Time-series Prediction Using Quantum Reservoirs

Will Lienert[1] and Francesco Campaioli[1,2]

[1] *RMIT University*

October 31, 2025

Abstract

This project explores the feasibility of using quantum reservoir computing (QRC) for time series prediction by simulating small many-body quantum systems using Python and QuTiP. Reservoir computing offers a way to process and store temporal data by using a fixed recurrent network and training only a linear output layer. Extending this concept to quantum systems allows us to take advantage of the high-dimensionality of Hilbert space in a quantum system to generate complex feature representations of an input. We implemented a ten-qubit transverse-field Ising model as a quantum reservoir and tested its ability to predict the Mackey–Glass time series, a well-known chaotic time series produced from a delayed differential equation. Techniques for constructing a classical memory state was explored, including a new "split-permutation" technique that separates single-site and co-occurrence observables in the classical memory. The model achieved strong predictive performance with a low mean-squared error (MSE) and revealed an interesting trade-off between feature richness and noise as the number of qubits is increased. These results demonstrate the effectiveness for even small simulated quantum reservoirs to capture non-linear temporal relations leading to accurate time-series prediction. While scaling to real hardware introduces challenges such as decoherence and noise, the findings highlight the potential of QRCs as a useful architecture for hybrid quantum–classical machine learning.

1 Background

Modelling complex, dynamic systems over time remains a significant challenge in machine learning, requiring architectures that can model systems with long-term dependencies. This is well-suited to a type of neural network called a recurrent neural network (RNN). An RNN

feeds its own state and output back into itself, thus evolving its state over time, much like humans do. These networks retain abstract memory of the past context and, therefore, are more effective in time-series predictions, making them an ideal area of research [1].

A **Reservoir** Computer (RC) is a type of RNN that consists of an interconnected set of nodes that communicate with fixed weights and biases. This fixed network is known as the reservoir. A common implementation of an RC uses an echo-state-network (ESN) as the reservoir. An ESN consists of a set of nodes that represent a state $x(t)$ at time t. The state then evolves in time with the equation

$$x(t+1) = W_{res}x(t) + W_{in}u(t) + W_{fb}y(t) \tag{1}$$

As you can see in Equation 1, the state for a given time-step is a function of both the state and output from the previous time-step. An example of this can be seen within the large circle in Figure 1. Unlike a traditional neural network, the weights connecting the nodes in a reservoir are assigned randomly and remain fixed. During training, only a linear readout layer extracts features from the reservoir and is trained using a simple linear fit algorithm. This allows for a powerful set of complex interactions within the reservoir without increasing the complexity of training [2]. One of the biggest advantages of having fixed weights is that the reservoir can be swapped out for any complex dynamical system, even a physical one, since the linear readout layer can be trained to tap into the innate computational power of the physical system.

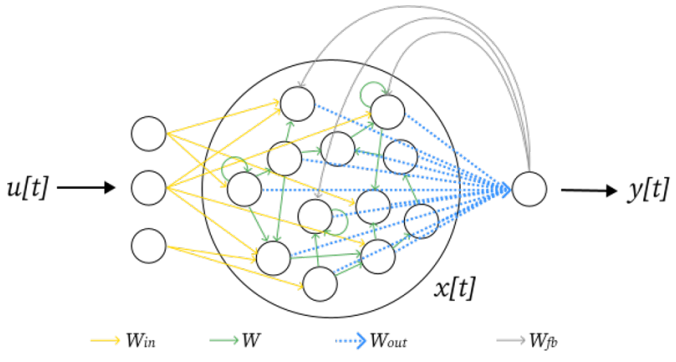

Figure 1: Diagram of a reservoir computer. $u[t]$ is an input vector from a single time-step in a time series, $x[t]$ is the state of the reservoir, and $y[t]$ is the output vector which predicts the value of the time series at a future time-step.

