



Enhancing the performance of quantum reservoir computing and solving the time-complexity problem by artificial memory restriction

Saud Čindrak ^{*}*Technische Universität Ilmenau, Institute of Physics, 98693 Ilmenau, Germany*Brecht Donvil *Institute for Complex Quantum Systems and IQST, Ulm University, Albert-Einstein-Allee 11, D-89069 Ulm, Germany*Kathy Lüdge  and Lina Jaurigue [†]*Technische Universität Ilmenau, Institute of Physics, 98693 Ilmenau, Germany*

(Received 17 May 2023; revised 21 November 2023; accepted 29 November 2023; published 16 January 2024)

We propose a scheme that can enhance the performance and reduce the computational cost of quantum reservoir computing. Quantum reservoir computing is a computing approach which aims at utilizing the complexity and high dimensionality of small quantum systems, together with the fast trainability of reservoir computing, in order to solve complex tasks. The suitability of quantum reservoir computing for solving temporal tasks is hindered by the collapse of the quantum system when measurements are made. This leads to the erasure of the memory of the reservoir. Hence, for every output, the entire input signal is needed to reinitialize the reservoir, leading to quadratic time complexity. Another critical issue for the hardware implementation of quantum reservoir computing is the need for an experimentally accessible means of tuning the nonlinearity of the quantum reservoir. We present an approach which addresses both of these issues. We propose artificially restricting the memory of the quantum reservoir by only using a small number inputs to reinitialize the reservoir after measurements are performed. This strongly influences the nonlinearity of the reservoir response due to the influence of the initial reservoir state, while also substantially reducing the number of quantum operations needed to perform time-series prediction tasks due to the linear rather than quadratic time complexity. The reinitialization length therefore provides an experimental accessible means of tuning the nonlinearity of the response of the reservoir, which can lead to significant task-specific performance improvement. We numerically study the linear and quadratic algorithms for a fully connected transverse Ising model and a quantum processor model.

DOI: [10.1103/PhysRevResearch.6.013051](https://doi.org/10.1103/PhysRevResearch.6.013051)

I. INTRODUCTION

The field of quantum computation promises a significant computational speedup over classical computation for certain sets of problems [1]. Machine learning is one such field where it is known that quantum computers can offer an advantage [2]. Several machine learning tasks have been experimentally realized on quantum systems—some examples are Refs. [3–6]—but broad applicability of machine learning on quantum devices is still hindered by the limitations of current quantum processing devices. One of these limitations is the inevitable noise these devices experience. For reservoir computing, a subfield of machine learning, this noise does not pose a hindrance and could even be a resource [7–9].

Reservoir computing is a machine learning approach wherein only the output layer is trained [10–13]. Due to

this simple training scheme it is well suited for hardware implementation, meaning that an input signal is fed into a physical system and the dynamics of that physical “reservoir” are utilized to project the data into a high-dimensional latent space. The responses of the reservoir are then sent through a readout layer, which is trained in order to approximate the desired function.

There are two main avenues of research into quantum reservoirs, either quantum systems whose dynamics are generated by a Hamiltonian H or quantum circuits consisting of several qubits on which unitary operations can be performed. For the former, several studies have been dedicated to the Ising model [14–18], showing its viability as a reservoir for several benchmark tasks. The authors of Refs. [7,8,19] devised schemes for reservoir computing on a quantum circuit and implemented them on IBM quantum processors. A promising avenue of use for quantum reservoir computing is to aid in the measurement of quantum states [20,21].

There are still several hurdles to be overcome for the efficient hardware implementation of quantum reservoir computing (QRC). In this manuscript we address two of these hurdles: an experimentally accessible means of tuning the nonlinear response of the reservoir and a reduction of the number of quantum operations needed for temporal tasks.

^{*}saud.cindrak@tu-ilmenau.de[†]lina.jaurigue@tu-ilmenau.de

The output of both types of quantum reservoirs mentioned above are typically time series of one- or two-qubit observables. For each output, quantum measurements have to be performed, which poses a significant problem for the physical implementation of quantum reservoir computing [22]. With each measurement, the quantum system state collapses and all information about the input signal is lost. Therefore, for each time step of the output, the entire signal up to that point is needed to reinitialize the reservoir. This procedure leads to a time complexity quadratic in the length of the input signal. The authors of Ref. [14] propose as a solution to perform reservoir computing with nuclear-magnetic-resonance spin ensemble systems [23,24]. These large ensembles have the advantage that all copies of the ensemble can be simultaneously controlled such that they all follow the same dynamics. In this way expectation values can be measured with barely any backaction. The authors of Ref. [25] investigate the influence of weak measurements and additionally make the observation that due to the fading memory of the reservoir it is not necessary to reset the reservoir using the entire sequence of previous inputs. The latter was also previously investigated in Ref. [7] and can be understood as follows. Typically, information is encoded in the system state of some elements and then the closed system is evolved in time. Due to the successive overwriting of elements of the quantum reservoir, memory of past inputs is gradually lost. This means that the response of the reservoir is independent of inputs from the distant past and only a finite number of past inputs are needed to reinitialize the reservoir after each measurement.

In this paper we study the influence of restricting the number of reinitialization inputs after the reservoir is reset by a measurement. Not only does this reduce the time complexity of the reservoir computing algorithm [25], it also provides a means of experimentally tuning the nonlinearity of the quantum reservoir response, which addresses the need for task-dependent hyperparameter optimization. Task-dependent hyperparameter optimization is not an issue specific to QRC, but a general issue for reservoir computing, particularly for hardware-implemented reservoir computing where the accessible hyperparameters can be restricted and difficult or cumbersome to tune.

We demonstrate our approach on two simulated quantum reservoirs: a transverse field Ising Hamiltonian and a quantum circuit. In both cases we analyze their performance on the information processing capacity [26] and the Lorenz chaotic attractor [27]. We show that our proposed algorithm leads to improved performance for these tasks and addresses the problem of quadratically increasing reinitialization sequences for time-series tasks.

II. RESERVOIR COMPUTING

A reservoir computer can be understood as a recurrent neural network, where only the output weights \mathbf{W}^{out} are trained. The weighted connections between the internal nodes are either chosen randomly or are determined by the physical properties of the reservoir. To perform time-series prediction tasks using reservoir computing, an input sequence must be fed into the reservoir and then the responses to these inputs are sampled. The desired output is then approximated

by constructing a weighted sum of the sampled responses. The training procedure to determine the output weights is as follows. Assume an input series $\mathbf{u} = [u(t_1), u(t_2), \dots]$ and N_S readout nodes. At each time t_i the i th input u_i is introduced into the system and the output $[s_1(t_i), \dots, s_{N_S}(t_i)]$ is obtained. The target at each t_i is $y^{\text{targ}}(t_i)$ and the estimate by the reservoir is obtained by taking a linear combination of the output signal weighed by the vector \mathbf{W}^{out} ,

$$y(t_i) = \sum_j W_j^{\text{out}} s_j(t_i). \quad (1)$$

The optimal weights are obtained by

$$\begin{aligned} \mathbf{W}_{\text{opt}}^{\text{out}} &= \arg \min_{\mathbf{W}^{\text{out}}} \sum_i [y^{\text{targ}}(t_i) - y(t_i)]^2 \\ &= (\mathbf{S}^T \mathbf{S} + \lambda \mathbf{I})^{-1} \mathbf{S}^T \mathbf{Y}^{\text{targ}}, \end{aligned} \quad (2)$$

where $Y_{i,j}^{\text{targ}} = y_j^{\text{targ}}(t_i)$ are the elements of the target vector \mathbf{Y}^{targ} and $S_{i,j} = s_j(t_i)$ are the elements of the state matrix \mathbf{S} , where $j \in \{1, 2, \dots, N_S\}$ is the readout node index.

