On the Characteristics and Structures of Dynamical Systems Suitable for Reservoir Computing

Masanobu Inubushi, Kazuyuki Yoshimura, Yoshiaki Ikeda, and Yuto Nagasawa

Abstract We present an overview of mathematical aspects of Reservoir Computing (RC). RC is a machine learning method suitable for physical implementation, which harnesses a type of synchronization, called Common-Signal-Induced Synchronization. A precise criterion for this synchronization is given by a quantity called the conditional Lyapunov exponent. We describe a class of dynamical systems (physical systems) that are utilizable for RC in terms of this quantity. Then, two notions characterizing the information processing performance of RC are illustrated: (i) Edge of Chaos and (ii) Memory-Nonlinearity Trade-off. Based on the notion (ii), a structure of dynamical systems suitable for RC has been proposed. This structure is called the mixture reservoir. We review the structure and show its remarkable information processing performance.

1 Introduction

Recent developments in computer software, in particular machine learning, have remarkably improved the information processing accuracy such as in speech and image recognition tasks. On the other hand, in a research field of computer hardware, it is expected that an ultrafast computer can be realized based on a novel principle, such as a quantum computer. **Reservoir computing (RC)** is a machine learning method suitable for hardware implementation (Jaeger and Haas 2004). Indeed, many researchers are now actively implementing RC with various physical systems such as optoelectronic systems (Appeltant et al. 2011; Larger et al. 2017; Nakajima et al. 2018; Nakane et al. 2018; Takano et al. 2018), quantum systems (Fujii and Nakajima 2017), and soft materials (Nakajima et al. 2015). It is expected that utilizing physical characteristics for RC appropriately may lead to an innovative "computer" achiev-

M. Inubushi (✉)
Department of Applied Mathematics, Tokyo University of Science, Tokyo 162 - 8601, Japan
e-mail: inubushi@rs.tus.ac.jp

K. Yoshimura · Y. Ikeda · Y. Nagasawa
Department of Information and Electronics, Graduate school of Engineering, Tottori University, Tottori 680 - 8552, Japan

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_5

ing ultrafast processing speed with low energy consumption (see other chapters for various physical implementations).

The physical systems utilized for RC are modeled as **nonlinear dynamical systems**, and dynamical system theory (Morris et al. 2012) and nonlinear physics provide us useful mathematical tools. So far, these tools have enabled us to find some characteristics of RC as we introduce below. Nevertheless, at present, our understanding of the working principles behind RC is far from complete. For instance, we don't have any clear answer to the following fundamental question: "*For a given task, what characteristics and structures of a dynamical system are crucial for RC?*" or "*Does there exist a universal law governing the information processing ability of a dynamical system for RC less dependent on a task?*" If we find clear answers to the above questions, they are not only of theoretical interest but also practically useful since they will give design guidelines of RC. Unfortunately, at the present stage, various physical systems are employed for RC blindly without design guidelines. So, it is quite important to find the answers.

Here, focusing on characteristics and structures of dynamical systems suitable for RC, we overview the mathematical tools and theoretical results reported so far. In Sect. 2, we illustrate a mathematical formulation of RC including its general framework, the surprisingly simple learning method, a class of dynamical systems usable for RC, and a concrete example. In Sect. 3, two notions characterizing the information processing performance of RC are illustrated: (i) Edge of Chaos and (ii) Memory-Nonlinearity Trade-off. Based on the notion (ii), we introduce an effective structure of dynamical systems for RC with some numerical results.

While RC originated from neuroscience, recently a wide variety of physical phenomena are employed experimentally for the implementations. From a theoretical viewpoint, understanding of RC would need a multidisciplinary approach from mathematics, computational science, nonlinear science, neuroscience, and statistical physics. Here, we would like to introduce an aspect of this attractive research field.

2 Mathematical Formulation of Reservoir Computing

First, we formulate the framework of RC in an abstract form, and then, introduce a concrete example. Let a set of two sequences $D = \{s(k), y(k)\}_{k=1}^{K}$ be given, where $s(k)$ and $y(k)$ are one-dimensional input and output signals at time k and there exists some relation between these two sequences. The data set D is referred to as training data. The goal of RC is to learn the relation between s and y from the training data D, and then, after the training period $k > K$, give an estimation $\hat{y}(k)$ for predicting an unknown output $y(k)$ that corresponds to a given new input data $s(k)$. This is a typical supervised learning and the above estimation is called generalization. Considering time series prediction as an example, the goal of RC is to predict $s(k + \tau)$ corresponding to the given input sequence $\{s(i)\}_{i \leq k}$, i.e., $y(k) = s(k + \tau)$, which is called τ-step ahead prediction.

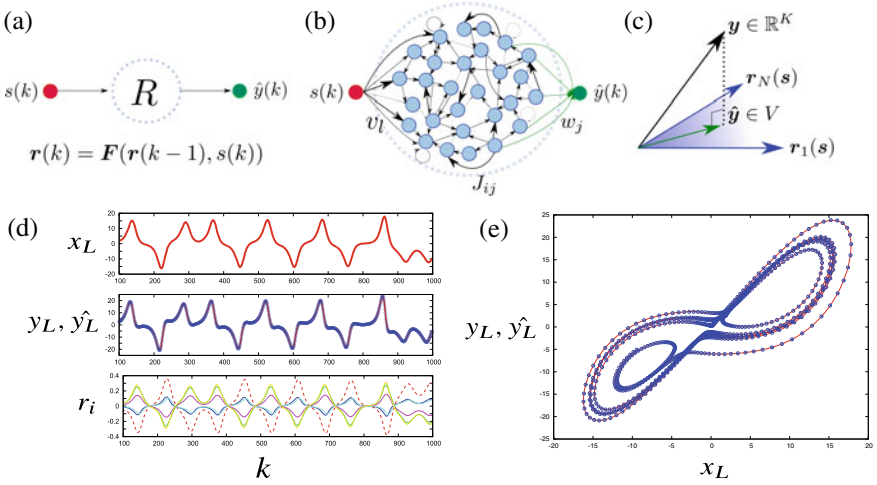

Fig. 1 **a**, **b** Illustration of RC framework. The red circle represents the input node, the blue dot circle represents the reservoir, and the green circle represents the output node. **c** Illustration of the vectors y, $\hat{y}$. **d** A demonstration of RC. The task is to infer the variable $y_L(k)$ of the Lorenz system from a given sequence of the variable $\{\ldots x_L(k-1), x_L(k)\}$ of the same Lorenz system. From top to bottom, the variables x_L, y_L, $\hat{y_L}$ which is inferred by the reservoir, and the time series of the reservoir variable $r_i (i = 1, \ldots, 5)$. **e** The projection of the orbit (the same data as **d**)

RC consists of the input/output nodes and a dynamical system driven by the input sequence $\{s(k)\}$, called *reservoir*. In Fig. 1a, an illustration of the reservoir is shown as "R" encircled by the blue dot circle. The reservoir is described by an N-dimensional dynamical system as follows:

$$\boldsymbol{r}(k) = \boldsymbol{F}(\boldsymbol{r}(k-1), s(k)), \tag{1}$$

where $\boldsymbol{r}(k) \in \mathbb{R}^N$ represents a state of the reservoir at time k which we call reservoir state, and $\boldsymbol{F} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ is a map describing the dynamics of the reservoir. In the case of physical implementation, $\boldsymbol{F}$ is determined by the physical law. The reservoir state $\boldsymbol{r}(k) = \{r_i(k)\}_{i=1}^N$ evolves in time according to the map $\boldsymbol{F}$ with the input signal $s(k)$.

