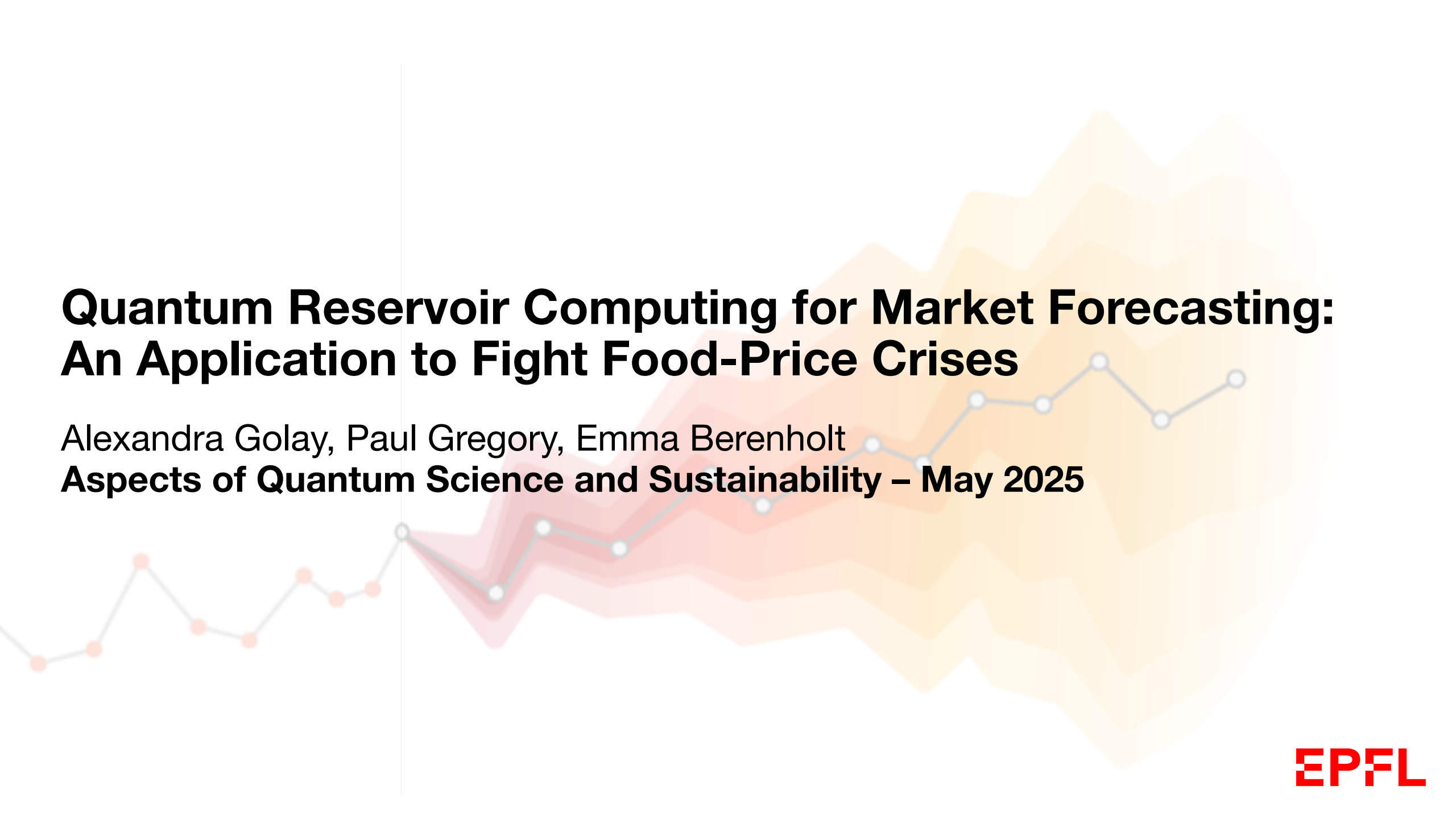
Quantum Reservoir Computing for Market Forecasting: An Application to Fight Food-Price Crises

Alexandra Golay, Paul Gregory, Emma Berenholt

Aspects of Quantum Science and Sustainability – May 2025



Outline

1. The Sustainability Challenge
2. QRC Pipeline: Workflow and Mathematical Formulation
3. Implementation and Analysis: The Tomato Case
4. Comparisons and Limitations of the QRC

Introduction

The Sustainability Challenge

- Food price spikes → insecurity, malnutrition, reduced access to health & education
- Climate change disrupts crops (ex: heatwaves → lower tomato yields → price surge)
- Poor households are hit hardest

Forecasting = key for early action & sustainable food systems

Why Forecasting Matters

- Enables **early action**: better planning for imports, aid, and farming decisions
- Reduces **market volatility** and **food waste** by aligning supply with demand
- Contributions to the SDGs

EPFL

Nigeria hit by wave of food looting as economic crisis deepens

Ransacking of warehouses and deadly stampede at grain sale spark fears of breakdown in law and order

Nigerians protesting in Abuja last month over the rising cost of living as food price inflation jumps above 35% © AFP via Getty Images

Food price inflation in Nigeria is more than 35%

Annual food CPI change (%)

Source: LSEG

- Food prices rose by more than **35%**; rice price **doubled** in one year
- People looted food warehouses; **7 people died** during a rice sale
- Fear of more protests and a breakdown of public order
- Farming is disrupted by **violence** and **kidnappings**; some farmers had to pay to use their own land
- **8%** of Nigerians are food insecure (IMF)
- Accusations of food hoarding led to **warehouses being taken over**

= Forecasting food prices helps act early, send food aid, and avoid social crisis

EPFL

Contributions to the SDGs

SDG 1 – No Poverty:
Food affordability is a central component of poverty. Early identification of price stress points can inform cash-transfer programs and price stabilization policies.

SDG 2 – Zero Hunger:
Anticipating price volatility allows for proactive food security strategies, ensuring that populations maintain access to sufficient and nutritious food.

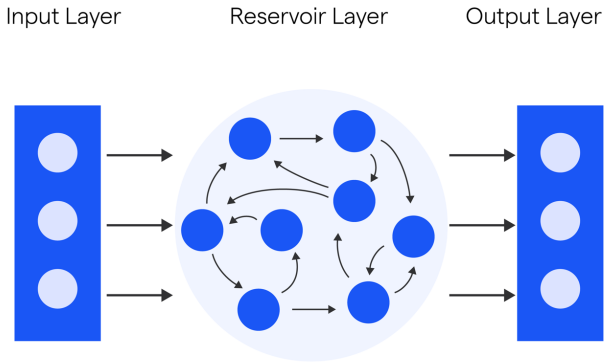
SDG 12 – Responsible Consumption and Production:
Accurate demand forecasting reduces inefficiencies across the food supply chain and lowers environmental impact.

SDG 13 – Climate Action:
Incorporating climate data into price models provides better insight into how global warming affects food systems, supporting adaptation policies and long-term resilience planning.

