Reservoir Computing for Predicting Chaotic Dynamical Systems

Taheer Jooma Abbajee[1], **Keegan Anderson**[1,4], **Muaaz Bhamjee**[2,5], **Maria Vivien Visaya**[1,3,4]

[1]Department of Mathematics and Applied Mathematics, University of Johannesburg, Johannesburg, South Africa
[2]Clean Energy Research Group, Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria, South Africa
[3]DSTI-NRF Centre of Excellence in Mathematical and Statistical Sciences, University of the Witwatersrand, Johannesburg, South Africa
[4]National Institute for Theoretical and Computational Sciences, Stellenbosch University, Stellenbosch, South Africa
[5]Department of Mechanical Engineering Science, University of Johannesburg, Johannesburg, South Africa

E-mail: `taheerjooma@gmail.com`

Abstract. Time-series prediction involves forecasting future values by analysing historical data to detect patterns, trends, and variations; chaotic systems are particularly challenging to predict due to their sensitivity on initial conditions. Two primary prediction approaches exist: data-driven and model-based methods. Reservoir computing (RC) is a data-driven model based on recurrent neural networks (RNNs), where the hidden layer is replaced by a "reservoir" represented by a given dynamical system. This model reduces certain complexity (compared to RNNs) while maintaining performance, and excels in predicting time-series from unknown systems. We demonstrate this using the logistic, sine and Hénon maps. The model provided successful short- and medium-term prediction, but remains limited in long-term forecasting. It also inferred key dynamical properties (e.g., fixed points, correlation maps, Lyapunov exponents) not explicitly present in the training data. Our results confirm that RC as an effective alternative to RNNs and highlights its potential for applications in complex systems modelling.

1 Introduction

Chaotic dynamical systems are deterministic but are often difficult to predict due to sensitive dependence on initial conditions. It is often the case that one might only have access to the observed chaotic time series generated by an unknown dynamical system [1]. Reservoir computing is a machine learning algorithm that emerged in the early 2000s as a simplified alternative to RNNs that can be trained on a sample of known time series data and can then be used to predict and analyse the dynamics of the underlying system [1]. In this study we intend to show that reservoir computing models can accurately predict the time-series data and capture the essential dynamics of the logistic, sine, and Hénon map.

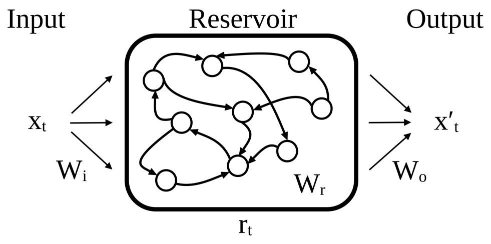

Figure 1: Schematic of a reservoir computer architecture.

2 Reservoir Computing Model

A RC model consists of three layers: an input layer, a reservoir, and an output layer (as shown in Figure 1). Consider a RC consisting of d_r nodes and d_x input and output nodes. At each time-step $t \in \mathbb{N}_0$, let $x_t \in \mathbb{R}^{d_x}$ be the input to the RC, $r_t \in \mathbb{R}^{d_r}$ the hidden reservoir state and $x'_t \in \mathbb{R}^{d_x}$ the output of the RC. Note that d_r is often chosen such that $d_r > d_x$, this is to ensure a higher dimensional mapping of input time-series to the reservoir space. The input layer consists of a matrix $W_i \in \mathbb{R}^{d_r \times d_x}$ which maps the input vector x_t to the reservoir space. The elements of W_i are randomly chosen from a uniform distribution between $[-\sigma, \sigma]$, where σ is chosen such that the elements of $W_i x_t$ lie between $(-1, 1)$. The reservoir is the most important layer and is responsible for computing the next reservoir state. The evolution of the reservoir states is given by

$$r_{t+1} = \tanh(W_i x_t + W_r r_t + b), \tag{1}$$

where $W_i x_t$ is the output of the input layer, $W_r \in \mathbb{R}^{d_r \times d_r}$ is the reservoir weight matrix, and $b \in \mathbb{R}^{d_r}$ is a constant bias term of the form $\beta \mathbf{1}$, where $\beta \in (-1, 1)$ and $\mathbf{1} \in \mathbb{R}^{d_r}$. The reservoir weight matrix can be thought of as an adjacency matrix that represents the connections of the nodes in the reservoir. The matrix W_r is chosen randomly to have a set sparsity (τ) and spectral radius (ρ). The output layer consists of a matrix $W_o \in \mathbb{R}^{d_x \times d_r}$ which projects the reservoir states back to the target dimension $\mathbb{R}^{d_x}$. The matrix W_o is found during training, and then used to found the output x'_{t+1} as given by

