VRIJE UNIVERSITEIT BRUSSEL

BACHELOR PAPER OF SCIENCE

Computational Capacity of Delay-based Reservoir Computing

Author:
Laurens EIROA SATTLER

Supervisor:
Dr. Guy VAN DER SANDE

A thesis submitted in fulfillment of the requirements for the degree of Bachelor in Physics and Astrophysics

August 12, 2019

VRIJE UNIVERSITEIT BRUSSEL

Abstract

Faculty of Science and Bioengineering Science
Physics and Astrophysiscs

Bachelor in Physics and Astrophysics

Computational Capacity of Delay-based Reservoir Computing

by Laurens EIROA SATTLER

Reservoir Computing is a novel, deep learning training paradigm suited for recurrent neuronal networks. It simplifies the most time and energy consuming part of the learning strategy, making even stronger the most powerful algorithms created for time-dependent data analysis. Natural and artificial highly non-linear dynamical systems can be used to perform the computations as long as they exhibit the *fading memory* condition, and if the system is sufficiently rich to transform the input data into a higher state-space for easier classification. In this work we investigate how a recently introduced dynamical system, namely the Delay-based Node, processes information. The study is made in terms of the Computational Capacity of the system. We find a direct relation between the Computational Capacity and the amount of independent *stimuli* that the system can identify and classify simultaneously. Our results confirm that such dynamical system is rich enough to process many simultaneous stimuli even in the presence of noise, in accordance with other research papers in the field.

Acknowledgements

I would like to express my sincere gratitude to everyone that in some way or another has been involved in this work. In the first place to my supervisor, Dr. Guy Van der Sande, for introducing and letting me contribute to a leading edge research and technology that has a major impact on peoples' daily lives. I highly appreciate his support, encouragement and motivation all the way till this work was fully completed, as well as his patience and time invested in supervising and reviewing my work. I also wish to express my deepest acknowledgement to Krishan Kumar Harkhoe for all the time he has dedicated to helping me to understand numerous research papers essential for getting this work done, for his advise about optimizing and writing the code, for sharing all his knowledge and ideas that have enriched my understanding of the subject, and for his motivating and supporting appreciations about my progress. Finally, I wish to thank the department of Applied Physics for letting me collaborate with them on their research as well as for their appreciations about my work.

This work brought me back to the years where I thought that studying Physics was the closest that I could get to magic. Through our formation in the University we do not only learn what are the governing rules of nature, but we also learn that these rules "can be twisted" for understanding phenomena that before could not be explained as nothing else than magic. This has boosted the welfare of society. Probably the most mind-blowing thing that I faced during this work was discovering that not only we can create instruments using nature, but that nature it self can be an incredible powerful instrument if the right mathematical and physical knowledge is used. Therefore, I want to thank all Professors and assistants of the Faculty and University in general that are sharing their knowledge and insights with newer generations. Thanks to all.

Last but not least, I wish to thank my closest familiar and personal circle for all their support and comprehension throughout these years. Without them I would not have been the person that I am today. Thanks.

Laurens Eiroa Sattler.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	**A**rtificial **N**euroal **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
TDNN	**T**ime **D**elay **N**eural **N**etwork
RC	**R**eservoir **C**omputing
DDE	**D**elayed **D**ifferential **E**quation
ODE	**O**rdinary **D**ifferential **E**quation
MZM	**M**ach-**Z**ehnder **M**odulator
CC	**C**omputational **C**apacity
NARMA	**N**on-linear **A**uto-**R**egressive **M**oving **A**verage
ESN	**E**cho **S**tate **N**etwork
NRMSE	**N**ormalized **R**oot **M**ean **S**quare **E**rror
MSE	**M**ean **S**quare **E**rror
DLR	**D**elay **L**ine **R**eservoir
DLRB	**D**elay **L**ine **R**eservoir with Feed-**B**ack connections
SCR	**S**imple **C**ycle **R**eservoir
SNR	**S**ignal to **N**oise **R**atio

List of Symbols

τ	Delay line feedback loop
W_{in}	Input connection weight matrix
$X(t)$	Reservoir state matrix at time t
W_{res}	Reservoir connection weight matrix
W_{out}	Output connection weight matrix
$u_{in}(t)$	Input signal at time t
Y	Desired value
$\hat{Y}$	Predicted value
σ	Standart deviation
$X^{\dagger}$	Moore-Penrose pseudoinverse
λ	Tikhonov parameter
θ	Node separation
M	Mask matrix
$J(t)$	Input stream
η	Feedback strength
γ	Input scaling factor
p	Non-linear exponent
$\mathcal{P}_i$	I-th degree normalized Legendre polynomial

Chapter 1

Introduction

1.1 The Brain versus the Von Neumann Architecture

Computers play a very important role in science, as well as in many other areas of society. For many human institutions and individuals, they are considered as an extension of the human brain. They are the strongest mathematical tools that humanity has ever known. They can carry out very fast computations, and compile hundreds of complex mathematical calculations, even before the user can stand up to prepare a warm cup of tea. Computers exceed by many orders of magnitude the mathematical processing speed capability of the human brain. These powerful mathematical tools are based on the so-called Von Neumann architecture, where an analog input signal is transformed into a stream of bits. Those bits are then analysed by some predefined logical statements to get the desired output stream of bits, which is again transformed by the same methods to an output analog signal.

However, these kinds of computer architectures fail when they try to accomplish tasks for what the human brain masters, such as recognising your favourite song when only the first seconds are played, smelling the fragrance of the sea when not even an eye has been open, preventing the collision of your bike against the nearest tree. Pattern recognition is not the only failure of the Von Neumann architecture in contrast with the brain. A substantial difference between animals, especially humans, and machines is the capability of learning. A distinct character of our technological advanced society is the ability of learning from what others have previously done. In fact, the brain processes information in a way absolutely different from the Von Neumann architecture. The brain consists of a vast collection of cells, called neurons, forming a neural network. Electro-chemical potential differences travelling through this network are able to process the information that is fed through stimuli from the outside world.

1.2 Artificial Neural Networks

The first mathematical model reproducing the interconnection structure of a neural network was published by IBM in 1959. This model was developed by interconnecting several of the so-called perceptrons, and assigning some specific values or weights to those interconnections. The perceptron (Figure 1.1) is a mathematical model developed to let machines make decisions. It works like a step function where the only possible outputs are two values, 1 or 0, yes or no. Each value can be obtained when the combination of some weighted input data lies inside or outside some threshold or bias b:

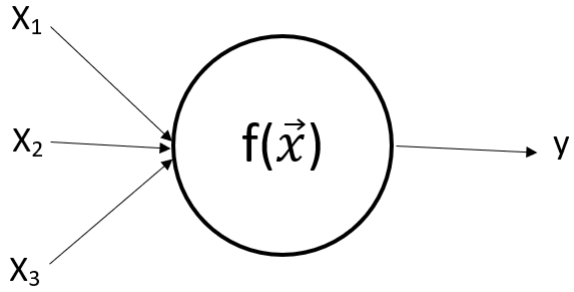


FIGURE 1.1: Overview of the perceptron mathematical model. Some weighted input data x_i feed the neuron and an output value y is obtained in terms of equation (1.1).

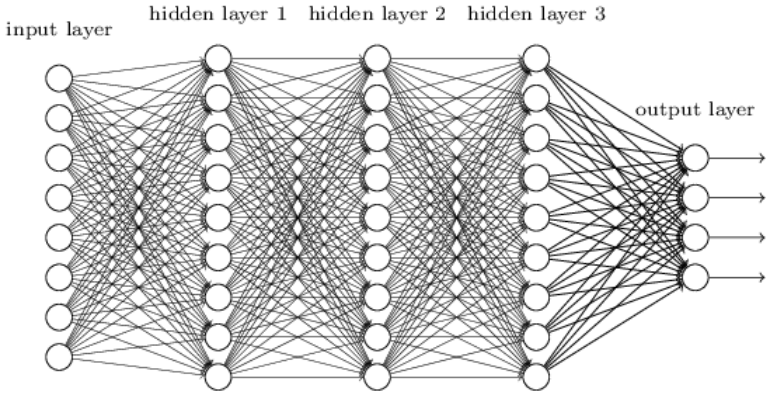


FIGURE 1.2: Overview of a multi-layer neural network [1].

$$f(\vec{x}) = \begin{cases} 1 & \text{if} \sum_i w_i x_i \leq b \\ 0 & \text{if} \sum_i w_i x_i > b \end{cases} \tag{1.1}$$

with x_i the value of the input data, and w_i the weight of the connection. The learning procedure consists in adjusting the weight values.

A binary output cannot differentiate small input variations. Hence, the operating function is replaced by a non-linear function of a weighted linear combination of the input signals and a bias. There are many different options to choose this function, e.g. a sigmoid function or a hyperbolic tangent. This function is referred to as the activation function.

The network is built by many neurons stacked in different layers, and the output values from each of the layers are sent to the next layer of neurons as input data. This enables to create more sophisticated and complex decisions than in the case of the perceptron, or to be able to make very detailed classifications. Figure 1.2 is a schematic representation of a neural network, where the input data are fed through the input-layer and propagate through the different neural layers till the output layer is reached.

The essence of the construction of a good performing neural networks resides in optimizing the connecting weights among the different neurons. Small variations in

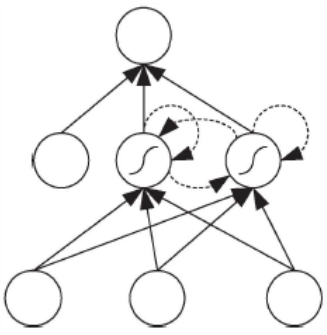


FIGURE 1.3: Schematic representation of a Recurrent Neural Network [4].

these weights can lead to very different outputs. Such optimization is the goal of the training session, and is made by using the Back-Propagation algorithm. It consists on determining the weights by minimizing the errors, which is made by taking the gradient of the errors [2]. The word *Back* in the name refers to the fact that the errors propagate from the output back to the inputs.

1.2.1 Types of Neural Networks

Many different kinds of artificial neural networks (ANNs) exist. We briefly outline the three ANNs relevant for this work:

- The previously described artificial neural network (Figure 1.2) is known as Feed Forward Neural Network (FFNN) [3], as the information only propagates in one direction, from the input layer along the system until the output layer is reached, and the output is computed. The layers between the input and output layer are called hidden layers.
- Recurrent Neural Network, or RNN. In addition to the connections in the FFNNs, connections to neurons in previous layers or to themselves are also allowed by RNNs. Figure 1.3 is a schematic view of a RNN, where the circles denote the neurons of the network, and the arrows the direction of the neural interconnections. The system takes into account previous states of the system acting as it would have some kind of short-term memory, remembering what had happened in the past [4]. They have an excellent performance where FFNNs fail, e.g. solving complex temporal tasks.
- Time Delay Neural Network, or TDNN. This kind of networks allows us to recognize temporal sequences by converting them into spatial ones [4]. The inputs are delayed in time in order to synchronize all with the final input, and all are used as input to simultaneously feed the system.

1.3 Reservoir Computing

RNNs in comparison with FFNNs show an incredible performance for solving complex temporary tasks. As already said, the key of the problem is training the interconnection weights to obtain a well performing network. For RNNs this is usually made by gradient descend on the error function. Nevertheless, due to the

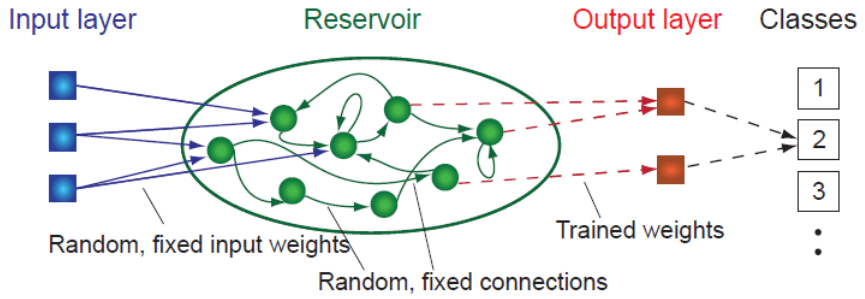


FIGURE 1.4: Schematic representation of the RC architecture [5].

highly non-linear nature of the problems, these gradient descend algorithms have faced many problems - e.g., the parameters get stuck in local minima and cannot go further to their absolute optimal values; they are slowly converging [6].

At the beginning of this century, two independent publications converged in their underlying similarities that solved the optimization procedure. One was a technical report [6], trying to make easier the training procedure of RNNs; and the other was developed by a research laboratory for robotics and neuroscience, with the goal of describing the cognitive behaviour of the brain [7]. Both contributions where unified into the Reservoir Computing (RC) paradigm.

RC establishes that it suffices to add an external layer to the recurrent network, now called reservoir, and to only train the weights of this readout layer. Thus, the dynamical connectivity of the reservoir is not adjusted, and hence the way of how information is processed by the reservoir remains unaltered. Figure 1.4 shows a schematic representation of the RC architecture. The information is fed through the input layer, and it passes through the reservoir, where randomly interconnected neurons process the information. Recurrent connections are also allowed. The state of each neuron is then sent to the output layer for later classification. This novel perspective of training RNNs was later confirmed by another publication, called the BackPropagation DeCorrelation rule [8]. That study shows that when training a RNN only the output connections are changing substantially, while the internal weights are not.

