ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Interferometric stabilisation of a fibre-based optical computer

Experimental study

Mémoire présenté en vue de l'obtention du diplôme
d'Ingénieur Civil physicien à finalité spécialisée

Denis Verstraeten

Directeur
Professeur Marc Haelterman

Co-Promoteur
Professeur Serge Massar

Superviseur
Lorenz Butschek

Service
Opera

Année académique
2018 - 2019

Abstract

Acknowledgements

Contents

Chapter 1

Introduction

For the past few years, interest in optical data processing devices has been increasing. Their main advantage over silicon-based computers is that they are intrinsically faster because the information is carried around at nearly the speed of light, which could allow to overcome the limit in processing speed soon to be reached by classical integrated circuit electronics.

This Master thesis tackles the implementation of an optical computer based on reservoir computing.

Chapter 2

Reservoir Computing

2.1 Introduction

Reservoir Computing (RC) is a bio-inspired artificial Recurrent Neural Network (RNN) which is based on the Echo State Network (ESN) introduced by Herbert Jaeger in [9]. This computation scheme is well suited for real-time data processing and for chaotic time series prediction[9, 10, 15], and achieves state of the art performances in those domains, as well as in speech recognition[25, 22, 13], nonlinear channel equalisation[9] and financial forecasting [2].

2.1.1 Artificial Neural Network

A Reservoir Computer (RC) is specific kind of Neural Network (NN), which is a computation paradigm mimicking the behaviour of a biological brain. The artificial neurons are simply interconnected entities carrying an activation level. The way the activation level is updated depends on the scheme, but the basic idea is common for all of them: a neuron receives a linear combination of the activation level of the neurons to which it is connected, and then computes a nonlinear transformation of this value. This gives the new activation level. On Figure 2.1, a feedforward NN is depicted. It is called feedforward because the computation goes from left (input neurons in red) to right (output neurons in green). Feedforward NN are organised in layers, and a neuron from one layer can only be influenced by neurons in the adjacent layers. This is shown on the Figure by the arrows representing the connections. The gray neurons in the middle belong to the hidden layers, which are used to improve the computing power of such networks. The results of a computation can be read on the activation level of the output neurons [19, p.727][3, p.225].

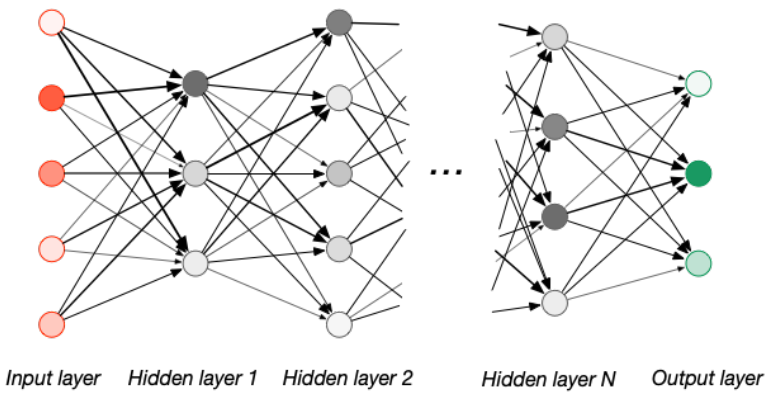

Figure 2.1: Schematic representation of a feedforward NN

2.1.2 Reservoir Computing

RC have been designed to process time dependent inputs, so their structure is inherently different from that of a feedforward NN, because they need to exhibit other properties. In this scheme, all the neurons are interconnected and form what is called a reservoir. The reservoir is fed with the time dependent input signal it should process. When the reservoir is properly set up, the activation level of each of the neurons becomes a systematic transformed version of the input signal [9]. This operating point is called the echo state and allows RC to reach their best performances [8, 10]. In this regime, RC exhibit a short-term memory of the previous inputs [9], which could explain why they perform so well in time dependent situations. There are many physical implementations of RC proposed in the literature, many of which are based on optical setups [20], that is why section 2.3 is devoted to them.

On Figure 2.2, a RC is shown. The neurons of the reservoir are represented in orange. They characterise what is called the state of the reservoir, which is encoded by $\mathbf{x}(t)$. They are coupled by the connection matrix $\mathbf{W}$. The input signal $u(t)$ is fed into the input neuron (blue) and is coupled to the reservoir *via* the input matrix $\mathbf{W}^{\text{in}}$. The output $y(t)$ is read on the output neuron (red) and is obtained thanks to the output matrix $\mathbf{W}^{\text{out}}$. This matrix is the only one that needs to be updated when the reservoir is learning. This task is not straightforward, that is why the next paragraph takes care of introducing the different approaches that can be followed to compute $\mathbf{W}^{\text{out}}$. For some applications, it can be useful to also have a feedback of the output sent back into the reservoir. This can be achieved by the feedback matrix $\mathbf{W}^{\text{fb}}$. The mathematical aspects mentioned in this paragraph are detailed in Section 2.2.

