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.

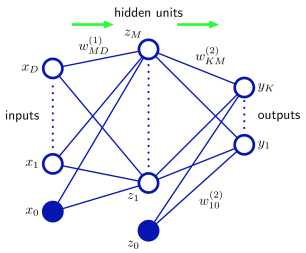
# Chapter 2

# Reservoir Computing

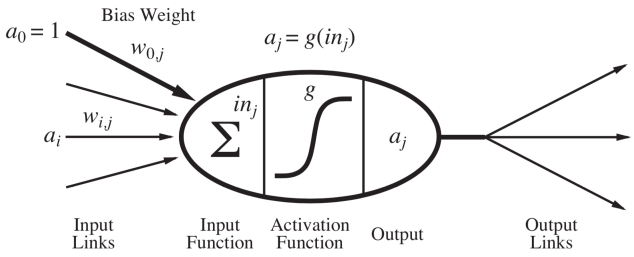
## 2.1 Introduction

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 [8]. This computation scheme is well suited for real-time data processing and for chaotic time series prediction[8, 9, 14], and achieves state of the art performances in those domains, as well as in speech recognition[23, 20, 12], nonlinear channel equalisation[8] and financial forecasting [2].

A Reservoir Computer (RC) is specific kind of Neural Network (NN), which is a computation paradigm mimicking the behaviour of a biological brain. As can be seen on Figure 2.1a, the artificial neurons are merely interconnected entities carrying an activation level. As shown on Figure 2.1b, the activation level of a neuron is updated according to the connection weights of the network, also known as the synaptic matrix, and with a nonlinear function, called the *activation function*, which allows NN to perform classification tasks [3, p.225][18, p.727].




(a) High-level overview a feedforward NN. Input data is fed into input neurons, and the result of the computation is read on the activation level of output neurons.
[3, p.228]




(b) Update of the activation level of a neuron. The activation level of the neurons from the previous layer are linearly combined through the input links, and a nonlinear function is applied to the result of the sum.
[18, p.728]

Figure 2.1: Neural Network seen from different points of view

The activation level of the neurons making up the reservoir characterise its state, which is a time-dependent object. The neurons are interconnected in such a way that they influence the dynamic of each other, leading to a complicated evolution of the state of the reservoir. What a first glance may seem to be a mathematical nightmare turns out to be the main advantage of RC. Indeed, when the reservoir is properly set up, the activation level of each of the neurons becomes a systematic transformed version of the input signal [8]. This is called the echo state

and this is the regime where RC reach their best performance. This is due to the fact that this is an operating point where the transients caused by the inputs are neither amplified nor damped, somehow providing a memory to the reservoir [7, 9]. The output of a RC is obtained by adequately combining the activation state of each neurons. The ideas developed in this paragraph are illustrated on Figure 2.2.

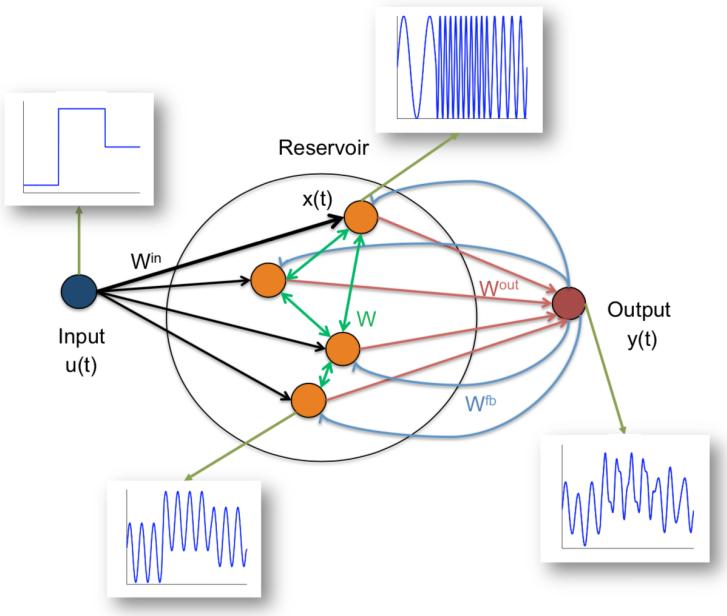
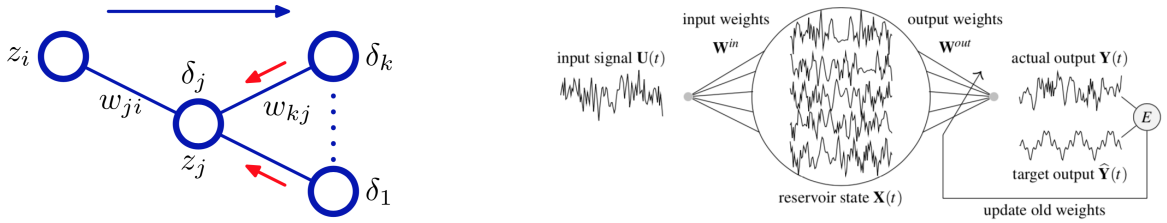



