Explorations in Echo State Networks

Adrian Millea

Master's Thesis

Supervised by

Dr. Marco Wiering (Department of Artificial Intelligence, University of Groningen, Groningen, The Netherlands)

Prof. Mark Embrechts (Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy NY, USA)

University of Groningen, Department of Artificial Intelligence
Nijenborgh 9 9747 AG, Groningen, The Netherlands

June 2014

Abstract

Echo State Networks are powerful recurrent neural networks that can predict time-series very well. However, they are often unstable, making the process of finding an ESN for a specific dataset quite hard. We will explore this process, by employing different versions of the activation function, different weight matrices and different topologies. We will show the close connection between the ESN and Compressed Sensing, a recent field in signal processing. Moreover, we will try to tackle some of the main problems in the ESN construction process: minimize the variability between different initializations of the weight matrix, automate the process of finding an ESN without the need for extensive manual trial-and-error sequences and finally eliminate noise from the activation function to increase precision and lower computational costs associated with it. A high level of performance is achieved on many time-series prediction tasks. We also employ the ESN to trade on the FOReign EXchange market using a basic trading strategy, and we achieve significantly more profit compared to previous research

Acknowledgements

I would like to express my appreciation and special thanks to my advisor, Dr. Marco Wiering, you have helped me a tremendous amount ! I would like to thank you for encouraging my research ideas and guiding me to grow as a researcher. Your advices on my research and the extended discussions we had these past few years were priceless for me and I am truly grateful for everything; to you in the first place, but also the University of Groningen for providing such a fruitful and motivating environment. Special thanks also to Prof. Mark Embrechts for very insightful comments and knowledge provided, and also for being present at the thesis presentation through Skype. I would like also to thank my friends Vali, Damian, Florin, Paul, Andrei, Mircea who also helped me a lot in keeping my balance and giving me fresh energy when I needed it the most. My mother and father were there for me all the time, supporting and encouraging me all the way. I am really grateful to have such a loving and caring family. My wife, Raluca, was critical to the success of this thesis, being there in good times and bad times (and the thesis had plenty of each), giving me strength to keep going.

"The great cosmic illusion is a hierophany.... One is devoured by Time, not because one lives in Time, but because one believes in its reality, and therefore forgets or despises eternity."

Mircea Eliade

Contents

1 Introduction **1**

1.1 History . . . 1

1.2 Related Work . . . 2

1.3 Context of Research . . . 2

1.4 Research Questions . . . 3

1.5 Outline . . . 4

2 Reservoir Computing and the Echo State Network **5**

2.1 Introduction . . . 5

2.2 Reservoir Computing . . . 6

2.2.1 Liquid State Machines . . . 6

2.2.2 Echo State Network . . . 6

2.3 Dynamics of the ESN . . . 6

2.3.1 Training . . . 8

2.3.2 Learning . . . 8

2.3.3 Testing . . . 9

2.4 Theoretical Background: Why it Works . . . 9

2.4.1 Bounds for the Echo State Property . . . 9

2.4.2 Dynamic Systems and Computing at the Edge of Chaos . . . 12

2.4.3 Memory Capacity . . . 13

2.5 Tuning of the ESN . . . 15

2.5.1 Spectral Radius . . . 15

2.5.2 Connectivity . . . 15

2.5.3 Weight Scaling . . . 15

2.6 Different Flavors of the ESN . . . 16

2.6.1 Intrinsic Plasticity . . . 16

2.6.2 Leaky Integrator Neurons and IIR Filter Neurons . . . 17

3 Improving the ESN on the MSO problem **20**

3.1 Introduction . . . 20

3.2 Linear ESN . . . 20

3.3 Using Orthonormal Matrices . . . 22

3.4 Related Work: Compressed Sensing (Compressive Sampling) . . . 29

3.4.1 Normed Vector Spaces . . . 29

3.4.2 Bases and Frames . . . 31

3.4.3 Sparse Models . . . 32
3.4.4 Geometry of Sparse Signals . . . 32
3.4.5 Sensing Matrices and Incoherence . . . 33
3.4.6 Nonlinear Recovery Algorithm . . . 35
3.4.7 Dynamical CS Matrices . . . 36
3.4.8 Orthogonal Dynamical Systems . . . 37
3.5 Discussion . . . 40

4 Exploring the ESN 42
4.1 Activation Function . . . 42
4.1.1 Tanh Neurons . . . 42
4.1.2 Linear Neurons . . . 42
4.1.3 Mixing Non-linear and Linear . . . 43
4.2 Feedback Scaling . . . 43
4.3 Different Read-outs . . . 48
4.3.1 Multi Layer Perceptron . . . 48
4.3.2 Ridge Regression . . . 48
4.3.3 Support Vector Machines . . . 50
4.4 Adding Randomness on a Row . . . 51
4.5 Other Datasets . . . 58
4.5.1 Santa Fe Laser . . . 58
4.5.2 Sunspots . . . 60
4.5.3 Mackey-Glass with $\tau = 30$ . . . 60
4.6 Discussion . . . 63

5 Efficient Methods for Finding Good ESNs 65
5.1 Random Optimization . . . 65
5.2 Column Connectivity . . . 66
5.3 Particle Swarm Optimization (PSO) on a Column . . . 67
5.4 Particle Swarm Optimization on a Row . . . 69
5.5 Echo State Networks as Complex Networks . . . 70
5.5.1 Scale-free Models . . . 70
5.5.2 The Erdos-Renyi Graph . . . 71
5.5.3 The Waxman Model . . . 71
5.5.4 The Watts-Strogatz Model . . . 72
5.6 Discussion . . . 73

6 Discussion 75
6.1 Summary . . . 75
6.2 Conclusion . . . 76
6.3 Future Research . . . 77

Chapter 1

Introduction

Machine learning (ML) is one of the main branches of Artificial Intelligence that is concerned with the study of systems which can learn from given data. To learn, in this context, is to be able to use the given data such that the system will pertinently represent the observed behavior according to the given data, and generalize well for unseen data. A particularization of this learning, and also one of the main problems in machine learning, is to predict some future sequence of values from some past sequence of observed values. The process is referred to as time series analysis for prediction. From weather to stock market prediction, useful data is analyzed, modeled in some way such that future predictions are closer and closer to actual events (or values) in the future. Prediction or forecasting as it is often called, can be useful in multiple fields of science, like for example: statistics, econometrics, seismology, geophysics, etc. In machine learning in particular, time series analysis can be employed in many other tasks besides prediction, like clustering, classification or anomaly detection. In this thesis we will deal mainly with the prediction task, but easy extensions to our approach can be imagined such that other types of tasks can be solved. Many approaches have been quite successful in predicting the future behavior of a system for some applications, however some time-series are highly chaotic, or altered by noise, and thus, are much harder to predict. We continue next with a short description of the approaches that have been used before to tackle the time series analysis and prediction problem.

1.1 History

The problem of predicting chaotic time-series is relatively recent, mainly because it is a computational problem, a problem based on data, and just in the late 1980 the computational resources became available for general use. However, analytical models and the theory behind this type of prediction were starting to be popular earlier (1976) with the Box-Jenkins methodology [11], even though the general ARMA (auto-regressive moving average) [114] model which is fundamental for stationary time-series analysis, was described in 1951. An extension to this is the ARIMA model (auto-regressive integrated moving average) which is used for non-stationary time-series analysis [71]. These models are combinations of three main classes of basic models: AR (autoregressive) models, I (integrated) models and MA (moving average) models. Extensions of these exist such that they are able to deal also with multidimensional time-series data (abbreviated with V from vector, e.g. VARMA), and also to be able to include a bias-like component referred to as exogenous models and abreviated with X (eg. XARMA). Later non-linear models used to take into account also the variance of the time-series

over time (called heteroskedasticity). These methods include ARCH [39] and GARCH [10] (which assume some behavior of the error given previous errors). Newer methods make use of the wavelet transform [25], or of hidden Markov models [37] (HMM), neural networks [74], radial basis function networks [23], support vector machines [26], dynamic bayesian networks [75], etc.

1.2 Related Work

A different class of neural networks, enabling more varied and advanced natural phenomena predictions like for example chemical processes [31], geophysical processes [77], physical control processes [64, 65], etc. are recurrent neural networks. One of the best performing recurrent neural networks, which uses a truncated version of gradient based learning are long short-term memory networks (LSTM) [47]. Other types of recurrent neural networks are the Hopfield networks [50] (which are symmetric, and the first ones to appear, in 1982), Elman networks [38] or Jordan networks [57]. Very recently (2001) another type of recurrent neural networks, which does not necessarily employ gradient based learning, and which have been used with great success over the previous approaches for predicting future time-series values are the echo state networks (ESN) [52]. The general problem with echo state networks is that the inner workings of the network are almost always not known to the modeler (this is a general problem pertaining to recurrent neural networks). A few exceptions exist, in which the networks are specially constructed, we will talk about this in chapter 2. Echo state networks function like a black-box, receiving some input sequence and then performing some unknown transformations on it. Such network are so powerful, that usually the training method employs just a linear regression after the input feeding process. No training takes place in the inner networks, but some weights are assigned to each neuron common to all time-steps as to sum up to the desired output signal. The problem is that when trying multiple initializations of the network too much variability in performance is encountered, and thus, often just the minimum error is taken from a series of repetitions. Training methods exist also for the inner weights, but they are generally tedious and time-consuming, because of the recurrence in the network (more details are given in chapter 2). We will describe now the general context of the research present in this thesis and the main motivation for it.

1.3 Context of Research

We talked about the variability problem and we describe next the practical approach to finding a feasible solution for a specific time-series. Using machine learning terminology, we could say that we will describe the biases that can be tuned such that the ESN is tailored to some specific time-series. We will describe next shortly each of them and its function in the general functioning of the ESN. For a detailed description see Chapter 2.

- *Regularization* is usually employed in the ESN as ridge regression (or Tikhonov regularization) in the learning step, or as noise added to each neuron at each time step in the training phase. Some kind of regularization is needed to stabilize the solutions (the variability problem) for many time-series. When noise in used, besides computational cost, precision cost is also an issue because the final precision of the solution can never be higher than the magnitude of the noise added [91]. Thus, one unresolved problem is to be able to eliminate noise from the states of the network such that higher precision can be reached.

