
Universitat
de les Illes Balears

# Autonomous dynamical systems based on hardware implementations of delay-reservoir computers

**MASTER'S THESIS**

Master's degree in Physics of Complex Systems

at the

UNIVERSITAT DE LES ILLES BALEARS

*Academic year* 2017/2018

*Author:*
Irene ESTÉBANEZ

*UIB Master's Thesis Supervisors:*
Miguel C. SORIANO
Ingo FISCHER

September 20, 2018

*«A good thesis starts aiming to answer one question and ends up with more than ten open questions.»*

# Abstract

Reservoir computing (RC) is a machine learning technique allowing for novel approaches to realize trainable autonomous nonlinear oscillators. Here we employ delay-based echo state networks with output feedback, simple yet powerful implementations of neuromorphic systems, to reproduce the dynamical behavior of a Rössler chaotic system. Our hardware implementation relies on a delay-based RC topology, and consists of two main elements: an analog Mackey Glass nonlinearity and a Raspberry Pi board. We demonstrate the capacity of our experiment to generate chaotic time-traces in an autonomous manner, and we prove that noise can play a constructive role in the training process, when realizing nonlinear oscillators based on closed-loop operation. We use phase-space reconstruction of the chaotic attractor and the comparison of the frequency spectra, along with recurrence quantification analysis (RQA), to perform a nonlinear data analysis with the aim of comparing the autonomous operation and the original time-series in more detail.

# Acknowledgements

First of all, because of their importance in this Master thesis, my supervisors: Ingo and Miguel. There are no words to explain my gratitude for keeping me highly motivated to improve myself every day. It is their constant supervision and their teaching ability that have enriched the pages of this TFM.

Because they have behaved like part of my family during this year in Mallorca, to my classmates, especially to my flatmate-brothers Oscar, Luis, and Giacomo. Last, but no the least, I would like to thank my parents, Mar and Faustino, for being always by my side, no matter what kind of crazy decision I had made.

# Contents

# Motivation

The twenty-first century is considered the information age. The expansion of the Digital Revolution, together with the greater capacity of storage devices and the popularity of online connections, has lead to the generation of vast amounts of data. Consequently, many disciplines arose seeking to extract useful information from these large volumes of data. Among them, the usually known as *machine learning*, is the field of computer science that aims to create algorithms and programs that learn on their own. Many techniques are included under this umbrella-name, we focus here on *Reservoir Computing* (RC) from the point of view of dynamical systems theory since it has achieved a state-of-the-art performance for processing sequential data. RC is a family of recurrent neural networks (RNN) whose recurrent part (the *reservoir*) is kept fixed. Reservoirs can be implemented with networks of random topology, but here we will focus on the ring topology introduced in [1], since it allows for a straightforward hardware implementation. In particular, we have built this reservoir implementation, also noted by *delay-based reservoir*, with a Mackey Glass delay system.

The aim of this Master thesis is to use delay-based RC in order to achieve autonomous chaotic time-series prediction. This is a really demanding task due to the sensitivity to initial conditions and the complex geometries of strange chaotic attractors. Despite these difficulties, the study of these systems deserve attention due to the useful applications in a wide variety of fields like weather forecasting and stock prediction.

### Contributions of this Master thesis

Following the first promising results in experiments [2] and modeling [3], we demonstrate the capability of our experimental setup to generate chaotic time-traces in an autonomous manner. As our main contribution, we prove that training the reservoir with random noise applied to the training input leads to better results in the autonomous prediction task. This input noise injection has been demonstrated to improve the robustness in the field of deep neural networks [4]. Even the naked eye is capable of recognizing that the phase-space reconstruction of the chaotic attractor is better performed by adding input noise to the reservoir. In addition, we perform a quantitative analysis by means of *Recurrence Quantification Analysis* (RQA) to validate our results.

### Structure of the master's thesis

Chapter 1 provides a short introduction to Reservoir Computing, with emphasis on delay-based RC. Here, we also introduce the Rössler system, as it is the one we aim to reproduce in an autonomous manner. Finally, we also introduce different techniques for the analysis of nonlinear time-series, such as attractor reconstruction and recurrence analysis.

In Chapter 2 methodology and experimental setup are presented. We explain the main components of our delay-based RC and we characterize it. For the sake of completeness, we program a saving-time tool in Matlab mimicking the experiment

in order to complement the experimental results. At the end, we present the optimal number of nodes in the reservoir and some features of the operating point.

Chapter 3 contains the experimental results for the one-step-ahead prediction and the autonomous operation. We review the main features of short-term and long-term predictions of the autonomous run. Specifically, we calculate the frequency spectra and some quantities of recurrence quantitative analysis (RQA). Also, we plot the phase-space reconstruction of the chaotic attractor and recurrence plots. In the end, we show the results when introducing input noise into the reservoir to train the output weights.

Finally, we review and explain the main contributions of this TFM in the chapter *Conclusions and Outlook*, and also we suggest some experiments to be done in future in chapter *Future Perspectives*.

# Chapter 1

# Introduction

This chapter is meant to be a brief review on concepts in reservoir computing (RC). Reservoir computing defines a class of artificial neural networks (ANN) that mimic neural microcircuits in the biological brain using an un-trained reservoir of neurons and a trained readout function. This technique has been developed following three different methods: Echo State Networks (ESN), Jaeger (2001) [5]; Liquid State Machines (LSM), Maass et al. (2002) [6]; and Backpropagation-Decorrelation (BPDC), Steil (2004) [7]. Since its start in the early 2000s, successful applications of RC include speech and handwriting recognition [8, 9], robot motor control [10], time-series prediction [11, 12], medical brain-computer interfacing [13] and signal recovery in optical communications [14].

Here, we aim to achieve versatile and robust trainable systems based on hardware implementations of ESN, capable of mimicking different nonlinear oscillators. In particular, we will reproduce the dynamical behavior of a Rössler dynamics as the target system. To this end, we perform both, one time-step prediction and then autonomous signal generation using the Rössler system. Finally, we will introduce some nonlinear analysis methods that will be used to explore to what extent the autonomously generated trajectories resemble the Rössler chaotic attractor.

## 1.1 General concepts on RC

The traditional RC concept comprises three distinct parts: an input layer, the reservoir, and an output layer, as sketched in Figure 1.1 **a)**. Through the input layer, the input signals are fed into the reservoir often using random weight connections. The reservoir usually constitutes a recurrent network composed by a large number of randomly interconnected nonlinear nodes. This network exhibits responses while the input signals are injected. Finally, these responses are read out at the output layer via a linear weighted sum of the individual node states. The readout weights are trained, while the input and reservoir connections are usually left unaltered.

There are three basic properties that should be fulfilled for a network to perform as a reservoir. The first one is known as the *separation property*, and provides that different inputs should be mapped onto different reservoir states. The second is the *approximation property*, it is used to guarantee that similar inputs will result in similar output states or will be mapped into the same output class. If not, even a small amount of noise would be enough to map identical inputs onto different targets. Finally, the reservoir is required to exhibit *fading memory*. This is to allow information processing in the context of previously injected information. This is important for many tasks, like, e.g., speech recognition. Usually, only recent inputs are relevant while those from the far past are not needed. As the reservoir can be regarded a complex dynamical system, these three properties can be realized choosing the proper

dynamical regime. A typical reservoir contains a large number D of internal nonlinear nodes $r_i(t)$ evolving in time. Under the assumption of discrete time $t = n \in \mathbb{Z}$, nonlinear nodes $r_i(n)$ evolve as given by

$$r_i(n) = f\left(\sum_{j=0}^{D-1} a_{ij} r_j(n) + b_i x(n)\right), \tag{1.1}$$

where $f$ is a nonlinear function, $x(n)$ is some input signal, and $a_{ij}$ and $b_i$ are time-independent coefficients tuning the dynamics of the reservoir. These coefficients are adjusted for the reservoir to work in the proper dynamical regime.

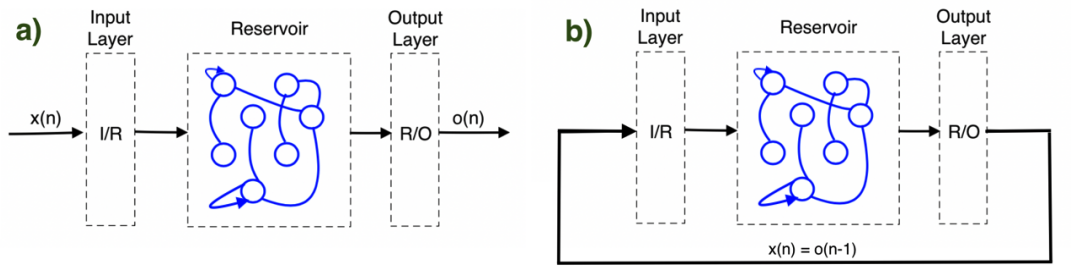



**Figure 1.1:** Schematic representation of a) the training stage and b) the autonomous run of a reservoir computer. During the training phase, the reservoir is driven by a teacher input signal $x(n)$, and the readout weights $w_i$ are optimized for the output to be as close as possible to the target $y(n)$. Later, in the autonomous run, the teacher signal is changed by its own output signal, $o(n-1)$. The readout weights $w_i$ are kept constant.

The main characteristic of RC is that the reservoir is kept fixed. Once we have obtained the responses at the output layer, the readout is performed via a linear weighted sum of individual node states. The computation of the proper weights takes place during the *training state*, usually via linear regression. This stage is performed using some interval of the input signal. Then, using another interval of the input signal, one has to check the validity of the weights in a *testing stage*. Finally, once the weights are calculated and tested, we can proceed to the *autonomous run*. Instead of working with some open-loop configuration in which we are feeding the reservoir with the original signal as illustrated in Figure 1.1 **a)**, we can close the loop, replacing the teacher signal by its own output signal (Fig. 1.1 **b)**). A detailed explanation of the different stages is provided below.

### 1.1.1 Training strage

RC requires a training procedure. This procedure was created mimicking a neuromorphic computational concept based on a dimensionality expansion due to random nonlinear mapping. Generally, the higher dimensional the state-space of the reservoir is, the more likely it is that the data become linearly separable. This concept is illustrated in Figure 1.2. As we can see in Fig. 1.2 **a)**, the yellow spheres and the red stars cannot be separated with a single straight line. Instead, when we move to a three-dimensional space, the spheres and stars might be separable by a single linear hyperplane as depicted in Fig. 1.2 **b)**.

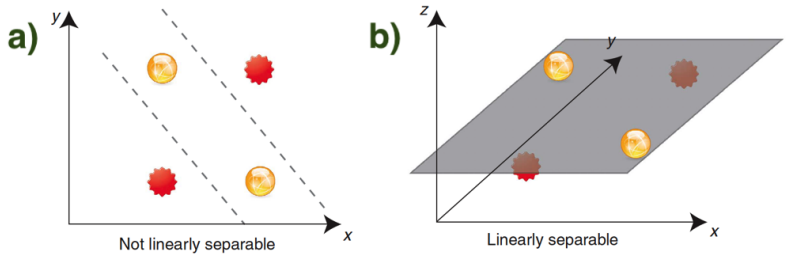



**Figure 1.2:** Illustration of the key aspects of reservoir computing. **a)** In its original $2D$ representation, yellow balls and red stars cannot be linearly separated. **a)** Adding an additional dimension, a linear separation via a linear hyperplane might become possible. Figure adapted from Appeltant et al. [1].

Mathematically, the readout of a reservoir computer is formed by a linear combination of the reservoir states. The reservoir computer produces an output signal $o(n)$ given by a linear combination of the states of its internal variables,

$$o(n) = \sum_{i=1}^{N} w_i r_i(n) \tag{1.2}$$

where $w_i$ represents the readout weights, trained in order to minimize the Normalised Root Mean Square Error (NRMSE) between the output signal $o(n)$ and the target signal $y(n)$.

$$NRMSE = \sqrt{\langle [y(n) - o(n))]^2 \rangle} \tag{1.3}$$

where $y(n)$ and $o(n)$ have been normalized.

Notice that the training stage is performed in an open-loop configuration as illustrated in Figure 1.1 **a)**. Each time step $n$, we are feeding the reservoir with the teacher signal and obtaining some reservoir response. Once we have collected all the responses of the reservoir to the input signal used for the training stage, we perform a linear regression to calculate the weights. Finally, we compute the output signal and determine the value of the NRMSE.

### 1.1.2 Testing state

During this stage we are going to use the weights calculated in the training stage. We note that during this stage we are also using an open-loop configuration. We introduce the interval of the input signal corresponding to the testing state into the reservoir and collect the responses. Then, an output signal $o(n)$ is computed via a linear weighted combination of the nodes using the weights obtained in the training stage. In the end, we can also calculate the NRMSE. This value is usually higher than the NRMSE obtained in the training state and illustrates if the system is working properly.

### 1.1.3 Autonomous run

The RC paradigm can be employed to perform nonlinear prediction of the next time steps of a given time-series. However, this could be limited by several factors, including noise and the properties of the time-series to be predicted. When working with chaotic time-series, one is more interested in obtaining time-series structurally

similar to the original system, but not identical. Here, we demand our system to generate such chaotic time-traces in an autonomous manner. To close the loop we only need a small modification in the architecture. As illustrated in Figure 1.1 **b)**, the output signal needs to be injected back into the reservoir. Since the RC can now switch from two different signals as input, we denote $Input(n)$ as the input signal, which can be either the external input signal $Input(n) = x(n)$ or its own output delayed by one time step, $Input(n) = o(n-1)$. The step of feeding back the output signal notably enriches the internal dynamics of the system, allowing the autonomous generation of time-series. There is no need of receiving an input signal. Conceptually, this is a small change, and it has been proved that the long-term prediction of time-series is possible [15]. This approach even achieves record performance for chaotic time-series prediction [16]. However, we would like to highlight here that experimental implementations have their own particularities that still need to be explored.

