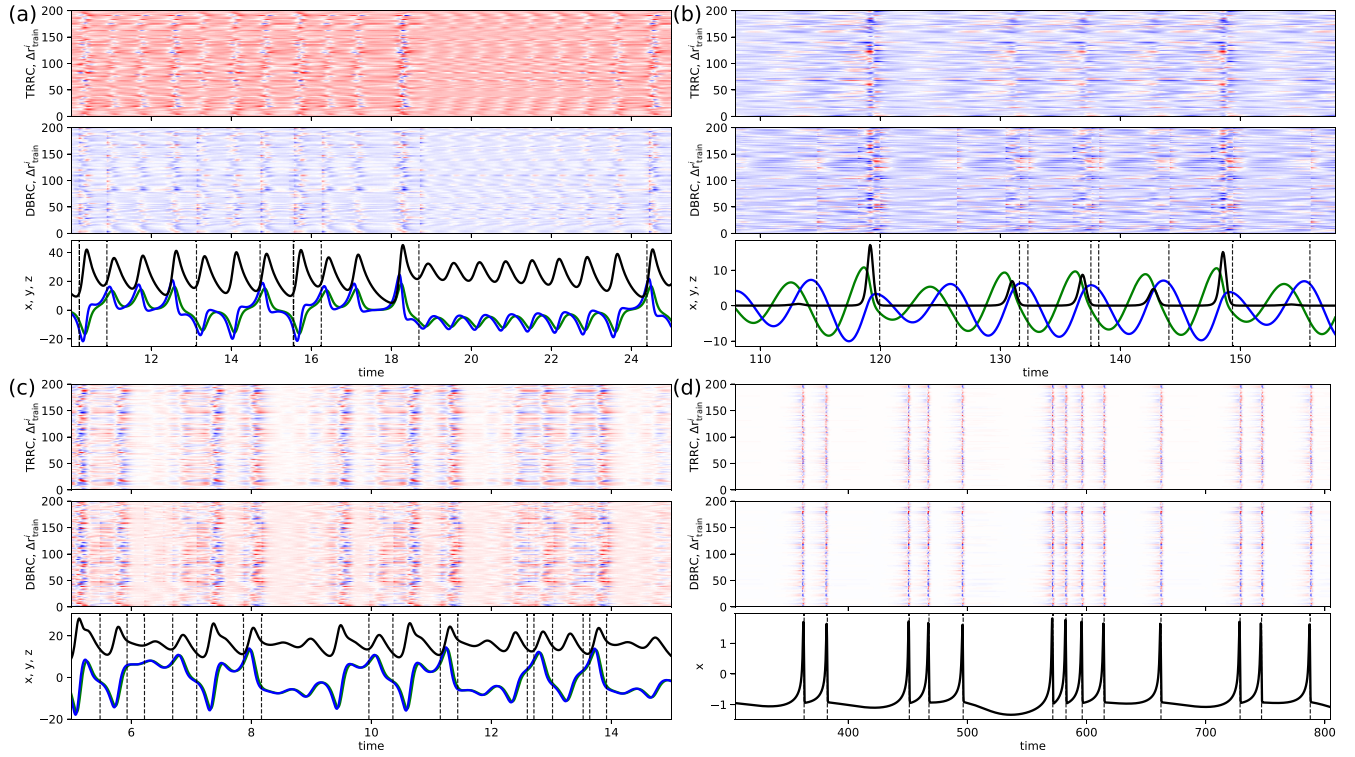


FIG. 13. Similar caption to Fig. 12 but for random hyperparameters ρ, d .

namic phenomenon of the HR system, which is also captured by the traditional RC. When the hyperparameters ρ and d are randomly chosen, the spatial-temporal plots for the dynamic difference $\Delta \mathbf{r}^t = \mathbf{r}^t - \mathbf{r}^{t-1}$ are shown in Fig. 13. In this case, the model's judgment of the location of dynamic change points is affected, but the DB-RC algorithm still shows relatively more distinct sudden changes. In Lorenz [Fig. 13(a)], the change points in the reservoir state are slightly shifted from the lobe switching locations. In Rössler [Fig. 13(b)], although more change points are identified, their locations are clustered around the original change points superimposed by the training signals. The Chen case [Fig. 13(c)] is similar to that of Rössler: More change points are found, but the DB-RC shows a more noticeable change near the location of $x = y = 0$ compared to the TR-RC model. In HR [Fig. 13(d)], both the DB-RC and TR-RC algorithms remain effective at capturing the system's spiking behavior. This is because the spiking characteristics are strong and easily identifiable.

