




QuantumReservoirPy: A Software Package for Time Series Prediction

Ola Tangen Kulseng ^{1*}, Stanley Miao^{2*}, Franz G. Fuchs ^{3*}, and Alexander Stasik ^{3,4*}¶

² David R. Cheriton School of Computer Science, University of Waterloo, Canada ¹ Department of Physics, Norwegian University of Science and Technology (NTNU), Norway ³ Department of Mathematics and Cybernetics, SINTEF Digital, Norway ⁴ Department of Data Science, Norwegian University of Life Science, Norway ¶ Corresponding author * These authors contributed equally.

DOI: [10.21105/joss.07994](https://doi.org/10.21105/joss.07994)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Daniel S. Katz](#) 

Reviewers:

- [@oblivateandsurrender](#)
- [@rudraprsd](#)
- [@Abinashbunty](#)

Submitted: 05 March 2025

Published: 10 June 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Reservoir computing on quantum computers has recently emerged as a potential resource for time series prediction, owing to its inherent complex dynamics. To advance Quantum Reservoir Computing (QRC) research, we have developed the Python package [QuantumReservoirPy](#), which facilitates QRC using quantum circuits as reservoirs.

The package is designed to be easy to use, while staying completely customizable. The resulting interface, similar to that of [reservoirpy](#) ([Trouvain et al., 2020](#)), simplifies the development of quantum reservoirs, and provides logical methods of comparison between reservoir architectures.

Statement of need

Reservoir computing (RC) is a paradigm in machine learning for time series prediction. With recent developments, it has shown a promising efficacy compared to conventional neural networks, owing to its relatively simple training process ([Tanaka et al., 2019](#)).

The main building block of RC is a dynamical system called a reservoir. By making the dynamics of the reservoir dependent on the input time series, the state of the reservoir becomes a high-dimensional, non-linear transformation of the time series. The hope is that such a low-to-high dimensional encoding enables forecasting using a relatively simple method like Ridge regression. A reservoir can be virtual, such as a sparsely-connected recurrent neural network with random fixed weights, termed an echo state network ([Jaeger & Haas, 2004](#)), or even physical, such as a bucket of water ([Fernando & Sojakka, 2003](#)). See [Figure 1](#) for an illustration of a typical RC pipeline.

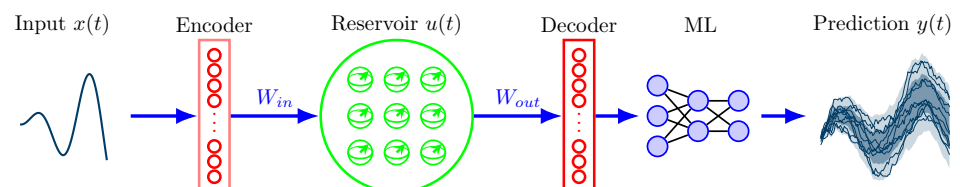


Figure 1: A quantum reservoir system consists of a learning task, an en- and de-coder (red), and the dynamic system itself (green). In standard RC the machine learning part is linear regression.

QRC is a proposed method of RC utilizing quantum circuits as reservoirs. Multi-qubit systems with the capability of quantum entanglement provide compelling non-linear dynamics that match the requirements for a feasible reservoir. Furthermore, the exponentially-scaling Hilbert space of large multi-qubit systems support the efficiency and state-storage goals of RC. As a result, quantum computers have been touted as a viable dynamical system to produce the intended effects of RC.

Existing implementations of QRC have used proprietary realizations on simulated and actual quantum computers. The lack of a shared structure between implementations has resulted in a disconnect in comparing reservoir architectures. In addition, individual implementations require a certain amount of redundant procedure prior to the involvement of specific concepts.

We observe that there is a need for a common framework for the implementation of QRC. As such, we have developed QuantumReservoirPy to solve the issues presented in current QRC research. QuantumReservoirPy is designed to handle every step of the RC pipeline, in addition to the pre- and post-processing needed in the quantum case. In providing this software package, we hope to facilitate logical methods of comparison in QRC architecture and enable a simplified process of creating a custom reservoir from off-the-shelf libraries with minimal overhead requirements to begin development.