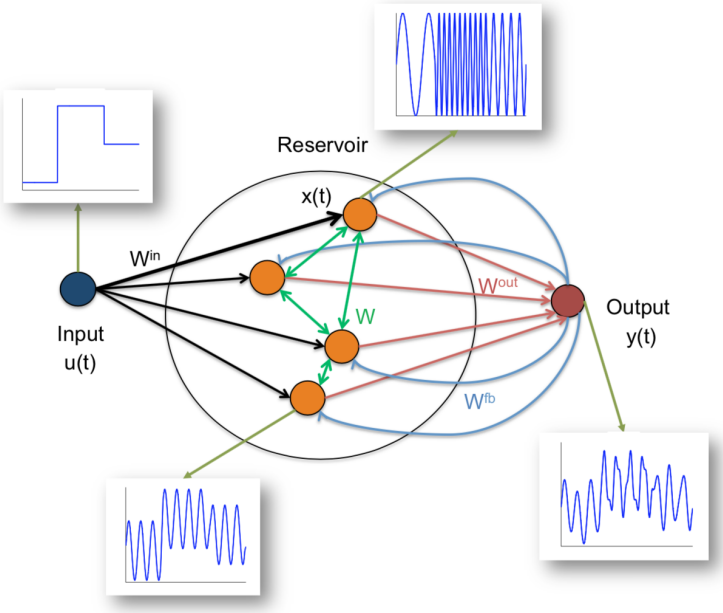

Figure 2.2: Schematic representation of a Reservoir Computer [2]

2.1.3 Machine Learning

Regardless of the learning scheme used to train a NN, the basic idea is always to minimise the difference between the desired and the actual outputs. In practice, this is achieved by updating the different connection coefficients of the NN [3, p.233][19, p.733]. This procedure often turns out to be a really complicated task for feedforward NN, which explains why the development of efficient Machine Learning (ML) algorithms is such a hot topic nowadays. In

contrast, as can be seen on Figure 2.3, RC only need their output weights to be adjusted when being trained, which makes them computationally lighter [9]. This is due to the fact that the connections of the reservoir should not contain any information about the task, but should only be used to reach the ESN regime, as mentioned in the previous paragraph. There are two main families of training methods for RC [11]. On the one hand, there is the *batch learning*, which comprises the methods requiring to first store a wealth of data regarding the task being taught before being able to actually compute the output weights. Once enough data is gathered, this kind of algorithms returns the optimal weights all at once. They present the advantage of involving only one training phase, after which the RC are ready to perform. However, the need for vast amount of data and the inability for the RC to adapt to an input evolving out of the range for which it has been trained are two drawbacks. On the other hand, *online learning* methods allow to iteratively improve the output weights. Therefore, starting from a first guess, these algorithms can converge to suitable output weights. They are much more adaptable than the batch learning ones. However, their convergence is not guaranteed and can be slow [12, 21].

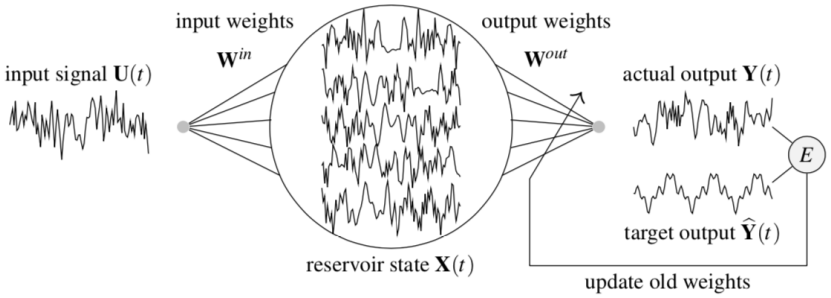

Figure 2.3: Learning procedure for Reservoir Computer [8]

2.2 Mathematical Model

In this section, an overview of the mathematical framework is given. First, the different objects are formally defined, and their dynamics is presented. Then, a few key elements about the computation of the output weights are introduced.

2.2.1 Dynamics of the reservoir

let us define the *state* of the reservoir. As said previously, the RC can be fully described by the activation level of each of its neurons. The state is therefore defined as a vector whose components are the activation levels of the neurons. If the number of neurons making up the reservoir is N, and if x_i is the activation level of the i^{th} neuron, then the state vector reads as follows:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{bmatrix} \tag{2.1}$$

The dynamics governing the state vector and the output of the reservoir proposed in [10] are presented below. In practice, it is too general for the implementations studied in this work.

However, the equations are introduced without loss of generality, and simplifying assumptions applying the photonic implementations of RC will be specified in the section devoted to them.

$$\mathbf{x}(n+1) = \mathbf{f}\left(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{fb}}\mathbf{y}(n)\right) \tag{2.2}$$

$$\mathbf{y}(n+1) = \mathbf{f}^{\text{out}}\left(\mathbf{W}^{\text{out}}\left(\mathbf{x}(n+1), \mathbf{u}(n+1), \mathbf{y}(n)\right)\right) \tag{2.3}$$

Different elements need to be defined:

- $n \in \{1, \ldots, T\}$ is the discrete time variable
- $\mathbf{u} \in \mathbb{C}^k$ is the input vector which enters the reservoir through the input neurons
- $\mathbf{W}^{\text{in}} \in \mathbb{C}^{N\times k}$ is the input matrix. It indicates how the k input neurons are connected to the neurons of the reservoir
- $\mathbf{x} \in \mathbb{C}^N$ is the state vector, as said previously
- $\mathbf{W} \in \mathbb{C}^{N\times N}$ is the synaptic matrix, or the connection matrix which has already been introduced
- $\mathbf{y} \in \mathbb{C}^m$ is the output vector of the reservoir whose value can be read out on the output neurons
- $\mathbf{W}^{\text{fb}} \in \mathbb{C}^{N\times m}$ is the feedback matrix. It couples the output back into the reservoir
- $\mathbf{f} : \mathbb{C}^N \mapsto \mathbb{C}^N$ is the nonlinear function mapping the linear combination it receives as argument to a valid state vector
- $(\mathbf{x}(n+1), \mathbf{u}(n+1), \mathbf{y}(n))$ is the concatenation of those three vectors
- $\mathbf{W}^{\text{out}} \in \mathbb{C}^{m\times(N+k+m)}$ is the output matrix of the reservoir. It is optimised through ML
- $\mathbf{f}^{\text{out}} : \mathbb{C}^m \mapsto \mathbb{C}^m$ is the output function of the reservoir

2.2.2 Computation of the output weights

To determine the output matrix $\mathbf{W}^{\text{out}}$ in the batch learning approach, one needs to perform a ridge (or Tikhonov) regression [22], which is an improved version of multivariate linear regression that improves the numerical stability of the scheme. By restricting the desired output to a scalar function $\hat{y}(n)$ and by taking a learning period of T time steps, one defines the matrices $\mathbf{X}$ and $\hat{\mathbf{Y}}$ and solves the following equation to find the output weights vector $\mathbf{W}^{\text{out}}$:

$$\mathbf{X} = \begin{bmatrix} x_0(0) & x_1(0) & \ldots & x_N(0) \\ x_0(1) & x_1(1) & \ldots & x_N(1) \\ \vdots & & & \vdots \\ x_0(T) & x_1(T) & \ldots & x_N(T) \end{bmatrix}, \quad \hat{\mathbf{Y}} = \begin{bmatrix} \hat{y}(0) \\ \hat{y}(1) \\ \vdots \\ \hat{y}(T) \end{bmatrix} \tag{2.4}$$

$$\left(\mathbf{X}^{\text{T}}\mathbf{X} + \epsilon\mathbf{I}\right)\mathbf{W}^{\text{out}} = \mathbf{X}^{\text{T}}\hat{\mathbf{Y}} \tag{2.5}$$

Here ϵ is the constant used for the Tikhonov regression. By setting ϵ to 0, one finds the *normal equation* that comes up when solving a linear regression [8] in the sense of the least squares. This procedure can be generalised to higher dimensions desired output vectors $\hat{\mathbf{y}}(n)$. Different optimisation algorithms can be used to compute the output weights in practice, but

their description is out of the scope of this work, see [14] for more details.

Different metrics can be used to capture the distance between the actual and the desired outputs. In the literature, one of the most frequent ones is the Normalised Mean Square Error (NMSE) [6] (with $\hat{\mathbf{y}}(n)$ the target vector, $\langle \dots \rangle_n$ the average with respect to n, and $\| \dots \|$ the euclidean norm):

$$\text{NMSE} = \frac{\langle \|\hat{\mathbf{y}}(n) - \mathbf{y}(n)\|^2 \rangle_n}{\langle \|\hat{\mathbf{y}}(n) - \langle \hat{\mathbf{y}}(n) \rangle_n \|^2 \rangle_n} \tag{2.6}$$

2.3 Introduction to Photonic Reservoir Computing

As already mentioned, different implementations of Photonic Reservoir Computing (PRC) have been proposed [20]. In this section, systems in which neurons are multiplexed in time are studied because they constitute a good first approach to PRC and because they bring insights that are interesting for the scheme explored in this thesis. However, it is worth mentioning that one can find among others spatially distributed RC based on fully integrated silicon-chip with nonlinearities stemming from Semiconductor Optical Amplifiers [23], and on diffractively coupled Vertical-Cavity Surface-Emitting Lasers [4].

In this section, the assumptions applying to reservoir with Time Division Multiplexing (TDM) of the neurons are first presented. Indeed, the equations introduced in the previous section can be substantially simplified when one is working with this kind of reservoir. After that, setups where neurons are encoded in the intensity of the light are considered. Finally, experiments in which the neurons are represented in the phaser of the electric field are presented. Recalling that the light intensity is proportional to the squared modulus of the phaser of the electric field, it is shown that the outputs of these two kinds of reservoir are different, but analogous, and that they rely on the same mathematical interpretation. The latter scheme is studied with greater length since it reaches state of the art performances in classical benchmark tasks and since the novel implementation which is the main topic of this work shares some features with it. As a last remark, and to give a better understanding on the flexibility of RC, in [7] the researchers managed to perform speech recognition and to resolve the XOR problem[1] in a bucket of water.

2.3.1 Time Division Multiplexing of the neurons

Many implementations of RC based on TDM of the neurons have been proposed in the literature [17, 1, 6, 5, 24, 26]. On Figure 2.4, one can observe how the neurons can be multiplexed in time. Here, T is the time scale of the input signal, and there are 5 neurons in the reservoir. Therefore, one can see that the allocated window for each of the neurons lasts $\theta = T/N$. Since θ cannot be arbitrarily small in practice[2], this implies that the greater the number of neurons, the greater T and thus the slower the time scale of the input data. This suggests that one should look for a tradeoff between accuracy and speed in data processing for this kind of RC.

