STAP REVIEW

Physical reservoir computing—an introductory perspective

Japanese Journal of Applied Physics **59**, 060501 (2020)
https://doi.org/**10.35848/1347-4065/ab8d4f**

STAP REVIEW
Physics and devices toward artificial intelligence

Physical reservoir computing—an introductory perspective

Kohei Nakajima*

Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, 113-8656 Tokyo, Japan

*E-mail: k_nakajima@mech.t.u-tokyo.ac.jp

Received February 17, 2020; revised April 20, 2020; accepted April 27, 2020; published online May 14, 2020

Understanding the fundamental relationships between physics and its information-processing capability has been an active research topic for many years. Physical reservoir computing is a recently introduced framework that allows one to exploit the complex dynamics of physical systems as information-processing devices. This framework is particularly suited for edge computing devices, in which information processing is incorporated at the edge (e.g. into sensors) in a decentralized manner to reduce the adaptation delay caused by data transmission overhead. This paper aims to illustrate the potentials of the framework using examples from soft robotics and to provide a concise overview focusing on the basic motivations for introducing it, which stem from a number of fields, including machine learning, nonlinear dynamical systems, biological science, materials science, and physics. © 2020 The Japan Society of Applied Physics

1. Introduction

Recently, a novel information-processing scheme that exploits physical dynamics as a computational resource has been proposed. This scheme is called *physical reservoir computing* (PRC). The current paper aims to introduce this framework concisely, focusing on its motivation and potential by using a number of examples. Understanding the original concept of reservoir computing (RC) is important to comprehend the concept of PRC. RC is a framework for recurrent neural network (RNN) training and was proposed in the early 2000s as a broad concept that allows one to deal with a number of different models of RNN, including the echo-state network (ESN)[1–3] and the liquid state machine (LSM),[4] under the same umbrella.[5–8]

Conventionally, to train an RNN, a backpropagation-through-time (BPTT) method[9] is frequently used. In this method, all the weights of the network are basically tuned toward the target function. In the RC framework, by preparing an RNN equipped with a massive amount of nonlinear elements coupled with one another, called a *reservoir*, only the readout part is usually trained toward the target function. In the simplest case, this readout part consists of linear and static weights that directly connect the reservoir nodes and output node [Fig. 1(a)]. Because of this unique system construction, RC has many advantages. Some typical examples are given below.

The first advantage comes from the ease in the training procedure, which makes the learning quick and stable. As noted above, in the conventional BPTT approach, all the weights in the network are tuned, which takes a significant amount of time in obtaining the optimal parameter set according to the type of the given target function. Furthermore, it is known to be unstable, in general, suggesting that it cannot always obtain the optimal set of weights after learning.[10] In the RC framework, the weights in the network are not always targeted for training. Instead, the training is mainly for the readout part, so the number of parameters that need to be tuned is generally small, making the training significantly faster [Fig. 1(a)]. In particular, if the readout part is set as linear and static weights, the training can be executed with a simple linear regression or ridge regression, and the optimal set of weights can be induced at once through a batch learning procedure, making the entire learning process simple and stable. Accordingly, there are many real-world application scenarios proposed in the literature. Starting from conventional signal processing for robust communication against noise,[3] learning of the grammatical structure of natural language,[11] robust speech recognitions,[12] or handwritten digit recognitions,[13] many attempts can be found for complex time series prediction tasks, including the time series of stock markets[14,15] or for the prediction of high-dimensional spatio-temporal dynamics found in nature,[16] including weather forecasting or the prediction of forest fire spreading. In robotics, for example, many cognitive tasks, which were previously difficult to implement because of the complicated procedure of RNN training, have been revived using RC for cognitive agents[17] and behavioral generations of robots, such as the emulation of motor controller,[18–21] inverse kinematics,[22] timing control,[23] or implementation of central pattern generator,[24] are successfully performed. In addition, researchers are now interested in applying the RC framework to sensory devices, in which the raw data are collected, and for executing processing natively on the sensory devices in real time, which is called edge computing.[25] Fonollosa et al. applied an RC framework to chemical gas sensory system and showed that it is suitable for real-time and continuous monitoring applications and improves the time response of the chemical sensory system.[26] Recently, the emulation of the functionality of a sensory device in a soft robotic platform was proposed using ESN, where the laser displacement sensor is emulated in a significantly high accuracy.[27] This approach is expected to replace the functionality of rigid components, that is, sensory devices, freeing soft robotic platforms from mechanical constraints to maintain their softness and flexibility. We should note that although the learning procedure of RC is simple, this does not imply that RC is less powerful than conventional machine learning techniques.[28] For example, it has been shown that ESN, which is a representative model system of RC, has a universal approximation property, and many studies are now proving its expressive power in different settings.[29,30] This implies that it is largely up to the experimenters using the framework and how they will utilize it to induce its potential. In a machine learning context, many improvements have been proposed to overcome the instability of RNN learning based on BPTT algorithms, which can be represented in the model of long-short term memory,[31] gated recurrent unit,[32] or unitary RNNs.[33,34] Among these

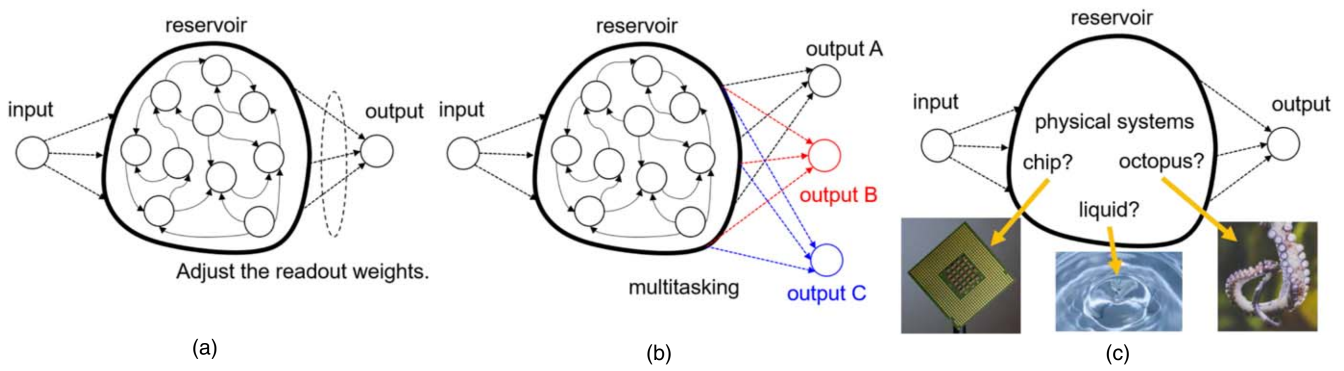

Fig. 1. (Color online) Typical settings and advantages in RC. (a). A typical ESN setting, a representative model in the RC framework. The reservoir is an RNN often equipped with a nonlinear activation function, such as $y = \tanh(x)$. Only the readout part is usually trained to the target function. (b) In the RC framework, multitasking can be safely implemented in principle, because no interference occurs among the tasks during the learning procedures. See the text for details. (c) Physical reservoir computing, which exploits the physical dynamics as a reservoir.

approaches, a recent systematic comparison analysis with RC has shown that each of these approaches has its merits and demerits (see Ref. 35 for more details), which suggests that the best approach depends on the experimental conditions and is largely up to what the experimenters wish to achieve.

The second advantage is its ease in multitasking or in sequential learning. Consider that the network is now implementing a task T_A to the output A according to the input u, which is expressed as $y_A = T_A(u)$. Now, we want to train the same network to additionally learn the task T_B to the output B according to the same input u, which is expressed as $y_B = T_B(u)$. In the conventional approach of backpropagation, the entire network is optimized for the task T_A first, and then the network is additionally trained for the task T_B using the backpropagation method, so these two tasks interfere during the update of weights within the same network. In this situation, there is danger that the network forgets the previously learned tasks. The extreme case for this phenomenon is called *catastrophic interference* or *catastrophic forgetting*,[36,37] and addressing this deficit remains a controversial topic for many researchers (see, e.g. Refs. 38–40). In the RC framework, because the training is basically limited at the readout part, no interference occurs among the tasks, so multitasking can safely be implemented in principle [Fig. 1(b)].

The third advantage is the arbitrariness and diversity in the choice of a reservoir. The basic concept of RC is exploiting the intrinsic dynamics of the reservoir by outsourcing learning, which requires some parameter tuning, to the readout part. According to this unique setting, reservoirs do not have to be an RNN anymore but can be any dynamical system. This idea naturally leads us to exploit the physical dynamics as a reservoir instead of using the simulated dynamics inside the PC [Fig. 1(c)]. This framework is called PRC and is a main theme of the current paper. This seemingly natural step makes the framework radically different from other machine learning methods. That is, PRC provides a novel insight not only into the machine learning community, but also into the dynamical systems field, physics, materials science, and biological science. This point will be elaborated on in detail later.

2. Prerequisite for a successful reservoir

As we verified in the previous section, there is a diversity in the choice of reservoir, and there is a freedom to use any kind of dynamical system if you wish. However, whether that reservoir works successfully is a different story. There exists a prerequisite to be used as a successful reservoir. The prerequisite is about the reproducibility of the input–output relation, which is an inevitable condition for any computational device. Namely, the reservoir should respond the same whenever the same input sequence is injected. Otherwise, every time you used it, the reservoir would respond differently, meaning that it would be operationally troublesome and unreliable. Considering that the reservoir is basically a dynamical system, this requirement is a somewhat severe condition because the behavior of dynamical systems is in general determined by the initial condition. If you can precisely select the initial condition of the reservoir and can control the timing to inject the input sequence into the system, then for the identical input sequence, you can always obtain the same response from the system. However, this constraint is powerful and restricts the usability of the computational device, and it is particularly annoying if you wish to exploit the natural and physical dynamics as a reservoir because it is generally difficult to infer or control the initial condition of the physical dynamics. It is preferable to guarantee the reproducibility of the response whenever you inject the same input sequence and, furthermore, to do so without controlling the initial condition of the reservoir. The property that realizes these conditions of the reservoir is called the *echo state property* (ESP).[1] Simply put, ESP requires the reservoir states to be expressed as a function of the previous input sequence only. A similar concept has been studied in the nonlinear dynamical systems field from a different angle as a synchronization phenomenon between two identical systems induced by a common signal (or noise) or a generalized synchronization between an input sequence and the corresponding response of the system (see, e.g. Ref. 41). This property suggests that even if the system is driven by a different initial condition, by injecting an input sequence, the corresponding response of the system becomes the same. Mathematical investigations of the concept of ESP (e.g. Refs. 42–44) and understanding its relation to the nonlinear dynamical systems field are still ongoing research topics (e.g. Ref. 45).

