Predicting chaotic dynamics from incomplete input via reservoir computing with $(D+1)$-dimension input and output

Lufa Shi, Youfang Yan, Hengtong Wang, Shengjun Wang, and Shi-Xian Qu[*]
School of Physics and Information Technology, Shaanxi Normal University, Xi'an 710119, China

(Received 22 October 2022; accepted 20 April 2023; published 12 May 2023)

Predicting future evolution based on incomplete information of the past is still a challenge even though data-driven machine learning approaches have been successfully applied to forecast complex nonlinear dynamics. The widely adopted reservoir computing (RC) can hardly deal with this since it usually requires complete observations of the past. In this paper, a scheme of RC with $(D+1)$-dimension input and output (I/O) vectors is proposed to solve this problem, i.e., the incomplete input time series or dynamical trajectories of a system, in which certain portion of states are randomly removed. In this scheme, the I/O vectors coupled to the reservoir are changed to $(D+1)$-dimension, where the first D dimensions store the state vector as in the conventional RC, and the additional dimension is the corresponding time interval. We have successfully applied this approach to predict the future evolution of the logistic map and Lorenz, Rössler, and Kuramoto-Sivashinsky systems, where the inputs are the dynamical trajectories with missing data. The dropoff rate dependence of the valid prediction time (VPT) is analyzed. The results show that it can make forecasting with much longer VPT when the dropoff rate θ is lower. The reason for the failure at high θ is analyzed. The predictability of our RC is determined by the complexity of the dynamical systems involved. The more complex they are, the more difficult they are to predict. Perfect reconstructions of chaotic attractors are observed. This scheme is a pretty good generalization to RC and can treat input time series with regular and irregular time intervals. It is easy to use since it does not change the basic architecture of conventional RC. Furthermore, it can make multistep-ahead prediction just by changing the time interval in the output vector into a desired value, which is superior to conventional RC that can only do one-step-ahead forecasting based on complete regular input data.

DOI: 10.1103/PhysRevE.107.054209

I. INTRODUCTION

Reservoir computing (RC) [1–5], as an effective machine learning approach, has attracted considerable attention of researchers. The basic working element of RC is a recurrent neural network (RNN) called a reservoir, which is actually a high-dimensional nonlinear dynamical system. The input layer feeds the time series into the reservoir, and the coupling between the input layer and the reservoir transforms the input data into spatiotemporal patterns in the corresponding high-dimensional state space. The transient responses of the neurons in the reservoir are transformed into an output signal by the weights of the readout layer. The distinguishing feature of RC is that only the weights of the readout layer are trained by a simple linear regression on the target data, but the input weights and the weights of the internal connections among the neurons in the reservoir are left untrained. This simple and fast learning process tremendously reduces the computational costs and overcomes the vanishing gradients problem associated with other RNN training algorithms [6,7]. Such advantages of RC strongly facilitate the development of physical reservoir computers [8–10] and trigger a great many applications to a wide variety of tasks, including chaotic time series prediction [3–5,11–13], reconstruction of chaotic attractor [14–17], speech and image recognition [9,18,19], nonlinear channel equalization [3], and inference of unmeasured state variables [20,21]. To improve the performance of RC and deal with the complex problems encountered in predicting the future evolution of dynamical systems, a variety of sophisticated schemes based on conventional RC have been proposed, for instance, the parallel scheme [5,11,12], which uses many spatially distributed machine learning devices to effectively deal with the problem that demands extensive computational resources when predicting the dynamics of large spatiotemporally chaotic systems and complex networks; the deep reservoir scheme [13] that uses cascaded nonlinear oscillators acting as a computational substrate to significantly improve the computational performance when comparing with a single-layer reservoir of identical size; and the rare update scheme [22], which incorporates sparsely sampled real data of the target system into the reservoir during the prediction phase and is capable of generating an arbitrarily long prediction horizon for a variety of chaotic systems.

Although the RC approach and its improvements have achieved great success in a variety of tasks, they usually deal with sequential data or time series. During their applications, one might often make a default assumption that the input data are complete and the datasets are prepared by rather dense sampling points with uniform intervals. Unfortunately, complete data with uniform interval are sometimes inaccessible, and the situation of missing data is often encountered in practice. For instance, it is impossible or very difficult to mine data in an extremely disastrous environment [23], and

[*]sxqu@snnu.edu.cn

certain measurements of data may be too inaccurate due to heavy pollution of noise so that one must discard them [24]. Hence, there rises a question of how to predict the future when one has incomplete information of the past? To solve it, we try to extend the applicability of RC to the situation of incomplete input in this paper. More precisely, we consider a benchmark task in the RC community, i.e., predicting chaotic time series. Regarding the incomplete input, we only consider the case in which certain data in the time series (or the past trajectories of dynamical systems) are randomly removed (or missing), but the knowledge of positions where the data are missing is available. In other words, we can access the information of time intervals between adjacent states in this incomplete time series in addition to the state variables themselves.

Obviously, a naïve application of RC is doomed to failure when the time series for training is incomplete. When the rate of missing data is very small, e.g., 1%, the length of prediction in Lyapunov time is not more than two for the logistic map and zero for Lorenz, Rössler, and Kuramoto-Sivashinsky (KS) systems. There have been several works demonstrating their feasibility for predicting chaotic systems based upon input data that are only a subset of states of the dynamical system, also called partial observations [25–28]. They commonly assume that they have an imperfect knowledge-based model, which may be due to the insufficient understanding and/or limited resolution of the underlying physical process. Their approach is a hybrid one, which combines the data-driven machine-learning approach with incomplete input and the imperfect knowledge-based model associated with data assimilation techniques. Data assimilation techniques may remedy the missing data by the knowledge-based model, resulting in complete input data. It is clear that the application of this method has strong limitations, which require that the knowledge of the dynamics of the system should be known even though it is imperfect. It is essentially different from the situation in this paper.

From another point of view, the incomplete input data involved in this paper can be considered a complete time series without uniform intervals, which is the only difference from conventional RC trained by complete time series with uniform interval. According to this argument, we propose a RC scheme that changes the D-dimensional input and output (I/O) vectors into $(D+1)$-dimension, where D is the dimension of the input time series, and the additional dimension is the time interval associated with the data in the time series. When the time interval is a constant, this model is very similar to conventional RC, where the extra dimension can be considered as bias. It has been reported that introducing bias into conventional RC may considerably improve its performance [21,29]. We must point out that the scheme of introducing an additional input channel for RC has been independently developed in Refs. [30–34] recently. It has been successfully employed to solve many important problems in the field of nonlinear dynamics, such as predicting critical transition [30], amplitude death [31] and a variety of bifurcations [32], studying nonstationary chaotic systems [33], accomplishing dynamical learning of dynamics [34], and so on. In these examples, a common point is that the additional channels store the control parameters or some other properties of the systems. Clearly, the additional input channel is somehow similar to the extra dimension of time intervals in our model. Designating an additional input channel for artificial neural networks has been widely used not only in RNNs like reservoir computers but also in feed-forward neural networks [35,36]. The success of these works support the feasibility of the scheme with $(D+1)$-dimension I/O developed in this paper. Pretty good results are obtained when we apply this scheme to predict the chaotic time series of the logistic map and Lorenz, Rössler, and KS models.

The rest of this paper is organized as follows. In Sec. II, the architecture of this RC scheme with $(D+1)$-dimension I/O is proposed, and the corresponding criteria to evaluate the quality of predictions and the determination of the hyperparameters are described. The results for the application of this scheme to the logistic map and Lorenz, Rössler, and KS systems are shown in Sec. III, where the corresponding analysis and discussion are also presented. The conclusion appears in the last section.

II. MODEL AND METHODS

A. RC scheme with $(D+1)$-dimension I/O

A reservoir computer usually consists of three main elements: a fixed input layer, a fixed dynamical system with a high-dimensional phase space, called the reservoir, and a linear trainable readout layer [14]. Revision or variation in any of the three elements may result in a different scheme. In the reservoir, the neurons are connected according to an internal connection matrix $\mathbf{A} \in \mathbb{R}^{D_r \times D_r}$, whose entries define the links of the reservoir network. This network is sparse and has an average degree of six [4]. The nonzero entries of $\mathbf{A}$ are drawn from uniformly distributed random numbers in [0, 1] and then scaled so that the absolute value of the largest eigenvalue of the matrix, i.e., the spectral radius, equals ρ. The internal dynamical state of the network at time step n is stored in vector $\mathbf{r}(n) \in \mathbb{R}^{D_r \times 1}$. In conventional RC, an input vector $\mathbf{u}(n)$ of dimension D is coupled to the D_r-dimension state space of the reservoir through weight matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{D_r \times D_{\text{in}}}$, whose elements are chosen from the random numbers distributed uniformly in the interval $[-\beta, \beta]$. The evolution of the internal dynamics of the reservoir is governed by the following equation:

$$\mathbf{r}(n+1) = \tanh[\mathbf{A}\mathbf{r}(n) + \mathbf{W}_{\text{in}}\mathbf{u}(n)]. \tag{1}$$

The D-dimension output vector $\mathbf{v}(n+1)$ can be obtained by

$$\mathbf{v}(n+1) = \mathbf{W}_{\text{out}}[\mathbf{r}(n+1), \mathbf{p}], \tag{2}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D_{\text{out}}}$ is a mapping from the D_r-dimensional space of the dynamic state of the reservoir to D_{out}-dimensional output state space, and $\mathbf{p}$ is an adjustable parameter [5], which can be obtained by linearly fitting Eq. (2) into the target $\mathbf{v}_d(n+1)$. For the sake of simplicity, we only discuss the case of $D_{\text{in}} = D_{\text{out}} = D$ in this paper.