VI. COMPARISON WITH OTHER METHODS

Our DB-RC algorithm hinges on the use of reference states to guide the reservoir toward the true attractor, which preserves a generalized synchronization between the well-trained reservoir states and the input time series. In general, the reservoir is forced by the true input signal during the training, and a generalized driver-response synchronization is achieved between the reservoir and the driving system. This training ensures that the network learns the function Φ that maps the input dynamics to the high-dimensional reservoir state. In the prediction phase, successful long-term prediction relies on the reservoir's ability to maintain this synchronous relationship when it becomes autonomous.

Several strategies are proposed to achieve this goal. Taking the three input variables ($x(t), y(t), z(t)$) of the Lorenz system as an example, we denote the state at time t of a trained reservoir as $(\tilde{x}(t), \tilde{y}(t), \tilde{z}(t))$. Instead of using $(\tilde{x}(t), \tilde{y}(t), \tilde{z}(t))$ to send it back to the reservoir to generate the next time point autonomously, we replace it with $(\tilde{x}(t), \tilde{y}(t), z(t))$ [26]. This allows for a long-term prediction for $(\tilde{x}(t), \tilde{y}(t))$ since the true variable $z(t)$ helps the reservoir evolve along the learned function Φ , which is further demonstrated in spatial-temporal chaotic systems [43,44]. Later, it is shown that the true variable $z(t)$ is necessary only when the accumulated prediction error reaches a threshold, i.e., longer than one Lyapunov time, leading to intermittent rare updates [13]. In these methods, the true input variable $z(t)$ enters the data input layer during the prediction process, forcing the reservoir to explicitly respond to the external input. To a large extent, this prediction phase is *not* fully autonomous due to the existence of the true variable $z(t)$. In addition, the true variable $z(t)$ is not available for real applications.

In our DB-RC approach, we choose a reference library from the training process. This setting is plausible in many applications, where it is unrealistic to obtain the true variable $z(t)$ to guide the reservoir. As we explained in Sec. IV F, the use of a reference state as a bias ultimately introduces a timely perturbation in the evolution of the reservoir, allowing the system to remain closer to the synchronous manifold of the true trajectory. Our dynamic bias $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$ from the historical reference library provides the reservoir with a weak recovery force, keeping the reservoir close to the reference manifold, rather than overwriting the natural evolution rules of the reservoir. The reference states introduce the restrictions to the attractor geometry during the evolution