Here, we would like to summarize the situation briefly [Fig. 2]. Consider that we have input $u(t)$ and the reservoir state $\boldsymbol{x}(t)$ at timestep t, and the reservoir dynamics is

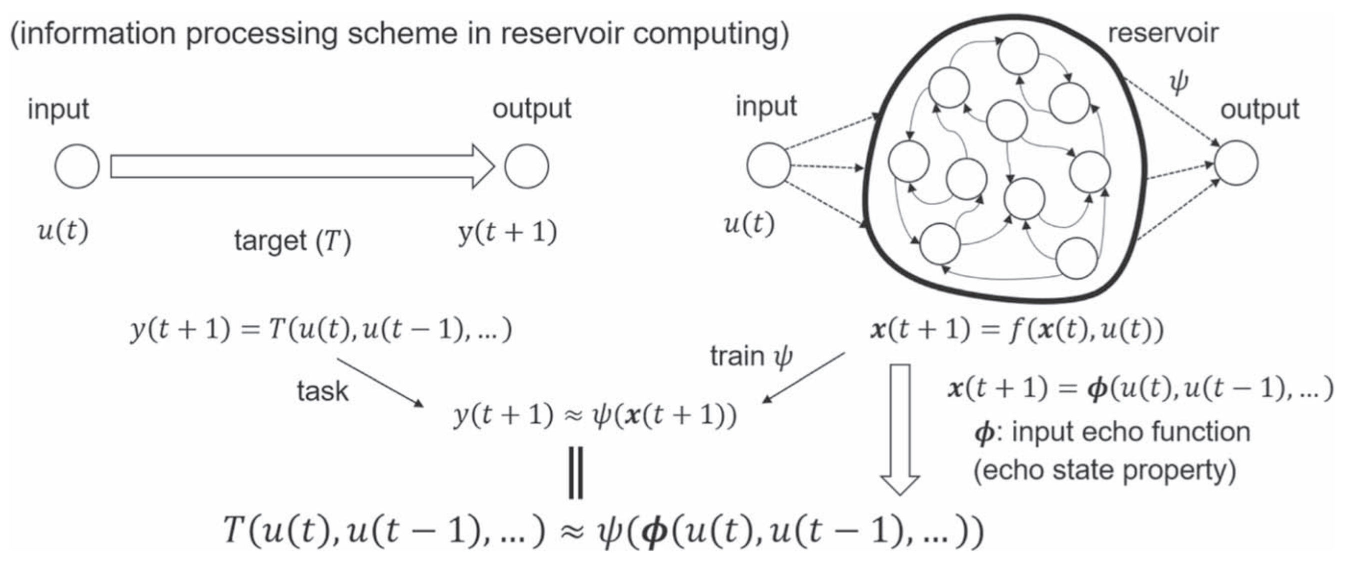
(information processing scheme in reservoir computing)
input
output
$u(t)$
target (T)
$y(t+1)$
input
reservoir
ψ
output
$u(t)$
$y(t+1) = T(u(t), u(t-1), \ldots)$
task
train ψ
$\boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), u(t))$
$y(t+1) \approx \psi(\boldsymbol{x}(t+1))$
$\boldsymbol{x}(t+1) = \boldsymbol{\phi}(u(t), u(t-1), \ldots)$
$\boldsymbol{\phi}$: input echo function
(echo state property)
$T(u(t), u(t-1), \ldots) \approx \psi(\boldsymbol{\phi}(u(t), u(t-1), \ldots))$

Fig. 2. Schematics showing how the echo state property works in RC. As can be seen in the diagram, input echo function ϕ is a part intrinsic to the reservoir, and the experimenters can adjust the output using readout function ψ. See the text for details.

expressed as $\boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), u(t))$. In general, a task T targeted by RNN is a function of the previous input sequence, which is sometimes called a temporal machine learning task; then, it is expressed as $y(t+1) = T(u(t), u(t-1), \ldots)$. In the RC scheme, by tuning the readout ψ (note that this readout function does not have to be linear in general), we aim to approximate the target $y(t)$, which is expressed as $y(t) \approx \psi(\boldsymbol{x}(t))$. Now, if the reservoir fulfills the ESP, then $\boldsymbol{x}(t) = \boldsymbol{\phi}(u(t-1), u(t-2), ..)$, where ϕ is called the *input echo function* in Ref. 1 and where it is a function intrinsic to the reservoir. This implies that the internal state of the reservoir is completely described by the driven input sequence and is related to the filter concept, which is discussed in more detail later. Note that we can easily understand that when the ESP holds, then the reservoir states from different initial conditions, which are expressed as $\boldsymbol{x}'(t)$ and $\boldsymbol{x}(t)$ and driven by identical input sequence, will respond the same or become synchronized, such as $|f(\boldsymbol{x}'(t), u(t)) - f(\boldsymbol{x}(t), u(t))| \approx 0$ for a sufficiently large t. In summary, the RC scheme can be expressed as exploiting the function intrinsic to the reservoir ϕ and adjusting the readout function ψ to approximate the target function T, which is expressed as $T(u(t), u(t-1), ...) \approx \psi(\boldsymbol{\phi}(u(t), u(t-1), ...))$.

From this viewpoint, evaluating the information processing capability or expressive power of a given reservoir is nothing but evaluating the property of the function ϕ. Currently, several approaches exist. The typical case is evaluating how well the given reservoir can output the previous input sequence, and this measure is called memory capacity.[46] Focusing on ESN, the behaviors of memory capacity and their related measures are studied in detail with a linear activation function[46–51] and, recently, with a nonlinear activation function.[52–54] This measure is further generalized and extended to be able to evaluate the nonlinear memory capacities by decomposing the function ϕ into the combinations of multiple orthogonal polynomials,[55] and the trade-off between the expressiveness of ϕ for linear and nonlinear functions is investigated.[55,56] Investigations of the relationships between the dynamical property of the reservoir and its information processing capability are now ongoing hot topics in the field.[57] Discussions that include how the bifurcation structure or the order-chaos transition (the critical point is often referred to as *edge of chaos*) affects the computational power of the reservoir are one such example.[58–61]

As we confirmed in this section, although, on the one hand, the learning procedure seems simple in RC, which is outsourced to the readout part, on the other hand, the reservoir part can be taken as a huge hyper parameter that is difficult to harness without knowledge of nonlinear dynamical systems.

3. Diverse variations of reservoir: toward exploiting physical dynamics

As we discussed in the previous sections, many types of reservoirs are now proposed. Among these, a framework that exploits the physical dynamics as the reservoir is called PRC. Because the natural physical dynamics is directly used as a computational resource, even if the same computation is implemented, according to the different physical property, there will be diverse application scenarios. Increasingly, many physical reservoirs have been reported worldwide [Fig. 3], such as the case using photonics,[62–72] spintronics,[73–84] quantum dynamics,[85,86–90] nanomaterials,[91–99] analog circuits and field programmable gate arrays,[100–107] mechanics,[108–123] fluids,[124–127] and biological materials.[128–130] Readers interested in which types of reservoirs are currently proposed can view the materials such as Refs. 131, 132.

Before going into PRC, which is the main theme of the current paper, we would like to overview the typical misapprehensions that we frequently face when it comes to the RC framework in this section. The first one is the belief that the weights of the reservoir should be set randomly. Of course, there exists a reservoir that implements a random weight matrix, such as ESN, but this is not an essential requisite. RC was originally inspired by the type of information processing that occurs in the brain, and the connections between neurons are usually not random but have specific structures. Accordingly, several reservoir settings implement brain-inspired connections[134] or simply implement the

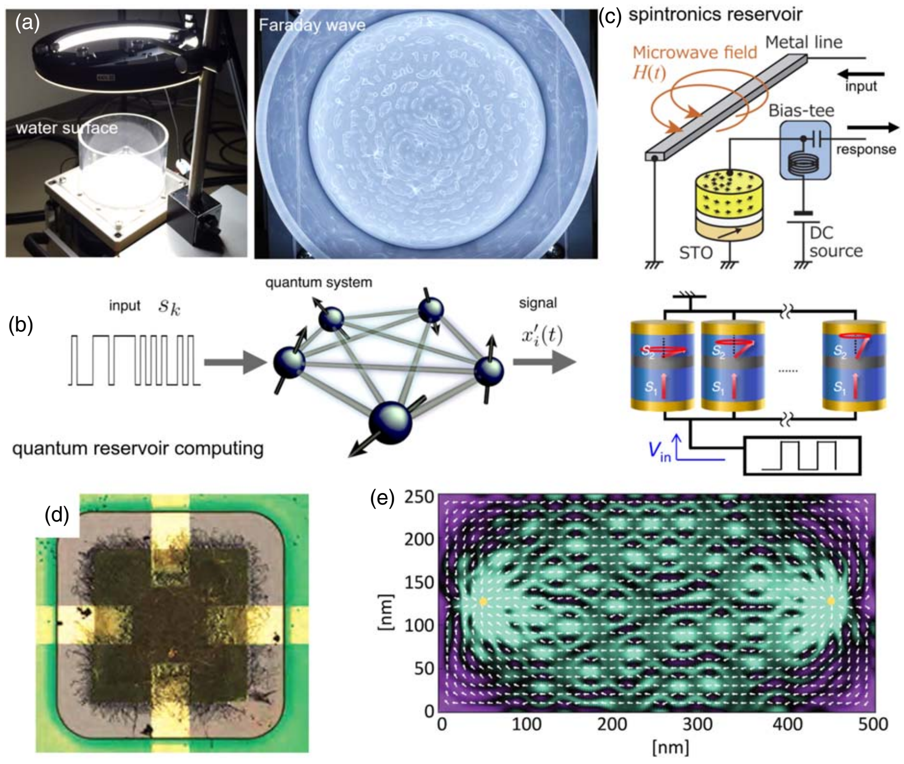

Fig. 3. (Color online) Variations of physical reservoirs. (a) The physical liquid state machine proposed in Ref. 124. It exploits the Faraday wave as a computational resource. (b) Quantum reservoir computing proposed in Ref. 85. It allows one to exploit disordered ensemble quantum dynamics as a computational resource. Figure reprinted with permission from Ref. 85, Copyright (2017) by the American Physical Society. (c) Variations of the spintronics reservoir. The upper and lower diagrams show reservoirs, which exploit vortex-type spintronics[73] and spatially multiplexed magnetic tunnel junctions,[74] respectively. The upper diagram is a figure reprinted with permission from Ref. 73 by the author. The lower diagram is a figure reprinted with permission from Ref. 74, Copyright (2018) by the American Physical Society. (d) Complex Turing B-type atomic switch networks proposed in Ref. 91. The complex nanowire network extends throughout the device and is probed via macroscopic electrodes. Figure reprinted with permission from Ref. 91, Copyright (2012) by John Wiley and Sons. (e) A skyrmion network embedded in frustrated magnetic films proposed in Ref. 133. The current path is visualized after the voltage is applied to the frustrated magnetic texture including Bloch skyrmions. Figure reprinted with permission from Ref. 133, Copyright (2018) by the American Physical Society.

neighboring connections,[4] introducing a spatial dimension that is not random at all. More coherent network structures, such as cyclic reservoirs, are also investigated.[50] One interesting aspect of RC is that it is capable of exploring the computational account of the structure of the reservoir, and as we will see later, this point is important for PRC.