Figure 2.2: Principle scheme of a Reservoir Computer [2]

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 coefficients in the synaptic matrix of the NN [3, p.233][18, p.733]. On Figure 2.3a, the learning procedure for a small portion of a feed-forward NN, for instance the one from Figure 2.1a, is shown. The NN is fed with data from the left, as can be seen with the blue arrows. The red arrows represent the error on the output being *backpropagated*, which means that the learning algorithm evaluates how each connection weight of the synaptic matrix should be updated in order to decrease the output error [3, p.241]. This procedure often turns out to be a really complicated task, which explains why the development of efficient Machine Learning (ML) algorithms is such a hot topic nowadays. On the other hand, as can be seen on Figure 2.3b, RC only need their output weights to be adjusted when being trained, which makes them computationally lighter[8]. This is due to the fact the connections of the reservoir should not contain any information about the task, but should only be used to reach the ESN regime, as previously mentioned. There are two main families of training methods for RC [10]. On the one hand, there is the *batch learning*, which comprises the methods requiring to first store a bunch 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 will converge to workable output weights. They are much more adaptable than the batch learning ones, however, their convergence is not guaranteed and can be slow [11, 19].




(a) Neural Network [3, p.244]

(b) Reservoir Computer [7]

Figure 2.3: Machine Learning for different kinds of Neural Networks

In order to reach the echo state regime, the constraints on the coupling between the neurons of the reservoir are actually quite loose. What is required is to introduce randomness and to break symmetries in the synaptic matrix. Once this condition is verified, the performance of the reservoir can be modified by tweaking only a small set of variables applying to the whole RC. In other words, as soon as the basic structure of the reservoir is set, there is no need to individually change the value of each connection to improve it, only modifying globally the reservoir is enough, for example by multiplying all the connection weights by the same constant. This property is a key element of the RC paradigm that allows it to be implemented in physical systems other than classical, silica-based computers. This has already been done several times in photonic experiments, leading to the novel concept of Photonic Reservoir Computing (PRC). Different implementations have been proposed: fully integrated photonic chips [21], fibre-based systems with time-multiplexed neurons coupled through a delay line with nonlinearities introduced by a Mach-Zehnder intensity modulator [16, 1, 5], by saturation of absorption [4, 22] and by the readout photodiodes [24]. This latter configuration even reached state of the art performances in different benchmark tasks, such as Memory Capacity Evaluation, NARMA10 (Nonlinear Auto-Regressive Moving Average of order 10), nonlinear channel equalisation, and isolated spoken digit recognition. In [6], the researchers even manage to perform speech recognition and the XOR task[1] in a bucket of water.

## 2.2 Mathematical Model

In this section, an overview of the mathematical framework linked to RC is given. First, 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 $i^{\text{th}}$ neuron, the the state vector reads as follows:

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

[1]The XOR task consists in reproducing the behaviour of a logical XOR gate, which is a task of historical importance for NN [15].

The dynamics governing the state vector and the output of the reservoir proposed in [9] 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 will be specified later.

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

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

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
- $\mathbf{x} \in \mathcal{S} \subseteq \mathbb{C}^N$ 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
- $\mathbf{f} : \mathbb{C}^N \mapsto \mathcal{S}$ 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

### 2.2.1 Simplifying assumptions for Optical RC

In what follows, the input and the output of the reservoir are just scalar values, which simplifies the previous equations. Thus, $\mathbf{W}^{\text{in}}$ becomes a simple 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 [5], they use a sinusoidal input mask whereas in [1, 24, 16] the input masks are randomly chosen. Once this vector is fixed, it should be multiplied by a constant to optimise the performance of the RC. 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 [13]. Therefore, once again, once the matrix $\mathbf{W}$ is fixed, one can multiply it by another constant for the same reason mentioned before. Thus, by defining $\alpha$ and $\beta$, the feedback[2] and input gains, respectively:

[2]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 $\alpha$ acts as a gain for the activation level of the neurons being fed back into the reservoir.

$$\mathbf{W} \longrightarrow \alpha\mathbf{A}, \quad \mathbf{W}^{\text{in}} \longrightarrow \beta\mathbf{m} \tag{2.4}$$

This is an example of what was introduced in the previous section. Indeed, the difficulty of tweaking each and every coefficients in order to improve the reservoir is transferred to just a few global variables. This is of great practical importance for physical implementations of RC. Indeed, the experimentalist can alter the behaviour of its RC by only modifying a few global experimental variables.