In fact, artificial neural networks are universal function approximators [37,38]. In the conventional scheme of implementing RC, the state of the dynamical system is the only input, and the next point of the dynamical trajectory serves as the target. Hence, the corresponding approximation results in the so-called one-step-ahead prediction, which may only deal with the situation where the input data are the state variables of the dynamics, and the time interval between any adjacent trajectories is assumed to be uniform. For the case of incomplete input data discussed here, it is actually a time series with nonuniform time intervals, which may provide both one-step- and multiple-step-ahead evolutional information of the dynamics in the past. Conventional RC is unable to sense the variation of the time intervals in the training process and thus cannot make a prediction if we simply feed this kind of input into the system. Therefore, something should be changed so that the RC can feel the variation of the time intervals. A simple idea comes to our mind: Why should we not also feed the information of the time intervals into the RC? We introduce a new $(D+1)$-dimension input vector:

$$\tilde{\mathbf{u}}(t_n) \equiv [\mathbf{u}(t_n), \Delta t_n]^\dagger, \tag{3}$$

where the † symbol denotes the transpose of a matrix. Here, $\tilde{\mathbf{u}}(t_n)$ includes information of both the dynamical states (the first D dimension) and time intervals (the time dimension). Here, the time interval is given by $\Delta t_n \equiv t_{n+1} - t_n$. In this way, we may equivalently treat both the dynamical states and time intervals as dynamical variables. Accordingly, the corresponding output and target are changed into $(D+1)$-dimension vectors, defined by, i.e.,

$$\tilde{\mathbf{v}}(t_{n+1}) \equiv [\mathbf{v}(t_{n+1}), \Delta t_{n+1}]^\dagger, \tag{4}$$

$$\tilde{\mathbf{v}}_\mathrm{d}(t_{n+1}) \equiv [\mathbf{v}_\mathrm{d}(t_{n+1}), \Delta t_{n+1}]^\dagger. \tag{5}$$

The coupling between the input and the internal dynamics should be redefined by the concatenation of $\mathbf{W}_\mathrm{in}^\mathrm{s}$ and $\mathbf{W}_\mathrm{in}^\mathrm{t}$, i.e.,

$$\tilde{\mathbf{W}}_\mathrm{in} \equiv \left(\mathbf{W}_\mathrm{in}^\mathrm{s}, \mathbf{W}_\mathrm{in}^\mathrm{t}\right), \tag{6}$$

where $\mathbf{W}_\mathrm{in}^\mathrm{s} \in \mathbb{R}^{D_\mathrm{r}\times D}$ is the coupling matrix between the state variables in the input and dynamical states of neurons in the reservoir, and $\mathbf{W}_\mathrm{in}^\mathrm{t} \in \mathbb{R}^{D_\mathrm{r}}$ is the coupling matrix between the time intervals of input and the dynamical states of the neurons. Thus, $\tilde{\mathbf{W}}_\mathrm{in}$ is a D_r by $(D+1)$ matrix. We must point out that $\tilde{\mathbf{W}}_\mathrm{in}$ will not change during the training and predicting phases if it is previously determined. The method to determine these matrices will be described in Sec. II C. Similarly, $\tilde{\mathbf{W}}_\mathrm{out} \in \mathbb{R}^{(D+1)}$ becomes the mapping from the dynamical state of the reservoir to a $(D+1)$-dimension output $\tilde{\mathbf{v}}(t_{n+1})$. It is the only training quantity for the RC and can be obtained by linear regression.

After reforming and redefining the input state, the output state, the target state, and the coupling matrices, the evolution of the internal dynamics of the reservoir can be rewritten, in the language of our $(D+1)$-dimension scheme, in the following form:

$$\mathbf{r}(t_{n+1}) = \tanh[\mathbf{A}\mathbf{r}(t_n) + \tilde{\mathbf{W}}_\mathrm{in}\tilde{\mathbf{u}}(t_n)]. \tag{7}$$

The $(D+1)$-dimension output vector $\tilde{\mathbf{v}}(t_{n+1})$ is obtained by

$$\tilde{\mathbf{v}}(t_{n+1}) = \tilde{\mathbf{W}}_\mathrm{out}[\mathbf{r}(t_{n+1}), \mathbf{p}]. \tag{8}$$

Now we have built up the framework of our $(D+1)$-dimension scheme, as shown in Fig. 1. Introducing an additional dimension of the time interval variable makes it generalize the RC to deal with input time series with nonuniform time intervals without changing the basic architecture, where the state variable and the time variable are treated

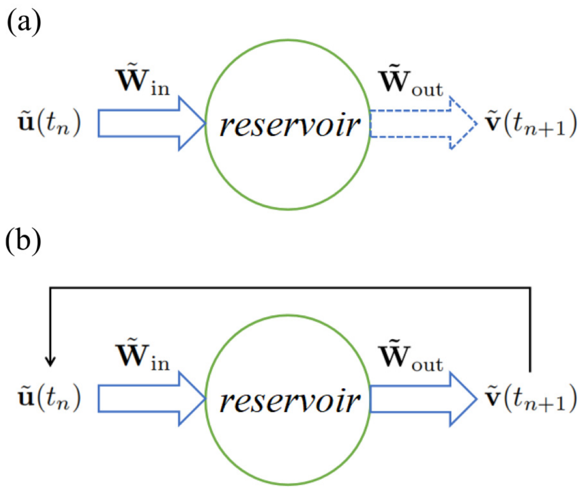
(a)
$\tilde{\mathbf{W}}_\mathrm{in}$
$\tilde{\mathbf{W}}_\mathrm{out}$
$\tilde{\mathbf{u}}(t_n)$
reservoir
$\tilde{\mathbf{v}}(t_{n+1})$
(b)
$\tilde{\mathbf{W}}_\mathrm{in}$
$\tilde{\mathbf{W}}_\mathrm{out}$
$\tilde{\mathbf{u}}(t_n)$
reservoir
$\tilde{\mathbf{v}}(t_{n+1})$

FIG. 1. Illustration of reservoir computing scheme with $(D+1)$-dimension input and output (I/O). (a) The training phase, where the $(D+1)$-dimension input vector $\tilde{\mathbf{u}}(t_n)$ is fed into the reservoir successively. (b) The predicting phase, where a $(D+1)$-dimension output $\tilde{\mathbf{v}}(t_{n+1})$ is obtained at each time step, which is then fed back into the input, making the system form a loop and run autonomously.

equivalently. In addition, the dimension of the time interval is similar to bias in conventional RC [21,29], especially when all the time intervals are the same. The scheme proposed here might be cataloged into the same class as the works that introduce an additional channel to feed into the control parameter or other parameters [30,32–34]. This similarity suggests we believe that this $(D+1)$-dimension scheme will be successful in making prediction based on the incomplete and irregular input time series. The examples for applying this scheme to predict the future evolution of dynamical systems from an incomplete input time series will be shown in the following section.

During the training phase, the reservoir will evolve from an initial time t_0 for N_W time steps to avoid the influence of arbitrary initial states, which is known as the warmup process. Then the system undergoes evolutions for N_T steps from time t_{N_W} to train the system. At the boundary of the time between warmup and training phases, one has $\tilde{\mathbf{u}}(\hat{t}_0) = \tilde{\mathbf{u}}(t_{N_\mathrm{W}})$ and $\tilde{\mathbf{v}}_\mathrm{d}(\hat{t}_0) = \tilde{\mathbf{u}}(\hat{t}_0)$. Thus, $\hat{t}_n$ in the training phase is the relative time with the starting point $\hat{t}_0 = t_{N_\mathrm{W}}$. After the system evolves for N_T steps from this time instant, we obtain a series of reservoir state vectors $\{\mathbf{r}(\hat{t}_i), i = 1, N_\mathrm{T}\}$, by which we may concatenate respectively the state vectors and their squares to form the following D_r by N_T matrices, i.e.,

$$\begin{aligned}\mathbf{R} &= \left[\mathbf{r}(\hat{t}_1), \mathbf{r}(\hat{t}_2), \mathbf{r}(\hat{t}_3), \ldots, \mathbf{r}\left(\hat{t}_{N_\mathrm{T}}\right)\right],\\ \mathbf{R}^2 &= \left[\mathbf{r}^2(\hat{t}_1), \mathbf{r}^2(\hat{t}_2), \mathbf{r}^2(\hat{t}_3), \ldots, \mathbf{r}^2\left(\hat{t}_{N_\mathrm{T}}\right)\right].\end{aligned} \tag{9}$$