The second misconception is that the reservoir weights should remain unchanged, and experimenters cannot tune them in any sense. This is untrue. This mistake is thought to be raised from the expression of the RC learning scheme that the training is performed in the readout part. This expression of the learning scheme is true, but this does not always mean the experimenter cannot tune the weights of the reservoir. An obvious counterexample is that when setting the ESN, it is common to tune the spectral radius of the reservoir weights.[1–3,5,135] This is nothing but the tuning, or preconditioning, of the internal weights before training the readout to some specific task. Another example can be found in cases that implement pretraining in the reservoir part before training the entire system for some specific target task. The use of recurrent infomax,[136] which maximizes the mutual information between the past and future within the internal dynamics, or the implementation of the plasticity rule, such as Hebbian learning[137] or spike-timing-dependent plasticity,[138,139] into the input-driven RNN have been reported in pretraining the reservoir. From this viewpoint, the recently introduced RNN called ALBERT[140] for language processing can be included as a pretrained reservoir whose internal networks are pretrained based on predicting the ordering of two consecutive segments of text in the language data set; here, the readout part is trained for specific language-processing tasks.

The third misunderstanding is that if the reservoir is exhibiting chaos, which is a frequently observed behavior of nonlinear dynamical systems, then this means it cannot be used successfully. Chaos can be characterized by sensitivity to initial conditions, where a slight initial difference in the state expands exponentially, and in this sense, the current state of the system is certainly affected by the initial condition. Accordingly, although the chaotic dynamics show a rich diversity of patterns for function emulation, it seems that chaos does not show ESP and is not suitable for RC. However, this is not the case. Even if the dynamical system exhibits chaos, when it is driven by the input sequence (or noise), chaos is sometimes suppressed, and generalized synchronization occurs between the input sequence and the response of the dynamics,[41] which is an outcome of ESP. In particular, chaos in a large ESN equipped with a sigmoidal function[141] can be suppressed with noise.[142] There exists a learning scheme that exploits this property of chaos suppression effectively, and it is found in the study of the first-order-reduced and controlled-error (FORCE) learning approach.[143] In the study of FORCE learning approach, it was found that a chaotic reservoir is capable of implementing coherent patterns by adjusting the readout weights with the output fed back to the reservoir, or interestingly, the learning performance was even better than a non-chaotic reservoir in this condition. Furthermore, because there is no fundamental difference between the output node fed back to the reservoir and the reservoir nodes interacting with each other, both through linear connection weights (although there is a slight difference concerning whether the output is injected into the nonlinear activation function before fed back to the reservoir), the FORCE learning scheme has been applied not only to the readout weights, but also to the internal weights of the reservoir.[143–145] Chaos is more apparently exploited in the learning scheme, which is called innate training.[146] Chaos has rich dynamics but does not guarantee reproducible input–output relations. Then, why not keep the richness of the dynamics and make it reproducible? In the innate training approach, preparing the chaotic reservoir at first and collecting its own chaotic dynamics as training data, the internal connection weights are trained using FORCE learning to output their own chaotic dynamics reproducibly. This approach can be also viewed as pre-training of the reservoir and has been applied for several machine learning and robot control tasks (see, e.g. Refs. 147–149). Recently, many neuromorphic devices have been shown to exhibit chaos (e.g. Refs. 150–152), and it is expected that these chaotic dynamics can be harnessed and exploited as a computational resource based on an RC framework.

Because the tuning of the internal weights were introduced in the above approaches, it may be helpful to clarify the difference between the conventional training scheme, such as BPTT, and the above introduced approaches. The main difference comes from the design of the cost function. In BPTT, there usually exists a global target function, and the gradient is obtained based on it; in addition, the error is backpropagated to each internal node to be used to update the concerning weights. In the above approaches, however, the internal weights are not usually tuned for the global target function but can be tuned for any global or local target function that the experimenter designs. In this sense, the above approaches contain more freedom in the setting of cost functions, or it may be more appropriate to say that these approaches even include the conventional setting of cost function. This RC property, which can be composed of multiple cost functions, is also an important aspect to be kept in mind when trying to step toward the PRC.

In this paper, we discuss what becomes interesting when we proceed from conventional RC driven inside a PC (this is also physical dynamics, though) to PRC that exploits physical dynamics as a reservoir. The story begins from the genesis of LSM, which is one of the original RC model systems.

4. Liquid state machine

4.1. "Wetware" and its implication

When Wolfgang Maass, Thomas Natschläger, and Henry Markram proposed the seminal model of the LSM, at around the same time, Wolfgang Maass presented some interesting insights in his paper entitled "Wetware" about the modality of information processing in the brain.[153] This paper starts as follows:

"If you pour water over your PC, the PC will stop working. This is because very late in the history of computing—which started about 500 million years ago—the PC and other devices for information processing were developed that require a dry environment. But these new devices, consisting of hardware and software, have a disadvantage: they do not work as well as the older and more common computational devices that are called nervous systems, or brains, and which consist of wetware. These superior computational devices were made to function in a somewhat salty aqueous solution, apparently because many of the first creatures with a nervous system were coming from the sea. We still carry an echo of this history of computing in our heads: the neurons in our brain are embedded into an artificial sea-environment, the salty aqueous extracellular fluid which surrounds the neurons in our brain. ..."

Maass's paper[153] subsequently discusses how to capture the information processing function of the human brain. The idea expressed in the above introductory paragraph already penetrates the fundamental aspect of PRC.

The important point that we should confirm here is that once computation, which is an abstract input–output operation in principle, was implemented in the real-world through a physical entity or substrate, then the physical property of the substrate and the influence of its execution environment came to affect the implemented computation and inevitably added a novel property/functionality to the system. The above example clearly suggests that even if the same computation is implemented, according to the choice of physics for the substrate (in the above case, the conventional PC and brain), the robustness against water is different.

The conventional PC consists of hardware and software; the hardware is the "physical" part of the PC, and the software is a set of commands used to run it. These two components function complementarily. That is, the hardware is specialized and designed to execute the command sent from the software. In contrast, the nervous system can function in a somewhat salty aqueous solution, but this physical condition is not fully designed for information processing. Rather, the nervous system exploits its given environmental constraints and physical conditions-which are shaped by its original context (i.e. many of the first creatures with a nervous system came from the sea)-to enable information processing.

When we look at the background of the seminal model of the LSM, it is evident that Wolfgang Maass and his colleagues were not adopting a conventional view of the brain as a network consisting of interacting elements (i.e. neurons) as many researchers do; instead, they characterized its behavior based on the surrounding liquid physical substrate. Furthermore, the idea is not merely a metaphor; the researchers even proposed a concrete sketch of their proposed model. This system is called the "liquid computer."[125]

4.2. Liquid computer and the liquid brain

Thomas Natschläger, Wolfgang Maass, and Henry Markram suggested that the brain is constantly exposed to a massive flow of sensory information, including both audio and visual inputs, and that it does not exist in the stable state frequently expressed as an attractor but rather in a transient state (except when it is in the "dead" state).[125] According to this view, they proposed a scheme to exploit the surface of a liquid (such as a cup of coffee) for computation.

We focus on a transformation over a time series. We consider the issue by mapping from input sequences $\boldsymbol{u}(\cdot)$, which are a function of time, to output sequences $\boldsymbol{v}(\cdot)$, which are also a function of time; this transformation is usually called a filter (or operator).

See Fig. 4(a). The schematics show the conceptual design of a "liquid computer."[125] To illustrate this concept, one could imagine a situation where he or she prepares a cup of coffee and perturbs the coffee surface by using a spoon or dropping a cube of sugar in, thereby injecting an "input." Consider that we have a video camera that can monitor the coffee's surface in real time and define this camera image at time t as the liquid state $\boldsymbol{x}(t)$ [Fig. 4(b)]. The liquid (in this case, coffee) transforms the input time series $\boldsymbol{u}(\cdot)$ into a liquid state $\boldsymbol{x}(t)$, expressed as $\boldsymbol{x}(t) = (L\boldsymbol{u})(t)$, where L is called a liquid filter. The image is sent to the PC, and by using the state of the surface, the PC processes the state and outputs the result. The interesting point of this system is that one can design various filters without using the memory storage inside the PC; in other words, this process can be carried out with the memory-less readout f, expressed as $\boldsymbol{v}(t) = f(\boldsymbol{x}(t))$.

Let us consider an example of information processing using this system. Assume that we want the system to output the number of cubes of sugar injected over the last two seconds. Because the readout part in the PC is memory-less, to perform this task, the current liquid state should be able to express the number of cubes of sugar dropped inside over the last two seconds in a distinguishable form. Let us call this ability to distinguish the previous input state as a difference in the current liquid state the "separation property" of the liquid. Then, to perform the task, it is necessary to map the separated states into the required output (e.g. the liquid states perturbed in the order of "*spoon* $\rightarrow$ *cube* $\rightarrow$ *cube*" and "*cube* $\rightarrow$ *spoon* $\rightarrow$ *cube*" should be mapped to output "2"). This property of the readout function is called the "approximation property." Interestingly, it has been shown that any time invariant filter with fading memory can be approximated in an arbitrary precision composing these two properties (with a filter bank containing point-wise separation property and readout function having universal approximation property) (see, Refs. 4, 154, 155 for detailed discussions). In the paper, Ref. 125, it is stated that the formalization of this liquid computer is an LSM and is proposed to understand the information processing of a neural circuit[125] [Fig. 4(c)].

As soon as the concept of the liquid computer and its formalization under an LSM were proposed, two computer scientists, Chrisantha Fernando and Sampsa Sojakka from the University of Sussex, integrated the idea into a physical system; they called this model the "liquid brain" [Fig. 5]. In their paper,[126] they described it as follows:

"... Here we have taken the metaphor seriously and demonstrated that real water can be used as an LSM for solving the XOR problem and ..."

Using this system, they showed that water in a bucket is capable of implementing an XOR task and a speech recognition task.[126] We can confirm that water in a bucket is not made or designed for computation but can be exploited for it. (Note that, recently in complex systems study, cognitive networks that lack stable connections and static elements are also called liquid brains.[156]) These networks include such as ant and termite colonies, immune systems, and slime moulds.)

