Self-learning physical reservoir computer

Md Raf E Ul Shougat
Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, North Carolina 27695, USA

XiaoFu Li and Edmon Perkins*
LAB2701, Atwood, Oklahoma 74827, USA

(Received 28 June 2023; revised 31 October 2023; accepted 14 May 2024; published 5 June 2024)

A self-learning physical reservoir computer is demonstrated using an adaptive oscillator. Whereas *physical reservoir computing* repurposes the dynamics of a physical system for computation through machine learning, *adaptive oscillators* can innately learn and store information in plastic dynamic states. The adaptive state(s) can be used directly as physical node(s), but these plastic states can also be used to self-learn the optimal reservoir parameters for more complex tasks requiring virtual nodes from the base oscillator. Both this self-learning property for reconfigurable computing and the morphable logic gate property of the adaptive oscillator make this an ideal candidate for a multipurpose neuromorphic processor.

DOI: 10.1103/PhysRevE.109.064205

I. INTRODUCTION

Physical reservoir computers (PRCs) are a subset of neuromorphic computers that repurpose the nonlinear dynamics of a physical system for computation [1–6]. PRCs were a natural progression of nonphysical reservoir computers, which are deep neural networks that have hidden layers with untrained weights. Growing from *echo state networks* [7] and *liquid state machines* [8], reservoir computing has several advantages, including lower training costs and robustness to overfitting. Notably, PRCs differ from traditional Turing machines, as the information storage is not static. PRCs have been employed for a wide variety of tasks, including wake word recognition [9], digit recognition [6,9–11], image recognition [12,13], time series prediction [2,6,14–19], and logic operations [20–24]. PRCs have been created from optoelectronics [13], soft robots [25–29], tensegrities [30,31], the Duffing oscillator array [20,20,32], quantum systems [33–35], and limit cycle oscillators [9,36,37].

While PRCs employ machine learning to unlock their computational ability, adaptive oscillators (AOs) have an innate ability to both learn and store information in dynamic plastic states. AOs are inherently a subset of nonlinear oscillators. Although relatively little work has explored these oscillators, they have many potential applications such as central pattern generators [38–41], robotic locomotion [42–44], medical gait analyzers [45], and analog signal analyzers [46,47]. The Hopf AO has been implemented as various circuits [48–50], and its frequency analysis capabilities were benchmarked with a range of tasks. When the base oscillator is a pendulum, it was recently found that adaptive oscillators can learn a resonance condition, which is *not* the external forcing frequency [51,52]. Importantly, this resonance tracking can be utilized for energy-harvesting purposes, without the need for digital frequency analysis. It should be noted that plasticity at the synaptic level is often incorporated into spiking neural networks, with a comprehensive review of this topic provided by Ref. [53]. However, this type of plasticity has not previously been implemented for physical reservoir computing.

Work on the Hopf PRC provided insights into constructing a robust reservoir computer utilizing resonance, Arnold tongues [54,55], and synchronization [24,36,37]. However, it also poses a significant challenge, as these resonance conditions require rather precise frequencies. For Arnold tongues, robustness is gained at the cost of an amplified signal, which increases the required energy.

In this paper, an architecture of a PRC is presented which uses an AO as the reservoir. Notably, the AO itself is a neuromorphic computer that only uses dynamics for memory storage and learning. The PRC composed of an AO exhibits self-learning, which increases its own power as a PRC without any programming. It should be noted that other self-learning has been recently demonstrated, such as a Hamiltonian echo backpropagation technique [56]. The method presented in this paper uses the AO's intrinsic dynamic learning to optimize the PRC's computational ability. Further, this AO PRC has no synapses, so the oscillator's adaptive states are the only source of plasticity for this system, and only one of the oscillator's states is used for creating virtual nodes. This form of self-learning provides enhanced robustness and reconfigurability. Specifically, this reservoir-level self-learning unlocks the potential of the PRC to be used for applications with a wide range of frequencies and sampling rates. In this paper, examples of the PRC's self-learning property are highlighted using broadband time series data with a wide range of sampling rates.

This AO PRC uses time multiplexing to perform different tasks. Notably, this same AO can also be used as a morphable logic gate without the need for multiplexing [57]. Thus, this AO can be used as both a self-learning reconfigurable PRC and a multiplexing-free morphable logic gate. These two

*edmon@lab2701.com

abilities make it an ideal candidate for a multipurpose neuromorphic processor.

II. ADAPTIVE OSCILLATOR PHYSICAL RESERVOIR COMPUTER EQUATIONS

Based on the classical Hopf oscillator [48], the Hopf-based three-state AO is described as follows [48–50,58]:

$$\begin{aligned}\dot{x} &= (\mu - (x^2 + y^2))x - \omega y + a(f(t)),\\ \dot{y} &= (\mu - (x^2 + y^2))y + \omega x,\\ \dot{\omega} &= -k(f(t))y.\end{aligned} \tag{1}$$

Here, $f(t)$ is a normalized, time-varying external force acting on the adaptive oscillator. This AO has a Hopf oscillator as its base (e.g., x and y). In the classical Hopf oscillator model, the Hopf oscillator has a constant resonance parameter, ω_0, while the AO in Eq. (1) has a dynamic frequency state, ω. When this AO is forced with a sinusoid, such as $\sin(\Omega t)$, the ω state will converge to Ω. Appendix A discusses the nonexistence of limit cycles for adaptive oscillators.

The AO given by Eq. (1) is used as a PRC in this paper. As the ω state innately learns and stores vibratory information, the AO is implemented as a self-learning PRC that leverages reservoir-level learning to better exploit the reservoir computing architecture. The AO is further modified to inject the external force into the limit cycle radius term, such that

$$\begin{aligned}\dot{x} &= (\mu f(t) - (x^2 + y^2))x - \omega y + a(f(t)),\\ \dot{y} &= (\mu f(t) - (x^2 + y^2))y + \omega x,\\ \dot{\omega} &= -k(f(t))y.\end{aligned} \tag{2}$$

$f(t)$ is a continuous function, which is constructed from a discrete function, $h(z)$, and a sinusoid. First, g is a continuous function, which is equal to a value of h for a length of time equal to the pseudoperiod, T_p. T_p is an arbitrary length of time, which is used to also define a pseudoperiod as $\omega_p = \frac{2\pi}{T_p}$. The continuous function, $g(t)$, is defined as

$$g(t) = h(z_n) \text{ for } (n)T_p \leqslant t < (n+1)T_p, n \in \mathbb{Z}^+. \tag{3}$$

Thus, $g(t)$ holds the value of $h(z_n)$ for a single pseudoperiod. For logic tasks, $h(z)$ is composed of a random series of -1 (e.g., "False") and $+1$ (e.g., "True") values. For time series tasks, $h(z)$ is a normalized time series. The forcing function, $f(t)$, is $g(t)$ multiplied by a sinusoid of frequency Ω. During each pseudoperiod, *virtual nodes* are collected from the y state and a ridge regression is applied for training. In this paper, a regularization penalty of 0.001 was used for the ridge regression, 80% of the data was used for training, and the remaining 20% was used for testing.