[1] The XOR task consists in reproducing the behaviour of a logical XOR gate, which is a task of historical importance for NN [16].

[2] If θ gets too short, it exceeds the bandwidth of measurement devices, so it becomes impossible to measure the neurons.

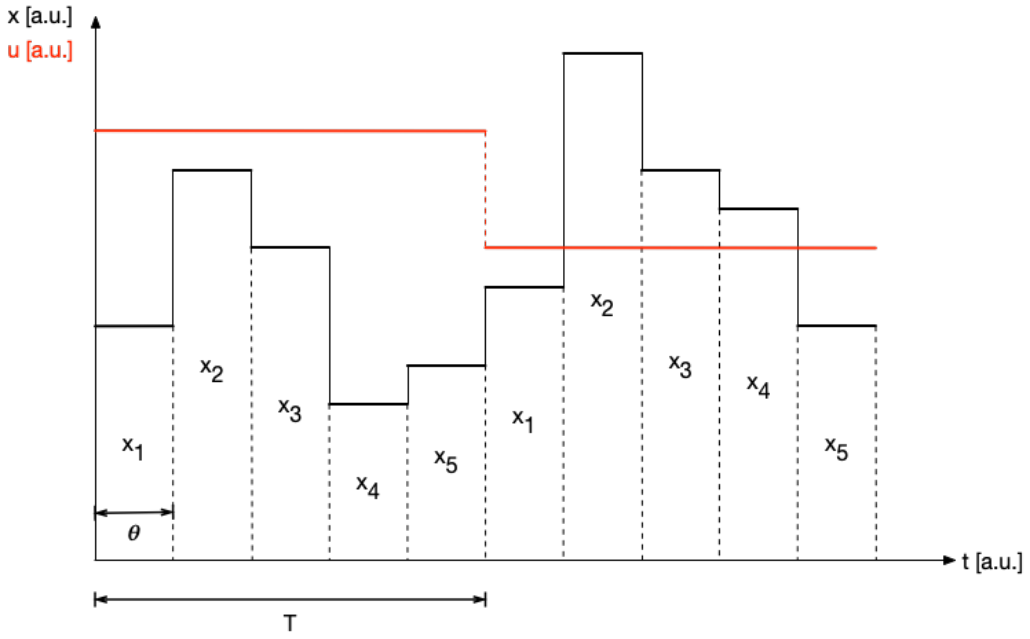

Figure 2.4: Schematic representation of the evolution of the neurons in time for a TDM reservoir

2.3.2 Simplifying assumptions

In this section, the mathematical framework introduced in section 2.2 is adapted to describe the behaviour of TDM PRC in a suitable way. Further details concerning specific types of PRC are added in the following sections which are devoted to them.

Input and connection matrices The input of the reservoir is real scalar function $u(n)$ for PRC, therefore the expression of $\mathbf{W}^{\text{in}}$ becomes a real vector of length N which is called the *input mask* $\mathbf{m}$ in the literature. The input mask can be chosen in different ways: in [6], they use a sinusoidal input mask whereas in [1, 26, 17] the input masks are randomly chosen.

Very few constraints apply to the creation of the connection matrix $\mathbf{W}$. It can be randomly generated and sparse. However, to make the occurence of the echo state more likely to happen, one wants to work with a spectral radius $\rho(\mathbf{W}) < 1$. If this condition is not verified, as well as degrading the performance of the reservoir, this can also lead to instabilities [14].

The matrices $\mathbf{W}$ and $\mathbf{W}^{\text{in}}$ can be rescaled, and this scaling can alter the performances of the reservoir. One should therefore design the PRC in such a way that those scaling factors are easily accessible experimentally. If one defines α and β, the feedback[3] and input gains, the input and connection matrices become:

$$\mathbf{W} \longrightarrow \alpha\mathbf{A}, \quad \mathbf{W}^{\text{in}} \longrightarrow \beta\mathbf{m} \tag{2.7}$$

Feedback and output The output signal of a PRC is a real scalar function $y(n)$, which means that the output matrix $\mathbf{W}^{\text{out}}$ becomes a real vector. Furthermore, the concatenation of $\mathbf{x}(n+1), \mathbf{u}(n+1), y(n)$ appearing in equation (2.3) is not used, but only the state vector $\mathbf{x}$, hence the fact that $\mathbf{W}^{\text{out}}$ is of dimension N. Regarding the feedback of the output into the

[3]This may seem like a misnomer at this point since it has nothing to do with $\mathbf{W}^{\text{fb}}$, but this name is used because α acts as a gain for the activation level of the neurons being fed back into the reservoir.

reservoir, it is currently not doable in practice. This is due to the fact that, when running a RC experiment, some post-processing of the data needs to be performed on a computer in order to obtain the output.