Quantum Reservoir Computing (QRC). A many-body quantum system consists of multiple quantum objects such as electrons, ions, or photons that interact in some way. If these objects have strong interactions with each other, the system is notoriously complex and difficult to simulate with a classical computer. This is because the description of the wave-function generally requires knowledge of a set of complex coefficients, of which the number grows exponentially with the number of quantum objects involved [3]. However, advantageously, when a real quantum system is used, the computation required no longer grows exponentially with the number of quantum objects, yet these complex coefficients can be measured and may encode rich relational information about the system. For this reason, many researchers have been attempting to use quantum systems as the reservoir in RCs. A quantum reservoir allows you to tap into these complex coefficients by tracking not only the occurrence of individual qubit readouts but also the co-occurrence of qubit readouts. For these reasons, the potential benefits of quantum reservoir computers could be significant.

One of the most promising systems to act as a reservoir is a quantum annealer (QA). A QA is a specialised type of quantum computer that is designed for solving optimisation problems and may be ideal for use in QRCs [4]. QAs allow a set of qubits to interact in way that is described by the Ising Hamiltonian.

$$H = -\sum_{i<j} J_{i,j} Z_i Z_j - \sum_i h_i Z_i \tag{2}$$

The Ising Hamiltonian consists of a coupling term and a local field term, making it analogous to a classical reservoir network that consists of connections between nodes and an input that is applied to each node. Unlike gate based quantum computers, QAs have been functional for decades and can complete computations with over 5,000 qubits. This means that QRCs have the potential to be of practical use and going beyond the theoretical. QRCs have potential use in predicting weather, stock markets, biological systems, and other complex time series' [5]. One time series that QRCs have had some success with predicting is the Mackey-Glass (MG) equation. The MG equation is a delay differential equation of the form

$$\frac{dx}{dt} = \beta \frac{x(t-\tau)}{1 + x(t-\tau)^n} - \gamma x(t) \tag{3}$$

It is known to be chaotic and difficult to predict as it is very sensitive to its initial conditions. An example of a MG time series can be seen in Figure 2.

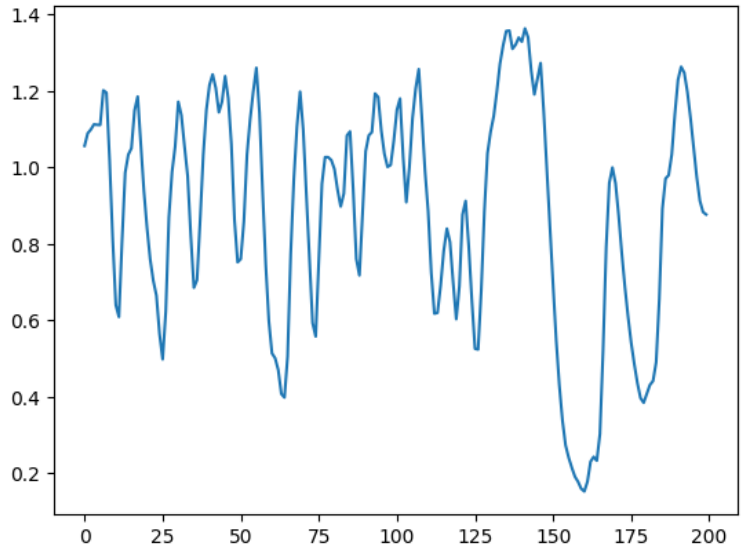

Figure 2: Example of a Mackey-Glass Chaotic time series.

Although using quantum systems come with their benefits, QRCs don't come without obstacles. Current research runs into two main problems when compared to classical systems. Firstly, taking any measurement of the system collapses its state, leading to the loss of its rich temporal memory created over prior inputs to the system [6]. Researchers have proposed solutions. For example, Kobayashi et al. [6] have demonstrated a feedback-driven architecture, where measurement from the collapsed state is encoded back into the next state of the reservoir. While Settino et al. [7] attempt to construct a classical reservoir state and use cyclic permutations on the classical reservoir state to encode time. A diagram of their approach can be seen in Figure 3. Their classical memory equation shows a permutation