The approximation $\hat{y}(k)$ for the desired output $y(k)$ is obtained by "observation" of the reservoir state with linear weights:

$$\hat{y}(k) := \sum_{i=1}^{N} w_i r_i(k). \tag{2}$$

This output weight $\{w_i\}_{i=1}^N$ is optimized so that $\hat{y}(k) \simeq y(k)$ as explained below. The determination of the weight $\{w_i\}_{i=1}^N$ by the training data D is called learning. In short, RC is the information processing method with **learning output weight only**.

In other words, the evolution law, $\boldsymbol{F}$ in the Eq. (1), is not optimized but fixed. Considering the conventional training methods of neural network where both w and $\boldsymbol{F}$ are learned, it may be surprising that information processing can be performed by a task-independent, even randomly generated, evolution law $\boldsymbol{F}$. Since control of physical laws is difficult in general, physical implementations of the conventional neural networks, which require thousands of adjustable parameters and stable controls of them with high precision, are also difficult. On the other hand, the framework of RC does not require such control, and therefore is suitable for physical implementations.

The learning of the linear output weights $\boldsymbol{w}$ is just performed via the least squares method by using the training data D. Let $\boldsymbol{y} = (y(1), \ldots, y(K))^T$ be a sequence of the desired outputs and $\boldsymbol{w} = (w_1, \ldots, w_N)^T$ be a weight vector, and $\Phi_{kj} = r_j(k)$ which is the so-called design matrix. The linear readout (2) can be described as $\hat{\boldsymbol{y}} = \Phi \boldsymbol{w}$. From the condition to minimize the square error $E(w) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2$ where $\|\cdot\|$ is the l_2-norm, we obtain the set of equations $\partial_{w_i} E(\boldsymbol{w}) = 0$ $(i = 1, \ldots, N)$ and its solution as follows:

$$\boldsymbol{w} = (\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{y}. \tag{3}$$

What we need for the learning for RC is just to calculate the inverse matrix of $\Phi^T \Phi$ whose size is the number of the nodes, $N \times N$. Thus, the learning for RC is computationally cheap. While this formulation is similar to the basic least squares method, the difference is that the design matrix has information on the whole of history of the past input signals as $\Phi_{kj} = r_j(k) = r_j(s(k), s(k-1), \ldots)$. For the practically important tasks such as time series prediction or speech recognition, history of the past input signals is essential. Hence, RC enables us to solve the temporal task requiring the past input signals at a low computational cost.

2.1 A Class of Dynamical System Usable for Reservoir Computing: Common-Signal-Induced Synchronization

What kind of a dynamical system can we use for RC? The dynamical system utilized for RC should at least have a property that the same output sequence $\{y(k)\}_{k=1}^T$ is generated when repeatedly given the same input sequence $\{s(k)\}_{k=1}^T$, not depending on the initial condition of the dynamical system. If we use a dynamical system lacking this property, we have different outputs from RC for the same tasks, depending on the initial state of the reservoir. Considering speech recognition tasks, the reservoir computer is useless if it recognizes completely different words for the same voice input.

In order to formulate this property, let us consider two different initial conditions $\boldsymbol{r}(0)$, $\tilde{\boldsymbol{r}}(0)$ of the reservoir states. These states evolve in time according to the map $\boldsymbol{F}$ in Eq. (1) with a common input signal $s(k)$. If these different states always converge to

the same state, not depending on the initial conditions, i.e., $\|\boldsymbol{r}(k) - \tilde{\boldsymbol{r}}(k)\| \to 0\ (k \to \infty)$, we say the signal-driven dynamical system (1) shows **common-signal-induced synchronization (CSIS)**. In other words, there exists a certain synchronized state $\boldsymbol{r}^*(k)$ evolving in time only depending on the input sequence. The synchronized state attracts states in its neighborhood, i.e., asymptotically stable as mentioned later. The input sequence determines the synchronized state $\boldsymbol{r}^*(k)$ except the initial transient period, and also determines uniquely the output sequence: $\hat{y}(k) = \sum_i w_i r_i^*(k)$. This kind of reproducibility to the input signal is also referred to as *Echo State Property* (Maass and Markram 2002) or *Consistency* (Uchida et al. 2004). Any dynamical system possessing this property can be used for reservoir computing in principle.

This property is exhibited by not only a particular dynamical system but also general dynamical systems in the following sense. For instance, it can be shown easily that any dynamical system with a stable fixed point as $\boldsymbol{r}(k) = A\boldsymbol{r}(k-1) + \boldsymbol{s}(k)$ $(\|A\| < 1)$ shows CSIS. It has been shown that the CSIS occurs in a variety of limit-cycle oscillators (Teramae and Tanaka 2004) and one-dimensional delay dynamical systems (Yoshimura et al. 2008a) when they are driven by the white noise input. Moreover, for the colored noise, the previous study (Yoshimura et al. 2007) has shown that the CSIS occurs for the various dynamics including the limit cycle, chaotic dynamics, and one-dimensional time delay system. Considering there exist stable fixed points and limit cycles in many dissipative dynamical systems, we can use various physical systems as a reservoir for a resource of information processing.

A mathematical tool from the dynamical system theory, called **conditional Lyapunov exponent**, is useful to characterize CSIS (Louis et al. 1991; Teramae and Tanaka 2004; Yoshimura et al. 2008a, b). The conditional Lyapunov exponent is defined by the exponential growth/decay rate of the infinitesimal perturbation $\delta\boldsymbol{r}(k)$ to the state $\boldsymbol{r}(k)$ as follows:

$$\lambda := \lim_{T\to\infty} \frac{1}{T} \ln \|\delta\boldsymbol{r}(T)\|, \tag{4}$$

where $\delta\boldsymbol{r}(k)$ is determined by the variational equation

$$\delta\boldsymbol{r}(k) = DF_{\boldsymbol{r}(k-1),s(k)}\delta\boldsymbol{r}(k-1) \tag{5}$$

and $DF_{\boldsymbol{r}(k-1),s(k)}$ denotes the Jacobian matrix of $\boldsymbol{F}$ evaluated at $(\boldsymbol{r}(k-1), s(k))$. Equation (4) can be interpreted as $\|\delta\boldsymbol{r}(k)\| \propto \exp(\lambda k)$, and thus $\lambda < 0$ implies the asymptotic stability, i.e., the common-signal-induced synchronization. Note that the conditional Lyapunov exponent is a characteristic value with respect to the invariant measure (distribution) which is determined by not only the dynamical system, the map $\boldsymbol{F}$, but also the stochastic property of the input sequence, $s(k)$. The Lyapunov exponent introduced above is *conditioned* by the stochastic property of $s(k)$, and thus, we say the conditional Lyapunov exponent. The characterization of the CSIS by the conditional Lyapunov exponent has been introduced and often used in the field of nonlinear physics, and recently, the random dynamical system theory in the

field of mathematics has justified it rigorously under some conditions (Flandoli et al. 2017). In conclusion, any signal-driven dynamical system with a negative conditional Lyapunov exponent can be used for RC.

2.2 Concrete Example: Echo State Network Model

Echo State Network (ESN) is one of the standard structures of the reservoir that uses a recurrent neural network as shown in Fig. 1b. Let k $(= 1, 2, \ldots)$ be the time and $r_i(k)$ be the state of the ith node $(i = 1, 2, \ldots, N)$ at time k. The reservoir state evolves in time as follows:

$$r_i(k) = \phi\left[\sum_{j=1}^{N} J_{ij} r_j(k-1) + v_i s(k)\right], \tag{6}$$

where $\phi[\cdot]$ is called the activation function[1] and typically $\phi[u] = \tanh gu$ is used. $g \in \mathbb{R}$ is a parameter. v_i and J_{ij} are connection weights between nodes in the network, and importantly, the values of them are fixed once they are determined by random numbers at the initial. Supervised machine learning is applied only to the output weight $\{w_i\}_{i=1}^{N}$ as noted before.