Time multiplexing

In order to increase the number of sampled reservoir responses, it can be beneficial to time multiplex. This means that each input u_i in the input sequence \mathbf{u} is fed into the reservoir for a duration T and during this time the response of each of the N_S readout nodes is sampled at N_V different points in time. The total number of sampled reservoir responses per input is then $N_S \times N_V$ and the number of columns in the state matrix is increased accordingly. This reservoir computing scheme is depicted in Fig. 1. Due to the nature of the time evolution of the quantum reservoirs, in contrast to common practice when using classical reservoirs [28], no input mask is used.

III. QUANTUM MEASUREMENT FOR TIME-SERIES PREDICTIONS

A. Quadratic scheme

In quantum reservoir computing the system state collapses when the reservoir response is measured. The usual procedure to experimentally implement time-series tasks in quantum reservoirs involves reinitializing the reservoir with the entire input history before the next reservoir response is measured. This procedure is depicted in Algorithm 1. The inputs to this algorithm include a unitary operation U , the set of observables to be measured $\{O_o\}_o$, the initial state $|\Psi_I\rangle$, an input series $\mathbf{u} = \{u_i\}_{i=1}^M$ of length M , and a scheme which encodes the input into a state $|\Psi_E(u_i)\rangle$. For each time step i , the system is initialized to $|\Psi_I\rangle$ and the signal up until u_i is fed into the reservoir. Afterwards, the measurement of the set of observables $\{O_o\}_o$ is performed. Since the i th. input requires i unitary operations, the complexity of this algorithm for a time series of length M is determined by

$$T_1(M) = \sum_{i=1}^M i = \frac{M(M+1)}{2} \in O(M^2). \quad (4)$$

For large or continuous time series ($M \rightarrow \infty$), this approach therefore becomes unfeasible. Current literature on quantum

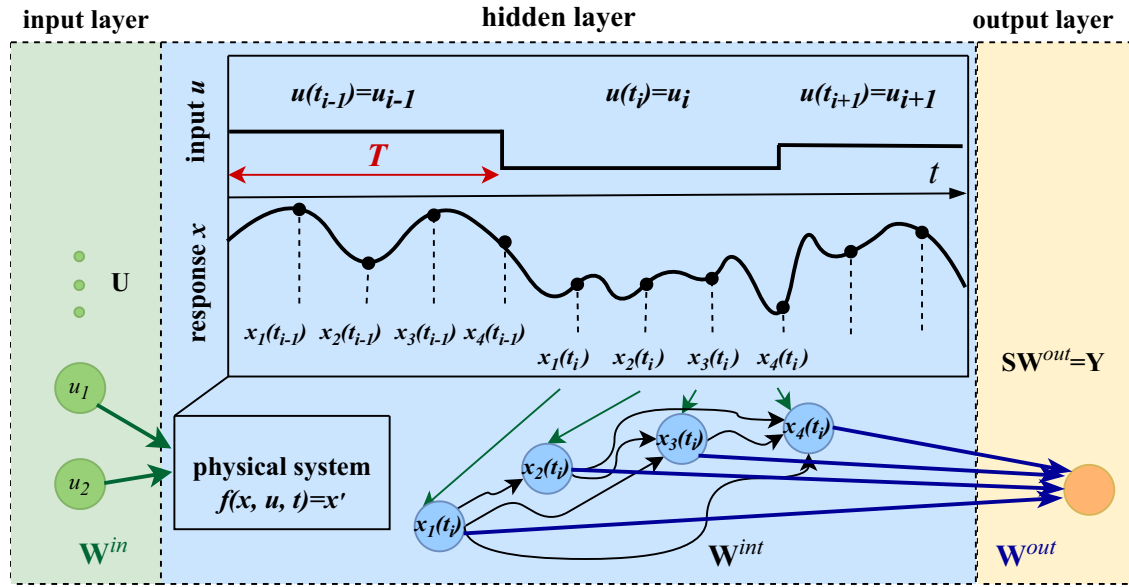


FIG. 1. Sketch of a time-multiplexed reservoir computer scheme. The inputs u_i are sequentially fed into the reservoir and the corresponding responses of the reservoir $x_j(t_u)$ are sampled in time, where $j \in \{1, \dots, N_V\}$ and N_V is the number of sampled responses for each input u_i . Due to the internal dynamics of the reservoir the $x_j(t_i)$ are coupled forward in time. The coupling matrix \mathbf{W}^{int} depends on the properties of the reservoir. The output Y_i is obtained by multiplying the vector of sampled reservoir responses with the vector of output weights \mathbf{W}^{out} , which is determined in the training phase.

reservoir computing for time-series tasks analyzes the properties of the reservoir using this scheme with quadratic time complexity [7,8,14,15,17].

B. Linear scheme

We propose an alternative scheme for reservoir computing. The scheme is based on the fading memory property usually assumed for reservoir computers [10,11], and the memory-nonlinearity trade-off which is known to occur in reservoir computing [29–31]. Qualitatively the fading memory property states that the reservoir forgets inputs far into the past. It can be characterized by the linear contribution to the information processing capacity IPC_1 . The information processing capacity is a generalization of the linear memory capacity [26]

which quantifies the ability of the reservoir to construct non-linear transforms of all possible combinations of past inputs into the reservoir (see Appendix D for details) and can be used to predict the performance on certain tasks [28,32].

The scheme we propose here is to only insert the previous $m = n - 1$ inputs to initialize the system before for each input u_i , rather than the last $i - 1$ inputs, where n can be much shorter than the fading memory of the reservoir. This results in a total of n unitary operation per input u_i and leads to a total of

$$T_2(M) = nM \in O(M) \tag{5}$$

unitary operations. This approach is computationally feasible for continuous or large time series ($M \rightarrow \infty$), as at any given input u_i only $n + 1$ unitary operations are required and thus can be computed in real time. The difference between our approach and that of Ref. [25] is that we consider much smaller n , which leads to a fundamentally different input response of the reservoir, as we will show in the subsequent sections. Our linear algorithm is depicted in Algorithm 2.

Algorithm 1. Quadratic complexity quantum algorithm (QCQA).

```

Require:  $U, |\Psi_I\rangle, |\Psi_E(u_m)\rangle, \{O_o\}_o$ 
Require:  $\mathbf{u} \leftarrow [u_1, u_2, \dots, u_M]$  ▷ Input series
1:  $i \leftarrow 1$ 
2: while  $i \leq M$  do
3:    $\rho(0) \leftarrow |\Psi_I\rangle\langle\Psi_I|$ 
4:    $j \leftarrow 1$ 
5:   while  $j \leq i$  do
6:      $\tilde{\rho}(j) = |\Psi_E(u_j)\rangle\langle\Psi_E(u_j)| \otimes \text{Tr}_1[\rho(j-1)]$ 
7:      $\rho(j) = U\tilde{\rho}(j-1)U^\dagger$ 
8:      $j \leftarrow j + 1$ 
9:   end while
10:   $S_{m,o} \leftarrow \text{Tr}[\rho(t_m)O_o]$  ▷ collapse of state
11:   $i \leftarrow i + 1$ 
12: end while
13: return  $S$ 

```

Algorithm 2. Linear complexity quantum algorithm (LCQA).

```

 $\rho(0) \leftarrow |\Psi_I\rangle\langle\Psi_I|$ 
 $k \leftarrow 0$ 
while  $k \leq n$  do
   $j \leftarrow i - n + k$ 
   $k \leftarrow k + 1$ 
   $\tilde{\rho}(j) = |\Psi_E(u_j)\rangle\langle\Psi_E(u_j)| \otimes \text{Tr}_1(\rho(j-1))$ 
   $\rho(j) = U\tilde{\rho}(j)U^\dagger$ 
end while