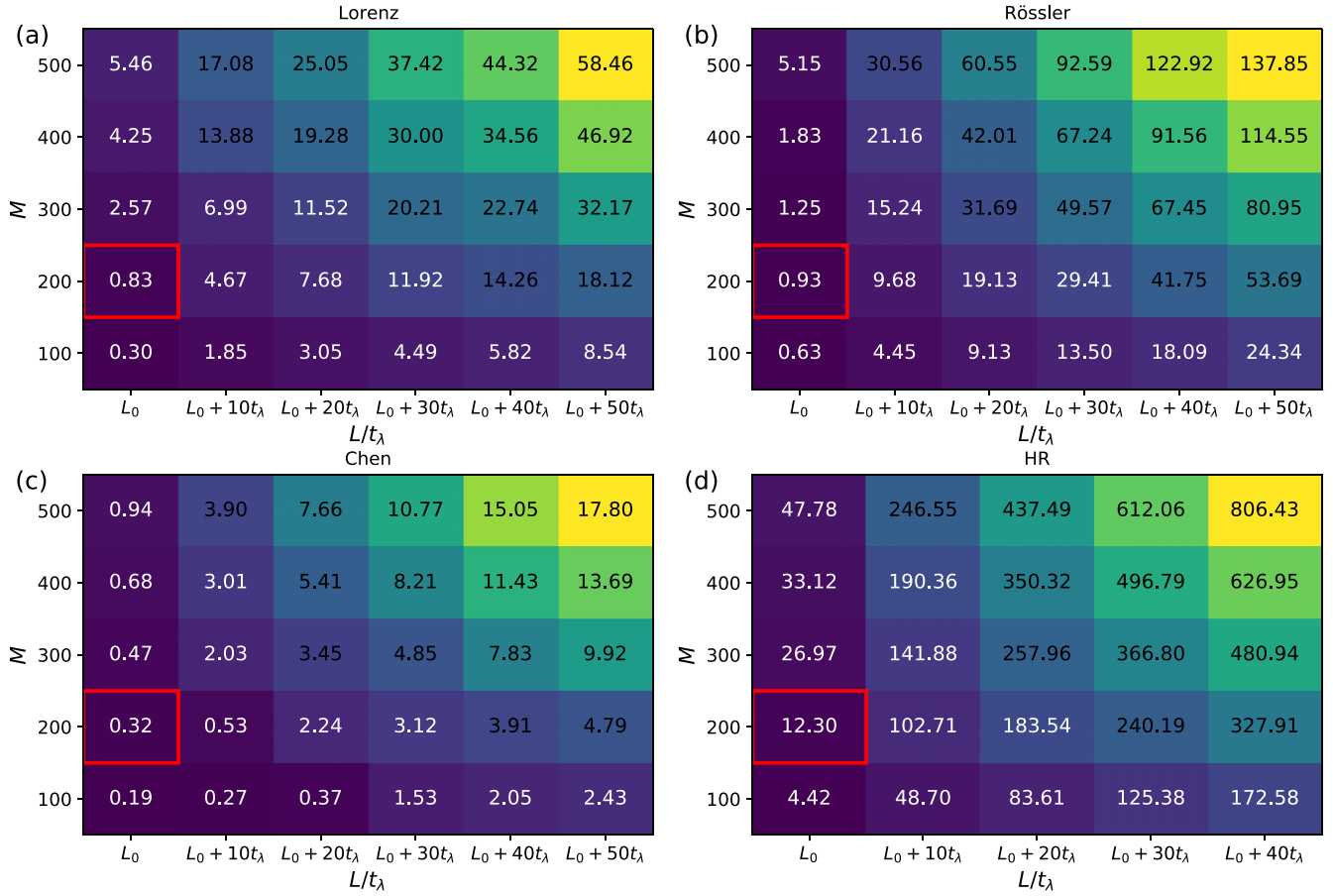


FIG. 14. Execution time for varying reservoir size M and length of reference library L (divided by t_λ). The red square boxes correspond to the cases that are reported in the main body of the paper. (a) Lorenz, (b) Rössler, (c) Chen, and (d) HR.

in the high-dimensional space $\mathbf{W}_r$. In the prediction phase, the reference library $\mathbf{r}_{\text{ref}}$ is no longer affected by the external input. Therefore, the reservoir operates autonomously, which is distinct from Refs. [13,26,43]. This also explains the prediction stability, showing less sensitivity to the random choices for hyperparameters.

VII. LIMITATIONS AND OUTLOOKS

In high-dimensional spatiotemporal chaotic systems, e.g., Kuramoto-Sivashinsky (KS), both TR-RC and DB-RC frameworks will face difficulties that are less pronounced in low-dimensional chaotic systems. These challenges primarily include high computational complexity, complex state space structure, and the lack of explicit dynamical priors.

A. Limitations

(1) For TR-RC, to adequately capture the typical dynamical behavior of the KS system, a parallelized reservoir configuration is often required. For example, when we implement 64 reservoirs in parallel, each reservoir typically requires $M > 5 \times 10^3$ and training data length often spans 70 000 steps [3]. Under such conditions, the computational cost of a single RC training run is high. Systematically scanning or optimizing hyperparameters ($\rho, \sigma, \alpha, \beta$) quickly

becomes prohibitively expensive. Moreover, even with the same set of hyperparameters, it is difficult to obtain highly stable prediction results across repeated experiments.

(2) DB-RC faces inherent challenges related to dimensionality and computational complexity. Its core mechanism is based on the selection of similarity from the reference states, where the dot product at each time step incurs a computational complexity of $O(M \times L)$, where M is the reservoir dimension and ($L = T_R - T_1$) is the number of reference states. Figure 14 illustrates the total execution time covering the warm-up, training, and prediction phases—averaged over ten runs in four typical systems as M and L vary. Other parameters are used consistently as in Fig. 8.