As an example, we demonstrate a result of an inference task solved by the ESN model in Fig. 1d, e. The task is to infer the variable y_L of the Lorenz equation[2] from another variable x_L of the Lorenz equation at the same time (Lu et al. 2017). In other words, the input and the output are $s(k) = x_L(k)$ and $y(k) = y_L(k)$, respectively. The Lorenz equation is deterministic, and there exists some relation between these variables, x_L and y_L. ESN learns the relation from the data set $D = \{x_L(k), y_L(k)\}_{k=1}^{K}$, and determines the readout weight $\boldsymbol{w}$ for the inference. We remark that the variable $y_L(k)$ is determined not solely by the variable $x_L(k)$ but the sequence of the variables: $\{\ldots, x_L(k-1), x_L(k)\}$. In other words, generally, RC can approximate the function of the sequence of the input signal as $y(k) = \mathcal{F}\Big(\{\ldots, s(k-1), s(k)\}\Big)$. The top figure in Fig. 1d shows the time series of $x_L(k)$, and the middle shows those of $y_L(k)$ and $\hat{y_L}(k)$. Learning only the readout weight leads to the almost perfect inference. As a reference, the time series of the reservoir variable $r_i (i = 1, \ldots, 5)$ are shown in the bottom figure in Fig. 1d.

---

[1] Note that the activation function ϕ differs from the design matrix Φ.

[2] The Lorenz equation is a simplified model of the thermal convection: $\dot{x} = -\sigma x + \sigma y$, $\dot{y} = rx - y - xz$, $\dot{z} = -bz + xy$, where $\sigma, r, b \in \mathbb{R}$ are the parameters, and, in the main text and later, the subscript L represents the variable of the Lorenz equation. The Lorenz equation has been well studied in the field of nonlinear physics and mathematics as a simple continuous dynamical system exhibiting chaos. In the research field of reservoir computing, the Lorenz equation is used for the time series prediction task, e.g., the input is $s(t) = x_L(t)$ and the output is $y(t) = x_L(t + \tau)$ $(\tau > 0)$, and the inference task of the hidden variable, e.g., the input is $s(t) = x_L(t)$ and the output is $y(t) = z_L(t)$ (Lu et al. 2017; Pathak et al. 2017).

2.3 Geometrical Interpretation of Reservoir Computing

The geometrical interpretation can give insight into the fundamental aspect of RC, in particular shedding light on the difference between RC and the conventional method using the recurrent neural network.

Here, let us consider the readout vector $\hat{\boldsymbol{y}} = \Phi \boldsymbol{w} = \sum_i w_i \boldsymbol{r}_i$ geometrically, where $\boldsymbol{r}_i = (r_i(1), \ldots, r_i(K))^T$. The vector $\hat{\boldsymbol{y}}$ can be regarded as the projection of the vector $\boldsymbol{y} \in \mathbb{R}^K$ to the N-dimensional subspace $V = \mathrm{Span}\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N\}$ as follows:

$$\hat{\boldsymbol{y}} = \Phi(\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{y} =: P\boldsymbol{y}, \tag{7}$$

where $P : \mathbb{R}^K \to V$ and we assume $K > N$ since in practice the number of the data set is larger than that of nodes in the network or the dimension of the dynamical system utilized for the reservoir.

The reservoir dynamics, where the common-signal-induced synchronization must occur, is determined by the input sequence except the initial transient period, the state vector $\boldsymbol{r}_i$ is a function of "input vector" $\boldsymbol{s} = (s(1), s(2), \ldots, s(K))^T$, i.e., $\boldsymbol{r}_i = \boldsymbol{r}_i(\boldsymbol{s})$, and the subspace V is also the same $V = V(\boldsymbol{s})$. Considering ESN as a standard model of RC, it can be interpreted that the randomness of the connection weights J_{ij}, v_i varies the response of each node state to the input signal, and enhances the "degree" of linear independence of the vectors $\boldsymbol{r}_i (i = 1, \ldots, N)$. See the illustration of Fig. 1c. As an extreme case, if the connection weights are not random, more precisely J_{ij} and v_i take respective constant values which are independent of i and j, and all nodes in the network become the same state $\boldsymbol{r} \equiv \boldsymbol{r}_i$. In that case, the dimension of the subspace V shrinks to one, and apparently, the approximation by the projection onto V results in failure.

In the conventional method of the recurrent neural network, each connection weight of J_{ij}, v_i is trained as well as the readout weight $\boldsymbol{w}$. This corresponds to generate an appropriate subspace $V'(D)$ spanned by the basis vectors $\boldsymbol{r}_i$ which are determined by the training data D. Since training J_{ij}, v_i requires a high computational cost, it is more difficult to employ a large number of nodes compared with the method of RC. Therefore, in the conventional method of the recurrent neural network, the vector $\boldsymbol{y}$ is approximated within the "well-trained low-dimensional" subspace $V'(D)$. On the other hand, in the method of RC, the vector $\boldsymbol{y}$ is projected onto the "randomly generated high-dimensional" subspace $V(\boldsymbol{s})$.

When a large amount of data is available for the training, it would be natural to assume $K \gg N$. In that case, projection of the vector $\boldsymbol{y} \in \mathbb{R}^K$ to the randomly generated N-dimensional subspace V would be expected to result in a poor approximation. However, even in such a case, RC solves the task well. In order to clarify the reason, it would be necessary to take into account the fact, at least, that (i) the well-solved task by RC has a specific structure, e.g., $y(k)$ does not depend on a long past input $s(k - \tau)$ $(\tau \gg 1)$, and (ii) the response characteristics of the reservoir dynamics, e.g., $r_i(k)$ does not depend on a long past input as well.

3 Characteristics of Dynamical Systems Suitable for Reservoir Computing

Here we introduce two notions (empirical laws), Edge of Chaos and Memory-Nonlinearity Trade-off.

3.1 Edge of Chaos

The conditional Lyapunov exponent is useful not only for conditioning dynamical systems usable for the reservoir, but also for characterizing *good* dynamical systems for the reservoir with respect to information processing. Edge of Chaos is an empirical law for the good dynamical systems in the following sense. The Edge of Chaos law has been supported by many numerical and physical experiments (Bertschinger 2004; Boedecker et al. 2012).

Let us consider two dynamical systems driven by a common input signal. By gradually changing a parameter of the dynamical system, it is often observed that a transition from a synchronized state ($\lambda < 0$) to an asynchronized state ($\lambda > 0$) occurs. For the asynchronized state, a distance between nearby orbits diverges exponentially even if the input signals are the same. The asynchronized state is similar to chaos in deterministic dynamical systems, and hereafter, we refer to it as chaos. For instance, in the ESN model, we observe the transition by increasing the spectral radius [3] of the connection matrix J_{ij}. Edge of Chaos is the empirical law where state RC achieves its highest performance in information processing *slightly before* the transition point, i.e., $\lambda < 0$ and $\lambda \simeq 0$. Many researchers have reported results of numerical and physical experiments supporting Edge of Chaos[4] for various reservoirs and tasks (Bertschinger 2004; Boedecker et al. 2012).