```

C. Input encoding

In the following section we compare the quadratic (QCQA) and the linear complexity quantum algorithms (LCQA) as a function of the reset length n . We study two different systems: an Ising model and a quantum circuit. For both systems, we encode our input series in the first qubit $|\Psi_I\rangle$ and set the initial state of the reservoir $|\Psi_I\rangle$ and the encoding state $|\Psi_E(u_i)\rangle$ depending on an input u_i to

$$|\Psi_I\rangle = |0000\rangle \quad (6)$$

$$|\Psi_E(u_i)\rangle = \sqrt{\frac{1-u_i}{2}}|0\rangle + \sqrt{\frac{1+u_i}{2}}|1\rangle. \quad (7)$$

IV. ISING MODEL

The dynamics of the fully connected transverse field Ising model are described by the Hamiltonian

$$H = \sum_{i=1, j>i}^{N_S} J_{ij} X_i X_j + \sum_{i=1}^{N_S} h Z_i. \quad (8)$$

X_i , Y_i and Z_i are the Pauli matrices of the i th. particle and are given by

$$X_i, Y_i, Z_i = \left(\bigotimes_{k=1}^{i-1} I_2 \right) \otimes \sigma_{x,y,z} \otimes \left(\bigotimes_{k=i+1}^{N_S} I_2 \right), \quad (9)$$

where $\sigma_{x,y,z}$ are the one-qubit Pauli matrices and I_2 the one-qubit identity operator. The J_{ij} are the coupling strengths between two particles in the x direction and are sampled from a uniform distribution on the interval $[0.25, 0.75]$, while $h = 0.5$ is the coupling strength to an external magnetic field in the z direction.

We numerically study the Ising model with four qubits $N_S = 4$. The outputs of the reservoir are the expectation values $\{\langle Z_i \rangle\}_{i=1}^4$, where Z_i is the z Pauli matrix for the i th. qubit. Additionally, we perform time multiplexing: each input signal is fed into the reservoir for an evolution time T , during which we perform $N_V = 30$ measurements. This leads to a total of $4 \times 30 = 120$ observables or readout nodes for each input step i . The unitary time evolution operation is given by

$$U = \exp -iHT, \quad (10)$$

with an evolution time (clock cycle) of $T = 20$.

For the Ising model, the linear memory capacity as a function of the steps into the past is shown in Fig. 2(a). Using the entire history to reset the system, i.e., the QCQA, the recall ability of this Ising model reservoir fades to zero after approximately 15 steps into the past (black line), meaning that a reset length of $n = 15$ should be sufficient to emulate the QCQA. To demonstrate the influence of the reset length in a general and task-independent manner, we calculate the information processing capacities as a function of n . Figures 2(b) and 2(c) show the summed information processing capacities (IPCs) of polynomial order one to six (IPC_1 – IPC_6) and Fig. 2(d) shows the total IPC summed over all polynomial orders. In each case the QCQA limit (dashed lines) is reached as n approaches the maximum number of steps into the past for which the

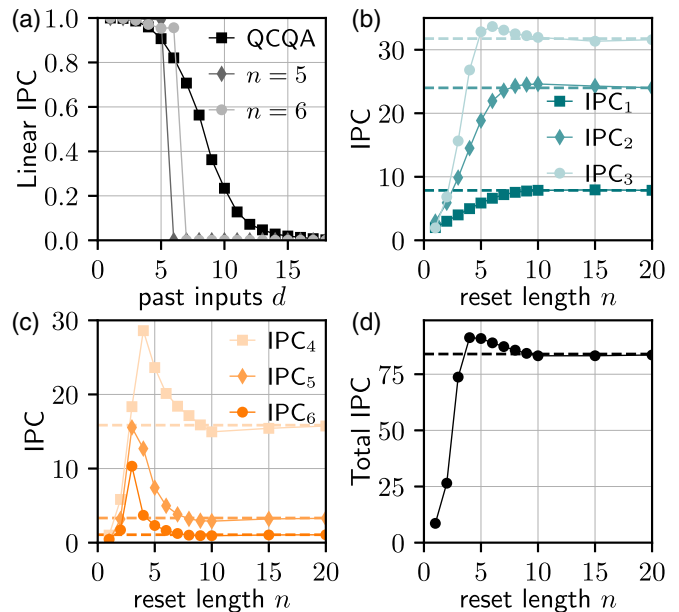


FIG. 2. Ising model: (a) Linear IPCs as a function of the steps into the past d for the LCQA with $n = 5, 6$ (grays) and for the QCQA (black). Summed IPCs of polynomial orders (b) 1–3 and (c) 4–6, and (d) the total summed IPC, in dependence of the reset length n using the LCQA. The corresponding QCQA limits are indicated by the dashed lines. We have calculated the standard deviation for ten different realizations of Ising Hamiltonians, where J_{ij} were sampled randomly.

reservoir has recall capabilities ($n \approx 15$). For very small n the IPCs are decreased due to the artificial memory restriction that is being imposed on the reservoir. However, there is an intermediate range for the reset length n where the total IPC and IPC_2 – IPC_6 are increased compared with the QCQA limit. This is an insight which can be used to substantially reduce the number of quantum operations needed for time-series tasks, while simultaneously optimizing the performance.

To gain more insight into this effect, in Fig. 2(a) the linear memory (delay d resolved individual linear IPCs) is plotted for $n = 5, 6$. Here, it can be seen that the distribution of the linear IPCs is changed compared with the QCQA case. Although the summed linear IPC (IPC_1) is decreased for $n = 5, 6$ [see Fig. 2(b) IPC_1 at $n = 5, 6$], the capacities for the past inputs which can be reconstructed are higher. This increase in the memory is related to the initial state of the reservoir. We initialize the reservoir in the pure state Eq. (6), but, as input data is added and the system evolves, the state of the reservoir becomes mixed and dependent on the input history. If we require the reservoir to recall an input from, for example, four steps into the past, then there is less variation in the reservoir response, and hence better recall performance, if the state of the reservoir before four steps was always the same and if this state was a pure state. The influence of the initial state can be seen in Fig. 3. Here, the linear IPCs are shown for a reset length of $n = 8$ for various initial states (see Appendix E for more results). For past inputs $d \leq n$ the LCQA with the pure initial state shows the best performance. When the initial state is chosen randomly, the performance is worst for small d , but approaches the QCQA for $d = n$. This

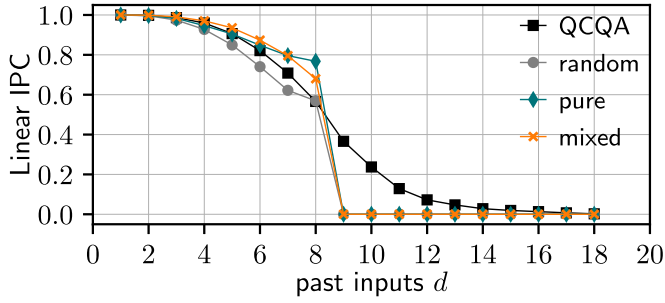


FIG. 3. Ising model: Linear IPCs as a function of the steps into the past d for the LCQA with reset length $n = 8$ and using three different initial conditions for reinitializing the reservoir; random state (gray), pure state [Eq. (6), green], and mixed state (orange). The QCQA is indicated in black.

means that the dynamics of the reservoir before the past state that is to be recalled are detrimental to the performance. This explains the increase in the summed higher-order IPCs, since the high-order IPCs are composed of inputs from fewer steps into the past, as can be inferred by the small reset lengths n needed to reach the QCQA limit as the polynomial order is increased [see Fig. 2(c)].