5. Soft robotics

A computer is, in simple terms, a machine that is made to compute. Accordingly, the hardware structure of a computer is specialized to implement computation in general. Here, the concrete form of computation is determined beforehand in a top-down manner, and to realize it, the component arrangement is designed and decided in detail. On this point, the liquid computer and liquid brain are composed in somewhat opposite directions compared with the conventional computer. They both started from the physical property of liquid, and by considering how to exploit this property for computation, they came to invent a novel scheme to implement it, which is a bottom-up approach.

5.1. Embodiment and morphological computation

In robotics, a concept that accounts for these unexpected and intrinsic properties associated with the physical body when implementing computations, abstract operations, or behavior control has been around for a long time. This concept is called "embodiment."[157–159] For example, a seminal platform called a "passive dynamic walker" can walk naturally like a human without having an external controller.[160] Just

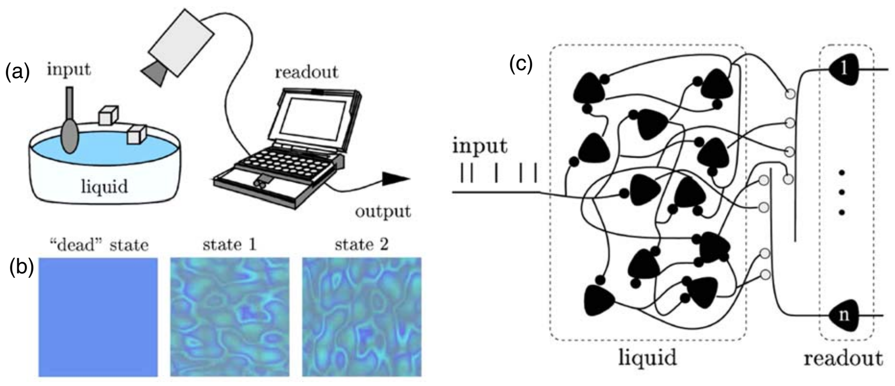

Fig. 4. (Color online) A Natschlager–Maass–Markram-type liquid computer and its analogy to neural information processing. (a) Schematics of a "Liquid computer." Figure reprinted with permission from Ref. 125 by the author. (c) The system takes a video image of a liquid surface as a state of the system. The liquid surface shows different spatiotemporal patterns according to how it is perturbed (e.g. manual perturbations using a spoon or dropping a cube of sugar, and their temporal orderings make the patterns of the liquid surface different, such as "state 1" and "state 2"). Figure reprinted with permission from Ref. 125 by the author. (c) Understanding neural circuits as an LSM. Figure reprinted with permission from Ref. 125 by the author.

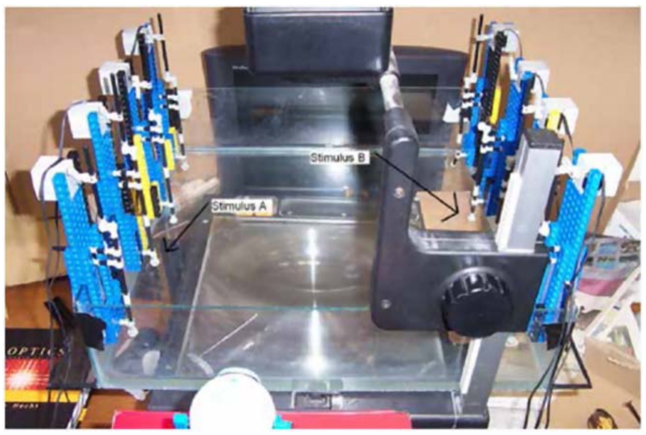

Fig. 5. (Color online) A Fernando–Sojakka-type liquid brain. Figure reprinted with permission from Ref. 126, Copyright (2003) by Springer Nature.

by using a well-designed body (a compass-like shape) and a well-designed environment (a slope), the natural walking behavior can be realized, where the behavior control is partially outsourced to the physical body. In bio-inspired robotics, this property of embodiment is studied in various platforms, including not only bipedal walkers, but also quadruped robots (e.g. Refs. 161–164). Similar properties can be found in animals. There is a famous experiment that used the dead body of fish (a trout, specifically) where the body was able to generate a vivid and natural swimming motion by exploiting the vortex in a water tank.[165] In this experiment, because the fish was dead, we can guarantee that the central nervous system of the fish was not functioning at all, so we can also confirm that the specific morphology and material property of the body and its interaction with the vortex in the surrounding water environment were capable of realizing the natural swimming motion of a fish.[165] In the field of self-assembling systems, there are many studies that investigate how the shape of each element induces or affects the global behavior of the system (e.g. Refs. 166–169). Here, the research field that aims at investigating and pursuing the nature of how the shape or morphology of the system affects the behavior of the entire system is called *morphological computation*.[170]

Are there any quantitative ways to characterize the intrinsic information processing capability of the physical body? Helmut Hauser et al. tried to propose a framework to theoretically investigate the morphological computation of compliant bodies.[108] In their study, they considered a mass-damper system, which is often used to model the body of robots, and explained that by using a linear mass-damper system, it is possible to compose a filter bank, which we discussed earlier [Fig. 6(a)]. This implies that if you design the readout function nicely, it is possible to approximate time-invariant filters with a fading memory property using a linear mass-damper system, which is consistent with the arguments corresponding to the LSM model. Furthermore, the authors numerically demonstrated that by using a complex nonlinear mass-damper system, even the nonlinearity required in the readout function can be outsourced to the mass-damper system, and the system would be capable of emulating nonlinear filters with fading memory only by composing the linear readouts. That is, this approach suggests that the physical body of robots can be, in some conditions, used to emulate nonlinear filters with a fading memory, which implies that the physical body can be used as a successful reservoir. Subsequently, Helmut Hauser et al. have investigated the role of feedback on the mass-damper system implementing nonlinear limit cycles based on this framework.[109] Tensegrity structures serve as an appropriate testbed to implement this framework, where it enables the structures to embed closed-loop control and realize locomotion by exploiting its intrinsic body dynamics as a computational resource, here being a controller[110,115,119] [Fig. 6(b)]. (Note that, although we do not go into details in this paper, information theoretic approaches to characterize morphological computation have also been investigated.[171,172])

5.2. PRC using a soft robotic arm

Soft robotics is a recently developed field that actively investigates the account of soft and compliant bodies to functionality and behavioral control.[174–176] Compared with conventional rigid-bodied robots, soft robots introduce a number of novel challenges and viewpoints into the field regarding the material properties and deformable morphology of the body and the complexity and diversity of body dynamics. Soft robots hold many advantages, which are linked to the mechanical softness of the body.[174–177] For example, they are considered to be useful in the situation of human–robot interaction, rescue, or biomedical applications because they do not damage people in the same way that rigid robots do; in other words, they are generally considered a safer option. These robots, however, include challenges in terms of control.[177] Soft robots are often classified into the category of an underactuated system, where the number of the actuation points are less than the degrees of freedom. Furthermore, they usually generate diverse and complex body dynamics when actuated, which are high-dimensional, nonlinear, and contain short-term memory.[178–180] These properties make soft robots difficult to control using the conventional control scheme.

On the other hand, these seemingly undesirable properties of soft robot control can be viewed as a positive from PRC perspectives. That is, we can exploit the diverse, rich dynamics of a soft body as a computational resource-more specifically, as a reservoir [Figs. 6(c), 6(d), and 6(e)]. In previous studies, we have shown that a silicone-based soft robotic arm inspired by an octopus can be used as a successful reservoir by taking the actuation sequence as the input and sensory reading as the reservoir state; indeed, this method exhibits high information-processing capability in some conditions[114,116,121,122] [Fig. 6(d)]. Interestingly, octopus arms have characteristic muscle organizations termed muscular-hydrostats.[181] In these structures, the volume of the organ remains constant during their motion, enabling diverse and complex behaviors. We showed that using biologically plausible parameter settings, the dynamic model of the muscular-hydrostat system has the computational capacity to achieve a complex nonlinear computation.[111,112,117] Furthermore, by incorporating the feedback-loop from the output to the next input (i.e. the next actuation pattern), we have demonstrated that the robot's behavioral control for the next time step can be implemented by using its current state of its body as a computational resource, suggesting that the "controller" and "to be controlled" is the same in this scheme.[114] This concept has been

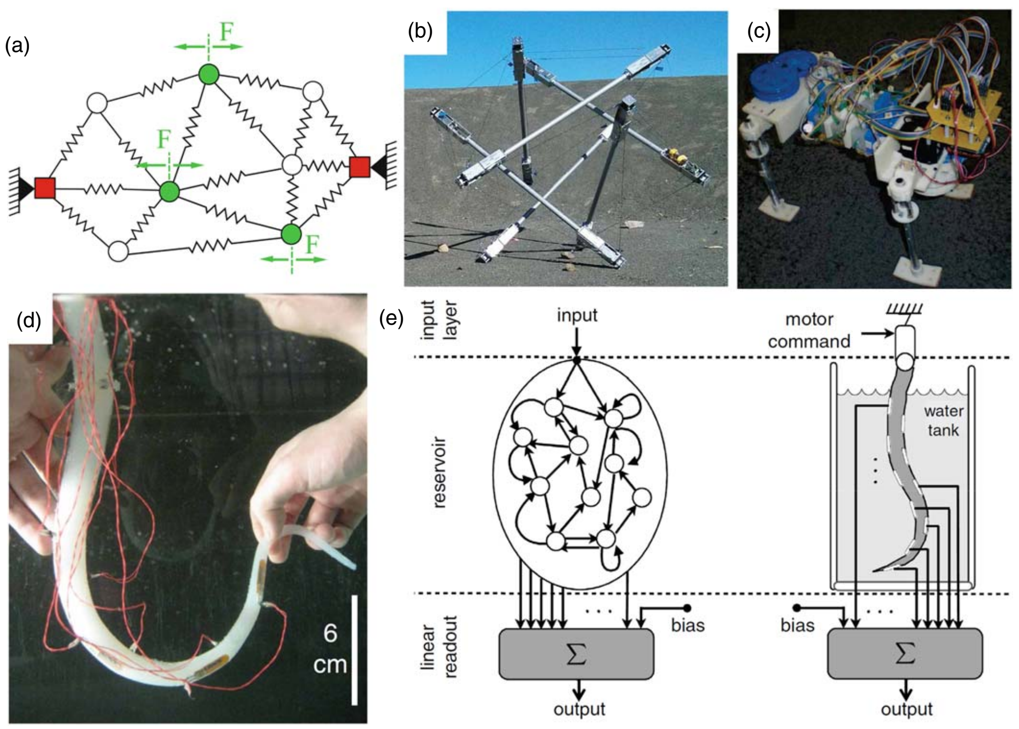

Fig. 6. (Color online) Physical reservoir computing using compliant and soft bodies. (a) A generic mass-spring network used as a reservoir in Ref. 108. Figure reprinted from Ref. 108 under the Creative Commons CC-BY-NC license. (b) A tensegrity robot called SUPERball proposed in Ref. 173. Figure reprinted with permission from Ref. 173, Copyright (2015) by IEEE. (c) A quadruped robot called Kitty proposed in Ref. 113, which exploits soft spine dynamics as a reservoir. Figure reprinted with permission from Ref. 113, Copyright (2013) by IEEE. (d) A picture of a physical soft robotic arm inspired by the octopus used in the experiment in Ref. 116. It is made of silicone and embeds ten bending sensors, monitoring the soft body dynamics every 0.03 [s]. Figure reprinted from Ref. 116 under the Creative Commons license. (e) Schematics explaining how to exploit the soft robotic arm as a reservoir. Figure reprinted from Ref. 116 under the Creative Commons license.

also applied to the study of a quadruped robot, where the robot exploits its spine dynamics as a physical reservoir to control its actuation patterns and locomotions[113] [Fig. 6(c)]. In short, the drawbacks of a soft robot control became assets for control from a PRC viewpoint.