The corresponding target $\tilde{\mathbf{V}}_\mathrm{d}$ is defined by

$$\begin{aligned}\tilde{\mathbf{V}}_\mathrm{d} &= \left[\tilde{\mathbf{v}}_\mathrm{d}(\hat{t}_1), \tilde{\mathbf{v}}_\mathrm{d}(\hat{t}_2), \tilde{\mathbf{v}}_\mathrm{d}(\hat{t}_3), \ldots, \tilde{\mathbf{v}}_\mathrm{d}\left(\hat{t}_{N_\mathrm{T}}\right)\right]\\ &= \left[\tilde{\mathbf{u}}(\hat{t}_1), \tilde{\mathbf{u}}(\hat{t}_2), \tilde{\mathbf{u}}(\hat{t}_3), \ldots, \tilde{\mathbf{u}}\left(\hat{t}_{N_\mathrm{T}}\right)\right],\end{aligned} \tag{10}$$

which is an $(D+1)\times N_\mathrm{T}$ matrix. Following Ref. [5], in this paper, we express the right-hand side of Eq. (8) as

$$\tilde{\mathbf{W}}_\mathrm{out}[\mathbf{r}(t_{n+1}), \mathbf{p}] = \mathbf{P}_1\mathbf{r}(t_{n+1}) + \mathbf{P}_2\mathbf{r}^2(t_{n+1}). \tag{11}$$

Then we may relate $\tilde{\mathbf{V}}_\mathrm{d}$ to $\mathbf{R}$ and $\mathbf{R}^2$ by

$$\tilde{\mathbf{V}}_\mathrm{d} = \mathbf{P}_1\mathbf{R} + \mathbf{P}_2\mathbf{R}^2. \tag{12}$$

Here, $\mathbf{P}_1$ and $\mathbf{P}_2$ are the $(D+1)$ by D_r coupling matrices, which can be determined by the ridge regression [5,11] that minimizes the least squares difference:

$$\ell = \|\mathbf{P}_1\mathbf{R} + \mathbf{P}_2\mathbf{R}^2 - \tilde{\mathbf{V}}_\mathrm{d}\|^2 + \alpha\|\tilde{\mathbf{V}}_\mathrm{d}\|^2, \tag{13}$$

where α is a small positive regularization constant to avoid overfitting.

Once the mapping $\tilde{\mathbf{W}}_\mathrm{out}$ is obtained, the system turns into the prediction phase, where the relative time $\hat{t}_n$ with the starting point $\hat{t}_0 = t_{N_\mathrm{W}+N_\mathrm{T}}$ is used. At the boundary of the time between training and predicting phases, one has $\tilde{\mathbf{u}}(\hat{t}_0) = \tilde{\mathbf{u}}(t_{N_\mathrm{W}+N_\mathrm{T}})$ and $\tilde{\mathbf{v}}_\mathrm{d}(\hat{t}_0) = \tilde{\mathbf{u}}(\hat{t}_0)$. The output $\tilde{\mathbf{v}}(\hat{t}_{n+1})$ is given by

$$\tilde{\mathbf{v}}(\hat{t}_{n+1}) = \mathbf{P}_1\mathbf{r}(\hat{t}_{n+1}) + \mathbf{P}_2\mathbf{r}^2(\hat{t}_{n+1}). \tag{14}$$

Feed back $\tilde{\mathbf{v}}(\hat{t}_{n+1})$ to $\tilde{\mathbf{u}}(\hat{t}_n)$ [or say replace $\tilde{\mathbf{u}}(\hat{t}_n)$ by $\tilde{\mathbf{v}}(\hat{t}_{n+1})$], and let the RC evolve; it then becomes an autonomous dynamical system and is ready to make prediction. Here, we must mention that the time intervals in the training phase take the values of those in the input data. However, during the prediction phase, it may be simply set to $\Delta\hat{t}_n =$ constant (usually $\Delta\hat{t}_n = 1$ for the time-discrete maps, and $\Delta\hat{t}_n = \hat{\tau}$ in the time-continuous systems, where $\hat{\tau}$ is the integral time step). Running autonomously for N_P time steps, the system will produce a prediction of time series, i.e., $\{\mathbf{v}(\hat{t}_1), \mathbf{v}(\hat{t}_2), \mathbf{v}(\hat{t}_3), \ldots, \mathbf{v}(\hat{t}_{N_\mathrm{P}})\}$, where $\mathbf{v}(\hat{t}_n)$ is the state variable in $\tilde{\mathbf{v}}(\hat{t}_n)$, or the first D-dimension component of it.

B. Criteria for the efficiency of prediction

To test the feasibility and validity of the scheme proposed in this paper, as the prototypes, we employ four well-known theoretical models (which will be shown in the next section). These models produce chaotic time series to coin simulated incomplete inputs and provide standards to measure the validity of the predictions. Suppose that the predicted time series reads $\{\mathbf{v}(\hat{t}_1), \mathbf{v}(\hat{t}_2), \mathbf{v}(\hat{t}_3), \ldots, \mathbf{v}(\hat{t}_N)\}$, and the corresponding true counterparts produced by the theoretical models are $\{\mathbf{v}_\mathrm{d}(\hat{t}_1), \mathbf{v}_\mathrm{d}(\hat{t}_2), \mathbf{v}_\mathrm{d}(\hat{t}_3), \ldots, \mathbf{v}_\mathrm{d}(\hat{t}_N)\}$, with

$$\mathbf{v}_\mathrm{d}(\hat{t}_n) = \mathbf{F}[\mathbf{v}_\mathrm{d}(\hat{t}_{n-1})], \tag{15}$$

considering the one-step-ahead nature of the RC, where $\mathbf{F}(\cdot)$ is a time-discrete mapping. For the logistic map, it reduces to a one-dimensional mapping, i.e., $f(x_n) = \mu x_n(1-x_n)$, while for a general N-dimension dynamical system, it is formally an N-dimension map, which is the iteration solution of the time-continuous differential equation.

Now we introduce two measures to determine the validation of the predicted time series. The first measure is the so-called valid prediction time (VPT) denoted by T_λ, which is the longest duration of time before the normalized error $E(\hat{t}_N)$ of prediction exceeds a threshold ε. Here, we set $\varepsilon = 0.1$. Then it is given by the following formula:

$$T_\lambda \equiv \lambda_\mathrm{m} \max(\hat{t}_N - \hat{t}_0), \quad \text{if } E(\hat{t}_N) \leqslant \varepsilon. \tag{16}$$

where λ_m is the maximum Lyapunov exponent, and thus, T_λ is in Lyapunov time $1/\lambda_\mathrm{m}$. The normalized error is defined by

$$E(\hat{t}_N) = \frac{\|\mathbf{v}_\mathrm{d}(\hat{t}_N) - \mathbf{v}(\hat{t}_N)\|}{\sqrt{\langle \mathbf{v}_\mathrm{d}^2\rangle}}, \tag{17}$$

where the mean square of the target is given by

$$\langle \mathbf{v}_\mathrm{d}^2\rangle = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{v}_\mathrm{d}(\hat{t}_n)\|^2. \tag{18}$$

Another measure denoted by σ is actually the root mean square error (RMSE) of $\{\mathbf{v}(\hat{t}_n)\}$, which is defined by

$$\sigma = \left(\frac{1}{N_\mathrm{P}-1}\sum_{n=1}^{N_\mathrm{P}}\{\mathbf{v}(\hat{t}_n) - \mathbf{F}[\mathbf{v}(\hat{t}_{n-1})]\}^2\right)^{1/2}. \tag{19}$$

This quantity reveals globally how far the prediction is away from the real one or the precision of the reconstruction of the dynamics (or attractor) hidden in the input time series.

C. Hyperparameter optimization

As we have mentioned previously, there are three pre-determined matrices in the current $(D+1)$-dimension scheme. They are $\mathbf{A}$ with spectral radius ρ, $\mathbf{W}_\mathrm{in}^\mathrm{s}$ with its elements drawn from uniformly distributed random numbers in interval $[-\beta, \beta]$, and $\mathbf{W}_\mathrm{in}^\mathrm{t}$ with those in interval $[-\gamma, \gamma]$. The value of γ is set to $\gamma = \beta\sqrt{\langle \mathbf{u}^2\rangle}$ with

$$\langle \mathbf{u}^2\rangle = \frac{1}{N_\mathrm{T}}\sum_{i=1}^{N_\mathrm{T}}\|\mathbf{u}(\hat{t}_i)\|^2. \tag{20}$$

The reason for this setting lies in the fact that it is a plausible choice to make the terms of state variable $\mathbf{W}_\mathrm{in}^\mathrm{s}\mathbf{u}(\hat{t}_n)$ and time variable $\mathbf{W}_\mathrm{in}^\mathrm{t}\Delta\hat{t}_n$ have approximately the same contribution for the reservoir neural network. We have empirically found that this setting of γ works well for our $(D+1)$-dimension scheme, and thus, further optimizing of γ is not required.

As is well known, ρ, β, and α are the hyperparameters which show serious influence on the performance of the RC and should be chosen very carefully. Furthermore, different choices of $\mathbf{A}$ and $\mathbf{W}_\mathrm{in}^\mathrm{s}$ may also affect the result of prediction [5]. Therefore, coarse grid sweeps are carried out in the parameter regimes of ρ, β, and α, respectively. One hundred distinct random network realizations are generated for each set of the hyperparameters. Then the average of the VPT T_λ and the RMSEs σ over all the network realizations are carried out. Finally, one may find a set of optimum values of them, by which T_λ reaches its maximum or σ gets its minimum.