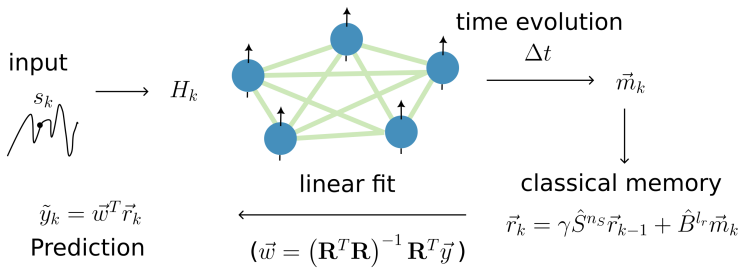

Figure 3: Example of the architecture used by Settino et al. [7] to implement a classical memory state for a quantum reservoir computer.

operator, $\hat{S}^{n_S}$, that is defined as

$$\hat{S}_{ij}^{n_S} = \delta_{(i+n_S) \bmod l_r, j} \tag{4}$$

An example of the transformation is as shown,

$$\hat{S}^1 \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ z_4 \\ c_1 \end{pmatrix} \tag{5}$$

Both of these solutions enabled accurate predictions of chaotic time series. However the current technique for encoding time through cyclic permutations treats all the observables with no discrimination. This is an area that has potential for improvement by experimentation with more complex techniques that separate the observables during permutations. The second obstacle is how to encode and decode the input and output data to take advantage of Hilbert space. The use of a quantum system does not simply translate to a higher feature space unless the input is encoded optimally. Schütte et al. [8] state that new input injection paradigms are needed to take advantage of a bigger slice of Hilbert space. The bottlenecks I have stated highlight the need for more research, but also provide direction for future studies [9].

2 Aims

In this project, our aim was to research and understand the feasibility of using a quantum reservoir computer for time series prediction by implementing a simulation of a small quantum reservoir computer using Python and QuTiP. We were successful in implementing a flexible framework for generating an arbitrarily sized quantum reservoir and tested multiple techniques for encoding data into the network to make time series predictions.

Objectives

1. Simulate a single-qubit two-level quantum system and perform an annealing schedule using QuTiP.

2. Extend the simulation to a 2-qubit two-level quantum system to explore quantum many-body dynamics.

3. Implement a 6-qubit quantum reservoir computer using an annealing schedule with QuTiP to perform a one-step-ahead time series prediction.

3 Methodology

In this project, we simulated many-body quantum systems to implement a time-series prediction approach based on quantum reservoir computing and evaluated its performance. A quantum system is described by its wave function $\Psi(t)$, and its evolution over time is completely described by the time-dependent Schrodinger equation.

$$i\hbar\frac{d}{dt}\left|\Psi(t)\right\rangle = H(t)\left|\Psi(t)\right\rangle \tag{6}$$

$H(t)$ is the Hamiltonian; which describes the total energy of the system. This is all we need to model the future dynamics of the system. The Hamiltonian we used in this project followed the form of the transverse-field Ising Hamiltonian with an additional input-dependent term that represents a local magnetic field in the z-direction at each qubit site,

$$H = \sum_{i<j} J_{i,j} Z_i Z_j + h\sum_i X_i + u_k \sum_i C_i Z_i \tag{7}$$

where:

- Z_i/X_i – Pauli matrix in the z/x-direction for the i-th qubit.
- $J_{i,j}$ – Coupling between qubits i and j.
- h – Constant transverse-field applied in the x-direction to all qubits.
- u_k – Value of the input time-series at step k.
- C_i – Weight of input time-series for qubit i.

The Hamiltonian shown in Equation (7) consists of three terms. The first term defines the energy cost of qubits i and j being aligned in the z-direction. The second term defines a transverse magnetic field along the x-direction. This introduces quantum fluctuations that drive the system to diverge from classical mechanics, allowing for tunnelling and superposition of states. The final term allows the input time-series to be encoded into the Hamiltonian.