- *Spectral radius* is the largest among the absolute values from the eigenvalues of the weight matrix. Its value is closely related to the Echo State Property (2.4.1) and can tune the ESN to perform better for shorter or longer timescales.
- *Scaling* is critical for optimal non-linear interactions. There are two scaling parameters, one for the input and if output feedback is present, one for the feedback vector. If input scaling is too small than the dynamics is almost linear, if it is too large then the dynamics is too truncated and a lot of useful information is lost. Different values for the input scaling are used for different time-series for optimal performance. Input scaling is also dependent to a certain degree on the size of the network and on the connectivity, since the amount of interaction between neurons dictates the ultimate contribution of the input to each neuron.
- *Connectivity* is the percent of connections between neurons different than 0. Usually, ESNs are sparsely connected (less than 1% connectivity).
- *Network size* is another important parameter for the capacity of the network. Increasing the size usually increases the ESN capacity.
- *Leaking rate* is the rate at which the neuron values 'leaks' over time, when using leaky-integrator neurons.

The practical approach for finding a good ESN for a specific time-series usually involves a lot of manual tuning for many of the parameters involved. For more details of this process see [67]. Needless to say that this process can take a while, depending on the dataset, and can become frustrating for the researcher. Now that we have set up the context in which our research finds itself, we will proceed to describe exactly what questions we will try to answer.

1.4 Research Questions

We will attempt to answer the following questions:

1. Can we construct some ESNs to minimize the variability between different initializations (the variability problem) ? This would be very useful for real world problems for the following reason: we won't have any values to compare with for the prediction phase, in other words we won't have a test set. Thus, prediction consistency is critical for real world problems.
2. Can we find (time) efficient methods for finding a good ESN for some specific dataset, without extensive manual experimentation ? For different types of time-series or different tasks, the precision need not be too high, but the general shape of the signal will suffice. This means that a method which can trade precision with time efficiency could be useful for example for some control tasks, where time is of the essence.
3. Is noise critical for stabilizing the network and in some cases finding good ESNs ? Adding noise to the state equation has been shown to stabilize the network, but is computationally more expensive and decreases precision [53, 55, 91].

1.5 Outline

We will try to answer these questions by dealing with sequences from 5 time-series generally used in the literature: the Multiple Sumperimposed Oscillations (MSO) problem, the Mackey-Glass (MG) chaotic attractor (two versions of it, one mildly chaotic and one highly chaotic), the Santa Fe laser time-series and the sunspots time-series. In chapter 2 we give a description of Reservoir Computing and Echo State Networks. We describe in detail the dynamics of the ESN, the control parameters and factors which influence performance of the ESN and give the theoretical details on why it works; we end this chapter by showing some successful approaches for dealing with chaotic time-series prediction. In Chapter 3 we investigate the ESN behavior when dealing with the Multiple Superimposed Oscillation problems (MSO) and obtain very good results compared to the previous best results in the literature, by employing a few simple alterations of the ESN. We continue by showing the connection between recurrent neural networks and the new and exciting field of Compressed Sensing (CS); we then explain the basic mathematical theory behind CS and show two succesful approaches of the combination of the two fields: ESNs and CS. In Chapter 4 we explore the state equation and different parameters of the ESN when dealing with various time-series, and finally we discover an interesting perturbation method, that improves a lot the performance of the ESN compared to previous results found in the literature. In Chapter 5 we describe a few (time) efficient methods for finding good echo state networks for some time-series and then we employ models from the field of complex networks to act as ESNs. In Chapter 6 we draw conclusions and we describe future research directions.

Chapter 2

Reservoir Computing and the Echo State Network

2.1 Introduction

Machine learning was dominated a good part of its history by the feed-forward models like artificial neural networks and Bayesian networks to deal with different problems which exist in artificial intelligence or intelligent systems. These are very helpful in dealing with non-temporal problems, however, when an intrinsic temporal dynamics is encountered, some adaptation, simplification or specific modeling choice needs to be done such that time is represented somehow in the non-temporal model. While these networks are in general employed for a variety of statistical pattern recognition tasks, extensions exist, such that they are able to deal also with temporal data, but their performance is not the very best. They usually make use of some iterative unsupervised training scheme, where they are driven by the input until some type of equilibrium or convergence is reached. These are strongly rooted in statistical physics. Other probabilistic models besides Bayesian networks exist, which can include temporal models (Hidden Markov Models [5], Dynamic Bayesian Networks [43]) or models used for probabilistic planning (Markov Decision Processes [7], Partially Observable Markov Decision Processes [95]) or probabilistic generative models: DBNs (Deep Belief Networks) [46], RBMs (Restricted Boltzmann Machines [94]). These approaches are highly valued in some situations, but in many real life problems and contexts, when the operating conditions start to drift away from the training conditions their performance drops significantly (concept drift [104]). They also have an additional overhead of choosing the right parameters for the models and putting together the right choice of training data. Some temporal approaches to neural networks include: time delayed neural networks [109] and recurrent neural networks [80] (among which we find also the long short term memory networks [47]). The most powerful have been shown generally to be the recurrent neural networks even though they suffer from a different type of problem, namely the training approach. Until recently the training of recurrent neural networks was performed using back-propagation through time [88] (which actually means unfolding the network in time, so constructing a much bigger network, and then performing back-propagation on this new network). However, besides the fact that this process is very slow it does not always guarantee a good solution, because of the fading gradient issue [44]. A very new approach to training recurrent neural networks is the reservoir computing approach [68]. In reservoir computing, one recurrent network is created at first, the recurrent connections are directed, so it is

not symmetric, and the inner weights of the network remain fixed throughout the learning process. These function in fact as dynamical systems driven by the input, or from another point of view, as non-linear filters of the input. This learning scheme is usually supervised. This thesis finds itself in one of the two main sub-fields of this new paradigm, namely the echo state approach [52].

2.2 Reservoir Computing

Reservoir computing (RC) is a novel framework to designing and training recurrent neural networks [68]. The relatively simple architecture and design makes this class of neural networks particularly attractive compared to other types of networks, especially considering the training phase which almost always consists of some linear approach, like linear regression, the pseudo-inverse approach, or other such simple methods. However, a problem exists in this framework: understanding the dynamics of such a network. Most approaches using echo state networks initialize the weights (synapses) of the network randomly and a trial-and-error methodology is used for finding a good network for a specific time-series or dataset. In general, echo state networks and liquid state machines [69] are used for pattern classification, dynamic feature extraction, time-series prediction, etc. [52, 55, 59].

2.2.1 Liquid State Machines

Liquid state machines (LSMs) are a type of recurrent neural networks which are part of the reservoir computing paradigm. They were developed by Maass in [69] independent from Jaeger's echo state networks [52, 53]. This is the computational neuroscience approach to RC as this is the primary field of Maass. The LSM transforms the time-varying input, the time-series, into spatio-temporal patterns of activations in the spiking neurons. The LSM was formulated at first as a cortical micro-column and since then, it has been extensively studied in both the field of Artificial Intelligence and also in the field of Computational Neuroscience. This simple learning scheme has been combined very recently with a new and very interesting reinforcement learning approach which drives the local learning of the inner neurons, thus being more and more biologically plausible [66].

2.2.2 Echo State Network

As we mentioned earlier the echo state network (ESN) was developed by Jaeger in [52, 53] independent of Maass' LSMs. One could say that this is a computer scientist's approach to RC, as this is the primary field of Jaeger. The echo state network uses real valued neurons (usually with values between -1 and 1). Otherwise the training procedure is very similar to the LSMs.

2.3 Dynamics of the ESN

The echo state network is a recent type of recurrent network which has a very low computational cost for the training phase. The inner weights of the echo state network are fixed at the beginning of the experiment and then a set of weights (called read-out weights) are trained using some type of linear fitting technique (a nonlinear technique can also be used, usually improving performance)

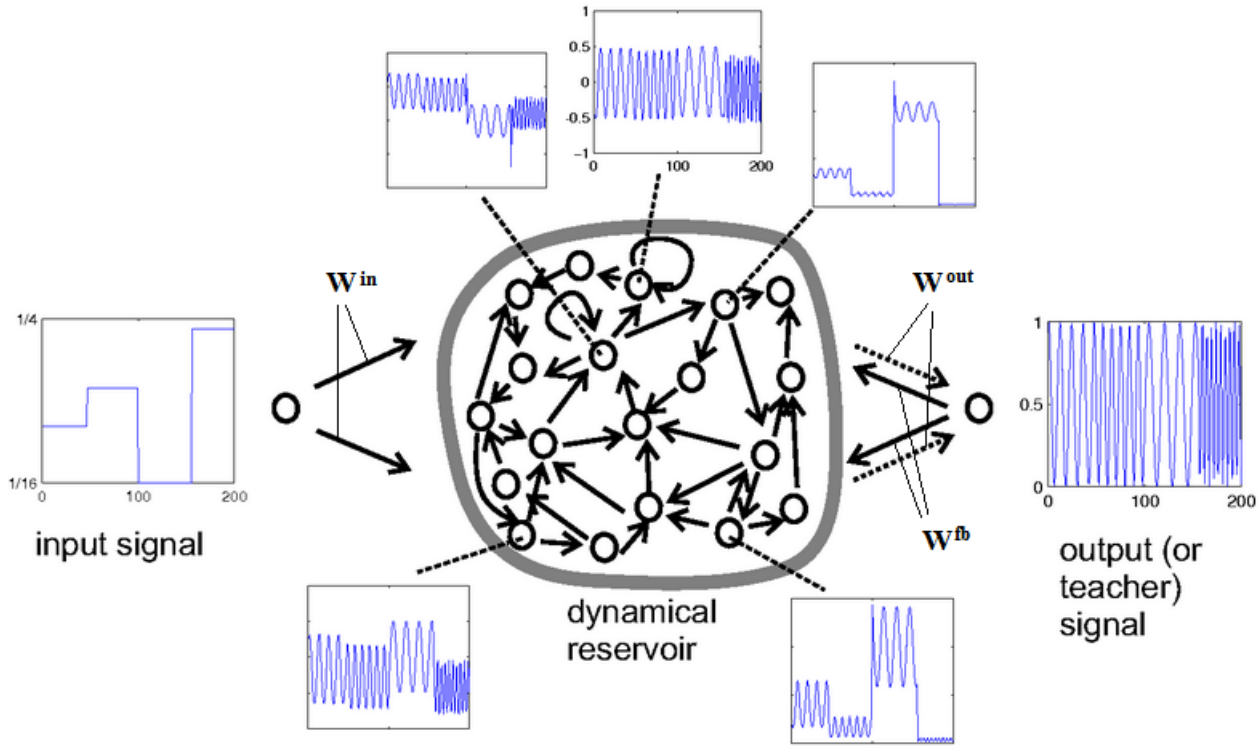


Figure 2.1: Overview of the ESN. Picture taken from http://www.scholarpedia.org/article/Echo_state_network