In practice, the feedback of the reservoir output is neglected, especially for physical implementations, so $\mathbf{W}^{\text{fb}}$ should be null. Also, the concatenation is not observed. Indeed, when computing the output, only state vectors are considered. Speaking of them, they belong to $\mathcal{S}$ because the fact that the functions $\mathbf{f}$ used in practice are bounded, so the vectors $\mathbf{x}$ span a space *smaller* than $\mathbb{C}^N$. The simplified version of the previous equations reads:

$$\mathbf{x}(n+1) = \mathbf{f}\left(\alpha\mathbf{A}\mathbf{x}(n) + \beta\mathbf{m}u(n+1)\right) \tag{2.5}$$
$$y(n+1) = f^{\text{out}}\left(\mathbf{W}^{\text{out}}\mathbf{x}(n+1)\right) \tag{2.6}$$

As far as PRC are concerned, two approaches to determine the functions $\mathbf{f}$ and $f^{\text{out}}$ are followed in the literature. The first kind of PRC are those using optical components exhibiting nonlinear behaviour, such as Mach-Zehnder intensity modulators [5, 16, 1], semiconductor optical amplifiers [22] or semiconductor saturable absorber mirror [4] 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 objects 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:

$$y(n+1) = \sum_{i=1}^{N} W_i^{\text{out}} x_i(n+1) \tag{2.7}$$

On the other hand, in [24], the neurons are encoded in the complex phaser representation of the electric field and are linearly coupled using a delay line. This scheme is a linear reservoir:

$$\mathbf{x}(n+1) = \alpha\mathbf{A}\mathbf{x}(n) + \beta\mathbf{m}u(n+1) \tag{2.8}$$

Therefore, it appears more clearly why the condition mentioned previously regarding $\rho(\alpha\mathbf{A})$ is relevant. Indeed, 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 $\lambda_j$ is too small, the state $\mathbf{x}_j$ will be damped too quickly, deteriorating the 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:

$$y(n+1) = \sum_{i=1}^{N} W_i^{\text{out}} |x_i(n+1)|^2 \tag{2.9}$$

The latter scheme is of particular interest for the remaining of this work, because the novel proposed implementation relies on the same principle. It is also interesting to notice that is quite similar to the first NN paradigm, the *perceptron*, introduced by F. Rosenblatt in 1958 [17].

Equations (2.7) and (2.9) 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 [8]. 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 intuitively be seen as a basis in which one can try to express the output of the reservoir, which is nothing but 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 [8]. The higher the number of neurons, the closer to a genuine series development one gets. Still in the same perspective, the $W_i^{\text{out}}$ can be seen as the Fourier coefficients of the output function.

To determine the output matrix $\mathbf{W}^{\text{out}}$, as said in the previous section, there exists several techniques, but the underlying intuitive idea is always the same. One wants to minimise the deviation between the actual output and the target. Different metrics can be used to encapsulate the distance between those values. In the literature, one of the most frequent is the NMSE, for Normalised Mean Square Error:

$$\text{NMSE} = \frac{\langle \|\hat{\mathbf{y}}(n) - \mathbf{y}(n)\|^2 \rangle_n}{\langle \|\hat{\mathbf{y}}(n) - \langle \hat{\mathbf{y}}(n) \rangle_n \|^2 \rangle_n} \tag{2.10}$$

With $\hat{\mathbf{y}}(n)$ the target vector, $\langle \dots \rangle_n$ the average with respect to $n$, and $\| \dots \|$ the euclidean norm [5]. Different optimisation algorithms can be used for the optimisation in practice, but their description is out of the scope of this work, see [13] for more details.

## 2.3 Simulations

In this section, two different benchmark tasks on which RC are evaluated are briefly introduced, and then the results obtained in numerical simulations are shown. Both of them are implementations of the discrete model described in [24], which is the linear reservoir with the quadratic output for which an overview was given in the previous section:

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

Here, the $i^{th}$ neuron is sent to another one at each update of the state vector, with $k$ being the detuning parameter determining to which one it is mapped. $e^{j\phi}$ is the phase acquired by the complex electric field when it propagates in the delay line. $A_0$ is the input bias.