## 1.2 Delay-based RC

Nonlinear systems with delayed feedback, often called simply *delay systems*, have drawn the attention of the scientific community not only due to their fundamental interest, but also because they appear in a diversity of real-world systems [17]. The delay has been proved to have different impacts on the dynamical behavior of the system, acting as a stabilizing or destabilizing influence [18]. One of the most impact-full examples of delay systems was found in optics. Particularly, when the output light of a semiconductor laser is injected back into it (e.g. due to an external mirror at a certain distance), depending on the feedback strength, it can induce a variety of behavious, ranging from stable via periodic and quasiperiodic oscillations to deterministic chaos [19, 20]. This property, initially considered a nuisance, is now viewed as a resource that can be beneficially exploited. One of the simplest possible delay systems consists of a single nonlinear node whose dynamics is influenced by its own output a time $\tau$ in the past. Such a system is only composed of two elements, a nonlinear node and a delay loop [1]. For this reason, they seem very attractive to implement RC experimentally.

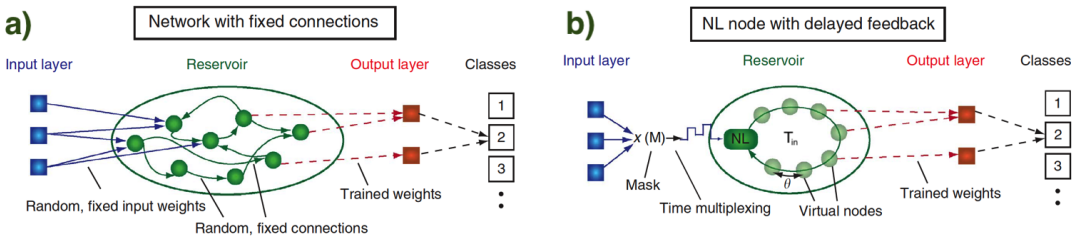



**Figure 1.3: a)** Classical RC scheme. The input is coupled into the reservoir via a randomly connected input layer to the D nodes in the reservoir. **b)** Scheme of RC utilizing a nonlinear node with delayed feedback. A reservoir is obtained by dividing the delay loop into D intervals. Figure adapted from Appeltant et al. [1].

Traditional RC architectures (Section 1.1) employ a large number of nonlinear reservoir nodes to obtain good performance. Delay-based reservoirs propose to implement a reservoir computer in which the usual structure is replaced by a dynamical system comprising a single or few nonlinear nodes subjected to delayed feedback, as schematically shown in 1.3 **b)**. Mathematically, the dimensionality expansion is given by the nature of the delay system. The infinite dimension of the state-space of a delay system is given because their state at time $t$ depends on the output of

the nonlinear node during the continuous time interval $[t-\tau, t)$. In addition to this time scale, we can distinguish the data injection time $T_{in}$, defined by the number of virtual nodes $D$ and the node separation $\theta$, such that $T_{in} = D \cdot \theta$. The virtual nodes are placed regularly along the delay line as shown in Figure 1.3 **b)**.

### 1.2.1 Input driving

In order to work with the architecture introduced in Figure 1.3 **b)** we need to feed the reservoir with a stream $I(t)$ constant during $T_{in}$. For that purpose, the input signal $x(t)$ undergoes a sample and hold operation, such that

$$I(t) = x(n) \quad \text{where} \quad T_{in}n \leq t < T_{in}(n+1). \tag{1.4}$$

In addition to this, we also need to implement the weights that connect the input layer with the reservoir like in traditional RC architectures. Here, we cannot implement a scaling factor in the virtual node itself, since there is only one nonlinear node driving the other virtual nodes. What Appeltant et al. proposed in [1], is to introduce a masking function $M(t)$ to insert the coupling weights from the stream $I(t)$ to the virtual nodes. This is defined as the *masking procedure*, illustrated in Figure 1.4. This mask function is constant during a node distance $\theta$ and periodic of period $T_{in}$. The values of the mask function are drawn from some probability distribution at random. The different nodes $i$ are multiplied by different weights, this is denoted has $\mathbf{W}_i^{in}(n) = M(t)$. In the end, the input signal injected into the reservoir is $J(t) = M(t) \cdot I(t)$[1].

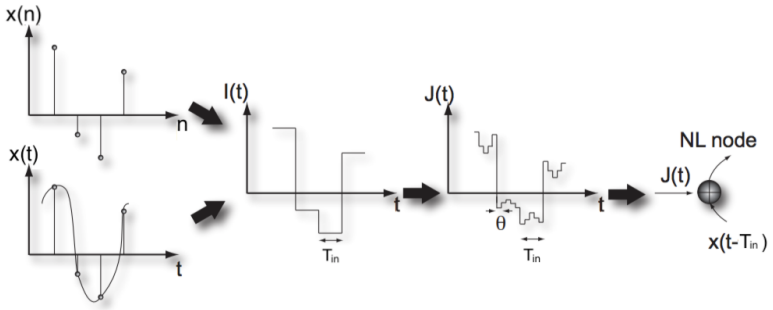



**Figure 1.4:** Starting either from a time-continuous or time-discrete input stream, after going under a sample and hold operation, we obtain an input stream constant over a data injection time $T_{in}$ before it is updated. The temporal input sequence, feeding the input stream to the virtual nodes, is then given by $J(t) = M(t) \cdot I(t)$. Figure adapted from Appeltant et al. [1].

### 1.2.2 Interconnection structure

Once we know how the input signal is mapped into the reservoir, we need to know how the nodes are coupled within the reservoir. Imagine we are dealing with a simple evolution equation for a delayed feedback system,

$$\dot{\mathbf{r}}(t) = F(\mathbf{J}(t), \mathbf{r}(t-\tau)) \tag{1.5}$$

[1]It is also possible to implement this configuration when the input signal has more than one dimension. In that case, if the input consists of $Q$ values $I^j(t)$, we generate a separate mask $M^j(t)$ for each input $j$ and subsequently they are all summed together $J(t) = \sum_{j=1}^{Q} I^j(t) \cdot M^j(t)$

where $F$ describes a dynamical system. As any flow dynamical system, it has a certain time response $T$. When subjected to a particular input, this time response represents how the state of the dynamic system changes. This introduces an additional time scale $T$, that defines the dynamical properties of the RC system together with the time separation of the $D$ virtual nodes $\theta$, the sampling data rate $T_{in}$, and the delay time $\tau$. Depending on the choice of $\theta$ with respect to $T$ different regimes arise. These regimes lead to different connections between the nodes in the reservoir.

1. **Flow Regime.** When $\theta < T$, the state $\mathbf{r}(t)$ of the system at time $t$ depends on the states of the previous neighbouring virtual nodes, as illustrated in Figure 1.5. Even if $T_{in} = \tau$, all nodes are connected to the adjacent nodes. The strength of this connection decays exponentially while increasing the separation of the virtual nodes $\theta$ [1]. Still, if $T/\theta$ is too large, the system is not responding to the instantaneous value of the feedback and input, but only to the average taken over many previous nodes, as depicted in **b)**. Coupling between virtual nodes is desired, but without too much averaging.

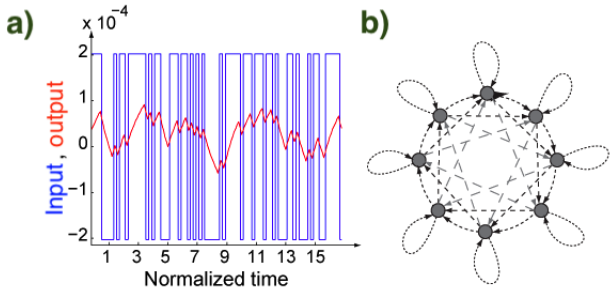



**Figure 1.5: a)** Input time trace and **b)** the corresponding interaction graph. In **a)** the input (blue) and the output signal (red) when the $\theta < T$. Here, the system does not have the time to reach a steady state. Therefore, the dynamics of the nonlinear node couples neighboring virtual nodes, as depicted in **b)**. Figure adapted from [1].

2. **Map Limit Regime.** When $T << \theta$ the system is able to reach its steady state for each virtual node. Here, the reservoir state $\mathbf{r}(t)$ is only affected by the input signal $\mathbf{x}(t)$ and the state of the reservoir one delay time ago $\mathbf{r}(t-\tau)$. Consequently, choosing $T_{in} = \tau$ only provides self-coupling between nodes, and the diversity of the reservoir states goes down.

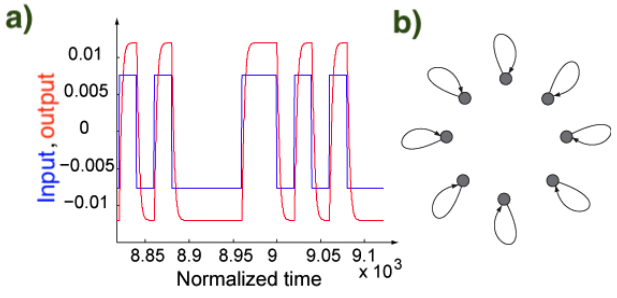



**Figure 1.6: a)** Input time trace and **b)** the corresponding interaction graph. In **a)** the input (blue) and the output signal (red) when the $T << \theta$. Here, the system rapidly reaches a state that is independent of previous inputs. That means the $D$ nodes behave independently, each one is coupled only to itself at the previous time-step. Figure adapted from [1].

Here, the coupling has to be introduced by a mismatch between the delay time and the input sampling period, $\tau \neq T_{in}$. This misalignment can be quantified using $\alpha = (\tau - T_{in})/\theta$. Depending on the selection of $\alpha$ different topologies arise. In Figure 1.7 we show a schematic representation of the ring topology that derives from the choice $\alpha = 1$ when $D = 6$.

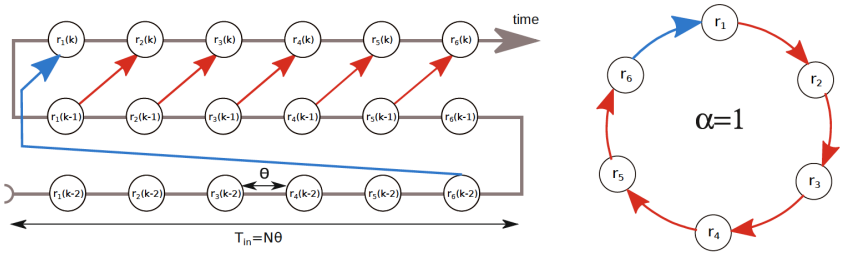



**Figure 1.7:** Schematic representation of the virtual nodes along the delay line (left) and the corresponding interaction graph (right). Red arrows indicate connections with nodes one step time before and the blue arrow indicates a connection with a node two steps back in time. Figure courtesy of S. Ortín.

Under the conditions illustrated in Fig. 1.7 the state of the virtual nodes can be described by the following equations,

$$\begin{aligned} r_i(n) &= F(\gamma \mathbf{W}_i^{in}\mathbf{x}(n) + \beta r_{i-1}(n-1)),\ i = 2, ..., D \\ r_1(n) &= F(\gamma \mathbf{W}_i^{in}\mathbf{x}(n) + \beta r_D(n-2)) \end{aligned} \tag{1.6}$$

where $\beta$ and $\gamma$ are feedback and input scaling factors, respectively.

In this Master thesis, our experimental implementation will be operating in the map limit regime. Thus, the coupling between nodes will be introduced by a mismatch between the injection data time $T_{in}$ and the delay time $\tau$. The next section contains brief notes about the characteristics of the input signal that will feed up our reservoir, the Rössler dynamical system.

## 1.3 The Rössler dynamical system

Since the Poincaré-Bendixson theorem was formulated in 1901, it is proven that the number of dynamical degrees of freedom necessary to exhibit deterministic chaos is three. On this basis, Otto Rössler came up with some prototype systems of ordinary differential equations with the minimum ingredients for continuous time chaos. These prototypes, appearing in different publications [21–24], were inspired by the geometry of three-dimensional flows. In particular, by the *reinjection principle*. This reinjection principle is a feature of some relaxation-type systems having a Z-shaped slow manifold in the phase-space. Trajectories move along the slow manifold until they suddenly jump to another branch of the manifold. This kind of flows can produce periodic relaxation oscillations in 2D, see Figure 1.8 **a)**. When adding a third dimension, reinjection can also induce chaotic behavior if the motion is spiraling out on one branch of the slow manifold, see Figure 1.8 **b)**.

The most relevant of these prototype systems is formulated as

$$\begin{aligned} \dot{x} &= -y - x \\ \dot{y} &= x + ay \\ \dot{z} &= bx - cz + xz. \end{aligned} \tag{1.7}$$

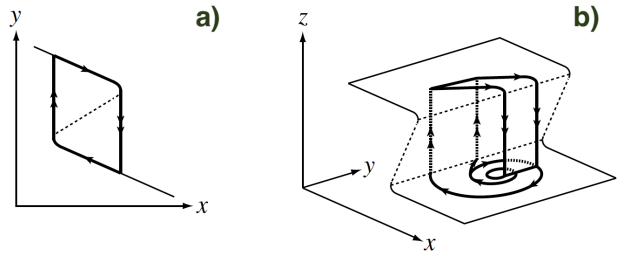



**Figure 1.8:** Illustration of the reinjection principle between the two branches of a Z-shaped slow manifold in a) 2D b) and 3D. Figure addapted form [26].

where $(x, y, z)$ are the three variables evolving in time $t$ and $(a, b, c)$ three parameters. In this system (1.7), the linear terms in the two first equations are creating oscillations in the $(x, y)$ plane. The strength of these oscillations depends on the value of $a > 0$. The motion in $(x, y)$ is then coupled to the $z$ variable through the third equation, whose nonlinear term induces the reinjection back to a spiraling out motion. Depending on the values of the parameters, trajectories are oscillating around the two stationary points (one at the origin and the other at some distance from the origin), exhibiting stationary, periodic, quasiperiodic, or chaotic attractors.