D. Preparing an incomplete input time series by randomly removing data

For the dynamical systems involved in this paper, we can formally express their evolution by the following iteration:

$$\bar{\mathbf{u}}_{i+1} = \mathbf{F}(\bar{\mathbf{u}}_i), \tag{21}$$

by which we produce a time series $\{\bar{\mathbf{u}}_i,\ i = 0, N_0\}$ with uniform time interval $\tau_i \equiv 1$. An incomplete time series can be coined by simply removing data with a uniform probability from this time series, for instance, the resulting time series $\{\bar{\mathbf{u}}_n\} = \{\bar{\mathbf{u}}_0, \bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2, \bar{\mathbf{u}}_4, \bar{\mathbf{u}}_5, \bar{\mathbf{u}}_8, \bar{\mathbf{u}}_9, \ldots\ldots\}$ and the corresponding time intervals $\{\tau_n\} = \{1, 1, 2, 1, 3, 1, 1, \cdots\cdots\}$. It is an incomplete time series with missing data or, say, a time series with nonuniform time intervals. Here and afterward, the dimensionless integer time interval $\tau_n = \Delta\hat{t}_n/\hat{\tau}$ will be used to replace $\Delta\hat{t}_n$. Hence, N_0 is the training time in units of $\hat{\tau}$. The dropoff rate of the incomplete time series is defined by

$$\theta = N_{\text{off}}/N_0, \tag{22}$$

where N_0 is the initial number of states produced by the mapping in Eq. (21), and N_{off} is the number of states randomly removed.

If the abovementioned incomplete time series is an input of the RC in the training phase, then the target is $\{\bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2, \bar{\mathbf{u}}_4, \bar{\mathbf{u}}_5, \bar{\mathbf{u}}_8, \bar{\mathbf{u}}_9, \ldots\}$, considering the one-step-ahead prediction nature. In our notation system for the $(D+1)$-dimension scheme setup in Sec. II A, the input and the corresponding target are expressed respectively by

$$\tilde{\mathbf{u}}(t_n) : \begin{pmatrix}\bar{\mathbf{u}}_0\\1\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_1\\1\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_2\\2\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_4\\1\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_5\\3\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_8\\1\end{pmatrix}\ldots, \tag{23}$$

where $0 \leqslant n \leqslant N_{\text{W}} + N_{\text{T}} - 1$, and

$$\tilde{\mathbf{v}}_{\text{d}}(t_n) : \begin{pmatrix}\bar{\mathbf{u}}_1\\1\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_2\\2\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_4\\1\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_5\\3\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_8\\1\end{pmatrix}\begin{pmatrix}\bar{\mathbf{u}}_9\\1\end{pmatrix}\ldots, \tag{24}$$

where $1 \leqslant n \leqslant N_{\text{W}} + N_{\text{T}}$, and $N_{\text{W}} + N_{\text{T}} = (1-\theta)N_0$. Here, N_0 is invariable to ensure the RC is trained for the same time period but different numbers of states as θ varies.

We must emphasize that different configurations of missing data will be drawn when coining an incomplete time series, even when the dropoff rate is the same, which may affect the efficiency of predictions. Therefore, the averages of the characteristic quantities for predictions should be done over a number of configurations, especially when one intends to investigate the effect of the dropoff rate on the results. In this paper, 100 configurations are involved in the averages. Therefore, the VPT mentioned in the following sections is the mean value, denoted by $\langle T_\lambda \rangle$, which is obtained by the average over the reservoir realizations and the configurations of the input data.

III. RESULTS AND DISCUSSION

A. Logistic system

The first example for applying this RC scheme with $(D+1)$-dimension I/O is the logistic map, which is a one-dimensional discrete dynamical system. The mapping function is

$$x_{n+1} = \mu x_n(1 - x_n). \tag{25}$$

Here, the control parameter μ is set to 3.98, at which the system is in a chaotic state. The corresponding Lyapunov exponent λ is calculated through

$$\lambda(x_0) = \frac{1}{N}\sum_{n=1}^{N} \ln|f'(x_n)|, \tag{26}$$

where x_0 is the initial condition, $f'(\cdot)$ is the derivative of the mapping function, $N = 30\,000$, and $10\,000$ transient states

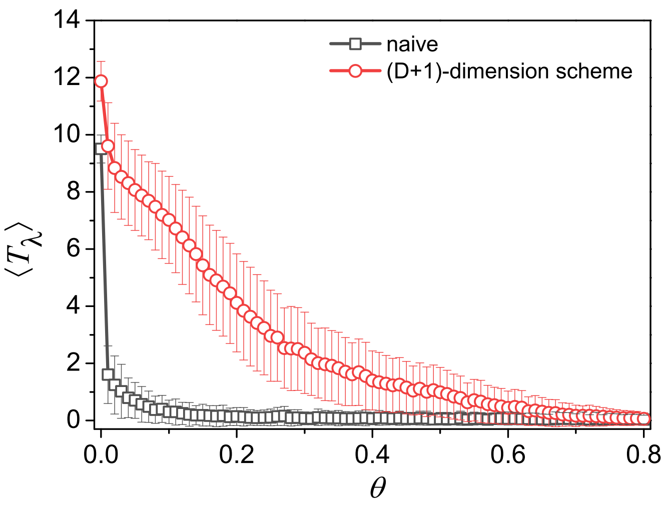

FIG. 2. The θ dependence of valid prediction times (VPTs) for the logistic map. The black squares are $\langle T_\lambda \rangle$ predicted by conventional reservoir computing (RC), and the red circles are those by our RC with $(D+1)$-dimension input and output (I/O).

are discarded for each x_0. In general, one may assume that the Lyapunov exponent calculated by this equation is dependent on the choice of the initial states. Thus, the average of $\lambda(x_0)$ over 100 randomly chosen initial conditions is done to obtain the Lyapunov exponent. The resulting value is $\lambda = 0.616 \pm 0.001$. In the performance of our RC for this system, the time divides are set as follows, i.e., $N_0 = 900$, $N_{\text{W}} = 50$, $N_{\text{T}} = N_0(1-\theta) - N_{\text{W}}$, and $N_{\text{P}} = 1000$. The number of nodes in the reservoir is $D_{\text{r}} = 500$. The hyperparameters are $\rho = 0.2$, $\beta = 1.9$, and $\alpha = 1 \times 10^{-10}$, determined through the complete time series.

In Fig. 2, the variations of the VPT $\langle T_\lambda \rangle$ with respect to the dropoff rate θ for different feeding-in schemes are shown. One may obviously see that conventional RC exhibits very poor predictability when we naïvely feed the incomplete time series into the reservoir, as shown by the black squares in Fig. 2. The curve shows monotonic decrease and reaches a plateau at $\theta \simeq 20\%$, where the VPT is very close to zero. The VPT is not more than 2 Lyapunov time even when the dropoff rate θ is very small, e.g., 1%, where the data missing are negligible. It can be easily understood if we look back to the structure of the input data. In an incomplete time series, there are states with different time intervals, such as one-, two-, three-step, and even larger ones, which correspond to the information of the first-, second-, third-, and higher-order mapping, respectively. While conventional RC uses the one-step-ahead forecasting mechanism, the time interval is assumed to be one in the training process. If we simply feed the incomplete input into the RC, it treats all the states with intervals other than one step as the one-step state, which is to say that the machine blends the second- and higher-order iterations of the map and learns the erratic information of the dynamics of the mapping and thus cannot give satisfactory results.

The quality of prediction has been considerably improved when applying our RC with the $(D+1)$-dimension I/O scheme. The result is illustrated by the red circles in Fig. 2. We found that, at very low dropoff rate ($\theta \simeq 1\%$), the VPT

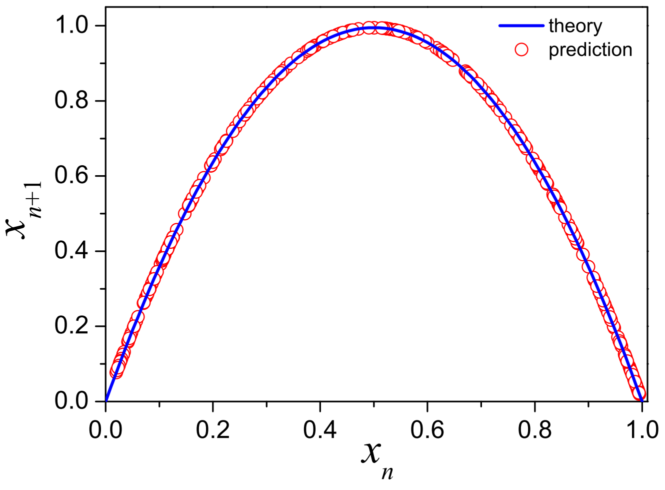

FIG. 3. Reconstruction of the logistic map when $\theta = 1\%$. The line represents the theoretical values, and the circles are their predicted counterparts. The root mean square error (RMSE) for the reconstruction is $\sigma = 1 \times 10^{-6}$.