Effectively, this PRC undergoes two types of learning. The first type of learning is *self-learning*, which is a property of the adaptive oscillator itself. Self-learning in this context means that the adaptive oscillator learns frequency information from the time-varying signal, $f(t)$, which increases the efficacy of the PRC to compute different tasks. This is quite important as different signals have a wide range of timescales and sampling rates. Without self-learning, the PRC would only work for narrow bands of timescales, as exhibited in Sec. IV. Said another way, this PRC capitalizes on the adaptive oscillator's innate ability to learn certain properties of signals to increase its efficacy as a PRC with reprogrammability.

The second type of learning used in this PRC architecture is ridge regression. Ridge regression is a commonly used method to obtain a set of weights for PRCs (a more thorough description of the ridge regression may be found in Ref. [59]), and it is often chosen because of its simplicity. The ridge regression is used to find a set of weights that minimize the error between the dot product of the virtual nodes and weights and some desired output (e.g., a prediction of a future value, a logic gate, speech recognition, etc.). Thus, the ridge regression is used to reprogram the PRC for different tasks.

III. SELF-LEARNING BENCHMARK TASKS

Notably, physical systems oscillate at considerably different timescales, which necessitates vastly different sampling rates. For instance, the human heart is commonly monitored with a smart watch, and the heart's range is often between 40 and 140 beats per minute [60]. With this low frequency, heart data can be sampled at a relatively low rate. On the other hand, human speech has information content in the range from approximately 100 Hz to 3000 Hz (with higher harmonics being mostly only useful in a musical context) [61]. Thus, speech requires a substantially higher sampling rate to preserve information. As the computational ability of the PRC is dependent on the resonance frequency of the PRC's oscillator (as discussed in Ref. [36]), different sampling rates necessitate a modified resonance frequency. To highlight the self-learning of the AO PRC, the results for several benchmark tasks are shown in this section, which includes heart, speech, and chaotic time-series prediction tasks, with an image processing benchmark provided in Appendix C.

A. Heart ECG prediction

A recording of a heart ECG signal [62] from PhysioNet [63] is used in this section to perform a prediction task. This recording had a sampling rate of 1000 Hz, but was downsampled to 250 Hz. This slower frequency preserves much of the major features of the time history, but this more modest sampling rate could be more easily managed by wearables. A time series of 1600 seconds was used for training and 400 seconds for testing.

By setting the pseudoperiod equal to this sampling rate and choosing $\Omega = 2\omega_p$, the resonance frequency of the AO adapts to the Ω value. This allows the AO PRC to perform real-time processing on the heart data. An example of this is shown in Fig. 1.

B. Speech prediction

The spoken digit data set is composed of multiple individuals saying single digit numbers, which are recorded at a sampling rate of 8000 Hz [64]. This sampling rate is a popular choice, as it is high enough to preserve information content of human speech, but low enough to allow for faster processing for voice recognition tasks. A time series of 80 seconds was used for training, and 20 seconds for testing.

Following the same procedure as discussed for the heart ECG data set, the pseudoperiod is set to the sampling rate

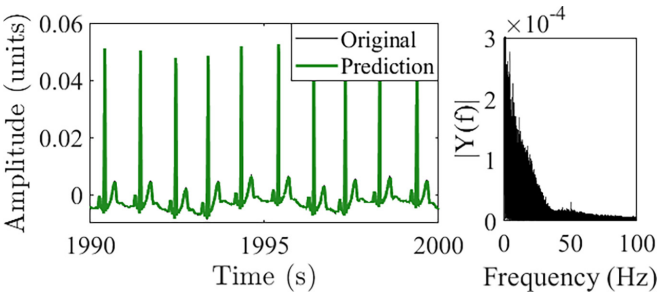

FIG. 1. AO PRC performs one-step-ahead heart ECG signal prediction task (left), with the fast Fourier transform of the signal for reference (right). The RMSE for the test portion of this task was 0.00086, while the RMSE of the baseline (i.e., assuming a prediction equal to the average value of the signal) was 0.02104. Here, $\mu = 100$, $k = 100$, $a = 100$, and $N = 200$.

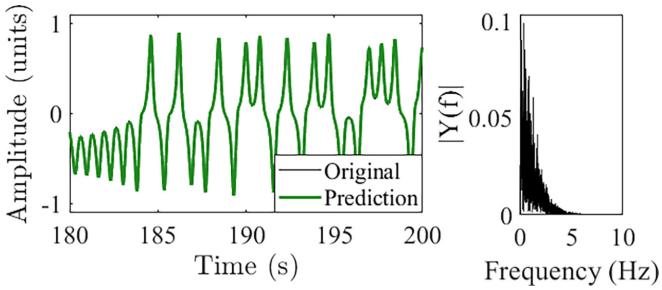

FIG. 3. AO PRC performs one-step-ahead Lorenz chaotic time series prediction (left), with the fast Fourier transform of the signal for reference (right). The RMSE for the test portion of this task was 0.04665, while the RMSE of the baseline (i.e., assuming a prediction equal to the average value of the signal) was 0.43561. Here, $\mu = 100$, $k = 100$, $a = 100$, and $N = 200$.

and $\Omega = 2\omega_p$. The resonance frequency of the AO adapts to the Ω value. This allows the AO PRC to perform real-time processing on the audio data. An example of this is shown in Fig. 2.

C. Lorenz chaotic time series prediction

The Lorenz time series is a classical chaotic time series. The Lorenz system is given below, which was simulated with the parameters $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$:

$$\begin{aligned} \dot{a} &= \sigma(b - a), \\ \dot{b} &= a(\rho - c) - b, \\ \dot{c} &= ab - \beta c. \end{aligned} \tag{4}$$

Following the same procedure as before, the Lorenz time series prediction task is shown in Fig. 3. A time series of 160 units was used for training and 40 units for testing.