The two papers that gave birth to RC share a set of properties, which are sufficient and necessary conditions [7] to satisfy the required computation needs:

- Separation property: to recognise patterns or to classify multi-variable problems, the system needs to separate the inputs. This is a difficult task if the dimensionality of the state-space is low. However, if the system is sufficiently high-dimensional and the problem can be mapped in this higher dimensional state-space, the input values can be easily separated . This property depends mostly on the complexity of the system. An example of this is shown in Figure 1.5, where it can be observed that data that are not separable by one line in two dimensions (a), can be separable by one plane in a higher dimension space (b).
- Approximation property: given a non-linear dynamical regime, a good capability is needed to transform and map the internal states of the system into readout signals. Those signals can be used to approximate the desired computation with arbitrary precision, while making it robust against noise.

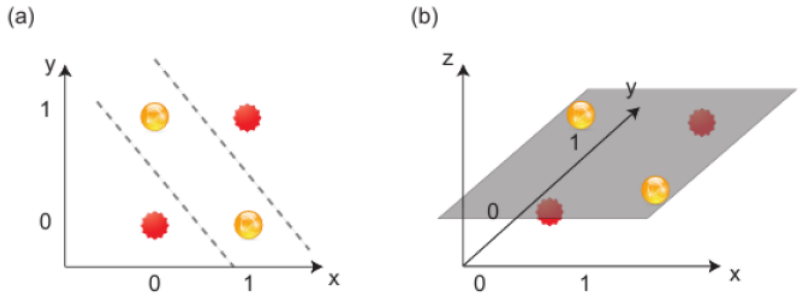


FIGURE 1.5: Representation of the separation property [5].

TABLE 1.1: Different examples of applications for RC [16].

Category	Examples
Biomedical	EEG, fMRI, ECG, EMG, heart rates, biomarkers, BMI, eye movement, mammogram, lung images.
Visual	Images, videos.
Audio	Speech, sounds, music, bird calls.
Machinery	Vehicles, robots, sensors, motors, compressors, controllers, actuators.
Engineering	Power plants, power lines, renewable energy, engines, fuel cells, batteries, gas, flows, diesel oil, coal mines, hydraulic excavators, steam generators, roller mills, footbridges, air conditioners.
Communication	Radio waves, telephone calls, Internet traffic.
Environmental	Wind power and speed, ozone concentration, PM2.5, wastewater, rainfall, seismicity.
Security	Cryptography.
Financial	Stock price, stock index, exchange rate.
Social	Language, grammar, syntax, smart phone.

- Fading Memory: also known as short-term memory in an informal way. This means that the RC is only influenced by the most recent input values. The reader is referred to [9] and [10] for a mathematical, formal definition.

Dynamical systems can also be used as reservoirs, where the nodes are intrinsically embedded in those systems and are, therefore, considered as virtual nodes.

It has been found that a bucket of water [11], the gene regulatory network of a bacterium [12], or the physical properties of tendons in a finger due to the response to external force [13], can be used as reservoirs. Furthermore, this idea has been taken to the extreme, and it has even been suggested that the entire universe could act as a reservoir itself ([14], [15]), which undoubtedly is highly interesting from a theoretical perspective, although with little applicability.

RC is an exponentially increasing field of study due to its enormous functionality. Table 1.1 shows some examples of the wide range of disciplines where RC can be applied. This table has directly been taken from [16].

1.3.1 Non-Linear Delayed Feedback Systems as reservoirs

It has been shown that Non-Linear Delayed Feedback Systems, also known as delay systems, can be implemented as a RC architecture, reducing drastically

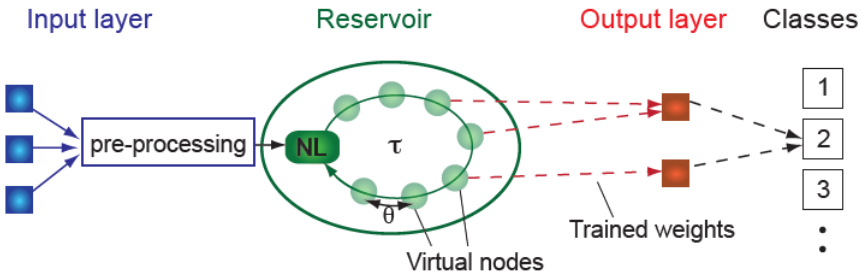


FIGURE 1.6: Schematic representation of the non-linear delayed single node reservoir architecture [5].

the structural complexity based on a single dynamical, non-linear node subject to delayed-feedback [5]. Due to the appearance in real life (e.g. systems as traffic dynamics [17], gene regulation [18], or predator-prey models [19]) delay systems have attracted considerable attention [20]. Delay connections are also found in the brain, e.g. axonal conduction between neurons [21]. Figure 1.6 is a schematic plot of these systems, where the information is fed into the system through the input layer, and before passing through the reservoir, the data are pre-processed with an input mask. After the pre-processing, the sequential data pass through the single non-linear node. These non-linear transformed sequential data are then sampled to obtain the state of the virtual neurons. These states are sent to the output layer for later classification.

Mathematically, delayed systems are described by Delayed Differential Equations (DDEs), which are very similar to the Ordinary Differential Equations (ODEs). However, the time-dependent solution is not uniquely determined by its state at a given moment. The initial conditions need to be given on a continuous interval of one delay time τ. The general form of such DDE is given by:

$$\frac{dx(t)}{dt} = F(x(t), x(t-\tau)) \tag{1.2}$$

being F a linear or non-linear function. A key feature of such system for being implemented in RC is that its state-space becomes infinite dimensional (separation property) and exhibits short-term memory (fading memory property).

Delay based RC was introduced to reduce the complexity of hardware implementations. The idea of a delay based reservoir was introduced in [5] and in [22] by using only a single non-linear node with delayed feedback. A practical setup was first developed in electronics by Appeltant et al. [5]. In optics, delayed networks of semiconductor lasers can be used due to the fact that light needs time to travel through empty space, or through an optical fiber. The exact setup in the laboratory of such reservoir is outside the scope of this work; nonetheless, the reader is referred to [5], [23], [24], and [25] for some examples.

Figure 1.7 shows an example of a real physical implementaton on which the numerical simulations performed in this study are based. Light from a continuously emitting laser source (DFB telecom laser) is sent to a Mach-Zehnder modulator (MZM), where the beam of light is split in two. The two beams travel through different paths, where one path consists of a $LiNbO_3$ crystal. This crystal induces an optical path length difference depending on the applied voltage. When the beams

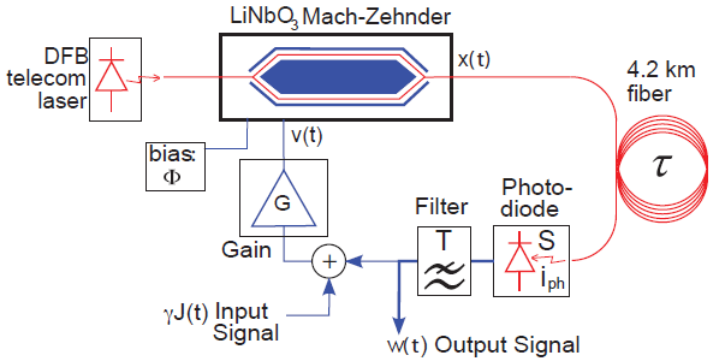


FIGURE 1.7: Schematic representation of a RC device that uses light as the dynamical medium where computation is performed. Such device is an analog implementation of a non-linear delayed single node reservoir [25].

are recollected together a voltage controlled phase difference is created. The masked input data are fed via this voltage; hence, the data are modulated on the optical beam. After that, light travels through a coiled optical fiber playing the role of the delay line. The dynamical state of the system can be read using a photo-diode. The detected electronic signal is filtered and amplified to drive the MZM, which creates the feedback loop. Equivalent devices have proved to process information at Gigabytes/s rates [26].

1.4 Goal and structure

A relevant aspect in RC is training the output connection weights; connecting extra output layers does not disturb the dynamical regime of the system. One may think that when connecting any number of output layers to the reservoir any number of simultaneous computations can take place. This is partially true as several number of independent computations can simultaneously be performed by the reservoir. However, this number is constrained to an upper limit given by the number of nodes that the reservoir has [27]. The number of simultaneous computations of the system is denoted as the computational capacity (CC) of the system.

The main purpose of this Bachelor Paper of Science is to perform a numerical simulation of the non-linear delayed single node to find its CC. Such research is of high interest due to the similarities between the delayed node reservoir and the previously described analog photonic hardware implementation. Establishing its CC can be used to get an idea of the potential of such kind of reservoirs in comparison with other reservoirs used in machine learning. Moreover, it would be one extra confirmation that the device proposed by Appeltant in [5] is a realistic reservoir that can be used in the RC paradigm. Thus, the structure and content of this work is described in detail below.

Before studying the CC of the delayed node reservoir, in Chapter 2 a system identification task is performed. This system identification task has been carried out for two different RC reservoirs, the Echo State Network (section 2.1) and the delay line reservoir (section 2.2), both built from scratch. The Echo State Network has been

introduced to setup the training protocol, its configuration is given in section 2.1.1. Since the dynamical behaviour of the Echo State reservoir is not well understood (see section 2.1.2), three other reservoirs have been used. Namely, the Delay Line Reservoir, the Delay Line Reservoir with feedback connections, and the Simple Cycle Reservoir. In section 2.2 we also search for the regime in which the Delay-based reservoir operates in stable conditions, constructing its Orbit diagram. In section 2.2.1 we outline the Delay-based configuration, together with a parameter optimization strategy for finding the appropriate parameters of the reservoir. In section 2.3 the training procedure is explained for both reservoirs, together with the overfitting strategy to ensure that the system is able to be generalized for unseen data. In section 2.3.2 we describe the NARMA system identification benchmark task used to ensure that the reservoirs operate as expected. In section 2.4 we present the results for the different ESNs (section 2.4.1) and for the Delay-based RC (section 2.4.2).

Chapter 3 presents the work carried out concerning the CC of the Delay-based reservoir. Section 3.1 describes the procedure followed to determine its CC, and a brief discussion of what information is giving about the reservoir. In section 3.2 the research together with the configuration of this reservoir is described. Once the correct parameter range was found, the total CC of the reservoir for different parameter values was computed. Then, we discuss the trade-off between linear and non-linear memory, and how the different degrees of non-linearities complement the linearity of the system. Since real dynamical systems are affected by noise, we also study how the reservoir is affected by the noise amplitude in terms of the signal amplitude, i.e., the Signal to Noise ratio. A summary of the results for the CC of the Delay-based reservoir is given in section 3.3.

Chapter 2

Introduction to RC

To introduce the reader to the concept of RC we start with a demostration of two RC-implementations of a simple benchmark test (the NARMA benchmark task, see section 2.3.2) The two implementation are the Echo State Network (ESN) and the Delay-based RC.

2.1 Echo State Network

First, we use an ESN (Figure 1.4), which is the reservoir introduced in the technical report [6] that gave birth to RC. The RC paradigm consists of three parts: the input layer, with M input nodes that feed the system with the input data; the reservoir, with N nodes; and the output layer, where L independent read-out nodes are placed, each one computing its own task. Three different matrices are built to connect the different layers:

- Input-Reservoir connections, W_{in}. This is a (MxN)-matrix, that connects each node in the input-channel with each of the neurons in the reservoir. The values of the *i* row and *j* column represent how strong the *i* input-channel is connected with the j neuron.
- Reservoir-Reservoir connections, W_{res}. By constructing a (NxN)-matrix, each of the neuron in the system is interconnected with the remaining N-1 neurons and to itself (creating recurrence loops inside the reservoir that allow the fading memory property).
- Reservoir-Output connections, W_{out}. A (NxL)-matrix is used to connect each of the neurons to the output layer, allowing L simultaneous computations.

As discussed in section 1.3, one part of the paradigm relies on training the output weights; that means, allocating correct values of the W_{out} weight matrix. Nothing is mentioned about the specific nature of the input or reservoir weights, W_{in} and W_{res}, respectively. This suggests that the particular properties of W_{in} and W_{res} are not relevant for the training procedure, as long as they are globally scaled and remain constant over time. Therefore, taking those connection weights from random distributions the requirements of the architecture are fulfilled. Such simplification is one of the main merits of RC because it provides an incredible simplification of the training procedure, which results in benefits for a hardware implementation [28]. Multiplying W_{in} with a scalar value (the input scaling factor) we determine how strongly the system is driven by the input. By rescaling W_{res} the reservoir is set to a suitable dynamical regime, which is usually made by setting the spectral radius of the matrix (highest eigenvalue) close to one.

The state of the reservoir is determined by [28]:

$$X(k) = f(W_{in}u_{in}(k) + W_{res}X(k-1)) \tag{2.1}$$

where $X(k)$ is a matrix representing the values of each neurons of the system at step k; W_{in} and W_{res} are the input an reservoir matrices, respectively; $u_{in}(t)$ represents the input signal at step k, and f is a non-linear function that allows the high-dimensional mapping of the input data. The predicted output of the system for a given input is given by a weighted linear combination of the reservoir states.