### 2.3.1 NARMA10

Nonlinear Auto-Regressive Moving Average of order 10 is a model often used because it exhibits a similar behaviour to that of a time series [16]. If $u(n)$ is a random variable uniformly distributed along the interval $[-0.5, 0.5]$, the recurrent equation for NARMA10 reads:

$$\hat{y}(n+1) = 0.3\hat{y}(n) + 0.05\hat{y}(n) \left( \sum_{i=0}^{9} \hat{y}(n-i) \right) + 1.5u(n-9)u(n) + 0.1 \tag{2.12}$$

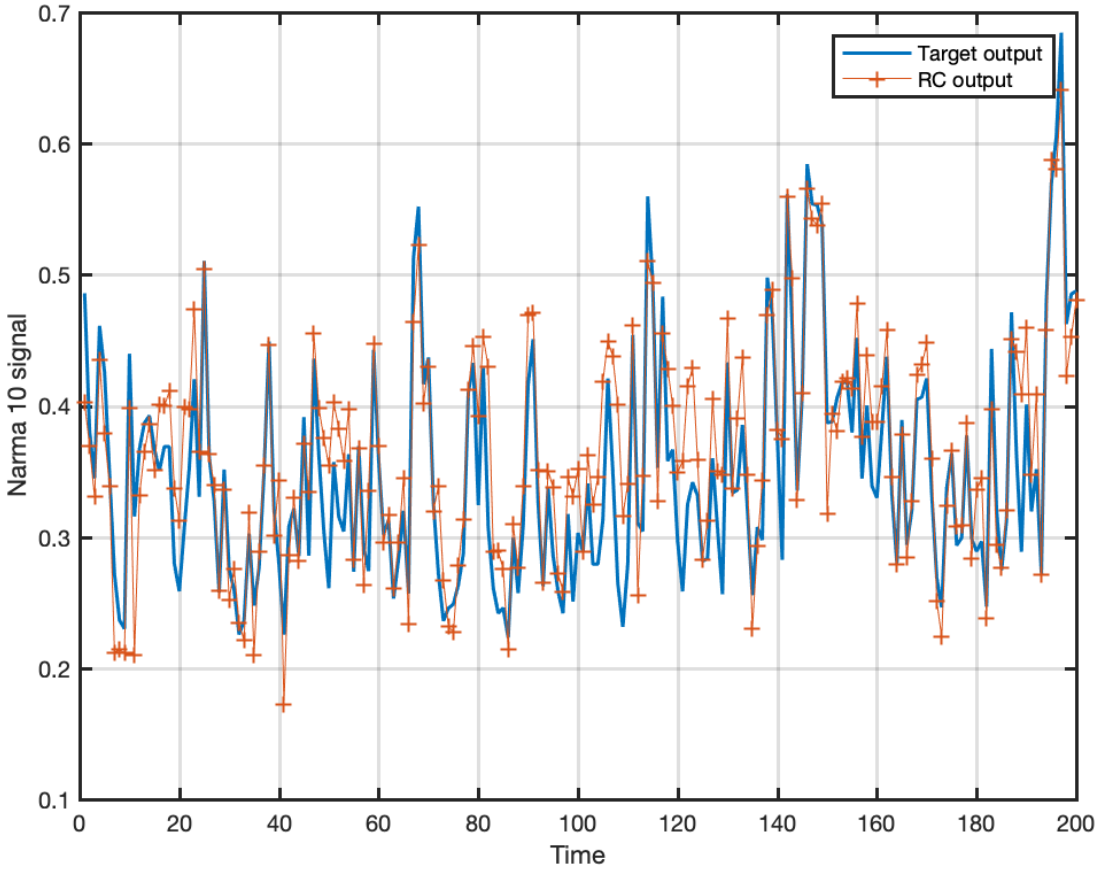



Figure 2.4: NARMA10 task with NMSE equal to 0.1541. This reservoir is made of 50 neurons. $\alpha = 0.5$, $k = 7$, $\phi = 0$ rad, $\beta = 1$, $A_0 = 1$, $\mathbf{m}$ is a random vector distributed between 0 and 1. The first 300 time steps were discarded in order to let enough time to the reservoir 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 [24].