IV. PARAMETRIC STUDIES

In this section, an exclusive OR (XOR) task will be used as a benchmark to quantify the behavior of both the Hopf PRC and AO PRC. For logical tasks, the most relevant metric is a modified version of Shannon's information rate. Comparing this to the original work [65], the correct output of the logical task acts as the sent signal, and the predicted output of the task from the PRC acts as the received signal. The information rate, IR, is a function of $H(x)$, the *Shannon entropy*, and $H_y(x)$, the *conditional entropy*. The amount of information in a signal is calculated from the Shannon entropy, which sets an upper limit on the information rate such that $\mathrm{IR} = H(x) - H_y(x)$. The probability of an incorrect bit is given by the conditional entropy. A detailed explanation of this modified Shannon's information rate is provided in Ref. [36]. For the XOR task considered here, the PRC works perfectly as a logic gate when $\mathrm{IR} = 1.0$.

An example of the XOR task prediction is shown for reference in Fig. 4. A series of 3200 units was used for training and 800 units for testing.

By focusing on a set forcing frequency, Ω, a computational Arnold tongue may be seen [36]. This behavior is shown in Fig. 5. The Hopf PRC is highly susceptible to mistuning of the static ω_0 term. The computational Arnold tongue suggests that this may be partially compensated by increasing the gain. However, this comes at the price of increasing the energy expenditure required by the Hopf PRC.

Next, the behavior of three different cases for ω_0 are compared to the behavior of the AO in Fig. 6. When $\omega_0 = \frac{\omega_p}{\frac{1}{2}}$, the Hopf PRC has poor performance for most values of $\frac{\omega_p}{\Omega}$, although it has perfect performance at $\frac{\omega_p}{\Omega} = \frac{1}{2}$. When $\omega_0 = \frac{\omega_p}{\Omega}$, it has nominal performance at several Farey

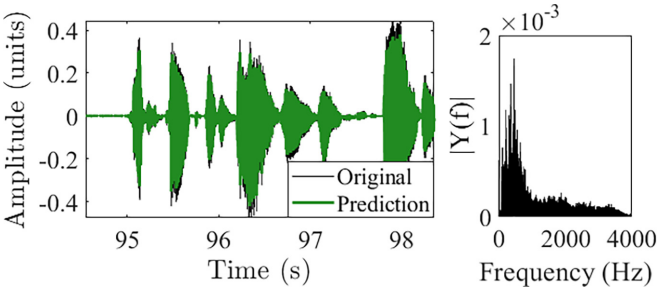

FIG. 2. AO PRC performs one-step-ahead speech prediction task (left), with the fast Fourier transform of the signal for reference (right). The RMSE for the test portion of this task was 0.00966, while the RMSE of the baseline (i.e., assuming a prediction equal to the average value of the signal) was 0.05838. Here, $\mu = 100$, $k = 100$, $a = 100$, and $N = 200$.

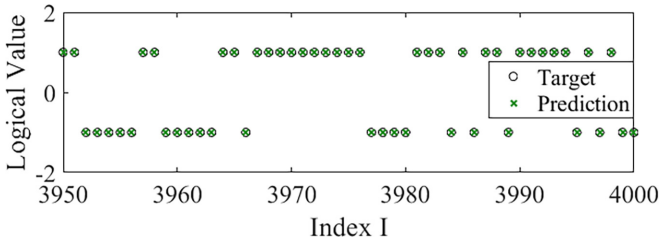

FIG. 4. The AO PRC calculates the exclusive OR task (XOR). The modified Shannon's information rate for the test portion of this task was equal to 1, while the IR of the baseline (i.e., assuming a prediction equal to the average value of the signal) was 0. Here, $\mu = 100$, $k = 100$, $a = 100$, and $N = 100$.

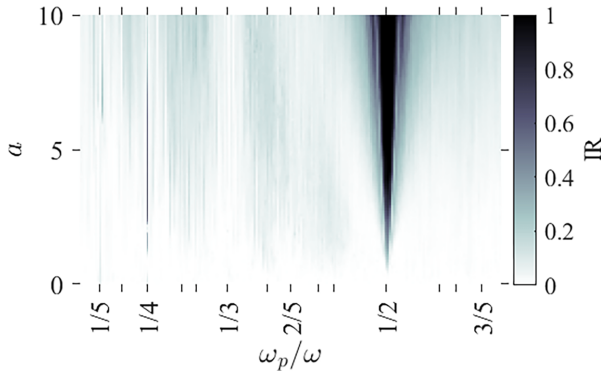

FIG. 5. Using an XOR task as a benchmark of the Hopf PRC, a computational Arnold tongue is shown at $\frac{\omega_p}{\omega} = \frac{1}{2}$. This computational Arnold tongue tapers down as a decreases. Thus, a small mistuning of the resonance value will require a larger amount of power as compensation. Here, $\mu = 3$, $\omega_p = 20\pi$, and $\Omega = 40\pi$.

sequence ratios. However, a very small mistuning (here, 1%) can cause a drastic degradation of the performance. Meanwhile, the initial condition for the ω state of the AO was set to a 5% error for the starting value. However, unlike the case of 1% error for the Hopf PRC, the AO PRC learns the correct frequency, and the AO PRC has perfect performance for several Farey sequence numbers. This points to the practical advantage of leveraging self-learning to provide robustness to physical reservoir computing.

To highlight the AO PRC's self-learning ability, it was compared with the Hopf PRC in Fig. 7. For this comparison, the pseudofrequency was varied over an arbitrary range of frequencies, and Ω was set equal to $2\omega_p$ to match the resonance condition shown in Fig. 6. Both ω_0 (for the Hopf oscillator) and ω (for the adaptive oscillator) were set equal to 0.80Ω, which is a 20% error from a resonance condition. The Hopf oscillator has a narrow region of good performance.

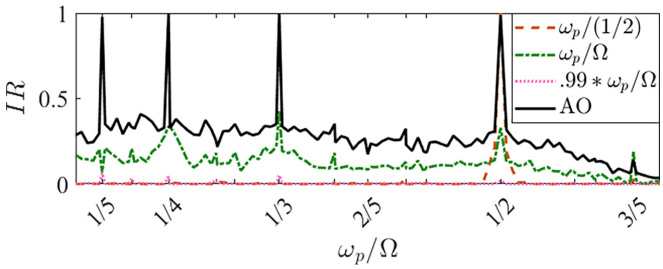

FIG. 6. The effects of the static resonance, ω_0, on the computational ability of the Hopf PRC are compared to the dynamic resonance, ω. When $\omega_0 = \frac{\omega_p}{\frac{1}{2}}$, the Hopf PRC has good performance when $\frac{\omega_p}{\Omega} = \frac{1}{2}$. When $\omega_0 = \frac{\omega_p}{\Omega}$, the overall performance is fairly low, but maximums coincide with several Farey sequence numbers. When $\omega_0 = .99\frac{\omega_p}{\Omega}$, this very small mistuning (here, 1%) causes a drastic degradation of the performance. Meanwhile, the initial condition for the ω state of the AO was set to $.95\frac{\omega_p}{\Omega}$, which is a 5% error for the starting value of ω. However, the AO learns the correct frequency, so it has high performance at several Farey sequence numbers. Here, $\mu = 5$, $a = 5$, $k = 25$, and $\omega_p = 20\pi$.