reaches 9.6 Lyapunov time. Nevertheless, the shape of the logistic mapping has been successfully reconstructed for 80 out of 100 distinct random network realizations, as shown in Fig. 3. The RMSE for the reconstruction is very small, i.e., $\sigma = 1 \times 10^{-6}$. Even when the dropoff rate reaches 30%, the reconstruction still fits into the mapping function very well, and the RMSE keeps a low value, i.e., $\sigma = 0.01$. In contrast, conventional RC simply fed by the incomplete time series cannot do so. Globally, the VPT $\langle T_\lambda \rangle$ decreases monotonically with the increase of the dropoff rate. It is <1 when $\theta > 50\%$, which gives the maximum dropoff rate before which this scheme of RC can make a good prediction, denoted by $\theta_m = 50\%$. The result reveals that our approach can only tolerate the incomplete time series with a lower dropoff rate. The reason is explained as follows. In general, an incomplete time series coined by randomly removing states from its complete counterpart consists of states with different time intervals. The dominant states in the series are the ones with the time interval $\tau_n = 1$ when the dropoff rate is very low. As θ increases, the portion of this kind of state will reduce, while those with $\tau_n > 1$ increase. The greater the dropoff rate, the more it contains states with larger time intervals. The states with larger time intervals may become dominant when θ is very large. Here, our approach is set to make one-step-ahead forecasting. Therefore, it loses the capacity to make a prediction since the system has learned very little information about the one-step iterations during the training phase.

To verify our argument, we prepare a specific type of incomplete time series by randomly dropping off data from a complete time series with the limitation that no two adjacent data can be removed. Obviously, the resulting time series consists of only two kinds of states: one with $\tau_n = 1$ and the other with $\tau_n = 2$. Thus, the number of states removed is the same as the number of states with $\tau_n = 2$. In principle, the current RC approach may realize the two-step-ahead forecast by simply changing the time interval in the output vector to $\tau_n = 2$. Based on this special incomplete time series, we train our RC first and then let it carry out one- and two-step-ahead prediction, respectively. The results are plotted in Fig. 4. One may immediately find that the dropoff rate dependence of the VPT $\langle T_\lambda \rangle$ ends at $\theta = 50\%$ in Fig. 4(a). The VPT $\langle T_\lambda \rangle$ by the one-step-ahead forecasting goes to zero but goes to its maximum in the two-step-ahead forecasting. The reason is that all the states in the series are those with two-step states (i.e., $\tau_n = 2$) in this occasion. Therefore, the machine cannot predict the one-step iterations. When $\theta = 0$, the VPT by one-step-ahead forecasting is $\langle T_\lambda \rangle \sim 12$, but that by two-step-ahead forecasting is $\langle T_\lambda \rangle = 0$. This is because the time series stores only the knowledge of one-step iterations in this case; the reservoir learned the full information of one-step iterations but learned nothing of the two-step-ahead iterations. Hence, it cannot make a prediction for the two-step iteration. As θ increases, the VPT by one-step-ahead forecasting decreases, but the VPT by the latter increases. The crossover happens at $\theta = 25\%$, where both kinds of forecasting give approximately the same VPT since half of the training time is for the one-step and the other for the two-step iterations. The plots in Figs. 4(b) and 4(c) are the corresponding reconstructions of the first- and second-order mapping, respectively, which are shown by the circles. The line represents the theoretical

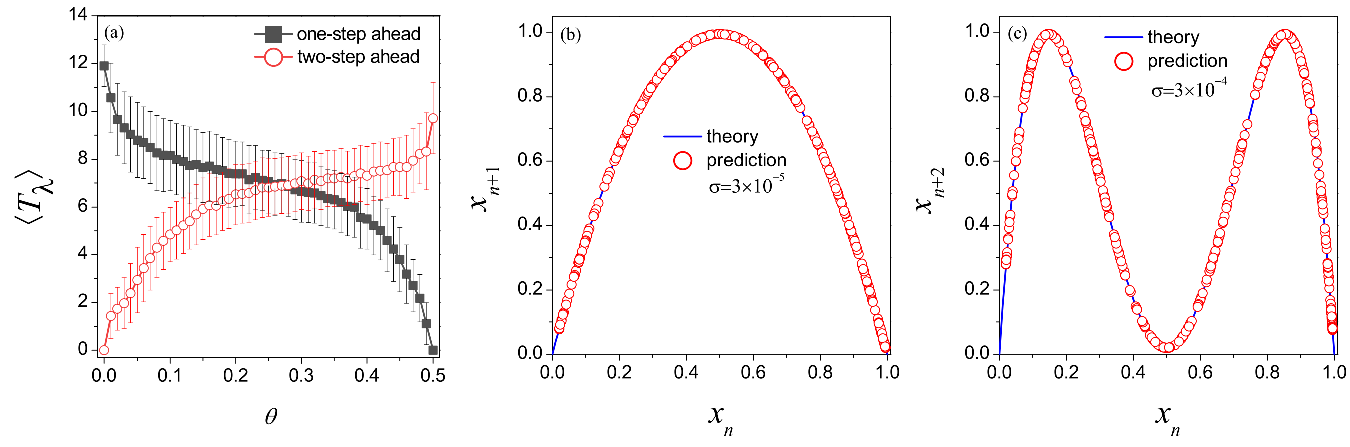

FIG. 4. Prediction results based on input with two kinds of states. (a) The dropoff rate dependence of valid prediction times (VPTs) by one-step-ahead (black squares) and two-step-ahead (red circles) predictions, respectively, where the input time series has only two kinds of time intervals, i.e., $\tau_n = 1$ and 2. Reconstruction of the (b) first-order and (c) second-order mapping functions when $\theta = 25\%$.

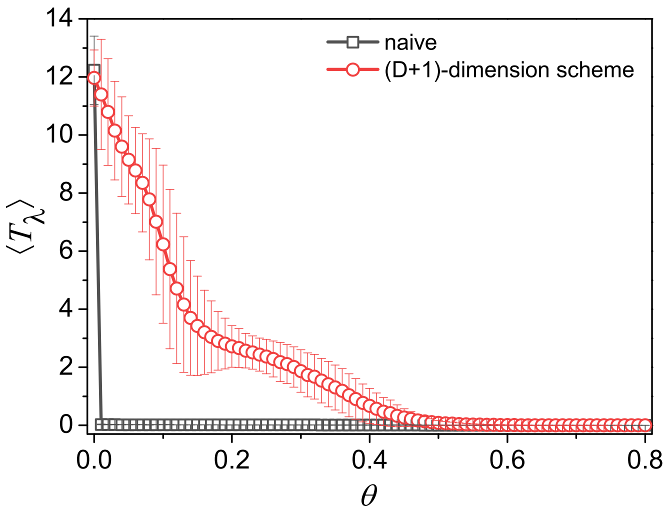

FIG. 5. The θ dependence of valid prediction times (VPTs) for the Lorenz system. The black squares are the predictions by conventional reservoir computing (RC), and the red circles are those by our RC scheme with $(D+1)$-dimension input and output (I/O).

curve. Obviously, the prediction results fit into the theory very well. The RMSEs for the reconstruction are $\sigma = 3 \times 10^{-5}$ and 3×10^{-4}, respectively.

B. Lorenz and Rössler systems

Now we extend the application of this scheme to the differential dynamical systems, e.g., the Lorenz and Rössler systems. The fourth-order Runge-Kutta method is employed to generate dynamical trajectories of the two systems, and the Lyapunov exponents are calculated by the *QR* decomposition algorithm, where the time steps are $\hat{\tau} = 0.02$ and 0.05, respectively. In the simulation, the time interval $\Delta\hat{t}_n$ is rescaled by $\hat{\tau}$ to get an integer number. Here, the time divides for our RC are set as follows, i.e., $N_0 = 3100$, $N_\mathrm{W} = 100$, $N_\mathrm{T} = N_0(1-\theta) - N_\mathrm{W}$, and $N_\mathrm{P} = 5000$. The number of nodes in the reservoir is $D_\mathrm{r} = 500$.

The Lorenz system is described by the following differential equations:

$$\dot{x} = a(y-x), \quad \dot{y} = -xz + bx - y, \quad \dot{z} = xy - cz. \tag{27}$$

Here, the parameters are set to $a = 10$, $b = 28$, $c = \frac{8}{3}$. The Lyapunov exponents are $\{\lambda_1, \lambda_2, \lambda_3\} = [0.9, 0, -14.5]$. The hyperparameters for the RC are $\rho = 0.7$, $\beta = 0.05$, and $\alpha = 1 \times 10^{-10}$.

The variations of the VPTs with respect to the dropoff rate are shown in Fig. 5. As shown by the black squares, a significant difference is that conventional RC cannot give any prediction at all, even when the dropoff rate is very small. However, our RC with the $(D+1)$-dimension I/O scheme clearly improves the predictability, especially when θ is smaller, as shown by the red circle in the figure. The VPT shows a very similar dropoff rate dependence to that for the logistic map (as shown by the red circles in Fig. 2). The VPT becomes <1 when $\theta > 38\%$, i.e., the maximum dropoff rate is $\theta_\mathrm{m} = 38\%$. The overall magnitudes of the VPTs are lower than those for the logistic map.