$$x'_{t+1} = W_o r_t. \tag{2}$$

3 General Set-up of our RC model

In order to set-up our RC model we follow a procedure similar to the one described by Hara and Kokubu [2]. Assume we have observed time-series data $\{x_t\}_{t \in \mathbb{N}_0}$, then choose $T_0, T_1, T_2 \in \mathbb{N}$ such that $T_0 < T_1 < T_2$ and we can partition the time-series data into three subsets: $\{x_t\}_{t=0}^{T_0}$, $\{x_t\}_{t=T_0+1}^{T_1}$, and $\{x_t\}_{t=T_1+1}^{T_2}$. Lastly, we choose r_0, W_i, W_r and b as described in Section 2.

During the warm-up phase $(0 \le t \le T_0)$ the reservoir states are updated according to equation (1). This phase is used to eliminate transient behaviour caused by the choice of initial conditions x_0 and r_0. During the training phase $(T_0 + 1 \le t \le T_1)$ we continue to update the reservoir states according to equation (1) and collect all the reservoir states and input vectors into the respective matrices $R = [r_{T_0+1} \quad r_{T_0+2} \quad r_{T_0+3} \quad \cdots \quad r_{T_1}]$ and $X = [x_{T_0+1} \quad x_{T_0+2} \quad x_{T_0+3} \quad \cdots \quad x_{T_1}]$. We can then find W_o by solving the minimisation problem $\arg\min_{W_o \in \mathbb{R}^{d_x \times d_r}} \|W_o R - X\|^2$ which has the least squares solution $W_o = XR^T(RR^T)^{-1}$.

A one-step prediction $(T_1 + 1 \le t \le T_2)$ can now be completed using equations (1) and (2). The output time series $\{x'_t\}_{t=T_1+1}^{T_2}$ is expected to be nearly identical to the target time series $\{x_t\}_{t=T_1+1}^{T_2}$. Since $x'_t = W_o r_t$ is expected to be approximately equal to x_t, we can substitute equation (2) for x_t in equation (1) to obtain

$$r_{t+1} = \tanh(W_i W_o r_t + W_r r_t + b). \tag{3}$$

Equation (3) is known as the autonomous reservoir system. After training, the reservoir evolves on its own and can be thought of as a dynamical system of the form $r_t : \mathbb{R}^{d_r} \to \mathbb{R}^{d_r}$. In order to find the optimal hyper-parameters for our RC model, we employ a two-step procedure: first, we explore the hyper-parameter space and find promising regions using random searches; thereafter, Bayesian optimisation is used to refine and extract the optimal hyper-parameters. Table 1 contains all optimum hyper-parameters used in our study.

4 Generalised Synchronisation

Generalized synchronisation (GS) is a type of synchronisation that occurs when the state of one system is completely dependent on the state of another system. The independent system is often called the drive system while the dependent system is referred to as the response system [3].

Hyper-parameters				
Dynamical system	ρ	τ	σ	β
Logistic map	0.866350431222052	0.94	1.2667028204891229	$8.680828069765185 \times 10^{-5}$
Sine map	0.8353366916477455	0.96	1.5641523755434774	$7.422063288375644 \times 10^{-6}$
Hénon map	0.6074166603943667	0.95	0.35371283527605013	-0.5996761514948162

Table 1: Hyper-parameters used for all simulations.

Definition 4.1 (Drive and response systems) *Suppose a drive system is given by a dynamical system (M, ϕ), then a response system $F : \mathbb{R}^N \times M \to \mathbb{R}^N$ is given by $r_{t+1} = F(r_t, x_t)$, where $x_t = \phi^t(x_0) \in M$, $x_0 \in M$, $r_t \in \mathbb{R}^N$, and $t \in \mathbb{N}_0$.*

A reservoir computer is essentially a drive-response system where the discrete dynamical system (M, ϕ) is the drive system, and the evolution of the reservoir states is the response system [4].