such that the sum of all neurons, each multiplied by its read-out weight, matches the desired time-series value. The read-out weights are just weightings for each neuron in the overall output. You can see an illustration of the ESN in Figure 2.1. The dotted arrows are the read-out weights, they are the only ones trained during the learning process. The network power comes mostly from the inner dynamics of the network. If the inner weights are 'just right' then the dynamics develops a high memory capacity and can catch the specific features of the input dynamics. The problem is that for the weights to be appropriate for the task, a lot of repetitions with random initializations need to be made. This increases by a high factor the computational overhead needed to find a good echo state network tailored to one specific dataset or time-series. A few approaches exist in the literature which try to improve on finding a good echo state network. For example neurons which act like band-pass filters [92, 49] have been applied successfuly to tune individual neuron signals to specific frequencies, thus decreasing the mutual information of neurons and building a richer inner dynamics from more varied signals. One problem with such an approach is that it takes (computational) time to tune the neurons on specific frequencies. Another type of approach involves evolutionary algorithms, that is training the inner weights in an evolutionary manner, having a fitness function, a population, mutation and crossover rules [86, 105]. This also gives good results compared to the normal approach but again the computational time increases a lot. Yet another good method is having a few networks which have different dynamics of their own and combine them to have the final result; this approach is called decoupled echo state networks [115]. However all of the above approaches seem to have problems obtaining a good performance when dealing with the multiple superimposed oscillations (MSO) problem [48]. In [62] the authors find a way of balancing the echo state network such that errors with a big factor smaller than previously reported errors, are achieved. In this thesis, even much smaller errors are obtained, in our opinion making the MSO problem obsolete. We believe that after we report our current findings, the MSO problem will no longer be used as a benchmark problem, or

the benchmark will be modified in some way, taking into account the size of the training sequence or the size of the testing sequence. Now we proceed with giving the formal description of the echo state network.

2.3.1 Training

The main equation of the echo state network, where we do not use any inputs, but just the output feedback, is:

$$\mathbf{x}(\mathbf{t}+\mathbf{1}) = f(W \cdot \mathbf{x}(\mathbf{t}) + W^{fb} \cdot \mathbf{y}(t)) \tag{2.1}$$

or alternatively, with inputs:

$$\mathbf{x}(\mathbf{t}+\mathbf{1}) = f(W^{in} \cdot \mathbf{u}(\mathbf{t}) + W \cdot \mathbf{x}(\mathbf{t}) + W^{fb} \cdot \mathbf{y}(t)) \tag{2.2}$$

where $x(t)$ is the vector containing all the reservoir states at time-step t, W is the reservoir matrix, where every entry W_{ij} corresponds to the connection between neuron i and j, W^{fb} is the feedback vector matrix, and $y(t)$ is the output at time t. In the second version of the equation we see an input at time t, $\mathbf{u}(\mathbf{t})$, multiplied by the input vector W^{in} . This equation represents the initial driving phase of the network, where the output actually functions as an input, driving the dynamics of the network. The function f is usually chosen to be the hyperbolic tangent for the inner neurons ($tanh$) and the identity function for the output neuron. Some noise can also be inserted into the network update rule (equation 2.1), which depending on the signal might be beneficial or not. It is obvious that when the noise has a large value, the network does not perform well at all, however the noise is usually taken to be around 10^{-6}-10^{-7}. The network is then let to run for a number of steps and the states are collected in a state matrix M which has on each row the state vector $x(t)$, at each time step t. So on columns it has each neuron's state. Therefore it is a matrix of $training_steps$ rows and $network_size$ columns. We have to mention here that the first initial steps of the network are discarded when constructing the matrix M with the purpose of washing out the initial state, which is usually $[0, 0...0]_n$, with $n = network_size$. The number of discarded steps usually depends on the nature of the time-series, as more chaotic ones tend to need more discarded initial steps than simpler functions.

2.3.2 Learning

After collecting the states in all time steps, the usual procedure is performing a simple pseudo-inverse operation:

$$W^{out} = pinv(M) * T \tag{2.3}$$

where W^{out} is the read-out vector, and T is the desired output vector (a 1-by-m vector, where m is the size of the training sequence, the sequence where the input is known, not computed). So, to sum it up: we have a set of m equations with n unknowns, where n is the number of neurons, the size of W^{out}, and the entries of W^{out} are the respective weightings of the neurons' states. The pseudoinverse, or Moore-Penrose pseudoinverse, is a generalization of a matrix inverse, but for matrices which are

not rectangular. Let $\mathbf{A}$ be a $m \times n$ matrix, then the Moore-Penrose inverse is unique, we denote it $\mathbf{A}^*$, it has the size $n \times m$ and it satisfies the following four conditions:

$$\begin{aligned} 1.\mathbf{A}\mathbf{A}^*\mathbf{A} &= \mathbf{A} \\ 2.\mathbf{A}^*\mathbf{A}\mathbf{A}^* &= \mathbf{A}^* \\ 3.(\mathbf{A}^*\mathbf{A})^T &= \mathbf{A}^*\mathbf{A} \\ 4.(\mathbf{A}\mathbf{A}^*)^T &= \mathbf{A}\mathbf{A}^* \end{aligned} \tag{2.4}$$

2.3.3 Testing

After this, the network is again let to run on the test data, which has as initial condition the last training time step (so the neurons' states at time 0 in the testing phase are the neurons' states from time m in the training phase). The difference is now that the output is computed by the network using the W^{out} computed before, so it is not known like before. The equations for the test phase are:

$$\widehat{\mathbf{y}}(\mathbf{t}) = \mathbf{f}^{out}\left(\mathbf{x}(\mathbf{t}) * W^{out}\right) \tag{2.5}$$

$$\mathbf{x}(\mathbf{t}+\mathbf{1}) = \mathbf{f}(W \cdot \mathbf{x}(\mathbf{t}) + W^{fb} \cdot \widehat{\mathbf{y}}(t)) \tag{2.6}$$

As you see the equation is the same, just $\widehat{y}$ is the calculated output after the pseudo-inverse calculation. In our equations (and code) from Chapter 3 we use an identity output function, however in equation 2.3, some non-linear transformation can be applied, like for example the *tanh*. Also when computing the read-out weights (W^{out}) we could use a non-linear technique, like a multi-layer perceptron, or an SVM, or ridge regression, but we will discuss about this in more detail later. Finally, to evaluate the network, we usually calculate the Normalized Root Mean Squared Error (NRMSE) which is:

$$NRMSE = \sqrt{\frac{\|\widehat{\mathbf{y}} - \mathbf{y}\|_2^2}{m * \sigma_y^2}} \tag{2.7}$$

where σ_y^2 is the variance of the desired output signal $\mathbf{y}$, m is the testing sequence length, $\mathbf{y}$ is the desired output, and $\widehat{\mathbf{y}}$ is the output computed by the network after learning ($\mathbf{y}$ and $\widehat{\mathbf{y}}$ are both vectors of length m).

2.4 Theoretical Background: Why it Works

2.4.1 Bounds for the Echo State Property

In the initial formulation of the echo state network, Jaeger defines the echo state property, which in short says that any network associated with a weight matrix satisfying certain algebraic properties, related to the singular values of the matrix, will forget its initial conditions and be completely driven by the input signal. If the reservoir matrix has a spectral radius (the spectral radius is defined as the maximum absolute value of all the eigenvalues of a matrix) bigger than 1, and the input signal contains the 0 value, then the network does not have the echo state property, for a proof see [51]. In the literature there is a misconception about the spectral radius which has to be smaller than 1,

however, with different inputs, the echo state property is satisfied by many matrices, even if they have the spectral radius bigger than 1. We will now proceed with a formal definition of the echo state property as stated in [51].
Let X and U be two compact sets with $X \subset \mathbb{R}^N$ and $U \subset \mathbb{R}^K$ and $f(x_k, u_{k+1}) \in X$ and $u_k \in U, \forall k \in \mathbb{Z}$. The compactness of the state space X is assured by the nature of the transfer function, which is usually *tanh* and which is bounded, and thus satisfies the Lipschitz condition:

$$\begin{aligned}
&d(f(\mathbf{x}', \mathbf{u}), f(\mathbf{x}, \mathbf{u})) \\
&= d(f(W \cdot \mathbf{x}' + W^{fb} \cdot \mathbf{u}), f(W \cdot \mathbf{x} + W^{fb} \cdot \mathbf{u})) \\
&\leq d(W \cdot \mathbf{x}' + W^{fb} \cdot \mathbf{u}, W \cdot \mathbf{x} + W^{fb} \cdot \mathbf{u}) \\
&= d(W \cdot \mathbf{x}', W \cdot \mathbf{x}) \\
&= \left\| W\mathbf{x}' - W\mathbf{x} \right\| \\
&\leq \Lambda d(x', x)
\end{aligned}$$

this means that the distance between two states x' and x, $(d(x', x))$ shrinks with a factor of Λ (the largest singular value of matrix W) at every step, independent of the value of the input. In practice, the input is usually also bounded, so the compactness of U is also assured. Let $U^{-\infty} = u^{-\infty} = (...u_k, ..., u_{-1}, u_0)|u_k \in U,\ \forall k \leq 0$ and $X^{-\infty} = x_{-\infty} = (...x_k, ..., x_{-1}, x_0)|x_k \in X\ \forall k \leq 0$ denote the left infinite input and state vector sequences. We then say that $x^{-\infty}$ is compatible with $u^{-\infty}$ when $x_k = F(x_{k-1}, u_k),\ \forall k \leq 0$. The definition of the echo state property as given in Jaeger [51] follows:

Definition 1. (echo state property, from [117]):
A network $F : X \text{ x } U \to X$ (with the compactness condition) has the echo state property with respect to U if for any left infinite input sequence $u^{-\infty} \in U^{-\infty}$ and any two state vectors sequences $x^{-\infty}$ and $y^{-\infty} \in X^{-\infty}$ compatible with $u^{-\infty}$ it holds that $x_0 = y_0$.

This is mentioned in the literature as the backward-oriented definition. We state next the forward-oriented echo state property (with $U^{+\infty} = \{u^{+\infty} = (u_1, u_2...)|u_k \in U\ \forall k \geq 1\}$ and $X^{+\infty} = \{x^{+\infty} = (x_0, x_1...)|x_k \in X\ \forall k \geq 0$ denoting the right infinite input and state vector sequences):

Theorem 1.1. A network $F : X \text{ x } U \to X$ (with the compactness condition) has the echo state property with respect to U if and only if it has the uniform contraction property, i.e. if there exists a null sequence $(\delta_k)_{k \geq 0}$ such that for all $u^{+\infty} \in U^{+\infty}$ and for all $x^{+\infty}, y^{+\infty} \in X^{+\infty}$ it holds that for all $k \geq 0, \|x_k - y_k\| \leq \delta_k$.