The transition from stationary to periodic attractors happens via a Hopf bifurcation. Then, a period doubling cascade occurs until the onset of chaotic dynamics. The corresponding chaotic attractors have one single lobe, in contrast to the popular Lorenz attractor which has two lobes, as illustrated in Figure 1.9 **a)**.

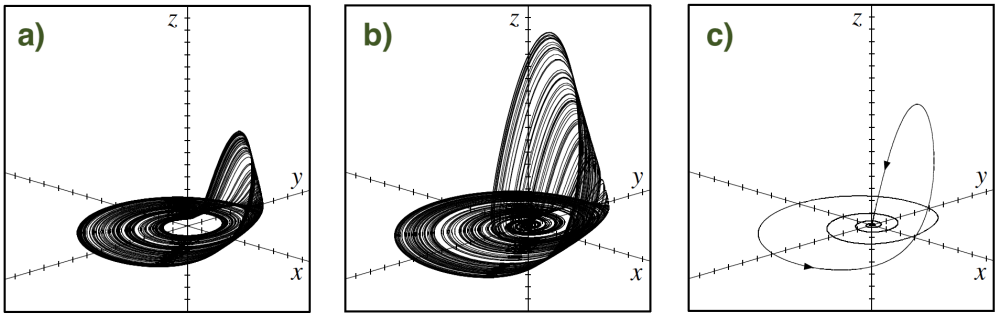



**Figure 1.9:** Phase portraits of the Rössler system. **a)** Spiral type chaos **b)** Screw-type chaos, and **c)** a Shil'nikov type homoclinic orbit. Figure adapted from [25].

This system also exhibits a transition to a screw-type chaos in which the oscillations are irregular in both, amplitudes and reinjection times, as shown in Figure 1.9 **b)**. This kind of chaos is related to the presence of a Shil'nikov homoclinic orbit (Figure 1.9 **c)**). Away from the homoclinic orbit, the dynamics of the system becomes complex showing both, periodic and chaotic attractors [26].

Among all these possible behaviors, we will restrict our attention to the study of the chaotic region. Due to its sensitivity to initial conditions and the complex geometry of the chaotic attractors, the one-step prediction and the autonomous signal generation of this dynamics are challenging tasks. Here, our aim is to mimic the dynamical behavior of the first variable $x(n)$ in Eq. (1.7) when the system is chaotic. For such purpose we choose parameters $a = 0.2$, $b = 0.2$ and $c = 5.7$. In particular, we consider a discrete time series $x(n)$, obtained from the continuous system such that 20 points per oscillation are kept. This time-series is normalized with zero mean and standard deviation one. The time-series is divided into two sets where 3400 points

are used for training and 350 points for testing purposes. A sample of the time-series used in the prediction of $x(n)$ is shown in Figure 1.10.

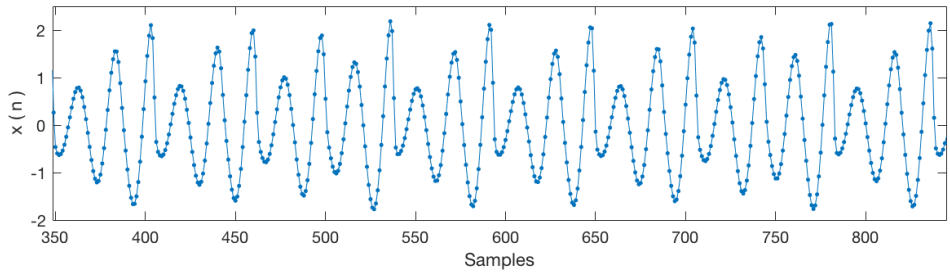



**Figure 1.10:** Fraction of the time-series used in the prediction of $x(n)$. The system in Eq. (1.7) is solved using a Runge-Kutta 5 method. The step size of integration is 0.1, and the sampling in our time-series is performed each 0.3. This code is available online in [27].

## 1.4 Nonlinear time-series analysis

Extracting meaningful information from nonlinear time-series is a demanding task. For such purposes, there exists a variety of methods encompassed in the area of "Nonlinear time-series analysis" offering potentially powerful tools to learn about key properties behind some observed time-ordered data. Time-series analysis is especially useful for dynamical systems in which nonlinearities give rise to a complex temporal evolution that is not captured by classical linear techniques.

The analysis of these time-series is performed via dynamical systems theory. Using the concept of state-space reconstruction, one is able to compute relevant quantities as fractal dimensions, Lyapunov exponents, or K-S entropy. State-space reconstruction was first introduced by Packard et al. in 1979 and 1980 [28], and then formalized by Takens [29]. Although the reconstructions obtained via this method are not identical to the original dynamics, they are topologically identical to the full dynamics. This is remarkably useful since many important properties of dynamical systems are invariant under diffeomorphism [30].

### 1.4.1 Attractor reconstruction

Takens' embedding theorem provides a tool to reconstruct chaotic attractors from the measurement of a single degree of freedom. Imagine we have solved the set of equations for some chaotic system with three variables $(x, y, z)$ and the values for $y$ and $z$ are no longer available. Then, this theorem offers a method to reconstruct a phase-space very similar to that of the full solution through a delay-coordinate embedding generated only using the values of $x$. Specifically, one constructs $m$-dimensional reconstruction-space vectors $\mathbf{R}(t)$ from $m$ time-delayed samples of the measurements $x(t)$, such that

$$\mathbf{R}(t) = [x(t), x(t-\tau), x(t-2\tau), ..., x(t-(m-1)\tau)] \tag{1.8}$$

where $\tau$ is the so-called *delay time* and $m$ is the *embedding dimension*. At first, the statement of this theorem is striking. One might think that eliminating all but one observed degrees of freedom will eliminate information, besides, extra information is coming from the dynamics.

The main difficulty developing a phase-space reconstruction is the choice of the delay time $\tau$ and the embedding dimension $m$. In principle, the method works for almost any value of $\tau$. Only choosing a value commensurate with some aspect of the system will restrict the region of the phase-space that is sampled, like, e.g., a multiple of any orbit's period. In practice, if we consider a very small $\tau$, the $m$ coordinates in each of these vectors will be strongly correlated, and, thus, the embedded dynamics will lie close to the main diagonal of the reconstruction space. As $\tau$ is increased, that reconstruction unfolds off that subspace. For example, in the case of the Rössler system introduced in section 1.3, an embedding using a value of $\tau = 1$ will produce a figure indistinguishable from a diagonal line if its thickness is smaller than the measurement noise level. Increasing the value of $\tau$ the attractor unfolds as shown in Figure 1.11. Usually, by simply plotting the attractor for a variety of values of $\tau$ one finds a satisfactory choice for $\tau$.

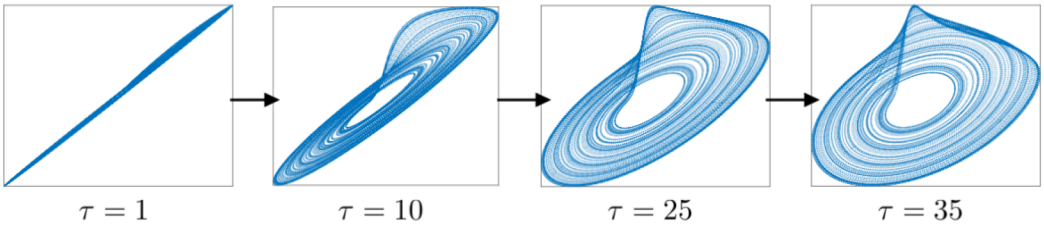



**Figure 1.11:** Attractor reconstruction for different values of the delay time $\tau$. Figure addapted from [30].

Also the choice of $m$ is not obvious. One might think that any two-dimensional manifold can be embedded within a two-dimensional real space. This is not true for a variety of examples like the Möebius strip (which is a two-dimensional manifold) that fits in a three-dimensional space, and a Klein bottle (also two-dimensional) that fits in a four-dimensional space. According to Whitney's embedding theorem, the embedding dimension $m$ has to fulfill $m \geq 2d + 1$, with $d$ being the true dimension of the underlying dynamics, to guarantee no crossings in the phase-space. In Figure 1.11, one can observe trajectory crossings that do not exist in the real attractor. This means the original attractor and the reconstructed one do not have the same topology.

In fact, there is a range of values for $\tau$ and $m$ for which the attractor embedding works reasonably well. Here, we will use the value of $\tau$ resulting from the first peak of the autocorrelation function of the time series. The optimal value of $m$ will be the smallest embedding dimension such that there is enough room to stretch the mapping to remove crossings and false nearest neighbors. However, depending on the questions being asked, crossings may be acceptable. If one is able to extract the same information ignoring these crossings, savings in computational time and data storage can be realized [31].

### 1.4.2 Recurrence Plots

A recurrence plot (RP) is a two-dimensional visualization of a sequential data set. From a $N-$point sequence $\mathbf{x}_1$, $\mathbf{x}_2$,..., $\mathbf{x}_N$, pixels located at $(i, j)$ and $(j, i)$ are black if the distance between them in the time series falls within some *threshold corridor*

$$\delta_l < ||\mathbf{x}_i - \mathbf{x}_j|| < \delta_h \tag{1.9}$$

for some appropriate choice of norm, and white otherwise [32]. Within this measure, one is capable of representing graphically all the times the phase-space trajectory of the dynamical system visits roughly the same area in the phase-space. In this manner, many properties arise. Considering a periodic trajectory of period $T$, black pixels will be separated by a distance multiple of $T$, and visible as diagonal lines. Other examples are depicted in Figure 1.12.

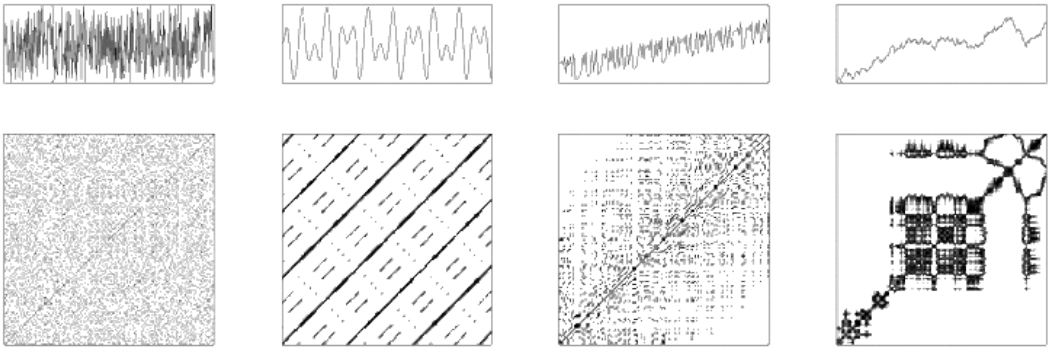


**Figure 1.12:** Typical examples of recurrence plots (top row: time series (plotted over time); bottom row: corresponding recurrence plots). From left to right: uncorrelated stochastic data (white noise), harmonic oscillation with two frequencies, chaotic data with linear trend (logistic map) and data from an auto-regressive process. By Pucicu at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=35311862.

In order to quantify the different structures appearing in the RPs, Recurrence Quantification Analysis (RQA) was developed by Zbilut and Webber Jr. [33, 34] and extended with new measures of complexity by Marwan et al. [35]. These measures are usually computed in windows along the main diagonal. This allows the study of time dependencies, and thus, it can be used for the detection of transitions. These measures can also be defined for each diagonal parallel to the main diagonal separately, enabling the study of time delays, unstable periodic orbits, and similarities between processes.

Together with the attractor reconstruction, different measures of the RQA can be computed. Particularly, the following quantities:

1. **Recurrence Rate (RR):** This measures the density of recurrence points in a RP, that is, the probability that a specific state will recur.

$$\mathrm{RR} = \frac{1}{N^2} \sum_{i,j=1}^{N} \mathbf{R}(i,j). \tag{1.10}$$

2. **Determinism (DET):** This is the percentage of recurrence points which form diagonal lines in the recurrence plot of minimal length $\ell_{min}$. It is related to the predictability of the dynamical system. Considering a white noise time-series, its recurrence plot will be full of only single dots and very few diagonal lines. Instead, a deterministic process has a recurrence plot with very few single dots but many long diagonal lines.

$$\mathrm{DET} = \frac{\sum_{\ell=\ell_{\min}}^{N} \ell\, P(\ell)}{\sum_{\ell=1}^{N} \ell P(\ell)}, \tag{1.11}$$

where $P(\ell)$ is the frequency distribution of the lengths $\ell$ of the diagonal lines.

3. **Averaged diagonal line length (L):** This measures the lengths of the diagonal and vertical lines.
$$\mathrm{L} = \frac{\sum_{\ell=\ell_{\min}}^{N} \ell\, P(\ell)}{\sum_{\ell=\ell_{\min}}^{N} P(\ell)} \tag{1.12}$$

4. **Shannon entropy (ENTR):** The probability $p(\ell)$ that a diagonal line has exactly a length $\ell$ can be estimated from the frequency distribution $P(\ell)$ with $p(\ell) = \frac{P(\ell)}{\sum_{\ell=l_{\min}}^{N} P(\ell)}$. The Shannon entropy of this probability,
$$\mathrm{ENTR} = -\sum_{\ell=\ell_{\min}}^{N} p(\ell) \ln p(\ell), \tag{1.13}$$
is a manifestation of the complexity of the deterministic structure in the system.