Here we demonstrate the Edge of Chaos law. Figure 2a shows a result of a function approximation task $y(k) = \sin(s(k-1))$ solved by the ESN model. In the upper panel, we show the normalized mean square error (NMSE) in the vertical axis as a standard value for performance evaluation of RC (Appeltant et al. 2011; Inubushi and Yoshimura 2017; Rodan and Tino 2011):

$$\mathrm{NMSE} = \frac{\langle (y(k) - \hat{y}(k))^2 \rangle_T}{\langle (y(k) - \langle y \rangle_T)^2 \rangle_T} \tag{8}$$

[3] For the matrix $J \in \mathbb{R}^{N \times N}$, the definition of the spectral radius is $\rho := \max_i |\mu_i|$ where $\{\mu_i\}_{i=1}^N$ are the eigenvalues of the matrix J. In the absence of the input signal, the origin of the ESN model $\boldsymbol{r} = \boldsymbol{0}$ is the stationary state (fixed point), and the stability of the stationary state is determined by the spectral radius ρ.

[4] The asynchronized state ($\lambda > 0$) is not *deterministic* chaos, and thus, the terminologies of "edge of criticality" or "edge of stability" would be more appropriate.

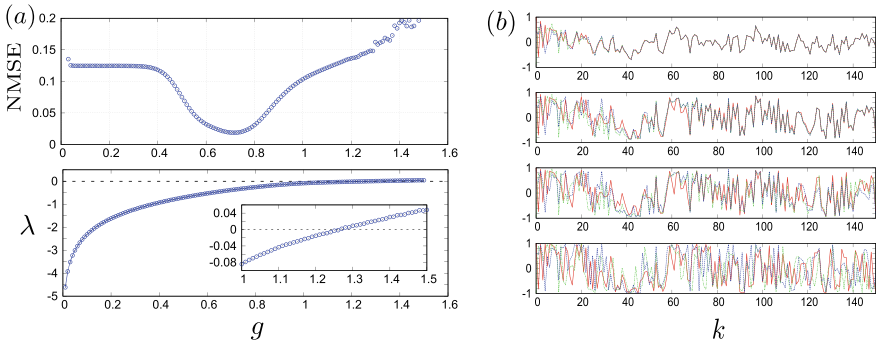

Fig. 2 **a** The upper panel: the normalized mean square error (NMSE) for the function approximation task, which is solved by the ESN model with $N = 100$ nodes. The horizontal axis is the parameter g of the ESN model. The vertical axis is NMSE. The lower panel: the conditional Lyapunov exponent. The horizontal axis is the same as in the upper panel. The transition from CSIS to chaos occurs at $g = 1.25$. **b** The time series of the first component of the state vector, $r_1^\sigma(k)$ $(\sigma = a, b, c)$, from the three different initial conditions $\boldsymbol{r}_1^\sigma(0)$. The time evolutions of ESNs are described by the same Eq. (6) with the common input signals, i.e., the differences are only in the initial conditions. From the top to the bottom, the parameter increases as $g = 1.0, 1.2, 1.3, 1.5$

where the brackets represent the time average $\langle z(t)\rangle_T := 1/T \sum_{t=1}^T z(t)$ for any sequence $\{z(t)\}_{t=1}^T$. The horizontal axis is the parameter g of the ESN model (6). NMSE takes the minimum value at $g \simeq 0.7$. In the lower panel of Fig. 2a, the conditional Lyapunov exponents λ corresponding to the upper panel are shown. The conditional Lyapunov exponent increases with increasing g, and, at $g \simeq 1.25$, we observe the transition from negative to positive, which corresponds to a transition from the synchronized state to the chaotic state. See the inset for the enlarged view around the transition point. The ESN model exhibits the maximum performance for the function approximation task at $g \simeq 0.7$, where the conditional Lyapunov exponent is in the vicinity of zero but a negative value ($\lambda \simeq -0.36$), i.e., slightly before the transition point. Our numerical example also supports the law of Edge of Chaos.

To see the transition from the synchronized state to the chaotic state in detail, we show time series of a variable of the same ESN model with three different initial conditions, i.e., $\boldsymbol{r}^a(0) \neq \boldsymbol{r}^b(0)$, $\boldsymbol{r}^b(0) \neq \boldsymbol{r}^c(0)$, and $\boldsymbol{r}^c(0) \neq \boldsymbol{r}^a(0)$. The numerical settings are the same as the case shown in Fig. 2a. Driven by the common signals, the variables $\boldsymbol{r}^\sigma(k)$ $(\sigma = a, b, c)$ evolve in time according to (6). As an example, the time series of the first component of the state vectors, $r_1^\sigma(k)$ $(\sigma = a, b, c)$, are shown in Fig. 2b. From the top to the bottom, the parameter g is changed as $g = 1.0, 1.2, 1.3, 1.5$.

The conditional Lyapunov exponents at $g = 1.0$ and $g = 1.2$ are negative as shown in Fig. 2a, and correspondingly, the convergence $|r_1^\sigma(k) - r_1^{\sigma'}(k)| \to 0$ $(\sigma \neq \sigma')$, i.e., CSIS, is observed in the upper two panels in Fig. 2b. Note that the convergence occurs at $g = 1.2$ slower than at $g = 1.0$, which is consistent with the absolute values of the conditional Lyapunov exponents. On the other hand, the conditional

Lyapunov exponents at $g = 1.3$ and $g = 1.5$ are positive as shown in Fig. 2a, and correspondingly, CSIS is no longer observed in these two cases in the lower two panels in Fig. 2b.

As demonstrated in Fig. 2b, the closer to the transition point the ESN is, the slower the convergence is, which is trivial by definition of the conditional Lyapunov exponent. It is believed that *the slow convergence leads to a large "memory capacity"*. For an illustration of this statement, let us assume the case that differences in the initial conditions $\boldsymbol{r}^\sigma(0)$ $(\sigma = a, b, c)$ in Fig. 2b are caused by the difference in the input signal $s(0)$. More precisely, these states evolve in time with the common input signal from the remote past, i.e., $\{s(-k)\}_{k=1}^{\infty}$. If the conditional Lyapunov exponent is negative, they should have converged to the same state $\boldsymbol{r}^\sigma(-1) \equiv \boldsymbol{r}(-1)$. And then, if there exist differences in the input signal at $k = 0$ denoted by $s^\sigma(0)$, the differences in the input signal make different states $\boldsymbol{r}^\sigma(0)$ $(\sigma = a, b, c)$ according to (6). Again, ESN receives the common signal $\{s(k)\}_{k=1}^{\infty}$, and the differences in the states dissipate; $\boldsymbol{r}^\sigma(k) \to \boldsymbol{r}(k)$ $(k \to \infty)$ as shown in the top two panels in Fig. 2b.

The memory capacity can be interpreted as an amount of information of the past input signal that can be reconstructed from the present reservoir state. Some working definitions of memory capacity, which measure the accuracy of the reconstruction of the past input signal from the present reservoir state, have been proposed and investigated so far (Jaeger 2002). The reconstruction needs at least the difference in the states. For instance, in the top panel in Fig. 2b, it is impossible to distinguish between the states $\boldsymbol{r}^\sigma(k)$ at $k = 100$, i.e., $\boldsymbol{r}^\sigma(k) \equiv \boldsymbol{r}(k)$. Therefore, at $k = 100$, it is impossible to reconstruct any information of $s^\sigma(0)$ from $\boldsymbol{r}^\sigma(k)$. In this case, it is natural to interpret that the state of ESN at $k = 100$ has no memory of, or forgets, the information in the past input signal $s^\sigma(0)$. As this example shows, the time T required for the convergence of the states dominates the memory in the sense that the ESN retains no memory about the past input signals supplied more than T ago. The time T is roughly estimated by a reciprocal of the conditional Lyapunov exponent $T \propto 1/\lambda$. In this sense, the memory has been referred to as *short term* memory as well.