2.1.1 ESN Configuration

The parameter regime for a good performing reservoir is task-dependent. For this specific task the system has the following characteristics:

- The input layer consists of one node. Through this input layer, the same input data used to create the NARMA sequence (see section 2.3.2) are sent into the reservoir.
- Each element of the input matrix W_{in} has been taken randomly from the set $\{-1, 0, +1\}$. The matrix has been rescaled by multiplying the input connections by a factor of 0.2 to reduce the input driving the system, as explained in section 2.1.
- A reservoir with 400 nodes has been used.
- A hyperbolic tangent has been used to describe the state of the reservoir as the non-linear function in equation (2.1).
- The reservoir interconnection matrix, W_{res}, has been taken from a normal distribution of mean value 0 and standard deviation 1. Once it was created, its highest eigenvalue (also called spectral radius) has been set to 0.95; thus, a more interesting performance is obtained [29].
- The readout layer consists of only one node. In this node, the NARMA sequence is fitted (predicted), which corresponds to the input sequence u, and is used to feed the system and to create the NARMA sequence in the training (testing) scenario.
- The output weight matrix, W_{out}, used in the test phase is the mean output weight matrix computed in the training sessions.
- From equation 2.1, which denotes the reservoir state, we note that no previous state of the system exists for the first element. Therefore, the first time step has been removed.
- One bias node has been added to the reservoir. The output value of such node is one and constant over time. This gives a better amplitude to the output signal.

2.1.2 Special ESN Reservoirs

Not all ESNs are well suited for practical applications [31]. Some properties of the reservoir are not well understood [32], e.g. the choice of random connectivity matrices does not give a good understanding of the organization of the reservoir's

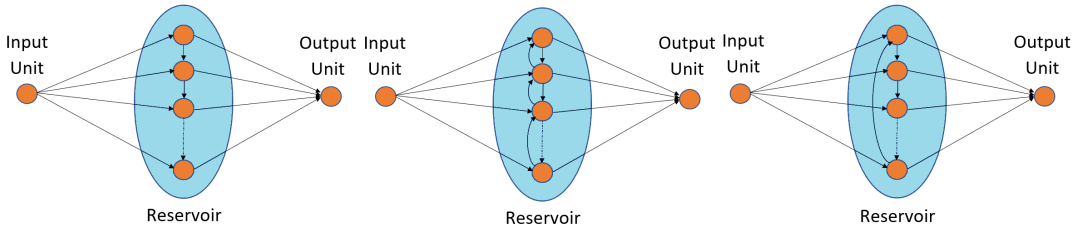


FIGURE 2.1: Schematic representation of the different topologies that have been used for comparison with the ESN [34]. Left: Delay Line Reservoir (DLR). Middle: Delay Line Reservoir with feedback connections (DLRB). Right: Simple Cycle Reservoir (SCR).

dynamics [33]. Due to this, in this work an ESN with a non-random specific connectivity (proposed in [34]) has also been simulated and compared to the traditional ESN. We will specifically consider:

- Delay Line Reservoir (DLR), left panel in Figure 2.1. In this system, the node in the input layer and in the read-out layer are connected to all the neurons inside the reservoir. Inside the reservoir, only feed-forward connections are allowed. Each neuron can only be connected to two other neurons, an incoming connection and an outgoing one, each having the same weights r. This reservoir topology in matrix form is given by a null matrix only filled with elements in the sub-diagonal:

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ r & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & r & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & r & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & r & 0 \end{bmatrix}$$

  For this specific simulation the value of r has been set to 0.92, which after trial-and-error has been found to be a good value.

- Delay Line Reservoir with feedback connections (DLRB), middle panel in Figure 2.1. The construction of this topology is similar to the DLR; however, now a back-connection weight is allowed to the previous node (all back-connection weights b are the same). This is represented by filling also the upper-diagonal of the matrix.

$$\begin{bmatrix} 0 & b & 0 & \cdots & 0 & 0 & 0 \\ r & 0 & b & \cdots & 0 & 0 & 0 \\ 0 & r & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & b & 0 \\ 0 & 0 & 0 & \cdots & r & 0 & b \\ 0 & 0 & 0 & \cdots & 0 & r & 0 \end{bmatrix}$$

  After trial and error, r has been set to 0.9 and b to 0.04.

- Simple Cycle Reservoir (SCR), also known as ring reservoir, right panel in Figure 2.1. This reservoir has a ring topology, we can imagine each neuron placed in a circumference, being each neuron connected to its nearest neighbors and

to the input and output layers. Its matrix form is given by:

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & r \\ r & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & r & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & r & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & r & 0 \end{bmatrix}$$

After trial and error, the value of r is set to 0.9 in this simulation.

2.2 Delay-based Reservoir Computing

The architecture of delay-based RC is very different from the ESN described in section 2.1, as can be observed from Figures 1.4 and 1.6. The input data pass through the non-linear node at the start of the delay line to fed the reservoir. In order to recognise the temporal sequences, the input needs to be pre-processed and time-multiplexed. The virtual nodes are represented by N equidistant points in the delay loop; the separation between the nodes is given by θ. Multiplying the node separation by the number of nodes in the reservoir we obtain the length of the delay line τ ($\tau = \theta N$).

Each of the input values $u(k)$ (with k the corresponding time step) from the input stream is sent sequentially to all the neurons inside the reservoir. To send the input values each of the values is repeated N times to create the input stream $I(t)$ (where $I(t) = u(k)$ for $\tau k \leq t < \tau(k+1)$).

As for the ESN, there exists a matrix that connects the input values with the neurons. In this case we call this matrix the mask $m(t)$, and is a $(N \times M)$-matrix (recall that N is the total number of neurons, and M the number of nodes in the input layer) taken from a specific set and that is repeated every delay line. Therefore, the actual value that each of the node receives is given by:

$$J(t) = \sum_{j=1}^{M} I_j(t) \cdot m_j(t) \tag{2.2}$$

with $I_j \in I$, and $m_j \in m$, and where J becomes a N dimensional vector for a time step t_0 that represents the input sequence for the interval $[t_0, t_0 + \tau]$. This is made to enhance the variability of the reservoir's response. A representation of the pre-processing and masking procedure is shown in Figure 2.2.

The node separation plays an important role in the dynamics of the reservoir. When θ is too short, the response to the input will be too small to be measured and therefore information will be lost. If θ is too long the interconnection between the virtual nodes will not be available (Figure 2.3 b). The best performance of the system is found when the values of θ are such that the system is in a transient state (the diversity of states is high) for the entire delay line [28], and the state of the virtual nodes depends on their predecessors.

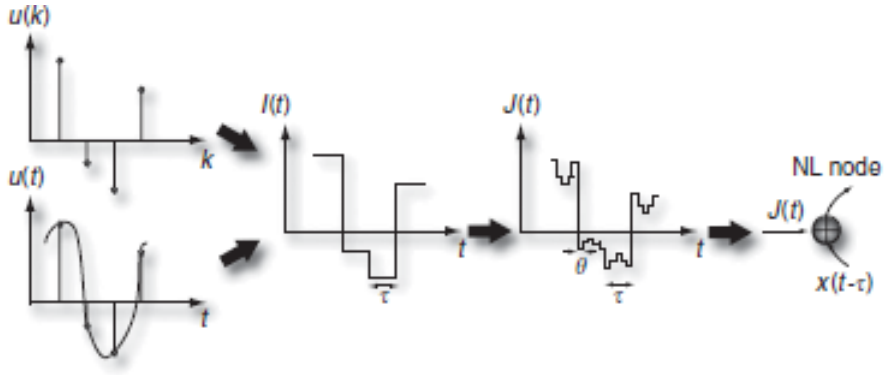
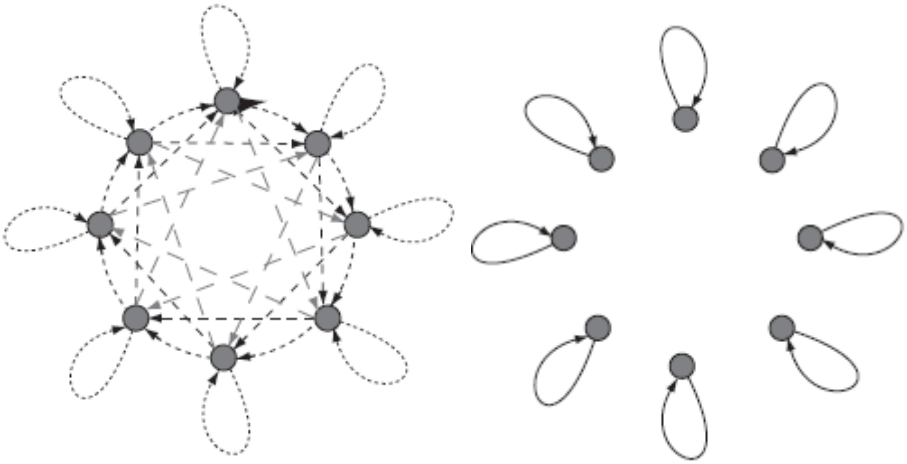


FIGURE 2.2: Schematic representation of the pre-processing and masking procedure [5]



(a) (b)

FIGURE 2.3: Schematic representation of the interconnection structure between the nodes in the delayed reservoir for small values of the node separation in (a) and large values in (b).

The behaviour of dynamical systems is modeled by equation (1.2). However, we need to take into account the influence of the input signal in this delay line reservoir. For this, we will use the Mackey-Glass equation:

$$\frac{dX(t)}{dt} = -X(t) + \frac{\eta[X(t-\tau) + \gamma J(t)]}{1 + [X(t-\tau) + \gamma J(t)]^p} \tag{2.3}$$

where $X(t)$ represents the state of the dynamical system at time t, τ the delay line feedback loop, and η represents how strong the connection is between the current neuron and previous neuron states and is called feedback strength. Analogous to the ESN, the strength of the (pre-processed) input signal $J(t)$ is tuned by the input scaling γ. The parameter p determines the non-linear degree of the evolution equation and is, therefore, called the non-linear exponent. The feedback strength allows the system to operate in a stable regime, even in the absence of any input. This DDE function comes originally from simulating model blood cell regulation [35]. It has been extensively used since the origins of RC ([36], [6], [37]), and characterizes the dynamical evolution of the system. The state of the i-th individual neurons at time

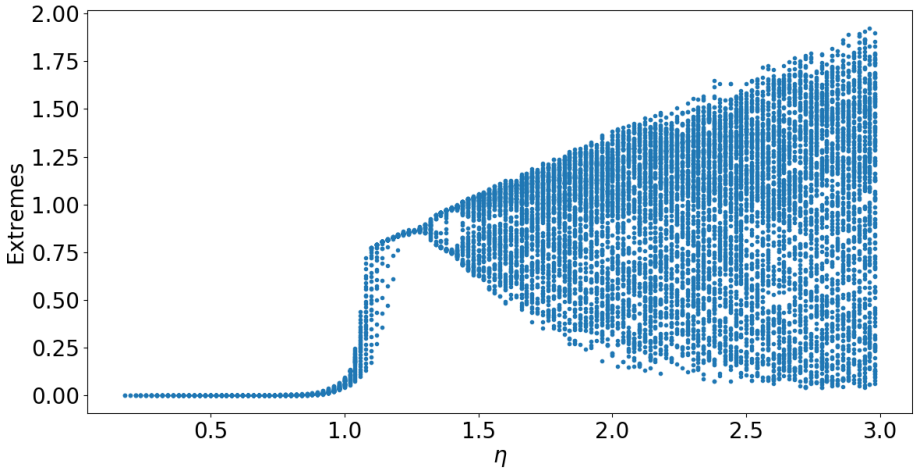


FIGURE 2.4: Orbit diagram of the Mackey-Glass DDE (equation (2.3)) without injection. The values used are $N = 400$, $p = 1$.

step k will be given by:

$$X_i(k) = X(k\tau - \frac{\tau}{N}(N - i)) \tag{2.4}$$

Equation (2.3) cannot be solved analytically. To obtain the corresponding value for the state of the system at time t, namely $X(t)$, numerical methods have to be applied. In this work we use the Euler method [38].

Orbit diagram

The dynamical regimes of the evolution equation (2.3) are identified by means of an Orbit diagram. This is shown in Figure 2.4, where we observe the behaviour of the maxima and minima (blue dots) of the temporal evolution of the internal reservoir nodes for different feedback strengths η for a system without injection ($\gamma J(t) = 0$). When injection is introduced the system behaves in a similar way. In the range of η between 0 and 1 the figure shows that the values for the extremas are very close to zero and the maximas equal the minimas, meaning that the systems is in a stable regime. Here is where we preferably allocate the dynamical regime, although no information can be extracted for the optimal value of η. We note that in a very small interval of η, the values at which the extrema are found rapidly increase; however, they are still very close to each other. Finally, in the right part of Figure 2.4 we see that the values of the extrema are found everywhere; they do not follow any pattern and seem to be stochastically distributed.

2.2.1 Reservoir Configuration

Some of the characteristics of the ESN configuration have been repeated for the Delay-based RC. The same number of input and output nodes have been used; the implemented number of neurons (N) is 400. Further, the same number of training and test sets, together with the same set lengths have been used, except for the parameter optimization where only one training set has been used (see below). The output weight matrix, W_{out}, has been obtained by taking the mean of the W_{out} from the training session. The following characteristics are specific for the delayed node system:

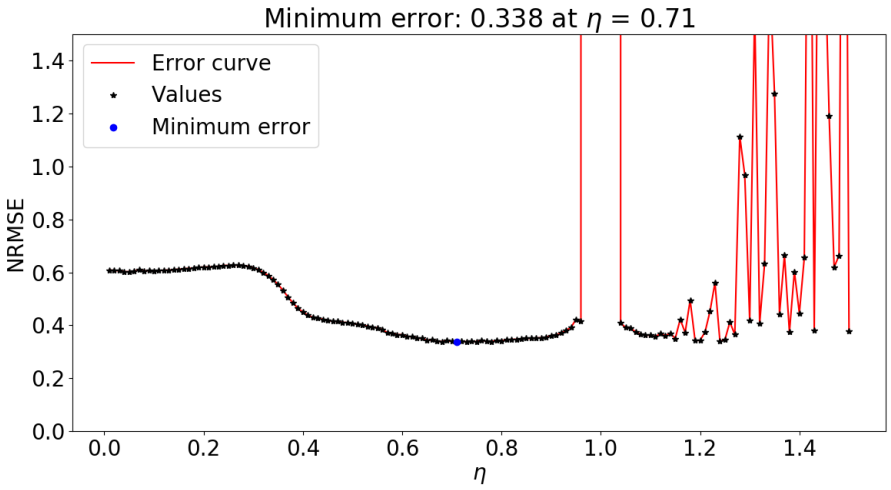


FIGURE 2.5: Feedback optimization procedure for the Delay-based reservoir with $N = 100$, $\theta = 0.2$, $\gamma = 0.1$, $p = 1$. Here, the lower the NRMSE the better the performance of the reservoir. We can see that the best parameter regiem lies around η equal to 0.7.