In practice, the usual methodology is to take a random matrix W and then scale it such that its spectral radius $\rho(W)$ is less than 1. However simple and clean this definition is, and even though it is now widely applied in practice, it seems, as proven by later investigations, it is not either sufficient or necessary to ensure the echo state property, as we will see next. However before proceeding we want to give a tighter bound in the same direction from [14], but using a weighted operator norm, or induced norm.

The weighted operator norm

First, we will give the notations used in this section. Let $\mathbb{F} = \mathbb{C}$ *or* $\mathbb{R}$. For a square matrix $W \in \mathbb{F}^{n \times n}$, let $\overline{\sigma}(W)$ denote the largest singular value of the matrix W and $\rho(W) =$ the spectral radius of W as

defined previously, the largest absolute value of the eigenvalues of W. Some scaling matrices will be used, denoted by $\mathcal{D} \equiv \mathbb{F}^{n\times n}$ and $\overline{\mathcal{D}} = \{diag(\delta_1, \delta_2, ...\delta_n), \delta_i \in \mathbb{F}\ for\ i = 1, ..., n\}$. D_δ will be used to denote diagonal matrices in $\overline{\mathcal{D}}$ and D will be used to denote full matrices in $\mathcal{D}$. We should note that in the context of ESNs, $\mathbb{F} = \mathbb{R}$.
In linear algebra there exists a so called induced norm, or operator norm, which for a matrix is arbitrarily close to the spectral radius of the matrix. Formally:

Theorem 1.2. For every matrix W $\in \mathbb{F}^{n\times n}$ and every $\epsilon > 0$, there exists an operator norm $\|\cdot\|_D$ such that

$$\rho(W) \leq \|W\|_D \leq \rho(W) + \epsilon$$

The desired operator norm is achieved by using a weighted operator norm: $\|W\|_D = \left\|DWD^{-1}\right\|$ with $D \in \mathbb{F}$ non-singular, that is specific to the matrix W. This weighted operator norm does not depend on the norm used in the right side of the equality. Any p-norm with $p = 1, 2$ or ∞ can be used. Thus, the weighted operator norm depends on the weighted matrix D which is selected based on the matrix W. Even though the matrix D might change with the type of norm used, all finite-dimensional norms are equivalent so any norm can be used, but the authors of the study described here [14], choose the 2-norm for computational reasons. After this, the operator norm chosen needs to be minimized, and this is done by the choice of the matrix D, which being arbitrary can be chosen such that $\|W\|_D = \overline{\sigma}\left(DWD^{-1}\right)$ satisfies Theorem 1.2. for a given ϵ. If D is allowed to have full structure then:

$$\underset{D\in\mathcal{D}}{inf}\ \overline{\sigma}(DWD^{-1}) = \rho(W)$$

where infimum is used instead of minimum since D (or D^{-1}), in many cases, will be approaching a singular matrix. If $\mathcal{D}$ is a set of matrices which has a structure imposed on it ($\overline{\mathcal{D}}$), then $\|W\|_{D_\delta} = \overline{\sigma}(D_\delta W D_\delta^{-1})$ will not necessarily approach the spectral radius of W. Instead the following is true:

$$\rho(W) \leq \underset{D\in\mathcal{D}}{inf}\ \overline{\sigma}(DWD^{-1}) \leq \overline{\sigma}(W)$$

In this equation the upper bound is obvious since $D_\delta = I$ might be an option. For a more general W, taking the infimum over all possible $D_\delta \in \overline{\mathcal{D}}$ will always be less than $\overline{\sigma}(W)$ and greater than $\rho(W)$. There are, however, some classes of matrices for which the lower bound is exact, these are normal matrices and upper and lower triangular matrices. This leads to a theorem which then leads to a new sufficient condition for the echo state property.

Theorem 1.3. Let W $\in \mathbb{F}^{n\times n}$ be in one of the two classes of matrices:
1) normal matrices
2) upper and lower triangular matrices
Then there exists a $D_\delta \in \overline{\mathcal{D}}$ such that $\|W\|_{D_\delta} = \rho(W) + \epsilon$ for all $\epsilon > 0$.

The proof can be found in [14], as well as the rest of the theory of this new sufficient condition for the echo state property. We now give the actual condition.

Theorem 1.4. Given an echo state network with an internal weight matrix $W \in \mathbb{R}$ and given the squashing function which satisfies the element-wise Lipschitz condition $|f(v) - f(z)| \leq |v - z|, \forall v, z \in$

$\mathbb{R}$. If $\inf_{D \in \overline{\mathcal{D}}} \overline{\sigma}(D_\delta W D_\delta^{-1}) < 1$, then the network has the echo state property, meaning $\lim_{k \to \infty} \|y_k\|_{D_\delta} = 0$ for all the right infinite sequences $u^{+\infty} \in U^{+\infty}$.

2.4.2 Dynamic Systems and Computing at the Edge of Chaos

As we mentioned earlier, echo state networks are in fact dynamical systems driven by the input (or in some cases just/also by the output, when teacher forcing is used). In consequence, some well known approaches from dynamical systems theory can be applied also to the analysis and design of echo state networks. There are many examples in this direction: [100, 76, 99, 8, 56]. The computational power of echo state networks has been proven by many to increase as the regime in which the network functions is close to the critical line which separates the ordered networks from chaotic networks. Multiple ways exist in which the critical line can be estimated. We will not describe in detail the methods used, however there are two approaches which seemed to us to dominate the specific literature. One involves the Hamming distance and the so called Derrida plots [8] which can be used to show the chaotic or ordered functioning of a dynamic system and the other one involves the Lyapunov exponent which again is informative for the underlying dynamics analyzed [56, 108]. We have to mention here that the Lyapunov exponent has been applied to neural networks before the invention of the RC framework [81]. We show in Figure 2.2 examples of three networks with input: an ordered network (left), a chaotic network (right), and a network which is between the two, about which we say is in the critical region between chaos and order. As uncovered in a recent paper on reverse engineering a reservoir

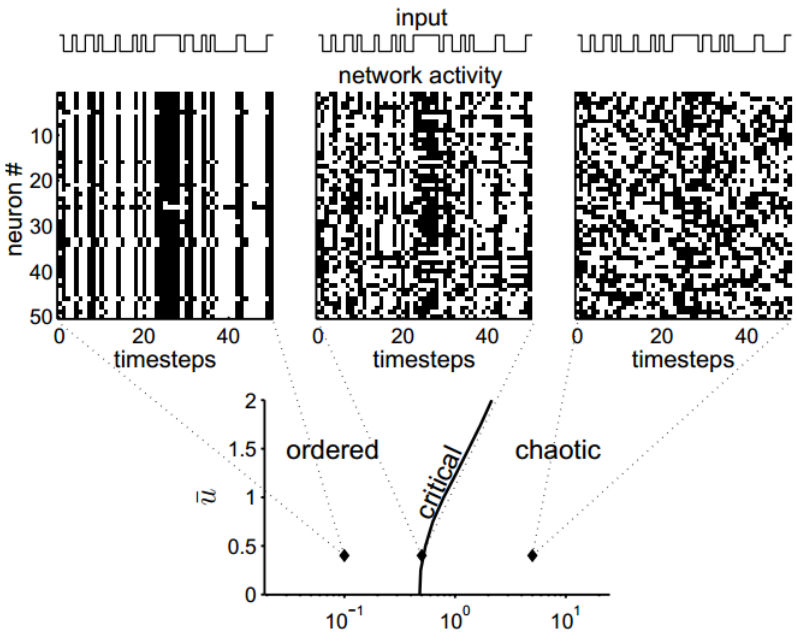
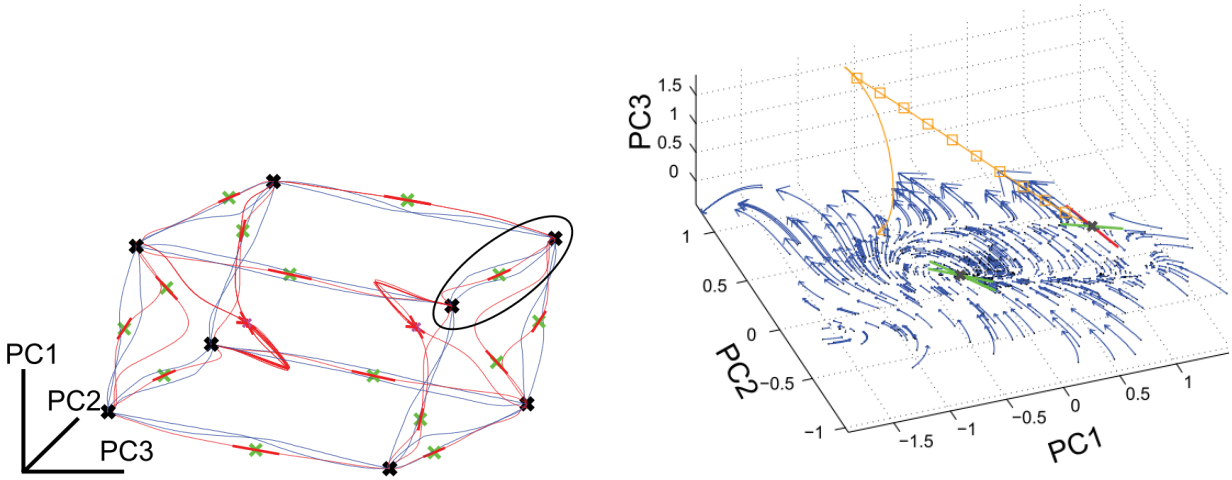


Figure 2.2: Example of ordered (left), critical (middle), and chaotic (right) networks of 50 neurons. On the X-axis is the time and on the Y-axis is the value of the neuron at the respective time-step (0-white, 1-black, the network is binary.). Figure taken from [100].

[100], the memory in such a system is constituted of attractors. Attractors are special subsets of the whole state space to which the system converges (as a function of the input or not) for a certain period of time, usually finite. There are many types of attractors, like point attractors (one dimensional), plane attractors (two dimensional), n-dimensional attractors and strange attractors (also called chaotic attractors). The main idea in such a model, is to construct some energy function of the network, which,

when minimized, gives us the fixed points (or slow points) in which the memory resides. This process usually involves linearization of the system in a small neighborhood of the points of interest, and even though the process of computing the energy function in such sets is not exact, the approximation is sufficiently good. We show in Figure 2.3 graphical examples of such attractors.



(a) Attractors for the 3-bit flip-flop task.

(b) Attractors for the 2-point moving average task.