Approaching the transition point, $\lambda \to -0$, makes the convergence slow, which would lead to a large memory capacity of ESN. That is, the ESN retains a memory of the past input signals of a long time ago. At Edge of Chaos, the memory capacity attains its maximum, which has been shown by using the dynamic mean field theory (Toyoizumi and Abbott 2011). The memory capacity is essential for solving a wide variety of tasks such as the speech recognition tasks and the time series prediction tasks (see the next subsection for an explicit example). Therefore, information processing with RC works well at Edge of Chaos.

Note that in the above argument, we only consider "necessary condition" for the memory. Namely, if ESN can store some memory, i.e., it is possible to reconstruct information about the past input $s^\sigma(0)$, then there exists a difference between the states $\boldsymbol{r}^\sigma(k)$. However, this does not imply the converse, i.e., "if there exists a difference between states, it is possible to reconstruct the past input", and also we cannot

say anything about the *accuracy* of the reconstruction.[5] In this sense, the problem of memory capacity is subtle. Information theoretic quantity such as the mutual information would be more useful than the conditional Lyapunov exponent as discussed in Inubushi and Yoshimura (2017).

3.2 Memory-Nonlinearity Trade-Off

While the memory of the input signal is important, information processing requires in general *nonlinear transformation* of the input signal as well. Here we focus our attention on the two "functions" of the reservoir necessary for information processing, i.e., the short term memory and the nonlinear transformation of the input signal. For the two functions, it is known that there exists a kind of trade-off based on the linearity/nonlinearity of the reservoir dynamics as follows.

Before explaining the trade-off, we introduce an explicit example of a task that requires both memory and nonlinear transformation. The task we consider is a time series prediction. Suppose that we need to predict a future value of a variable $z(t)$, $(z, t \in \mathbb{R})$, and that the variable $z(t)$ is described by an equation $\frac{dz}{dt} = G(z)$, where G is some nonlinear function. Given a time series, $\{\ldots, z(t-h), z(t)\}$ where h is a sampling period, the goal is to predict the future value $z(t+h)$. As one of the numerical schemes of ordinary differential equations, the Adams-Bashforth method has been often used. The explicit (two-step) formula is $z(t+h) = z(t) + h\left(\frac{3}{2}G(z(t)) - \frac{1}{2}G(z(t-h))\right) + O(h^3)$. Considering to predict the value $z(t+h)$ by RC with the Adams-Bashforth method in mind, the reservoir needs to store the memory $z(t-h)$ and perform the nonlinear transformation $G(z(t-h))$. Obviously, these two functions are both essential for the prediction; however, there is a trade-off as described below.

First, we introduce a property of the memory capacity of the input signal. Many researchers have reported so far that the linearity of the reservoir dynamics (the map $\boldsymbol{F}$

[5] In other words, CSIS state $\boldsymbol{r}(k)$ only depends on the recent past input signal, $\{s(k-j)\}_{j=0}^{T}$, and thus, the state vector $\boldsymbol{r}(k)$ is a function of the set of the input signals, i.e., $\boldsymbol{r}(k) = \boldsymbol{r}\left[\{s(k-j)\}_{j=0}^{T}\right]$. Using these notations, the above arguments can be expressed as follows: if the derivatives vanish,

$$\frac{\partial \boldsymbol{r}\left[\{s(k-j)\}_{j=0}^{T}\right]}{\partial s(k-K)} = \boldsymbol{0} \quad (\text{for}^{\forall} k) \tag{9}$$

then, in ESN at the time k, there is no information about $s(k-K)$. However, its inverse, i.e., "if the above derivative does not vanish, then it is possible to reconstruct the information about the past input $s(k-K)$", is not necessarily true. Considering the chaotic regime as an extreme case, $T \to \infty$ and the above derivative is not zero in general due to the sensitive dependence on the initial condition. However, it seems to be difficult to reconstruct the past input from the chaotic state. Hence, we cannot conclude the negation holds.

Table 1 Memory-Nonlinear Trade-off. In the column, the functions of the reservoir, the memory or the nonlinear transformation of the input signal, are shown. In the row, the types of the map $\boldsymbol{F}$ of the reservoir in (1) (the activation function ϕ in the ESN model (6)), linear or nonlinear, are shown

	Short term memory	Nonlinear transformation
Linear	◯	×
Nonlinear	×	◯

in (1)) is preferable for the large memory capacity (Dambre et al. 2012; Ganguli et al. 2018; Toyoizumi 2012). In other words, the nonlinearity of the reservoir dynamics reduces the memory capacity.

Next, we consider the nonlinear transformation of the input signal. Focusing only on the memory capacity, the best strategy seems to be just using the linear reservoir; however, apparently, the linear reservoir cannot perform the nonlinear transformation of the input signal by definition. One of the advantages of RC is to be able to solve linearly non-separable problems via mapping the input signal into a higher dimensional space in a nonlinear way. Thus, the nonlinearity of the reservoir dynamics plays an essential role in general information processing.

Strengthening the nonlinearity of the reservoir increases the ability of the nonlinear transformation of the input signal, but decreases the memory capacity. On the other hand, weakening the nonlinearity of the reservoir increases the memory capacity, but decreases the ability of the nonlinear transformation (Table 1). This relation between these two functions of the reservoir is referred to as *Memory-Nonlinearity Trade-off*, which has been supported numerically (Dambre et al. 2012; Inubushi and Yoshimura 2017; Verstraeten et al. 2010).

Here we give a numerical result illustrating the existence of Memory-Nonlinearity Trade-off. For simplicity, we employ the function approximation task $y(k) = \sin(\pi \nu s(k-\tau))$, where $\nu \in \mathbb{R}$, $\tau \in \mathbb{N}$ are the task parameters. The input signal $s(k)$ is the random variable which is independently and identically drawn from the uniform distribution: $\mathcal{U}(-1, +1)$ at each time t. Let us consider solving this task by the readout from the reservoir state at time k. To this end, two functions are needed: storing the information of the past input signal $s(k-\tau)$ for τ steps (short term memory), and approximating the sin function (nonlinear transformation). The task parameters (ν, τ) control, respectively, the "strength" of the nonlinear transformation and the memory capacity required for solving the task.

To confirm the trade-off summarized in Table 1, we use the ESN model described in (6) and compare the performance of ESNs with linear function $\phi[a] = a$ and the nonlinear function $\phi[a] = \tanh a$. We refer to ESNs with $\phi[a] = a$ and with $\phi[a] = \tanh a$, respectively, as linear ESN and nonlinear ESN. Figure 3a shows the results of the direct comparison in the task parameter space (ν, τ). For a given set of parameters (ν, τ), if the error with the linear ESN is lower than that with the nonlinear ESN, we mark a red square at (ν, τ) in the diagram. If the error with the nonlinear ESN is lower than that with the linear ESN, we mark a blue circle. See Inubushi and Yoshimura (2017) for the details of this diagram.