- The mask matrix, $m(t)$, plays a similar role as W_{in} for the ESN. Therefore, a similar matrix is built, with one dimension, and its N elements are taken randomly from the set $\{-1, 0, +1\}$.

- A numerical integration of equation (2.3) is performed in order to obtain the reservoir state matrix. The value of the node separation θ is taken as 0.2 (obtained from [5]), such that delay line length $\tau(=N\theta)$ is 80. The chosen value of the non-linear exponent p is 1; in this way, the system possesses a weak non-linearity [5]. From the optimization procedure of the feedback strength η (Figure 2.5), it can be noticed that the minimum can be found around the region of $\eta = 0.7$; thus, we adopt this value of η for the simulation. To perform the numerical determination of the neurons, the reservoir has been built as a one dimensional sequence of $N \times T$ elements (with T the length of the input sequence u). The numerical integration has been chosen to have 5 time steps between each of the neurons. Once the dynamical system was simulated, the reservoir sequence has been reshaped into a $(N \times T)$-matrix in order to be able to compute W_{out} as described in section 2.3.

Parameter optimization

From the orbit diagram (Figure 2.4) we expect the best dynamical regime for values of the feedback strength (η) smaller than 1. The same characteristics of the reservoir have been used for the simulation, but this time a system with 100 internal nodes and only one training data set have been used. In Figure 2.5 we investigate the RC performance for the NARMA task (see section 2.3.2), expressed as error (NRMSE) values versus the feedback strength η increased in steps of 0.01. In this parameter optimization procedure, the lower the NRMSE value, the better the performance (the error values have been connected to guide the eye along the error curve). We observe that for small values of η the error curve is constant with a relatively NRMSE high value (in terms of a good performing reservoir). Then, we see that the error curve starts to monotonically decrease till the global minimum is

reached at $\eta = 0.71$ (blue dot). Then, the error curve stars to increase smoothly till the value of η becomes almost 1, where the error becomes huge. This part corresponds to the vertical line in Figure 2.4; here the values of the error become so big that if they would be shown, the global minimum would not have been appreciated. On the last part of the error curve, we see how error values appear again in the plot, but the curve is not smooth anymore and seems to have a stochastic performance. This can be related to the last part of the orbit diagram where the system is no longer in a desirable, stable regime since there is no control on the performance.

2.3 Training and Testing

Training the readout layer consists on finding the coefficients of the linear combination of the reservoir state such that the result of the linear combination is the closest to the desired value. Training becomes a least square problem, and the output matrix W_{out} is given by:

$$W_{out} = \min_{W} \parallel X \times W - Y \parallel^2 \tag{2.5}$$

where X are the reservoir states given by equation (2.1), and Y are the desired values.

In practice, when the RC is implemented, a set of data D is used. This data-set of length $\mid D \mid$ is split in two subsets, one used for training (D_{train}), and a second one used to evaluate the performance of the training (D_{test}). Each of the training and testing subsets consists of multiple samples of input data with T time steps. An $(M \times T)$ input signal is fed into the system and gives a $(L \times T)$ output signal Y. The input signal is used to determine the reservoir state given by equation (2.1). Depending if the training or testing session are taking place, two different procedures will follow:

- Training session. Once the state of the system is computed, the W_{out} matrix can be obtained by the least squares fitting procedure, so that the computed output $\hat{Y} = W_{out}X$ is the closest to the desired output Y. In practice, equation (2.5) is rewritten as [29]:
$$W_{out} = (X^T X)^{-1} X^T Y \tag{2.6}$$
or in a more compact way:
$$W_{out} = X^{\dagger} Y \tag{2.7}$$
where $X^{\dagger} = (X^T X)^{-1} X^T$ is the Moore-Penrose pseudoinverse of the state matrix X [39].

- Testing session. Once the system has been trained W_{out} is obtained and remains fixed, the state for the reservoir in the testing session is computed. Then, the prediction of the unseen test sample is given by:
$$\hat{Y} = W_{out}X \tag{2.8}$$

The Normalized Root Mean Square Error (NRMSE) is introduced to quantify the accuracy of the predicted values $\hat{Y}$ with respect to the desired sequence (Y). The lower the NRMSE, the better the accuracy of the model. Its value is given by:

$$NRMSE = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} \sqrt{\frac{(Y(t_i) - \hat{Y}(t_i))^2}{\sigma_Y^2}} \tag{2.9}$$

where σ_Y^2 is the variance of the desired output Y.

In the simulations 5 independent data sets with 800 elements in the sequence have been used. Four sets (80% of all data) have been used for the training, and one set (20% of the data) for testing.

2.3.1 Overfitting

When the output weights are too much adjusted to the data set that have been used for training, overfitting occurs. This happens because the model adds irrelevant information of the training data (e.g. noise) instead of identifying the common properties and patterns of the input data. Thus, the model cannot generalize unseen data, and its performance is reduced outside the training session. Dealing with this problem is called Regularization. Many techniques are known to solve this problem; among others e.g. Gaussian regularization, that consists of inserting noise (obtained from a Gaussian distribution) to the input, or the Lasso and Tikhonov (or Ridge) regularisation, that consists of modifying the least square procedure.

Both methods are equivalent and should give similar results ([40]). In both cases an extra parameter value needs to be found for an optimal operating reservoir. In the case of Gaussian regularization, the amplitude of the added Gaussian noise minimizes the error. The Ridge regression is built adding an extra term to the least square procedure (equation 2.5), which needs to be found. After a quick comparison between both the Gaussian and Tikhonov regularizations, it has been found that Tikhonov regularizarion gives better results. Its output weight matrix is given by:

$$W_{out} = \min_W \parallel X \times W - Y \parallel^2 + \lambda \parallel W \parallel^2 \tag{2.10}$$

with $\lambda > 0$. The extra term is added to minimize the error and is called the regularization term. It can be noticed that when $\lambda = 0$ the least square method is recovered. This equation is convex for W and, therefore, it has only one local minimum. By minimizing the error, the solution of the regularization is found to be:

$$W_{out} = (X^T X + \lambda I)^{-1} X^T Y \tag{2.11}$$

where I is the identity matrix. To maximize the performance of the reservoir an appropriate value of λ needs to be found. For this task, by trial and error, the global minimum of equation (2.11) has been found for $\lambda = 2 \times 10^{-5}$. This is the value used in the simulation.

2.3.2 NARMA

To present our own numerical simulations of the ESN and delay-based RC, we consider as our system identification benchmark task the Non-linear Auto-Regressive Moving Average (NARMA) task. This sequence is long-term time dependent and highly non-linear. The benefit of such sequence is that it can be artificially be created *in-situ* without the need of any external generated input data.

NARMA was first introduced in a survey and unification paper [41], and used in many different publications in the RC context (e.g. [42] and [43]). Different constructions of this sequence can be created depending on the degree of the long-term

time dependency. In our case we have chosen a sequence of degree 10, and each term of the sequence is created in the following recursive way:

$$y_{k+1} = 0.3y_k + 0.05y_k\Big[\sum_{i=0}^{9} y_{k-i}\Big] + 1.5u_k u_{k-9} + 0.1 \tag{2.12}$$

where u_k is a random number taken from a uniform distribution inside the interval [0, 0.5]. The 10 terms inside the sum establish that this sequence is a 10-th order degree serie. For other orders the construction is similar; however, the coefficients of each term would also change.

It can be noticed that the system is in a transient regime until the 10-th interaction starts. Therefore, the first 10 elements have been removed; in this way, the transient regime does not affect the output weights.

2.4 Results

2.4.1 ESN

In section 2.1 and 2.3 the procedure to simulate the ESN proposed by Jaeger [6] as a way to reduce the complexity of the training procedure for RNN has been described. The same procedure has been made for the NARMA system identification benchmark task. As pointed out before, due to the low understanding of the dynamics in the reservoir [32] three other reservoirs proposed by [34] have been simulated, so that the different results can be compared. They are listed in Table 2.1 as NRMSE errors (equation (2.9)), which determine the accuracy of the predictions.

TABLE 2.1: Computed NRMSE errors of the 4 training sessions, together with the mean training error and the test session error for the Randomly-connected Echo State Network (RESN), Delayed Line Reservoir (DLR), Delayed Line Reservoir with Feed-Back connections (DLRB) and Simple Cycle Reservoir (SCR)

	RESN	DLR	DLRB	SCR
Training Error 1	0.071	0.106	0.112	0.108
Training Error 2	0.098	0.108	0.118	0.113
Training Error 3	0.084	0.093	0.108	0.117
Training Error 4	0.084	0.116	0.119	0.115
Mean Training Error	0.084	0.106	0.113	0.113
Test Error	0.107	0.151	0.148	0.144

Figure 2.6 shows the results of the training procedure for the ESN reservoir. The same set of input values u (equation (2.12)) has been used to feed the sytem and to create a NARMA sequence. The reservoir has been simulated according to equation (2.1). Once the NARMA sequence (blue continue line in the figure) and the reservoir state were determined, the output matrix W_{out} was computed by fitting the NARMA sequence with respect to the reservoir states by means of equation (2.11). The fit is represented with the red dashed line in the plot. This has been made for 4 different training sets; each time a new independent, identically distributed input sequence u was created, and fed into the NARMA sequence and the reservoir state. The values

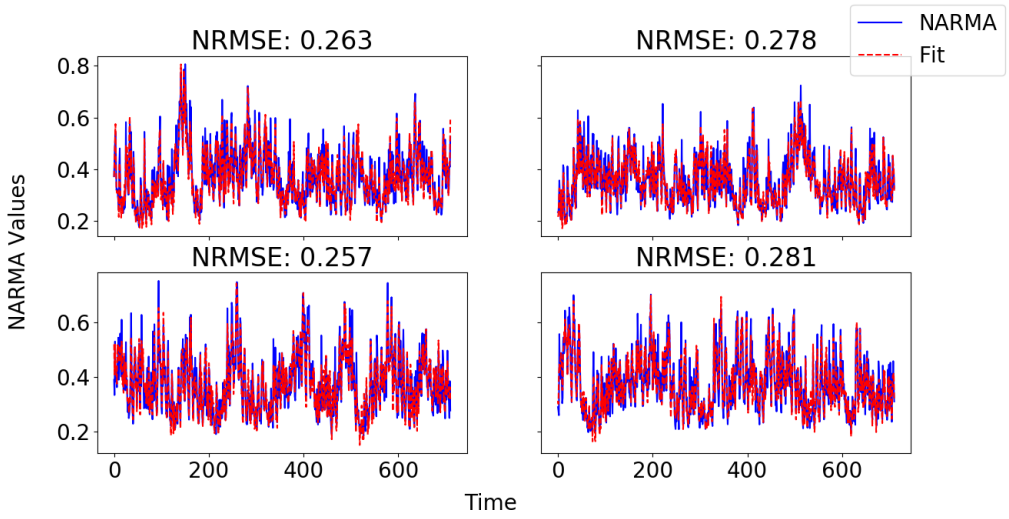


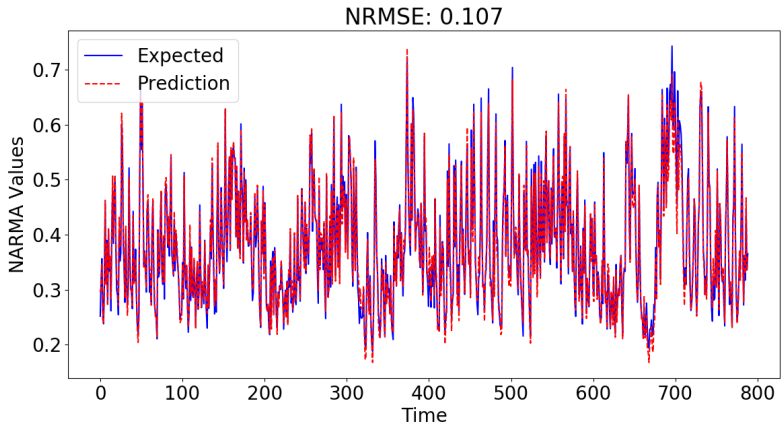
FIGURE 2.6: Computed NARMA values for the ESN reservoir vs. the corresponding fictitious time steps, represented by the blue continuous line for the four training sessions. The red dashed lines represent the obtained fitted values. These fitted values will be later used for testing the reservoir's performance on an unseen data set.

of the NRMSE errors for the ESN correspond to the first four elements of the first column in Table 2.1, together with their mean training error.