To demonstrate that the observed increases in the IPCs can translate to improved performance for a time-series prediction task, we also calculate two tasks related to the Lorenz chaotic attractor [27] (see Appendix B). In both tasks the x variable of the Lorenz system is inserted into the reservoir. The first task (LXX) is to predict the x variable one step ahead. The second task (LXZ) is to cross-predict the z variable one step ahead. Figure 4(a) shows the normalized root-mean-squared error (NRMSE) (as defined in Appendix B) for the LXX and LXZ tasks in dependence of the reset length n . For both tasks, compared with the QCQA limit (dashed lines), a lower NRMSE is achieved for small n . For this particular reservoir, the minimum NRMSE for both tasks is achieved at $n = 3$, which corresponds to the reset length at which IPC_5 and IPC_6 exhibit a maximum [see Fig. 2(c)]. These results demonstrate that the reset length can be used as a tuning parameter which does not require any changes to the physical reservoir. In general, the optimal reset length will depend on both the task and reservoir.

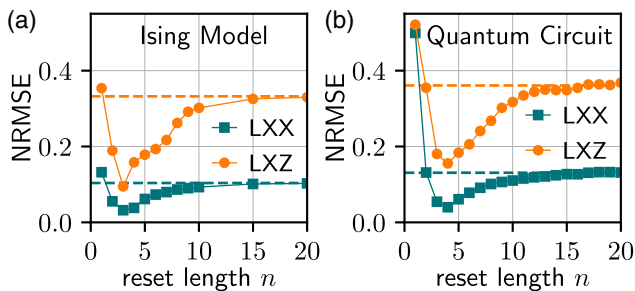


FIG. 4. Lorenz tasks: NRMSE of the LXX (green) and LXZ (orange) tasks for the LCQA as a function of the reset length n using (a) the Ising Reservoir and (b) the quantum circuit. The QCQA limit is indicated by dashed lines.

V. QUANTUM CIRCUIT

To demonstrate the universality of our restricted memory approach, the second type of quantum reservoir we consider is an N_S -qubit circuit. Each layer of the circuit consists of two sublayers of two-qubit unitary operators W_j and V_j acting on neighboring qubits. The unitaries W_j and V_j are of the form

$$\begin{aligned} W_j &= w_{j,1} U_{2j-1,2j}(a_j, b_j, c_j) w_{j,2} \\ V_j &= v_{j,1} U_{2j,2j+1}(d_j, e_j, f_j) v_{j,2}, \end{aligned} \quad (11)$$

with

$$U_{k,l}(a, b, c) = e^{iaX_k X_l + ibY_k Y_l + icZ_k Z_l}$$

and

$$\begin{aligned} w_{j,k} &= \left(\bigotimes_{k=1}^{j-2} I_2 \right) \otimes u_{j,k,1} \otimes u_{j,k,2} \left(\bigotimes_{k=j+1}^{N_S} I_2 \right) \\ v_{j,k} &= \left(\bigotimes_{k=1}^{j-1} I_2 \right) \otimes g_{j,k,1} \otimes g_{j,k,2} \left(\bigotimes_{k=j+2}^{N_S} I_2 \right), \end{aligned}$$

where the $u_{j,k,l}$ and $g_{j,k,l}$ are single-qubit unitaries drawn from the Haar measure [33] and a_i, \dots, f_j are uniformly drawn from the interval $[-k, h]$. The two sublayers are then defined by

$$W = \prod_j W_j \quad \text{and} \quad V = \prod_j V_j. \quad (12)$$

Recently, the authors of Ref. [34] showed that if the single-qubit unitaries w, v are drawn from the Haar distribution, see, e.g., Ref. [35] (Sec. 58), and $k = h$, then by changing h the system undergoes a transition between a localized and an ergodic phase. In our case, we draw the parameters randomly from the interval

$$a_i, \dots, f_j \in [0.1, 0.2]. \quad (13)$$

We implement both the QCQA and the LCQA scheme, where the unitary operation U is given by repeating the sublayers W and V , $N_W = 10$ times,

$$U = (VW)^{N_W}, \quad (14)$$

and where we time multiplex by performing an additional measurement after the application of each V and W layer individually. We use $N_S = 4$ qubits, thus, in total, there are $8 \times 10 = 80$ outputs of the reservoir for each input.

Figure 5 shows the various components of the IPC as a function of the reset length n . Here, we find the same qualitative results as in the Ising model case (see Fig. 2). The IPC_1 – IPC_6 can be increased with respect to the QCQA limit, and for sufficiently large n the QCQA limit is reached. The exact influence of the reset length on the distribution of the IPCs depends on the dynamics of the reservoir, as can be seen by the differences between Figs. 2 and 5.

For the quantum circuit, optimization of the reset length also leads to an improvement in the performance of the LXX and LXZ tasks, as shown in Fig. 4(b). Here, the best performances occur for $n = 4$, which corresponds to the maximum IPC_5 and IPC_6 for the quantum circuit.

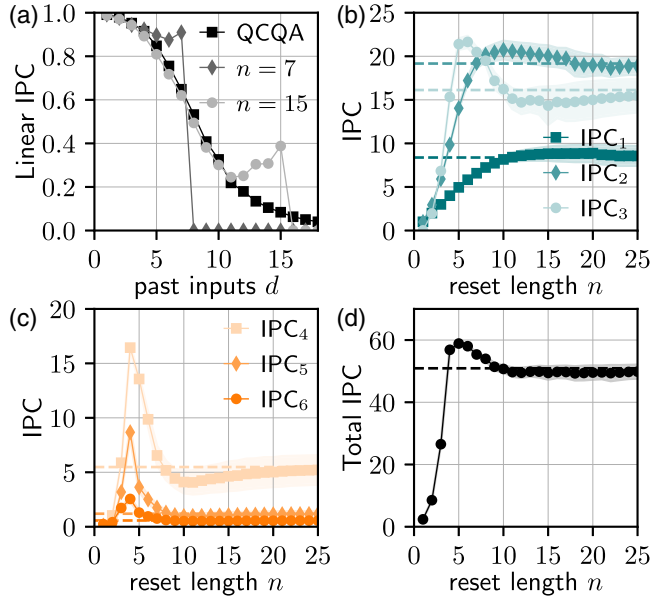


FIG. 5. Quantum circuit: (a) Linear IPCs as a function of the steps into the past d for the LCQA with $n = 7, 15$ (grays) and for the QCQA (black). Summed IPCs of polynomial orders (b) 1–3 and (c) 4 to 6, and (d) the total summed IPC, in dependence of the reset length n using the LCQA. The corresponding QCQA limits are indicated by the dashed lines. We have calculated the standard deviation for ten different realizations of the input signal and different random parameters a_j, \dots, f_j and single-qubit unitaries w, v in Eq. (11).

VI. CONCLUSION

In this work, we presented the LCQA for quantum reservoir computing. The algorithm allows the reset length to be used as a tuning parameter which controls the nonlinearity of the reservoir response and successfully reduces the time complexity of quantum reservoir computing for time-series tasks from quadratic to linear, thus making physical implementations for long time series feasible.

We have compared our LCQA approach to the established QCQA on a fully connected Ising chain and a quantum processor reservoir computer. We found that LCQA outperforms the currently utilized QCQA scheme both in the information processing capacity and in Lorenz time series prediction tasks.

The proposed approach allows the nonlinearity of the reservoir response to be tuned at the expense of the linear memory. For tasks requiring greater memory, the LCQA can be supplemented with memory augmentation methods on the input or the output of the reservoir, such as those presented in Refs. [36–39].