Figure 2.3: Attractors in the Echo State Network. a) 3-bit flip-flop task. The eight memory states are shown as black x. In blue, we see all the 24 1-bit transitions between the memory states. The points denoted by the green x are the saddle points with one unstable dimension. The thick red line shows the dimension of instability of these saddle points. In thin red lines are shown the network trajectories started just off the unstable dimensions of the saddle points. The state space of the network is plotted on the 3D space defined by the three principal components of the network. b) Example of a 2D plane attractor in a 2-point moving average task. There are two fixed points denoted by the black x, one has two stable dimensions, and one has also one unstable dimension. The blue lines represent trajectories started from the slow points on the manifold. The orange trajectory is showing the system computing the moving average when presented with new input.

2.4.3 Memory Capacity

The memory capacity of an echo state network is a way of measuring the capacity of a network to store previous input values. When computing the memory capacity, multiple independent output neurons are used, and each one is trained on different delays of the single-channel input. This is defined in Jaeger [52] as short-term memory (STM) and the definition is given as:

Definition 2. Let $v(n) \in U$ (where $-\infty < n < +\infty$ and $U \subset \mathbb{R}$ is a compact interval) be a single channel stationary input signal. Assume that we have an RNN, specified by its internal weight matrix $\mathbf{W}$, its input weight (column) vector $\mathbf{w}^{in}$ and the output functions $\mathbf{f}$ and $\mathbf{f}^{out}$ (where $\mathbf{f}$ is the reservoir function, usually $tanh$ as we mentioned above, and $\mathbf{f}^{out}$ is the output function, applied only to the output neuron, usually the identity function). The network receives $v(n)$ as its input; for a given delay k and an output unit y_k with connection weight (row) vector $\mathbf{w}_k^{out}$ we consider the determination

coefficient:

$$d\left[\mathbf{w}_k^{out}\right](v(n-k), y_k(n)) = \\ = \frac{cov^2(v(n-k), y_k(n))}{\sigma^2(v(n))\sigma^2(y_k(n))}$$

where *cov* denotes covariance and σ^2 denotes variance. Then every k-delay is defined as:

$$MC_k = \max_{\mathbf{w}_k^{out}} d\left[\mathbf{w}_k^{out}\right](v(n-k), y_k(n))$$

and then the total Short Term Memory capacity is given by:

$$MC = \sum_{k=1}^{\infty} MC_k$$

MC_k is called the determination coefficient and is actually the squared correlation coefficient. It can take values between 0 and 1, when 0 means no correlation and 1 means complete correlation. In short, it measures how much of the variance of one signal is explained in the other signal. We show in Figure 2.4 some plots of the memory capacity with different settings.

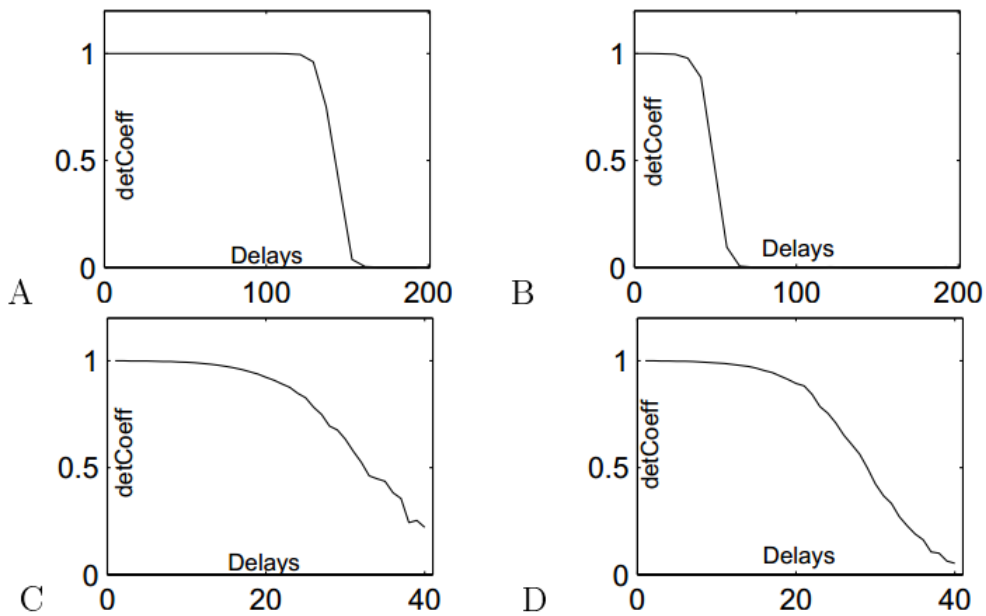


Figure 2.4: A. Forgetting curve of a trained, full-memory capacity linear network with 400 units, with delays up to 200. B. Same as A, but with a sigmoid activation function. C. Same as A but with noisy network update; delays up to 40. D. Same as B but with noisy network update. [picture taken from [52]]

We can conclude from this that linear networks are much more powerful than sigmoidal networks (we will also see this in the next chapter of the thesis); another to-be-expected conclusion is that when adding noise (to both linear and sigmoidal networks) the memory capacity decreases significantly. However, the memory capacity is still much less than the theoretical maximum achievable result, which is N, the number of neurons in the network [52]. In [52] they achieve an almost maximum memory capacity (395) by using a unitary weight matrix (this was done by changing the singular value diagonal matrix of the Singular Value Decomposition (SVD) of W with the unity matrix and then multiplying the resulting matrix with a constant $C = 0.98$).

2.5 Tuning of the ESN

As we mentioned in the introduction, the ESN is usually constructed by manually experimenting with a number of control parameters. We choose to explain in more detail the three most common to all architectures and learning approaches in the ESN literature.

2.5.1 Spectral Radius

As we described in section 2.4.1. the spectral radius is a critical tuning parameter for the echo state network. Usually the spectral radius is related to the input signal, in the sense that if lower timescale dynamics is expected (fast-oscillating signal) then a lower spectral-radius might be sufficient. However if longer memory is necessary then a higher spectral radius will be required. The downside of a bigger spectral radius is bigger time for the settling down of the network oscillations. Translating this into an experimental outcome means having a smaller region of optimality when searching for a good echo state network with respect to some dataset. The spectral radius is considered to be the most important tuning parameter by the creator of the ESN [53]. Having a spectral radius bigger than 1, does not mean that the echo state network thus constructed will be necessary bad, but it gives very inconsistent results, thus making the search for a good ESN a much more random process than it already is.

2.5.2 Connectivity

Connectivity is another important parameter in the design of a good echo state network. Especially if one considers all the possible architectures for the echo state network. Connectivity is defined as the number of non-zero weights from the total number of weights in the network (for example if we have a 10 neuron network, we will have 100 network weights; if we set the connectivity to 0.6 then the number of 0-valued weights will be 0.4 * 100 = 40). As we already mentioned, architectures containing multiple smaller ESNs are possible (DESN [115] or scale-free ESN [31]) where each value of the different connectivity layers might be different from the other ones. In the DESN case, multiple smaller networks (each one might have a different value for the connectivity parameter) are connected to each other through a special set of connections (which is in itself a connectivity parameter), which have the effect of decoupling the functioning of the smaller networks. In the case where one considers orthonormal weight matrices (as we will also do in the next chapter) the connectivity seems to be one of the critical defining parameters for the solution space. This happens only for a linear ESN. In the nonlinear case, when using a *tanh* activation function for example, some researchers have reported no effect of the connectivity value, meaning that fully connected networks perform as good as sparsely connected networks for some specific time-series prediction problems like the MSO [62], however many researchers also reported the connectivity to be of critical importance [96, 54].

2.5.3 Weight Scaling

As stated by Jaeger in the first description of the ESN ([52]) input scaling is very important for the ESN to catch the dynamics of the signal. If the input weights are too small the network will be driven more by the inner dynamics, and lose the characteristics of the signal, if the input weights are too big then there will be no short-term memory and the inner states will be completely driven by the signal.

Developing on this idea further, in a later article [62] they showed that what actually matters most is the ratio between the input signal to the neuron and the signals fed from the other neurons to the respective neuron.

2.6 Different Flavors of the ESN

When constructing echo-state networks, even though some general constraining conditions are outlined by Jaeger [52, 51] and by many others [85, 76, 62], still the problem of adapting the reservoir to a specific task remains unresolved. When performing multiple repetitions, a big variance is encountered, some networks being completely fit for the current task, while others perform pretty bad. This problem has been mentioned in many papers and is referred to as the performance variability of the ESN. Multiple ways of adapting the inner dynamics to a specific task have been proposed, the simplest being local adaptation rules. We will describe next a few of the most interesting approaches which yield good performance with respect to the variability mentioned.

2.6.1 Intrinsic Plasticity

Intrinsic plasticity (IP) was first introduced in [98] based on the neurological process of homeostatic plasticity, however it was for spiking neuron models (biological neurons usually adapt to a fixed average firing rate). In short, this rule is local and unsupervised, and uses, for example, a Fermi transfer function [97]. However, more general characteristics have been outlined later in the literature; we give next the three principles which generally describe an IP rule:
(1) **information maximization**: the output of any neuron should contain as much information from the input as possible; this can be achieved by maximizing the entropy of the output firing rates.
(2) **constraints on the output distribution**: neurons with specific output ranges can be constructed, having specialized sets of neurons, each set having a different output response; this makes sense even biologically.
(3) **adapt the neurons internal parameters**: a biological neuron however, is able to adjust just its internal excitability response, not its individual synapses.
Different versions of the IP rule exist [97, 90, 103, 6], which generally satisfy the above mentioned principles. First, in [1] it has been shown that when an exponential distribution is used for the firing rates, this maximizes the information output of the neurons given a fixed energy budget. A gradient descent rule is usually derived for this type of learning. When considering the maximum entropy distribution with certain moments, for example with a fixed mean and values in the interval $[0, \infty)$ we get the exponential distribution or for a fixed mean and certain standard deviation and values in $(-\infty, +\infty)$ we get the Gaussian distribution for the firing rates. In the first case (when using the first moment only) Fermi neurons can be used, and when adding the second moment, *tanh* neurons can be used. We show below the formulas for the functions and for the gradient update rules:

$$y = f(x) = \frac{1}{1 + exp(-x)} \text{Fermi transfer function also known as a sigmoid function or the logistic function}$$

$$y = f(x) = tanh(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \text{Hyperbolic tangent transfer function}$$

Then the general form of the function is:

$$f_{gen}(x) = f(ax + b)$$

The learning rule for the Fermi neurons and the exponential firing rate (p_{exp}) as in [90] follows (μ is the mean firing rate):

$$p_{exp}(y) = \frac{1}{\mu} exp(-\frac{y}{\mu})(\text{this is the targeted exponential distribution})$$

The details of this derivation can be found in [103]. The gradient descent rules for the gain a and the bias b is (η is the learning rate):

$$\Delta b = \eta(1 - (2 + \frac{1}{\mu})y + \frac{y^2}{\mu})$$
$$\Delta a = \frac{\eta}{a} + \Delta bx$$

To measure the difference between the desired exponential distribution (or Gaussian) and the empirical distribution, Kullback-Leibler divergence is used:

$$D_{KL}(\widetilde{p}, p) = \int \widetilde{p}(y) log \frac{\widetilde{p}(y)}{p(y)} dy,$$

where $\widetilde{p}(y)$ is the empirical probability density of the neuron and $p(y)$ is the desired probability density. For a hyperbolic tangent transfer function, the update rules are as follows:

$$\Delta b = -\eta(-\frac{\mu}{\sigma^2} + \frac{y}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y))$$
$$\Delta a = \frac{\eta}{a} + \Delta bx$$

We can easily see that the relation between Δa and Δb is the same for both desired distributions, in fact they are the same for any desired distribution. We show in Figure 2.5 the actual (estimated) and desired (expected) exponential and Gaussian distributions (see details in the capture of Figure 2.5).