# Chapter 2

# Experimental Setup and Methodology

In the previous chapter, we have introduced delay-based RC as a potential platform to implement RC experimentally. Here we introduce a practical implementation of the single node delayed feedback reservoir. In this case, the RC implementation is based on a single Mackey Glass (MG) nonlinear element with delay. Choosing a MG oscillator is motivated by the existence of a well-tested design using simple electronic components. Another beneficial characteristic of this nonlinearity is its tunability, by changing components we can access optimal operating points depending on the desired task. The experimental setup was previously built and tested by M.C. Soriano. The author of this TFM has developed new numerical simulations and experiments based on previous publications [1, 36].

## 2.1 Mackey Glass Delayed Feedback Oscillator

Originally, the work of Mackey and Glass was introduced to show that a variety of physiological systems could be described in terms of simple nonlinear delay differential equations. Equation (2.1) was first introduced in a paper entitled *Oscillation and chaos in physiological control systems* in 1977 [37]. They suggested that many physiological diseases, like, e.g., apnea, could be described by Eq. (2.1) when changing parameters $\gamma$, $\beta$, $n$ and $\tau$.

$$\frac{dx}{dt} = \beta \frac{x(t-\tau)}{1+x(t-\tau)^n} - \gamma x(t), \quad \gamma, \beta, n > 0 \tag{2.1}$$

Our RC implementation is not going to use the dynamical equation but the discrete map,

$$X_{out} = \frac{C \cdot X_{in}}{1 + b^p (X_{in})^p}, \tag{2.2}$$

where $C$, $b$ and $p$ are parameters adjusted via *curve_fit* function in Python to approximate the experimental transfer function. In Figure 2.1 we show the Mackey-Glass experimental transfer function and the fitted curve where parameters were found to be $C = 2.1345$, $b = 0.0019$ and $p = 9.8212$ in Eq. (2.2).

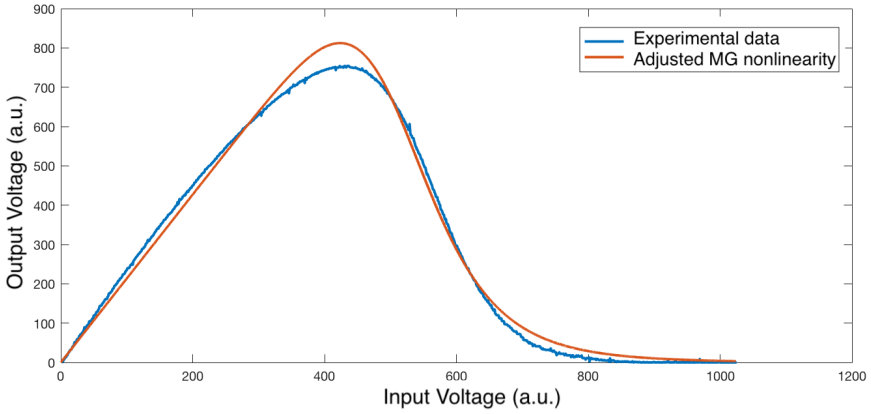



**Figure 2.1:** Mackey-Glass nonlinearity shape and numerical fit. The experimental transfer function (blue) is compared with the fit of the Mackey-Glass equation (orange). Fit parameters correspond to $C = 2.1345$, $b = 0.0019$ and $p = 9.8212$ in Eq. (2.2). Due to analogical to digital conversion, measured voltages take values between 0 and 1024 which correspond to 0 and 5 volts.

## 2.2 Experimental Setup

Our experimental setup, schematized in Figure 2.2 **a)**, consists of two main components: an analog Mackey Glass electronic circuit and the Raspberry Pi board. The analog circuit is made of electronic transistors, integrated circuits, capacitors and resistors, as depicted in Figure 2.2 **b)**. In the digital part, the Raspberry Pi controls the input signal generation and the multiplication with the input connection weights. The input signal is then converted to analog via Digital to analog converter (DAC) and goes through the analog nonlinear part. The output signal of the nonlinearity is then digitized and processed again by the Raspberry Pi.

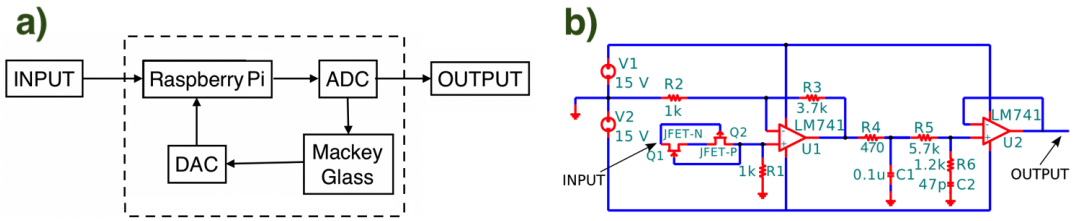



**Figure 2.2:** Panel **a)**: Schematic view of the RC implementation based on a single MG nonlinear element with delay. The time constant of the MG system is $T = 47\ \mu s$. The delay loop is implemented digitally by means of Analog to Digital and Digital to Analog Converters (ADC and DAC). The preprocessing to create the input stream $\gamma J_i(n) + \Phi$ with $\gamma$ the adjustable input gain and $\Phi$ bias voltage described in Eq. (2.3) and Eq. (2.4), and the post-processing to create the output $\mathbf{o}(n)$ are also realized digitally. Panel **b)**: Schematic representation of the MG electronic implementation.

Delay-based RC techniques explained in Section 1.2 update the reservoir considering the state of the reservoir one delay time ago. For this purpose, we need an *initialization* stage. This initialization stage is performed by feeding the reservoir with the two first points of the time-series. Each point that is fed to the reservoir leads to a change in all $D$ node states of the reservoir. For the first two points in the time-series the reservoir is updated according to Eq. (2.3).

$$r_i(n) = \frac{C \cdot (\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi)}{1 + b^p (\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi)^p} = F(\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi) \tag{2.3}$$

Please note that in equation (2.3), together with the masking procedure, an input rescaling is performed. This operation guarantees that the nodes of the reservoir are exploring a certain region of the nonlinearity. In Figure 2.3 parameters $\gamma$ and $\Phi$ define the weight and the center of this operational region respectively.

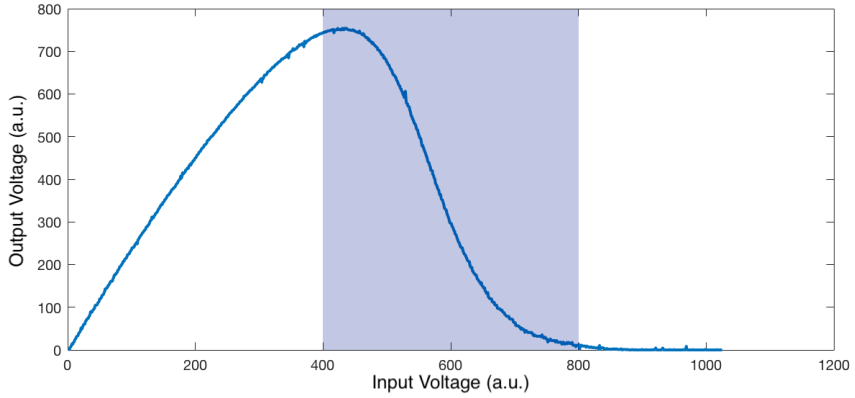



**Figure 2.3:** The experimental transfer function and the operational area. Parameters $\gamma$ and $\Phi$ define the weight and the center of the blue area respectively. Due to analogical to digital conversion, voltages take values between 0 and 1024 which correspond to 0 and 5 volts.

Once we have fed the input into the reservoir according to Eq. (2.3) we can implement the delay-based RC. According to Section 1.2.2, two different regimes arise depending on the choice of $\theta$ with respect to $T$. Here, the experimental setup is implemented in the map limit regime, i.e., the time response of the system $T \sim 47\ \mu s$ is much smaller than the time separation between the virtual nodes $\theta \sim 300\ \mu s$. In Figure 2.4, we plot the system response (blue line) for different values of the input mask function (green).

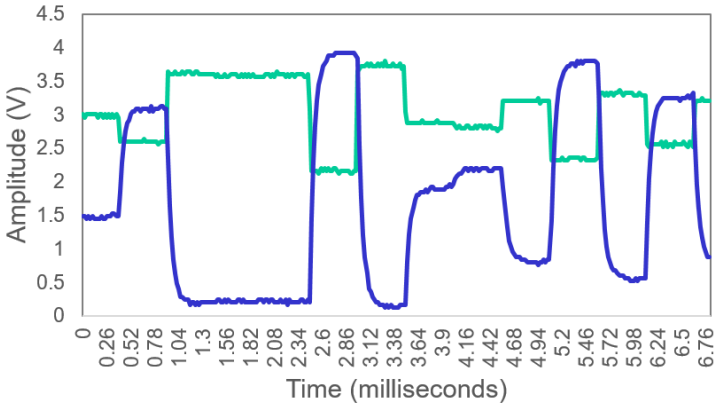



**Figure 2.4:** Experimental time-series corresponding to the response of the MG system (blue) for different values of the input mask function (green).

Under these conditions, we have introduced an interconnection structure mimicking the ring topology presented in Figure 1.7. The reservoir is then updated according to the following equations

$$\begin{aligned} r_i(n) &= \frac{C \cdot (\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi + \beta r_{i-1}(n-1))}{1 + b^p (\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi + \beta r_{i-1}(n-1))^p} \\ &= F(\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi + \beta r_{i-1}(n-1)), \quad i = 2, ..., D \\ r_1(n) &= \frac{C \cdot (\gamma \mathbf{W}_1^{in} \mathbf{x}(n) + \Phi + \beta r_D(n-2))}{1 + b^p (\gamma \mathbf{W}_1^{in} \mathbf{x}(n) + \Phi + \beta r_D(n-2))^p} \\ &= F(\gamma \mathbf{W}_1^{in} \mathbf{x}(n) + \Phi + \beta r_D(n-2))) \end{aligned} \tag{2.4}$$

where $\beta$ is the feedback scaling factor. This feedback scaling factor determines the fading memory of the system. The greater the value of $\beta$, the more important are the previous time steps in the actual state of the reservoir.

## 2.3 Methodology

In the previous section, we defined the basics of our experimental setup. Here we explain how to introduce the data in order to perform all the stages of the delay-based RC. Training and testing stages are performed in open-loop configuration, while the autonomous run uses closed-loop operation. The computation of the weights required to obtain the output signal and the autonomous operation are developed in two different programs written in C++ and compiled on the Raspberry Pi.

**Calculating weights**

1. One has to upload the string of $N$ values of the time series $\mathbf{x}(n)$ together with the mask function $M(t) = \mathbf{W}_i^{in}$ into the Raspberry Pi. This mask function is a string of $D$ values, one for each node in the reservoir.

2. For the first two points in the time-series $\mathbf{x}(n)$, the Raspberry Pi performs the rescaling input operation $\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi$.

3. The value $\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi$ is analogically converted via DAC converter.

4. This value passes through the Mackey Glass circuit producing some output $F(\gamma \mathbf{W}_i^{in} \mathbf{x}(n) + \Phi)$.

5. Then we convert it into a digital value via ADC converter.

6. Once we have completed this procedure for the first two points in the time-series $\mathbf{x}(n)$, we repeat steps 2-5 but implementing Eq. (2.4) instead of Eq. (2.3), i.e., the Raspberry Pi combines the masked input with the delayed signal, introducing the feedback. This is done by adding $\beta r_{i-1}(n-1)$ or $\beta r_D(n-2)$ to the rescaled input.

When the program finishes, we get the matrix of reservoir responses of dimension $N \times D$ (Number of input samples $\times$ Number of virtual nodes). In order to obtain the output time-series $\mathbf{o}(n)$ we perform a multiple linear regression $\mathbf{y} = \mathbf{b}\mathbf{x}$ with *regress* function in Matlab. Here, the $\mathbf{b}$ coefficients are the output weights $\mathbf{w}_i$, the variable

$\mathbf{y}$ is the input signal one step ahead $\mathbf{x}(n+1)$, and the variable $\mathbf{x}$ is the reservoir response of the system $\mathbf{r}_i(n)$. This computation is performed offline.

$$\mathbf{x}(n+1) = \sum_{i=1}^{N} \mathbf{w}_i \mathbf{r}_i(n) := \mathbf{o}(n) \tag{2.5}$$

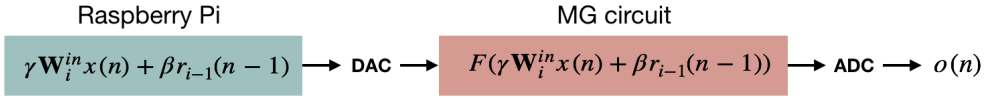
Please note that $\mathbf{o}(n)$ represents the output of the readout layer for the input signal $\mathbf{x}(n)$, but $\mathbf{o}(n)$ aims to reproduce the input signal one step ahead $\mathbf{x}(n+1)$.

**Autonomous run**

1. In this case, we have to upload the string of $N$ values of the time series $\mathbf{x}(n)$, the mask function $M(t) = \mathbf{W}_i^{in}$, and the calculated weights $w_i$ into the Raspberry Pi. The output weights constitute a string of $D$ values, one for each node in the reservoir, plus and additional bias offset.

2. For the first two points in the time-series $\mathbf{x}(n)$ we follow steps 2-5 explained above. We are not going to multiply these values by the weights, they are only used to initialize the delay-based RC. The first two points of the prediction are zeros.