Definition 4.2 (Generalised Synchronisation [3]) *Let $V \subseteq \mathbb{R}^N$. A GS between a drive and a response system is a map $g : M \to \mathbb{R}^N$ such that for any $r_0 \in V$ and for any $x_0 \in M$, the following holds $g(x_{t+1}) = F(r_t, x_t) = r_{t+1}, \quad$ for all $t \in \mathbb{N}_0$. If $V \subset \mathbb{R}^N$, then g is called a local GS, and if $V = \mathbb{R}^N$ then g is called a global GS.*

Closely related to GS is the Echo State Property (ESP).

Definition 4.3 (Echo State Property [5]) *Let $r_0, r_0' \in \mathbb{R}^{d_r}$ be initial reservoir states. Let $\{x_t\}_{t\in\mathbb{N}_0}$, $x_t \in \mathbb{R}^{d_x}$, be a given input time-series. A reservoir map $F : \mathbb{R}^{d_r} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_r}$ has the ESP if the sequences $r_{t+1} = F(r_t, x_t) \quad$ and $\quad r'_{t+1} = F(r'_t, x_t)$ satisfy $\|r_{t+1} - r'_{t+1}\| \to 0$ as $t \to \infty$.*

Theorem 4.1 ([6]) *Every reservoir map that has the ESP also has the property of GS.*

The ESP ensures stability in our model as past inputs gradually lose their influence over time, just as the intensity of an echo fades with time. In order to better understand the ESP we will look at contracting reservoir maps.

Definition 4.4 (Contraction of reservoir maps) *A reservoir map $F : \mathbb{R}^{d_r} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_r}$ is called globally state contracting if there exists a constant $c \in (0, 1)$ such that for any $r, r' \in \mathbb{R}^{d_r}$ and $x \in \mathbb{R}^{d_x}$ it follows that $\|F(r, x) - F(r', x)\| \leq c\|r - r'\|$.*

Theorem 4.2 ([7]) *A reservoir map $F : \mathbb{R}^{d_r} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_r}$ that is globally contracting has the global ESP.*

In the RC model that was previously described, we find that our model is contracting and hence has the ESP if and only if the largest singular value of the reservoir weight matrix is less then one [8]. Alternatively, as a sufficiency condition we require the spectral radius of the reservoir weight matrix to be less then one [9].

5 Simulations and Predictions

To quantify the performance of our RC model we choose $T_0 = 50$, $T_1 = 100$, $T_2 = 150$, and apply our model to the following dynamical systems:

- The logistic map with parameter 3.91,
  $x_{t+1} = 3.91x_t(1 - x_t)$ with $x_0 = 0.63$;
- The sine map with parameter 0.955,
  $x_{t+1} = 0.955 \sin(\pi x_t)$ with $x_0 = 0.71$;
- The reduced 1-D Hénon map with classical parameters $a = 1.4$ and $b = 0.3$,
  $x_{t+1} = 1 - 1.4x_t^2 + 0.3x_{t-1}$ with $x_0, x_1 = 0$.

The code used to simulate and study the above dynamical systems is available on GitHub[1].

5.1 One-step and reservoir prediction

Inspecting Figure 2 and Table 2, it can be observed that the one-step predictions are remarkably similar to the actual time series, however analysing the reservoir prediction we see that our predictions have deteriorated slightly but are still similar to the actual time series. This is due to fact that the reservoir prediction has no access to the observed time-series and instead uses its own past predictions to calculate future predictions.

[1]`https://github.com/TaheerJooma/SAIP_rcs`

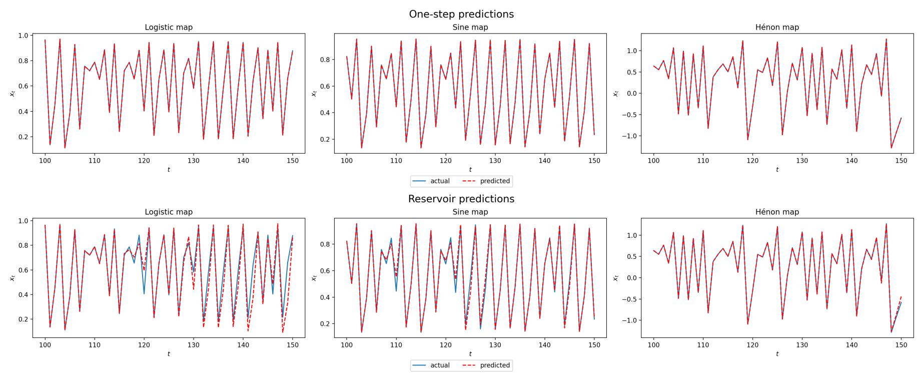

Figure 2: Actual vs predicted trajectories for the one-step and reservoir predictions.

one-step and reservoir prediction errors						
Dynamical system	MAE		MSE		RMSE	
	one-step	reservoir	one-step	reservoir	one-step	reservoir
Logistic map	0.00158448	0.04413210	0.00000310	0.00676420	0.00176089	0.08224477
Sine map	0.00081580	0.01575847	0.00000082	0.00097293	0.00090569	0.03119181
Hénon map	0.00070524	0.00716956	0.00000085	0.00053447	0.00092096	0.02311868

Table 2: Error analysis for the one-step and reservoir predictions.