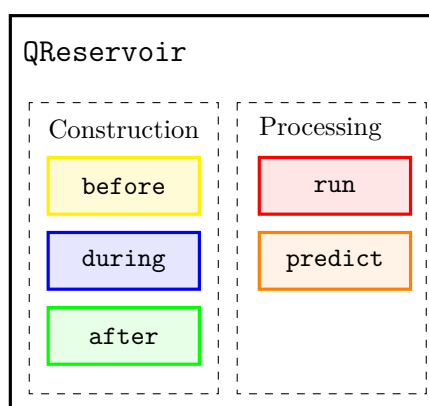
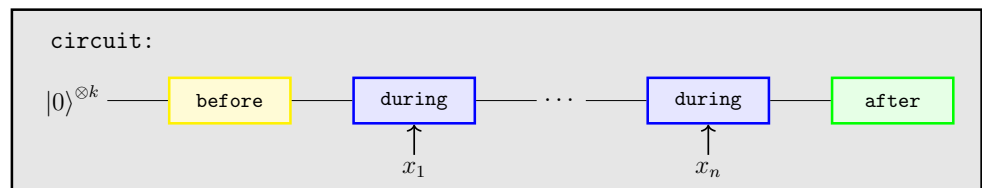


Figure 2: Quantum circuit construction may be customized through the before, during, and after methods and a timeseries processed with the run and predict methods.

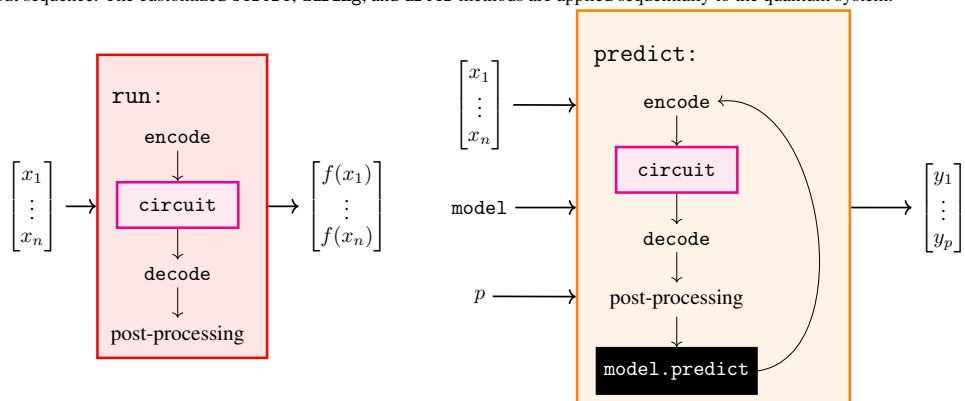
Design and implementation

We intend QuantumReservoirPy to provide flexibility to all possible designs of quantum reservoirs, with full control over pre-processing, input, quantum circuit operations, measurement, and post-processing. In particular, we take inspiration from the simple and flexible structure provided by the ReservoirPy software package [reservoirpy](#) (Trouvain et al., 2020).

The construction methods of QuantumReservoirPy serve as sequential operations performed on the quantum system. All of them may include an arbitrary combination of operations from the [Qiskit Circuit Library](#). before and after are independent of the time series, and are applied before and after the time series is processed, respectively. Operations in during are applied per timestep, and most closely determine the dynamical properties of the reservoir. [Figure 3a](#) demonstrates the aforementioned arrangement, which is implemented as a hidden intermediary process in a QuantumReservoirPy quantum reservoir.



(a) A functional overview of the hidden quantum circuit architecture common to all quantum reservoirs, where x_t is the observed input sequence. The customized before, during, and after methods are applied sequentially to the quantum system.



(b) The input timeseries is encoded into the quantum circuit. After measurements are decoded from the quantum circuit, they are post-processed into the transformed feature vector. (c) The feature vector produced by encoding, decoding, and pre-processing is used by the model to predict the next step in the timeseries. This process is repeated p times and returns the resulting prediction sequence.