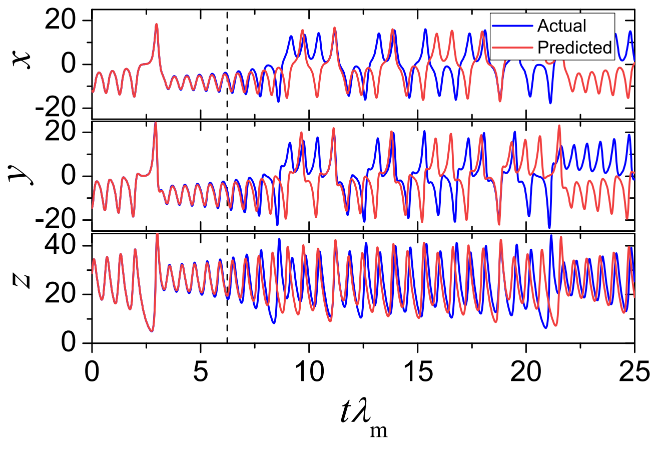

FIG. 6. Comparison between predicted and actual time series of the dynamic state of the Lorenz system when $\theta = 30\%$.

In Fig. 6, plotted are the comparisons between the predicted time series of the dynamical states and the actual ones. The vertical dashed-line marks the position of the VPT. The components of the dynamical states predicted by the RC scheme with $(D+1)$-dimension I/O are presented by the red lines. In this figure, one may find that the predicted states coincide with the actual ones very well before time $t\lambda_\mathrm{m}$ reaches the VPT. When the time is beyond the VPT, the x and y components exhibit significant deviation, and the z component shows phase synchronization with the actual dynamics. The reason is due to the Lyapunov exponents, which are a positive, zero, and a negative number along the x, y, and z directions, respectively. Therefore, going from the top to the bottom row of the graph, the deviation of the forecast state components against the actual ones reduces according to the order of the descent of the corresponding Lyapunov components.

The other example of the time-continuous dynamical system is the Rössler system, which is described by the following differential equations:

$$\dot{x} = -y - z, \quad \dot{y} = x + ay, \quad \dot{z} = b + (x-c)z, \tag{28}$$

where $a = 0.5$, $b = 2.0$, and $c = 4.0$, at which the system is chaotic. The Lyapunov exponents are $\{\lambda_1, \lambda_2, \lambda_3\} = [0.12, 0, -2.9]$. The hyperparameters for the RC are $\rho = 0.5$, $\beta = 0.15$, and $\alpha = 1 \times 10^{-9}$.

Figure 7 shows the dependencies of the VPTs for the Rössler system by two schemes, in which a similar climate but with more significant improvement is observed when compared with those in Fig. 5. The maximum dropoff rate at which the VPT is <1 occurs at $\theta_\mathrm{m} = 52\%$. The time series of the predicted dynamic states display a similar picture to that of the Lorenz system, as shown in Fig. 8.

Comparing the results in Figs. 5 and 7 with those in Fig. 2, one may find that the overall enhancements on the predictability of this RC scheme trained by incomplete dynamic states of the three systems are in the following descending order, i.e., the Rössler system, the logistic map, and then the Lorenz system. The details of the characteristic quantities for the predictions are listed in Table I. It reveals that the maximum dropoff rates θ_m and the VPTs at the typical dropoff rates for

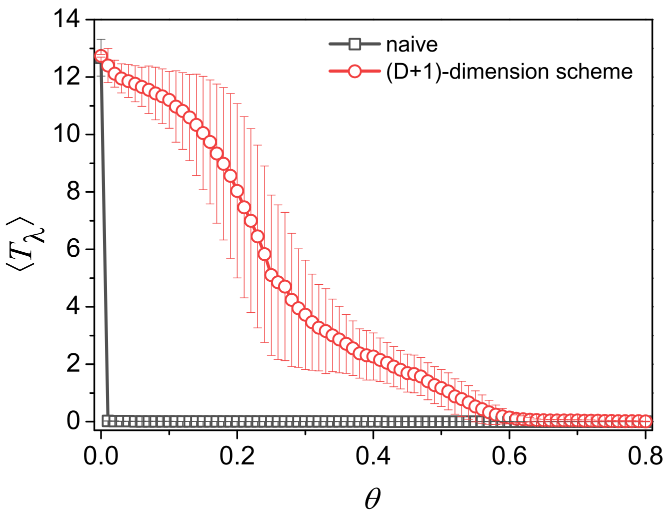

FIG. 7. The θ dependence of valid prediction times (VPTs) for the Rössler system. The black squares are the predictions by conventional reservoir computing (RC), and the red circles are those by our RC scheme with $(D+1)$-dimension input and output (I/O).

the three systems show roughly negative correlation with the values of the corresponding maximum Lyapunov exponents. Thus, the predictability is dependent on the structure and complexity of the chaotic attractors. In Fig. 9, we show the reconstructed Lorenz and Rössler attractors when $\theta = 30\%$. The plots reveal that the predicted orbits reproduce the climate of the original attractors very well, where the precision is $\sigma = 8 \times 10^{-3}$ and 3×10^{-4}, respectively.

C. KS system

As an illustrative model for a high-dimensional spatiotemporally chaotic system, we consider the KS equation, i.e.,

$$y_t = -yy_x - y_{xx} - y_{xxxx}, \tag{29}$$

where $y(x,t)$ is a scalar field defined in the spatial domain $0 \leqslant x \leqslant L$. The periodic boundary condition is employed such that $y(x,t) = y(x+L,t)$. Here, we choose $L = 60$, and the maximum Lyapunov exponent of the system is $\lambda_m = 0.089$. By numerically integrating Eq. (29) on a uniformly spaced grid of size $Q = 128$ with $\hat{\tau} = 0.25$, we obtain a simulated dataset with Q time series:

$$\mathbf{u}(t) = [y(\Delta x,t), y(2\Delta x,t), \ldots, y(Q\Delta x,t)]^T, \tag{30}$$

where $\Delta x = L/Q$. When applying this RC scheme with $(D+1)$-dimension I/O to this system, the time divides are set as follows: $N_0 = 50\,000$, $N_\mathrm{W} = 10\,000$, $N_\mathrm{T} = N_0(1-\theta) - N_\mathrm{W}$, and $N_\mathrm{P} = 5000$. The number of nodes in the reservoir is $D_\mathrm{r} = 5000$. The hyperparameters for the RC are $\rho = 0.4$, $\beta = 0.5$, and $\alpha = 1 \times 10^{-5}$.

In Fig. 10(a), plotted is the actual state of the KS system obtained by numerical integration, and the corresponding predicted state by our RC without missing data along the temporal axis in the input and the deviation of the predicted state from the true counterpart are shown in Figs. 10(b) and 10(c), respectively. One may see that this RC scheme with $(D+1)$-dimension I/O gives a pretty good short-term prediction, where the VPT is ~6 Lyapunov time. A visual inspection of the figure shows that the reservoir with this scheme may have learned the correct climate of the KS system even after the time duration is beyond the VPT. The result of the prediction by our RC with $D_\mathrm{r} = 5000$ in this paper shown in Fig. 10 agrees well with (or even better than) that by conventional RC with $D_\mathrm{r} = 9000$ in Ref. [4], which is to say that our RC is an effective scheme for predicting dynamics of the high-dimensional spatiotemporally chaotic system like the KS equation.

When 10% of the data along the temporal axis are missing, this RC scheme has also been successfully applied to predict the dynamics of the KS system. The corresponding results

TABLE I. Comparison of the characteristic quantities among the logistic map and the Lorenz and Rössler systems.

Characteristic quantities	Logistic map ($\lambda_\mathrm{m} = 0.616$)	Lorenz system ($\lambda_\mathrm{m} = 0.90$)	Rössler system ($\lambda_\mathrm{m} = 0.12$)
θ_m	50%	37%	51%
$\langle T_\lambda \rangle\rvert_{\theta=0\%}$	11.9	11.9	12.7
$\langle T_\lambda \rangle\rvert_{\theta=1\%}$	9.6	11.4	12.4
$\langle T_\lambda \rangle\rvert_{\theta=10\%}$	7.0	6.2	11.2
$\langle T_\lambda \rangle\rvert_{\theta=20\%}$	4.1	2.7	8.0
$\langle T_\lambda \rangle\rvert_{\theta=30\%}$	2.4	1.9	3.7

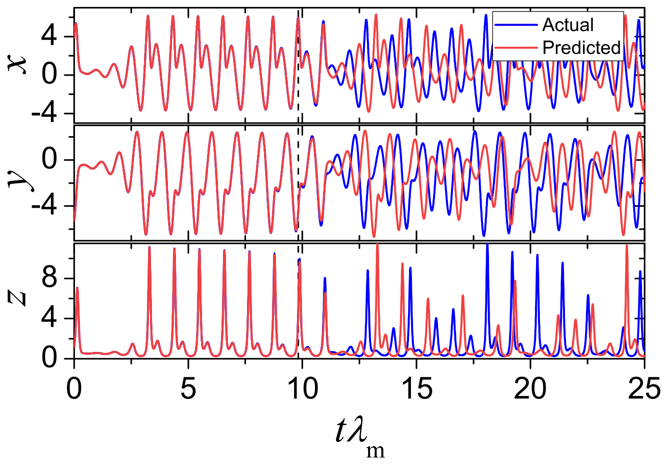

FIG. 8. Comparison between predicted and actual time series of the dynamic state of the Rössler system when $\theta = 30\%$.