The general upward trend in computational time reflects the combined influence of M and L . Since Fig. 14 captures the entire workflow rather than isolated arithmetic steps, the scaling is not strictly linear. Actual execution times are further modulated by system-specific factors, including time-series length, training duration, readout regression overhead, and hardware-level memory caching. This explains the performance variations observed between different systems even under identical M and L configurations. Note that the computation time in Fig. 14 is measured in seconds and all calculations were performed on a PC running Windows 10 (64 bit). The system is equipped with an Intel CPU (Intel64

Family 6 Model 151 Stepping 2, GenuineIntel, with 24 logical cores), and GPU acceleration was not used. The code was executed with Python 3.11.7 (CPython).

In spatial-temporal chaotic systems, where large reservoir sizes are necessary to capture complex dynamics, the computational cost per step increases greatly. Furthermore, similarity measures in these high-dimensional spaces are susceptible to the “curse of dimensionality,” leading to reduced discriminative power in nearest neighbor searches. Consequently, this indicates that the direct extension of DB-RC to high-dimensional systems involves not only computational cost issues but also concerns regarding the effectiveness of similarity measures.

(3) In low-dimensional chaotic systems (such as the examples in this work), the attractor structure can be visualized through phase-space projections, and their spatial extents are relatively well defined, providing empirical guidance for determining the length and coverage of the reference state matrix. However, this approach scales poorly to high-dimensional or nonstationary processes. The lack of visualizability and the multiscale nature of spatiotemporal dynamics make predefining reference library and coverage more complex. Transitioning from empirical heuristics to adaptive data-driven construction methods for reference matrices is therefore essential to extending the DB-RC framework to high-dimensional real-world applications.

B. Possible solutions

(1) To overcome the curse of dimensionality in spatiotemporal systems, we propose integrating a low-dimensional latent space—similar to an autoencoder—to perform similarity-based reference library selection. By projecting reservoir states into a feature-rich, lower-dimensional manifold, we enhance the discriminative effectiveness of similarity measures while reducing computational overhead. In addition, future iterations will extend this by replacing instantaneous similarity with local dynamical consistency, matching states based on their short-term evolutionary trajectories. This trajectory-based approach more faithfully reflects the system’s phase-space manifold, offering a better criterion for high-dimensional chaotic attractor reconstruction.

(2) In our future work, we will construct an efficient reference library that depends on the targeted coverage of recurrent dynamical modes and coherent structures rather than the exhaustive sampling of the entire attractor manifold. As shown in Sec. V, the system evolution primarily activates a concentrated subset of reference states—specifically those governing critical transitions, such as lobe switching in the Lorenz system. This indicates that a reference library focused on recurrent, high-density regions of the manifold is sufficient to maintain structural bias without requiring uniform coverage of the entire state space.

Consequently, for complex spatiotemporal systems or real-world data, prior dynamical analysis can be leveraged to identify these characteristic structures and inform the composition of the reference library. For example, the chaotic orbits of the KS system are not entirely disordered but are organized around recurrent coherent structures that exhibit statistical regularity across spatial and temporal scales

[45–48]. These coherent structures recur repeatedly during system evolution, constituting a significant component of the system dynamics, and thus can serve as a key basis for reference state selection.

(3) From a complementary perspective, chaotic dynamics can be decomposed into ordered and disordered components using Koopman theory combined with delay embedding, framing the system as an intermittently forced linear system [49]. In diverse applications—including the Lorenz model, geomagnetic field reversals, and epidemic outbreaks—the forcing signals extracted via this method correlate strongly with macroscopic transients, such as lobe switching or sudden field reversals. Because these transients represent the system’s most significant transitions, prioritizing segments that capture these “forced” events offers a robust methodology for constructing the reference matrix in the DB-RC framework. This approach provides a systematic pathway for applying structural bias to complex, high-dimensional, and nonstationary datasets where traditional attractor visualization is unavailable.

VIII. CONCLUSIONS

We propose a dynamic structural bias reservoir computing (DB-RC) algorithm by introducing a structured bias term collected from historical information, which aims to improve the shortcomings of traditional RC, such as strong hyperparameter dependence and unstable prediction ability. The DB-RC incorporates a structural bias term $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$ guided by the reference states into the reservoir dynamics, which exerts a manifold constraint on autonomous prediction. This enables the reservoir to maintain the synchronization manifold with the target system’s attractor geometry, leading to stable predictions for random hyperparameters.