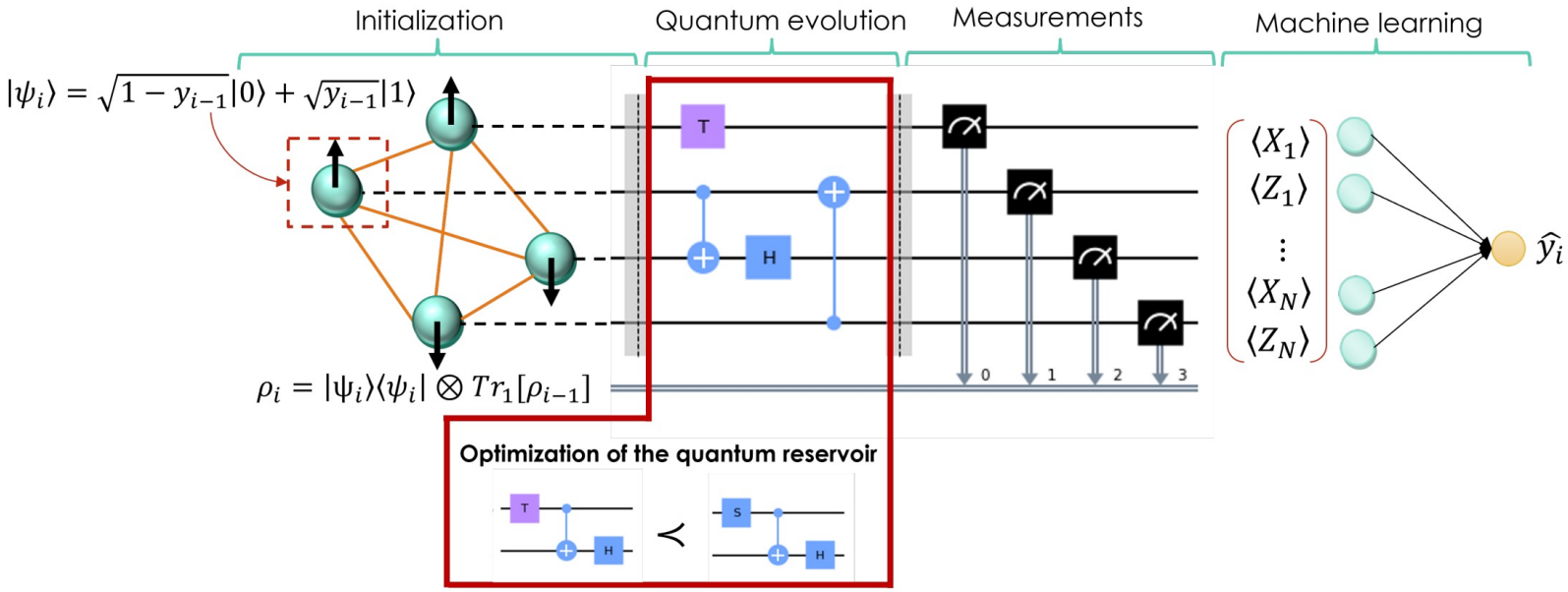
EPFL

QRC Pipeline Overview

Let $\{y(t)\}_t$ a n–dimensional time series. Given the past values of the series $\{y(\tau)\}_{\tau \leq t}$, the goal is to predict the value of $y(t + \Delta t)$.



Abstract Implementation of a general RC



Concrete Implementation of a QRC

EPFL

Step 1: Input Encoding

Single-input: n = 1

$(i)\ y(t) \in \mathbb{R} \rightarrow \tilde{y}(t) \in [0,1]$ (Normalization)

$(ii)\ |y(t)\rangle = \sqrt{1-\tilde{y}(t)}|0\rangle + \sqrt{\tilde{y}(t)}|1\rangle$ (Encoding)

Where (ii) can be done using a $R_Y(\Theta)$ gate:

$R_Y(2\arcsin\tilde{y}(t)|0\rangle = \sqrt{1-\tilde{y}(t)}|0\rangle + \sqrt{\tilde{y}(t)}|1\rangle$

(Multi-input: n > 1)

$(i)\ y(t) \in \mathbb{R}^n \rightarrow \tilde{y}(t) \in [0,1]^n$

$(ii)\ |y(t)\rangle = \otimes_{i=0}^{n-1}(\sqrt{1-\tilde{y}_i(t)}|0\rangle + \sqrt{\tilde{y}_i(t)}|1\rangle)$

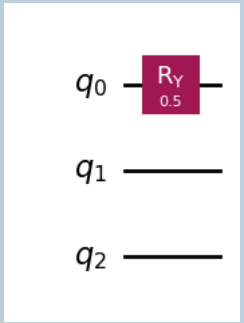
Example.

$y(t) = \sin(t)$ $t = 0,1,2, \ldots$ (as rad.)

$\sin(t) \in \mathbb{R} \rightarrow$ single-input case (n=1)

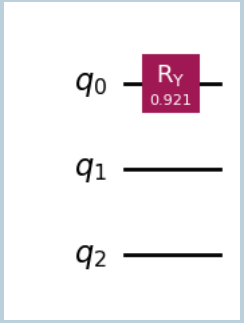
$$\tilde{y}(t) = \frac{sin(t)+1}{2} \in [0,1]$$

QR with **N = 3 qubits** (n=1 input qubit and N-n=2 memory qubits)



$|y(0)\rangle = \sqrt{1-\tilde{y}(0)}|0\rangle + \sqrt{\tilde{y}(0)}|1\rangle$

$= R_Y(2\arcsin\tilde{y}(0)|0\rangle$



$|y(1)\rangle = \sqrt{1-\tilde{y}(1)}|0\rangle + \sqrt{\tilde{y}(1)}|1\rangle$

$= R_Y(2\arcsin\tilde{y}(1)|0\rangle$

...

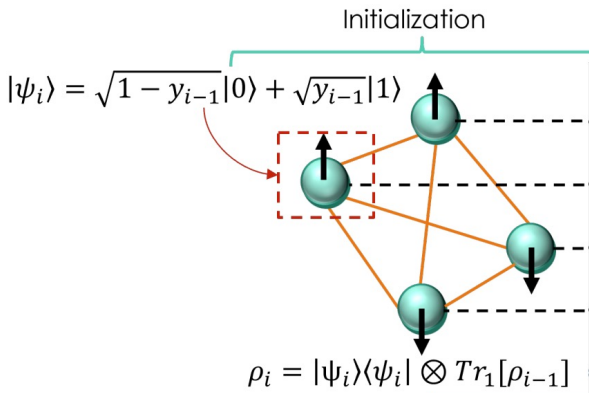
EPFL

Step 2: Reservoir State Initialization

The full reservoir at time *t* incorporates both the new input and the memory of past ones:

$$\rho(t) = |y(t)\rangle\langle y(t)| \otimes Tr_n(\rho(t - \Delta t))$$

Where $Tr_n(\rho(t - \Delta t))$ is the partial trace over the *n* input qubits, isolating the memory stored in the remaining *N-n* qubits.



Example.

$$\rho(0) = |y(0)\rangle\langle y(0)| \otimes \rho_{init}$$

$$\rho(1) = |y(1)\rangle\langle y(1)| \otimes Tr_0(\rho(0))$$

$$\rho(2) = |y(2)\rangle\langle y(2)| \otimes Tr_0(\rho(1))$$

...