6. Exploiting physical dynamics for computational purposes

We began by reviewing the concept of wetware by Wolfgang Maass, and from there, we illustrated the development of physical platforms, such as the liquid computer, liquid brain, mass-damper systems, and silicone-based soft robotic arms inspired by octopuses. In this section, we would like to review three significant phases that we can find in this evolution.

6.1. Phase 0: inferring the computational power of physical systems

PRC provides a method to exploit natural physical dynamics as a computational device. This implies that this method is also useful for investigating which physical systems are suitable to implement which types of computation and for analyzing the information-processing capability of the physical dynamics. In particular, if we use linear and static readouts to generate outputs for specific tasks requiring a certain amount of nonlinearity and memory, because we are not adding any nonlinear terms and memory externally, by evaluating the task performance, we can infer back which amount of nonlinearity and memory has been positively contributed or exploited from the physical reservoir to perform the task [Fig. 7(a)]. That is, in this way, if we use a previously introduced symbol, we can pursue the nature of the function ϕ in the physical systems. Systematic investigations are needed to reveal the response characteristics of the physical system against the type, the intensity, and the timescale of the input, and these properties are intrinsic to each physical system. Accordingly, we can expect the diversity of the type of information processing according to the type of physics, where each physical system has a preference in terms of the type of function that it can express.

In neuroscience, there are several studies that have inferred the computational capability of the neural circuits[128] or the cultured neural systems.[129,130] Obviously, their motivation is not to make a high-performance computer but rather to reveal the functional characteristics of the natural systems from information-processing perspectives. This approach can also be applied to infer the functionality of the body of living systems quantitatively. As we discussed in the concept of embodiment, a functionality that is thought to be handled by the brain is often partially outsourced to the physical body. Unlike the randomly coupled ESN, the biological body has a specific structure or morphology that is intrinsic to respective living organisms. This specific morphology is evolved through the respective ecological niche of living things, which is a driving force of the diversity of morphology. It is expected that the PRC framework has the potential to reveal the property of the body's morphology from information-processing perspectives. (Related to this issue, there exists a research project that aims to characterize RC from evolutionary perspectives.[182]) The above directions of research can be summarized and stated as the study of ϕ within

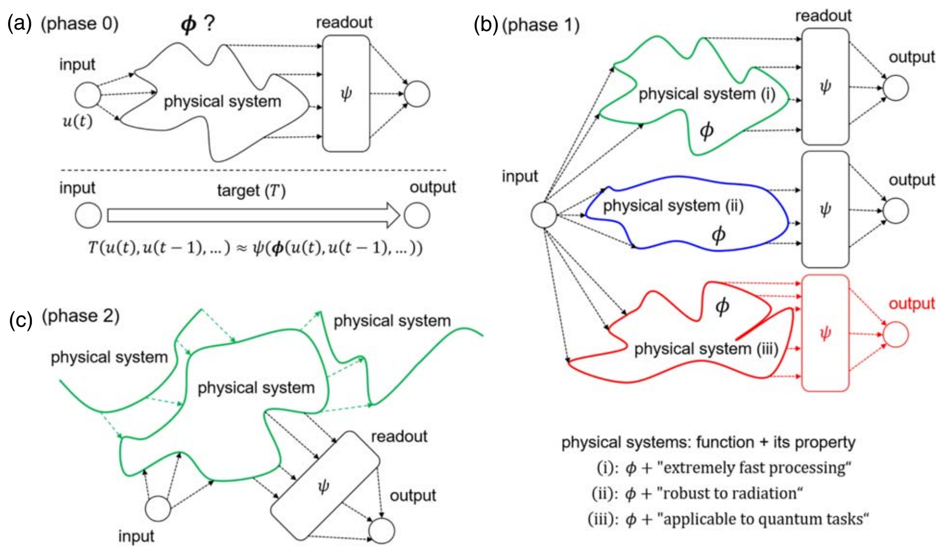
(a) (phase 0)
ϕ ?
readout
input
physical system
ψ
$u(t)$
input
target (T)
output
$T(u(t), u(t-1), \ldots) \approx \psi(\phi(u(t), u(t-1), \ldots))$
(b) (phase 1)
readout
output
physical system (i)
ϕ
input
physical system (ii)
physical system (iii)
physical systems: function + its property
(i): ϕ + "extremely fast processing"
(ii): ϕ + "robust to radiation"
(iii): ϕ + "applicable to quantum tasks"
(c) (phase 2)
physical system
readout
output
input

Fig. 7. (Color online) Three phases in PRC. (a) Phase 0. PRC can be used as a method to infer the information processing capability of natural physical dynamics. (b) Phase 1. Physical properties, which are potentially different according to the type of physics, are added to the reservoir in PRC. (c) Phase 2. PRC enables to exploit physical dynamics as a computational resource that is already functioning for different purposes. See the text for details.

the physical system. This penetration is the basics and is fundamentally important in the PRC framework and can thus be taken as a ground basis, which we call phase 0.

We should note, however, that once the function ϕ of the physical system is revealed, then because it is a mathematical description in principle, there is no meaning to use the actual physical system as an information-processing device anymore, but we can implement the same functionality of the physical system using a conventional PC. If we only stick to this perspective, then PRC does not differ so much from the original RC anymore. Shortly, the diversity we can find here is in fact the diversity of function ϕ. Now, much like as we overviewed from the examples starting from wetwares, PRC has the potential to go beyond this perspective. That is, the PRC framework can deal with a property that is not described in ϕ. This point is elaborated subsequently in phase 1 and phase 2.

6.2. Phase 1: physical properties of a computer

A computer is a machine that is designed for computation. As long as it is made of a physical entity, it inevitably and, sometimes unexpectedly, adds physical and material properties to the system that are not always directly connected to the computational purpose (these properties can present as both advantages and disadvantages for the user). We have clearly confirmed this point using the example of wetware, and this constraint is also true for PRC. Even if you implement the same computation, depending on the type of physics you exploit, you may gain additional or unexpected properties beyond the computation itself [Fig. 7(b)]. For example, if you use a laser as a computational resource, you can implement an extremely fast computation, or if you use water as a substrate, the system will be tolerant of water [Fig. 7(b)]. Many currently discussed assets of physical reservoirs can be understood from this perspective. In particular, spintronics devices have been gaining attention as an appropriate substrate for PRC because of their compactness, high-speed processing, and energy efficiency while being able to function at normal temperatures.[73] These assets are somewhat common in the computer science field, but spintronics devices also contain an interesting additional property: they show high durability in radioactive environments[183] [Fig. 7(b)]. This property opens up the potential for spintronics reservoirs to be used as a computational substrate in extreme environments where conventional electronic devices break down or do not function at all. Another example can be found in quantum RC. Since the first conception to exploit quantum dynamics as a reservoir in Ref. 85, there have been many variants and extensions proposed in the literature.[86–90] In quantum RC, by using the property of quantum computational supremacy, a huge amount of computational nodes can be equipped, which then provide a direct influence to the information-processing capability of the system.[85,86] Another important property is that because quantum RC exploits quantum dynamics, it is capable of implementing a quantum task (a task defined in the quantum scale) [Fig. 7(b)]. In Ref. 88, the preparation of desired quantum states, such as single-photon states, Schrödinger's cat states, and two-mode entangled states, is introduced as an effective application domain.

To induce these assets, which originate from the physical properties of a reservoir, current technology still requires conventional electronics and external devices, such as for the readout part, to maintain the temperature during the reservoir executions and to make the physical reservoir work in the real environment. This is a weakness of currently available technologies; these points should be improved, and a novel scheme should be proposed in the future.

6.3. Phase 2: exploiting a physical substrate that is not made for computation for computation

If we think of the body of robot, it is, of course, not made for computation. The body is an essential constituent of a robot and is inevitably associated when generating behaviors. That is, the robot's intended functionality is to realize behaviors in the real world. As we have seen in the example of soft robots, if the body itself exerts certain dynamic conditions, then according to the PRC framework, the body can also be used as a computational resource [Fig. 7(c)]. This implies that the two functionalities-"behavioral generation" and "information

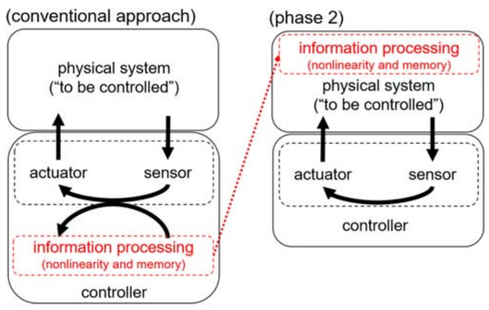

Fig. 8. (Color online) Schematics of the closed-loop control in phase 2 of PRC. In conventional control, information processing is prepared outside a system that is to be controlled or acted on (left diagram). In phase 2, information processing is accompanied by the behavior of the system (right diagram). The system behaves in a certain manner and performs information processing simultaneously. Note that the required information processing to generate behavioral control can be bypassed from the digital processor and embedded in the system itself.

processing"-are associated with the same physical body. Then, when the robot generates behavior, we can simultaneously use its dynamics for information processing. Considering this property, as we confirmed in the above examples of soft robotic arms, together with an incorporation of the feedback-loop, the resulting body dynamics of the target behavior can be exploited to calculate the target motor command that controls its own behavior. This shows that this approach is more efficient than any other controllers attached externally to realize target robotic behavior [Fig. 8].

Phase 2 may be classified as a derivative of Phase 1, but the major turn from Phase 1 to Phase 2 is that in the latter, the physical substrate is not prepared for computational purposes whatsoever in the first place. The most interesting point of PRC among other computational frameworks can be found here. PRC can easily generate the transition from Phase 1 to Phase 2. This is because the RC framework allows one to exploit the natural dynamics of physical systems for information processing. Accordingly, in PRC, we do not need to precisely design the physical substrate specific to target computation in many cases; rather, the implemented information processing depends on the input-driven dynamics of the physical substrate, which results in a diversity of information processing.