With all these simplifications, equations (2.2) and (2.3) reduce to:

$$\mathbf{x}(n+1) = \mathbf{f}\left(\alpha \mathbf{A}\mathbf{x}(n) + \beta \mathbf{m}u(n+1)\right) \tag{2.8}$$
$$y(n+1) = f^{\text{out}}\left(\mathbf{W}^{\text{out}}\mathbf{x}(n+1)\right) \tag{2.9}$$

2.3.3 Neurons encoded in light intensity

There are two major families of TDM PRC. The first kind of PRC are those using optical components exhibiting nonlinear behaviour, such as Mach-Zehnder Modulator (MZM) [6, 17, 1], Semiconductor Optical Amplifier (SOA) [24] or semiconductor saturable absorber mirror [5] to couple the neurons. In an actual optical experiment, the measurements have to be done with photodiodes. These devices can only inform about the intensity of the light, which is the squared modulus of the phaser representation of the electric field, and not about the actual electric field. However, in the scheme presented above, the input and the activation level of the neurons are real valued functions appropriately encoded in the intensity of the light, and can therefore be directly read out by a photodiode, hence this simple expression for the output of the reservoir:

$$y(n+1) = \sum_{i=1}^{N} W_i^{\text{out}} x_i(n+1) = (\mathbf{W}^{\text{out}})^{\mathrm{T}} \cdot \mathbf{x}(n+1) \tag{2.10}$$

2.3.4 Neurons encoded in phaser representation of the electric field

On the other hand, in [26], the neurons are encoded in the complex phaser representation of the electric field and are linearly coupled using a delay line. In this scheme, the reservoir is linear, as can be seen on equation (2.12). This kind of RC is described with greater length because the new approach studied in this thesis relies on on a linear reservoir as well. In this equation, α and β are the feedback and input gains, respectively A_0 is the input bias and ϕ is the phase acquired by the electric field during by going through the delay line.

$$x_i(n+1) = \begin{cases} \alpha e^{j\phi} x_{i-k}(n) + \beta\left(m_i u(n) + A_0\right) & \text{if } k \leq i \leq N \\ \alpha e^{j\phi} x_{N+i-k}(n-1) + \beta\left(m_i u(n) + A_0\right) & \text{if } 0 \leq i \leq k \end{cases} \tag{2.11}$$

The equation can be rewritten in a compact way:

$$\mathbf{x}(n+1) = \alpha \mathbf{A}\mathbf{x}(n) + \beta \mathbf{m}u(n+1) \tag{2.12}$$

It thus appears more clearly why the condition mentioned previously regarding $\rho(\alpha\mathbf{A})$ is relevant. Indeed, by neglecting the inputs, if $\mathbf{x}_j$ is an eigenvector of $\alpha\mathbf{A}$ with eigenvalue $\lambda_j > 1$, the above equation will lead to an exponential divergence $\mathbf{x}_j(n) \sim \lambda_j^n \mathbf{x}_j(0)$. On the other hand, if λ_j is too small, the state $\mathbf{x}_j$ will be damped too quickly, deteriorating the short-term memory capabilities of the reservoir and preventing it from reaching the echo state.

In the linear reservoir, since the neurons and the input are inscribed in the complex electric field, the nonlinearity is introduced by the read out photodiodes. The output is therefore given by:

$$y(n+1) = \sum_{i=1}^{N} W_i^{\text{out}} |x_i(n+1)|^2 \tag{2.13}$$

From a historical point of view, it is interesting to notice that this reservoir shares features with the first artificial NN, the *perceptron*, introduced by F. Rosenblatt in 1958 [18], which also had a linear connection matrix, and a nonlinear output function.

Equations (2.10) and (2.13) give an interesting intuition on the meaning of the ESN. Indeed, as said previously, when a RC reaches the echo state, each of the neurons tends to systematically reproduce a modified version of the input, the actual modification being an individual characteristic of each neuron [9]. One can see this feature in the perspective of linear algebra. When the RC is fed with an input, it creates a set of time varying functions that can be seen as a basis in which one can try to express the output of the reservoir, which is a vector in the vectorial space of functions. This is why it is often said in the literature that a RC maps an input to a higher dimensional space. Therefore, one can in theory approach any target function with an arbitrary precision, depending on the number of neurons in the reservoir [9]. The higher the number of neurons, the closer to a genuine series development one gets.

2.3.5 Simulations

In this section, the performance of the linear reservoir with quadratic output from [26] are estimated with simulations. Two benchmark task are tackled, first NARMA10 and then nonlinear channel equalisation.

NARMA10

Nonlinear Auto-Regressive Moving Average (NARMA) 10 is a model often used to simulate chaotic time series because it exhibits similar properties [17]. If $u(n)$ is a random variable uniformly distributed along the interval $[-0.5, 0.5]$, the recurrent equation for NARMA10 reads:

$$\hat{y}(n+1) = 0.3\hat{y}(n) + 0.05\hat{y}(n)\left(\sum_{i=0}^{9} \hat{y}(n-i)\right) + 1.5u(n-9)u(n) + 0.1 \tag{2.14}$$