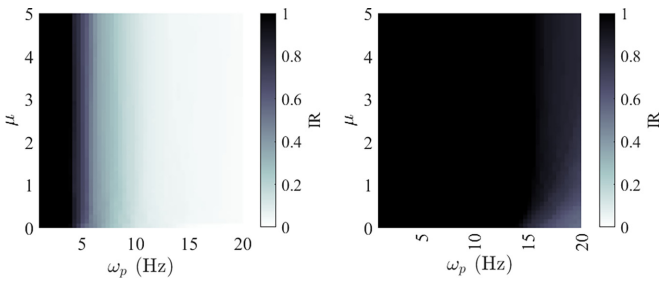

FIG. 7. The effect of self-learning of the AO PRC is highlighted. To show this, the pseudofrequency, ω_p, was varied, and the forcing frequency was set as $\Omega = 2\omega_p$ to match the resonance condition shown in previous figures. The initial condition for the ω state of the AO and the static resonance parameter, ω_0, of the Hopf oscillator were both set to 0.80Ω. The Hopf PRC (left) only works for small values of ω_p, while the AO PRC (right) works for a wide range of frequencies. Here, $a = 10$, $k = 10$, and $\Omega = 2\omega_p$.

The adaptive oscillator, on the other hand, is able to learn the correct resonance condition. This self-learning allows the AO PRC to function effectively.

V. FIELD-PROGRAMMABLE ANALOG ARRAY EXPERIMENT

In this section, an analog circuit is constructed on a field-programmable analog array (FPAA) to experimentally validate the AO PRC. FPAAs are the analog counterpart to field-programmable gate arrays, and they use a switched-capacitor technology [66]. Due to their reconfigurability [67], FPAAs have been used to study the Lorenz system [68,69], neuron models [70], chaotic oscillators [71–74], and four-state AO [50].

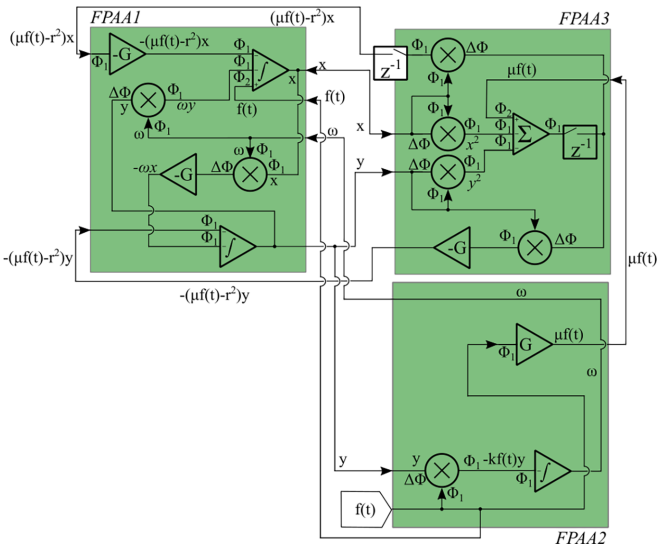

FIG. 8. The experimental field-programmable analog array circuit schematic is shown here. Three FPAA AN231E04 chips (Anadigm, Paso Robles, CA) were used, which are designated as FPAA1, FPAA2, and FPAA3. The external forcing function, $f(t)$, is sent to the FPAA from MATLAB via a National Instruments 9263 module.

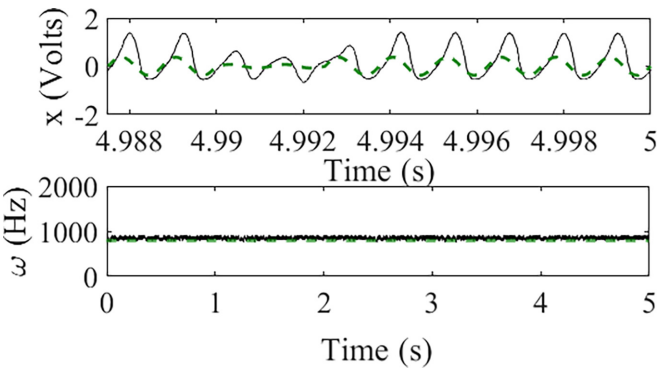

FIG. 9. An example of the response of the FPAA circuit. On the top, a portion of the time history of x (black line) is plotted with $f(t)$ (green dashed line). On the bottom, the ω state (black line) is plotted with the Ω value (green dashed) for reference. For this example, the information rate was calculated to be 0.99. Here, $\mu = 3$, $a = 0.15$, $k = 0.05$, $N = 100$, $\omega_p = 400$ Hz, and $\Omega = 800$ Hz.

In the circuit shown in Fig. 8, the limit cycle's radius is $r = \sqrt{x^2 + y^2}$. This circuit used three of the four AN231E04 chips on an Anadigm QuadApex board. The external signal, $f(t)$, was generated in MATLAB and sent to FPAA2 via a National Instruments 9263 module. The x, y, and ω states were collected back to MATLAB via a National Instruments 9201 module. In this circuit, several configurable analog modules (CAMs) were used. These included summation, multiplication, integration, sample-and-hold (represented by Z^{-1}), and DC voltage CAMs (represented by a circle with a $\pm$ symbol inside).

An example of a response of the AO PRC is shown in Fig. 9. For the XOR task, the circuit performed with a modified information rate of 0.99. The response for this set of parameters is representative over the operational range of the circuit.

In Fig. 10, the operational range of the FPAA circuit is explored for a set of parameter values. It should be noted that the operational range can be increased by optimizing the circuit design and implementing it as a printed circuit board, but that is beyond the scope of the current paper. The experimental AO PRC has near ideal performance from approximately 20 Hz to almost 490 Hz. This provides both robustness to mistuned

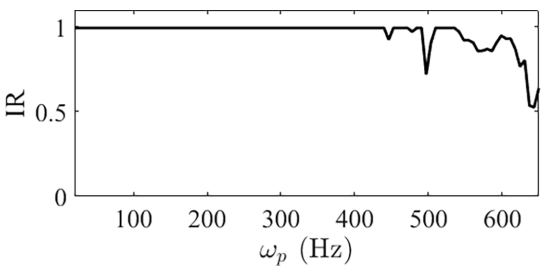

FIG. 10. The operational range of the FPAA circuit is shown here. As physical circuits have limiting constraints not present in the idealized model, the physical circuit has a reduced operational range. The FPAA circuit has near ideal performance over the range from approximately 20 Hz to 490 Hz. Here, $\mu = 3$, $a = 0.15$, $k = 0.05$, $N = 100$, $\omega_p = 400$ Hz, and $\Omega = 800$ Hz.

frequencies of the oscillator through self-learning. Moreover, the response of the PRC has degraded performance outside this region, as compared to the steep drop in performance for the Hopf PRC. In real-world scenarios, this would provide a further degree of robustness. For each of the values of ω_p, a time series of 1600 units was used for training and 400 units for testing.