### 2.3.2 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 (2.13). After that, a nonlinear 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 (2.14).

$$\begin{aligned} q(n) = {} & 0.08\hat{y}(n+2) - 0.12\hat{y}(n+1) + \hat{y}(n) + 0.18\hat{y}(n-1) \\ & - 0.1\hat{y}(n-2) + 0.091\hat{y}(n-3) - 0.05\hat{y}(n-4) \\ & + 0.04\hat{y}(n-5) + 0.03\hat{y}(n-6) + 0.01\hat{y}(n-7) \end{aligned} \tag{2.13}$$

$$u(n) = q(n) + 0.036q(n)^2 - 0.011q(n)^3 + \nu(n) \tag{2.14}$$

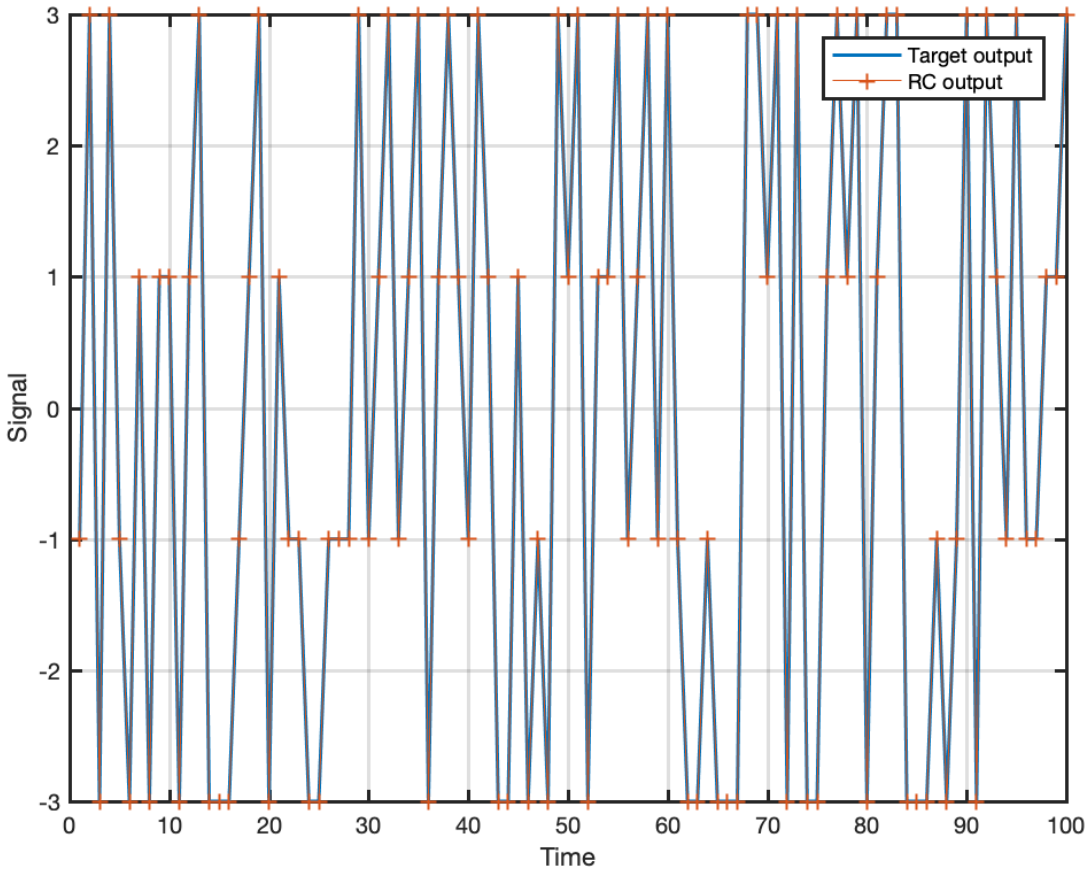



Figure 2.5: Nonlinear channel equalisation task with a signal error rate of 5 $10^{-4}$, with signal to noise ratio equal to 32 dB. This reservoir is the same as the one in Figure 2.4, as well as the number of time steps for the washout, training and testing phases. The symbols predicted by the RC are found by changing the continuous valued output by the closest symbol.

# Chapter 3

# Optical RC with frequency multiplexed neurons

# Chapter 4

# Interferometric stabilisation of RC optical resonator

# Chapter 5

# Results

# Chapter 6

# Conclusion

# Acronyms

**ESN** Echo State Network 6, 7, 11

**ML** Machine Learning 7–9

**NARMA** Nonlinear Auto-Regressive Moving Average 4, 8, 11, 12

**NN** Neural Network 6–8, 10

**PRC** Photonic Reservoir Computing 8, 10

**RC** Reservoir Computer 4, 6–11, 13–15

**RC** Reservoir Computing 4, 6, 8, 11

**RNN** Recurrent Neural Network 6

# 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] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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).

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.