2.6.2 Leaky Integrator Neurons and IIR Filter Neurons

Besides normal *tanh* neurons (which are the usual choice), some other type of neurons can be used, like for example leaky-integrator neurons. Leaky-integrator neurons have an additional parameter which needs to be optimized, the leaking rate; this is the amount of the excitation (signal) a neuron discards, basically it implements the concept of leakage. This has an effect of smoothing the network dynamics, yielding an increased modeling capacity of the network, for example when dealing with a low frequency sine wave. The equations of the continuous time leaky-integrator neurons, described in [55], are:

$$\mathbf{x}(\mathbf{t+1}) = \frac{1}{c}(-a\mathbf{x}(\mathbf{t}) + f(W \cdot \mathbf{x}(\mathbf{t}) + W^{in} \cdot \mathbf{u}(\mathbf{t}) + W^{fb} \cdot \mathbf{y}(\mathbf{t})) \tag{2.8}$$

$$\mathbf{y}(\mathbf{t}) = g(W^{out}[\mathbf{x}; \mathbf{u}]) \tag{2.9}$$

where $c > 0$ is a time constant, $a > 0$ is the leaking rate of the reservoir, and g is the output function which can be the identity function or the *tanh* function. When using an Euler discretization with the

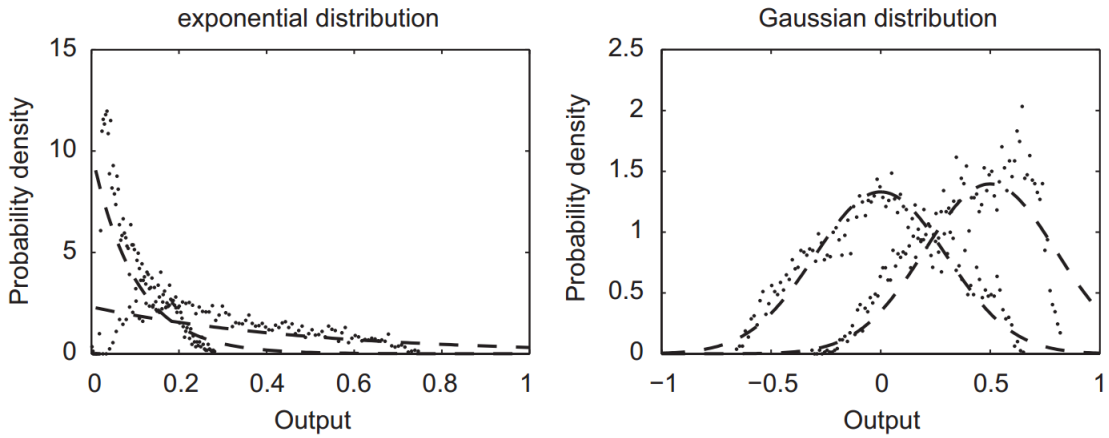


Figure 2.5: Figure taken from [90]. Plots showing a comparison between estimated and expected probability density for a reservoir of 100 neurons during 1000 steps. The estimated distributions (dots) are generated by collecting the neuron outputs in 200 bins. The expected distributions are shown by the dashed lines. For each distribution (exponential and Gaussian) two values of the expected mean are shown.

step size equal to δ and a discrete-time input sample $\mathbf{u}((t)\delta)$ the equations become:

$$\mathbf{x}(t+1) = (1 - \frac{a\delta}{c})\mathbf{x}(t) + \frac{\delta}{c} f(W\mathbf{x}(t) + W^{in}\mathbf{u}((t+1)\delta) + W^{fb}\mathbf{y}(t))$$
$$\mathbf{y}(t+1) = g(W^{out}[\mathbf{x}(t); \mathbf{u}((t\delta))])$$

Introducing $\gamma = \frac{\delta}{c}$ and assuming that W has a unit spectral radius, the new state update equations become:

$$\mathbf{x}(t+1) = (1 - a\gamma)\mathbf{x}(t) + \gamma f(\rho W\mathbf{x}(t) + W^{in}\mathbf{u}((t+1)\delta) + W^{fb}\mathbf{y}(t) + v(t+1))$$
$$\mathbf{y}(t+1) = g(W^{out}[\mathbf{x}(t); \mathbf{u}(t)]$$

where ρ is the effective spectral radius of the weight matrix and $v(t+1)$ an additional noise term. The analogy with the low-pass filter is obvious. We give below the equation of the discrete time low-pass filter with one pole in the transfer function [93].

$$\widetilde{y}(n) = (1 - \theta)\widetilde{y}(n-1) + \theta\widetilde{u}(n).$$

where $\widetilde{u}(n)$ is the input, $\widetilde{y}(n)$ is the output, θ is the decay time of the system and must satisfy $0 < \theta \leq 1$. The relation between θ and the cutoff frequency f_C of the digital filter is given by:

$$\theta = 1 - e^{-2\pi f_C}$$

One could easily derive a high-pass filter from the low-pass filter. For example, one could subtract from the original signal, the low pass-filter. Another option would be to create a band-pass filter with the equations:

$$\mathbf{x}_{LP}(t+1) = (1 - \gamma_1)\mathbf{x}_{LP}(t) + \gamma_1 f(\rho W\mathbf{x}(t) + W^{in}\mathbf{u}((t+1)\delta) + W^{fb}\mathbf{y}(t) + v(t+1)) \quad (2.10)$$
$$\mathbf{x}_{HP}(t+1) = (1 - \gamma_2)\mathbf{x}_{HP}(t) + \gamma_2\mathbf{x}_{LP}(t+1) \quad (2.11)$$
$$\mathbf{x}(t+1) = \mathbf{x}_{LP}(t+1) - \mathbf{x}_{HP}(t+1) \quad (2.12)$$

where γ_1 determines the cutoff frequency of the low-pass filter and γ_2 determines the cutoff frequency for the high-pass filter. The echo state property is thus fulfilled if (because the maximum band-pass response cannot exceed the maximum response of the high-pass or low-pass filter):

$$0 \leq \rho < 1, \qquad 0 < \gamma_1 \leq 1, \qquad 0 \leq \gamma_2 < 1 \tag{2.13}$$

Actually the band-pass response will usually be lower than the low-pass response, thus the gain output signal can be re-normalized by a gain M to leave the echo state property unchanged:

$$M = 1 + \frac{\gamma_2}{\gamma_1} \tag{2.14}$$

We show in Figure 2.6 a general structure of such a band-pass neuron and in Figure 2.7 a set of

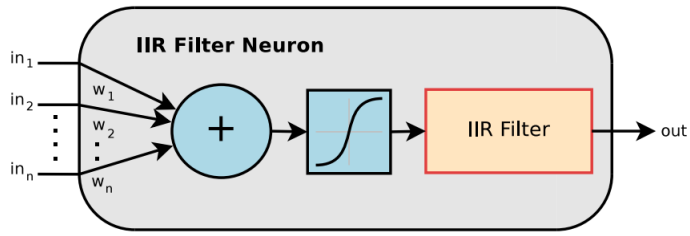


Figure 3.1: Structure of an analog neuron with an additional IIR filter.

Figure 2.6: Analog neuron with an additional IIR filter. Figure taken from [48].

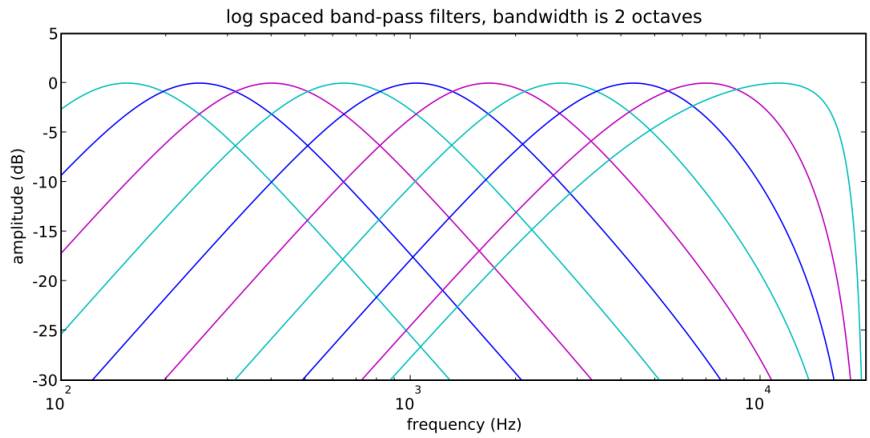


Figure 2.7: Band-pass filters from 170 Hz to 19000 Hz logarithmically, at a sampling rate of 44100 Hz (frequency axis is also logarithmic), printed every 10th neuron of a reservoir with 100 neurons. Each filter has a bandwidth of 2 octaves. Figure taken from [48].

example neuron responses. We can expect from this kind of neuron responses a lot of diversity in the reservoir, which is a most desirable effect. The reader is referred to [48] for an impressive list of results. However the computational resources needed for such an approach are not low. We will continue next with our own approach to some of the time-series prediction tasks used in the literature.

Chapter 3

Improving the ESN on the MSO problem

3.1 Introduction

A particular hard to solve problem with this kind of networks is predicting a convolution of sine signals, called the multiple superimposed oscillation (MSO) problem. The sinus signals are chosen to have small differences in their periodicity. It was hypothesized that this kind of prediction is almost impossible for the echo state network to perform successfuly. However, in a very recent paper [62], the authors make a very simple analysis of the inner dynamics of the network which unravels at least to a sufficient degree the requirements for a good predictor echo state network in the MSO problem. They achieve a performance (measured by normalized root mean squared error or NRMSE) with several orders of magnitude lower than current best results found in the literature. The paper mentioned is the first of its kind, considering a very simple aspect of building an ESN.