Our findings from the evaluation of the LCQA scheme using the quantum circuit indicate that this algorithm shows great potential for the hardware implementation of quantum reservoir computing not only to reduce the time needed to perform computations, but also to improve the performance. Furthermore, our results open questions about how the initial state of the quantum system can be used to tailor the non-linear response of the quantum reservoir. Another promising avenue of research is the applicability of this approach to hardware-implemented classical reservoirs, where one would

TABLE I. Simulation parameters.

Parameter	Value
Initialization steps	10 000
Training steps	50 000
Testing steps	5 000
Tikhonov regularization parameter quantum circuit	10^{-2}
Noise regularization parameter Ising model	10^{-6}
Sampling, quantum circuit	$a_j, \dots, f_j \in [0.1, 0.2]$
Haar distribution with parameters	Uniform on $[0.25, 0.75]$
Sampling, Ising model	$h_i = 5$
coupling J_{ij}	$T = 20$
Time evolution for Ising model	$N_Q = 4$
Number of qubits	

find a trade-off between tunability of the reservoir and computational cost.

ACKNOWLEDGMENT

L.J. and K.L. acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG), Grant No. LU 1729/3-1 (Projektnummer 445183921).

APPENDIX A: PARAMETERS

The simulation parameters are listed in Table I.

APPENDIX B: LORENZ TASK

The Lorenz 63 system is commonly used for time-series prediction benchmark tasks [27]. The Lorenz attractor dynamics are governed by Eq. (B1),

$$\begin{aligned} \dot{X} &= a(Y - X) \\ \dot{Y} &= X(b - Z) - Y \\ \dot{Z} &= XY - cZ. \end{aligned} \quad (\text{B1})$$

For time-series prediction, the system variables will be discretized, such that the series $X_n = X(n\Delta t)$, $Y_n = Y(n\Delta t)$, and $Z_n = Z(n\Delta t)$ are constructed. In this manuscript the Lorenz XX and Lorenz XZ tasks are used as target functions. The Lorenz XX task tries to predict the future of the X_n variable, where the reservoir is driven with the X_n . The Lorenz XZ task gets X_n as an input and tries to predict Z_n . The most commonly used parameters in research are $a = 10$, $b = 28$, and $c = 8/3$ with the discretization $\Delta t = 0.1$, which will be used here.

APPENDIX C: ERROR MEASURE

The NRMSE between $\mathbf{y} = (y_1, y_2, \dots, y_N)$ and $\mathbf{y}^{\text{targ}} = (y_1^{\text{targ}}, y_2^{\text{targ}}, \dots, y_N^{\text{targ}})$ is defined as

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^N (y_i - y_i^{\text{targ}})^2}{N \text{var}(\mathbf{y}^{\text{targ}})}} = \sqrt{1 - C(\mathbf{y}, \mathbf{y}^{\text{targ}})} \quad (\text{C1})$$

with

$$C(\mathbf{y}, \mathbf{y}^{\text{targ}}) = \frac{\text{cov}(\mathbf{y}, \mathbf{y}^{\text{targ}})}{\sigma^2(\mathbf{y})\sigma^2(\mathbf{y}^{\text{targ}})}, \quad (\text{C2})$$

where $\text{cov}(\cdot)$ and $\sigma(\cdot)$ are the covariance and standard deviation, respectively.

APPENDIX D: INFORMATION PROCESSING CAPACITY

The information processing capacity is a measure that quantifies the computational power of a reservoir computer and was introduced in Ref. [26]. This measure quantifies the ability of the reservoir to recall nonlinear combinations of past inputs by mapping the input response of the reservoir to a set of orthogonal functions. We choose the Legendre polynomials as the orthogonal basis. In the following we will show how the IPCs are calculated and define the summed IPC.

1. Legendre polynomials

The Legendre polynomials l_n are defined by

$$\begin{aligned} l_0(x) &= 1 \\ l_1(x) &= x \\ l_2(x) &= \frac{1}{2}(3x^2 - 1) \\ l_{n+1}(x) &= \frac{2n+1}{n+1}xl_n(x) - \frac{n}{n+1}l_{n-1}(x), \quad \text{for } n \geq 2. \end{aligned} \quad (\text{D1})$$

They are orthogonal under the scalar product:

$$\int_{-1}^1 l_i(x)l_j(x)dx = \delta_{i,j}. \quad (\text{D2})$$

2. Capacity

For a reservoir computer with a state matrix \mathbf{S} and the output $\mathbf{y} = \mathbf{S}\mathbf{W}^{\text{out}}$ with the target \mathbf{y}^{targ} , the capacity to calculate the target \mathbf{y}^{targ} is defined as

$$C(\mathbf{y}, \mathbf{y}^{\text{targ}}) = \frac{\text{cov}(\mathbf{y}, \mathbf{y}^{\text{targ}})}{\sigma^2(\mathbf{y})\sigma^2(\mathbf{y}^{\text{targ}})}. \quad (\text{D3})$$

$\text{cov}(\mathbf{y}, \mathbf{y}^{\text{targ}})$ is the covariance between the output \mathbf{y} and target \mathbf{y}^{targ} , while $\sigma(\mathbf{y})$ and $\sigma(\mathbf{y}^{\text{targ}})$ are the standard deviations of the output and target.

3. Calculating the information processing capacity

The information processing capacity tries to classify the computational power of a reservoir computer by examining how well the system maps a set of orthogonal functions, in our case Legendre polynomials. To do this the capacity of the reservoir to construct all possible combinations of the Legendre polynomials of all polynomial orders and all admitted combinations of the past inputs need to be calculated. For this the reservoir is fed with a sequence of input chosen from a uniform distribution between -1 and 1 .

a. Combinations of the Legendre polynomials

Let k be the order of a polynomial and let $\mathcal{D}^k = [d_1, d_2, \dots, d_q]$ denote a tuple with $0 < d_j \leq k$ and $d_a \leq d_b$ for $a < b$, such that $\sum_{j=1}^q d_j = k$ holds true. For each order k there exists a finite number of tuples N_k with these conditions. The o th. tuple is addressed by $\mathcal{D}_o^k = [d_{o,1}, d_{o,2}, \dots, d_{o,q}]$, where $o \in \{1, 2, \dots, N_k\}$ holds. Let M_o^k be the length of \mathcal{D}_o^k . For example, the L th. tuple of the K th. order is defined as $\mathcal{D}_L^K = [d_{L,1}, d_{L,2}, d_{L,3}]$, with $M_L^K = 3$ and $d_{L,1} + d_{L,2} + d_{L,3} = K$. The o th. polynomial of p th. order is defined as

$$p_o^k = \prod_{j=1}^{M_o^k} l_{d_{o,j}}, \quad (\text{D4})$$

where $l_{d_{o,j}}$ is the $d_{o,j}$ th. Legendre polynomial. We will give an example for the polynomial of order $k = 3$. There exist three combinations such that the above conditions are met: $\mathcal{D}_1^3 = [d_{1,1}] = [3]$, $\mathcal{D}_2^3 = [d_{2,1}, d_{2,2}] = [1, 2]$, and $\mathcal{D}_3^3 = [d_{3,1}, d_{3,2}, d_{3,3}] = [1, 1, 1]$. From these tuples the lengths $M_1^3 = 1$, $M_2^3 = 2$, and $M_3^3 = 3$ can be calculated. The polynomials of the third order are given by

$$\begin{aligned} p_1^3 &= \prod_{j=1}^{M_1^3} l_{d_{1,j}} = l_3 \\ p_2^3 &= \prod_{j=1}^{M_2^3} l_{d_{2,j}} = l_1 l_2 \\ p_3^3 &= \prod_{j=1}^{M_3^3} l_{d_{3,j}} = l_1 l_1 l_1. \end{aligned} \quad (\text{D5})$$