5.2 Valid prediction time

The valid prediction time (VPT) [10] provides an indication of how many future reservoir predictions our RC model can perform within a given threshold $\epsilon > 0$, and is defined as

$$VPT = \min_{t \in \mathbb{N}} RMSE(n) = \min_{t \in \mathbb{N}} \sqrt{\frac{1}{n} \sum_{t=1}^{n} (x_t - x'_t)^2} > \epsilon,$$

where n is the number of data points, $\epsilon = std(\{x_t\}_{t=T_0+1}^{T_1})$, x_t is the actual value at time-step t, and x'_t is the predicted value at time-step t. In order to gain an idea of the number of reservoir predictions our model can perform within a given threshold, we calculate the VPT for a hundred different initial conditions of our target system and analyse the data in a boxplot illustrated in Figure 3.

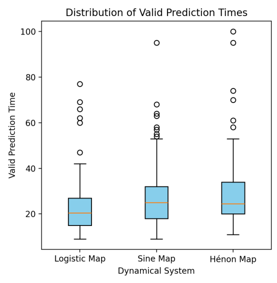

Figure 3: Distribution of VPTs.

Mean of VPT over 100 samples	
Dynamical system	Mean VPT
Logistic map	23.06
Sine map	28.19
Hénon map	29.44

Table 3: Mean VPT for 100 samples.

Notice that the data in Figure 3 contains numerous outliers. This can be explained by considering the overall stability of the initial condition used. If the initial condition lies in or around a stable region of the target dynamical system then the VPT will be fairly large as the time series is more predictable. Conversely, if the initial condition lies in or around an unstable region of the target dynamical system then the VPT will be smaller as the time series is more unstable, hence less predictable. From Table 3, we see that our model on average can reliably predict between 23-29 steps into the future, depending on which map is being forecasted.

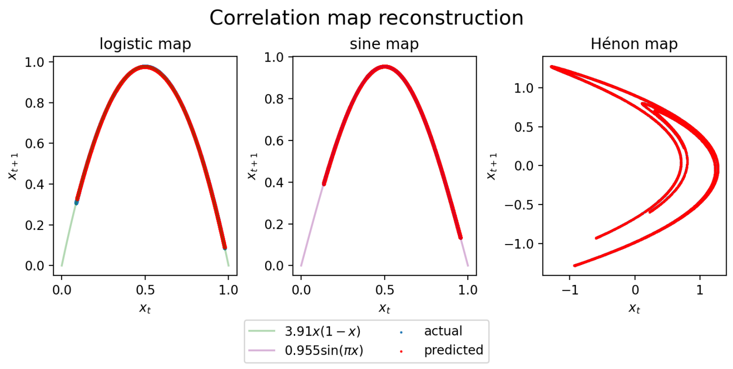

Figure 4: Actual vs predicted correlation maps.

Comparison of fixed points				
Dynamical system	Actual	Predicted		MAE
		Spurious	Non spurious	
Logistic map	0, $\frac{291}{391}$	1.5297709	-0.00994626, 0.74397722	0.00994626, 0.00026830
Sine map	0, 0.72540262	$\pm$ 2.06426943, 0.7252752	-0.00000396, 0.72505301	0.00000396, 0.00034961
Hénon map	$\frac{-7\pm\sqrt{609}}{28}$	None	0.63147419, -1.13309976	0.00011971, 0.00174529

Table 4: Comparison of actual and predicted fixed points.

5.3 *Correlation map reconstruction*

A correlation map between x'_t and x'_{t+1} describes and visualises how x'_t is related to x'_{t+1}. In order to visualise the predicted correlation map we plot x'_t vs x'_{t+1} for 10 000 points. From Figure 4 it can be seen that the predicted correlation maps are almost identical to the actual correlation maps in both shape and structure.