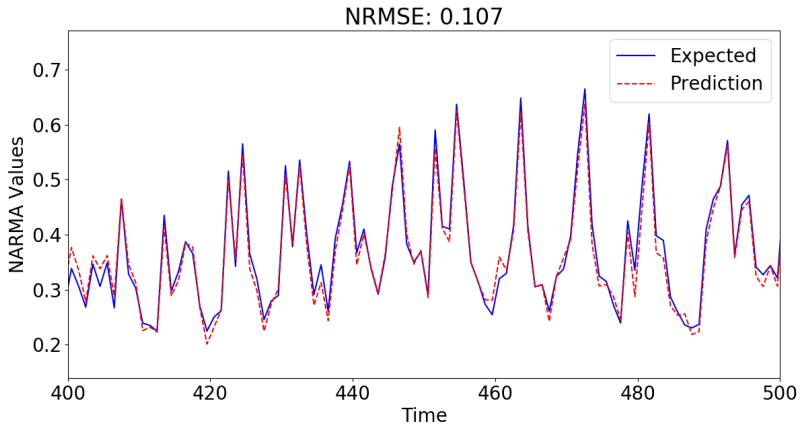
Once the output weights W_{out} were obtained for each training data-set, the mean value of the different W_{out} has been used for the test session. The results of the test session are shown in Figure 2.7 (a). In this case, a new similar input u has been created and fed into the system in order to predict how the NARMA sequence will look according to this input. This is made by multiplying W_{out} with the reservoir state matrix (equation (2.8)). The prediction is represented by the red dashed line. To compare the prediction with the real NARMA sequence, the corresponding sequence produced by introducing the test input u has been determined (blue continue line in Figure 2.7 (a)). Figure 2.7 (b) is a zoom of the [400, 500] interval to get a better view of the prediction. The value of the NRMSE error in the test case is 0.107 (Table 2.1), very close to 0.099, the best state-of-the-art performance obtained by Jaeger et al. [42].

The same procedure has been followed with the DLR, DLRB and SCR reservoirs. The figures corresponding to the training sessions are plotted in Appendix A. Figure 2.8 shows the prediction of the NARMA benchmark of the three reservoirs. The actual value of the NARMA sequence is represented by the continuum blue line and its prediction by the red dashed line. NRMSE errors are listed in Table 2.1.

We note that although the training errors are very similar - the largest difference is 0.029 (Table 2.1), they are slightly smaller for the ESN reservoir. The larger differences arise during the validation of the obtained output weight connections in the test session, being the largest difference in this case between the performances 0.044. The values for the test session of the three extra reservoirs can be considered as acceptable, even they can be considered as good, and yield some information about the nature of choosing a random interconnection reservoir weight. Such interconnection structure makes the reservoir dynamically rich and enables good non-linear



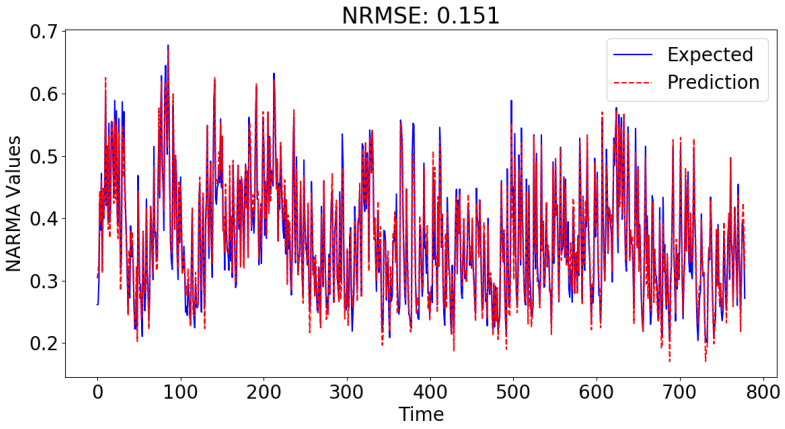
(a)



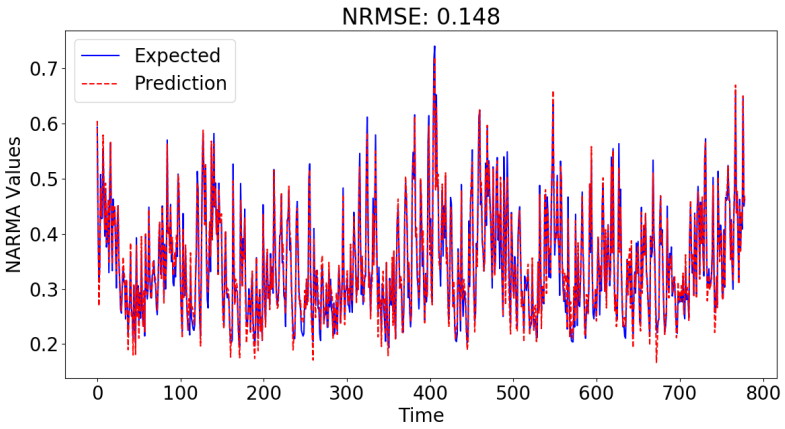
(b)

FIGURE 2.7: Performance of the ESN reservoir on the NARMA benchmark task for the obtained output weights during the training session (a). The blue continuous line represents the expected NARMA values vs. the fictitious time steps. The red dashed line represents the predicted NARMA values. In (b) we zoom the interval [400, 500] for a better insight of the reservoir's performance.

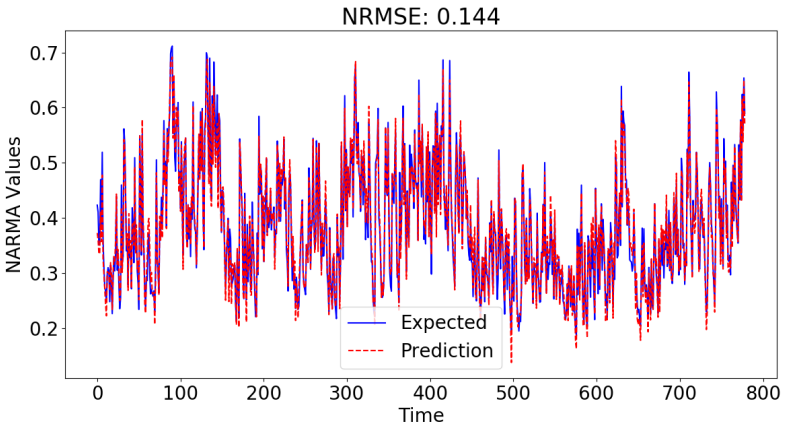
transformations of the driven input-data within the reservoir, allowing a better identification of events.



(a) DLR



(b) DLRB



(c) SCR

FIGURE 2.8: Computed values for the performance of the three alternative reservoirs with a well-known internal topology on an unseen data set. The reservoirs are: (a) Delayed Line Reservoir. (b) Delayed Line Reservoir with Feed-Back connections. (c) Simple Cycle Reservoir. The blue continuous lines represent the expected NARMA sequence for the unseen input data, and the red dashed lines represent the prediction on this unseen data using the computed output weights during the training session.

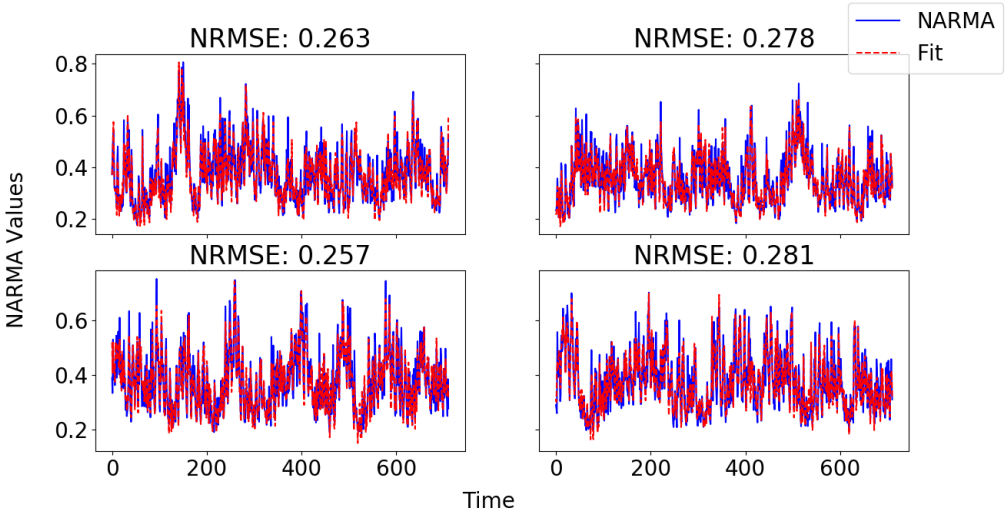


FIGURE 2.9: The four training sessions of the NARMA task for the delayed node reservoir. The blue continuous lines represent the NARMA values vs. the ficitious time step. The red dashed lines represent the fitted results. The values of the output weights from the fit will be later used to test the reservoir's performance for an unseen data set

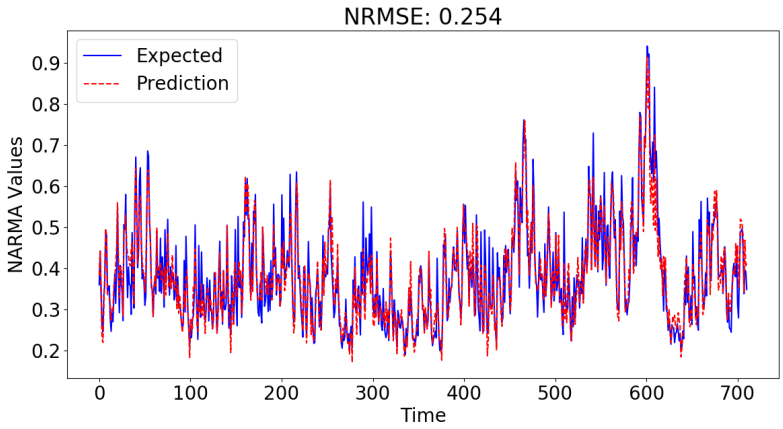
2.4.2 Delay-Based Reservoir

In this section we present the results of the NARMA system identification benchmark task by using a Delay-based reservoir. Such reservoir is analog to the hardware described in the introduction (section 1.3.1). These kinds of reservoirs are easily implemented in an optical chip (among other applications), which explains one of its interests. In this numerical simulation a Mackey-Glass oscillator (equation 2.3) has been used for the evolution of the reservoir's state. The Euler numerical method has been applied to estimate the value of each state.

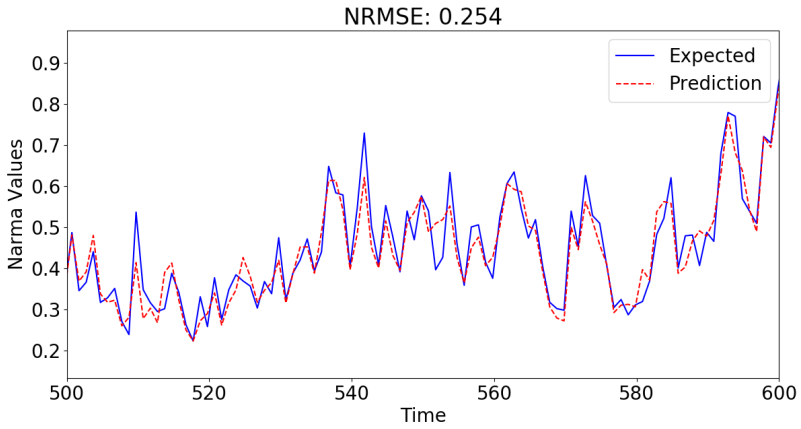
First an orbit diagram is obtained to determine the different dynamical regimes where the dynamical reservoir operates (equation (2.3)). This have been made for a reservoir without injection, Figure 2.4. Once the stable regime was determined, a search for the optimal values of the feedback strength was carried out (Figure 2.5). To perform the simulation, the reservoir characteristics are listed in section 2.2.1.

Figure 2.9 shows the training procedure for the NARMA task. The system has been fed with the same inputs u as the ones provided for the construction of the NARMA sequence (equation 2.12). This sequence corresponds to the blue continuous line of the plots, while the red dashed line represents the fit based on equation (2.8) using the computed W_{out} and the reservoir states. The simulation has been carried out for 4 independent and identically distributed input data sets. NRMSE errors are listed in Table 2.2 together with the mean error for the four training sets.

Figure 2.10 (a) shows the prediction of the NARMA task for an unseen data set. The red dashed line represents the prediction, while the computed NARMA sequence corresponding to the unseen set of data is shown by the blue continuum line. Part (b) of the figure is a zoom on the [500, 600] interval.



(a)



(b)

FIGURE 2.10: Results of the NARMA benchmark task vs. the fictitious time steps for the Delay-based reservoir using the obtained output weights during the training sessions. The blue continuous line represents the expected NARMA values for a given unseen data set. The red dashed line gives us the predicted values. We have used the following parameter values: $N = 400$, $\eta = 0.7$, $\gamma = 0.1$ and $p = 1$.

The obtained NRMSE value of the test session is 0.254, which is a reasonable performance [35]. If more parameters were tuned (e.g. mask, input strength, node separation) a closer value to the state-of-the-art performance obtained by Appeltant ([35] would be expected, but this is out of the scope of this work. From Figure 2.10 it can be noticed that the behaviour of the prediction matches the behaviour of the sequence. Obtaining a better amplitude of the output signal would match the peaks better.

TABLE 2.2: Training errors, the mean error of the training session and the error of the test session.