Where ρ_{init} can be chosen randomly (**ESP property**), e.g., let's say $\rho_{init} = |00\rangle\langle 00|$

$$\mathrm{Tr}_0(\rho(0)) = \begin{bmatrix} 0.25 & 0.114 + 0.037i & -0.028 + 0.248i & 0.049 - 0.109i \\ 0.114 - 0.037i & 0.25 & 0.023 + 0.118i & 0.028 - 0.248i \\ -0.028 - 0.248i & 0.023 - 0.118i & 0.25 & -0.114 + 0.037i \\ 0.049 + 0.109i & 0.028 + 0.248i & -0.114 + 0.037i & 0.25 \end{bmatrix}$$

...

EPFL

Step 3: Quantum Evolution via fixed *U*

Next, the entire *N*-qubit system undergoes a fixed quantum evolution:

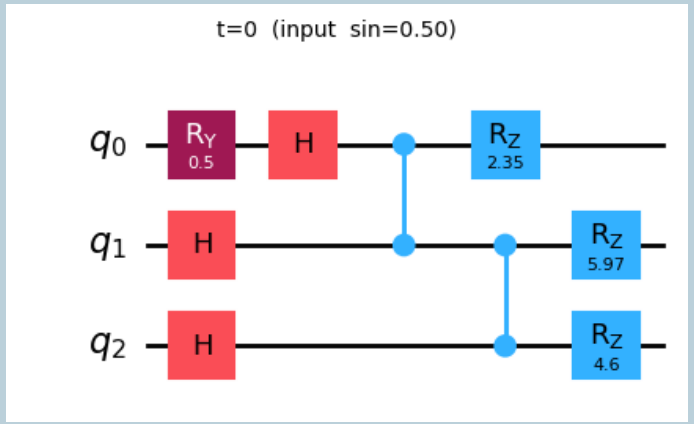
$$\rho'(t) = U\rho(t)U^{\dagger}$$

The unitary operator U is sampled randomly once at initialization and remains fixed for all time steps.

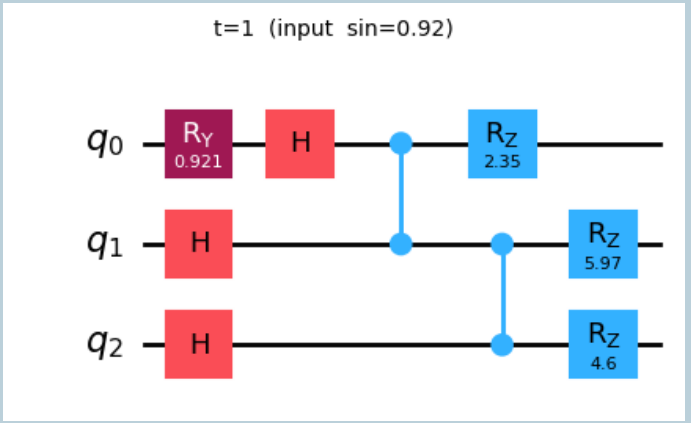
Common choice: random quantum circuit built from the universal gate set: $\mathrm{G} = \{CNOT, H, T\}$

Example.

$\mathrm{t} = 0$:



$\mathrm{t} = 1$:



...

Step 4: Measurement and classical feature extraction

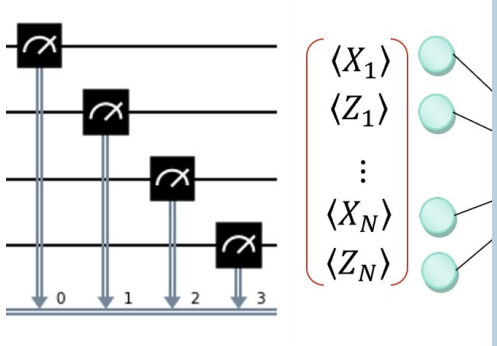
At the end of each time step, the quantum state $\rho'(t)$ is measured to extract classical information that will serve as features for the prediction.

In this work we compute the expectation values of the Pauli *X* and *Z* operators on each qubit:

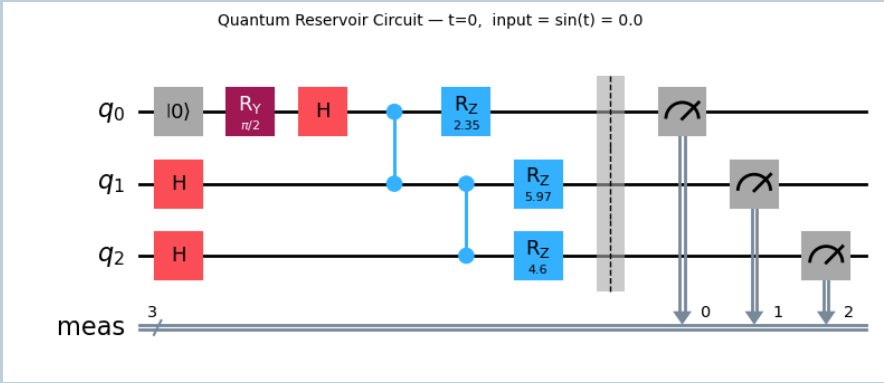
$$\langle P_i \rangle(t) = Tr[P_i \rho'(t)], \quad P_i \in \{X_i, Z_i\}$$

These measurements yield a classical feature vector:

$$\vec{f}(t) = \begin{pmatrix} \langle X_1 \rangle \\ \langle Z_1 \rangle \\ \vdots \\ \langle X_N \rangle \\ \langle Z_N \rangle \end{pmatrix} \in \mathbb{R}^{2N}$$



Example.



Feature vectors f(t):

	<X0>	<Z0>	<X1>	<Z1>	<X2>	<Z2>
t=0	-6.162976e-33	1.000000	0.000000	0.000000e+00	4.163336e-17	5.551115e-17
t=1	0.000000e+00	0.246451	-0.097575	5.551115e-17	-7.334021e-02	-5.551115e-17
t=2	2.763182e-01	0.141994	0.008456	3.959186e-01	4.326006e-01	-7.334021e-02
t=3	-9.231335e-03	0.975531	0.953252	-5.955142e-02	8.035054e-01	4.326006e-01
t=4	-6.392955e-01	0.372790	0.020097	9.771621e-01	3.013385e-01	8.035054e-01
t=5	-3.792994e-02	0.064477	0.022641	5.390888e-02	2.090132e-01	3.013385e-01
t=6	1.052087e-01	0.905218	0.549807	-3.511470e-01	1.957629e-01	2.090132e-01
t=7	3.675652e-01	0.513110	-0.021380	6.073754e-01	-5.781106e-01	1.957629e-01
t=8	-2.937448e-02	0.016715	-0.013559	-4.166817e-02	6.337733e-01	-5.781106e-01
t=9	-3.449099e-01	0.797682	-0.126191	-8.111583e-01	3.612699e-01	6.337733e-01
t=10	8.413024e-02	0.656561	-0.251054	-1.581965e-01	1.307226e-01	3.612699e-01
t=11	2.695989e-01	0.000015	0.000009	-3.823769e-01	4.260884e-01	1.307226e-01
t=12	-3.207709e-01	0.665340	-0.065628	6.094180e-01	-5.815577e-01	4.260884e-01

...

Step 5: Linear Regression

At each time t, the QR outputs a feature vector $\vec{f}(t) \in \mathbb{R}^{2N}$ built from expectation values of Pauli observables like $\langle X_i \rangle, \langle Z_i \rangle$.