The core of our algorithm hinges on the use of dot-product similarity calculations in rapidly identifying the most similar historical information $\mathbf{r}_{\text{ref}}^j$ to the current state $\mathbf{r}^t$ of the reservoir from the reference library and subsequently guiding the reservoir’s evolution with this similar historical information. This process is motivated by the Poincaré recurrence theorem in dynamical systems. We show that frequently chosen reference states play a decisive role in the update of the reservoir, which have been demonstrated by paradigmatic low-dimensional chaotic systems. Consequently, the reservoir can maintain a structure similar to that of the target system more easily, mitigating issues such as divergence caused by random initialization and parameter uncertainty. Future work should focus on the challenges related to reservoir dimensionality and computational complexity. Especially, we need to construct a low-dimensional reference library for a spatial-temporal chaotic system, and an effective reference library for a nonstationary process where recurrence is weak.

To systematically evaluate the effectiveness of our algorithm, we assess the difference between DB-RC and the TR-RC from the following four perspectives: (1) the prediction horizon under single prediction conditions, (2) the same set of fixed hyperparameters under repeated experiments, (3) random choices for hyperparameters under repeated experiments, and (4) the phase diagram annotated by prediction accuracy. These results indicate that DB-RC

greatly extends the upper limit of predictable time, DB-RC demonstrates higher stability across the feasible hyperparameter range under various test conditions, and compared to traditional RC, DB-RC's dependence on hyperparameter selection is considerably reduced, offering greater freedom in parameter tuning.

From the viewpoint of algorithm, we further show the effects of the reference library from three aspects. (1) The similarity selection method: When the reference state with the maximum similarity is chosen, the DB-RC prediction performance is optimal. States with other similarity levels can also be used, but the similarity level should be kept unchanged during evolution to maintain geometry structure consistency. (2) The length of the reference states: Optimal prediction performance is achieved when the length of the reference matrix is about 5 Lyapunov times, which is sufficient to form an effective constraint to the synchronous manifold. (3) The form of the reference state: We show that the structural bias $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$ from the historical reference library performs better than other forms, such as directly with $\mathbf{r}_{\text{ref}}$ and the differential form $\mathbf{r}_{\text{ref}} - \mathbf{r}$. A well-trained reservoir achieves a generalized synchronization between the reservoir states and the input signal, which is kept stable when $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$ is used. Therefore, we conclude that the structural bias $\mathbf{W}_r(\mathbf{r}_{\text{ref}}^j - \mathbf{r}^t)$ effectively keeps the reservoir close to the target attractor. This property is further validated by the fractional dimension D_2 , which shows the long-term stability in the prediction climate. Further explanations from the point of conditional Lyapunov exponents [50] are required for future research.

A key advantage of the DB-RC approach is its structural noninvasiveness, which allows for the optimization of dynamical stability in physical reservoirs without requiring a redesign of the hardware substrate. Physical reservoir computing (PRC) uses the intrinsic dynamics of physical substrates to process information [51]. By restricting the training process to the readout layer, PRC significantly reduces algorithmic complexity and minimizes reliance on traditional digital computing resources, establishing itself as a vital branch of neuromorphic computing [19]. Typical architectures include delay-coupled systems [52,53], spatial multiplexing [54–56], and *in-materia* computing schemes [57]. Among these, delay-coupled RC is particularly effective [52]; through time multiplexing, it can generate high-dimensional virtual states using a minimal number of physical nodes. Although its feedback structure naturally encapsulates the historical evolution of the system, this information has traditionally been utilized uniformly through fixed delays, lacking a targeted selection mechanism.

The core mechanism of the DB-RC method lies in the conditional selection and reuse of historical information rather than structural modifications to the network. This provides a blueprint for implementing DB-RC on existing physical hardware: instead of reconstructing the hardware substrate, the required dynamic bias can be introduced by dynamically modulating existing control parameters—such as feedback gain or delay intervals. For instance, via history-dependent parameter tuning, the system can be steered to prioritize the reuse of virtual nodes representing typical dynamical modes,

potentially leading to more stable and representative evolutionary trajectories.