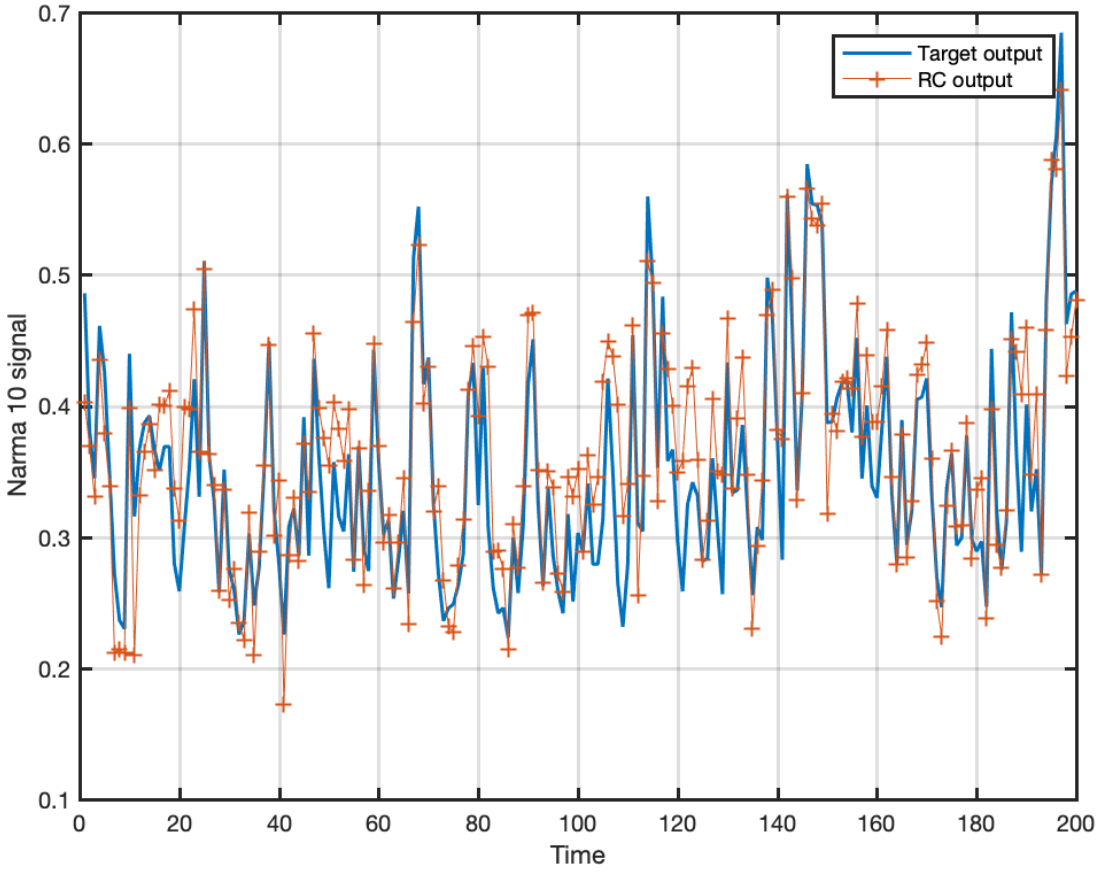

Figure 2.5: NARMA10 task with NMSE equal to 0.1541. This reservoir is made of 50 neurons. $\alpha = 0.5$, $k = 7$, $\phi = 0$ rad, $\beta = 1$, $A_0 = 1$, $\mathbf{m}$ is a random vector distributed between 0 and 1. The first 300 time steps were discarded in order to let enough time to the reservoir to enter the echo state (washout). Then the reservoir was trained for 3000 time steps and tested over 6000 time steps. This reservoir is particularly well suited for NARMA10 since the nonlinearities in the signal are mostly quadratic.

Nonlinear channel equalisation

This task consists in the reconstruction of a signal after it has travelled through a nonlinear channel. The emitted signal $\hat{y}$ is randomly drawn from the symbol set $\{-3, -1, 1, 3\}$. It is first superposed with following and preceding symbols as can be seen in (2.15). After that, a third degree polynomial transformation is applied to the mixed signals, and a Gaussian noise, whose intensity can be set in order to adjust the signal to noise ratio, is added in (2.16). This metric used to evaluate the performance of the reservoir for this type of task is the Signal Error Rate (ser) and is defined as the ratio between the number of erroneous symbols and the total number of symbols.

$$\begin{aligned} q(n) = {} & 0.08\hat{y}(n+2) - 0.12\hat{y}(n+1) + \hat{y}(n) + 0.18\hat{y}(n-1) \\ & - 0.1\hat{y}(n-2) + 0.091\hat{y}(n-3) - 0.05\hat{y}(n-4) \\ & + 0.04\hat{y}(n-5) + 0.03\hat{y}(n-6) + 0.01\hat{y}(n-7) \end{aligned} \tag{2.15}$$