Using the Hamiltonian defined in Equation (7), we solve for the expectation values of a set

of observables, $\vec{m}_k$, using the Schrodinger equation, Equation (6). The set of observables consists of

$$\vec{m}_k = \left\{\langle\sigma_i^z\rangle\right\}_{i=1}^{N_q} \cup \left\{\langle\sigma_i^z\sigma_j^z\rangle\right\}_{i<j} \tag{8}$$

Note that the Hamiltonian is time-independent and depends on only a single value u_k for each readout, m_k. Each time we take a readout, we collapse the state of the system, losing any memory the reservoir had. We instead create a classical state using the observable readouts from the quantum reservoir, using their expected values to act as a memory.

$$r_k = \gamma R\vec{r}_{k-1} + \vec{m}_k \tag{9}$$

where R is a transformation matrix rotates the positions of the features,

$$R_{ij} = \delta_{(i+1) \bmod N,\, j} \tag{10}$$

We also tested a more complex transformation technique where we performed a split cyclic permutation like so,

$$R_{ij}^{(split)} = \begin{cases} \delta_{\,j,\,(i+1) \bmod N_q}, & 0 \le i < N_q, \\ \delta_{\,j,\,N_q + \left((i-N+1) \bmod (N-N_q)\right)}, & N_q \le i < N, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

N is the number of observables and N_q is the number is qubits. This is the equivalent of a cyclic permutation on the positions of the z-spin readouts and the co-occurrence readouts as separate vectors and then combining them back together. For example,

$$R\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ c_4 \\ c_5 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ z_1 \\ c_5 \\ c_4 \end{pmatrix} \tag{12}$$

For 3 qubits and 2 co-occurrence observables ($N_q = 3$, $N = 5$). This is done to keep the two observables, which are extracting fundamentally different features from the state, separate. This is an attempt to reduce unwanted chaos in the classical memory state. These values are then used to calculate a linear weighting vector $\vec{w}$ that can be applied to $\vec{a}_k$ to make a prediction $\tilde{y}_k$ for some future step in the time-series,

$$\tilde{y}_k = \sum_i \vec{w}^\top \vec{r}_k \tag{13}$$

For this project, we used the programming language Python and the QuTiP package. QuTiP is used to simulate quantum systems. It provides tools for defining Hamiltonians and solving the subsequent Schrodinger equation for expectation values of measurement operators we specify. The Hamiltonian shown in Equation (7) is defined using the sigmax() and sigmaz() functions shown in Listing 1.

The reservoir is initiated in the ground-state of the transverse term of the Hamiltonian,

$$\sum_i C_i Z_i \tag{14}$$

The state is then evolved using the *sesolve*() function over a time period (ΔT) of 1.0, and the expectation values of the observables from Equation 8. This is repeated for every value of k . Recording the classical memory state, r_k, at each step to be used later for the linear fit.

The code was developed in Visual Studio (VS) Code in a Jupyter Notebook (.ipynb) file. Version control was handled using a GitHub repository to keep track of changes.

```
J_couplings = [(i, random.randint(i+1, N), random.rand()*4-2) for i in range(N-1)]
Hj = [Jij * couple_operator(sz, i, sz, j, N) for (i, j, Jij) in J_couplings]
Hj = sum(Hj)

Hd = sum(site_operator(sx, i, N) for i in range(N))

w_input = random.rand(N)*4-2
Hin = sum(w_input[i] * site_operator(sz, i, N) for i in range(N))

H = Hj + Hd + input_function * Hin
```

Listing 1: Python code for defining the transverse field Ising Hamiltonian for a quantum reservoir.

4 Results

Our first task was to test if our model's architecture has a significant predictive capability beyond random chance, and if so, how much predictive capability. An ideal function to test this capability is the Mackey-Glass delay differential equation defined in Equation 3. We set the parameters as so, $\beta = 0.2$, $\gamma = 0.1$, time delay $\tau = 17$, time-step $\delta t = 3.0$, and the non-linearity parameter $n = 10$. A time series of 2000 steps was produced and was split into training and evaluation steps. The features were generated for both sections of the time series (this can be done before training, since the reservoir weights are not altered in training). The linear fit is then trained on steps between 1-1880 and the evaluation was done on steps 1890-1990. These ranges leave room for the predictions to trained on and evaluated against the true values 10 steps ahead. Figure 4 shows clear predictive capabilities with a small MSE of 0.0039. This reservoir consisted of 10 qubits with the split cyclic permutation transformation from Equation 10.