We want to predict the future value $y(t + \Delta t)$ using:

$$\hat{y}(t + \Delta t) = \vec{w}^T \vec{f}(t) + b$$

To do so, we define the matrix X and the target vector $\vec{y}$ (the values we want to predict)

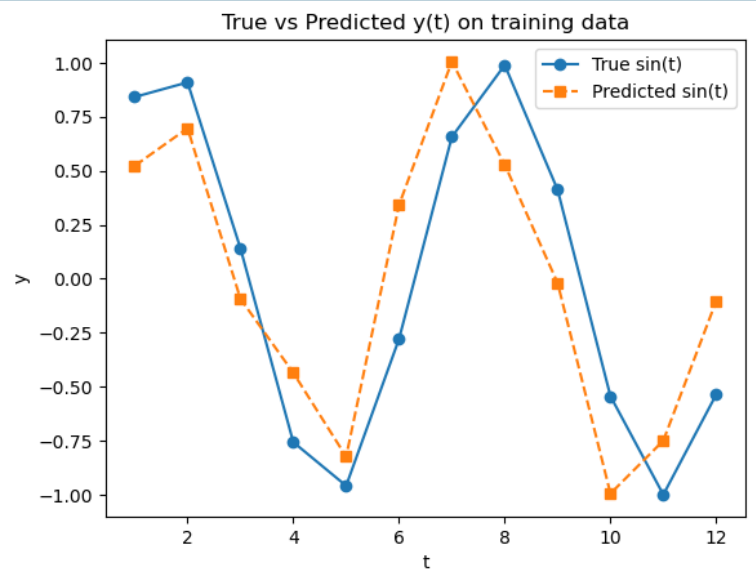
$$X = \begin{pmatrix} \vec{f}(0)^T & 1 \\ \vdots & \vdots \\ \vec{f}(T-1)^T & 1 \end{pmatrix} \qquad \vec{y} = \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(T) \end{pmatrix}$$

Note: If input runs from t=0 to T-1, targets go from t=1 to T.

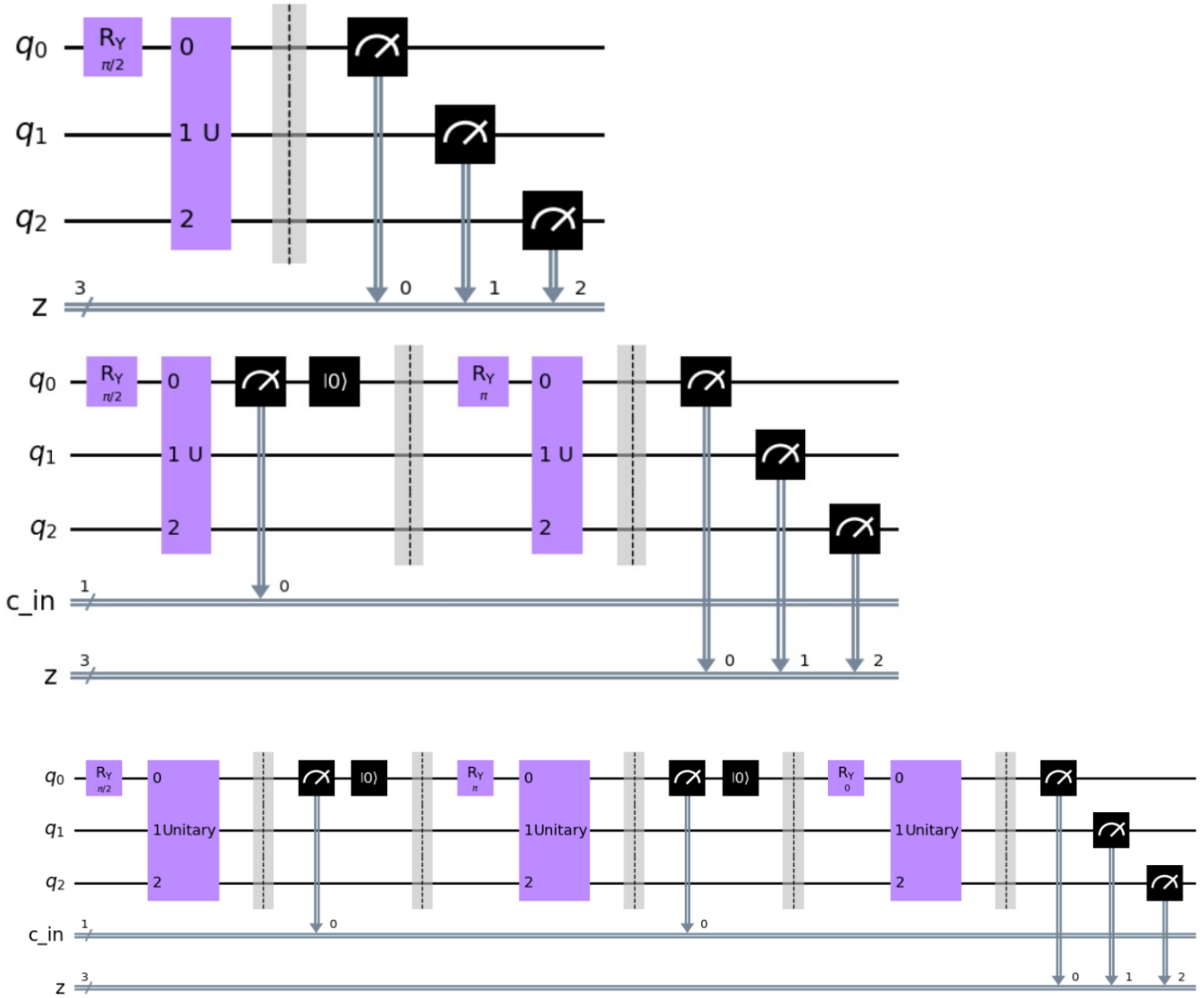
We now solve the LSM:

$$\hat{\theta} = argmin_\theta \, \|X\theta - \vec{y}\|^2 \;\rightarrow\; \theta = \begin{pmatrix} \vec{w} \\ b \end{pmatrix} = (X^T X)^{-1} X^T \vec{y}$$

Example.



Circuit Implementation with 3 qubits for the feature generation of a serie for 3 timesteps



The density matrix of each state at the end of the 3 circuits will be

1. $\rho(0) = |y(0)\rangle\langle y(0)| \otimes \rho_{init}$
2. $\rho(1) = |y(1)\rangle\langle y(1)| \otimes Tr_0(\rho(0))$
3. $\rho(2) = |y(2)\rangle\langle y(2)| \otimes Tr_0(\rho(1))$

Computation of the features :

$$f_i(t) = \langle Z_i \rangle_t = \frac{n_{0,i}(t) - n_{1,i}(t)}{S}, \qquad i \in \{0, 1, 2\}$$

For each circuit, we'll get a vector of features :

$$\mathbf{f}(t) = \left(f_0(t),\ f_1(t),\ f_2(t)\right)^\top \in \mathbb{R}^N, \qquad N = 3.$$

For T timesteps, the feature matrix will be :

$$\mathbf{F}_{\text{train}} = \begin{bmatrix} \mathbf{f}(0)^\top \\ \mathbf{f}(1)^\top \\ \vdots \\ \mathbf{f}(T-1)^\top \end{bmatrix} \in \mathbb{R}^{T \times N}, \qquad \mathbf{y}_{\text{train}} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(T) \end{bmatrix} \in \mathbb{R}^{T}.$$