$$u(n) = q(n) + 0.036q(n)^2 - 0.011q(n)^3 + \nu(n) \tag{2.16}$$

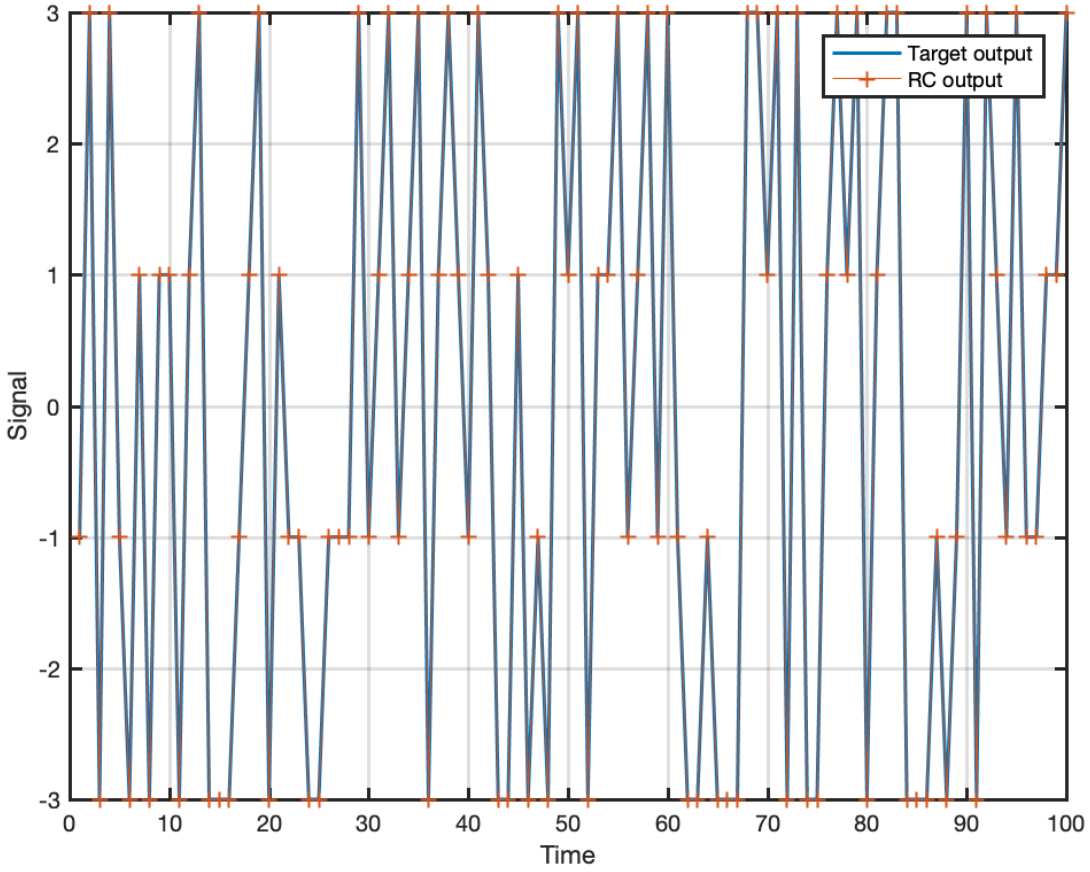

Figure 2.6: Nonlinear channel equalisation task with a signal error rate of 5 10^{-4}, with signal to noise ratio equal to 32 dB. This reservoir is made of 50 neurons. $\alpha = 0.5$, $k = 7$, $\phi = 0$ rad, $\beta = 1$, $A_0 = 1$, $\mathbf{m}$ is a random vector distributed between 0 and 1. The first 300 time steps were discarded in order to let enough time to the reservoir to enter the echo state (washout). Then the reservoir was trained for 3000 time steps and tested over 6000 time steps. The symbols predicted by the RC are found by changing the continuous valued output by the closest symbol.

Chapter 3

Wavelength Division Multiplexing neurons Reservoir Computer

3.1 Introduction

3.2 Mathematical model

3.2.1 Effect of a Phase Modulator

3.2.2 Mathematical model

3.3 Challenges

Chapter 4

Interferometric stabilisation of reservoir cavity

4.1 Experimental setup

4.2 Characterisation of the reservoir

4.2.1 Introduction

4.2.2 Transfer function of the cavity

Mathematical model

Simulations

Experimental results

4.2.3 Effective losses

4.2.4 Modulation depth

4.3 Pound-Drever-Hall stabilisation technique

4.3.1 Introduction

4.3.2 Error function

Mathematical model

Simulation

Experimental results

4.4 Characterisation of the stabilisation performance for different regimes

4.4.1 Introduction

4.4.2 Approach

4.4.3 Results

Chapter 5

Conclusion

Acronyms

ESN Echo State Network 7, 9, 14

ML Machine Learning 4, 8, 10

MZM Mach-Zehnder Modulator 13

NARMA Nonlinear Auto-Regressive Moving Average 14, 15

NMSE Normalised Mean Square Error 11, 15

NN Neural Network 4, 7, 8, 11, 14

PDH Pound-Drever-Hall 4, 19

PM Phase Modulator 4, 17

PRC Photonic Reservoir Computing 4, 11–13

RC Reservoir Computer 4, 7–11, 13, 14, 16, 17

RC Reservoir Computing 4, 7–9, 13, 14

RNN Recurrent Neural Network 7

ser Signal Error Rate 15

SOA Semiconductor Optical Amplifier 11, 13

TDM Time Division Multiplexing 4, 11–13

VCSEL Vertical-Cavity Surface-Emitting Laser 11

WDM Wavelength Division Multiplexing 4, 17

Bibliography

[1] Piotr Antonik et al. "Online Training of an Opto-Electronic Reservoir Computer Applied to Real-Time Channel Equalization". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.11 (Nov. 2017), pp. 2686–2698. DOI: 10.1109/tnnls.2016.2598655. URL: https://doi.org/10.1109/tnnls.2016.2598655.

[2] A. Bernal, S. Fok, and R. Pidaparthi. "Financial Market Time Series Prediction with Recurrent Neural Networks". In: (2012). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.278.3606&rep=rep1&type=pdf.

[3] Christopher Bishop. *Pattern recognition and machine learning*. New York: Springer, 2006. ISBN: 978-0387-31073-2.

[4] Daniel Brunner and Ingo Fischer. "Reconfigurable semiconductor laser networks based on diffractive coupling". In: *Optics Letters* 40.16 (Aug. 2015), p. 3854. DOI: 10.1364/ol.40.003854. URL: https://doi.org/10.1364/ol.40.003854.