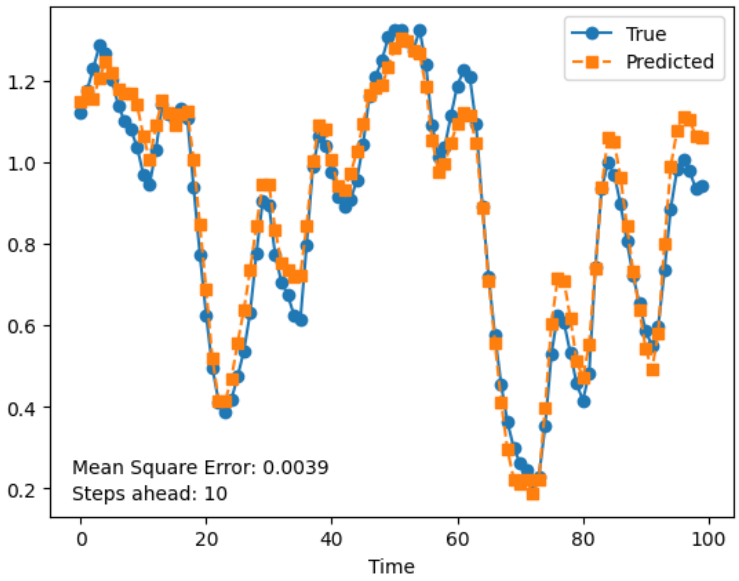

Figure 4: An time series prediction of a Mackey-Glass function for 10 steps ahead for a 10-qubit reservoir. Orange showing the prediction and blue showing the true value of the function.

Next we would like to determine the if the split permutation transformation has any advantages over the existing technique used by Settino et al. [7]. For this section of the project, we trained and evaluated our architecture with the split permutation technique and with the no-split permutation technique, measuring the mean square error as a function of the number of qubit used. Each point in Figure 5 is an average of 20 instances of the specified reservoir parameters trained on the same time series as shown in Figure 4. The uncertainty is shown with error bars. The plot clearly shows reduced MSE when using the split permutation method for a lower number of qubits with the no-split permutation method performing just as well if not better than the split permutation method.

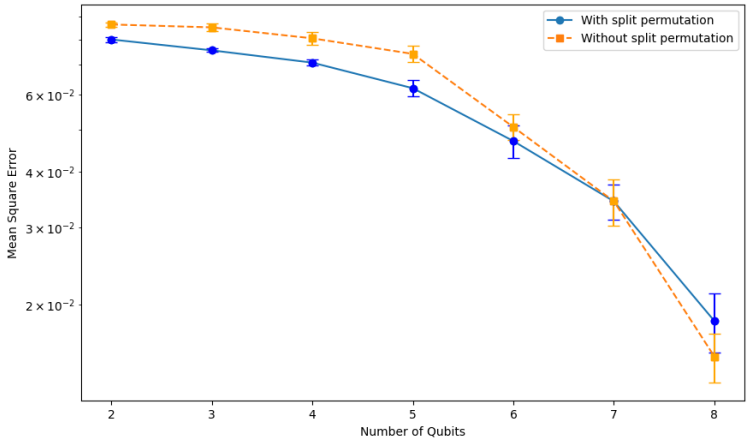

Figure 5: A comparison of MSE for two time encoding techniques for the classical memory state. The first (blue) using a split cyclic permutation method, and the second (orange) using a cyclic permutation over all the features together.

To further test the predictive capabilities of the architecture we have created, we would like to evaluate the model's predictive capabilities for varying steps ahead of the input. For the previous evaluations, we have trained the linear fit to predict 10 steps ahead of the input for each step. For the following section, we will evaluate the MSE for varying steps ahead and compare the results for differing numbers of qubits. A plot of the results is shown in Figure 6.

The results are initially what is expected with low error at the low steps ahead and an increase in error as the steps ahead increases, especially for a low number qubits. However, we see a decrease in error once the steps ahead move closer to the the value of τ that was used in generating the Mackey-Glass input function.

Next, to demonstrate the advantages of the extra features provided by quantum systems with an exponentially increasing amount of observables with the number of qubits used, we generated two prediction with an 8 qubit quantum reservoir, one using the co-occurrence observables and the other not. The predictions are shown in Figure 7. The results clearly show an advantage in predictive capability for the increased feature space provided with the co-occurrence operators.