In our example with T = 3 and 3 qubits,

$$\mathbf{F}_{\text{train}} = \begin{bmatrix} f_0(0) & f_1(0) & f_2(0) \\ f_0(1) & f_1(1) & f_2(1) \\ f_0(2) & f_1(2) & f_2(2) \end{bmatrix}, \qquad \mathbf{y}_{\text{train}} = \begin{bmatrix} y(1) \\ y(2) \\ y(3) \end{bmatrix}.$$

Training Weights :

We learn the weights by minimizing the regularized mean square error :

$$J(\mathbf{W}, b) = \frac{1}{T} \left\| \mathbf{y}_{\text{train}} - \mathbf{F}_{\text{train}}\mathbf{W} - b\mathbf{1}_T \right\|_2^2 + \alpha \left\| \mathbf{W} \right\|_2^2, \quad \alpha > 0$$

The closed-form solution of this equation is, solved by scikit learn in our implementation:

$$\mathbf{W}^* = \left(\mathbf{F}_{\text{train}}^\top \mathbf{F}_{\text{train}} + T\alpha \mathbf{I}_N \right)^{-1} \mathbf{F}_{\text{train}}^\top \tilde{\mathbf{y}}$$

Where $\tilde{\mathbf{y}} = \mathbf{y}_{\text{train}} - b\mathbf{1}_T$

Test phase

The feature matrix of the test set will be :

$$\mathbf{F}_{\text{test}} = \begin{bmatrix} \mathbf{f}_{\text{test}}(T)^\top \\ \mathbf{f}_{\text{test}}(T+1)^\top \\ \vdots \\ \mathbf{f}_{\text{test}}(T_{\text{tot}}-1)^\top \end{bmatrix} \in \mathbb{R}^{T_{\text{test}} \times N}, \quad \mathbf{y}_{\text{test}} = \begin{bmatrix} y(T+1) \\ y(T+2) \\ \vdots \\ y(T_{\text{tot}}) \end{bmatrix}$$

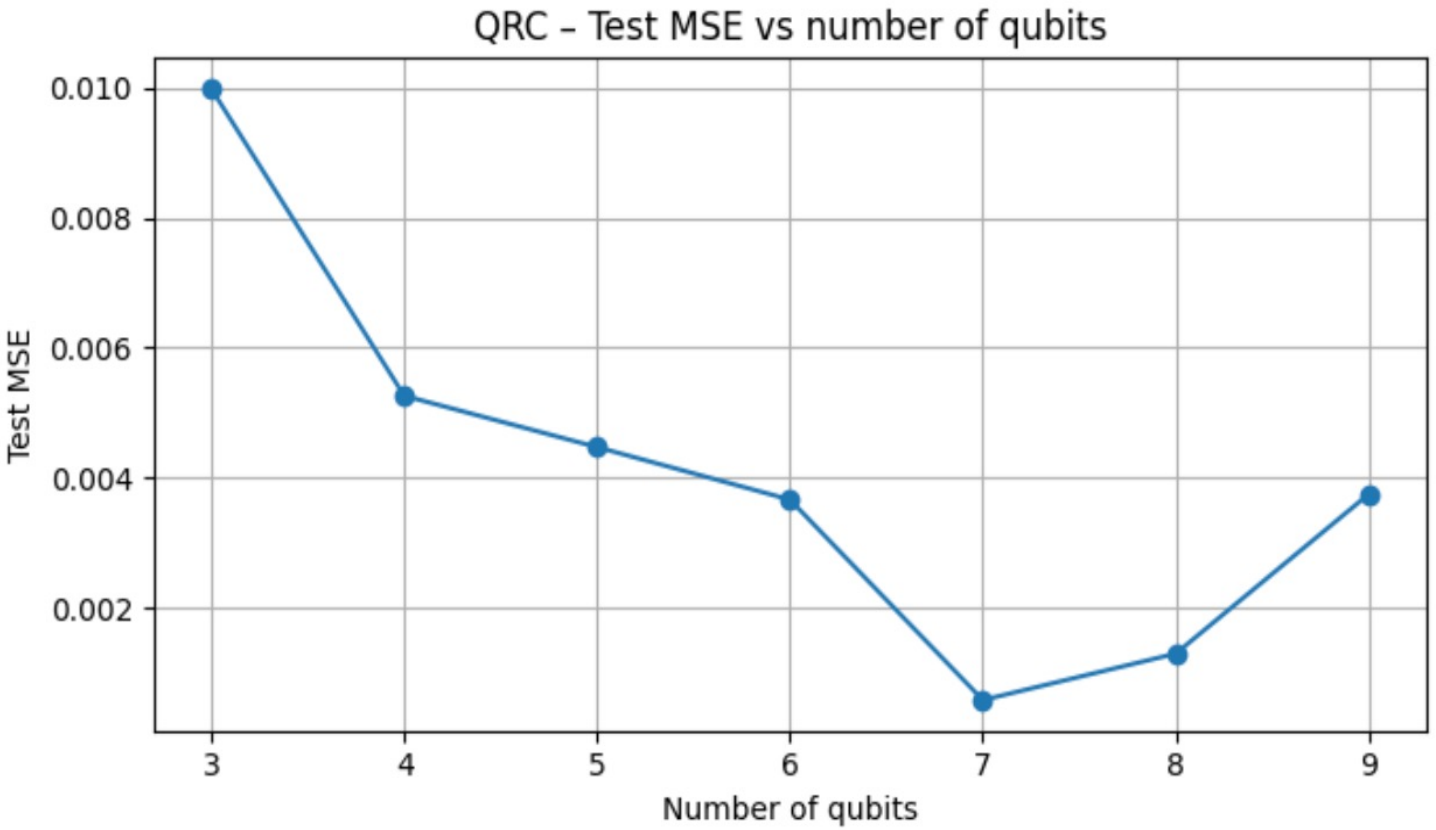
Using the optimized weigths and bias learnt on training, the one step forecast is :

$$\widehat{\mathbf{y}}_{\text{test}} = \mathbf{F}_{\text{test}} \mathbf{W}^\star + b^\star \mathbf{1}_{T_{\text{test}}} \in \mathbb{R}^{T_{\text{test}}}.$$