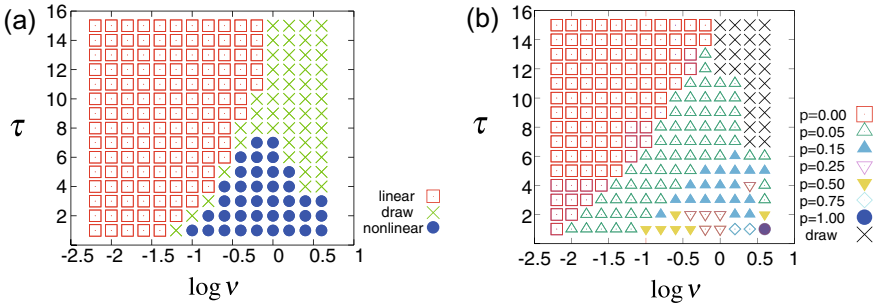

Fig. 3 Performance comparison diagram in task parameter space (ν, τ) (Inubushi and Yoshimura 2017). **a** Comparison between the linear ESN and nonlinear ESN for the demonstration of Memory-Nonlinearity Trade-off. **b** Comparison between the mixture ESNs with various mixture rates p which will be discussed in Sect. 4. Figure reproduced and modified with permission from Springer Nature

The results shown in Fig. 3a are summarized as follows: for the task requiring "large memory capacity and weak nonlinear transformation", e.g., $\log \nu \lesssim -1.0$, $\tau \gtrsim 4$, the linear ESN outperforms the nonlinear one. This observation corresponds to the first row in Table 1. For the task requiring "strong nonlinear transformation and less memory capacity", e.g., $\log \nu \gtrsim -0.5$, $\tau \lesssim 4$, the nonlinear ESN outperforms the linear one. This observation corresponds to the second row in Table 1. While these results are obtained by employing a particular functional form $f(x) = \sin x$ for the task, we confirmed that qualitatively the same results are obtained by employing other function forms $f(x) = \tan x$ and $x(1 - x^2)$. In this sense, the above direct comparison clearly shows the memory-nonlinearity trade-off, which is consistent with previous studies (Dambre et al. 2012; Verstraeten et al. 2010).

What is the mechanism behind the trade-off? In Table 1, the property in the right column is trivial by the definition; however, the property in the left column is nontrivial. Thus, the goal is to uncover the dynamical mechanism behind the degradation of the memory by the nonlinear dynamics of the reservoir. A possible mechanism illustrating the phenomenon has been proposed based on the variational Eq. (5) (Inubushi and Yoshimura 2017).

4 Dynamical Structure Suitable for Reservoir Computing

The simplest method to overcome Memory-Nonlinearity Trade-off would be to use a dynamical system consisting of both linear dynamics and nonlinear dynamics as a reservoir (Inubushi and Yoshimura 2017). Here, we introduce the reservoir where linear dynamics and nonlinear dynamics coexist; hereinafter, we refer to this type of reservoir as *mixture reservoir*. Some numerical results are shown to indicate that the mixture structure is suitable for RC.

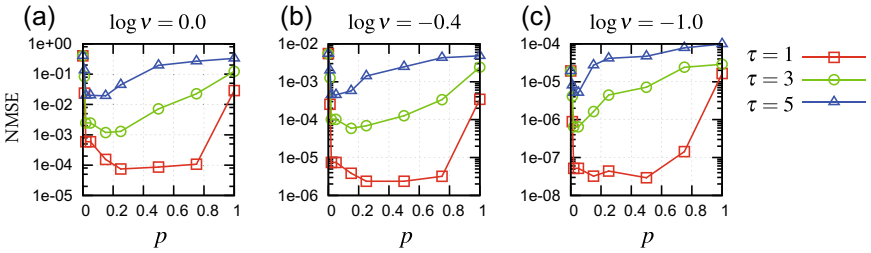

Fig. 4 Performance of the mixture ESN for the function approximation with various task parameters (Inubushi and Yoshimura 2017). The horizontal axis is the mixture rate p, and the vertical axis is the approximation error (NMSE). The task parameters are **a** $\log \nu = 0.0$, **b** $\log \nu = -0.4$, and **c** $\log \nu = -1.0$. The red squares, green circles, and blue triangles correspond to the task parameters $\tau = 1, 3, 5$, respectively. Figure reproduced and modified with permission from Springer Nature

Let us consider the ESN model consisting of N nodes again as an example of the mixture reservoir. In order to overcome the trade-off, it is expected to be effective to use an ESN consisting of "linear node", having the linear activation function $\phi[u] = u$, and "nonlinear node", having the nonlinear activation function $\phi[u] = \tanh u$. More precisely, we connect the linear nodes and the nonlinear nodes with random weights J_{ij}, v_i. As the conventional ESN model, the only readout weight $\{w_i\}$ is adjusted by the training data. We refer to this type of ESN as *mixture* ESN. The number of the nonlinear nodes is denoted by N_{NL}, and we introduce a mixture rate by $p = N_{NL}/N$ which plays a key role in the following discussion.

To study the performance of the mixture ESN, we employ the function approximation task $y(k) = \sin(\pi \nu s(k - \tau))$ as before. Figure 4 shows the numerical results of the function approximation by the mixture ESN with the total number of nodes $N = 100$. The horizontal axis is the mixture rate p, and the vertical axis is the approximation error (NMSE). The mixture ESN at $p = 0$ reduces to the linear ESN, and that at $p = 1$ reduces to the nonlinear, i.e., conventional ESN. The task parameters are (a) $\log \nu = 0.0$, (b) $\log \nu = -0.4$, and (c) $\log \nu = -1.0$. The red squares, green circles, and blue triangles correspond to the task parameters $\tau = 1, 3, 5$, respectively.

In all of the cases shown in Fig. 4, the mixture ESNs outperform the linear ESN ($p = 0$) and the nonlinear ESN ($p = 1$). In particular, for the task with small τ, the approximation error by the mixture ESN is surprisingly reduced in the vicinity of $p = 1$, compared with that by the nonlinear ESN. In other words, **replacing a small number of the nonlinear nodes with linear nodes in the conventional ESN drastically improves the performance**. These drastic improvements in performance are observed in the vicinity of $p = 0$ as well.

In order to study this remarkable improvement of the performance by introducing the mixture structure in more detail, we show, in the right panel of Fig. 5, the performance of a conventional ESN ($p = 1$) with *larger network sizes*. The task is the function approximation task again with $\nu = 1$ and $\tau = 1, 3, 5$. As a reference for the same task, Fig. 4a is shown in the left panel of Fig. 5. The conventional ESN shows better performance by increasing the network size (Rodan and Tino 2011)

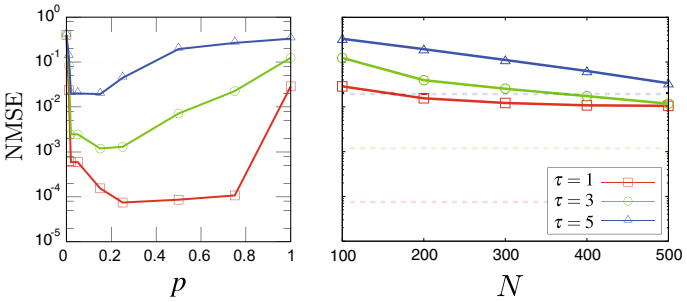

Fig. 5 The performance improvements by introducing the mixture structure versus those by increasing the network size. The left panel is the same as Fig. 4a for the reference. The right panel represents the performance improvements by increasing the network size from $N = 100$ to $N = 500$, where the vertical axis is NMSE. The dotted lines are the values of NMSE by the mixture ESNs at the optimal mixture rates for each task (see the left panel)

in general. Indeed, for each task ($\tau = 1, 3, 5$), the approximation errors are reduced monotonically by increasing the number of nodes ($N = 100, 200, 300, 400, 500$). However, the improvement of the performance by introducing the mixture ESN with fixed network size ($N = 100$) is considerably more effective than that by increasing the network sizes with a fixed mixture rate ($p = 1$). In the right panel of Fig. 5, the error values at the optimal mixture rates are plotted by dotted lines for each task. For instance, regarding the task with $\tau = 5$, while the conventional ESN reduces the error by one-tenth with $N = 500$ nodes, the mixture ESN at the optimal mixture rate can reduce the error at the same level (the blue dotted line) with *only* $N = 100$ nodes. For the tasks with $\tau = 1, 3$, the mixture ESNs at the optimal mixture rate clearly outperform the conventional ESN even with $N = 500$.