In software implementations, similarity-based reference search is often the primary source of computational overhead. However, as demonstrated in Sec. V, effective historical information is frequently concentrated within a sparse set of high-frequency reference states and characteristic dynamical modes. Based on this critical observation, future engineering implementations can employ targeted search strategies focused on these major modes, thus avoiding the need to traverse the entire historical archive. This optimization significantly reduces computational demand, providing a viable and efficient technical pathway for the deployment of DB-RC in high-performance physical and neuromorphic computing hardware.

Finally, a systematic study of the effects of noise on the DB-RC method will be an interesting topic for future work. There are generally two types of noise, dynamic and observational noise, which play different roles in RC. The DB-RC method does not explicitly inject dynamical or observational noise into the input signals. Instead, it imposes constraints on the internal state updates of the reservoir by a structural bias based on historical reference states. This bias acts directly on the dynamical evolution process of the reservoir, and its form is jointly determined by the intrinsic relationship between historical reference and current states. Thus, it is essentially a deterministic, history-dependent dynamical modulation, rather than a random disturbance to the observed data.

Functionally, this structural bias plays a constructive role similar to that of dynamical noise discussed in the literature [20,58]. Furthermore, through comparative experiments as shown in Fig. 7, we find that there are fundamental differences in performance between reservoir computing based on structural bias and that with injected white noise, demonstrating that structured dynamical modulation and unstructured random noise influence system dynamics differently. This finding aligns with the conclusions of existing studies on the differential effects of dynamical and observational noise [59]. We note that systematically investigating the response behavior and adaptive mechanisms of DB-RC under different types of noise—such as dynamical and observational noise—remains an important direction for future research.

ACKNOWLEDGMENTS

Parts of this work were financially supported by the National Natural Science Foundation of China (Grants No. 42461144209, No. T2525011, No. 42450183, No. 12275020, No. 12135003, and No. 12205025), the Natural Science Foundation of Shanghai (Grant No. 24ZR1420700), and the STI2030-Major Projects (Grant No. 2021ZD0202600). J.F.F. is supported by the Fundamental Research Funds for the Central Universities.

The authors declare that they have no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this article are openly available [60]; embargo periods may apply.