Then, which kind of physical reservoir is classified in Phase 2 other than a soft robotic arm inspired by an octopus? This question is a fundamental theme that should be further explored in the field of PRC. One direction would be to exploit real living things, such as animals (e.g. rats, fish, etc.) or the human brain, as a physical reservoir. In principle, living things are free from the intended purposes introduced by users. Needless to say, they are not made for computational purposes. Recently, there has been several studies suggesting that the brain wave of animals and humans exhibit consistent responses against external inputs, and it is expected that the PRC approach can be directly applied to brain waves (e.g. Ref. 184). This direction of research has long been studied in the field of brain-machine-interface. Together with the recent advancement of sensing technology that allows us to monitor massive amounts of data from living things (e.g. Ref. 185), the PRC approach presents a high potential for further study of the issue, and it can be actively applied not only to our daily devices, such as smart phones, but also to wearables and biomedical devices.

Acknowledgments

K. N. would like to acknowledge Taichi Haruna, Katsushi Kagaya, Megumi Akai-Kasaya, Atsushi Uchida, Kazutaka Kanno, Sumito Tsunegi, and Quoc Hoan Tran for their fruitful discussions and thoughtful suggestions. This work was based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). K. N. was supported by JSPS KAKENHI Grant Nos. JP18H05472 and by MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant No. JPMXS0118067394.

ORCID iDs

Kohei Nakajima https://orcid.org/0000-0001-5589-4054

1) H. Jaeger, German National Research Center for Information Technology GMD Technical Report 148, 2001, p. 13.
2) H. Jaeger, *Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and The "Echo State Network" Approach* (GMD-Forschungszentrum Informationstechnik, Bonn, 2002), Vol. 5, p. 01.
3) H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," Science **304**, 78 (2004).
4) W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations," Neural Comput. **14**, 2531 (2002).
5) D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," Neural Netw. **20**, 391 (2007).
6) B. Schrauwen, D. Verstraeten, and J. Van Campenhout, "An overview of reservoir computing: theory, applications and implementations," Proc. 15th European Symp. on Artificial Neural Networks, 2007, p. 471.
7) M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Comput. Sci. Rev. **3**, 127 (2009).
8) M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," KI-Künstliche Intelligenz **26**, 365 (2012).
9) P. J. Werbos, "Backpropagation through time: what it does and how to do it," Proc. IEEE **78**, 1550 (1990).
10) Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Trans. Neural Netw. **5**, 157 (1994).
11) M. H. Tong, A. D. Bickett, E. M. Christiansen, and G. W. Cottrell, "Learning grammatical structure with echo state networks," Neural Netw. **20**, 424 (2007).
12) M. D. Skowronski and J. G. Harris, "Automatic speech recognition using a predictive echo state network classifier," Neural Netw. **20**, 414 (2007).
13) N. Schaetti, M. Salomon, and R. Couturier, "Echo state networks-based reservoir computing for mnist handwritten digits recognition," Proc. 2016 IEEE Int. Conf. on Computational Science and Engineering (CSE) and IEEE Int. Conf. on Embedded and Ubiquitous Computing (EUC) and 15th Int. Sym. on Distributed Computing and Applications for Business Engineering (DCABES), 2016, p. 484.
14) I. Ilies, H. Jaeger, O. Kosuchinas, M. Rincon, V. Sakenas, and N. Vaskevicius, "Stepping forward through echoes of the past: forecasting with echo state networks," Proc. 2006/07 Forecasting Competition for Neural Networks & Computational Intelligence (NN3), 2007, p. 1.
15) X. Lin, Z. Yang, and Y. Song, "Short-term stock price prediction based on echo state networks," Expert Syst. Appl. **36**, 7313 (2009).
16) J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).
17) M. Inada, Y. Tanaka, H. Tamukoh, K. Tateno, T. Morie, and Y. Katori, "Prediction of sensory information and generation of motor commands for autonomous mobile robots using reservoir computing," Proc. 2019 Int. Symp. on Nonlinear Theory and its Applications (NOLTA2019), 2019, p. 333.
18) M. Salmen and P. G. Ploger, "Echo state networks used for motor control," Proc. 2005 IEEE Int. Conf. Robotics and Automation (ICRA), 2005, p. 1953.
19) T. Li, K. Nakajima, M. Calisti, C. Laschi, and R. Pfeifer, Proc. Int. Conf. on Mechatronics and Automation (ICMA), 2012, p. 948.

20) T. Li, K. Nakajima, M. Cianchetti, C. Laschi, and R. Pfeifer, Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2012, p. 4918.
21) T. Li, K. Nakajima, and R. Pfeifer, Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2013, p. 1288.
22) C. Hartmann, J. Boedecker, O. Obst, S. Ikemoto, and M. Asada, "Real-time inverse dynamics learning for musculoskeletal robots based on echo state Gaussian process regression," Proc. of Robotics: Science and Systems VIII, 2012, p. 15.
23) J. Kuwabara, K. Nakajima, R. Kang, D. T. Branson, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, Proc. Int. Joint Conf. on Neural Networks (IJCNN), 2012, p. 1.
24) F. Wyffels and B. Schrauwen, "Design of a central pattern generator using reservoir computing for learning human motion," Proc. 2009 Advanced Technologies for Enhanced Quality of Life, 2009, p. 118.
25) W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," IEEE Internet Things J. **3**, 637 (2016).
26) J. Fonollosa, S. Sheik, R. Huerta, and S. Marco, "Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring," Sens. Actuators B **215**, 618 (2015).
27) R. Sakurai, M. Nishida, H. Sakurai, Y. Wakao, N. Akashi, Y. Kuniyoshi, Y. Minami, and K. Nakajima, "Emulating a sensor using soft material dynamics: a reservoir computing approach to pneumatic artificial muscle," Proc. IEEE Int. Conf. on Soft Robotics (RoboSoft), 2020 (in press).
28) J. C. Principe and B. Chen, "Universal approximation with convex optimization: gimmick or reality?," IEEE Comput. Intell. Mag. **10**, 68 (2015).
29) L. Grigoryeva and J.-P. Ortega, "Echo state networks are universal," Neural Netw. **108**, 495 (2018).
30) L. Gonon and J.-P. Ortega, "Reservoir computing universality with stochastic inputs," IEEE Trans. Neural Netw. Learn. Syst. **31**, 100 (2020).
31) S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735 (1997).
32) K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," Proc. 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2014, p. 1724.
33) M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," Proc. 33rd Int. Conf. on Int. Conf. on Machine Learning, 2016, Vol. 48, p. 1120, [JMLR.org].
34) L. Jing, Y. Shen, T. Dubcek, J. Peurifoy, S. A. Skirlo, Y. LeCun, M. Tegmark, and M. Soljacic, "Tunable efficient unitary neural networks (EUNN) and their application to rnns," ICML, PMLR, 2017, p. 1733.
35) P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Networks **126**, 191 (2020).
36) M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: the sequential learning problem," Psychol. Learn. Motivation **24**, 109 (1989).
37) R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions," Psychol. Rev. **97**, 285 (1990).
38) J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proc. Natl Acad. Sci. **114**, 3521 (2017).
39) S. W. Lee, J. H. Kim, J. Jun, J. W. Ha, and B. T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," NIPS'17: Proc. of the 31st Int. Conf. on Neural Information Processing, p. 4655.
40) X. He and H. Jaeger, "Overcoming catastrophic interference by conceptors," Jacobs University Technical Report Nr 35, (2017).
41) R. Toral, C. R. Mirasso, E. Hernandez-Garcia, and O. Piro, "Analytical and numerical studies of noise-induced synchronization of chaotic systems," CHAOS **11**, 665 (2001).
42) I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," Neural Netw. **35**, 1 (2012).
43) G. Manjunath and H. Jaeger, "Echo state property linked to an input: exploring a fundamental characteristic of recurrent neural networks," Neural Comput. **25**, 671 (2013).
44) M. Komatsu, T. Yaguchi, and K. Nakajima, "Algebraic approach towards the exploitation of 'softness': the input–output equation for morphological computation," Int. J. Robot. Res.
45) Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," CHAOS **28**, 061104 (2018).
46) H. Jaeger, GMD Report 152 German National Research Center for Information Technology, 2001.
47) O. L. White, D. D. Lee, and H. Sompolinsky, "Short-term memory in orthogonal neural networks," Phys. Rev. Lett. **92**, 148102 (2004).
48) S. Ganguli, D. Huh, and H. Sompolinsky, "Memory traces in dynamical systems," Proc. Natl Acad. Sci. USA **105**, 18970 (2008).
49) M. Hermans and B. Schrauwen, "Memory in linear recurrent neural networks in continuous time," Neural Netw. **23**, 341 (2010).
50) A. Rodan and P. Tino, "Minimum complexity echo state network," IEEE Trans. Neural Netw. **22**, 131 (2010).
51) A. Goudarzi and D. Stefanovic, "Towards a calculus of echo state networks," Proc. Comput. Sci. **41**, 176 (2014).
52) T. Toyoizumi and L. F. Abbott, "Beyond the edge of chaos: amplification and temporal integration by recurrent networks in the chaotic regime," Phys. Rev. E **84**, 051908 (2010).
53) J. Schuecker, S. Goedeke, and M. Helias, "Optimal sequence memory in driven random networks," Phys. Rev. X **8**, 041029 (2018).
54) T. Haruna and K. Nakajima, "Optimal short-term memory before the edge of chaos in driven random recurrent networks," Phys. Rev. E **100**, 062312 (2019).
55) J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," Sci. Rep. **2**, 514 (2012).
56) M. Inubushi and K. Yoshimura, "Reservoir computing beyond memory-nonlinearity trade-off," Sci. Rep. **7**, 10199 (2017).
57) M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer, "A substrate-independent framework to characterize reservoir computers," Proc. R. Soc. A **475**, 20180723 (2019).
58) N. Bertschinger and T. Natschläger, "Real-time computation at the edge of chaos in recurrent neural networks," Neural Comput. **16**, 1413 (2004).
59) R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," Neural Netw. **20**, 323 (2007).
60) J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, "Information processing in echo state networks at the edge of chaos," Theory Biosci. **131**, 205 (2012).
61) G. Wainrib and M. N. Galtier, "A local echo state property through the largest Lyapunov exponent," Neural Netw. **76**, 39 (2016).
62) K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, P. Bienstman, and J. Van Campenhout, "Toward optical signal processing using photonic reservoir computing," Opt. Express **16**, 11182 (2008).
63) L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing," Opt. Express **20**, 3241 (2012).
64) Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," Sci. Rep. **2**, 287 (2012).
65) F. Duport, B. Schneider, A. Smerieri, A. M. Haelterman, and S. Masser, "All-optical reservoir computing," Opt. Express **20**, 22783 (2012).
66) D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," Nat. Commun. **4**, 1364 (2013).
67) K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," Nat. Commun. **5**, 3541 (2014).
68) J. Nakayama, K. Kanno, and A. Uchida, "Laser dynamical reservoir computing with consistency: an approach of a chaos mask signal," Opt. Express **24**, 8679 (2016).
69) L. Larger, A. Baylon-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, "High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification," Phys. Rev. X **7**, 011015 (2017).
70) G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," Nanophotonics **6**, 561 (2017).
71) J. Bueno, S. Maktoobi, L. Froehly, I. Fischer, M. Jacquot, L. Larger, and D. Brunner, "Reinforcement learning in a large-scale photonic recurrent neural network," Optica **5**, 756 (2018).
72) K. Takano, C. Sugano, M. Inubushi, K. Yoshimura, S. Sunada, K. Kanno, and A. Uchida, "Compact reservoir computing with photonic integrated circuit," Opt. Express **26**, 29424 (2018).
73) S. Tsunegi, T. Taniguchi, K. Nakajima, S. Miwa, K. Yakushiji, A. Fukushima, S. Yuasa, and H. Kubota, "Physical reservoir computing based on spin torque oscillator with forced synchronization," Appl. Phys. Lett. **114**, 164101 (2019).
74) T. Furuta, K. Fujii, K. Nakajima, S. Tsunegi, H. Kubota, Y. Suzuki, and S. Miwa, "Macromagnetic simulation for reservoir computing utilizing spin dynamics in magnetic tunnel junctions," Phys. Rev. Appl. **10**, 034063 (2018).
75) J. Torrejon et al., "Neuromorphic computing with nanoscale spintronic oscillators," Nature **547**, 428 (2017).