[5] Antoine Dejonckheere et al. "All-optical reservoir computer based on saturation of absorption". In: *Optics Express* 22.9 (Apr. 2014), p. 10868. DOI: 10.1364/oe.22.010868. URL: https://doi.org/10.1364/oe.22.010868.

[6] François Duport et al. "Fully analogue photonic reservoir computer". In: *Scientific Reports* 6.1 (Mar. 2016). DOI: 10.1038/srep22381. URL: https://doi.org/10.1038/srep22381.

[7] Chrisantha Fernando and Sampsa Sojakka. "Pattern Recognition in a Bucket". In: *Advances in Artificial Life*. Springer Berlin Heidelberg, 2003, pp. 588–597. DOI: 10.1007/978-3-540-39432-7_63. URL: https://doi.org/10.1007/978-3-540-39432-7_63.

[8] Alireza Goudarzi et al. "A Comparative Study of Reservoir Computing for Temporal Signal Processing". In: *CoRR* abs/1401.2224 (2014).

[9] H. Jaeger. "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication". In: *Science* 304.5667 (Apr. 2004), pp. 78–80. DOI: 10.1126/science.1091277. URL: https://doi.org/10.1126/science.1091277.

[10] H. Jaeger. *The "echo state" approach to analysing and training recurrent neural networks*. 2001.

[11] Herbert Jaeger. "Adaptive Nonlinear System Identification with Echo State Networks". In: *Proceedings of the 15th International Conference on Neural Information Processing Systems*. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, pp. 609–616. URL: http://dl.acm.org/citation.cfm?id=2968618.2968694.

[12] Herbert Jaeger. "Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach". In: *GMD-Forschungszentrum Informationstechnik, 2002.* 5 (Jan. 2002).

[13] Herbert Jaeger et al. "Optimization and applications of echo state networks with leaky-integrator neurons". In: *Neural Networks* 20.3 (Apr. 2007), pp. 335–352. DOI: 10.1016/j.neunet.2007.04.016. URL: https://doi.org/10.1016/j.neunet.2007.04.016.

[14] M. Lukoševičius and H. Jaeger. "Reservoir computing approaches to recurrent neural network training". In: *Computer Science Review* 3.3 (Aug. 2009), pp. 127–149. DOI: 10.1016/j.cosrev.2009.03.005. URL: https://doi.org/10.1016/j.cosrev.2009.03.005.

[15] M. Lukoševičius, M. Jaeger, and B. Schrauwen. "Reservoir Computing Trends". In: *KI - Künstliche Intelligenz* 26.4 (May 2012), pp. 365–371. DOI: 10.1007/s13218-012-0204-5. URL: https://doi.org/10.1007/s13218-012-0204-5.

[16] Marvin Minsky. *Perceptrons; an introduction to computational geometry*. Cambridge, Mass: MIT Press, 1969. ISBN: 9780262130431.

[17] Y. Paquot et al. "Optoelectronic Reservoir Computing". In: *Scientific Reports* 2.1 (Feb. 2012). DOI: 10.1038/srep00287. URL: https://doi.org/10.1038/srep00287.

[18] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review* (1958), pp. 65–386.

[19] Stuart Russell. *Artificial intelligence : a modern approach*. Upper Saddle River, New Jersey: Prentice Hall, 2010. ISBN: 978-0-13-604259-4.

[20] Guy Van der Sande, Daniel Brunner, and Miguel C. Soriano. "Advances in photonic reservoir computing". In: *Nanophotonics* 6.3 (Jan. 2017). DOI: 10.1515/nanoph-2016-0132. URL: https://doi.org/10.1515/nanoph-2016-0132.

[21] Benjamin Schrauwen, David Verstraeten, and Jan Campenhout. "An overview of reservoir computing: Theory, applications and implementations". In: Jan. 2007, pp. 471–482.

[22] Fabian Triefenbach et al. "Phoneme Recognition with Large Hierarchical Reservoirs". In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty et al. Curran Associates, Inc., 2010, pp. 2307–2315. URL: http://papers.nips.cc/paper/4056-phoneme-recognition-with-large-hierarchical-reservoirs.pdf.

[23] Kristof Vandoorne et al. "Experimental demonstration of reservoir computing on a silicon photonics chip". In: *Nature Communications* 5.1 (Mar. 2014). DOI: 10.1038/ncomms4541. URL: https://doi.org/10.1038/ncomms4541.

[24] Kristof Vandoorne et al. "Toward optical signal processing using Photonic Reservoir Computing". In: *Optics Express* 16.15 (July 2008), p. 11182. DOI: 10.1364/oe.16.011182. URL: https://doi.org/10.1364/oe.16.011182.

[25] D. Verstraeten, B. Schrauwen, and D. Stroobandt. "Reservoir-based techniques for speech recognition". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006. DOI: 10.1109/ijcnn.2006.246804. URL: https://doi.org/10.1109/ijcnn.2006.246804.

[26] Quentin Vinckier et al. "High-performance photonic reservoir computer based on a coherently driven passive cavity". In: *Optica* 2.5 (Apr. 2015), p. 438. DOI: 10.1364/optica.2.000438. URL: https://doi.org/10.1364/optica.2.000438.