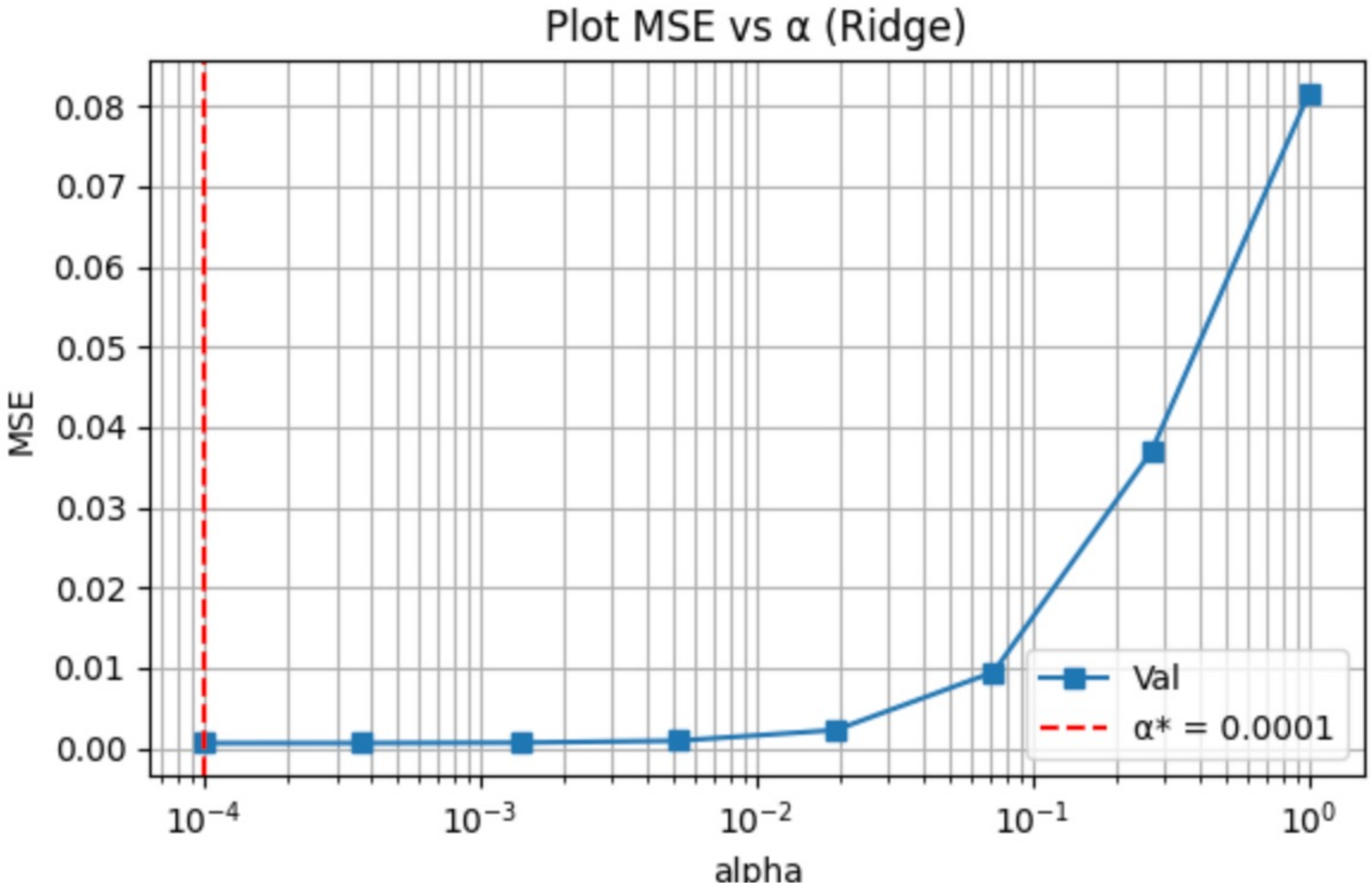
In order to evaluate the performance or our model, we will compute the MSE of the test set as follow :

$$\text{MSE}_{\text{test}} = \frac{1}{T_{\text{test}}} \left\| \mathbf{y}_{\text{test}} - \widehat{\mathbf{y}}_{\text{test}} \right\|_2^2.$$

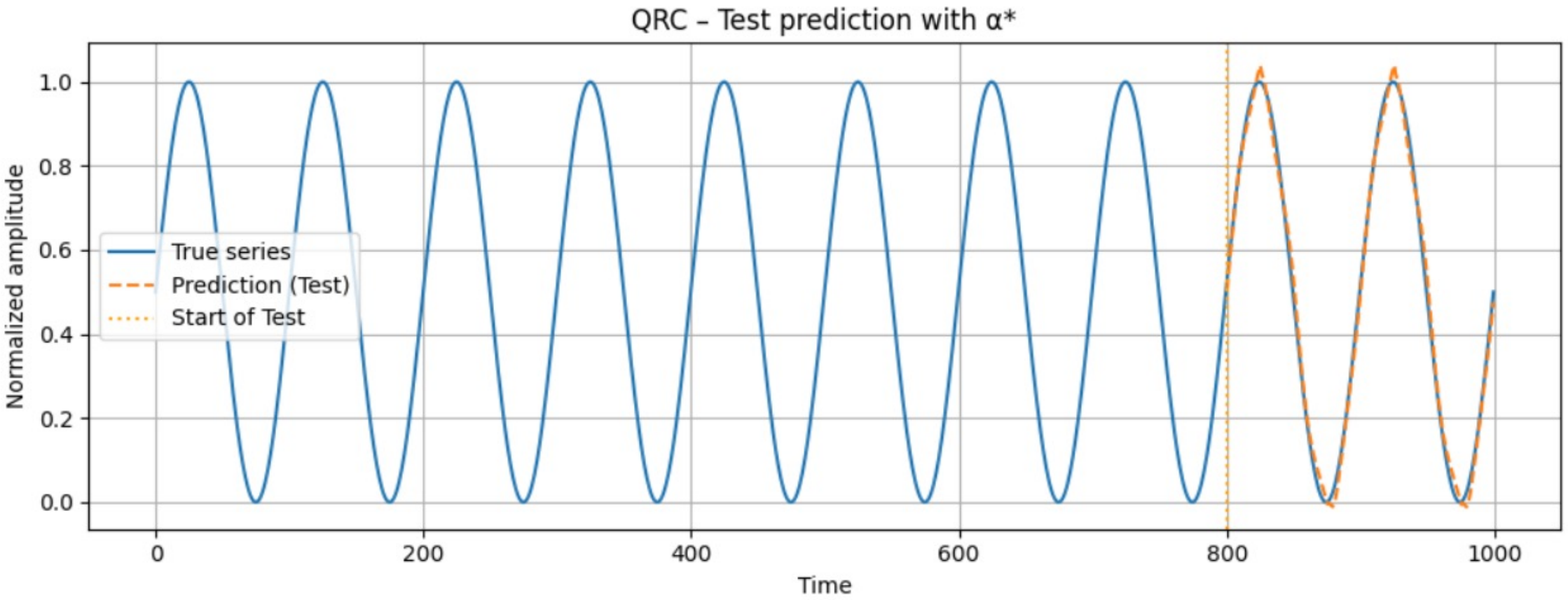
Changing the number of qubits in our reservoir



As the number of qubits increases, the quantum reservoir generates more features (one per qubit). This initially reduces the test MSE, as the model captures richer temporal dynamics. However, beyond a certain point, the increased number of features leads to overfitting.