Assume the input vector $\mathbf{u}_{\text{all}} = (u_{-b+1}, u_{-b+2}, \dots, u_0, u_1, u_2, \dots, u_n)$. This vector can be split into the vector for the state matrix $\mathbf{u} = (u_1, u_2, \dots, u_n)$ and the vector prior $\mathbf{u}_{\text{prior}} = (u_{-b+1}, u_{-b+2}, \dots, u_0)$. $u(m)$ and $u(m - i_1)$ denote the m th. input and the shifted input by i_1 time steps into the past, where $-b \leq m \leq n$ and $i_1 \leq b$ holds. The shift of the system vector \mathbf{u} by i_1 time steps into the past will be addressed with $\mathbf{u}(i_1) = (u_{1-i_1}, u_{2-i_1}, \dots, u_{n-i_1})$. The a th. Legendre polynomial of the input vector \mathbf{u} will be denoted as $\mathbf{l}_a(\mathbf{u}) = [l_a(u_1), l_a(u_2), \dots, l_a(u_n)]$ and the shifted a th. Legendre polynomial will be defined as

$$\mathbf{l}_a(i_1) = \mathbf{l}_a[\mathbf{u}(i_1)] = [l_a(u_{1-i_1}), l_a(u_{2-i_1}), \dots, l_a(u_{n-i_1})]. \quad (\text{D6})$$

The point-wise multiplication between two vectors $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)$ results in a new vector $\mathbf{c} = \mathbf{a} \odot \mathbf{b} = (a_1, a_2, \dots, a_n) \odot (b_1, \dots, b_n) = (a_1 b_1, a_2 b_2, \dots, a_n b_n)$.

The polynomials for different steps in time are calculated by

$$\mathbf{p}_o^k(i_1, \dots, i_q) = \bigodot_{j=1}^{M_o^k} \mathbf{l}_{d_{o,j}}(i_j). \quad (\text{D7})$$

As an example, the polynomials of third order with regards to the past inputs i_j are given by

$$\begin{aligned} \mathbf{p}_1^3(i_1) &= \bigcirc_{j=1}^{M_1^3} \mathbf{l}_{d_{1,j}}(i_j) = \mathbf{l}_3(i_1) \\ \mathbf{p}_2^3(i_1, i_2) &= \bigcirc_{j=1}^{M_2^3} \mathbf{l}_{d_{1,j}}(i_j) = \mathbf{l}_1(i_1)\mathbf{l}_2(i_2) \\ \mathbf{p}_3^3(i_1, i_2, i_3) &= \bigcirc_{j=1}^{M_3^3} \mathbf{l}_{d_{1,j}}(i_j) = \mathbf{l}_1(i_1)\mathbf{l}_1(i_2)\mathbf{l}_1(i_3). \end{aligned} \quad (\text{D8})$$

b. Legendre memory accuracy

Assume that the target function is of the form $\mathbf{y}^{\text{targ}}(i_1, i_2, \dots) = \mathbf{p}_l^k(i_1, i_2, \dots)$ for one tuple of (i_1, i_2, \dots) . The capacity of such a target is defined as

$$\begin{aligned} C_{k,l}^{\text{comb}}(i_1, i_2, \dots) &= \frac{\text{cov}[\mathbf{y}, \mathbf{y}^{\text{targ}}(i_1, i_2, \dots)]}{\sigma^2(\mathbf{y})\sigma^2[\mathbf{y}^{\text{targ}}(i_1, i_2, \dots)]} \\ &= \frac{\text{cov}[\mathbf{y}, \mathbf{p}_l^k(i_1, i_2, \dots)]}{\sigma^2(\mathbf{y})\sigma^2[\mathbf{p}_l^k(i_1, i_2, \dots)]}. \end{aligned} \quad (\text{D9})$$

c. Combinations of past inputs

Valid combinations of (i_1, i_2, \dots) are those that comply with the following two conditions:

- (1) Only combinations are allowed where all i_1, i_2, \dots are unequal to each other ($i_a \neq i_b$, if $a \neq b$).
- (2) The second condition is best understood with an example.

Let $\mathbf{p}_1^7(i_1, \dots, i_4) = \mathbf{l}_1(i_1)\mathbf{l}_1(i_2)\mathbf{l}_1(i_3)\mathbf{l}_2(i_4)\mathbf{l}_2(i_5)$. There are three Legendre polynomials of first order with indices (i_1, i_2, i_3) and two Legendre polynomial of second order with indices (i_4, i_5) . The second condition is on the indices where the Legendre polynomials are of the same order. In these subsets only decreasing indices are allowed, meaning $i_1 > i_2 > i_3$ and $i_4 > i_5$. It is noted that i_4 can be greater than i_3 . One valid combination is $(i_1, i_2, i_3, i_4, i_5) = (5, 4, 2, 3, 1)$.

Example. Let $(i_1, i_2, \dots, i_n) \in \text{comb}$, where the set comb consists of all valid combinations. The combinations for $1 \leq i_j \leq 4$ for the first three orders of polynomials and one realization of fourth order are given by:

- (1) First-order polynomials:

$$p_{1,1}(i_1) = l_1(i_1), \quad i_1 \in \text{comb}_{1,1} = \{1, 2, 3, 4\}. \quad (\text{D10})$$

- (2) Second-order polynomials:

$$\begin{aligned} p_{2,1}(i_1) &= l_2(i_1), \quad i_1 \in \text{comb}_{2,1} = \{1, 2, 3, 4\}, \\ p_{2,2}(i_1, i_2) &= l_1(i_1)l_1(i_2), \\ (i_1, i_2) \in \text{comb}_{2,2} &= \{(2, 1), (3, 1), (4, 1), \\ &\quad (3, 2), (4, 2), (4, 3)\}. \end{aligned} \quad (\text{D11})$$

- (3) Third-order polynomials:

$$\begin{aligned} p_{3,1}(i_1) &= l_3(i_1), \quad i_1 \in \text{comb}_{3,1} = \{1, 2, 3, 4\} \\ p_{3,2}(i_1, i_2) &= l_1(i_1)l_2(i_2), \\ (i_1, i_2) \in \text{comb}_{3,2} &= \{(2, 1), (3, 1), (4, 1), (1, 2), \end{aligned}$$

$$(3, 2), (4, 2), (1, 3), (2, 3),$$

$$(4, 3), (1, 4), (2, 4), (3, 4)\}$$

$$p_{3,3}(i_1, i_2, i_3) = l_1(i_1)l_1(i_2)l_1(i_3),$$

$$(i_1, i_2, i_3) \in \text{comb}_{3,3} = \{(3, 2, 1), (4, 2, 1),$$

$$(4, 3, 1), (4, 3, 2)\}. \quad (\text{D12})$$

- (4) One fourth-order polynomial:

$$p_{4,4}(i_1, i_2, i_3) = l_1(i_1)l_1(i_1)l_2(i_2)$$

$$(i_1, i_2, i_3) \in \text{comb}_{4,4} = \{(3, 2, 1), (4, 2, 1), (4, 3, 1),$$

$$(3, 1, 2), (4, 1, 2), (4, 3, 2),$$

$$(2, 1, 3), (4, 1, 3), (4, 2, 3),$$

$$(2, 1, 4), (3, 1, 4), (3, 2, 4)\}. \quad (\text{D13})$$

This example should help the reader to better understand how the i_j are picked for each polynomial. In the next step a sum over all valid combinations $\text{comb}_{k,l}$ of $C_{k,l}^{\text{comb}}$ is taken, to get the capacity $C_{k,l}$ in regards to the target polynomial \mathbf{p}_l^k ,

$$C_{k,l} = \sum_{\text{comb}_{k,o}} C_{k,l}^{\text{comb}}(i_1, i_2, \dots). \quad (\text{D14})$$

d. Summed information processing capacities

As described above, N_k different polynomial combinations of degree k exist, where $o \in \{1, 2, \dots, N_k\}$ is the index for these polynomials. The k th.-order information processing capacity IPC_k is given as the sum over the capacities of all polynomial combinations of degree k :

$$\text{IPC}_k = \sum_l C_{k,l}. \quad (\text{D15})$$

The total IPC is obtained from the sum over all IPC_k ,

$$\text{IPC} = \sum_k \text{IPC}_k. \quad (\text{D16})$$

For an actual calculation the combinations to be calculated need to be restricted to a finite number. For this the fading memory property of the reservoir can be used, which ensured that $\lim_{i_1 \rightarrow \infty} \rightarrow 0$. Reservoir-dependent cutoffs for the polynomial order must also be implemented. Additionally, the total IPC has an upper bounded $C \leq N_R$, where N_R is the number of readout nodes [26], which can aid in determining suitable cutoff conditions.