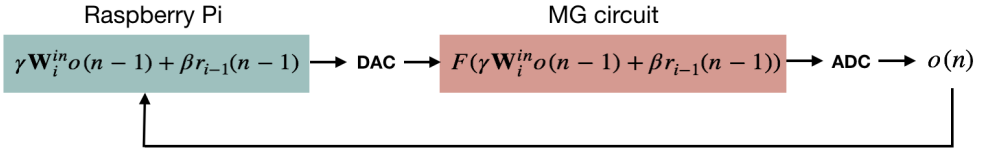
3. We need another step before closing the loop. We want our system to reproduce the dynamical behavior of the $x$ variable in the Rössler system, however, since we are starting from a certain point, we want our system to follow the same orbit for a while. To this effect, we let the system evolve according to Eq. (2.4) for the first 200 points in the time-series.




4. Finally, we are ready to close the loop. This is performed replacing $\mathbf{x}(n)$ by $\mathbf{o}(n-1)$ in Eq. (2.4).

$$\begin{aligned} r_i(n) &= \frac{C \cdot (\gamma \mathbf{W}_i^{in} \mathbf{o}(n-1) + \Phi + \beta r_{i-1}(n-1))}{1 + b^p (\gamma \mathbf{W}_i^{in} \mathbf{o}(n-1) + \Phi + \beta r_{i-1}(n-1))^p} \\ &= F(\gamma \mathbf{W}_i^{in} \mathbf{o}(n-1) + \Phi + \beta r_{i-1}(n-1)), \quad i = 2, ..., D \\ r_1(n) &= \frac{C \cdot (\gamma \mathbf{W}_1^{in} \mathbf{o}(n-1) + \Phi + \beta r_D(n-2))}{1 + b^p (\gamma \mathbf{W}_1^{in} \mathbf{o}(n-1) + \Phi + \beta r_D(n-2))^p} \\ &= F(\gamma \mathbf{W}_1^{in} \mathbf{o}(n-1) + \Phi + \beta r_D(n-2))) \end{aligned} \tag{2.6}$$

The output obtained performing a weighted sum of the response of the reservoir is fed back to substitute the teacher signal $\mathbf{x}(n)$.



These are the procedures to obtain the weights and to operate during the autonomous run. There are, however, still many questions that need to be answered. In the next sections, we will find the optimal values of $D$, $\gamma$, $\beta$ and $\Phi$.

## 2.4 Matlab Implementation

Let us consider we are dealing with an input string $\mathbf{x}(n)$ of 4000 points and a reservoir with 700 nodes. Then, as each node takes 300 $\mu s$ to pass through the MG circuit, we need $4000 \times 700 \times 300\ \mu s \sim 15$ min to obtain the reservoir response. Once the weights are computed we need another 15 minutes to collect the autonomous run. Since each experimental realization takes half an hour to be completed, we found it helpful to implement a program simulating the behavior of our experimental setup. The implemented nonlinearity is the orange line in Figure 2.1, and the noise of the experimental setup was introduced by adding some noise in the reservoir response before computing the linear regression. The strength of the noise in the reservoir was estimated from the experimental measurements.

The Matlab implementation was first tested with some values of $\beta$, $\gamma$ and $\Phi$ that were proven to work well in [36], and the values of the mask function are drawn from a Gaussian distribution. The next two figures, Fig. 2.5 and Fig. 2.6, represent the responses of the reservoir according to Eq. (2.4) with $\gamma = 60$ and $\Phi = 575$. The feedback scaling factor is $\beta = 0$ in Fig. 2.5 and $\beta = 0.4$ in Fig. 2.6. In panel **a)**, we show the output response of certain nodes of the reservoir with respect to the masked input $\mathbf{J}(n)$. In panel **b)**, the output response of certain nodes of the reservoir with respect to the values of the input variable $\mathbf{x}(n)$.

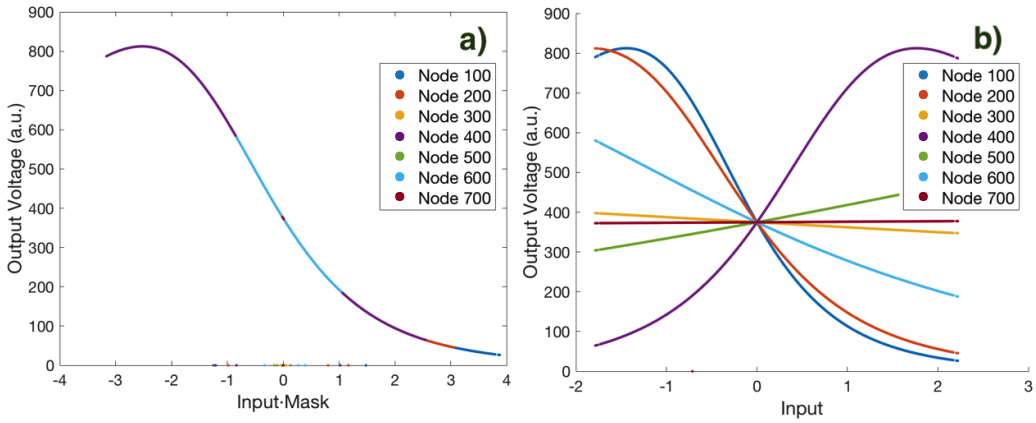



**Figure 2.5:** Output voltage of selected nodes of the reservoir when $\beta = 0$, $\gamma = 60$ and $\Phi = 575$ in (2.4). In panel **a)**, the output voltage is plotted against the value of $\mathbf{J}(n)$. In panel **b)**, we plot the output voltage with respect to the values of the input variable $\mathbf{x}(n)$. The input signal has 4000 points and the reservoir size is $D = 700$.

Figures 2.5 and 2.6 are very useful to understand what part of the nonlinearity is being visited by the different nodes of the reservoir. For instance, if we focus on *node 700* in Figure 2.5 **a)**, we notice that the product $\mathbf{J}(n) = \mathbf{x}(n) \cdot \mathbf{M}(n)$ leads to a small interval in the middle of the used nonlinear section. This results from the small value of the mask function for that virtual node. In contrast, another point of the reservoir, e.g., *node 400*, has a larger value of the mask function, covering a more extended interval of the nonlinearity.

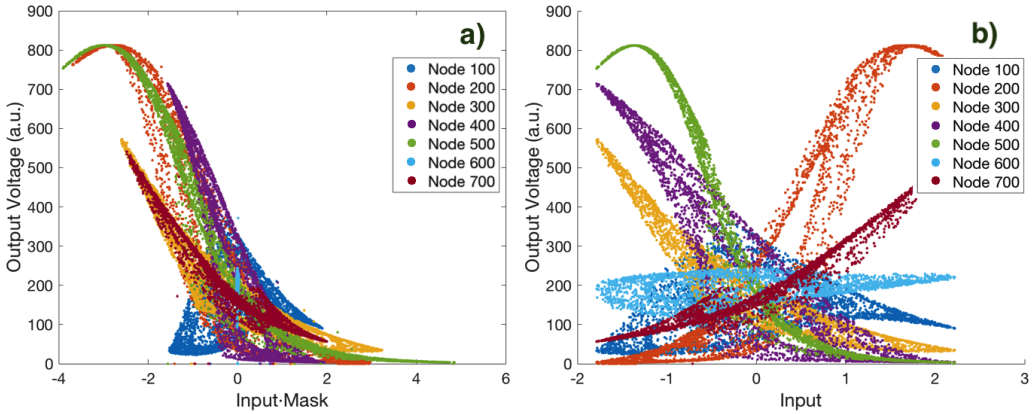



**Figure 2.6:** Output voltage of selected nodes of the reservoir when $\beta = 0.4$, $\gamma = 60$ and $\Phi = 575$ in (2.4). In panel **a)**, the output voltage is plotted against the value of $\mathbf{J}(n)$. In panel **b)**, we plot the output voltage with respect to the values of the input variable $\mathbf{x}(n)$. The input signal has 4000 points and the reservoir size is $D = 700$.

In Figure 2.5 **b)** we present the output voltage only with respect to the values of the input variable $\mathbf{x}(n)$. With this representation, it becomes evident the variety of responses of the reservoir. When increasing the value of the feedback scaling factor $\beta$, previous states of the reservoir have more importance in the actual state of the reservoir. Graphically, the curves broaden (see Fig. 2.6). The greater the value of $\beta$, the wider the region explored by the nonlinear nodes, and thus, the more the curves broaden.

In addition to parameters $D$, $\gamma$, $\beta$ and $\Phi$, we now need to explore how much noise we need to add to the reservoir responses in order to obtain quantitatively similar results in the experimental setup and in the numerical Matlab implementation. This noise is added before performing the linear regression to obtain output weights. If no noise is added, then the numerically NRMSE is lower than the one obtained by the experimental setup. Thus, we are now calibrating the numerical simulations to obtain results comparable to the experimental ones. In the next section, we will find the right amount of noise while searching for the optimal number of nodes in the reservoir $D$.

## 2.5 Number of Nodes in the Reservoir

In this section, we compare experimental and numerical results. In order to test our system, we determine the NRMSE for different masks and different number of nodes in the reservoir. The input signal and the masks are chosen to be the same in the experiment and in the Matlab program. Figure 2.7 illustrates the NRMSE during training and testing stages. The average and the standard deviation are obtained from 10 different masks.

In Figure 2.7 we show the experimental results and the numerical results when 0.25% of Gaussian noise is added to the reservoir response in training and testing stages. The amount of noise implemented in the Matlab program was chosen to reproduce the experimental NRMSE during the testing stage as much as possible.

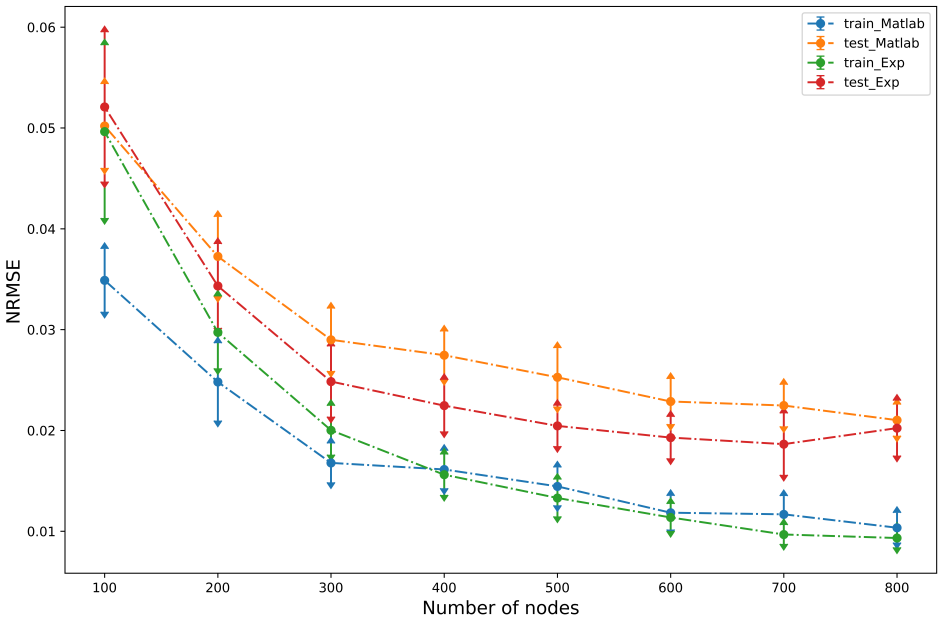



**Figure 2.7:** Normal Root Mean Square Error (NRMSE) vs. different number of nodes in the reservoir. Green and blue represent the NRMSE for the testing stage in the experiment and the Matlab program respectively. Red and orange stand for the NRMSE during the testing stage.

Note that the results for the numerical simulations follow the same trend of experimental results, however, the NRMSE of the experiments and the simulations in the training stage diverge when the reservoir has less than 300 nodes. This may be due to the existence of other sources of noise in the experimental setup. For instance, we do not consider in the numerical simulations the quantization noise that originates from the discretization of ADC and DAC converters. The influence of other sources of noise or experimental fluctuations seems to increase when the number of nodes decreases, resulting in the mismatch observed when the number of nodes is less than 300. As seen in Fig. 2.7, the optimal value for the number of nodes is $D = 700$ as it is the one with the lowest NRMSE in the testing stage.

## 2.6 Optimal Operating Region

It is commonly assumed that the lower the value of the $NRMSE$, the better the performance of the autonomous operation. Thus, we run numerical simulations to find the optimal operating region $(\gamma, \beta, \Phi)$ with the lowest $NRMSE$ during the testing state when the number of nodes is $D = 700$. Figure 2.8 **a)** shows the numerical results of the $NRMSE$ in the prediction of the x variable of the Rössler system during testing stage as a function of the feedback strength $\beta$ and input scaling $\gamma$. The phase for which the smallest $NRMSE$ during testing stage was found is plotted in Fig. 2.8 **b)**.

The operational region with the lowest Normal Root Square Mean Error is found when $\gamma \in [60, 80]$, $\beta \in [0.3, 0.4]$ and $\Phi \in [500, 600]$. The main feature that distinguishes this region is that the value of the bias voltage $\Phi \sim 550$ is placed in the middle of the decreasing slope. Moreover, the optimal values for $\gamma$ and $\beta$ allow the nodes to explore all the slope of the nonlinearity with a certain width.