- [1] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, *Science* **304**, 78 (2004).
- [2] M. Yan, C. Huang, P. Bienstman, P. Tino, W. Lin, and J. Sun, Emerging opportunities and challenges for the future of reservoir computing, *Nat. Commun.* **15**, 2056 (2024).
- [3] 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).
- [4] Y. Kawai, J. Park, and M. Asada, A small-world topology enhances the echo state property and signal propagation in reservoir computing, *Neural Networks* **112**, 15 (2019).
- [5] L. E. Suárez, A. Mihalik, F. Milisav, K. Marshall, M. Li, P. E. Vértés, G. Lajoie, and B. Misić, Connectome-based reservoir computing with the `conn2res` toolbox, *Nat. Commun.* **15**, 656 (2024).
- [6] K. Srinivasan, N. Coble, J. Hamlin, T. Antonsen, E. Ott, and M. Girvan, Parallel machine learning for forecasting the dynamics of complex networks, *Phys. Rev. Lett.* **128**, 164101 (2022).
- [7] A. Wikner, J. Harvey, M. Girvan, B. R. Hunt, A. Pomerance, T. Antonsen, and E. Ott, Stabilizing machine learning prediction of dynamics: Novel noise-inspired regularization tested with reservoir computing, *Neural Networks* **170**, 94 (2024).
- [8] H.-H. Ren, Y.-L. Bai, M.-H. Fan, L. Ding, X.-X. Yue, and Q.-H. Yu, Constructing polynomial libraries for reservoir computing in nonlinear dynamical system forecasting, *Phys. Rev. E* **109**, 024227 (2024).
- [9] F. Köster, K. Kanno, J. Ohkubo, and A. Uchida, Attention-enhanced reservoir computing, *Phys. Rev. Appl.* **22**, 014039 (2024).
- [10] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, Next generation reservoir computing, *Nat. Commun.* **12**, 5564 (2021).
- [11] S. K. Tavakoli and A. Longtin, Boosting reservoir computer performance with multiple delays, *Phys. Rev. E* **109**, 054203 (2024).
- [12] Y. Du, H. Luo, J. Guo, J. Xiao, Y. Yu, and X. Wang, Multifunctional reservoir computing, *Phys. Rev. E* **111**, 035303 (2025).
- [13] H. Fan, J. Jiang, C. Zhang, X. Wang, and Y.-C. Lai, Long-term prediction of chaotic systems with machine learning, *Phys. Rev. Res.* **2**, 012080 (2020).
- [14] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, A machine learning-based global atmospheric forecast model, *Geophys. Res. Lett.* **47**, e2020GL087776 (2020).
- [15] Y.-G. Ham, J.-H. Kim, and J.-J. Luo, Deep learning for multi-year enso forecasts, *Nature (London)* **573**, 568 (2019).
- [16] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification, *Phys. Rev. X* **7**, 011015 (2017).
- [17] W.-J. Wang, Y. Tang, J. Xiong, and Y.-C. Zhang, Stock market index prediction based on reservoir computing models, *Expert Syst. Appl.* **178**, 115022 (2021).
- [18] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, Experimental demonstration of reservoir computing on a silicon photonics chip, *Nat. Commun.* **5**, 3541 (2014).
- [19] X. Liang, J. Tang, Y. Zhong, B. Gao, H. Qian, and H. Wu, Physical reservoir computing with emerging electronics, *Nat. Electron.* **7**, 193 (2024).
- [20] I. Estébanez, I. Fischer, and M. C. Soriano, Constructive role of noise for high-quality replication of chaotic attractor dynamics using a hardware-based reservoir computer, *Phys. Rev. Appl.* **12**, 034058 (2019).
- [21] R. M. Zur, Y. Jiang, L. L. Pesce, and K. Drukker, Noise injection for training artificial neural networks: A comparison with weight decay and early stopping, *Med. Phys.* **36**, 4810 (2009).
- [22] C. M. Bishop, Training with noise is equivalent to tikhonov regularization, *Neural Comput.* **7**, 108 (1995).
- [23] 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).
- [24] R. Xiao, L.-W. Kong, Z.-K. Sun, and Y.-C. Lai, Predicting amplitude death with machine learning, *Phys. Rev. E* **104**, 014205 (2021).
- [25] H. Fan, L. Wang, Y. Du, Y. Wang, J. Xiao, and X. Wang, Learning the dynamics of coupled oscillators from transients, *Phys. Rev. Res.* **4**, 013137 (2022).
- [26] T. Weng, H. Yang, C. Gu, J. Zhang, and M. Small, Synchronization of chaotic systems and their machine-learning models, *Phys. Rev. E* **99**, 042203 (2019).
- [27] X. Chen, T. Weng, C. Gu, and H. Yang, Synchronizing hyperchaotic subsystems with a single variable: A reservoir computing approach, *Physica A* **534**, 122273 (2019).
- [28] L. Zhao, Z. Sun, M. Tang, S. Guan, and Y. Zou, Learning successive weak synchronization transitions and coupling directions by reservoir computing, *Chaos Solitons Fractals* **168**, 113139 (2023).
- [29] L.-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).
- [30] Y. Du, H. Fan, and X. Wang, Versatile reservoir computing for heterogeneous complex networks, *Phys. Rev. Appl.* **24**, L031002 (2025).
- [31] J. A. Platt, S. G. Penny, T. A. Smith, T.-C. Chen, and H. D. Abarbanel, A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics, *Neural Networks* **153**, 530 (2022).
- [32] 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).
- [33] E. Lorenz, Deterministic nonperiodic flow, *J. Atmos. Sci.* **20**, 130 (1963).
- [34] J. C. Sprott, *Chaos and Time-Series Analysis* (Oxford University Press, Oxford, 2003).
- [35] V. Le Guen and N. Thome, Shape and time distortion loss for training deep time series forecasting models, *Advances in Neural Information Processing Systems*, Vol. 32 (Curran Associates, Inc., 2019).
- [36] O. E. Rössler, An equation for continuous chaos, *Phys. Lett. A* **57**, 397 (1976).
- [37] T. Ueta and G. Chen, Bifurcation analysis of Chen's equation, *Int. J. Bifurcation Chaos* **10**, 1917 (2000).
- [38] J. L. Hindmarsh and R. M. Rose, A model of neuronal bursting using three coupled first order differential equations, *Proc. R. Soc. London Ser. B* **221**, 87 (1984).