VI. DISCUSSION

In this paper, an AO is proposed as the base of a PRC. By leveraging the AO's intrinsic ability to learn and store information in plastic states, this PRC exhibits self-learning. This provides robustness to frequency mistuning, which is a severe limitation for limit cycle-based PRCs. Self-learning allows the AO PRC to be used for real-time processing of various signals without modifying the system.

Recent work has shown that oscillator-based PRCs are quite powerful. Their vibratory nature makes them well-suited for audio recognition tasks. For instance, a Hopf-based PRC is capable of separating wake words, even without the assistance of a neural network [9]. As neural network-based reservoir computing is usually too computationally intensive for edge devices, the AO PRC could be used as a high performance computer for edge devices. Moreover, the PRC only utilizes a simple regression for training, which significantly reduces training costs.

The same AO can also be used as a morphable logic gate [57]. This allows the possibility for creating hardware that acts simultaneously as an AI inference processor and a general CPU. Highlighting the non-von-Neumann nature of this architecture, the AO PRC has the dual roles of storing information and providing computational power.

ACKNOWLEDGMENTS

M.R.E.U.S. and X.L. provided the handwritten digit recognition benchmark and FPAA expertise, respectively. E.P. invented and reduced to practice the AO PRC, ran simulations, completed analysis, designed the experiment, and conducted experiments.

APPENDIX A: NONEXISTENCE OF LIMIT CYCLES FOR ADAPTIVE OSCILLATORS

The classical Hopf oscillator is presented first for completeness:

$$\dot{x} = (\mu - (x^2 + y^2))x - \omega_0 y,$$
$$\dot{y} = (\mu - (x^2 + y^2))y + \omega_0 x. \tag{A1}$$

The Hopf oscillator is a limit cycle oscillator. When μ is positive, there is one unstable point, [0,0], while all other parameter combinations of $[x, y]$ will coalesce to a limit cycle. A limit cycle is an isolated periodic solution, which has no other periodic solutions close to it [75]. The Hopf oscillator shown in Eq. (A1) has one limit cycle. It should be noted that limit cycles require nonlinearity, and oscillators that exhibit limit cycles are often called *limit cycle oscillators*.

In Eq. (1), $f(t)$ is a normalized, time-varying external force that acts on the adaptive oscillator. This AO has a Hopf

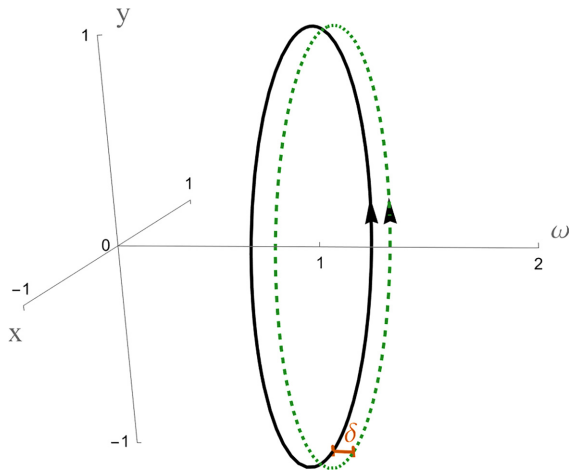

FIG. 11. For an adaptive oscillator, two periodic orbits are shown in the $x - y - \omega$ state space. They are separated by an amount, δ (depicted in orange), in the ω direction. By varying δ, the two orbits can be made arbitrarily close to each other, which shows that the AO does not have limit cycles.

oscillator as its base (e.g., x and y). In Eq. (A1), the Hopf oscillator has a constant resonance parameter, ω_0, while the AO in Eq. (1) has a dynamic frequency state, ω. When this AO is forced with a sinusoid, such as $\sin(\Omega t)$, the ω state will converge to Ω.

Interestingly, the AO shown in Eq. (1) is *not* a limit cycle oscillator. This can be easily seen by choosing two sets of initial conditions, $[x_0, y_0, \omega_0]$ and $[x_0, y_0, \omega_0 + \delta]$, where the first set of initial conditions are chosen to correspond to a closed periodic orbit and $\delta > 0$ is an arbitrary constant. For any $\epsilon > 0$, a δ can be chosen such that $\epsilon > \delta > 0$. This implies that the second set of initial conditions is arbitrarily close to the first set of initial conditions, so the AO cannot have isolated periodic solutions. The only difference in the two orbits described by these two sets of initial conditions is the resonance frequency, which is governed by the ω state. In the three-dimensional state space, $x - y - \omega$, they are shifted only an amount δ in the ω direction, as shown in Fig. 11. A local stability analysis of the Hopf-based adaptive frequency oscillator shows that the x and y states have a stable periodic solution that oscillates with a frequency of ω, but ω itself is neither stable nor unstable [48].

APPENDIX B: MODIFIED NYQUIST LIMIT FOR PRC

The PRC has poor performance at one-half the Nyquist frequency (e.g., one quarter of the sampling frequency). In Fig. 12, a sampling rate of 1000 samples per second was chosen, and sinusoids of various frequencies were used as the signal for the PRC. The minimal value occurs at half the Nyquist frequency (e.g., one-quarter of the sampling rate). Here, the ability is defined as $\frac{\text{abs}(\text{RMSE}-\text{RMSE}_{\text{baseline}})}{\text{RMSE}_{\text{baseline}}}$, where RMSE is the root mean squared error of the one-step-ahead prediction, and $\text{RMSE}_{\text{Baseline}}$ is the root mean squared error of the sinusoid itself. Thus, the sampling rate should be chosen to be sufficiently higher than the frequencies to be captured by the PRC.

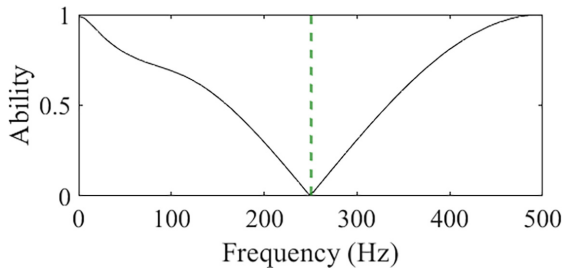

FIG. 12. For a one-step-ahead prediction of sinusoids of varying frequencies, the PRC works poorly at one-half the Nyquist frequency (green dashed line).