Figure 3: The intended functionality of the run and predict method. The observed input sequence is $\{x_t\}$ and the target sequence $\{y_t\}$. The reservoir f performs an evolution in time.

The processing methods do not affect the creation of the reservoirs, but are included to keep a coherent interface to `reservoirpy`. Calling `run` on a time series returns the non-linear embeddings for each timestep. Depending on the realization of QRC, such as averaging over multi-shot data, additional post-processing can be included in the `run` method to achieve the desired output. Figure 3b provides a visualization of the `run` method. `predict` functions as a complete forecasting process including encoding, decoding, and post-processing. Additionally, a trained simple machine learning model is used to predict the next step in the timeseries from the transformed and post-processed data. The resulting prediction is then fed in as input for the following prediction, which occurs as an iterative process until the specified number of forecasting steps is reached. At this point, the `predict` method returns the sequence of predictions from each iteration. Figure 3c provides a visualization of the `predict` method.

Package Details

Dependencies

The three main dependencies of `QuantumReservoirPy` are `numpy`, `qiskit`, and `scikit-learn`, with python versions above 3.9. Qiskit is [deprecating](#) python 3.9 support in the 2.1.0 version, and the package presented here is developed to support `qiskit=2.0.x`. As for the other packages, the supported versions of `scikit-learn` and `numpy` follows from their interrelated constraints as well as the constraint from `qiskit`. In the install script, we specify `numpy>1.17`. We strive for `QuantumReservoirPy` to support compatibility with existing reservoir computing and quantum computing workflows.

Much of existing research in QRC is performed on IBM devices and simulators (see Yasuda et al. (2023) and Suzuki et al. (2022)), programmed through the Qiskit software package. To minimize disruption in current workflows, `QuantumReservoirPy` is built as a package to

interact with Qiskit circuits and backends. It is expected that the user also uses Qiskit in the customization of reservoir architecture when working with QuantumReservoirPy.

License

QuantumReservoirPy is licensed under the GNU General Public License v3.0. QuantumReservoirPy also includes derivative work of Qiskit, which is licensed by IBM under the Apache License, Version 2.0.

Further Development

The authors continue to support and maintain the project. Users may report package issues and desired features by opening an issue on the public GitHub repository or contacting the authors by email. Additional opportunities for further development on QuantumReservoirPy include supplementary built-in processing schemes, expanded features for data visualization, and reservoir evaluation methods.

Acknowledgements

Work in this project was supported by the NTNU and SINTEF Digital through the International Work-Integrated-Learning in Artificial Intelligence (IWIL AI) program, in partnership with SFI NorwAI and the Waterloo Artificial Intelligence Institute (Waterloo.AI). IWIL AI is funded by the Norwegian Directorate for Higher Education and Skills (HK-dir).

References

- Fernando, C., & Sojakka, S. (2003). Pattern recognition in a bucket. *Advances in Artificial Life*, 588–597. ISBN: 978-3-540-39432-7
- Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667), 78–80. <https://doi.org/10.1126/science.1091277>
- Suzuki, Y., Gao, Q., Pradel, K. C., Yasuoka, K., & Yamamoto, N. (2022). Natural quantum reservoir computing for temporal information processing. *Scientific Reports*, 12(1), 1353. <https://doi.org/10.1038/s41598-022-05061-w>
- Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., & Hirose, A. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, 115, 100–123.
- Trouvain, N., Pedrelli, L., Dinh, T. T., & Hinaut, X. (2020). ReservoirPy: An efficient and user-friendly library to design echo state networks. In I. Farkas, P. Masulli, & S. Wermter (Eds.), *Artificial neural networks and machine learning – ICANN 2020* (pp. 494–505). Springer International Publishing. https://doi.org/10.1007/978-3-030-61616-8_40
- Yasuda, T., Suzuki, Y., Kubota, T., Nakajima, K., Gao, Q., Zhang, W., Shimono, S., Nurdin, H. I., & Yamamoto, N. (2023). *Quantum reservoir computing with repeated measurements on superconducting devices*. <https://arxiv.org/abs/2310.06706>