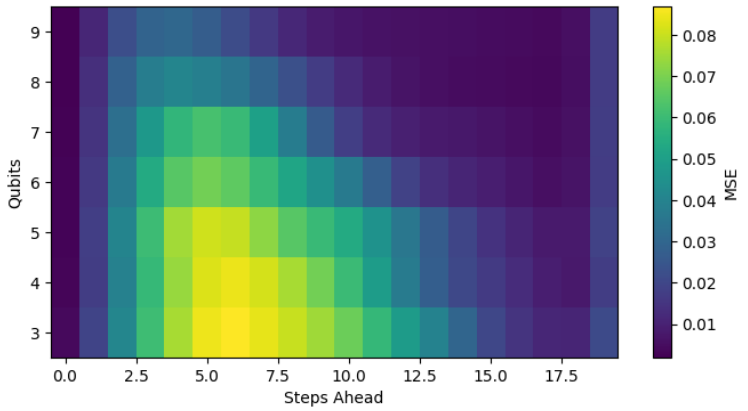

Figure 6: A heatmap showing the MSE of predictions of a Mackey-Glass time series as a function of number of qubits and number of steps ahead being predicted.

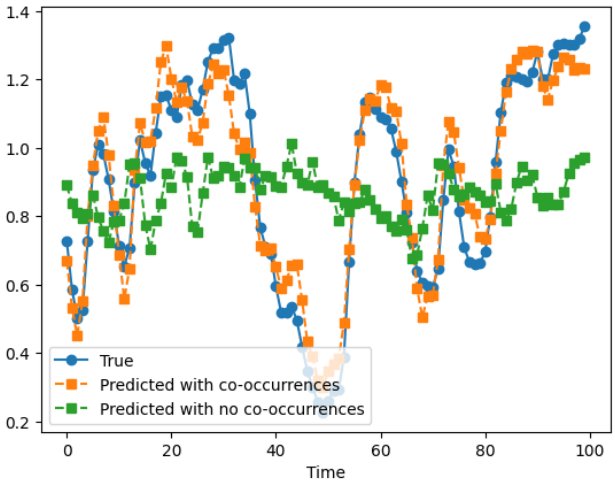

Figure 7: A plot of two predictions using a quantum reservoir computer, the first (orange) using both qubit spin expectation values and co-occurrence expectation values as features, the second (green) only using the qubit spin expectation values as features. Done for a 10-step-ahead prediction with 8 qubits.

5 Discussion

The results support our aim to understand and test the feasibility of creating a quantum reservoir computer using Python and QuTiP. The predictive capabilities go beyond our initial expectation with the low error in prediction of a Mackey-Glass time series. The Mackey-Glass time series is historically hard to predict, therefore Figure 4 demonstrates the potential of QRCs to act as time series predictors. It is important to recognise that while we simulated a small number of qubits, a real quantum annealer is able to support many more qubits with ease running thousands of times faster than even our small simulation.

The improved results with the split-permutation method at small N_q is likely due to the lack of dimensionality with less qubits and therefore the overlap of the single-site and co-occurrence observables in memory adds noise to the memory state that overwrite the simpler features. However, when the number of qubits is increased the memory is large enough to support this increased noise and therefore support an increase in the richness of the feature-space. This finding suggests a trade-off between feature richness and noise in the classical reservoir memory state. This further supports the idea that with a greater number of qubits available in a real quantum annealer, the potential for increased expressivity is feasible.

The results in Figure 6 are expected in hindsight, given that the Mackey-Glass function is a function of itself τ steps prior, it then make sense that the QRC would have improved predictive capabilities for predicting τ steps ahead due to the inputs influence over the function at that exact amount of steps ahead. These findings demonstrate that QRCs are able to predict certain time series' with better accuracy for more steps ahead if the time series follows a natural periodicity. For example, the weather would likely be able to be predicted best seconds or minutes ahead of the input, but it would also follow a natural 24 hour periodicity that would allow for prediction of 24 hours ahead to have a natural advantage due to its similarity with the current input values. This also highlights a limitation the duration and proper encoding of the classical memory. With an ideal memory, the linear fit would be able to tap into the information provided by previous inputs that best align with the number of steps ahead trying to be predicted. This outlines the potential for improved memory that is able to fade more slowly without interfering adding too much noise to the memory state. One potential solution would be to map the measured observables to a higher dimensional classical memory that has more space to encode temporal information. Another potential improvement would be to have the memory decay newer information quicker than older information instead of the linear decay in the current implementation. This would allow the current input to stand out from noisy memory while still retaining a longer context.