76) S. Tsunegi, T. Taniguchi, S. Miwa, K. Nakajima, K. Yakushiji, A. Fukushima, S. Yuasa, and H. Kubota, "Evaluation of memory capacity of spin torque oscillator for recurrent neural networks," Jpn. J. Appl. Phys. **57**, 120307 (2018).
77) R. Nakane, G. Tanaka, and A. Hirose, "Reservoir computing with spin waves excited in a garnet film," IEEE Access **6**, 4462 (2018).
78) D. Marković et al., "Reservoir computing with the frequency, phase, and amplitude of spin-torque nano-oscillators," Appl. Phys. Lett. **114**, 012409 (2019).
79) H. Nomura, T. Furuta, K. Tsujimoto, Y. Kuwabiraki, F. Peper, E. Tamura, S. Miwa, M. Goto, R. Nakatani, and Y. Suzuki, "Reservoir computing with dipole-coupled nanomagnets," Jpn. J. Appl. Phys. **58**, 070901 (2019).
80) T. Kanao, H. Suto, K. Mizushima, H. Goto, T. Tanamoto, and T. Nagasawa, "Reservoir computing on spin-torque oscillator array," Phys. Rev. Appl. **12**, 024052 (2019).
81) M. Riou et al., "Temporal pattern recognition with delayed-feedback spin-torque nano-oscillators," Phys. Rev. Appl. **12**, 024049 (2019).
82) W. Jiang, L. Chen, K. Zhou, L. Li, Q. Fu, Y. Du, and R. H. Liu, "Physical reservoir computing using magnetic skyrmion memristor and spin torque nano-oscillator," Appl. Phys. Lett. **115**, 192403 (2019).
83) H. Nomura, K. Tsujimoto, M. Goto, N. Samura, R. Nakatani, and Y. Suzuki, "Reservoir computing with two-bit input task using dipole-coupled nanomagnet array," Jpn. J. Appl. Phys. **59**, SEEG02 (2019).
84) F. A. Araujo et al., "Role of non-linear data processing on speech recognition task in the framework of reservoir computing," Sci. Rep. **10**, 328 (2020).
85) K. Fujii and K. Nakajima, "Harnessing disordered-ensemble quantum dynamics for machine learning," Phys. Rev. Appl. **8**, 024030 (2017).
86) K. Nakajima, K. Fujii, M. Negoro, K. Mitarai, and M. Kitagawa, "Boosting computational power through spatial multiplexing in quantum reservoir computing," Phys. Rev. Appl. **11**, 034021 (2019).
87) S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. Liew, "Quantum reservoir processing," npj Quantum Inf. **5**, 35 (2019).
88) S. Ghosh, T. Paterek, and T. C. Liew, "Quantum neuromorphic platform for quantum state preparation," Phys. Rev. Lett. **123**, 260404 (2019).
89) S. Ghosh, T. Krisnanda, T. Paterek, and T. C. Liew, Universal quantum reservoir computing arXiv:2003.09569.
90) J. Chen, H. I. Nurdin, and N. Yamamoto, Temporal information processing on noisy quantum computers arXiv:2001.09498.
91) A. Z. Stieg, A. V. Avizienis, H. O. Sillin, C. Martin-Olmos, M. Aono, and J. K. Gimzewski, "Emergent criticality in complex Turing b-type atomic switch networks," Adv. Mater. **24**, 286 (2012).
92) H. O. Sillin, R. Aguilera, H. H. Shieh, A. V. Avizienis, M. Aono, A. Z. Stieg, and J. K. Gimzewski, "A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing," Nanotechnology **24**, 384004 (2013).
93) M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer, "Evolving carbon nanotube reservoir computers," Proc. Int. Conf. on Unconventional Computation and Natural Computation, 2016, p. 49.
94) M. Dale, J. F. Miller, and S. Stepney, "Reservoir computing as a model for *in-materio* computing," Advances in Unconventional Computing. Emergence, Complexity and Computation (Springer, Cham, (2017) Vol. 22, p. 533.
95) C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, "Reservoir computing using dynamic memristors for temporal information processing," Nat. Commun. **8**, 2204 (2017).
96) K. S. Scharnhorst, J. P. Carbajal, R. C. Aguilera, E. J. Sandouk, M. Aono, A. Z. Stieg, and J. K. Gimzewski, "Atomic switch networks as complex adaptive systems," Jpn. J. Appl. Phys. **57**, 03ED02 (2018).
97) H. Tanaka, M. Akai-Kasaya, A. TermehYousefi, L. Hong, L. Fu, H. Tamukoh, D. Tanaka, T. Asai, and T. Ogawa, "A molecular neuromorphic network device consisting of single-walled carbon nanotubes complexed with polyoxometalate," Nat. Commun. **9**, 2693 (2018).
98) J. Moon, W. Ma, J. H. Shin, F. Cai, C. Du, S. H. Lee, and W. D. Lu, "Temporal data classification and forecasting using a memristor-based reservoir computing system," Nat. Electron. **2**, 480 (2019).
99) R. Midya, Z. Wang, S. Asapu, X. Zhang, M. Rao, W. Song, Y. Zhuo, N. Upadhyay, Q. Xia, and J. J. Yang, "Reservoir computing using diffusive memristors," Adv. Intell. Syst. **1**, 1900084 (2019).
100) L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," Nat. Commun. **2**, 468 (2011).
101) L. Appeltant, G. Van der Sande, J. Danckaert, and I. Fischer, "Constructing optimized binary masks for reservoir computing with delay systems," Sci. Rep. **4**, 3629 (2014).
102) M. L. Alomar, V. Canals, V. Martínez-Moll, and J. L. Rosselló, "Low-cost hardware implementation of reservoir computers," Proc. 2014 24th Int. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2014, p. 1.
103) M. L. Alomar, M. C. Soriano, M. Escalona-Morán, V. Canals, I. Fischer, C. R. Mirasso, and J. L. Rosselló, "Digital implementation of a single dynamical node reservoir computer," IEEE Trans. Circuits Syst. II **62**, 977 (2015).
104) P. Antonik, A. Smerieri, F. Duport, M. Haelterman, and S. Massar, "FPGA implementation of reservoir computing with online learning," Proc. 24th Belgian-Dutch Conf. on Machine Learning, 2015, p. 1.
105) M. L. Alomar, V. Canals, N. Perez-Mora, V. Martínez-Moll, and J. L. Rosselló, "FPGA-based stochastic echo state networks for time-series forecasting," Comput. Intell. Neurosci. **2016**, 3917892 (2016).
106) M. L. Alomar, V. Canals, A. Morro, A. Oliver, and J. L. Rossello, "Stochastic hardware implementation of liquid state machines," Proc. 2016 Int. Joint Conf. on Neural Networks (IJCNN), 2016, p. 1128.
107) P. Antonik, *Application of FPGA to Real-Time Machine Learning: Hardware Reservoir Computers and Software Image Processing* (Springer, Berlin, 2018).
108) H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, "Towards a theoretical foundation for morphological computation with compliant bodies," Biol. Cybern. **105**, 355 (2011).
109) H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, "The role of feedback in morphological computation with compliant bodies," Biol. Cybern. **106**, 595 (2012).
110) K. Caluwaerts, M. D'Haene, D. Verstraeten, and B. Schrauwen, "Locomotion without a brain: physical reservoir computing in tensegrity structures," Artificial Life **19**, 35 (2013).
111) K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2013, p. 1496.
112) K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm," Front. Comput. Neurosci. **7**, 91 (2013).
113) Q. Zhao, K. Nakajima, H. Sumioka, H. Hauser, and R. Pfeifer, Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2013, p. 1445.
114) K. Nakajima, T. Li, H. Hauser, and R. Pfeifer, "Exploiting short-term memory in soft body dynamics as a computational resource," J. R. Soc. Interface **11**, 20140437 (2014).
115) K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," J. R. Soc. Interface **11**, 20140520 (2014).
116) K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, "Information processing via physical soft body," Sci. Rep. **5**, 10487 (2015).
117) K. Nakajima, *Brain Evolution by Design* (Springer, Japan, 2017), p. 403.
118) J. C. Coulombe, M. C. York, and J. Sylvestre, "Computing with networks of nonlinear mechanical oscillators," PLoS One **12**, e0178663 (2017).
119) G. Urbain, J. Degrave, B. Carette, J. Dambre, and F. Wyffels, "Morphological properties of mass-spring networks for optimal locomotion learning," Front. Neurorobotics **11**, 16 (2017).
120) Y. Yamanaka, T. Yaguchi, K. Nakajima, and H. Hauser, *Lecture Notes in Computer Science 11141: Int. Conf. on Artificial Neural Networks (ICANN2018)* (Springer, Cham, 2018), p. 781.
121) K. Nakajima, T. Li, and N. Akashi, *Robotic Systems and Autonomous Platforms: Advances in Materials and Manufacturing* (Woodhead Publishing in Materials, Sawston, 2018), p. 181.
122) K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, "Exploiting the dynamics of soft materials for machine learning," Soft Robot. **5**, 339 (2018).
123) E. A. Torres, K. Nakajima, and I. S. Godage, Proc. IEEE Int. Conf. on Soft Robotics (RoboSoft), 2019, p. 441.
124) K. Nakajima and T. Aoyagi, "The memory capacity of a physical liquid state machine," IEICE Technical Report (2015), Vol. 115, p. 109.
125) T. Natschläger, W. Maass, and H. Markram, "The "Liquid Computer": a novel strategy for real-time computing on time series," TELEMATIK **8**, 39 (2002).
126) C. Fernando and S. Sojakka, *Lecture Notes in Computer Science 2801* (Springer, Berlin, 2003), p. 588.
127) K. Goto, K. Nakajima, and H. Notsu, Computing with vortices: bridging fluid dynamics and its information-processing capability (2020), arXiv:2001.08502.
128) D. Nikolić, S. Häusler, W. Singer, and W. Maass, "Distributed fading memory for stimulus properties in the primary visual cortex," PLoS Biol. **7**, e1000260 (2009).