APPENDIX C: IMAGE PROCESSING BENCHMARK

Other than time-varying signals, the AO PRC is also capable of solving other tasks. The image processing capability of the PRC is tested with the MNIST handwritten digit data set. The images were first preprocessed by rescaling the pixel values to a range between 0 and 1. Subsequently, a downsampling process was applied by performing a maximum pooling operation twice, resulting in a 7×7 pixel image. This 7×7 image is flattened to a 1D vector and sent to the oscillator (i.e., the pixels of the image were sent to the oscillator sequentially). The response of the oscillator was then combined with a logistic regression using the SCIKIT-LEARN package from Python for classification of images [11,76]. Unlike the other tasks, this image recognition task only used ten virtual nodes for each pixel. The confusion matrix provided in Fig. 13 shows the reservoir's performance for digit recognition. A value of 1 corresponds to a perfect prediction, while each column sums to 1. A series of 8000 images was used for training and 2000 untrained images for testing.

APPENDIX D: COMPARISON WITH OTHER NETWORKS

There are different types of recurrent neural networks (RNNs), and it is important to compare their performance

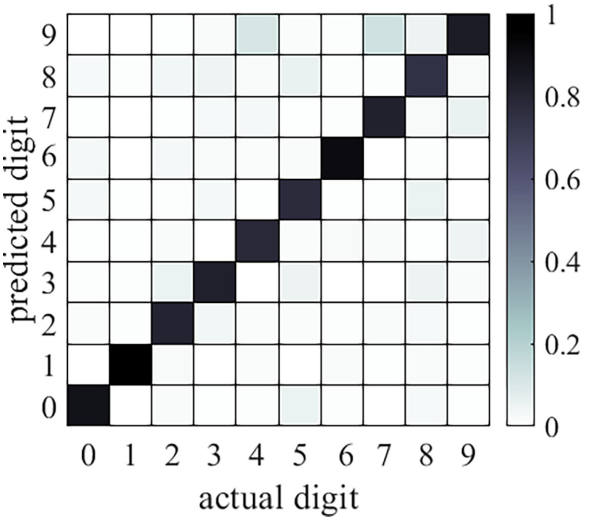

FIG. 13. The AO PRC is used for handwritten digit recognition. The resulting confusion matrix is shown here.

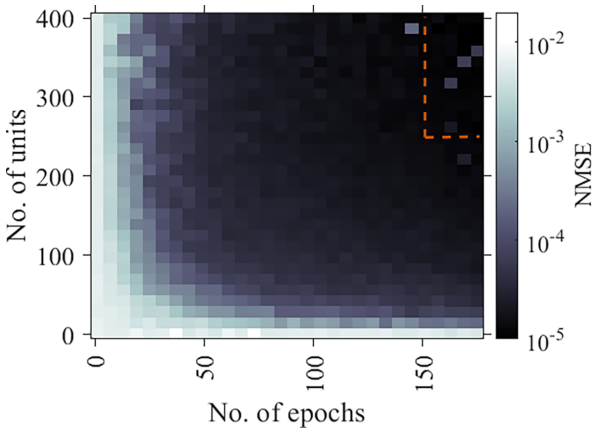

FIG. 14. Normalized mean squared error of the LSTM network with various numbers of hidden units and training epochs. As the number of units and epochs increases, the LSTM gains better performance. The orange dashed lines correspond to the minimal number of units (249 units) and epochs (151 epochs) that resulted in an error less than that of the AO PRC's NMSE of 2.4435×10^{-5}.

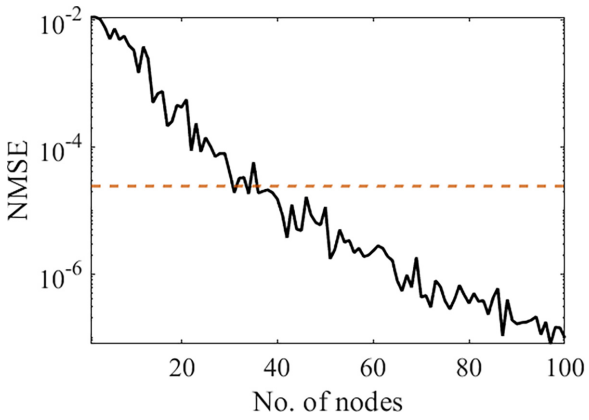

FIG. 15. Normalized mean squared error of the leaky integrator ESN with various numbers of fully connected nodes, which is trained with a ridge regression. As the number of nodes increases, the ESN gains better performance. The orange dashed lines correspond to the minimal number of nodes (31 nodes) that resulted in an error less than that of the AO PRC's NMSE of 2.4435×10^{-5}.

with their complexity [77,78]. Here, the AO PRC is compared with a long-short term memory (LSTM) neural network and a fully connected leaky integrator echo state network (LESN), which are two popular RNNs. A tenth-order nonlinear autoregressive moving average (NARMA) task was used as the benchmark [37,79]. For this NARMA task, the AO PRC's normalized mean square error (NMSE) was 2.4435×10^{-5}.

MATLAB'S DEEP LEARNING TOOLBOX was used to construct and train the LSTM. The LSTM is constructed as a sequence input layer, m LSTM layers, and a fully connected layer. As the number of hidden units and training epochs is increased, the LSTM has improved performance. The minimal number of units and epochs necessary for the LSTM to meet the AO PRC's error was 249 units and 151 epochs, which is shown in Fig. 14.

The leaky integrator echo state network is described in detail in Ref. [79]. The ESN has n nodes, which are fully connected. The random weights of the connections between these n nodes is left untrained, while only the output layer is trained with a ridge regression (in the same manner used for the AO PRC). The minimal number of nodes necessary for the ESN to meet the AO PRC's error was 31 nodes, which is shown in Fig. 15.

The main advantage of the AO PRC is that it is an analog system that can replace a relatively large portion of an ESN or LSTM. As each node or unit has some complexity, these must inherently take up physical space. Interconnects alone become cumbersome in hardware (for instance, the 31 nodes of the fully connected ESN has over 900 interconnects if constructed in hardware). In addition to performing as an analog network, it can also be used as a morphable logic gate [57]. However, whereas other networks have standardized training procedures, the AO PRC does not yet have a standardized approach, while only the readout layer is trained. Moreover, it is unclear whether the PRC would perform equally well for all tasks, and so more comprehensive benchmarking is required.