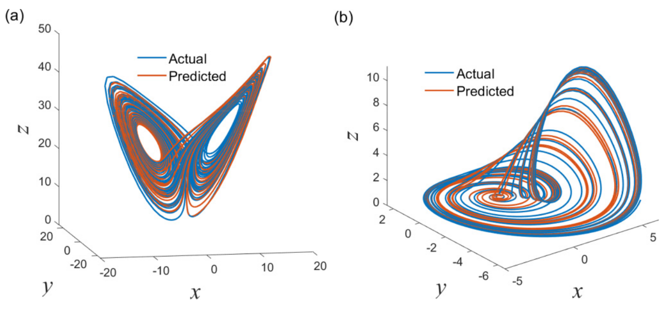

FIG. 9. The reconstruction of chaotic attractors. (a) Lorenz attractor. (b) Rösslor attractor.

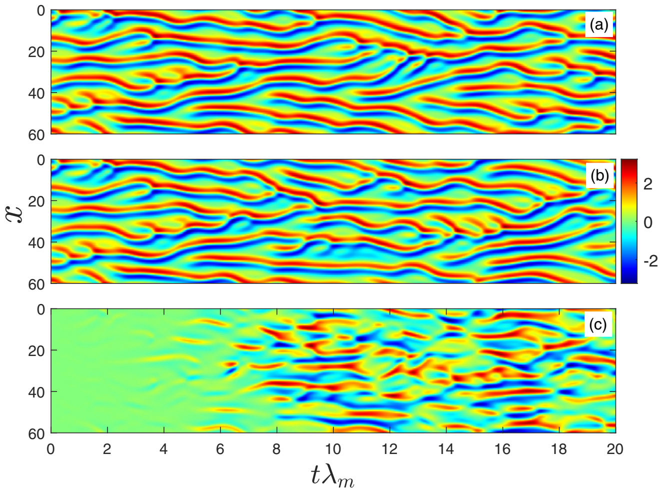

FIG. 10. Prediction of the Kuramoto-Sivashinsky (KS) system by our reservoir computing (RC) when $\theta = 0\%$. (a) True state of the KS system. (b) Reservoir prediction. (c) Error [(b) minus (a)] in the reservoir prediction.

are shown in Fig. 11, where (a) is the actual state of the KS equation, (b) is the predicted state based on the input data with the dropoff rate $\theta = 10\%$, and (c) is the deviation of the prediction from the actual state. One may find that the short-term forecasting is very good where the VPT is ~4 Lyapunov time. Obviously, the current RC scheme can also replicate the long-term climate of the KS system, even when the data drop happens. The VPT will decrease to 2 Lyapunov time when we further increase the dropoff rate to $\theta = 20\%$. The maximum dropoff rate happens at $\theta_m = 25\%$ when VPT equals 1. We must point out that θ_m and the VPTs at typical dropoff rates of the KS system are smaller than those of the time-continuous systems illustrated in the previous section. The reason is that the spatial dimension in the KS system is very large, and thus,

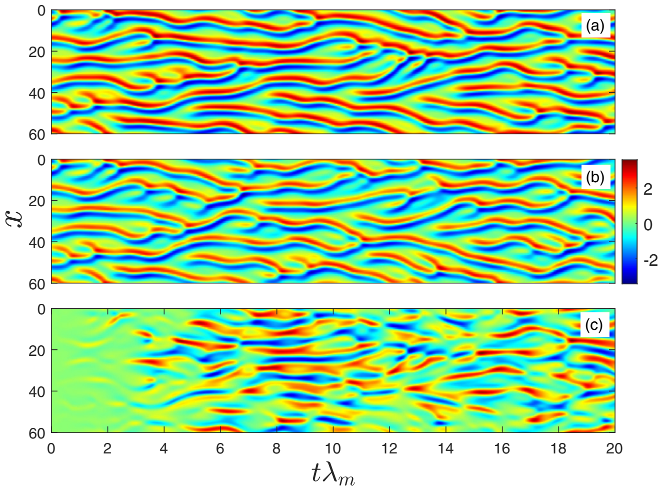

FIG. 11. Prediction of the Kuramoto-Sivashinsky (KS) system by our reservoir computing (RC) when $\theta = 10\%$. (a) True state of the KS system. (b) Reservoir prediction. (c) Error [(b) minus (a)] in the reservoir prediction.

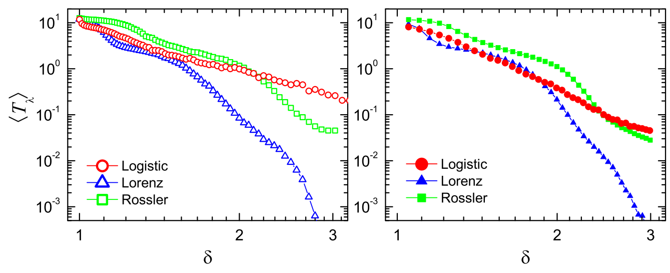

FIG. 12. δ dependence of the mean valid prediction time (VPT). (a) For the exponential distribution of time intervals. (b) Poisson distribution.

the system is more complex and intractable. More nodes of the reservoir are required to make a good prediction.

D. Effect of the time interval and its distribution on the prediction ability

The effect of the time interval distribution is an important problem, especially when we try to apply our RC to the case of nonuniform sampled intervals instead of missing data. The results mentioned previously are based on the incomplete data produced by the way stated in Sec. II D, where it is produced by randomly removing data from a complete time series with a uniform probability. The distribution of the time interval τ can be analytically obtained, which is

$$P(\tau) = (1 - \theta)\theta^{\tau - 1}. \tag{31}$$

It is an exponential distribution governed merely by the dropoff rate θ. The analytical expressions of the other statistical quantities are

$$\delta = \langle \tau \rangle = 1/(1 - \theta), \tag{32}$$

$$\sigma_n = \sqrt{\theta}/(1 - \theta), \tag{33}$$

$$P_1 = P(\tau = 1) = 1 - \theta = \delta^{-1}. \tag{34}$$

Here, δ is the mean time interval, σ_n is its standard error, and P_1 is the portion of time intervals which equal 1. All these quantities are uniquely determined by the dropoff rate. Hence, the decreasing of the mean VPTs with respect to the increasing of θ in the previous subsections can easily be changed into the mean interval or even its standard error dependence. In Fig. 12(a), the circles, triangles, and squares display the mean interval dependence of $\langle T_\lambda \rangle$ for the logistic map and Lorenz and Rössler systems, respectively. The obvious decreases of mean VPTs as the increasing of mean time interval are observed. This can be easily understood by the following argument. In conventional RC, predicting the dynamics is to reconstruct trajectories of the first-order iteration, in which the input with uniform time intervals contains only the first-order iterations, while in this paper, the time intervals are not uniformly sampled; not only does the input carry the first-order iteration but also the high-order ones since $\tau = 1$ and n ($n > 1$) correspond to the first- and high-order iterations, respectively. Thus, with the increasing of θ, the mean interval δ becomes >1, and P_1 reduces, as shown by the hollow circles in Fig. 13. The decreasing of P_1 is responsible

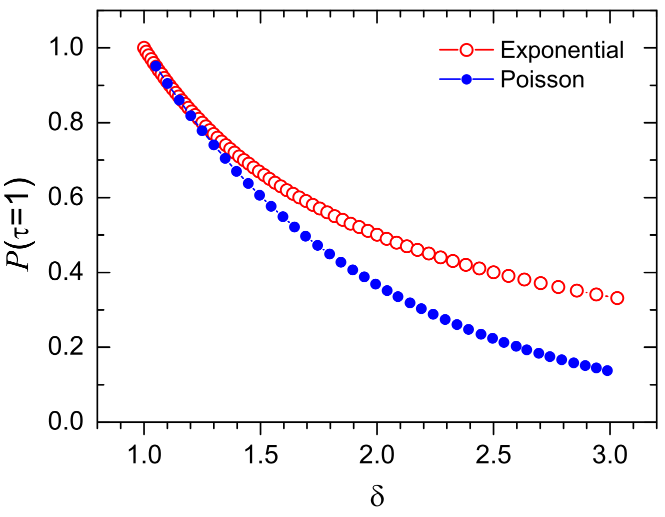

FIG. 13. δ dependence of $P(\tau = 1)$.

for the reduction of the mean VPT with the increasing of δ (or θ) since a good prediction requires a sufficient amount of samples. If $\langle T_\lambda \rangle \leqslant 1$ is set as a criterion for the bad forecast, we may get maximum mean intervals $\delta_m = 2.0$, 1.60, and 2.1, respectively. By Eqs. (34) and (32), we obtain the corresponding critical portions $P_{1c} = 0.50$, 0.63, and 0.49, and critical dropoff rates $\theta_m = 50\%$, 37%, and 51%, which are the same as those in Table I. This vision has been shown by Fig. 4, where the input contains only two kinds of intervals, i.e., $\tau = 1$ and 2, and the portion of the intervals which equal 1 is $P_1 = (1 - 2\theta)/(1 - \theta)$. Hence, as θ increases, the VPT of one-step-ahead prediction decreases, but that of two-step-ahead prediction increases.