APPENDIX E: SIMULATIONS WITH DIFFERENT INITIAL STARTING STATE

Figure 6 shows the capacity for reconstructing the input d steps into the past for different starting states using a reset length of $n = 15$. The legend is explained in Table II. The up, same random, entangled, mixed, and new random schemes all use the LCQA algorithm, where the starting state is changed according to the description in the table. Note that the random states are generated with the “rand_dm” method of QUTIP [40] with “density = 1”. The last scheme uses the QCQA scheme, where the previous state is reconstructed. For both the quantum circuit [Fig. 6(a)] and the Ising model [Fig. 6(b)], all capacitances show similar behavior up to $d = 10$. For further

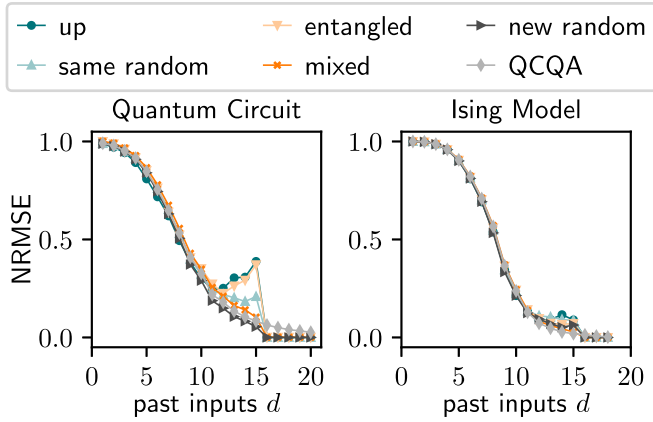


FIG. 6. Linear IPCs (C_1) for the quantum circuit (left) and Ising model (right) with a reset length of $n = 15$.

past inputs, we observe that the memory capacity is higher for some of the LCQA schemes. Qualitatively, we see that states with lower von Neumann entropy (i.e., the pure states “up” and “entangled”) remember their first inputs better than states with higher entropy (i.e., “same random,” or with even higher entropy, “new random” and “mixed”).

In all schemes, the input is inserted into the first qubit. The new density matrix of all qubits is formed by the tensor product between the density matrix of the first qubit $\rho_1(i)$ and the partial trace of the density matrix of the other qubits $\text{Tr}_1[\rho(i - 1)]$:

$$\rho(i) = \rho_1(i) \otimes \text{Tr}_1 \rho(i - 1). \quad (\text{E1})$$

TABLE II. Description of the legend of Fig. 6.

Label	Description
Up	$ 0000\rangle$
Same random	Same random state as starting state for each input
Entangled	$ 0000\rangle + 1111\rangle/\sqrt{2}$
Mixed	$\rho = Id_{16}/16$
New random	New random state as starting state for each input
QCQA	Scheme with quadratic complexity

By increasing the reset length n the newly constructed density matrix $\rho(i)$ is highly likely to be a mixed state. Similarly, a random state vector is expected to be mixed. The similarities between the new random state, QCQA scheme, and the mixed state suggest that the increase around the number of the reset length n of C_1 is influenced by two factors: (1) whether the starting state for each input is the same, and (2) whether the starting state for each input is mixed.

Similar behavior can be observed in the Ising model, albeit to a lesser extent. Once again, the “up,” “same random,” and “entangled” starting states remember early inputs better than the “mixed” starting state, “new random” starting state, and the QCQA scheme. In this case as well, C_1 of the QCQA scheme exhibits almost identical behavior to the new random and mixed states.

APPENDIX F: ISING MODEL: INFLUENCE OF THE QUBIT NUMBER

When the reservoir sampling dimension increases, so does the maximum possible total IPC [26]. This can translate to improved performance on specific tasks. In Fig. 7 the influence

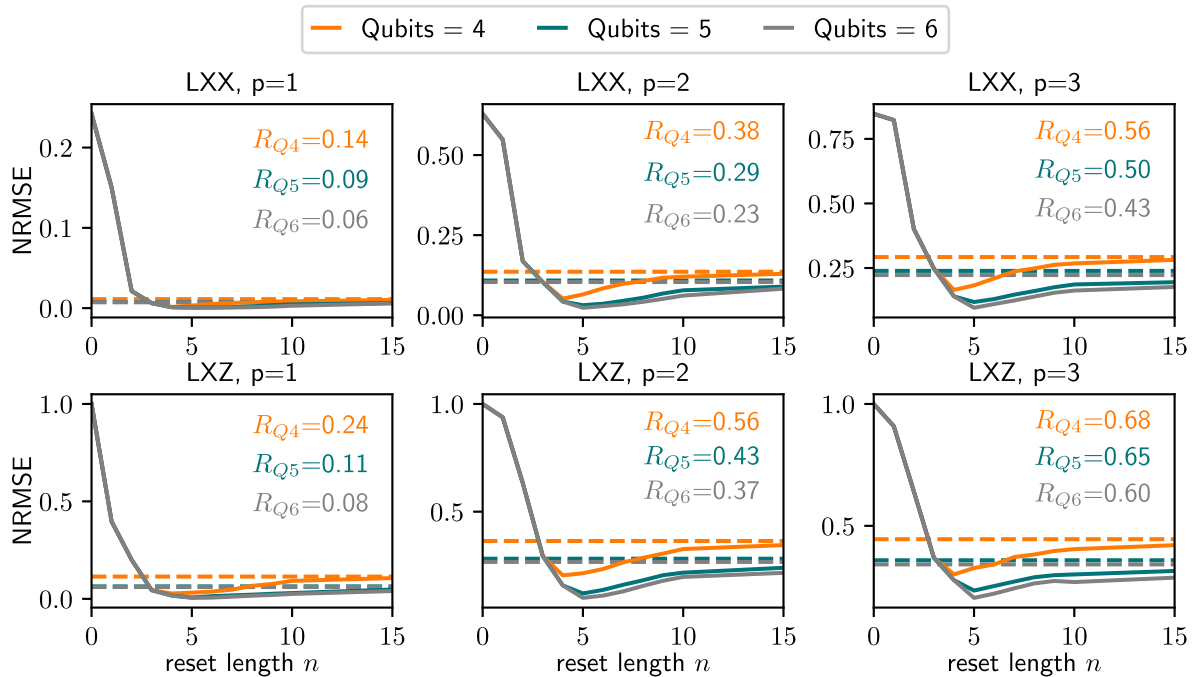


FIG. 7. NRMSE for the LXX and LXZ tasks with p step ahead prediction as a function of the reset length n for four-, five-, and six-qubit Ising model reservoirs. The QCQA limit is indicated by the dashed lines. The value R_{Q_i} gives the minimum NRMSE achieved with the LCQA divided by the QCQA limit for $i = 4, 5, 6$ qubits.

of increasing the sampling dimension is investigated by comparing Ising models with four, five, and six qubits. With time multiplexing this corresponds to sampling dimensions of 120, 150, and 180, respectively. In each case a clear improvement is evident when the reset length is optimized. The relative impact is slightly increased when the qubit number is larger, as indicated by the decrease in the R_Q (R_Q is the minimum NRMSE achieved with the LCQA divided by the QCQA limit) values as the qubit number increases. For example, for the Lorenz cross prediction task (LXZ, $p = 1$) the shortened

reset length reduces the NRMSE to 0.24, 0.11, and 0.08 of the QCQA limit for four, five, and six qubits, respectively.