It is interesting that the optimal mixture rate $p_{\text{opt.}}$, where the mixture ESN shows the best performance, changes depending on the tasks. In fact, as shown in Fig. 4a, for the tasks with $\tau = 1$, $\tau = 3$, and $\tau = 5$, the optimal mixture rate is $p_{\text{opt.}} \simeq 0.25$, $p_{\text{opt.}} \simeq 0.15$, and $p_{\text{opt.}} \simeq 0.1$, respectively. Note that the larger the parameter τ is, the smaller the optimal mixture rate $p_{\text{opt.}}$ is. This observation is consistent with Memory-Nonlinearity Trade-off. Since the linear nodes play a role to increase the memory capacity, a mixture ESN with a larger number of the linear nodes is effective for a task requiring a larger memory capacity.

To study this dependency, we show a performance comparison diagram in Fig. 3b (Inubushi and Yoshimura 2017). As shown in Fig. 3a, for a set of given task parameters (ν, τ), the optimal mixture rate $p_{\text{opt.}}(\nu, \tau)$ is depicted with different symbols, where the minimal value is numerically found in the set $p \in \{0.00, 0.05, 0.15, 0.25, 0.50, 0.75, 1.00\}$. See Inubushi and Yoshimura (2017) for the details. From this diagram, it is clarified that the optimal mixture rate depends on the task gradually, and, significantly, the mixture ESN ($0 < p < 1$) outperforms the linear and nonlinear reservoir ($p = 0, 1$) over a broad region in the task parameter space.

The above numerical results lead to a conjecture that *the mixture reservoir is one of the dynamical structures suitable for RC in general*. Does the mixture reservoir work well, being less dependent on the tasks or details of the reservoir? For studying this question, we change the topology of the mixture ESN as a detail of the reservoir. We conducted numerical experiments for two other types of the mixture ESNs with the random sparse coupling and the ring coupling. In both cases, nonzero elements of coupling weights J_{ij} ($\neq 0$) are determined by random numbers (Yoshimura et al. 2018). The performance of the mixture ESNs with the random sparse coupling and the ring coupling are shown in Fig. 6 (a) and (b), respectively. The same function approximation tasks are employed as in Fig. 4. Illustrations are shown in the right bottom of each graph in Fig. 6, where the nodes colored with yellow represent the linear nodes. The qualitative features found in the results of the random sparse coupling case and the ring coupling case (Fig. 6a, b) are almost the same as those of the full coupling case (Fig. 4).

Moreover, to confirm the independence of tasks, it has been reported that some standard tasks, the time series prediction for Santa Fe Laser data set and the so-called NARMA task, can be solved by the mixture ESN effectively (Inubushi and Yoshimura 2017). The mixture ESN is effective also for the time series prediction task for the Hénon map as shown in Fig. 7 (Yoshimura et al. 2018). In Fig. 7, the vertical axis shows NMSE for m-step ahead prediction of chaotic signal generated from Hénon map $z(k+1) = 1 - 1.4z(k)^2 + 0.3z(k-1)$, i.e., the input is $z(k)$ and the desired output is $z(k+m)$. The prediction errors with the mixture ESN for the cases of the full coupling, the random sparse coupling, and the ring coupling are shown in Fig. 7 (a), (b), and (c), respectively. In all the cases, the mixture ESNs are effective independently of the network topology. These numerical results so far support the above conjecture.

Introducing the mixture reservoir significantly improves the information processing performance. This improvement occurs independently of tasks and details of the reservoir. This suggests that the mixture rate p is an effective hyperparameter for

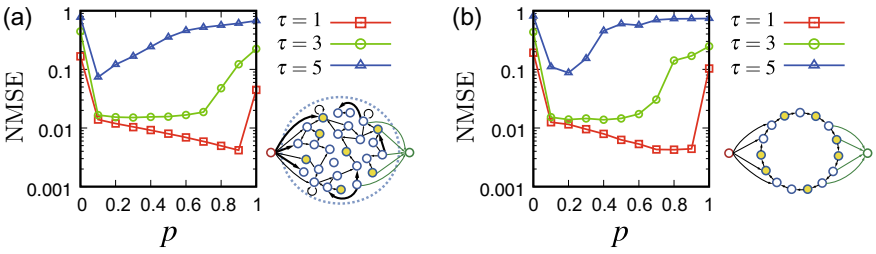

Fig. 6 Performance of the mixture ESN for the function approximation task ($\nu = 1.5$). The horizontal axis is the mixture rate p, and the vertical axis is the approximation error (NMSE). As a topology of the network, two types of the network are studied: **a** the network with the sparse random coupling, **b** the network with the ring coupling. In both cases, nonzero elements of coupling weights J_{ij} ($\neq 0$) are determined by random numbers (Yoshimura et al. 2018). Illustrations are shown in the right bottom of each graph, where the nodes colored with yellow represent the linear nodes

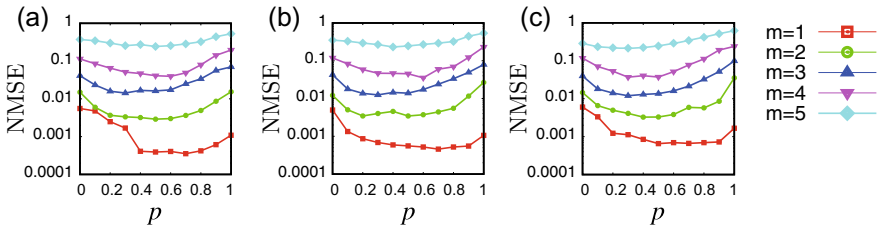

Fig. 7 Performance of the mixture ESN for m-step ahead prediction of the chaotic signal generated from the Hénon map. The horizontal axis is the mixture rate p, and the vertical axis is the prediction error (NMSE). As a topology of the network, three types of the network are studied: the network with **a** the full coupling, **b** the sparse random coupling, and **c** the ring coupling

the optimization of the reservoir. Considering ESN as an example, it is in general difficult to predict the response of the reservoir to changes in the parameters such as g. However, one can expect that decreasing the mixture rate p leads to an increase in the memory capacity of the reservoir. In this sense, the reservoir response to a change in the mixture rate is much simpler than that in the other parameters. This simpleness may make easy the optimization of p.

5 Conclusions and Future Works

In this article, we have given an overview of the mathematical aspects of RC, focusing on the characteristics and structures of the reservoir suitable for information processing. From this viewpoint, we here discuss two open problems of theoretical importance.

(i) Characterization of *good* reservoir:
The reservoirs (physical systems) have many parameters in general, and thus, some optimization over the parameters is required. Although Edge of Chaos is one of the crucial laws useful for optimization, the performance of RC cannot be determined solely by the conditional Lyapunov exponent, which just reduces the dimension of the parameter space by one. Is there a universal quantity, i.e., weakly dependent on tasks, that characterizes a good reservoir completely? In other words, does there exist some general property of dynamical systems which ensures the reservoir computer with high processing performance?