[1] K. Nakajima and I. Fischer, *Reservoir computing*, edited by K. Nakajima and I. Fischer (Springer Nature, Singapore, 2021).
[2] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).
[3] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comp. Sci. Rev. **3**, 127 (2009).
[4] B. Penkovsky, X. Porte, M. Jacquot, L. Larger, and D. Brunner, Coupled nonlinear delay systems as deep convolutional neural networks, Phys. Rev. Lett. **123**, 054101 (2019).
[5] N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, Reservoir computing with a single time-delay autonomous boolean node, Phys. Rev. E **91**, 020801(R) (2015).
[6] 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).
[7] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note, Bonn, Germany: German National Research Center for Information Technology, GMD Technical Report **148**, 13 (2001).
[8] T. Natschläger, W. Maass, and H. Markram, The "liquid computer": A novel strategy for real-time computing on time series, TELEMATIK **8**, 39 (2002).
[9] M. R. E. U. Shougat, X. Li, S. Shao, K. McGarvey, and E. Perkins, Hopf physical reservoir computer for reconfigurable sound recognition, Sci. Rep. **13**, 8719 (2023).
[10] R. Martinenghi, S. Rybalko, M. Jacquot, Y. K. Chembo, and L. Larger, Photonic nonlinear transient computing with multiple-delay wavelength dynamics, Phys. Rev. Lett. **108**, 244101 (2012).

[11] 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).
[12] S. Borlenghi, M. Boman, and A. Delin, Modeling reservoir computing with the discrete nonlinear Schrödinger equation, Phys. Rev. E **98**, 052101 (2018).
[13] P. Antonik, N. Marsal, D. Brunner, and D. Rontani, Human action recognition with a large-scale brain-inspired photonic computer, Nat Mach Intell **1**, 530 (2019).
[14] M. Inubushi and K. Yoshimura, Reservoir computing beyond memory-nonlinearity trade-off, Sci. Rep. **7**, 10199 (2017).
[15] R. Wang, E. Kalnay, and B. Balachandran, Neural machine-based forecasting of chaotic dynamics, Nonlinear Dyn. **98**, 2903 (2019).
[16] M. Rafayelyan, J. Dong, Y. Tan, F. Krzakala, and S. Gigan, Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction, Phys. Rev. X **10**, 041037 (2020).
[17] A. Röhm, D. J. Gauthier, and I. Fischer, Model-free inference of unseen attractors: Reconstructing phase space features from a single noisy trajectory using reservoir computing, Chaos **31**, 103127 (2021).
[18] 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).
[19] K. Srinivasan, N. Coble, J. Hamlin, T. Antonsen, E. Ott, and M. Girvan, Parallel machine learning for forecasting the dynamics of complex networks, Phys. Rev. Lett. **128**, 164101 (2022).
[20] M. R. E. U. Shougat, X. Li, T. Mollik, and E. Perkins, An information theoretic study of a duffing oscillator array reservoir computer, J. Comput. Nonlinear Dyn. **16**, 081004 (2021).
[21] G. Marcucci, D. Pierangeli, and C. Conti, Theory of neuromorphic computing by waves: machine learning by rogue waves, dispersive shocks, and solitons, Phys. Rev. Lett. **125**, 093901 (2020).
[22] G. Dion, S. Mejaouri, and J. Sylvestre, Reservoir computing with a single delay-coupled non-linear mechanical oscillator, J. Appl. Phys. **124**, 152132 (2018).
[23] F. Laporte, J. Dambre, and P. Bienstman, Simulating self-learning in photorefractive optical reservoir computers, Sci. Rep. **11**, 2701 (2021).
[24] M. R. E. U. Shougat and E. Perkins, The van der Pol physical reservoir computer, Neuromorphic Comput. Eng. **3**, 024004 (2023).
[25] G. Urbain, J. Degrave, B. Carette, J. Dambre, and F. Wyffels, Morphological properties of mass–spring networks for optimal locomotion learning, Front. Neurorobot. **11**, 16 (2017).
[26] 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).
[27] 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).
[28] 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).
[29] M. R. E. U. Shougat, S. Kennedy, and E. Perkins, A self-sensing shape memory alloy actuator physical reservoir computer, IEEE Sens. Lett. **7**, 1 (2023).
[30] 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).
[31] 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).
[32] J. C. Coulombe, M. C. York, and J. Sylvestre, Computing with networks of nonlinear mechanical oscillators, PLoS ONE **12**, e0178663 (2017).
[33] K. Fujii and K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning, Phys. Rev. Appl. **8**, 024030 (2017).
[34] L. C. G. Govia, G. J. Ribeill, G. E. Rowlands, H. K. Krovi, and T. A. Ohki, Quantum reservoir computing with a single nonlinear oscillator, Phys. Rev. Res. **3**, 013077 (2021).
[35] J. Chen, H. I. Nurdin, and N. Yamamoto, Temporal information processing on noisy quantum computers, Phys. Rev. Appl. **14**, 024065 (2020).
[36] M. R. E. U. Shougat, X. Li, and E. Perkins, Dynamic effects on reservoir computing with a Hopf oscillator, Phys. Rev. E **105**, 044212 (2022).
[37] M. R. E. U. Shougat, X. Li, T. Mollik, and E. Perkins, A Hopf physical reservoir computer, Sci. Rep. **11**, 19465 (2021).
[38] J. Cristiano, D. Puig, and M. Garcia, Efficient locomotion control of biped robots on unknown sloped surfaces with central pattern generators, Electron. Lett. **51**, 220 (2015).
[39] L. Righetti, J. Buchli, and A. J. Ijspeert, From dynamic hebbian learning for oscillators to adaptive central pattern generators, in *Proceedings of 3rd International Symposium on Adaptive Motion in Animals and Machines–AMAM 2005* (Verlag ISLE, Ilmenau, 2005).
[40] L. Righetti and A. J. Ijspeert, Programmable central pattern generators: an application to biped locomotion control, in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006* (IEEE, Orlando, Florida, 2006), pp. 1585–1590.
[41] J. Buchli and A. J. Ijspeert, Self-organized adaptive legged locomotion in a compliant quadruped robot, Auton. Rob. **25**, 331 (2008).
[42] X. Xiong, F. Wörgötter, and P. Manoonpong, Adaptive and energy efficient walking in a hexapod robot under neuromechanical control and sensorimotor learning, IEEE Trans. Cybern. **46**, 2521 (2015).
[43] M. Thor and P. Manoonpong, A fast online frequency adaptation mechanism for cpg-based robot motion control, IEEE Rob. Automation Lett. **4**, 3324 (2019).
[44] C. Liu, D. Wang, and Q. Chen, Central pattern generator inspired control for adaptive walking of biped robots, IEEE Trans. Syst. Man, Cyber.: Syst. **43**, 1206 (2013).
[45] H. Choi, K. Seo, S. Hyung, Y. Shim, and S.-C. Lim, Compact hip-force sensor for a gait-assistance exoskeleton system, Sensors **18**, 566 (2018).
[46] K. Maharatna, A. Ahmadi, and E. Magieri, Biologically inspired analogue signal processing: Some results towards developing next generation signal analyzers, in *Proceedings of*

the 2009 12th International Symposium on Integrated Circuits (IEEE, Singapore, 2009), pp. 542–545.