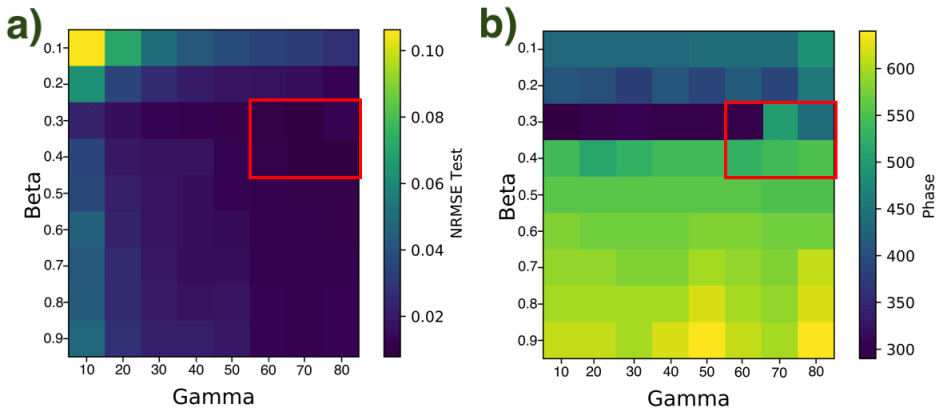



**Figure 2.8:** Numerical results for **a)** the NRMSE during testing stage and **b)** the phase $\Phi$ as a function of the feedback strength $\beta$ and input scaling $\gamma$. The red box defines the optimal operating region. The input signal has $4000$ points and the reservoir size is $D = 700$.

In the next chapter we will show the experimental time-series obtained when the experimental setup works within the range of the optimal conditions presented above, i.e., $\gamma \in [60, 80]$, $\beta \in [0.3, 0.4]$, $\Phi \in [500, 600]$ and $D = 700$.

# Chapter 3

# Results and Discussion

In this chapter we present the experimental results, as well as analyze and discuss them. First, we characterize the results for the one-step-ahead prediction in open-loop operation. Then, we realize autonomous operation in two different ways. First, using only the $x$ variable of the Rössler system time-series as the input signal, and second, applying input noise to the time-series before training the weights of the system. Attractor reconstruction and recurrence quantification analysis are performed in both cases to illustrate and quantify the similarity of the trained autonomous oscillator with the original Rössler system. From the optimal operating region defined in the previous section, we choose the operating point as $(\gamma, \beta, \Phi) = (60, 0.4, 575)$ and characterize the performance for a variety of input masks.

## 3.1 One-step-ahead prediction

In this section we present experimental results for the one-step-ahead prediction of the $x$ variable of the Rössler chaotic time-series. According to the method described in Sec. 2.3, we perform a first experiment to calculate the output weights of the system. Once we get the matrix of reservoir responses of dimension $4000 \times 700$, we split the output into training and testing stages. Exactly, the reservoir responses corresponding to points $200 - 3600$ of the input signal $\mathbf{x}(n)$ are used to calculate the output weights via linear regression, as indicated in Eq. 2.5. The reservoir responses corresponding to points $3650 - 4000$ of $\mathbf{x}(n)$ are used in the testing stage.

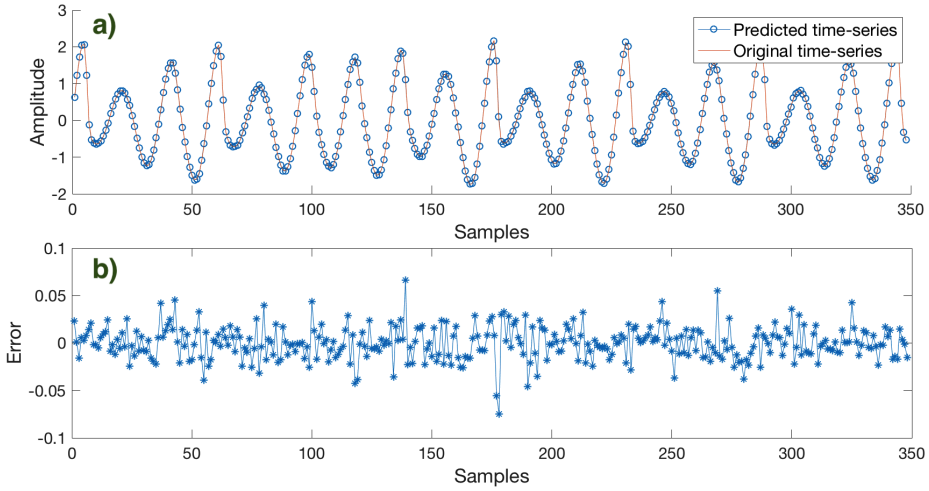



**Figure 3.1:** Experimental results for the nonlinear time-series prediction of the $x$ variable of the Rössler chaotic system during the testing stage when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$ and $D = 700$. Panel a): Original time-series (orange line) and prediction (blue circles). Panel b): Prediction error as the difference between original and predicted time-series.

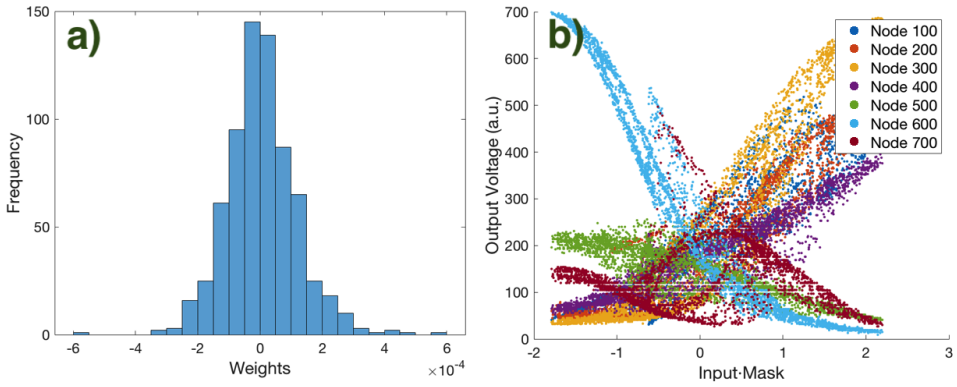



**Figure 3.2:** Experimental results for the nonlinear time-series prediction of the $x$ variable of the Rössler chaotic system when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$ and $D = 700$. Panel a): Histogram of the output weights. Panel b): Different output voltage responses of selected nodes in the reservoir induced by the input signal $\mathbf{x}(n)$.

Figure 3.1 **a)** shows an interval of the $x$ variable of the original Rössler time-series during the testing stage together with the prediction of the echo state network. The error between the predicted and the original time-series has an order of magnitude $10^{-1}$, as illustrated in 3.1 **b)**. The output weights computed for each of the nodes are of the order $10^{-4}$, as shown in Fig. 3.2 **a)**. In Fig. 3.2 **b)** we plot the node nonlinear response of some nodes of then reservoir induced by the input signal. This figure is the experimental version of Fig. 2.6 **b)**. Here, the node nonlinear response is more spread than in numerical results due to the noise in the experiments.

We have performed experiments for 6 different masks. Figures 3.1 and 3.2 are representative of all the masks, since all of them present the same overall shapes. However, each one has a slightly different $NRMSE$ value. In Table 3.1 we indicate the $NRMSE$ during training and testing stages.

| **Mask** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **NRMSE (Training)** | 0.0118 | 0.0108 | 0.0149 | 0.0121 | 0.0133 | 0.0108 |
| **NRMSE (Testing)** | 0.0150 | 0.0159 | 0.0172 | 0.0164 | 0.0175 | 0.0154 |

**Table 3.1:** NRMSE for the different masks in the training and testing stages when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$, $D = 700$ and $N = 4000$.

One of the questions this Master thesis aims to answer is: *Is the mask with the lowest NRMSE during the testing stage allowing for autonomous operation that mimics the original dynamics?* and, *Is it the the best one?* In the next section we will address these questions.

## 3.2 Autonomous Operation

The one-step-ahead prediction is a good starting point, since we aim to generate chaotic time-traces in an autonomous manner. In this section, we present the experimental results for the autonomous operation, computed following the steps explained in Sec. 2.3 when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$. The response obtained from the autonomous operation can be divided into short-term and long-term prediction. This can be viewed as prediction of *weather* and *climate* of the dynamics. These metaphors, which illustrate the different tasks very well, were introduced by Ott's

research group in [38]. The *weather* is composed by the first time-steps just after closing the loop, i.e., how the autonomously generated signal follows the original trajectory. When we talk about the *climate*, we are no longer interested in how the generated signal follows a certain trajectory, but how it reproduces the general features of the dynamics of the system.

### 3.2.1 Weather-like Forecasting

Here we analyze the short-term prediction of the autonomous operation. Figure 3.3 is an example of how autonomously generated signals diverge from the original one. The first 20 samples correspond to the open-loop operation in which the error is of order $10^{-1}$. We present the results of the autonomous operation for one of the masks in Fig. 3.3, in which we observe that trajectories go together for $\sim 70$ samples and then they diverge. Note that there is a maximum error corresponding to the size of the chaotic attractor.

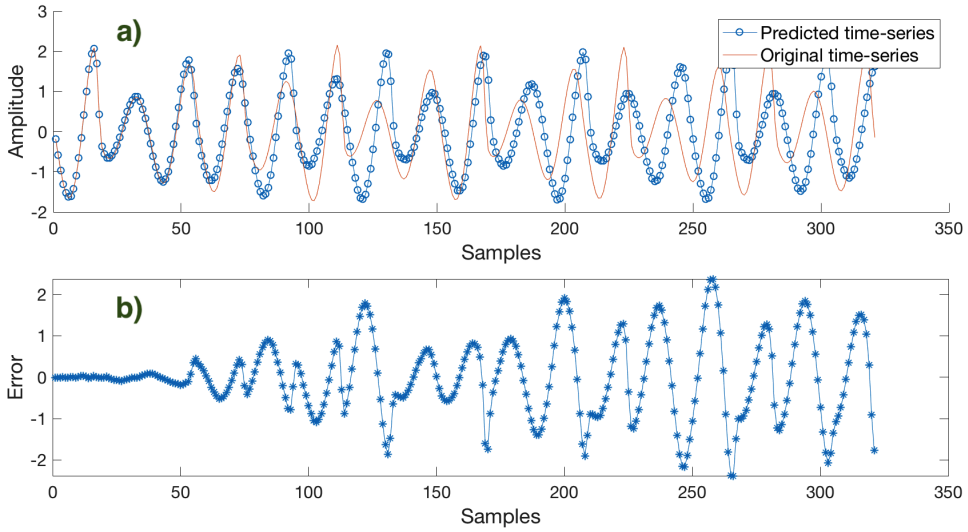



**Figure 3.3:** Experimental results for the nonlinear time-series prediction of the $x$ variable of the Rössler chaotic system during the autonomous operation when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$. Panel a): Original time-series (orange line) and prediction (blue circles). Panel b): Prediction error as the difference between original and predicted time-series.

The study of the 6 different masks leads to different short-term predictions. Table 3.2 shows the different number of samples for which the predicted signal follows the original one to a large extent.

| **Mask** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $\sim$ **samples predicted** | 55 | 90 | 60 | 60 | 150 | 60 |

**Table 3.2:** Short-term prediction of different masks when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$, $D = 700$ and $N = 4000$.

These results show that remarkably, the mask with the lowest $NRMSE$ during the testing stage does not imply the best short-term prediction. What is more, for this particular set of masks, the one with the best short-term prediction is the one with the largest $NRMSE$ in the open-loop prediction!

Lowest $NRMSE$ in the open-loop prediction $\not\Rightarrow$ Best weather forecasting

### 3.2.2 Climate-like Replication

In this section, we illustrate the capability of our experiment to replicate the *climate* of the $x$ variable of the Rössler system. There are not many investigations devoted to analyzing long-term prediction, so we decided to illustrate and quantify it by a set of different approaches: frequency power spectra, phase-space reconstruction of the chaotic Rössler attractor, and recurrence quantification analysis.

#### Frequency spectra

One of the features of deterministic time-series data is that they are ordered to some extent. Therefore, we can take advantage of this and find temporal patterns. Here, we calculate the frequency spectra of the original time series and the one obtained from trained autonomous operation. In Figure 3.4 we show the frequency spectrum of the original Rössler chaotic attractor (orange) and the frequency spectrum computed for the autonomous run (blue). Note that low frequencies are well captured by the generated signal, but those in the tail diverge from the original time-series.

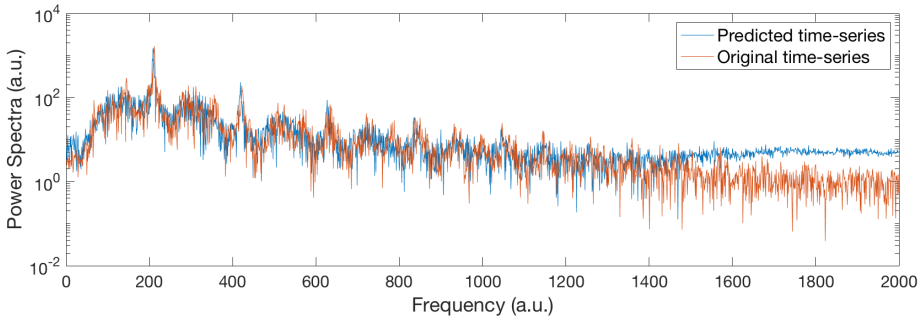



**Figure 3.4:** Frequency spectra of the $x$ variable of the Rössler chaotic attractor. The orange line corresponds to the original time-series and the blue line to the autonomously generated time-series when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$, $D = 700$ and $N = 4000$.

The quantitative analysis is performed via *corrcoef* function in Matlab. This function returns the Pearson[1] correlation coefficient of two random variables, measuring their linear dependence. Thus, in Table 3.3 we indicate the Pearson correlation coefficient between the predicted and the original Rössler time-series.

| **Mask** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Corrcoef** | 0.7937 | 0.7934 | 0.7304 | 0.7612 | 0.7528 | 0.7954 |

**Table 3.3:** Correlation coefficient between the predicted and the original frequency spectra for an input signal $\mathbf{x}(n)$ of 4000 points, when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$ and $D = 700$.

Again, the mask with the lowest $NRMSE$ in the open-loop prediction does not correspond to the best frequency spectrum reproduction.