(ii) Dynamical structure suitable for RC:
While the mixture reservoir would be one of the candidates of the suitable structures for RC, the quest for more efficient structures is important. For instance, the configuration of the linear nodes and the nonlinear nodes in the mixture ESN of our study is determined randomly. It is expected that optimizing the configuration improves the performance further. According to the results from on-going numerical experiments, better performance is obtained by a mixture ESN with one-way coupling

from the nonlinear nodes to the linear nodes. The application of the theoretical findings to physical implementation is also important. For instance, adding some linear dynamics to the conventional physical reservoir, "physical mixture reservoir" could outperform the conventional one significantly. Apart from the mixture reservoir, for instance, Deep ESN has been studied theoretically (see Chapter *Deep Reservoir Computing*), and recently the first physical implementation of a deep reservoir, "photonic deep RC", has been reported (Nakajima et al. 2018). It is an interesting future work to find the suitable structures of dynamical systems for RC combining theoretical approaches with experimental approaches.

While we have described mainly the mathematical understandings of RC in this article, finally here we discuss the future development of the whole research field of RC. First, more and more of the various physical systems will be utilized for RC. As illustrated in Sect. 2, if the number of nodes (degree of freedom) of the reservoir is the same as that of the conventional RNN, RC cannot outperform the conventional RNN in principle. Thus, in our opinion, the RC method shows its true ability when one implements it by harnessing physical systems with a huge degree of freedom. Hence, the physical systems with a huge degree of freedom such as nonlinear spatiotemporal systems having high scalability would be promising for the future reservoir. Needless to say, it is expected that the future physical implementation, e.g., using optical systems, realizes the ultrafast information processing with the low energy consumption, which is one of the great advantages of RC.

Recently, many researchers in the field other than information science have studied RC actively as well. Nonlinear physicists in Maryland have proposed the inference method of dynamical variable (Lu et al. 2017), the prediction method of spatiotemporal chaos (Pathak et al. 2018), and the calculation method of Lyapunov exponent from time series (Pathak et al. 2017). Moreover, these methods have been applied to study turbulence physics, where a combination of *transfer learning* approach and RC is proposed for efficient inference of physical quantities (Inubushi and Goto 2019, 2020a, b). It is expected that these findings will be "re-imported" to information science in the future.

While there still remain mysterious points in the RC method, it has great advantages of the suitability for physical implementation and simplicity of the training method. If the theoretical open problems mentioned above are solved, then we can obtain a solid foundation for design principles for the physical reservoir, which may lead to the wide practical use of RC in a future society.

References

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

N. Bertschinger, T. Natschläger, Real-time computation at the edge of chaos in recurrent neural networks. Neural Comput. **16**, 1413–1436 (2004)

J. Boedecker, O. Obst, J. Lizier, N. Mayer, M. Asada, Information processing in echo state networks at the edge of chaos. Theory Biosci. **131**, 205–213 (2012)
J. Dambre, D. Verstraeten, B. Schrauwen, S. Massar, Information processing capacity of dynamical systems. Sci. Rep. **2** (2012)
F. Flandoli, B. Gess, M. Scheutzow, Synchronization by noise. Probab. Theory Relat. Fields **168**(3), 511–556 (2017)
K. Fujii, K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning. Phys. Rev. Appl. **8**, 024030 (2017)
S. Ganguli, D. Huh, H. Sompolinsky, Memory traces in dynamical systems. Proc. Natl. Acad. Sci. **105**(48), 18970–18975 (2008)
M.W. Hirsch, S. Smale, R.L. Devaney, *Differential Equations, Dynamical Systems, and An Introduction to Chaos* (Academic, 2012)
M. Inubushi, K. Yoshimura, Reservoir computing beyond memory-nonlinearity trade-off. Sci. Rep. **7**(10199), 1–10 (2017)
M. Inubushi, S. Goto, Transferring reservoir computing: Formulation and application to fluid physics, in *Lecture Notes in Computer Science*, vol. 11731 (Springer, 2019)
M. Inubushi, S. Goto, Transfer learning for nonlinear dynamics and its application to fluid turbulence. Phys. Rev. E **102**, 043301 (2020)
M. Inubushi, S. Goto, Inference of the energy dissipation rate of turbulence by machine learning (2020b). In preparation
H. Jaeger, Short term memory in echo state networks, vol. 152. GMD-Report (2002)
H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
L. Larger, A. Baylón-Fuentes, R. Martinenghi, V.S. Udaltsov, Y.K. Chembo, M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification. Phys. Rev. X **7**, 011015 (2017)
Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, E. Ott, Reservoir observers: model-free inference of unmeasured variables in chaotic systems. Chaos **27**(4) (2017)
W. Maass, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Theor. Comput. Sci. **2560**, 2531–2560 (2002)
K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Information processing via physical soft body. Sci. Rep. **5**, 10487 (2015)
M. Nakajima, M. Inubushi, T. Goh, T. Hashimoto, Coherently driven ultrafast complex-valued photonic reservoir computing, in *Conference on Lasers and Electro-Optics*. OSA Technical Digest (online) (Optical Society of America, 2018), p. SM1C.4
M. Nakajima, S. Konisho, K. Tanaka, T. Hashimoto, Deep reservoir computing using delay-based optical nonlinear oscillator, in *Conference in Cognitive Computing 2018* (2018)
R. Nakane, G. Tanaka, A. Hirose, Reservoir computing with spin waves excited in a garnet film. IEEE Access **6**, 4462–4469 (2018)
J. Pathak, Z. Lu, B.R. Hunt, M. Girvan, E. Ott, Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. Chaos **27**(12) (2017)
J. Pathak, B. Hunt, M. Girvan, L. Zhixin, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. Phys. Rev. Lett. **120**(2), 24102 (2018)
L.M. Pecora, T.L. Carroll, Driving systems with chaotic signals. Phys. Rev. A **44**(4), 2374–2383 (1991)
A. Rodan, P. Tino, Minimum complexity echo state network. Trans. Neural Netw. **22**(1), 131–144 (2011)
K. Takano, C. Sugano, M. Inubushi, K. Yoshimura, S. Sunada, K. Kanno, A. Uchida, Compact reservoir computing with a photonic integrated circuit. Opt. Express **26**(22), 29424–29439 (2018)
J.N. Teramae, D. Tanaka, Robustness of the noise-induced phase synchronization in a general class of limit cycle oscillators. Phys. Rev. Lett. **93**(20), 1–4 (2004)
T. Toyoizumi, Nearly extensive sequential memory lifetime achieved by coupled nonlinear neurons. Neural Comput. **2699**, 2678–2699 (2012)

T. Toyoizumi, L.F. Abbott, Beyond the edge of chaos: amplification and temporal integration by recurrent networks in the chaotic regime. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **84**(5), 1–8 (2011)

A. Uchida, R. McAllister, R. Roy, Consistency of nonlinear system response to complex drive signals. Phys. Rev. Lett. **93**, 244102 (2004)

D. Verstraeten, J. Dambre, X. Dutoit, B. Schrauwen, Memory versus non-linearity in reservoirs, in *Proceedings of the International Joint Conference on Neural Networks* (2010)

K. Yoshimura, M. Inubushi, Y. Ikeda, Y. Nagasawa, T. Kogahara, Computation performance of mixture-unit echo state networks, in *Proceedings of 2018 International Symposium on Nonlinear Theory and Its Applications* (2018), pp. 404–407

K. Yoshimura, I. Valiusaityte, P. Davis, Synchronization induced by common colored noise in limit cycle and chaotic systems. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **75**, 026208 (2007)

K. Yoshimura, J. Muramatsu, P. Davis, Conditions for common-noise-induced synchronization in time-delay systems. Phys. D **237**, 3146–3152 (2008a)

K. Yoshimura, P. Davis, A. Uchida, Invariance of frequency difference in nonresonant entrainment of detuned. Prog. Theor. Phys. **120**(4), 1–13 (2008b)