3.2 Linear ESN

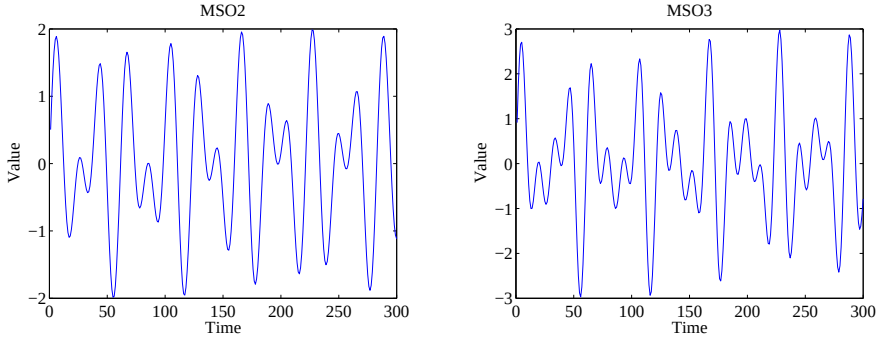
As reported by a few papers on ESNs [85, 107], for some time-series, the dynamics of the ESN seems to catch best the desired function when the weights are scaled such that the network functions in a sub-linear regime. Considering this and also the significant computational overhead of the non-linear transfer function we tried to apply the ESN with linear neurons to the MSO problem. Thus the equation of advancing the network to the next time step becomes:

$$\mathbf{x}(\mathbf{t}+\mathbf{1}) = W \cdot \mathbf{x}(\mathbf{t}) + W^{fb} \cdot \mathbf{y}(t) \tag{3.1}$$

The same for the testing phase. Everything else remains unchanged. We had the same problem when searching for a good network, as in many paper on ESNs, not finding a significant measure of the goodness of fit for specific parameters of the network, like connectivity, size, spectral radius or weight scaling, the variation being too big to take into account anything else other than the minimum of a series of repetitions. The MSO problem is as usual:

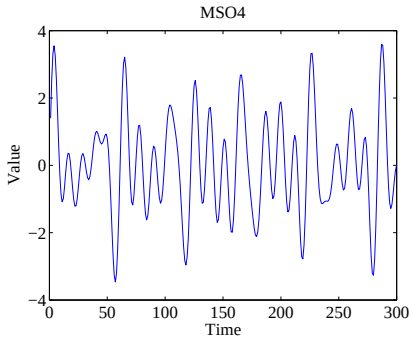
$$S = \sum_{i=1}^{N} sin(\alpha_i i) \tag{3.2}$$

With the α_i being equal to: $\alpha_1 = 0.2, \alpha_2 = 0.311, \alpha_3 = 0.42, \alpha_4 = 0.51, \alpha_5 = 0.63, \alpha_6 = 0.74, \alpha_7 = 0.85, \alpha_8 = 0.97,$. We show in Figure 3.1 the signals of the MSO problem.

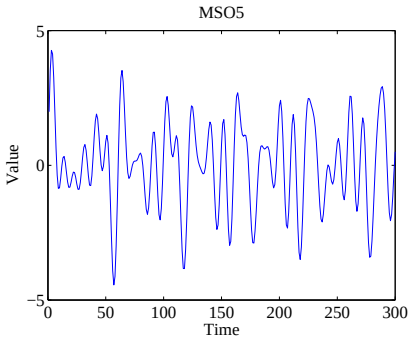


(a) MSO2

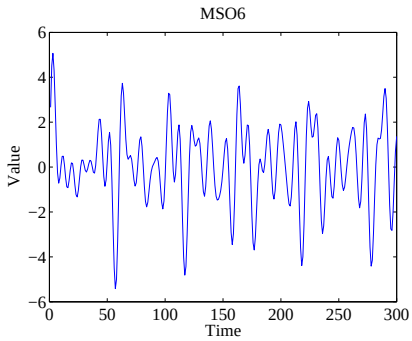
(b) MSO3



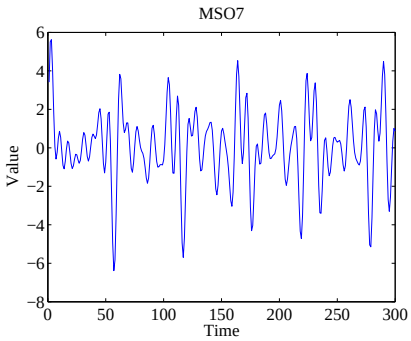
(c) MSO4



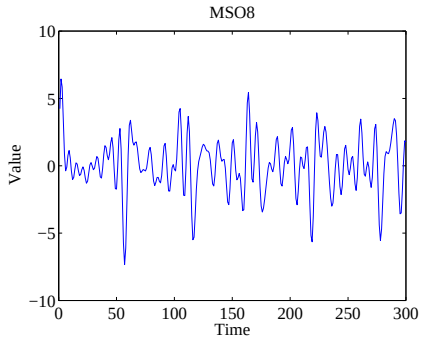
(d) MSO5



(e) MSO6



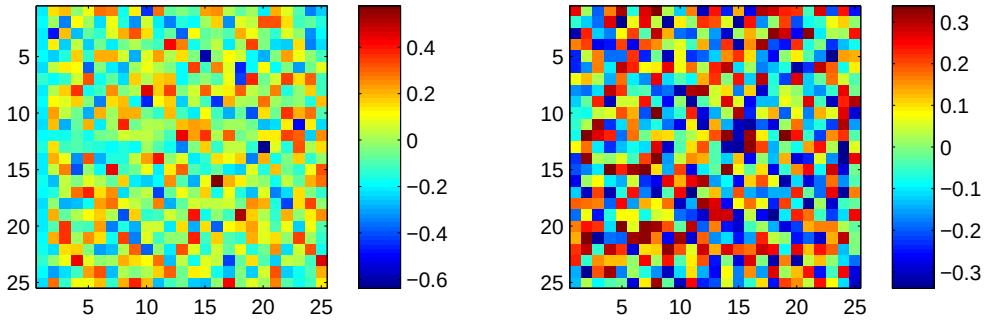
(f) MSO7



(g) MSO8

Figure 3.1: The MSO problems.

3.3 Using Orthonormal Matrices



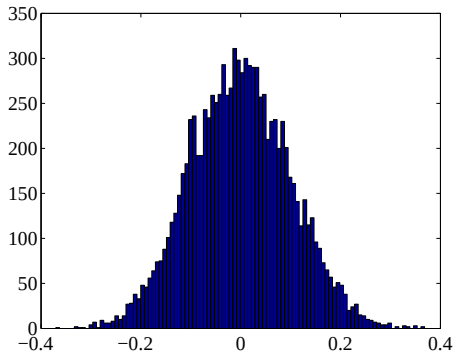
(a) Orthonormal matrix.

(b) Random scaled matrix, spectral radius 0.9

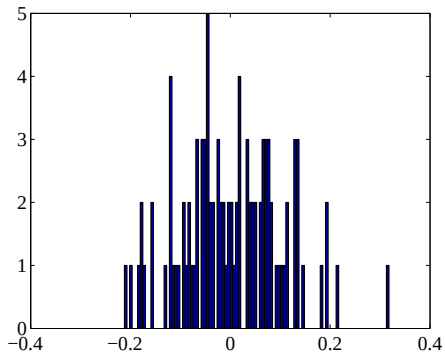
Figure 3.2: Color mapping of a random scaled and orthonormal matrix.

Performing extensive experiments with echo state networks, we had an intuition that the weights of the network function as a kind of dynamical basis for our original signal. Decomposing the signal into pertinent sub-components, by using a basis which maximizes the difference between them, like for example an orthonormal basis as the weight matrix, might give better results than just a simple random weight matrix. And it turned out it is indeed so. We will see in the next sections how this approach has a profound theoretical basis in the field known as Compressed Sensing (we discovered this after we got our initial results). When using orthonormal weight matrices (and a linear activation function, as we do), we don't need to set the spectral radius, we don't need to scale the matrix weights, or even the input weights (in most cases). We just set the input vector to ones (we did this for simplicity, however any random input vector can be used to give almost the same results) and get an orthonormal matrix out of a random weight matrix with weights from an uniform distribution between 0 and 1. For this we used the *orth* function from the Matlab software package. We show in Figure 3.2 how the values are distributed in the orthonormal matrix compared to the random scaled matrix. The results obtained on the MSO problem are with a huge factor better than previous results in the literature as you can see in Table 3.1. Table 3.2 shows the size range used for which we obtained the respective results of each of the MSO problems. We show in Figure 3.3 the histograms of weights in an orthonormal matrix (on a matrix column and on the whole weight matrix, network size = 100). We show also the eigenvalues of the two matrices in Figure 3.4. Interesting to note is the nature of the histogram for each column, which is skewed-Gaussian, the range of values for each column, which varies from column to column, and also the Gaussian distribution of the whole matrix. We used two functions from Matlab for regression. One is the usual pseudoinverse (*pinv*), and the other is multivariate regression (*mvregress* which we used for a one dimensional signal, but it gives significantly better results than *pinv*).

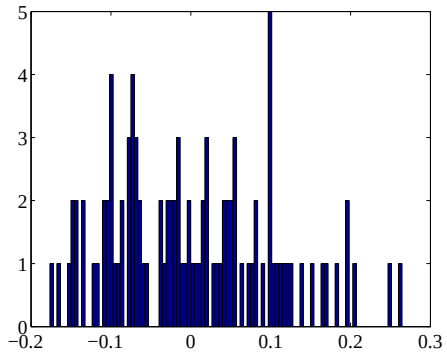
The results shown in the Table 3.1 show the best NRMSE achieved. We will describe shortly the other methods shown in the table. Balanced [62] is a simple but fundamental approach, where the two components present in the transfer function (one is the main input and the other is the input from all the other neurons) are balanced in such a way (the contribution of each component should be equal) that the dynamics of the MSO is caught with high precision, and the errors reached are smaller than any other previous method. However they test this method just on the MSO problem. Evolutionary [86] is a method which employs evolutionary algorithms to optimize the topology of the



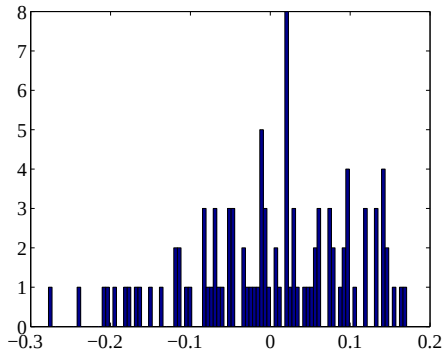
(a) Histogram for the values of the entire matrix



(b) Histogram for the values of one column



(c) Histogram for the values of one column



(d) Histogram for the values of one column

Figure 3.3: Histogram of values for an orthonormal matrix as the weight matrix.