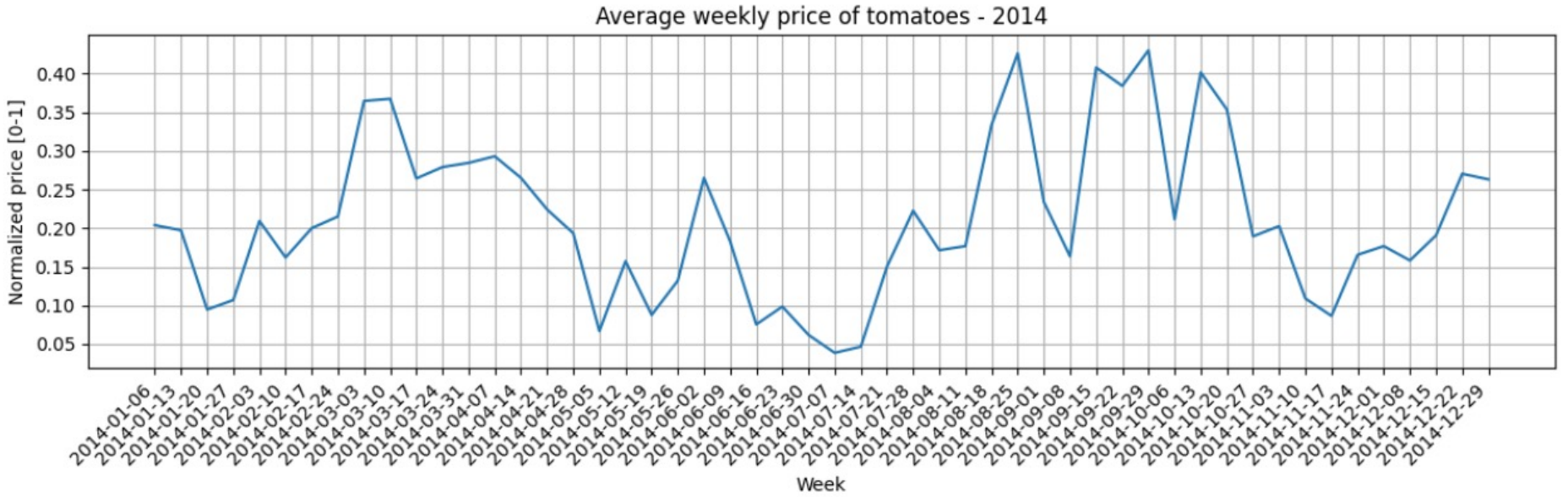

Plot MSE vs α (Ridge)
MSE
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
alpha
Val
α* = 0.0001

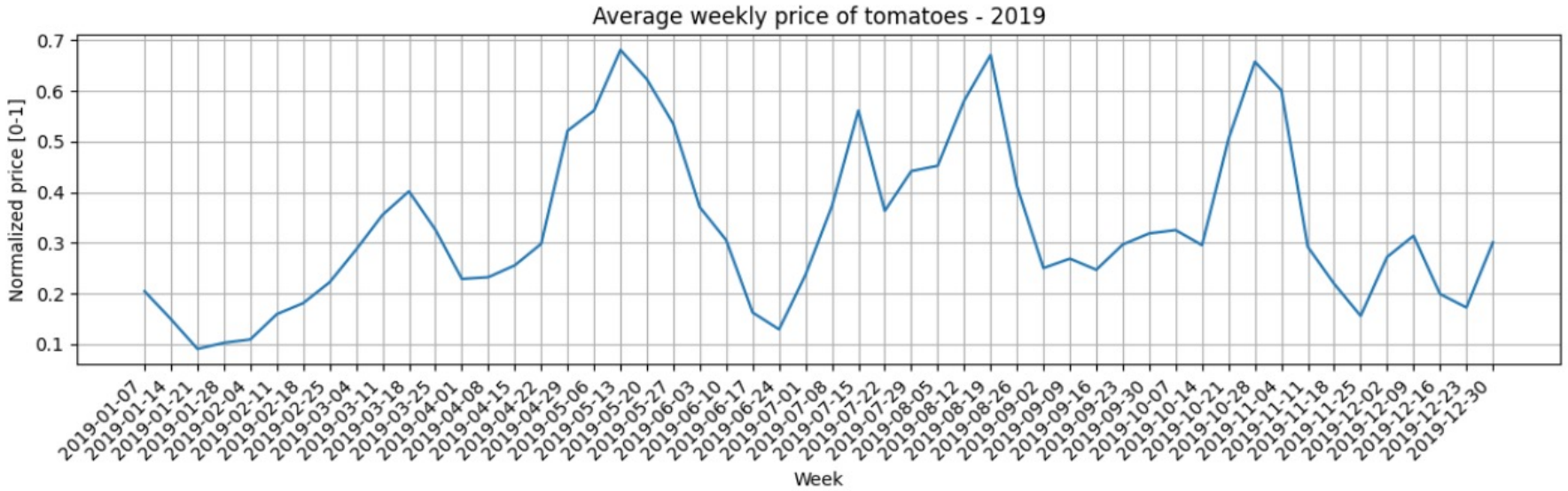
Simple example with n=7 and optimal regularization parameter



Dataset for food price forecasting:

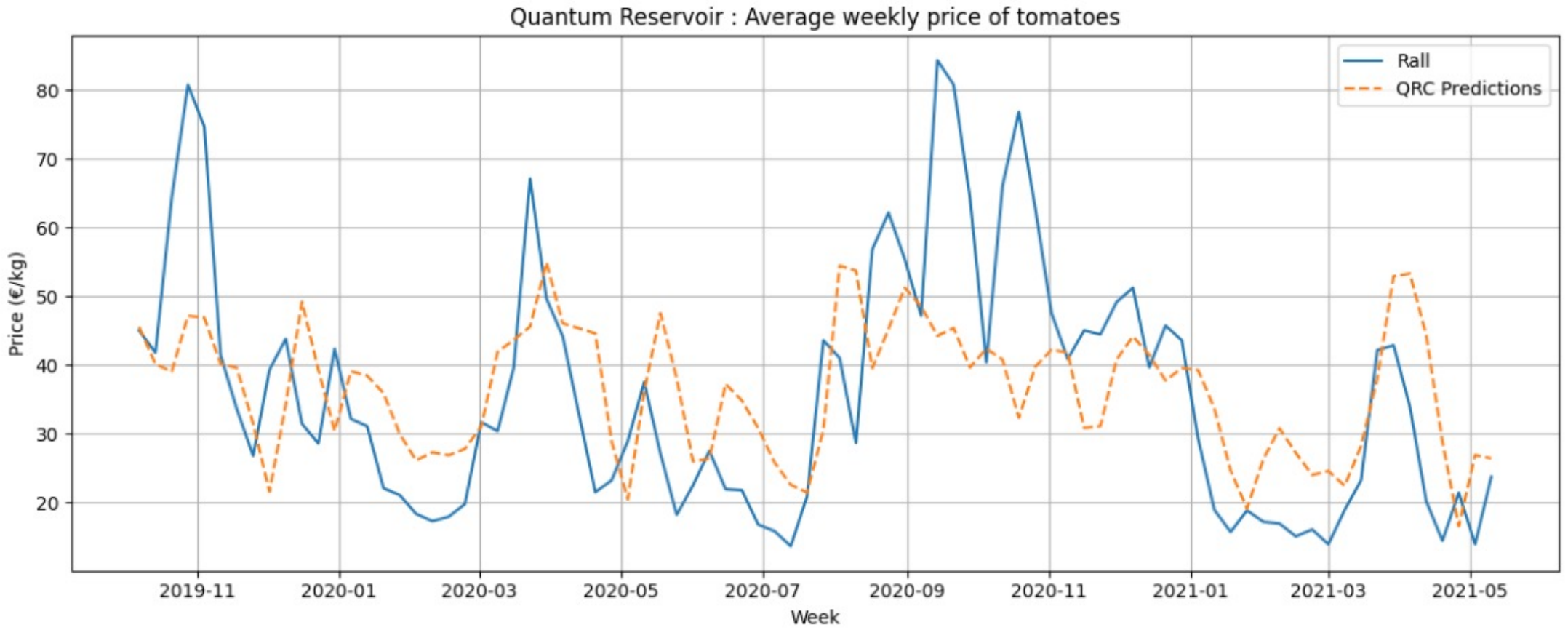
Daily price of tomatoes from january 2014 to May 2021.