	Error Training 1	Error Training 2	Error Training 3	Error Training 4	Mean Error Training	Error Test
Delayed node	0.263	0.278	0.257	0.281	0.270	0.254

2.5 Summary

In this chapter, we have used the NARMA benchmark task to ensure a well-functioning Delay-based reservoir to continue our research. This has been made by first building and ESN, and then building the correspondent Delay-based reservoir. Once we have concluded that both reservoirs give comparable results, we are sure that our Delay-based reservoir is working as expected. Therefore, we are able to start characterizing some internal properties of the reservoir (see next chapter).

Chapter 3

Computational Capacity of Delay-based RC

RC relies on simplifying the training procedure of RNN by only training the output connection weights and the input and reservoir connections remain untrained. Since the connection of output layers to the reservoir does not alter its dynamics, several simultaneous computations can occur by means of adding output layers. The number of independent computations, defined as the computational capacity (CC), is constrained by the number of linearly independent state variables, being equal to this number when the reservoir satisfies the fading memory condition (section 1.3) [27].

Delayed-based reservoirs have attracted the attention for RC implementations for the reasons outlined in section 1.3.1. Those reservoirs have been tested for many different benchmark tasks with satisfactory results. However, the mechanism by which those reservoirs process information has not been characterized yet. That characterization is carried out in this work. While the optimal parameter regimes are task dependent, quantifying the way how the delayed single node processes information on a task-independent way allows us to identify the intrinsic characteristics and computational properties of the system. This enables the comparison with other type of reservoirs and, therefore, provides a better knowledge for which tasks delayed single nodes are a more advantaged reservoir in comparison with others.

In this chapter, special attention is payed to the characterization of the delayed single node in terms of its CC. Furthermore, we analyze how the non-linearity of the dynamical system is affected by the node separation (and thus the delay line length) and the input scaling factor, which are easily tuneable parameters in practical implementations. Dynamical systems are always affected by noise. Consequently, a study has been made on how the CC of the delayed node is affected by noise.

3.1 Procedure

In order to quantify the CC of the reservoir, we use the theory developed in [27]. Given a time dependent function $Y(t)$, the capacity of the dynamical system to reconstruct this function of the inputs is expressed as follows:

$$C[X,Y] = 1 - \frac{\min_W MSE_T[\hat{Y}]}{< Y^2 >_T} \tag{3.1}$$

where X represents the state of the dynamical system, given in this case by equation (2.4); $\hat{Y}$ is the computed output, T the length of the input stream, and $< Y^2 >_T =$

$\frac{1}{T}\sum_{t=1}^{T} Y(t)^2$. The MSE (acronym of *Mean Square Error*) function estimates the quality of the linear regression on Y (equation 2.8), and is defined as:

$$MSE_T[\hat{Y}] = \frac{1}{T}\sum_{t=1}^{T}(\hat{Y}(t) - Y(t))^2 \tag{3.2}$$

the min_W operator of equation (3.1) refers to the optimal output weight matrix W_{out} that minimizes the MSE function. If $C[X, Y] = 0$, the reservoir is unable to reconstruct the function Y. On the other hand, if $C[X, Y] = 1$, the output Y can be perfectly reconstructed. Therefore, we expect for the capacity:

$$0 \leq C[X, Y] \leq 1 \tag{3.3}$$

Suppose that given a time step t', a non zero capacity is obtained for the function $Y(u(t')) = u(t' - h)$, where the function associated with the input value at time t' returns the input value at time step $t' - h$. That implies that the dynamical system X stores information in a linear subset of X for at least the last h time steps. If Y is given by a non-linear function of previous inputs, such as $Y(u(t')) = u(t' - h_1) \cdot u(t' - h_2)$ with $h_1, h_2 > 0$ and $h_1 \neq h_2$, then a non zero capacity indicates that it has memory and that is able to perform a non linear transformation on the inputs. The only place where non linear transformations can occur is within the reservoir; consequently, even though the estimator is formed by a simple linear regression we are characterizing properties of the dynamical system.

If two linearly independent functions Y and Y' with non zero capacities are measured, they give independent information from the dynamical system as linearly independent functions are orthogonal. Consequently, the associated capacities, $C[X, Y]$ and $C[X, Y']$, measure independent properties of the reservoir. Theorem 4 of [27] states that for any finite set of orthogonal functions $Y_L = \{y_1, y_2, ..., y_l\}$ of size L, the total sum over this set of functions is constrained by the number of nodes N inside the reservoir (in the limit of an infinite data set). This is defined as the total CC of the reservoir (C_T):

$$C_T = \lim_{T\to\infty} \sum_{l=1}^{L} C[X, y_l] \leq N \tag{3.4}$$

In order to extract some information of the non-linear transformation that happens inside the reservoir, products of polynomials with different orders of magnitude seem to be perfect candidates to take into account the different orders of non-linearities. Because independent information needs to be obtained, a set of orthonormal polynomial functions is highly desired. The Normalized Legendre polynomials fulfil this conditions since they form an orthonormal basis inside the interval $[-1, +1]$. Thus, products of Normalized Legendre Polynomials will be used to study the memory and non-linear transformation capability of the reservoir. Then, the output y_l function used to calculate the capacities is given by:

$$y_l = \prod_i \mathcal{P}_{d_i}(u(t - i)) \tag{3.5}$$

where t represents the elements' position of the input stream that will be analyzed, $\mathcal{P}$ states for the Normalized Legendre Polynomial of degree d_i. The Legendre Polynomial of degree zero is always equal to 1 and constant regardless of the variable;

this will not give any information because no information is stored in a memory subspace, thus, this degree is discarded. Since the set of $\{d_i\}$ indices is infinite, then the product will go to zero. To deal with the vanishing capacities, an exploration strategy needs to be introduced. It goes as follows:

- First the degree d of the polynomial that will be used for computing the capacity is fixed.
- A set S with all the different possibilities of the individual polynomials is created in order to know which product of Normalized Legendre Polynomials results in a polynomial of degree d (e.g. assuming $d = 2$, then the different possibilities are a Legendre polynomial of degree 2 or the multiplication of two Legendre Polynomials of degree 1).
- The product then goes through the multiplication of each of the individual polynomials, each with a different input value.
- To see how the memory subspace behaves, the input values of the individual polynomials are delayed. This means, input values more in the past are used to go over the product of the individual polynomials.
- Capacities are computed for each combination of individual polynomials and delayed input values.

When the capacity is below some threshold $C[X, \{d_i\}] < \epsilon$, the capacity is assumed to be zero. Then, the procedure is stopped for these individual polynomial combination, and we move forward to determine the capacity for the following combination. The threshold is established in order to deal with the small probability p of obtaining a positive capacity when the capacity is in fact zero. Following [27], all capacities that are smaller than ϵ are discarded, being the value of ϵ:

$$\epsilon = \frac{2k}{T} \tag{3.6}$$

with T the length of the input stream, and k is such that $P(\chi^2(N) \geq k) = p$, where a chi-square distribution of N degrees of freedom is used. The value of N is given by the number of internal neurons in the reservoir, and the value used for p is 10^{-4} as in [27].

3.2 Linear vs. Non-Linear Memory

In this section we present our findings of the CC for the Delay-based Reservoir. First, we describe the reservoirs configuration; then, we compute the total capacity of the reservoir putting special attention to the trade-off between linear and non-linear memory; finally, we conclude by showing how the CC is affected by noisy inputs.

The number of internal nodes N is set to 50 for the simulation. Each of the mask $m(t)$ elements has been stochastically taken from the set $\{0, 0.5, 1\}$. The non-linear exponent p of the Mackey-Glass function (equation 2.3) is set to 1. Input and outputs were recorded for a length of 10^6 time steps, of which the first 100 elements were removed to ensure that any transients are avoided. The limiting capacity for this amount of inputs and outputs is $1.92 \cdot 10^{-4}$. In order to establish which are the

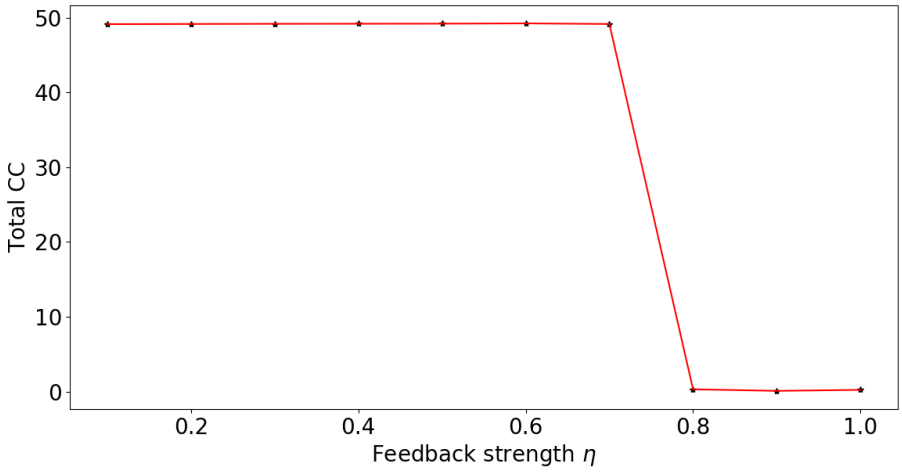


FIGURE 3.1: Measured computational capacity in function of the feedback strength. For $N = 50$, $p = 1$ and $\theta = 0.1$ and $\gamma = 0.1$.

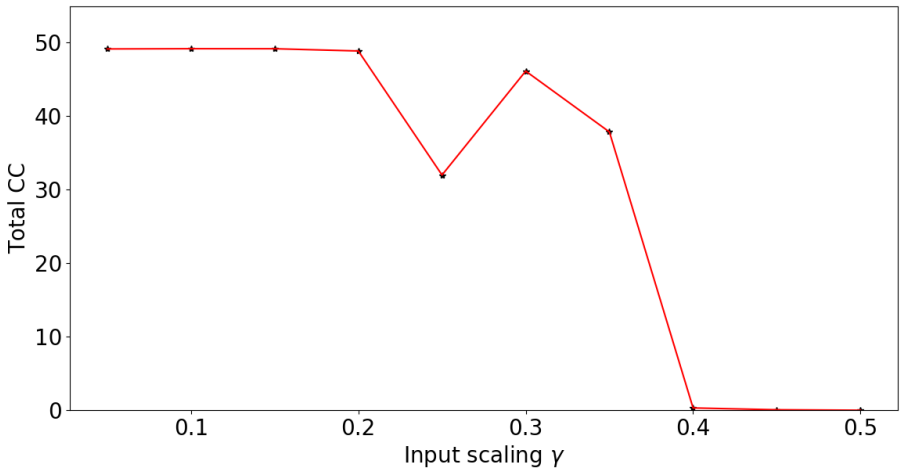


FIGURE 3.2: Measured computational capacity in function of the input scaling for a feedback strength of 0.6. For $N = 50$, $p = 1$ and $\theta = 0.1$ and $\eta = 0.6$.

optimal values for the feedback strength η and the input scaling γ, the following scans have been performed:

- First, using an input scaling value of 0.1, a search for the best feedback strength parameter was made in the stable regime of the reservoir, given by the orbit diagram (Figure 2.4), from 0 to 1 in steps of 0.1. Figure 3.1 illustrates the scanning of the feedback strengths, where the total obtained capacities up to the Legendre polynomials of degree 5 are plotted in function of the different values of the feedback strength. A red line is also plotted to guide the eye. It can be observed that the sum of capacities is found to be very close to the number of internal variables till the bifurcation point, and then drops very quickly to almost 0. The best value found for the capacities corresponds to η equal 0.6.
- Using the obtained value for the feedback strength search, now the capacities

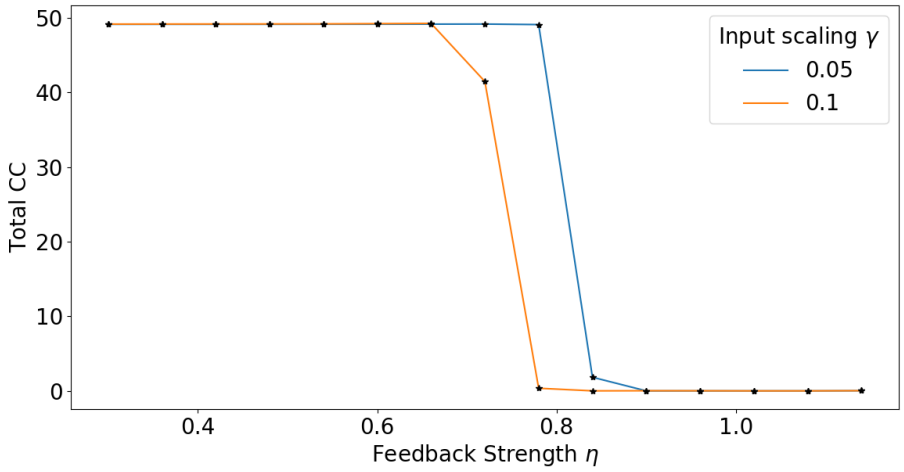


FIGURE 3.3: Total measured capacity in function of the feedback strength for two different values of the input scaling. For $N = 50$, $p = 1$ and $\theta = 0.1$.

were recorded varying values of the input scaling inside the interval [0.05, 0.5] in steps of 0.05. The resulting scan is shown in Figure 3.2. Here, the total capacities were recorded up to degree 6, but not for every value of γ the displayed capacity is the true capacity. For the input scaling values of 0.3, 0.35 and 0.4 only a lower limit could be found due to the long computational times that were needed to record higher order non-linearities, since the CC shifts to higher degrees when the input scaling increase. The best values of the CC is obtained for an input scaling equal to 0.05.