Lowest $NRMSE$ in the open-loop prediction $\not\Rightarrow$ Best frequency spectrum reproduction

[1] If each variable has N scalar observations, then the Pearson correlation coefficient is defined as: $\rho(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{A_i-\mu_A}{\sigma_A}\right)\left(\frac{B_i-\mu_B}{\sigma_B}\right)$ where $\mu_A$ and $\sigma_A$ are the mean and standard deviation of $A$, respectively, and $\mu_B$ and $\sigma_B$ are the mean and standard deviation of $B$.

### Phase-Space Reconstruction of the Chaotic Attractor

The embedding theorem explained in Sec. 1.4.1 is used for the phase-space reconstruction of the chaotic attractor. For aesthetic reasons, we have chosen $\tau$ as the time between 17 samples. For all the masks, the generated attractor keeps the main phase-space structure of the original one, as illustrated in Fig. 3.5. Even some fine structure is reproduced in the autonomously generated attractor, however a bit less defined than in the phase-space reconstruction from the original time-series. Some of these minor differences can be attributed to the finite signal-to-noise ratio of the experimental system, others can be improved by training the reservoir computer with an input signal with noise, as we will see in Section 3.3.

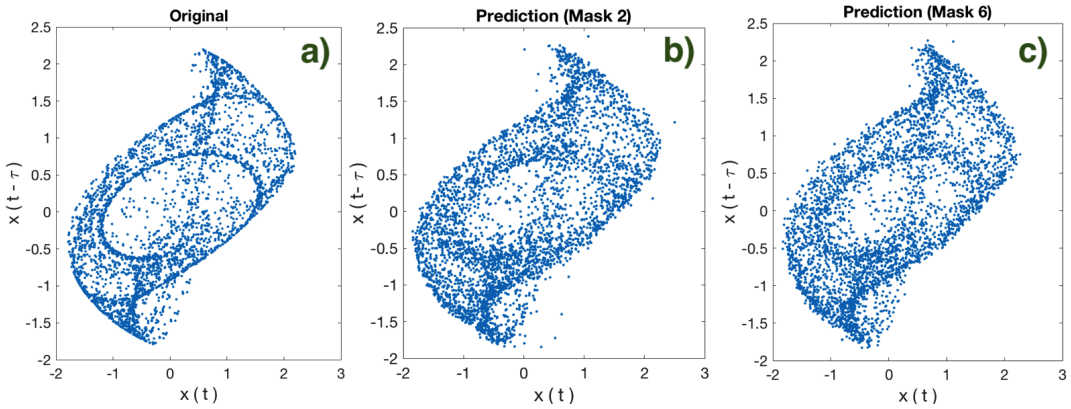



**Figure 3.5:** Phase-space representation of the Rössler chaotic attractor for an input string of 4000 points, $D = 700$ and $(\gamma, \beta, \Phi) = (60, 0.4, 575)$ in the plane $[x(t), x(t-\tau)]$ when $\tau = 17$. The different panels show the phase-space reconstruction for the original time-series (**a)**) and for the autonomously generated signal for different masks (**b)** and **c)**).

The phase-space reconstruction is a visual method in which we can easily recognize if the predicted time-series is mimicking the original time-series. However, this is a qualitative analysis and not a quantitative one. For this reason, in this Master thesis we propose other methods to characterize the performance of our experiments via recurrence analysis. The next section covers different measures of the recurrence quantitative analysis (RQA).

### Recurrence Analysis

In this section we use recurrence quantification analysis (RQA) to perform nonlinear data analysis for comparing autonomous operation and original time-series in more detail. In particular, we calculate recurrence rate (RR), determinism (DET), averaged diagonal line length (L) and Shannon entropy (ENTR) for both, the original and the predicted time-series for 6 different masks. Additionally, in order to calculate which mask performs the best recurrence analysis, we introduce a *Divergence* measure resulting from the sum of the absolute values differences between the original and the predicted quantities mentioned above:

$$\begin{aligned} \text{Divergence} = &\left|\frac{RR(o) - RR(p)}{RR(o)}\right| + \left|\frac{DET(o) - DET(p)}{DET(o)}\right| \\ &+ \left|\frac{L(o) - L(p)}{L(o)}\right| + \left|\frac{ENTR(o) - ENTR(p)}{ENTR(o)}\right| \end{aligned} \tag{3.1}$$

where $(o)$ stands for the original and $(p)$ for the predicted time-series.

| | RR | DET | L | ENTR | Divergence |
|---|---|---|---|---|---|
| **Original** | **0.0252** | **0.7970** | **4.2701** | **1.1665** | – |
| Mask 1 | 0.0224 | 0.7790 | 3.9726 | 1.1315 | 0.2334 |
| Mask 2 | 0.0197 | 0.7663 | 3.9846 | 1.1100 | 0.3721 |
| Mask 3 | 0.0242 | 0.7664 | 3.7168 | 1.1041 | 0.2611 |
| Mask 4 | 0.0194 | 0.7802 | 3.9047 | 1.1585 | 0.3960 |
| Mask 5 | 0.0203 | 0.7576 | 3.7260 | 1.1133 | 0.3437 |
| Mask 6 | 0.0237 | 0.7654 | 3.8940 | 1.1046 | 0.2403 |
| **Average** | $\mathbf{0.022 \pm 0.002}$ | $\mathbf{0.769 \pm 0.009}$ | $\mathbf{3.87 \pm 0.12}$ | $\mathbf{1.12 \pm 0.02}$ | $\mathbf{0.31 \pm 0.07}$ |

**Table 3.4:** Recurrence rate, determinism, averaged diagonal length and Shanon entropy for different masks for an input string of 4000 points, $D = 700$ and $(\gamma, \beta, \Phi) = (60, 0.4, 575)$.

Table 3.4 shows the measures of the RQA for different masks. As an example, we plot in Figure 3.6 the recurrence map for the original and the predicted time-series with Mask 1. We can recognize the same patterns in left and right panels, however, some sub-diagonal and super-diagonal lines are slightly displaced or diffused. The lines above and below the main diagonal are related to the number of times the system visits a certain region. This is another representation of the phase-space attractor reconstruction illustrated in Fig. 3.5. In that case, the whole attractor seemed diffused, here, the diagonal lines are those that are diffused.

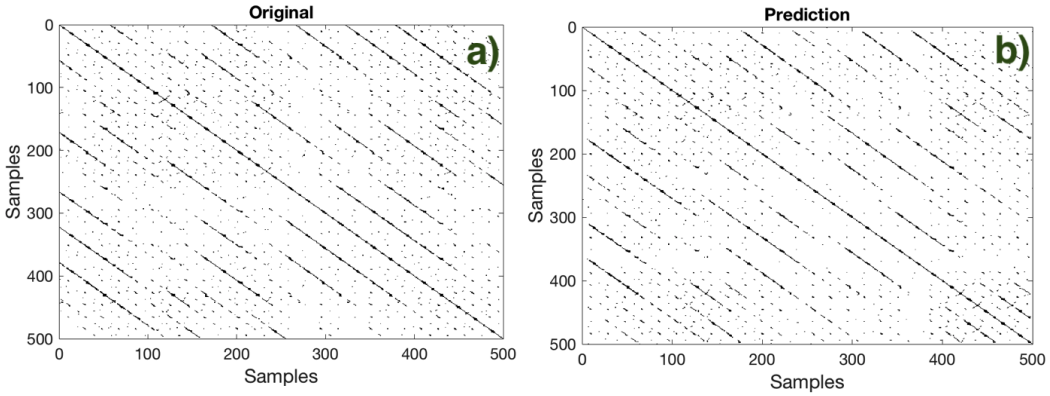



**Figure 3.6:** Recurrence plots for an input string of 4000 points, $D = 700$ and $(\gamma, \beta, \Phi) = (60, 0.4, 575)$. Panel **a)**: Original time-series. Panel **b)**: Signal autonomously generated by the echo state network with output feedback.

Also for these measures, the mask with the lowest $NRMSE$ in the open-loop prediction does not correspond to the best recurrence quantification analysis (RQA) at least in terms of our *Divergence* measure.

$$\text{Lowest } NRMSE \text{ in the open-loop prediction } \not\Rightarrow \text{ Best RQA (Divergence)}$$

## 3.3 The Role of Input Noise for the Autonomous Operation

In the previous sections we have studied the autonomously generated signals. We have shown that the autonomous operation can reproduce the dynamics of the original system. In this section, we want to go one step further and improve the correspondence of the climate of the dynamics using the previous measures: frequency spectra, phase-space attractor reconstruction, and recurrence analysis. To achieve this, we suggest a technique introduced in the field of deep neural networks.

Since their revival in 2006 by Hinton and co-workers in their seminal paper [39], deep neural networks have outperformed classifiers on many benchmark datasets related to images, speech, and text-based applications. However, Szegedy et al. [40] demonstrated that even the neural networks that have very good generalization properties and near human performance in classification tasks are not robust to perturbation in the dataset [41]. To solve this problem, it has been proposed that neural networks can be made more robust against noise by training them with the help of additive noise [42]. In our case, the experimental prediction will inevitably be subject to small fluctuations. Thus, by training the experiment with a perturbed input signal, we make it more likely that the autonomous operation will be less sensitive to small deviations in the prediction.

In this Master thesis, we propose to train the reservoir using the input signal with some additional noise. In particular, we suggest adding Gaussian white noise introducing a 2% error. We explore, whether that input noise injection results in a more precise climate prediction, improving the robustness of the RC operation. The first difference we observe, is that the distribution of the output weights changes when the system is trained with the perturbed input. As depicted in Figure 3.7, the output weight distribution has a less pronounced peak and a wider base than the one shown in Fig. 3.2 **a)**, nevertheless, the order of magnitude is still $10^{-4}$. The next sections study the effects of these output weights in the climate replication.

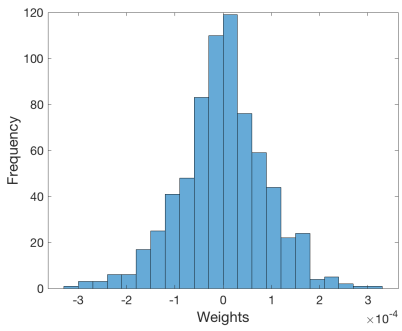



**Figure 3.7:** Histogram of output weights for an input string of 4000 with additional noise, $D = 700$ and $(\gamma, \beta, \Phi) = (60, 0.4, 575)$.

### Frequency Spectra

In Figure 3.8 we exhibit the frequency spectrum of the x variable of the original Rössler chaotic attractor (orange) and the frequency spectrum with the autonomously generated signal adding input noise (blue). In contrast to the results obtained in Fig. 3.4, we observe that the generated spectrum looks more similar to the one generated by the original time-series even for high frequencies. This is also reflected in the quantitative comparison, as detailed in Table 3.5.

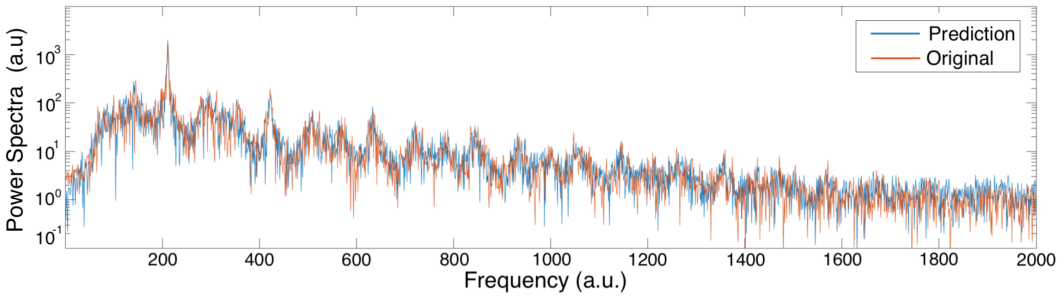



**Figure 3.8:** Frequency spectra of the $x$ variable of the Rössler chaotic attractor. The orange line corresponds to the original time-series and the blue line to the autonomously generated time-series when $(\gamma, \beta, \Phi) = (60, 0.4, 575)$, $D = 700$ and the input string $N = 4000$ has additional noise.

| **Mask** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Corrcoef** | 0.8295 | 0.8258 | 0.8133 | 0.8148 | 0.8203 | 0.8177 |

**Table 3.5:** Correlation coefficient between the predicted and the original frequency spectra. Here, the predicted frequency spectrum is obtained for an input noise.

We can see that for every mask the correlation coefficient has increased in comparison with that obtained in previous Section 3.2.2 (see Tab. 3.3). As a consequence, we can conclude that regardless of the mask used, the frequency spectra are better reproduced when we add noise to the input signal.

### Phase-Space Reconstruction of the Chaotic Attractor

Figure 3.9 exhibits the phase-space representation of the Rössler chaotic attractor in the plane $[x(t), x(t-\tau)]$ when $\tau = 17$. In the middle (panel **b)**), we plot the phase-space representation for the original time-series, and, on both sides, the phase-space reconstruction for the autonomously generated signal when the output weights are obtained for the input signal $\mathbf{x}(n)$ (panel **a)**), and when the output weights are obtained for the input signal with additional noise (panel **c)**).