In general, a higher reservoir sampling dimension will lead to better performance. But at some point this performance improvement will saturate. It is possible that if this limit is reached, then no further improvement will be found by tuning the reset length. However, our approach can enable the same performance while using a much smaller reservoir, which can make hardware implementation more feasible and more energy efficient.

-
- [1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2009).
- [2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
- [3] 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).
- [4] Z. Li, X. Liu, N. Xu, and J. Du, Experimental realization of a quantum support vector machine, *Phys. Rev. Lett.* **114**, 140504 (2015).
- [5] Y. Li, G. W. Holloway, S. C. Benjamin, G. A. D. Briggs, J. Baugh, and J. A. Mol, Double quantum dot memristor, *Phys. Rev. B* **96**, 075446 (2017).
- [6] V. Saggio, B. E. Asenbeck, A. Hamann, T. Strömberg, P. Schiansky, V. Dunjko, N. Friis, N. C. Harris, M. Hochberg, D. Englund, S. Wölk, H. J. Briegel, and P. Walther, Experimental quantum speed-up in reinforcement learning agents, *Nature (London)* **591**, 229 (2021).
- [7] J. Chen, H. I. Nurdin, and N. Yamamoto, Temporal information processing on noisy quantum computers, *Phys. Rev. Appl.* **14**, 024065 (2020).
- [8] Y. Suzuki, Q. Gao, K. C. Pradel, K. Yasuoka, and N. Yamamoto, Natural quantum reservoir computing for temporal information processing, *Sci. Rep.* **12**, 1353 (2022).
- [9] 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).
- [10] H. Jaeger, The Echo State' Approach to Analysing and Training Recurrent Neural Networks, GMD Report 148 (GMD - German National Research Institute for Computer Science, 2001).
- [11] 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).
- [12] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, An experimental unification of reservoir computing methods, *Neural Networks* **20**, 391 (2007).
- [13] 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).
- [14] K. Fujii and K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning, *Phys. Rev. Appl.* **8**, 024030 (2017).
- [15] 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).
- [16] A. Kutvonen, K. Fujii, and T. Sagawa, Optimizing a quantum reservoir computer for time series prediction, *Sci. Rep.* **10**, 14687 (2020).
- [17] R. Martínez-Peña, G. L. Giorgi, J. Nokkala, M. C. Soriano, and R. Zambrini, Dynamical phase transitions in quantum reservoir computing, *Phys. Rev. Lett.* **127**, 100502 (2021).
- [18] W. Xia, J. Zou, X. Qiu, and X. Li, The reservoir learning power across quantum many-body localization transition, *Front. Phys.* **17**, 33506 (2022).
- [19] P. Pfeffer, F. Heyder, and J. Schumacher, Hybrid quantum-classical reservoir computing of thermal convection flow, *Phys. Rev. Res.* **4**, 033176 (2022).
- [20] G. Angelatos, S. A. Khan, and H. E. Türeci, Reservoir computing approach to quantum state measurement, *Phys. Rev. X* **11**, 041062 (2021).
- [21] S. A. Khan, F. Hu, G. Angelatos, and H. E. Türeci, Physical reservoir computing using finitely-sampled quantum systems, *arXiv:2110.13849* (2021).
- [22] P. Mujal, R. Martínez-Peña, J. Nokkala, J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Opportunities in quantum reservoir computing and extreme learning machines, *Adv. Quantum Technol.* **4**, 2100027 (2021).
- [23] D. Cory, R. Laflamme, E. Knill, L. Viola, T. Havel, N. Boulant, G. Boutis, E. Fortunato, S. Lloyd, R. Martinez, C. Negrevergne, M. Pravia, Y. Sharf, G. Teklemariam, Y. Weinstein, and W. Zurek, NMR based quantum information processing: Achievements and prospects, *Fortschr. Phys.* **48**, 875 (2000).
- [24] J. A. Jones, Quantum computing with NMR, *Prog. Nucl. Magn. Reson. Spectrosc.* **59**, 91 (2011).
- [25] P. Mujal, R. Martínez-Peña, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Time-series quantum reservoir computing with weak and projective measurements, *npj Quantum Inf.* **9**, 16 (2023).
- [26] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, Information processing capacity of dynamical systems, *Sci. Rep.* **2**, 514 (2012).
- [27] E. N. Lorenz, Deterministic nonperiodic flow, *J. Atmos. Sci.* **20**, 130 (1963).
- [28] T. Hülser, F. Köster, L. C. Jaurigue, and K. Lüdge, Role of delay-times in delay-based photonic reservoir computing, *Opt. Mater. Express* **12**, 1214 (2022).
- [29] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen, Memory versus non-linearity in reservoirs, in *The 2010 International*

- Joint Conference on Neural Networks (IJCNN)* (IEEE, 2010), pp. 1–8.
- [30] J. B. Butcher, D. Verstraeten, B. Schrauwen, C. R. Day, and P. W. Haycock, Reservoir computing and extreme learning machines for non-linear time-series data analysis, *Neural Networks* **38**, 76 (2013).
- [31] M. Inubushi and K. Yoshimura, Reservoir computing beyond memory-nonlinearity trade-off, *Sci. Rep.* **7**, 10199 (2017).
- [32] T. Hülser, F. Köster, K. Lüdge, and L. C. Jaurigue, Deriving task specific performance from the information processing capacity of a reservoir computer, *Nanophotonics* **12**, 937 (2023).
- [33] J. Diestel and A. Spalsbury, *The Joys of Haar Measure*, Graduate Studies in Mathematics (American Mathematical Society, Providence, 2014).
- [34] C. Sünderhauf, D. Pérez-García, D. A. Huse, N. Schuch, and J. I. Cirac, Localization with random time-periodic quantum circuits, *Phys. Rev. B* **98**, 134204 (2018).
- [35] P. R. Halmos, *Measure Theory* (Springer, New York, 1974).
- [36] B. A. Marquez, J. Suarez-Vargas, and B. J. Shastri, Takens-inspired neuromorphic processor: A downsizing tool for random recurrent neural networks via feature extraction, *Phys. Rev. Res.* **1**, 033030 (2019).
- [37] E. Del Frate, A. Shirin, and F. Sorrentino, Reservoir computing with random and optimized time-shifts, *Chaos* **31**, 121103 (2021).
- [38] L. C. Jaurigue, E. Robertson, J. Wolters, and K. Lüdge, Reservoir computing with delayed input for fast and easy optimization, *Entropy* **23**, 1560 (2021).
- [39] T. L. Carroll and J. D. Hart, Time shifts to reduce the size of reservoir computers, *Chaos* **32**, 083122 (2022).
- [40] J. Johansson, P. Nation, and F. Nori, QUTIP 2: A python framework for the dynamics of open quantum systems, *Comput. Phys. Commun.* **184**, 1234 (2013).