[47] J. Buchli, L. Righetti, and A. J. Ijspeert, Frequency analysis with coupled nonlinear oscillators, Physica D **237**, 1705 (2008).

[48] X. Li, M. R. E. U. Shougat, S. Kennedy, C. Fendley, R. N. Dean, A. N. Beal, and E. Perkins, A four-state adaptive Hopf oscillator, PLoS ONE **16**, e0249131 (2021).

[49] X. Li, M. R. E. U. Shougat, T. Mollik, A. N. Beal, R. N. Dean, and E. Perkins, Stochastic effects on a Hopf adaptive frequency oscillator, J. Appl. Phys. **129**, 224901 (2021).

[50] X. Li, M. R. E. U. Shougat, T. Mollik, R. N. Dean, A. N. Beal, and E. Perkins, Field-programmable analog array (FPAA) based four-state adaptive oscillator for analog frequency analysis, Rev. Sci. Instrum. **94**, 035103 (2023).

[51] X. Li, P. Kallepalli, T. Mollik, M. R. E. U. Shougat, S. Kennedy, S. Frabitore, and E. Perkins, The pendulum adaptive frequency oscillator, Mechanical Systems and Signal Processing **179**, 109361 (2022).

[52] X. Li, A. Beal, D. Robert, and E. Perkins, Chaos in a pendulum adaptive frequency oscillator circuit experiment, Chaos Theory Appl. **5**, 11 (2023).

[53] C. Frenkel, D. Bol, and G. Indiveri, Bottom-up and top-down neural processing systems design: Neuromorphic intelligence as the convergence of natural and artificial intelligence, Proc. IEEE **111**, 623 (2023).

[54] S.-B. Shim, M. Imboden, and P. Mohanty, Synchronized oscillation in coupled nanomechanical oscillators, Science **316**, 95 (2007).

[55] S. Coombes and P. C. Bressloff, Mode locking and Arnold tongues in integrate-and-fire neural oscillators, Phys. Rev. E **60**, 2086 (1999).

[56] V. Lopez-Pastor and F. Marquardt, Self-learning machines based on Hamiltonian echo backpropagation, Phys. Rev. X **13**, 031020 (2023).

[57] M. R. E. U. Shougat, X. Li, and E. Perkins, Multiplex-free physical reservoir computing with an adaptive oscillator, Phys. Rev. E **109**, 024203 (2024).

[58] L. Righetti, J. Buchli, and A. J. Ijspeert, Dynamic Hebbian learning in adaptive frequency oscillators, Physica D **216**, 269 (2006).

[59] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, Hands-on reservoir computing: a tutorial for practical implementation, Neuromorphic Comput. Eng. **2**, 032002 (2022).

[60] A. M. Al-Kaisey, A. N. Koshy, F. J. Ha, R. Spencer, L. Toner, J. K. Sajeev, A. W. Teh, O. Farouque, and H. S. Lim, Accuracy of wrist-worn heart rate monitors for rate control assessment in atrial fibrillation, Int. J. Cardiol. **300**, 161 (2020).

[61] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments* (Springer Science & Business Media, New York, 2012).

[62] A. Nemcova, R. Smisek, K. Opravilová, M. Vitek, L. Smital, and L. Maršánová, Brno University of Technology ECG Quality Database (BUT QDB) PhysioNet (2020), doi: 10.13026/qcnm-a155.

[63] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals, Circulation **101**, e215 (2000).

[64] Z. Jackson, C. Souza, J. Flaks, Y. Pan, N. Hereman, and A. Thite, Jakobovski free-spoken-digit-dataset: v1.0.8, Zenodo (2018), doi: 10.5281/zenodo.1342401.

[65] C. E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. **27**, 379 (1948).

[66] H. Kutuk and S.-M. Kang, A field-programmable analog array (FPAA) using switched-capacitor techniques, in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96* (IEEE, Atlanta, Georgia, 1996), Vol. 4, pp. 41–44.

[67] R. Kilic and F. Y. Dalkiran, Reconfigurable implementations of chua's circuit, Inter. J. Bifurcation Chaos **19**, 1339 (2009).

[68] E. Tlelo-Cuautle, A. D. Pano-Azucena, O. Guillén-Fernández, and A. Silva-Juárez, *Analog/Digital Implementation of Fractional Order Chaotic Circuits and Applications* (Springer, Cham, Switzerland, 2020).

[69] E. Günay and K. Altun, Lorenz-like system design using cellular neural networks, Turk. J. Electr. Eng. Comp. Sci. **26**, 1812 (2018).

[70] N. Dahasert, İ. Öztürk, and R. Kiliç, Experimental realizations of the HR neuron model with programmable hardware and synchronization applications, Nonlinear Dyn. **70**, 2343 (2012).

[71] C. Li, W. J.-C. Thio, J. C. Sprott, H. H.-C. Iu, and Y. Xu, Constructing infinitely many attractors in a programmable chaotic circuit, IEEE Access **6**, 29003 (2018).

[72] S. Çiçek, FPAA based desıgn and implementation of Sprott N chaotic system, in *International Scientific and Vocational Studies Congress* (BILMES 2019 Ankara, Ankara, Turkey, 2019), pp. 476–482.

[73] F. Y. Dalkiran and J. C. Sprott, Simple chaotic hyperjerk system, Int. J. Bifurcation Chaos **26**, 1650189 (2016).

[74] A. Silva-Juárez, E. Tlelo-Cuautle, L. G. de la Fraga, and R. Li, FPAA-based implementation of fractional-order chaotic oscillators using first-order active filter blocks, J. Adv. Res. **25**, 77 (2020).

[75] A. H. Nayfeh and B. Balachandran, *Applied Nonlinear Dynamics: Analytical, Computational, and Experimental Methods* (John Wiley & Sons, Weinheim, Germany, 2008).

[76] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (O'Reilly Media, Inc., Sebastopol, California, 2022).

[77] W. Sun, N. Akashi, Y. Kuniyoshi, and K. Nakajima, Physics-informed recurrent neural networks for soft pneumatic actuators, IEEE Robot. Autom. Lett. **7**, 6862 (2022).

[78] 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).

[79] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, Information processing via physical soft body, Sci. Rep. **5**, 10487 (2015).