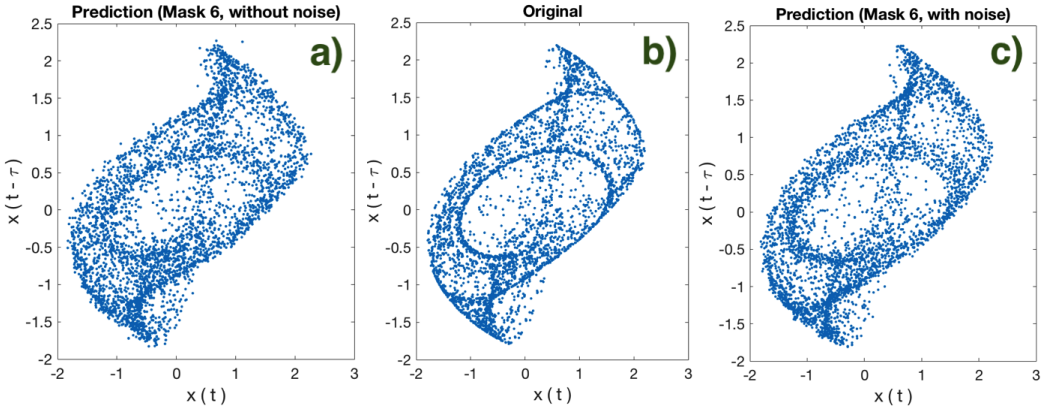



**Figure 3.9:** Phase-space representation of the Rössler chaotic attractor for an input string of 4000 points, $D = 700$ and $(\gamma, \beta, \Phi) = (60, 0.4, 575)$ in the plane $[x(t), x(t-\tau)]$ when $\tau = 17$. The different panels show the phase-space reconstruction for the original time-series (**b)**), and for the autonomously generated signal with noise added to the input signal (**c)**) or without noise added to $\mathbf{x}(n)$ (**a)**).

The conditions are the same in panels **a)** and **c)**, the operating point is $(\gamma, \beta, \Phi) = (60, 0.4, 575)$, the number of nodes in the reservoir is $D = 700$ and the length of the input string is $N = 4000$. The only thing that changes is the additional noise in the input signal. This change, that improved the correlation coefficient as seen in Table 3.5, also improves the phase-space attractor reconstruction visually.

### Recurrence Analysis

In this section we show the results of the RQA when noise is added to the input signal. The Table 3.6 summarizes the values of recurrence rate (RR), determinism (DET), averaged diagonal line length (L) and Shannon entropy (ENTR) of the different masks.

| | **RR** | **DET** | **L** | **ENTR** | **Divergence** |
|---|---|---|---|---|---|
| **Original** | **0.0252** | **0.7970** | **4.2701** | **1.1665** | – |
| Mask 1 | 0.0249 | 0.7919 | 4.1271 | 1.1586 | 0.0585 |
| Mask 2 | 0.0233 | 0.7889 | 3.9802 | 1.1534 | 0.1647 |
| Mask 3 | 0.0222 | 0.7854 | 3.8538 | 1.1603 | 0.2443 |
| Mask 4 | 0.0176 | 0.7741 | 4.3217 | 1.1481 | 0.3582 |
| Mask 5 | 0.0247 | 0.7756 | 4.0732 | 1.1281 | 0.1257 |
| Mask 6 | 0.0262 | 0.7679 | 4.0311 | 1.1019 | 0.1875 |
| **Average** | $\mathbf{0.023 \pm 0.003}$ | $\mathbf{0.781 \pm 0.009}$ | $\mathbf{4.06 \pm 0.16}$ | $\mathbf{1.14 \pm 0.02}$ | $\mathbf{0.19 \pm 0.11}$ |

**Table 3.6:** Recurrence rate, determinism, averaged diagonal length and Shanon entropy for different masks when input noise is added to train the weights of the system.

Table 3.6 reproduces the same quantities of the RQA shown in Tab. 3.4. Here, we observe that the calculated values approximate better the original quantities. The value of the divergence decreases on average and also for each of the masks. This is a quantitative additional evidence of the improvement achieved in the climate replication when reproducing the dynamics of the Rössler system if noise is added to the input signal.

# Conclusions and Outlook

Since our modern societies are becoming increasingly interested in weather forecasting, mathematical finances, or any other discipline aiming to predict the future behavior of a system, many machine learning techniques have emerged trying to offer a solution. In particular, time-series forecasting using reservoir computing techniques has been successfully investigated in modeling [3] and experiments [2].

The prediction of time-series is particularly interesting when working with chaotic systems. There are papers of high interest on this topic, e.g. [43], that have trained one dynamical system to emulate another, obtaining results that aim at cracking chaos based cryptography. Here, we perform experiments on a hybrid system mainly composed by an analog Mackey Glass nonlinearity and a Raspberry Pi board to reproduce the dynamics of the chaotic Rössler system. We demonstrate that the autonomously generated time-series has a comparable power spectrum and an attractor with similar geometry to the original system. Also, we use recurrence quantification analysis to perform a nonlinear data analysis for comparing autonomous operation and original time series in more detail.

This TFM has contributed to the reproduction of chaotic dynamics in two different ways. Firstly, we have obtained a climate-like reproduction of the $x$ variable by injecting the input signal $\mathbf{x}(n)$ into our reservoir computer using different random realizations of the input masks. These results were used to show that using the mask with the lowest value of $NRMSE$ during open-loop operation does not necessarily result in the best autonomous generation of the time-series. Secondly, we have improved the reproduction of the climate of the nonlinear dynamics by means of perturbing the original input data in the training stage. We have achieved these results adopting a technique introduced in the field of deep neural networks. The additional noise applied to the input signal results in a better long-term reconstruction of the attractor. We were able to demonstrate this quantitatively by comparing the correlation coefficient of the spectra and the recurrence quantification analysis for both approaches. Moreover, the improvement could also be appreciated visually in a phase-space reconstruction.

# Future Perspectives

This Master thesis opens many interesting questions. One key question is related to the amount of noise that needs to be added to the input signal during training to obtain autonomous operation, which is optimal in certain ways. In this TFM we have shown, using different approaches, the improvement achieved by means of perturbing the original input data in the training stage, however, due to time restrictions, we were not able to perform experiments for different amounts of noise.

There are also other interesting questions to solve. Regarding specific aspects of the comparison between numerical and experimental results, the Matlab program can be optimized by implementing the discretization due to ADC and DAC converters. In that case, we will have to fit only the source of noise intrinsic to the analog circuit. In addition to this, a detailed study of different operating points in the experiment is of high interest order to understand if the use of the lowest values of the $NRMSE$ during the testing stage in the open-loop operation reproduces the climate of the chaotic dynamics better. Regarding more general aspects, there are also different connection topologies between the nodes in the reservoir that can be implemented. Here, we have focused in a ring topology ($\alpha$=1) as introduced in Fig. 1.7 , however, other values of $\alpha$, or extra delay connections (see Fig. 3.10), would lead to changes in the fading memory of the reservoir that may result in a better reproduction of the dynamical behavior of the chaotic system via the autonomously generated signal.

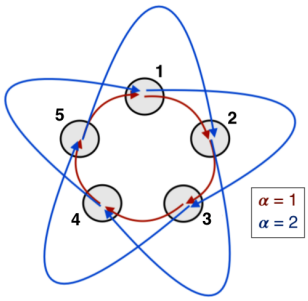



**Figure 3.10:** Schematic interaction graph of the virtual nodes with two delay connections.

This TFM represents an interesting starting point for future investigations at the interface of complex dynamical systems, machine learning and information processing, promising fundamental insights, as well as opening the perspective for novel applications.

# Bibliography


[1] L. Appeltant, M.C. Soriano, G. Van der Sande, et al. "Information processing using a single dynamical node as complex system". *Nat Commun*:2-468, 2011.

[2] P. Antonik, M. Haelterman, and S. Massar. "Brain-Inspired Photonic Signal Processor for Generating Periodic Patterns and Emulating Chaotic Systems," *Physical Review Applied*, 7:054014, 2017.

[3] J. Pathak, H. Brian, et al. "Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach." *Physical Review Letters* 120: 024102, 2018.

[4] R. Vicente. Course on "Deep Learning from a Statistical Physicist perspective". *GEFENOL Summer School*, IFISC, Palma de Mallorca, Spain, July 2-13, 2018.

[5] H. Jaeger. "Short term memory in echo state networks". *German National Research Center for Information Technology, Technical Report GMD Report 152*, 2001.

[6] W. Maass, T. Natschläger, and H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations". *Neural Computation*, 14(11):2531–2560, 2002.

[7] J. J. Steil."Backpropagation-decorrelation: online recurrent learning with O (N) complexity." *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on.* Vol. 2. IEEE, 2004.

[8] Y. Paquot, F. Duport, A. Smerieri et al. "Optoelectronic reservoir computing". *Scientific Reports*, 2: 287, 2012.

[9] M. Salehi, E. Abiri, and L. Dehyadegari. "Nanophotonic reservoir computing for noisy time series classification". *International Journal of Computer and Electrical Engineering*, 6(3):240, 2014.

[10] H. Burgsteiner. "Training networks of biological realistic spiking neurons for real-time robot control". In *Proc. of EANN*: 129–136, 2005.

[11] J. Butcher, D. Verstraeten et al. "Reservoir computing and extreme learning machines for non-linear time-series data analysis". *Neural Networks*, 38:76–89, 2013.

[12] M.L. Alomar, V. Canals, A.Morro, A. Oliver, and J.L. Rossello. "Stochastic hardware implementation of liquid state machines". In *Neural Networks (IJCNN), 2016 International Joint Conference on*, 1128–1133. IEEE, 2016.

[13] P.J. Kindermans, P. Buteneers, D. Verstraeten, B. Schrauwen. "An uncued brain-computer interface using reservoir computing". In *Workshop: Machine Learning for Assistive Technologies*, 2010.

[14] A. Apostolos, J. Bueno, and I. Fischer. "Photonic machine learning implementation for signal recovery in optical communications." *Scientific Reports* 8: 8487, 2018.

[15] H. Jaeger and H. Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication", *Science* 304, 78, 2004.

[16] The 2006/07 Forecasting Competition for Neural Networks & Computational Intelligence, http://www.neuralforecasting\unhbox\voidb@x\hbox{-}competition.com/NN3/, 2006.

[17] T. Erneux. "Applied Delayed Differential Equations" *Springer Science + Business Media* , 2009 .

[18] E. Schöll, and H.G. Schuster. "Handbook of Chaos Control" 2nd edn, *Wiley-VCH*, 2008.

[19] M.C Soriano, J. García-Ojalvo, C.R. Mirasso and I. Fischer. "Complex photonics: Dynamics and applications of delay-coupled semiconductors lasers". *Reviews of Modern Physics*, 85(1), 42, 2013.

[20] X. Porte, M.C. Soriano, and I. Fischer. "Similarity properties in the dynamics of delayed-feedback semiconductor lasers." *Physical Review A* 89: 023822, 2014.

[21] O.E. Rössler. "An equation for continuous chaos." *Phys. Lett. A* 57: 397-398, 1976.

[22] O.E. Rössler. "Different types of chaos in two simple differential equations." *Z. Naturforsch. A* 31: 1664-1670, 1976.

[23] O.E Rössler. "Continuous chaos". *Synergetics: A Workshop*, edited by H. Haken, New York: Springer, 84-199, 1977.

[24] O.E Rössler. "Continuous chaos - four prototype equations." *Ann . NY Acad. Sci.* 316: 376-392, 1979.

[25] P. Gaspard and G. Nicolis. "What Can We Learn From Homoclinic Orbits in Chaotic Dynamics?" *J. Stat. Phys.* 31: 499-518, 1983.

[26] P. Gaspard. "Rössler systems". *Encyclopedia of Nonlinear Science*, 808-811, 2005.

[27] Umesh Prajapati (2016). "The Rossler Attractor, Chaotic simulation" https://es.mathworks.com/matlabcentral/fileexchange/56600-the-rossler-attractor-chaotic-simulation, MATLAB Central File Exchange. Retrieved Sep 3, 2018.

[28] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. "Geometry from a time series." *Physical Review Letters* 45.9: 712, 1980.

[29] F. Takens. "Detecting strange attractors in fluid turbulence". In *D.A. Rand and L.S. Young, Eds., Lecture Notes in Mathematics*: 366-381, 1981.

[30] E. Bradley and K. Holger. "Nonlinear time-series analysis revisited. *Chaos* 25.9: 097610, 2015.

[31] N. Gershenfeld. "An experimentalist's introduction to the observation of dynamical systems." *Directions In Chaos*, vol. 2: 310-353, 1988.

[32] J.P. Eckmann, S.O. Kamphorst and D. Ruelle. "Recurrence plots of dynamical systems". *Europhysics Letters*, 4(9): 973, 1987.

[33] J. P. Zbilut. "Embeddings and delays as derived from quantification." *Physics Letters A* 171: 199-203, 1992.

[34] C.L. Webber Jr., C.L and J.P. Zbilut. "Dynamical assessment of physiological systems and states using recurrence plot strategies." *Journal of Applied Physiology* 76.2 : 965-973, 1994.

[35] N. Marwan, N. Wessel, U. Meyerfeldt, A. Schirdewan and J. Kurths. "Recurrence-plot-based measures of complexity and their application to heart-rate-variability data". *Physical Review E* 66(2), 026702, 2002.

[36] M.C. Soriano, et al. "Delay-based reservoir computing: noise effects in a combined analog and digital implementation." *IEEE transactions on neural networks and learning systems* 26.2: 388-393, 2015.

[37] M.C. Mackey and L. Glass. "Oscillation and chaos in physiological control systems." *Science* 197.4300: 287-289, 1977.

[38] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan and E. Ott. "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data". *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[39] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets", *Neural Comput.*,18(7):1527–1554, 2006.

[40] C. Szegedy et al. "Intriguing properties of neural networks." in *Proc. Int. Conf. Lear. Represent.* [Online]. Available: https://arxiv.org/abs/1312.6199, 2014.

[41] S. W. Akhtar, S. Rehman, et al. "Improving the robustness of neural networks using K-support norm based adversarial training." *IEEE Access 4*: 9501-9511, 2016.

[42] V. Minh and G. Hinton. "Learning to label aerial images from noisy data". In *Int. Conf. on Machine Learning (ICML)*, 2012.

[43] P. Antonik, et al. "Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronisation and cryptography." *arXiv preprint arXiv*:1802.02844, 2018.