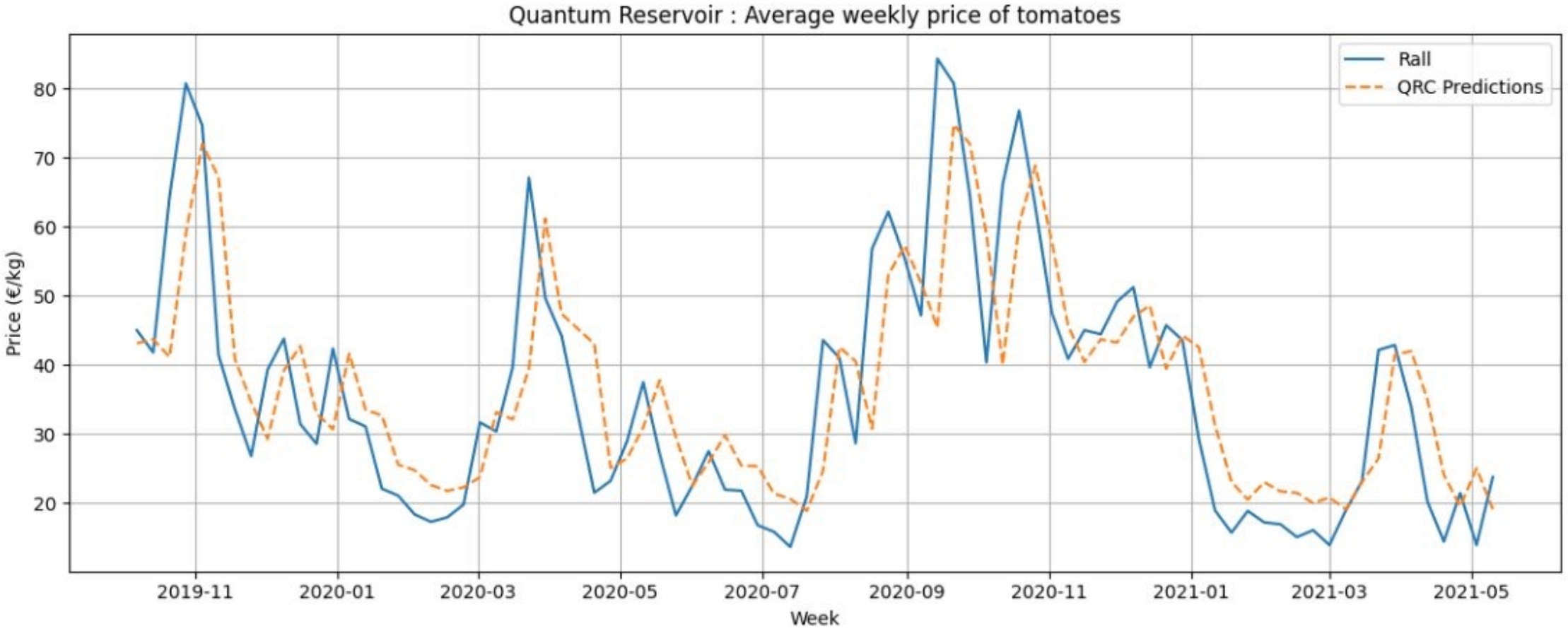


Predictions of the Quantum Reservoir

Here the Reservoir is a fixed random unitary circuit generated by Qiskit.



Unitary circuit with the set of gates $G = \{CNOT, H, T\}$



Optimal quantum reservoir computing for market forecasting: An application to fight food price crises https://arxiv.org/pdf/2401.03347

Comparisons between QRC and classical methods

Forecasting Accuracy	Memory Capacity	Execution Time	Energy Efficiency	Scalability
- Classical RC already excels at complex time-series (ex. **beat ARIMA & LSTM** on food prices).	- QRC can exploit **exponentially large state space**.	- Today, classical RC is **faster** (runs on conventional hardware in ms).	- Classical implementations (optical, analog) are extremely **energy-efficient**	- QRC scales differently: adding qubits exponentially increases feature space (**10 qubits → ~1000-dimensional space**).
- QRC (with optimal design) achieved **similar low error and high trend accuracy** (MDA) on volatile price data.	- QRC can pack memory easily with a trace that has fixed dimension.	- QRC is limited by quantum hardware **speeds** and **repeated measurements**.	- Current QRC hardware requires **heavy infrastructure** (ex. cryogenics), so energy cost is high.	- Demonstrations up to ~**100 qubits show feasibility**. However, noise grows with system size.

Limitations of QRC

- **Hardware constraints:** Real quantum processors are still limited in qubit count, have short coherence times, and can be error-prone.
- **Measurement overhead:** Extracting information from a quantum reservoir requires measurements that can demand multiple runs, which complicates the design and could slow down operation.
- **Sustainability Trade-Off:** Current quantum hardware is still energy-intensive. If the goal is to implement a solution for food security, one must consider whether the benefits of QRC justify the resource usage, especially when classical methods are pretty effective already.
- **Noise and error:** Interestingly, some research suggests that a bit of noise in QRC might actually help (by preventing overfitting, analogous to regularization), but too much noise will destroy useful information.

EPFL

Bibliography

1. **Domingo et al. (2024)** – *Optimal Quantum Reservoir Computing for Market Forecasting: An Application to Fight Food Price Crises*. Introduces QRC for staple-food price prediction; central to our study. arXiv:2401.03347
2. **O'Gorman et al. (2023)** – *Quantum Reservoir Computing*. Detailed methodology and performance analysis of quantum reservoirs. arXiv:2310.07455
3. **Fujii & Nakajima (2017)** – *Harnessing Disordered-Ensemble Quantum Dynamics for Reservoir Computing*. Seminal paper proposing quantum reservoirs built from random circuits. Phys. Rev. Applied 8, 024030
4. **Nokkala et al. (2021)** – *Gaussian States of Continuous-Variable Quantum Systems Provide Universal and Versatile Reservoir Computing*. Extends QRC theory to continuous-variable platforms. *Commun. Phys.* 4, 53
5. **Bharti et al. (2022)** – *Noisy Intermediate-Scale Quantum Algorithms*. Comprehensive review of NISQ-era techniques relevant to QRC. *Rev. Mod. Phys.* 94, 015004
6. **Jaeger (2001)** – *The "Echo State" Approach to Analysing and Training Recurrent Neural Networks*. Classic introduction to classical reservoir computing. GMD Report 148
7. **Financial Times (2024)** – *"Nigeria Hit by Wave of Food Looting as Economic Crisis Deepens"*. Illustrates the social impact of sudden food-price spikes.
8. **Kaggle (2023)** – *Tomato Daily Prices Dataset*. Dataset used to benchmark QRC on real agricultural time series. Kaggle link

EPFL

Image References

Front page: https://medium.com/analytics-vidhya/time-series-forecasting-c73dec0b7533
RC abstract implementation: https://botpenguin.com/glossary/reservoir-computing
Others: available in bibliography.