Although a real quantum annealer would allow for increased qubits and likely an increased expressiveness, there would also be added noise and decoherence in a real world setup that did not exist in our simulation. This is why it is extremely important to test QRCs with real quantum systems, to test both the effects of decoherence and scalability of these systems. With that said, there are still many aspects of QRCs that can be tested with simulations. In our project, we used very basic encoding techniques, but effective encoding of the input time series into the Hamiltonain is what allows for the quantum advantage over classical systems [8]. This is because to propagate the input through the Hilbert-space of the system the input must meaningfully alter the quantum state, not just shift it trivially. The way this information is injected determines how deeply the system explores its Hilbert space, which in turn affects how complex the resulting state becomes and therefore determines the richness of the observables measured. On the Bloch sphere, this can be visualised as how much space the state covers. A well-designed encoding scheme causes the qubit states to trace rich, non-linear paths across the sphere, covering a larger portion of the state space and generating more diverse correlations between qubits [10].

6 Scientific Mastery

In this project I have been required to develop the Python code to implement a quantum reservoir with my own framework for generating the transverse field Ising Hamiltonian for an arbitrary number of qubits and any choice of couplings, as shown in Figure 1. On top of this I have developed the code to generate all the required objects involved in a quantum reservoir, such as the weightings for the input function, the individual terms of the Hamiltonian, and the measurement operators. All of which can be generated for any sized system. The code can be seen in Listing 2

```python
def build_reservoir(N=6,
                    seed=13):

    random.seed(seed)
    input_sites = tuple(range(N))
    J_couplings = [(i, random.randint(i+1, N), random.rand()*4-2) for i in range(N-1)]
    h_local = [random.rand()*4-2 for _ in range(N)]

    Hp = generate_ising_Hamiltonian(N, h_local, J_couplings)
    Hd = sum(site_operator(sx, i, N) for i in range(N))

    w_input = random.rand(N)*4-2
    Hin = sum(w_input[i] * site_operator(sz, i, N) for i in range(N))

    measure_ops = []
    for i in range(N):
        op = site_operator(sz, i, N)
        measure_ops.append(op)
        for j in range(N-i-1):
            op = couple_operator(sz, i, sz, (i+j+1), N)
            measure_ops.append(op)

    return {...}
```

Listing 2: Python code for building the quantum reservoir for a reservoir computer, including the different Hamiltonian terms, measurement operators, and input weights.

Beyond this I had to work through many stages of the reservoir having little to no predictive capability, requiring me to read through many research papers on the topic and not only implementing a classical memory that has been done before, but improving on it with my split-operator permutation method shown in Equation 10. The code again for this was developed with no reference and can be seen in Listing 3.

```python
def generate_features(input_function, Nq, T=1.0, seed=13, split_permutations=True):
    res = build_reservoir(Nq, seed=seed)
    Hp, Hd, Hin, measure_ops = res['Hp'], res['Hd'], res['Hin'], res['measure_ops']
    N = len(measure_ops)
    psi = Hd.groundstate()[1]
    rk = np.zeros(N)
    rk_list = []
    gamma = 0.8
    for i, input_value in enumerate(input_function):
        H = -(0.5 * Hp + 0.01 * Hd + 1.0 * Hin * input_value)
        mk = drive_Hamiltonian(H, psi, measure_ops, T=T)
        if split_permutations:
            rk_qb = rk[:Nq]
            rk_j = rk[Nq:]
            rk_qb = np.roll(rk_qb, 1)
            rk_j = np.roll(rk_j, 1)
            rk = np.concatenate((rk_qb, rk_j))
        else:
            rk = np.roll(rk, 1)
        rk = mk + gamma * rk
        rk_list.append(rk)
    return rk_list
```

Listing 3: Python code for driving the reservoir with the input function and implementing split-operator permutations to evolve a classical memory.