129) M. R. Dranias, H. Ju, E. Rajaram, and A. M. J. VanDongen, "Short-term memory in networks of dissociated cortical neurons," J. Neurosci. **33**, 1940 (2013).
130) T. Kubota, K. Nakajima, and H. Takahashi, *Lecture Notes in Computer Science 11731: Int. Conf. on Artificial Neural Networks (ICANN2019)* (Springer, Cham, 2019), p. 137.
131) G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: a review," Neural Netw. **115**, 100 (2019).
132) K. Nakajima and I. Fischer (ed.) *Reservoir Computing: Theory, Physical Implementations, and Applications* (Springer, Berlin, 2020), in preparation.
133) D. Prychynenko, M. Sitte, K. Litzius, B. Krüger, G. Bourianoff, M. Kläui, J. Sinova, and K. Everschor-Sitte, "Magnetic skyrmion as a nonlinear resistive element: a potential building block for reservoir computing," Phys. Rev. Appl. **9**, 014034 (2018).
134) Y. Kawai, J. Park, and M. Asada, "A small-world topology enhances the echo state property and signal propagation in reservoir computing," Neural Netw. **112**, 15 (2019).
135) M. C. Ozturk, D. Xu, and J. C. Príncipe, "Analysis and design of echo state networks," Neural Comput. **19**, 111 (2007).
136) T. Tanaka, K. Nakajima, and T. Aoyagi, "Effect of recurrent infomax on the information processing capability of input-driven recurrent neural networks," Neurosci. Res.
137) D. Norton and D. Ventura, "Preparing more effective liquid state machines using hebbian learning," Proc. 2006 IEEE Int. Joint Conf. on Neural Network, 2006, p. 4243.
138) J. Yin, Y. Meng, and Y. Jin, "A developmental approach to structural self-organization in reservoir computing," IEEE Trans. Autonomous Mental Dev. **4**, 273 (2012).
139) F. Xue, Z. Hou, and X. Li, "Computational capability of liquid state machines with spike-timing-dependent plasticity," Neurocomputing **122**, 324 (2013).
140) Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," arXiv:1909.11942.
141) H. Sompolinsky, A. Crisanti, and H. J. Sommers, "Chaos in random neural networks," Phys. Rev. Lett. **61**, 259 (1988).
142) L. Molgedey, J. Schuchhardt, and H. G. Schuster, "Suppressing chaos in neural networks by noise," Phys. Rev. Lett. **69**, 3717 (1992).
143) D. Sussillo and L. F. Abbott, "Generating coherent patterns of activity from chaotic neural networks," Neuron **63**, 544 (2009).
144) D. Sussillo and L. Abbott, "Transferring learning from external to internal weights in echo-state networks with sparse connectivity," PLoS One **7**, e37372 (2012).
145) B. DePasquale, C. J. Cueva, K. Rajan, G. S. Escola, and L. F. Abbott, "full-FORCE: a target-based method for training recurrent networks," PLoS One **13**, e0191527 (2018).
146) R. Laje and D. V. Buonomano, "Robust timing and motor patterns by taming chaos in recurrent neural networks," Nat. Neurosci. **16**, 925 (2013).
147) V. Goudar and D. V. Buonomano, "Encoding sensory and motor patterns as time-invariant trajectories in recurrent neural networks," eLife **7**, e31134 (2018).
148) K. Inoue, K. Nakajima, and Y. Kuniyoshi, Proc. Int. Symp. on Micro-NanoMechatronics and Human Science (MHS), 2019, p. 60.
149) K. Inoue, K. Nakajima, and Y. Kuniyoshi, Designing spontaneous behavioral switching via chaotic itinerancy arXiv:2002.08332.
150) M. Yamaguchi, Y. Katori, D. Kamimura, H. Tamukoh, and T. Morie, "A chaotic Boltzmann machine working as a reservoir and Its analog VLSI implementation," Proc. of Int. Joint Conf. on Neural Networks (IJCNN 2019), N-20163, 2019.
151) T. Taniguchi, N. Akashi, H. Notsu, M. Kimura, H. Tsukahara, and K. Nakajima, "Chaos in nanomagnet via feedback current," Phys. Rev. B **100**, 174425 (2019).
152) T. Yamaguchi, N. Akashi, K. Nakajima, S. Tsunegi, H. Kubota, and T. Taniguchi, "Synchronization and chaos in a spin-torque oscillator with a perpendicularly magnetized free layer," Phys. Rev. B **100**, 224422 (2019).
153) W. Maass, *TAKEOVER: Who is Doing the Art of Tomorrow (Ars Electronica* (Springer, Berlin, 2001), p. 148.
154) S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with Volterra series," IEEE Trans. Circuits Syst. **32**, 1150 (1985).
155) W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," J. Comput. Syst. Sci. **69**, 593 (2004).
156) R. Solé, M. Moses, and S. Forrest, "Liquid brains, solid brains," Phil.Trans. R. Soc. B **374**, 20190040 (2019).
157) R. Pfeifer and C. Scheier, *Understanding Intelligence* (MIT Press, Cambridge, MA, 2001).
158) R. Pfeifer and J. Bongard, *How the Body Shapes the Way We Think: A New View of Intelligence* (MIT Press, Cambridge, MA, 2006).
159) R. Pfeifer, M. Lungarella, and F. Iida, "Self-organization, embodiment, and biologically inspired robotics," Science **318**, 1088 (2007).
160) T. McGeer, "Passive dynamic walking," I. J. Robot. Res. **9**, 62 (1990).
161) Q. Zhao, H. Sumioka, X. Yu, K. Nakajima, Z. Wang, and R. Pfeifer, Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), 2012, p. 66.
162) Q. Zhao, K. Nakajima, H. Sumioka, X. Yu, and R. Pfeifer, Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2012, p. 2449.
163) N. Schmidt, M. Hoffmann, K. Nakajima, and R. Pfeifer, "Bootstrapping perception using information theory: case studies in a quadruped robot running on different grounds," Adv. Complex Syst. **16**, 1250078 (2013).
164) Q. Zhao, H. Sumioka, K. Nakajima, X. Yu, and R. Pfeifer, "Spine as an engine: effect of spine morphology on spine-driven quadruped locomotion," Adv. Robot. **28**, 367 (2014).
165) J. C. Liao, "Neuromuscular control of trout swimming in a vortex street: implications for energy economy during the Kármán gait," J. Exp. Biol. **207**, 3495 (2004).
166) A. M. T. Ngouabeu, S. Miyashita, R. M. Füchslin, K. Nakajima, M. Goldi, and R. Pfeifer, Proc. Int. Conf. on the Simulation and Synthesis of Living System (Artificial Life XII), 2010 (Cambridge, MA), (MIT Press), p. 232.
167) S. Miyashita, A. M. T. Ngouabeu, R. M. Füchslin, K. Nakajima, C. Audretsch, and R. Pfeifer, Studies in Computational Intelligence, 2011, Vol. 355, p. 173.
168) K. Nakajima, A. M. T. Ngouabeu, S. Miyashita, M. Goldi, R. M. Füchslin, and R. Pfeifer, "Morphology-induced collective behaviors: dynamic pattern formation in water-floating elements," PLoS One **7**, e37805 (2012).
169) S. Miyashita, K. Nakajima, Z. Nagy, and R. Pfeifer, "Self-organized translational wheeling motion in stochastic self-assembling modules," Artificial Life **19**, 79 (2013).
170) R. Pfeifer and G. Gómez, *Creating Brain-Like Intelligence* (Springer, Berlin, 2009), p. 66.
171) K. Ghazi-Zahedi and N. Ay, "Quantifying morphological computation," Entropy **15**, 1887 (2013).
172) K. Ghazi-Zahedi, *Morphological Intelligence: Measuring the Body's Contribution to Intelligence* (Springer, Berlin, 2019).
173) A. P. Sabelhaus, J. Bruce, K. Caluwaerts, P. Manovi, R. F. Firoozi, S. Dobi, A. M. Agogino, and V. SunSpiral, "System design and locomotion of SUPERball, an untethered tensegrity robot," Proc. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA), 2015, p. 2867.
174) S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: a bioinspired evolution in robotics," Trends Biotechnol. **31**, 287 (2013).
175) C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: technologies and systems pushing the boundaries of robot abilities," Sci. Robot. **1**, eaah3690 (2016).
176) D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," Nature **521**, 467 (2015).
177) T. Li, K. Nakajima, M. J. Kuba, T. Gutnick, B. Hochner, and R. Pfeifer, Vie et Milieu **61**, 211 (2012).
178) K. Nakajima, T. Li, H. Sumioka, M. Cianchetti, and R. Pfeifer, Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), 2011, p. 110.
179) K. Nakajima, T. Li, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), 2012, p. 1273.
180) K. Nakajima, N. Schmidt, and R. Pfeifer, "Measuring information transfer in a soft robotic arm," Bioinsp. Biomim. **10**, 035007 (2015).
181) D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: biological inspiration, state of the art, and future research," Appl. Bionics Biomech. **5**, 99 (2008).
182) L. F. Seoane, "Evolutionary aspects of reservoir computing," Phil. Trans. R. Soc. B **374**, 20180377 (2019).
183) D. Kobayashi et al., "Influence of heavy ion irradiation on perpendicular-anisotropy CoFeB-MgO magnetic tunnel junctions," IEEE Trans. Nucl. Sci. **61**, 1710 (2014).
184) K. Kitajo, T. Sase, Y. Mizuno, and H. Suetani, "Consistency in macroscopic human brain responses to noisy time-varying visual inputs," bioRxiv645499 (2019).
185) C. Stringer, M. Pachitariu, N. Steinmetz, C. B. Reddy, M. Carandini, and K. D. Harris, "Spontaneous behaviors drive multidimensional, brainwide activity," Science **364**, 255 (2019).