Figure 3.3 is a plot of the total CC in function of the feedback strength for two different values of the input scaling. A colored line is used to connect the obtained capacities for the different γ's. Capacities where measured up to the 6-th degree. It can be observed that in both cases the computed capacities are close to the number of nodes inside the reservoir till the bifurcation point is reached; after this, the CC drops rapidly to very small values.

By measuring the capacities one can also obtain some information of how the system processes the inputs. Figure 3.4 shows the relative CC with respect to the total capacity as a function of different values for the input scaling. The relative CC is defined as the calculated capacity divided by the number of degrees of freedom that the system posseses, i.e., the number of internal nodes that are used for the computation. The CC were measured only till the 5-th degree due the long computation time needed for determining the capacities for higher order degrees. The figure also shows the relative capacity for the different order degrees computed for the product of Normalized Legendre Polynomials. We see how the non-linearity degree of how the system processes the information increases for increasing values of the input scaling. It also shows how the CC is distributed along the different degrees of basis functions for different values of the input scaling. It can be observed that for the smaller degrees, a larger CC is found for small values of γ; for larger values of the input scaling factor, the tendency is that the CC is mainly obtained from higher order non-linearities. This makes it specially difficult to calculate its CC. First, much more combinations of the individual polynomials exist for higher order degrees. Second,

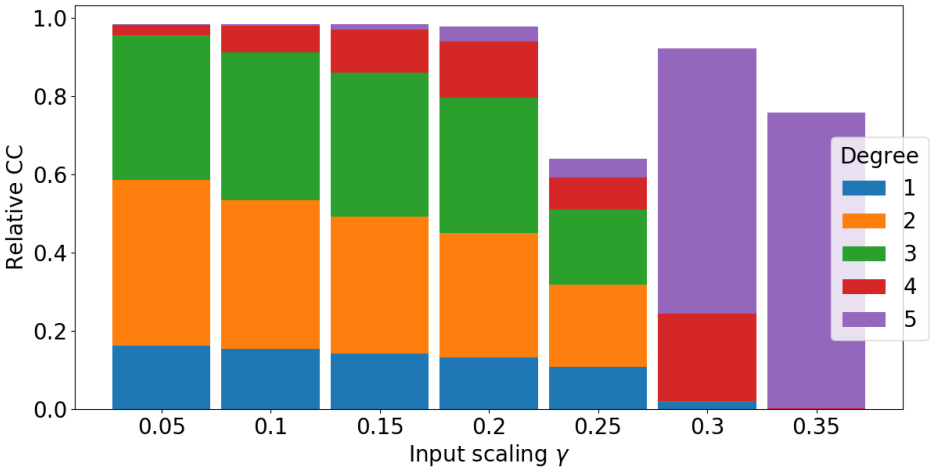


FIGURE 3.4: Relative measured computational capacity for different degrees of basis functions versus the input scaling. For $N = 50$, $p = 1$, $\theta = 0.1$ and $\eta = 0.6$.

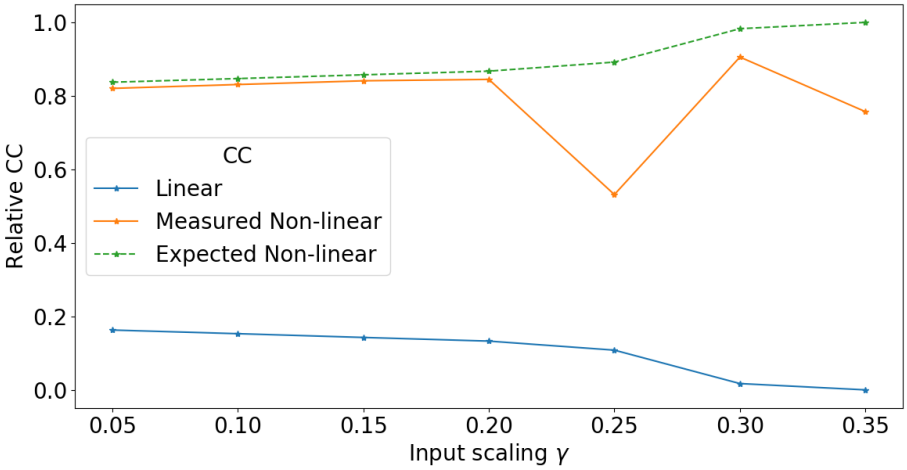


FIGURE 3.5: Trade-off between the linear and non-linear measured relative computational capacity in function of the input scaling. For $N = 50$, $p = 1$, $\theta = 0.1$. and $\eta = 0.6$.

the capacities decrease when the degree of the polynomial is higher [27], meaning that a tremendous amount of computations are needed to obtain the total CC that belongs to these higher order degrees of basis functions.

The trade-off between linear and non-linear relative CC versus the input scaling can be seen in Figure 3.5. The blue line guides the eye along the values of the relative measured linear memory capacity. The orange line guides the eye along the computed values for the non-linear memory capacity; these values are obtained by adding all the capacities for degrees higher than one. In a dynamical system with N linearly independent nodes, fully obeying the fading memory condition, the linear and non-linear information processing needs to be fully complementary [27]. The green dashed line represents the expectation of the non-linearity curve of such system (1 minus the linear relative CC for systems that totally obey the fading memory

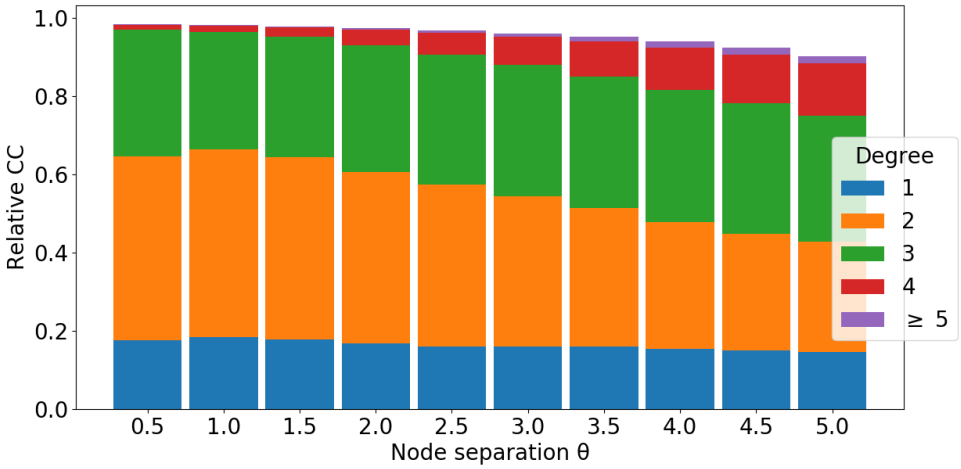


FIGURE 3.6: Relative measured computational capacity for each of the degrees of the basis functions with respect to the node separation. For $N = 50$, $\eta = 0.7$, $\gamma = 0.05$ and $p = 1$.

condition). The expectation would be that, when more orders of non-linearity are measured, the orange line converges to the green dashed line. This is because, due to the fading memory property, the CC would be shifted to higher degrees and not to input values more in the past.

To study the influence of the node separation, the system has been simulated for a feedback strength of 0.7, and an input scaling of 0.05. Figure 3.6 shows this simulation. The relative CC is measured up to the 8-th degree for different values of the node separation. It can be observed that, while the linear memory capacity does not change substantially, the information processing becomes more and more non-linear, meaning that the high order non-linearities become more important in the computation. The relative linearity and non-linearity capacities do not change considerably for different node separations.

In real dynamical system noise is always present decreasing the CC of the system. Thus, it is interesting to see the influence of noisy inputs on the reservoir performance. In order to simulate this fact, the system has been fed with an input signal u together with a noise source v (both taken from an uniform distribution in the interval $[1, 1]$). The aim is to carry out the computation only on the input signal u. Depending on the amplitude of the noise, the total CC of the dynamical system will vary. In order to do this, we introduce the notion of Signal to Noise Ratio (SNR), that is defined as follows:

$$SNR = 10 \log_{10} \left(\frac{var(u)}{var(v)} \right) \tag{3.7}$$

where *var* stands for the variance. The variation of the CC with the SNR is plotted in Figure 3.7, where three regimes can be distinguished. CC rises linearly for SNR values in the range 0 to 10 dB. The curvature in the range ∼10-20 dB arises because CC starts to saturate. From ∼ 20 dB, CC increases asymptotically to the theoretical optimal CC. We point out that we expect such behaviour of the SNR curve. Since the reservoir is fed with $u + v$ and the output layer is only trained to recuperate

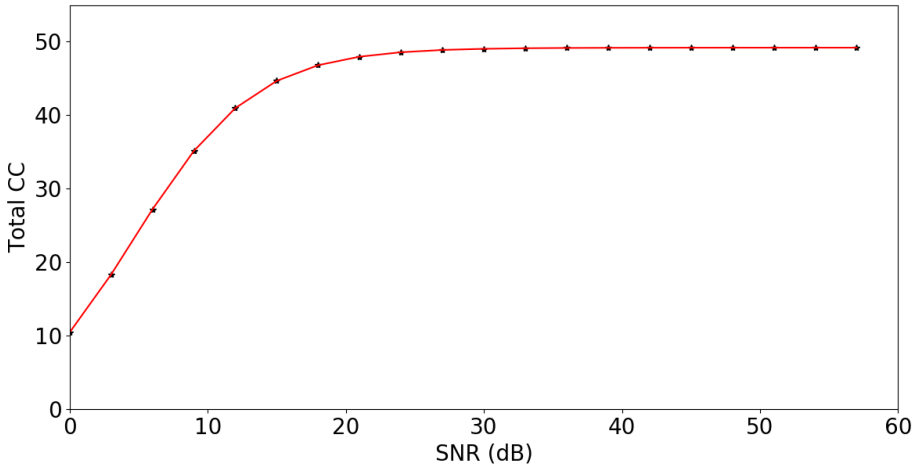


FIGURE 3.7: Signal To Noise Ratio. For N=50, η=0.7, γ=0.05, p=1 and θ=0.1.

functions of u, the CC does not achieve its maximum value. This improves when the SNR improves, meaning that the noise term v has less impact on the reservoir state values.

3.3 Summary

In this chapter a study has been carried out on the Computational Capacity of the Delay-based reservoir, which has allowed us to characterize some properties of the system. First, we have found the best parameter regime using the total capacity of the system (Figures 3.1 and 3.2). Those figures show that the best CC is found for small values of both parameters. The CC curve for different feedback strengths can be seen for different values of the input scaling in Figure 3.3. For a closer look to the linear versus non-linear memory trade-off, the CC has been plotted as a function of the input scaling (Figure 3.5). These figures provide, for the different degrees of Legendre basis functions, a good representation of the linear and non-linear processing of the inputs. The Delay-based reservoir has a non-linear behaviour. However, for higher values of the input scaling parameter, all linearities seem to disappear at the same time as the system becomes more and more non-linear (Figure 3.4). Here it can also be observed that the higher the value of the input scaling, the higher the non-linear response to the input stimuli. When the node separation varies, the small linearity of the system remains practically unchanged. However, the system shifts to a higher non-linearity (Figure 3.6). Finally, the influence of the noise has also been analyzed in terms of the signal-to-noise ratio (Figure 3.7), and it is found that the CC curve starts to saturate approaching the theoretical optimal value when the variance of the noise is a factor of 10^2 smaller than the input signal.

Chapter 4

Conclusion

Society in general, from academic institutions to industries, is discovering how powerful the combination of large amounts of data with new mathematical models that can identify patterns and make decisions can be. Deep-learning models that can process information in a really accurate and efficient way are highly demanded, and they constitute a burning field of study. In this work we analyze the Reservoir Computing training paradigm suited for Recurrent Neural Networks. Those networks are excellent dealing with temporal problems, but they are very costly to train. Reservoir Computing avoids this disadvantage by only training the output layer connection weights and leaving the system in a correct operational regime. Natural and artificial dynamical systems that fulfil the conditions described in section 1.3 can be used as reservoirs when training the deep-learning model.

In chapter 2 we have seen the performance of different models of reservoirs on the NARMA system identification task. The reservoir computing paradigm permits a very efficient and easy implementation so that state-of-the-art performances are rapidly achieved. In our case, the best NRMSE value we obtained was 0.107 when the trained reservoir was exposed to unseen data; this value compares very well with the best NRMSE value found of 0.099 in the literature. We have also introduced a delay-based reservoir. After finding its best operational regime, it was used for the same NARMA identification task, obtaining a reasonable performance.

The main goal of this work was to study how the delay-based reservoir processes information. More specifically, what is the computational capacity of this reservoir model. Once the reservoir was in a correct dynamical and parameter regime (since the optimal parameter values are task-dependent), it has been found that the number of simultaneous computations that the system can perform almost equals the number of linearly independent state variables that the system possesses, for non-noise inputs. This is expected for systems obeying the *fading memory* condition. The trade-off between the linear and non-linear behaviour has been recorded for different values of node separation and input scaling. While this trade-off is highly affected by the input scaling parameter, the linear/non-linear behaviour remains substantially unaltered for different node separations. In both cases, the higher the input scaling and the node separation, the higher the non-linear input processing behaviour of the reservoir. If the amplitude of the noise equals the amplitude of the signal, the performance of the system would be very poor. If the variance of the noise signal is around a factor of 10^2 smaller than the variance of the input signal, the computational capacity of the system starts saturating to a value very close to the number of linearly independent state variables.