- [39] H. Ma, D. Prosperino, and C. R  th, A novel approach to minimal reservoir computing, *Sci. Rep.* **13**, 12970 (2023).
- [40] J. A. Platt, S. G. Penny, T. A. Smith, T.-C. Chen, and H. D. I. Abarbanel, Constraining chaos: Enforcing dynamical invariants in the training of reservoir computers, *Chaos* **33**, 103107 (2023).
- [41] P. Grassberger and I. Procaccia, Characterization of strange attractors, *Phys. Rev. Lett.* **50**, 346 (1983).
- [42] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, 2nd ed. (Cambridge University Press, Cambridge, 2004).
- [43] X. Chen, T. Weng, H. Yang, C. Gu, J. Zhang, and M. Small, Mapping topological characteristics of dynamical systems into neural networks: A reservoir computing approach, *Phys. Rev. E* **102**, 033314 (2020).
- [44] X. Chen, T. Weng, and H. Yang, Synchronization of spatiotemporal chaos and reservoir computing via scalar signals, *Chaos Solitons Fractals* **169**, 113314 (2023).
- [45] J. M. Hyman and B. Nicolaenko, The Kuramoto-Sivashinsky equation: A bridge between PDE's and dynamical systems, *Physica D* **18**, 113 (1986).
- [46] S. Toh, Statistical model with localized structures describing the spatio-temporal chaos of Kuramoto-Sivashinsky equation, *J. Phys. Soc. Jpn.* **56**, 949 (1987).
- [47] R. W. Wittenberg and P. Holmes, Scale and space localization in the Kuramoto-Sivashinsky equation, *Chaos* **9**, 452 (1999).
- [48] Y. Lan and P. Cvitanovi  , Unstable recurrent patterns in Kuramoto-Sivashinsky dynamics, *Phys. Rev. E* **78**, 026208 (2008).
- [49] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz, Chaos as an intermittently forced linear system, *Nat. Commun.* **8**, 19 (2017).
- [50] J. D. Hart, Attractor reconstruction with reservoir computers: The effect of the reservoir's conditional Lyapunov exponents on faithful attractor reconstruction, *Chaos* **34**, 043123 (2024).
- [51] G. Tanaka, T. Yamane, J. B. H  roux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, *Neural Networks* **115**, 100 (2019).
- [52] Y. Kuriki, J. Nakayama, K. Takano, and A. Uchida, Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers, *Opt. Express* **26**, 5777 (2018).
- [53] J. Sun, W. Yang, T. Zheng, X. Xiong, Y. Liu, Z. Wang, Z. Li, and X. Zou, Novel nondelay-based reservoir computing with a single micromechanical nonlinear resonator for high-efficiency information processing, *Microsyst. Nanoeng.* **7**, 83 (2021).
- [54] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Information processing using a single dynamical node as complex system, *Nat. Commun.* **2**, 468 (2011).
- [55] S. Kan, K. Nakajima, Y. Takeshima, T. Asai, Y. Kuwahara, and M. Akai-Kasaya, Simple reservoir computing capitalizing on the nonlinear response of materials: Theory and physical implementations, *Phys. Rev. Appl.* **15**, 024030 (2021).
- [56] F. Stelzer, A. R  hm, K. L  dge, and S. Yanchuk, Performance boost of time-delay reservoir computing by non-resonant clock cycle, *Neural Networks* **124**, 158 (2020).
- [57] D. Nishioka, T. Tsuchiya, W. Namiki, M. Takayanagi, M. Imura, Y. Koide, T. Higuchi, and K. Terabe, Edge-of-chaos learning achieved by ion-electron-coupled dynamics in an ion-gating reservoir, *Sci. Adv.* **8**, eade1156 (2022).
- [58] A. Banerjee, J. D. Hart, R. Roy, and E. Ott, Machine learning link inference of noisy delay-coupled networks with optoelectronic experimental tests, *Phys. Rev. X* **11**, 031014 (2021).
- [59] A. Banerjee, S. Chandra, and E. Ott, Network inference from short, noisy, low time-resolution, partial measurements: Application to *C. elegans* neuronal calcium dynamics, *Proc. Natl. Acad. Sci. USA* **120**, e2216030120 (2023).
- [60] J. Liu and Y. Zou, GitHub, <https://github.com/yongPIK/DynamicBiasedRC> (2026).