To see the effect of the interval distribution, another sampling method is employed to produce the time intervals directly by the Poisson distribution. The distribution reads

$$P(\tau) = \frac{e^{-\mu}\mu^{\tau-1}}{\Gamma(\tau)}, \tag{35}$$

where $\Gamma(\cdot)$ is the Gamma function, and μ is the event rate, which determines the detail of the distribution. Then the other corresponding quantities are

$$\delta = \langle \tau \rangle = 1 + \mu, \tag{36}$$

$$\sigma_n = \sqrt{\mu}, \tag{37}$$

$$P_1 = P(\tau = 1) = e^{-\mu} = e^{-(\delta-1)}, \tag{38}$$

$$\theta = \mu/(1 + \mu) = 1 - 1/\delta. \tag{39}$$

The mean VPTs for the three systems are drawn in Fig. 12(b). Comparing with the plots in (a), we found that the mean VPTs have lower values and show faster decay with the increase of the mean time interval. These can also be explained by the smaller value of P_1 and its steeper decay, as shown by the solid circles in Fig. 13. Similarly, if we set $\langle T_\lambda \rangle \leqslant 1$ as a criterion, we get $\delta_m = 1.67$, 1.7, and 2.0, respectively. Then according to Eq. (38), we get the corresponding critical portions $P_{1c} = 0.50$, 0.48, and 0.36. The differences on the decay rate of VPTs can be clearly illustrated if we aggressively fit the data in the figures into a scaling relation $\langle T_\lambda \rangle \propto \delta^{-k}$. In the case of exponential distribution, the slopes in the double logarithm plots are $k = 3.67 \pm 0.04$, 9.8 ± 0.2, and 5.34 ± 0.08, respectively, while for Poisson distribution, the corresponding slopes are $k = 5.53 \pm 0.04$, 10.8 ± 0.4, and 6.8 ± 0.2.

In both kinds of distributions, there is a simple relation between the mean time interval and the dropoff rate, i.e., $\theta = 1 - 1/\delta$. The above results imply that the time interval distribution shows a significant influence on the portion of the time interval that equals 1 and thus affects the forecast skill of the RC.

IV. CONCLUSIONS

We have proposed a RC with $(D + 1)$-dimension I/O vectors, in which the first D dimensions store the state vector, and the additional dimension is the corresponding time interval. This scheme is a pretty good generalization to RC and is easy to use since it does not change the basic framework of conventional RC except for increasing an additional dimension to the I/O vectors. Containing both the state and time information in the input vector enables the reservoir to learn the whole knowledge of the dynamics of system, and thus, it can make effective prediction based on complex input. Here, the complex input can either be the ones with regularly or randomly missing data or data series created by irregular sampling, in general, the time series with nonuniform time intervals. Not only can it handle the problems that conventional RC can but also the ones of incomplete and complex inputs with irregular time intervals. Furthermore, this scheme can do multistep-ahead prediction just by changing the time interval in the output vector to a desired value, which is superior to conventional RC that can only do one-step-ahead forecasting based on regular input data. The applications of this scheme to the logistic map and Lorenz, Rössler and KS systems reveal that it can effectively predict the future evolution of the time-discrete and time-continuous chaotic dynamical systems and the spatiotemporally chaotic system, based on either complete or incomplete input. We have acquired quite a long VPT for the short-term prediction and a satisfactory long-term prediction, i.e., the reconstruction of map functions or attractors. The quality and efficiency of the prediction strongly depends on the structure and complexity of chaotic attractors. In addition, the effect of the time interval distribution on the prediction is discussed. The result reveals that it has a significant influence on the forecast skill, and the portion of intervals which equal one is the main cause to govern the variation of VPTs at different dropoff rates. Meanwhile, it is found that the mean time interval dependence of the mean VPT obeys the scaling law at least for the logistic map.

ACKNOWLEDGMENTS

This paper is supported by the National Natural Science Foundation of China under Grant No.11975144. SXQ's gratitude is due to Prof. Ying-Cheng Lai for introducing the field of machine learning.

[1] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks, GMD Report 148, German National Research Center for Information Technology (2001).
[2] W. Maass, T. Natschläger, and H. Markram, Neural Comput. **14**, 2531 (2002).
[3] H. Jaeger and H. Hass, Science **304**, 78 (2004).
[4] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, Chaos **27**, 121102 (2017).
[5] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Phys. Rev. Lett. **120**, 024102 (2018).
[6] P. Werbos, Proc. IEEE **78**, 1550 (1990).
[7] R. J. Williams and D. Zipser, Neural Comput. **1**, 270 (1989).
[8] L. Appeltant, M. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. Mirasso, and I. Fischer, Nat. Commun. **2**, 468 (2011).
[9] N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, Phys. Rev. E **91**, 020801(R) (2015).
[10] G. Tanaka, T. Yamane, J. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Neural Networks **115**, 100 (2019).
[11] K. Srinivasan, N. Coble, J. Hamlin, T. Antonsen, E. Ott, and M. Girvan, Phys. Rev. Lett. **128**, 164101 (2022).
[12] S. Baur and C. Räth, Phys. Rev. Res. **3**, 023215 (2021).
[13] B. Penkovsky, X. Porte, M. Jacquot, L. Larger, and D. Brunner, Phys. Rev. Lett. **123**, 054101 (2019).
[14] A. Röhm, D. J. Gauthier, and I. Fischer, Chaos **31**, 103127 (2021).
[15] P. Antonik, M. Gulina, J. Pauwels, and S. Massar, Phys. Rev. E **98**, 012215 (2018).
[16] X. Chen, T. Weng, H. Yang, C. Gu, J. Zhang, and M. Small, Phys. Rev. E **102**, 033314 (2020).
[17] M. U. Kobayashi, K. Nakai, Y. Saiki, and N. Tsutsumi, Phys. Rev. E **104**, 044215 (2021).
[18] S. Borlenghi, M. Boman, and A. Delin, Phys. Rev. E **98**, 052101 (2018).
[19] R. Martinenghi, S. Rybalko, M. Jacquot, Y. K. Chembo, and L. Larger, Phys. Rev. Lett. **108**, 244101 (2012).
[20] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, Chaos **27**, 041102 (2017).
[21] R. S. Zimmermann and U. Parlitz, Chaos **28**, 043118 (2018).
[22] H. Fan, J. Jiang, C. Zhang, X. Wang, and Y.-C. Lai, Phys. Rev. Res. **2**, 012080(R) (2020).
[23] M. H. Trauth, J. C. Larrasoa, and M. Mudelsee, Quat. Sci. Rev. **28**, 399 (2009).
[24] L. Cheng, K. E. Trenberth, J. Fasullo, T. Boyer, J. Abraham, and J. Zhu, Sci. Adv. **3**, e1601545 (2017).
[25] A. Wikner, J. Pathak, B. R. Hunt, I. Szunyogh, M. Girvan, and E. Ott, Chaos **31**, 053114 (2021).
[26] G. A. Gottwald and S. Reich, Chaos **31**, 101103 (2021).
[27] N. Chen and Y. Li, Chaos **31**, 113114 (2021).
[28] J. Brajard, A. Carrassi, M. Bocquet, and L. Bertino, J. Comput. Sci. **44**, 101171 (2020).
[29] T. Weng, H. Yang, C. Gu, J. Zhang, and M. Small, Phys. Rev. E **99**, 042203 (2019).
[30] L.-W. Kong, H.-W. Fan, C. Grebogi, and Y.-C. Lai, Phys. Rev. Res. **3**, 013090 (2021).
[31] R. Xiao, L.-W. Kong, Z.-K. Sun, and Y.-C. Lai, Phys. Rev. E **104**, 014205 (2021).
[32] J. Z. Kim, Z. Lu, E. Nozari, G. J. Pappas, and D. S. Bassett, Nat. Mach. Intell. **3**, 316 (2021).
[33] D. Patel, D. Canaday, M. Girvan, A. Pomerance, and E. Ott, Chaos **31**, 033149 (2021).
[34] C. Klos, Y. F. Kalle Kossio, S. Goedeke, A. Gilra, and R.-M. Memmesheimer, Phys. Rev. Lett. **125**, 088103 (2020).
[35] R. Falahian, D. Mehdizadeh, M. Molaie, S. Jafari, and S. Gharibzadeh, Nonlinear Dyn. **81**, 1951 (2015).
[36] C.-D. Han, B. Glaz, M. Haile, and Y.-C. Lai, Phys. Rev. Res. **3**, 023156 (2021).
[37] L. Grigoryeva and J.-P. Ortega, Neural Networks **108**, 495 (2018).
[38] G. Cybenko, Math. Control, Signals Systems **2**, 303 (1989).