4.1 Further work

In this work we have analyzed the Computational Capacity of Delay-Based reservoirs using the Mackey-Glass evolution equation. Each delay system is different and characterized by its own tunable parameters. Thus, it would be quite interesting to carry out a similar analysis in real (not simulated) Delay-based dynamical systems that could be used for Reservoir Computing; e.g. putting special attention in optical delay systems that could lead to state-of-the-art performances when analyzing and processing data at high rate speeds. The Computational Capacity provides us a tool to characterize internal properties of the reservoir and, therefore, it can be an appropriate tool to compare different types of reservoirs. It would also be interesting to see how the different types of masking procedures (e.g. masks that have not the same length as the delay-line length of the reservoir) can affect the Computational Capacity.

Appendix A

Figures

A.1 Training of the DLR, DLRB and SCR Reservoirs, section 2.4.1

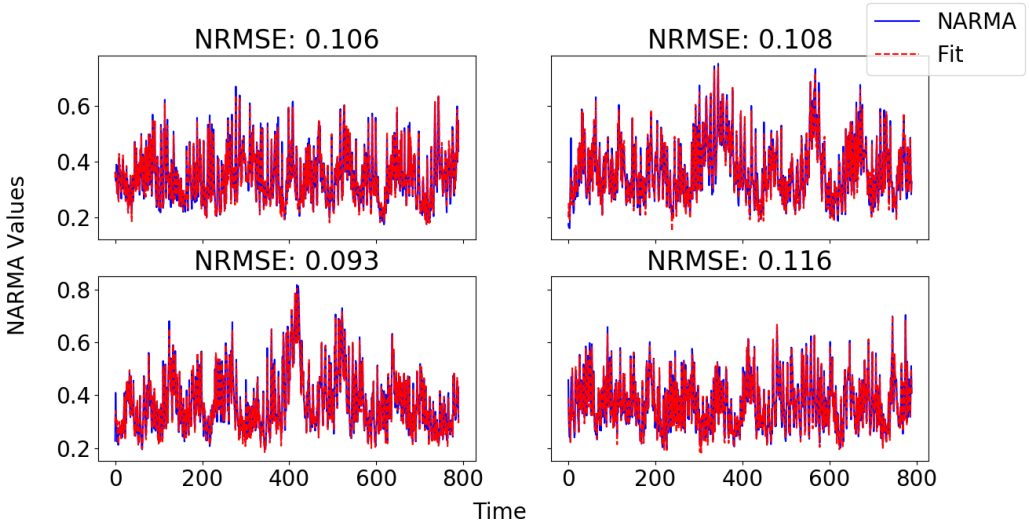


FIGURE A.1: Plot of the training sessions for the Delayed Line Reservoir. The blue continuous line indicates the values of the NARMA sequence, and the red dashed line corresponds to the fitted values.

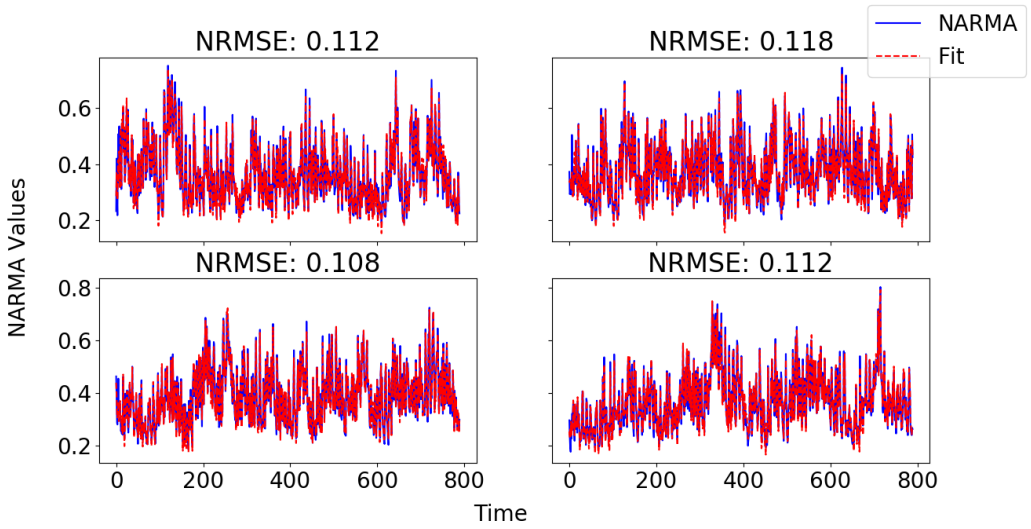


FIGURE A.2: Plot of the training sessions for the Delayed Line Reservoir with Feed-Back connections. The blue continuous line indicates the values of the NARMA sequence, and the red dashed line corresponds to the fitted values.

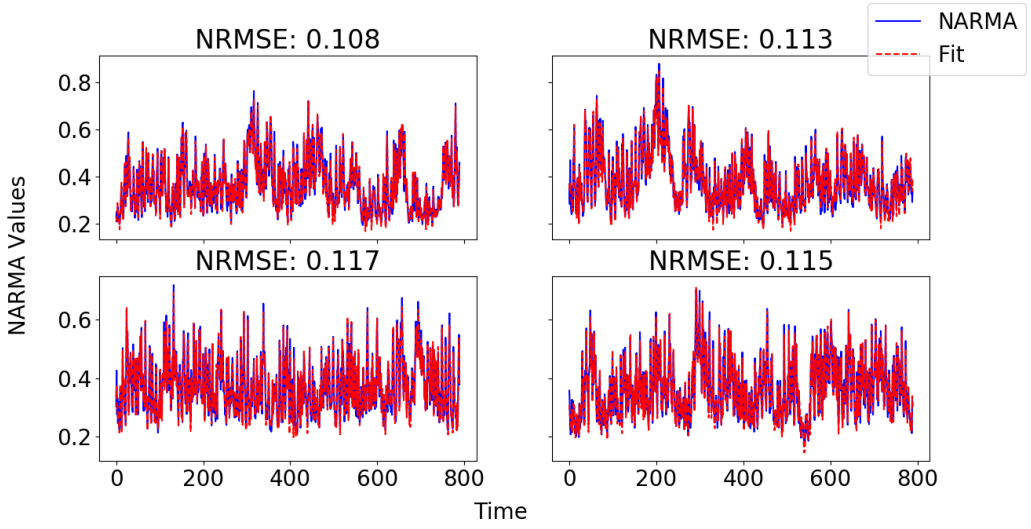


FIGURE A.3: Plot of the training sessions for the Simple Cycle Reservoir. The blue continuous line indicates the values of the NARMA sequence, and the red dashed line corresponds to the fitted values.

Bibliography

[1] V. G. Maltarollo, K. M. Honório, A. B. Ferreira da Silva. Applications of Artificial Neural Networks in Chemical Problems. In: K. Suzuky. ed. Artificial Neural Networks - Atchitectures and Aplications. Chicago: IntechOpen, 2013, pp 205.

[2] E. Alpaydm. Multillayer preceptons. In: T. Dietterich. Third edition. Introduction to Machine Learning. Massachusetts: MIT Press, 2014, pp 283-290.

[3] J. M. Zurada. Fundamental Concepts and Models of Artificial Neural Systems. In: First edition. Introduction to Artificial Neural Systems. St. Paul, New York, Los Angeles, San Francisco: West Publishing Company, 1992, pp 37-42.

[4] E. Alpaydm. Multillayer preceptons. In: T. Dietterich. Third edition. Introduction to Machine Learning. Massachusetts: MIT Press, 2014, pp 304-306.

[5] Appeltant, L. et al. Information processing using a single dynamical node as complex system. Nat. Commun. 2, 468 (2011).

[6] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology.

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

[8] J.J. Steil. Backpropagation-decorrelation: online recurrent learning with O(N) complexity. 2004 IEEE International Joint Conference on Neural Networks. 2004.

[9] Boyd, S., & Chua, L. O. (1985). Fading memory and the problem of approximating nonlinear operators with Volterra series. IEEE Trans. on Circuits and Systems, 32, 1150–1161.

[10] Maass, W., Natschläger, T., and Markram, H. (2004c). Fading memory and kernel properties of generic cortical microcircuit models. Journal of Physiology – Paris, 98(4–6):315–330.

[11] Fernando, C. and Sojakka, S. (2003). Pattern recognition in a bucket. In Proceedings of the 7th European Conference on Artificial Life, pages 588–597.

[12] Jones, B., Stekel, D., Rowe, J., and Fernando, C. (2007). Is there a liquid state machine in the bacterium escherichia coli? pages 187–191.

[13] Valero-Cuevas, F., Yi, J., Brown, D., McNamara, R., Paul, C., and Lipson, H. (2007). The tendon network of the fingers performs anatomical computation at a macroscopic scale. IEEE Transactions on Biomedical Engineering, 54(6 Part 2):1161–1166.

[14] Lloyd, S. (2002). Computational capacity of the universe. Physical Review Letters, 88(23):237901–237901.

[15] Fredkin, E. (2003). An introduction to digital philosophy. International Journal of Theoretical Physics, 42(2):189–247.

[16] G. Tanakaa, T. Yamanec, J. B. Herouxc, R. Nakanea, N. Kanazawac, S. Takedac, H. Numatac, D. Nakanoc and A. Hirosea. Recent Advances in Physical Reservoir Computing: A Review. 2018

[17] Erneux, T. Applied Delayed Differential Equations (Springer Science Business Media, 2009).

[18] Chen, L. and Aihara, K. Stability of genetic regulatory networks with time delay. IEEE Trans. Circuits Syst. I 49, 602 (2002).

[19] Martin, A. and Ruan, S. Predator-prey models with delay and prey harvesting. J. Math. Biol. 43, 247–267 (2001).

[20] Erneux, T. Applied Delayed Differential Equations (Springer Science Business Media, 2009).

[21] Haken, H. Brain dynamics: synchronization and activity patterns in pulse-coupled neural nets with delays and noise (Springer Verlag GmbH, Berlin, Germany, 2006).

[22] Paquot Y, Dambre J, Schrauwen B, Haelterman M, Massar S. Reservoir computing: a photonic neural network for information processing," in Proc. SPIE 7728, Nonlinear Optics and Applications IV 2010;7728:77280B–12.

[23] Nguimdo RM, Verschaffelt G, Danckaert J, Van der Sande G. Fast photonic information processing using semiconductor lasers with delayed optical feedback: role of phase dynamics. Opt Express 2014;22:8672–86.

[24] Paquot Y, Duport F, Smerieri A, et al. Optoelectronic reservoir computing. Sci Rep 2012;2:287.

[25] Larger L, Soriano MC, Brunner D, et al. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. Opt Express 2012;20:3241–9.

[26] Daniel Brunner et al. "Parallel photonic information processing at gigabyte per second data rates using transient states". In: Nature communications 4 (2013), p. 1364.

[27] J. Dambre, D. Verstraeten, B. Schrauwen, S. Massar. Information Processing Capacity of Dynamical Systems. Scientific Reports. 2012.

[28] G. Van der Sande, D. Brunner, M. C. Soriano. Advances in photonic reservoir computing. Nanophotonics, De Gruyter. (2016).

[29] D. Verstraeten, Reservoir Computing: Computation with Dynamical Systems. PhD, Ugent, 209. pp 104.

[30] C. Bauckhage. Bonn University. NumPy / SciPy Recipes for Data Science: Regularized Least Squares Optimization. (2015).

[31] D. Prokhorov, "Echo state networks: appeal and challenges," in In Proc. of International Joint Conference on Neural Networks (pp. 1463-1466). Montreal, Canada., (2005).

[32] Y. Xue, L. Yang, and S. Haykin, Decoupled echo state networks with lateral inhibition, Neural Networks, vol. 20, pp. 365–376, (2007).

[33] M. C. Ozturk, D. Xu, and J. Principe, Analysis and design of echo state network, Neural Computation, vol. 19(1), pp. 111–138, 2007.

[34] A. Rodan, P. Tino. Minimum Complexity Echo State Network. IEEE Transactions on Neural Networks. (2010).

[35] L. Appeltant, Reservoir Computing Based on Delay-dynamical systems. Joint PhD, Vrije Universiteit Brussel, Universitat de les Illes Balears, May 2012. pp 3.

[36] Mackey, M. & Glass, L. Oscillation and chaos in physiological control systems. Science 197, 287–289 (1977).

[37] Farmer, J. Chaotic attractors of an infinite-dimensional dynamical system. Physica D 4, 366–393 (1982).

[38] W. E. Boyce, R. C. DiPrima. Elementary Differential Equations and Boundary Value Problems. Rensselaer. Wiley, 10-th Edition. Numerical Methods, The Euler or Tangent Line Method, pp. 451-460.

[39] Penrose, R. (1955). A generalized inverse for matrices. 51(1955):406–413.

[40] Bishop, C. Training with noise is equivalent to tikhonov regularization. Neural Comput. 7, 108–116 (1995).

[41] A.F. Atiya and A.G. Parlos. New results on recurrent network training: Unifying the algorithms and accelerating convergence. IEEE Trans. Neural Networks, 11(3):697- 709,2000.

[42] Jaeger, H. Adaptive nonlinear system identification with echo state networks: in Advance in Neural Information Processing Systems (eds Becker, S., Thrun,S. ,Obermayer, K.) Vol 15, 593–600 (MIT Press, 2003).

[43] Rodan, A. & Tino , P. Minimum complexity echo state network. IEEE T. Neural Netw. 22, 131–144 (2011).