7 Conclusion

In this project we successfully demonstrated the feasibility of simulating a quantum reservoir computer, demonstrating that even a small quantum system can produce accurate time-series predictions. The quantum reservoir framework with a classical memory state we developed in Python and QuTiP captured temporal features from the Mackey–Glass equation with impressive predictive capabilities, supporting the idea that quantum systems can serve as effective reservoirs with the extra features in. The split-permutation transformation method had performance benefits for systems with a smaller number of qubits, as separating the single-site and co-occurrence observables helped to reduce noise in the classical memory state. This revealed insight into the important balance between feature diversity and noise, suggesting that there is an optimal level of complexity based on the size of the reservoir. As the number of qubits increases, this balance will likely shift, leading to new behaviour that needs to be tested on real quantum hardware. A big focus of this project was how we encoded the classical memory and evolved it over time, however, further gains in performance and scaling is likely to be found in how information is encoded into the Hamiltonian. A long non-linear trajectory of the qubits through the Hilbert space can likely corresponds to greater expressivity and stronger predictive capability. This exploration of the Hilbert space is what allows for potential advantage of QRCs over classical systems. Therefore, future work should explore more advanced encoding methods and the use of real quantum annealers to experiment with a larger number of qubits. Overall, this project demonstrates that quantum reservoir computing is not only theoretically sound but also practically achievable even with simulation of small networks, supporting the idea of a path toward real-world quantum predictive systems.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory", Neural Computation **9**, 1735–1780 (1997).

[2] *A quick start to reservoirpy*, ReservoirPy, `https://reservoirpy.readthedocs.io/en/latest/user_guide/quickstart.html` (visited on 10/18/2025).

[3] 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 Information **9**, `10.1038/s41534-023-00682-z` (2023).

[4] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: introduction and review", Reports on Progress in Physics **85**, 104001 (2022).

[5] Q. Li, C. Mukhopadhyay, A. Bayat, and A. Habibnia, *Quantum reservoir computing for realized volatility forecasting*, 2025.

[6] K. Kobayashi, K. Fujii, and N. Yamamoto, "Feedback-driven quantum reservoir computing for time-series analysis", PRX Quantum **5**, 10.1103/prxquantum.5.040325 (2024).

[7] J. Settino, L. Salatino, L. Mariani, M. Channab, L. Bozzolo, S. Vallisa, P. Barillà, A. Policicchio, N. L. Gullo, A. Giordano, C. Mastroianni, and F. Plastina, *Memory-augmented hybrid quantum reservoir computing*, 2024.

[8] N.-E. Schütte, N. Götting, H. Müntinga, M. List, D. Brunner, and C. Gies, *Expressivity limits of quantum reservoir computing*, 2025.

[9] M. Kornjača, H.-Y. Hu, C. Zhao, J. Wurtz, P. Weinberg, M. Hamdan, A. Zhdanov, S. H. Cantu, H. Zhou, R. A. Bravo, K. Bagnall, J. I. Basham, J. Campo, A. Choukri, R. DeAngelo, P. Frederick, D. Haines, J. Hammett, N. Hsu, M.-G. Hu, F. Huber, P. N. Jepsen, N. Jia, T. Karolyshyn, M. Kwon, J. Long, J. Lopatin, A. Lukin, T. Macrì, O. Marković, L. A. Martínez-Martínez, X. Meng, E. Ostroumov, D. Paquette, J. Robinson, P. S. Rodriguez, A. Singh, N. Sinha, H. Thoreen, N. Wan, D. Waxman-Lenz, T. Wong, K.-H. Wu, P. L. S. Lopes, Y. Boger, N. Gemelke, T. Kitagawa, A. Keesling, X. Gao, A. Bylinskii, S. F. Yelin, F. Liu, and S.-T. Wang, *Large-scale quantum reservoir learning with an analog quantum computer*, 2024.

[10] A. Kutvonen, T. Sagawa, and K. Fujii, *Optimizing a quantum reservoir computer for time series prediction*, 2020.