5.4 *Prediction of fixed points*

In order to find predicted fixed points we will first solve for r^* in the equation $r^* = \tanh((W_iW_o + W_r)r^* + b)$. The predicted fixed points of the target system is then given by $x'^* = W_or^*$. It is important to note that since x'^* is a projection of r^* to a lower dimensional space $\mathbb{R}^{d_x}$, both x'^* and r^* share the same stability. The stability is determined by analysing the eigenvalues of the Jacobian matrix $J(r) = \text{diag}\big[1 - \tanh^2((W_iW_o + W_r)r^* + b)\big](W_iW_o + W_r)$. Table 4 lists and compares the actual and computed fixed points, and reveals numerous spurious fixed points. The spurious fixed points can easily be identified and eliminated as they are outliers when compared to the training data. Examining the errors in Table 4 it can be observed that our RC model is capable of predicting actual fixed points up to at least two decimal places. Lastly, all the actual and predicted fixed points were found to be unstable.

5.5 *Prediction of Lyapunov exponents*

Lyapunov exponents characterize the rate of separation of infinitesimally close trajectories. A positive Lyapunov exponent is associated with sensitivity to initial conditions and is often used as indicator of chaos. From the errors in Table 5 we see that our model is capable of predicting all Lyapunov exponents up to at least two decimal places.

Comparison of Lyapunov exponents			
Dynamical system	Actual	Predicted	MAE
Logistic map	0.49123091	0.50002128	0.00879036
Sine map	0.45938807	0.46568606	0.00629799
Hénon map	0.42081703, -1.62478983	0.41453935, -1.61709175	0.00627768, 0.00769809

Table 5: Comparison of actual and predicted Lyapunov exponents.

6 Conclusions

This study looked at a RC approach for predicting and analysing chaotic dynamical systems based solely on observed time-series. In conclusion our reservoir computer was capable of performing short- to medium-term time series predictions. Our model was also capable of learning and preserving key dynamical properties that were not explicitly contained in the training data. However, our model remains limited in long-term forecasting. Possible future work includes exploring the embedding of the reservoir space and further testing on higher dimensional systems.

Acknowledgements

MVV acknowledges funding from the DSI-NRF Centre of Excellence in Mathematical and Statistical Sciences (CoE-MaSS). The authors acknowledge the scholarship funding for this research from the Faculty of Science, University of Johannesburg, and the Centre for High Performance Computing (CHPC), South Africa, for providing computational resources to this research project. Opinions expressed, and conclusions arrived at are those of the authors and are not necessarily to be attributed to the CoE-MaSS, NITheCS, the CHPC, the University of Johannesburg or the University of Pretoria.

References

[1] A. Aussem, "Dynamical recurrent neural networks towards prediction and modeling of dynamical systems," *Neurocomputing*, vol. 28, no. 1-3, pp. 207–232, 1999.

[2] M. Hara and H. Kokubu, "Learning Dynamics by Reservoir Computing (In Memory of Prof. Pavol Brunovský)," *Journal of Dynamics and Differential Equations*, vol. 36, no. S1, pp. 515–540, Feb. 2024.

[3] N. F. Rulkov, M. M. Sushchik, L. S. Tsimring, and H. D. Abarbanel, "Generalized synchronization of chaos in directionally coupled chaotic systems," *Physical Review E*, vol. 51, no. 2, p. 980, 1995.

[4] A. Nazerian, C. Nathe, J. D. Hart, and F. Sorrentino, "Synchronizing chaos using reservoir computing," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 10, pp. 103–121, 2023.

[5] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural networks*, vol. 35, pp. 1–9, 2012.

[6] L. Grigoryeva, A. Hart, and J.-P. Ortega, "Chaos on compact manifolds: Differentiable synchronizations beyond the Takens theorem," *Physical Review E*, vol. 103, no. 6, p. 062204, 2021.

[7] A. Hart, J. Hook, and J. Dawes, "Embedding and approximation theorems for echo state networks," *Neural Networks*, vol. 128, pp. 234–247, 2020.

[8] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks—with an erratum note," *Bonn, Germany: German national research center for information technology gmd technical report*, vol. 148, no. 34, p. 13, 2001.

[9] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.

[10] J. A. Platt, S. G. Penny, T. A. Smith, T.-C. Chen, and H. D. Abarbanel, "A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics," *Neural Networks*, vol. 153, pp. 530–552, 2022.