MSO	Using *mvregress**	Using *pinv**	Balanced	Evolutionary	Evolino	IIR ESN
2	$4.13\text{x}10^{-14}$	$3.71\text{x}10^{-14}$	$2.51x10^{-12}$	$3.92x10^{-8}$	$4.15x10^{-3}$	$3x10^{-9}$
3	$7.29\text{x}10^{-13}$	$1.21\text{x}10^{-12}$	$4.57x10^{-10}$	-	$8.04x10^{-3}$	$3x10^{-7}$
4	$3.75\text{x}10^{-12}$	$2.58\text{x}10^{-11}$	$5.72x10^{-8}$	-	0.110	10^{-5}
5	$1.33\text{x}10^{-11}$	$4.29\text{x}10^{-10}$	$1.06x10^{-6}$	$2.54x10^{-2}$	0.166	$8x10^{-5}$
6	$6.89\text{x}10^{-11}$	$4.82\text{x}10^{-10}$	$8.43x10^{-5}$	-	-	-
7	$6.07\text{x}10^{-11}$	$2.47\text{x}10^{-9}$	$1.01x10^{-4}$	-	-	-
8	$8.15\text{x}10^{-11}$	$2.61\text{x}10^{-9}$	$2.73x10^{-4}$	$4.96x10^{-3}$	-	-

Table 3.1: Compared results of the MSOx problem, where x=2,...,8. The other results are taken from [62]. Results with * are our results that make use of a linear activation function and an orthonormal weight matrix.

MSO	Size range
2	2-20
3	10-30
4	20-40
5	20-40
6	40-60
7	50-70
8	90-100

Table 3.2: Settings of the ESN for the MSO experiments. 20 repetitions were performed for each value of the size and connectivity.

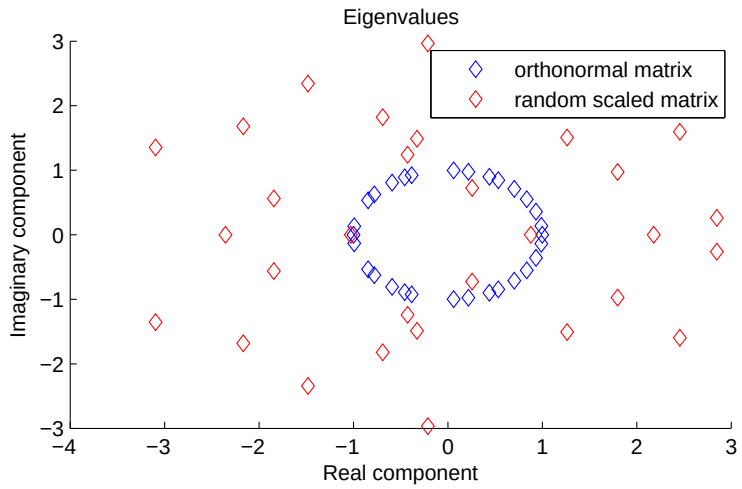


Figure 3.4: Eigenvalues for an orthonormal matrix and a random scaled matrix.

network, its size, and also the synaptic connections in the inner reservoir. The paper also addresses the trade-off between maximum reservoir accuracy and minimum reservoir complexity. Evolino [89] is using an evolutionary algorithm to modify just the internal weights while computing optimal linear mappings to the output. Evolino is employed also for training Long Short-Term Memory networks (LSTM) [47]. IIR ESN [49] uses band-pass filters to have unique distributions for every neuron, it is the same method described in Section 2.6.2.

We wanted to see what is the difference between an orthonormal weight matrix and a random scaled

C	p-value	Confidence Interval (CI)
10^{-5}	0.0506	$[-0.1995; 0.0003]$
10^{-6}	0.0713	$[-0.2106; 0.0088]$
10^{-7}	0.1944	$[-0.1976; 0.0404]$
10^{-8}	0.5965	$[-0.1372; 0.0791]$
10^{-9}	0.0453	$[0.0011; 0.1065]$

Table 3.3: Results of the unpaired two-sample t-tests comparing results with the orthonormal matrix and with the random scaled weight matrix. The t-tests are comparing the probabilities of getting an error smaller than C.

weight matrix. We performed a series of 16 experiments for every value of the connectivity parameter (which goes from 0 to 1 in increments of 0.01) and then calculated the probability of a network to give a certain error as the ratio between the number of times the error was less than a certain threshold (e.g. 10^{-10}) and the repetitions performed. In Figures 3.5 and 3.6 we show the probability of reaching an error smaller than C with $C = 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}$ as a function of connectivity (we fixed the size to 30 neurons) for an orthonormal weight matrix and a scaled weight matrix respectively. We then performed a two-sample unpaired t-test for checking if indeed the orthonormal matrix performs better. We selected as the input for t-test the probabilities for $C = 10^{-9}$ for an orthonormal matrix (first sample) and for the random scaled matrix with spectral radius set to 0.9 (for the second sample). So the null hypothesis is that the two samples come from distributions with equal means. We show the results in Table 3.3. The confidence interval is a $100 * (1 - \alpha)$ for the true difference of population means ($\alpha = 0.05$). We see that the only C for which the null hypothesis can be rejected at 5% significance level is $C = 10^{-9}$. So the performance difference between an orthonormal matrix and a random scaled matrix with spectral radius = 0.9 is not significant for other values of C, but it is for the smallest value.

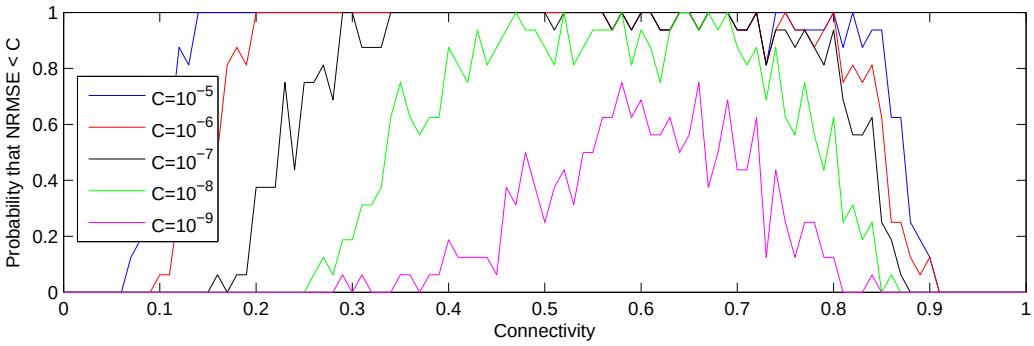


Figure 3.5: Probability of finding a network with a $NRMSE < C$ when using an orthonormal weight matrix for the MSO5 problem as a function of connectivity and size.

The minimum NRMSE achieved when using a scaled weight matrix is 2.5×10^{-11}, when using an orthonormal weight matrix the minimum NRMSE is 1.02×10^{-11} and when using a Gaussian distribution for the weight matrix, with mean 0 and 0.2 standard deviation, then the minimum error reached is also small: 1.83x10^{-11}. It is also interesting to see (Figure 3.7) how the solution space changes shape with size and connectivity compared to the orthonormal matrix. We see that the connectivity is critical for finding a good echo state network which accurately predicts the future signal. In addi-

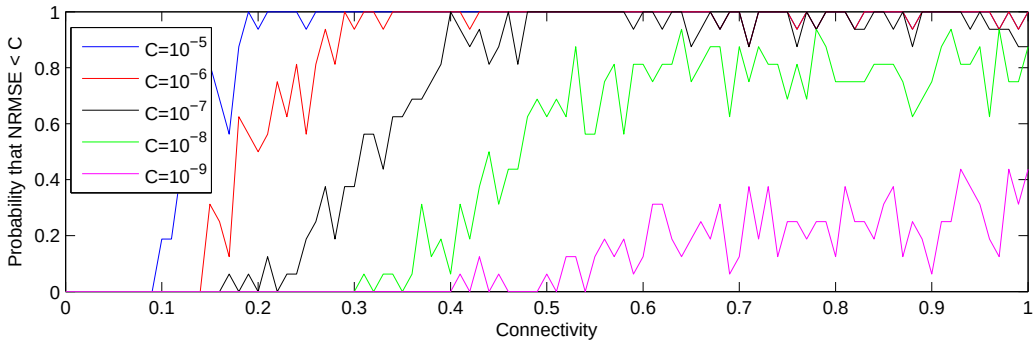
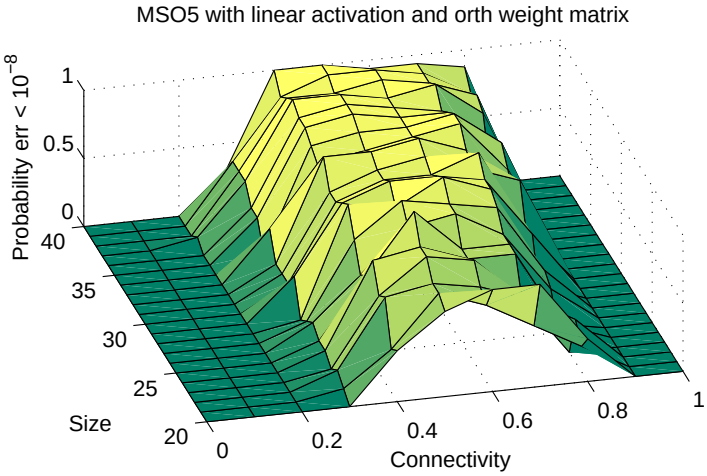


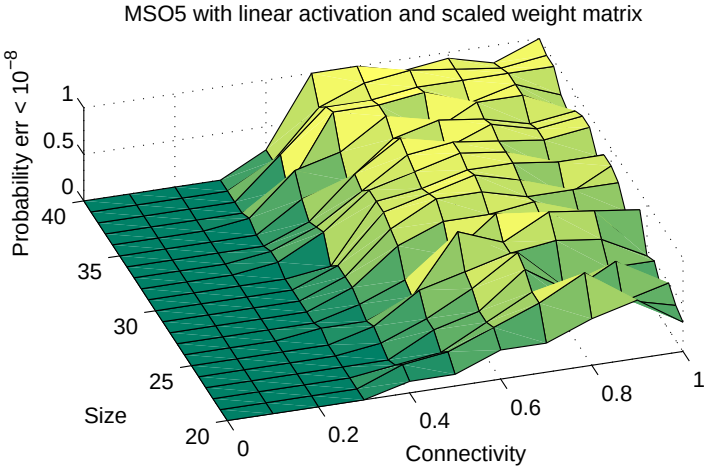
Figure 3.6: Probability of finding a network with a $NRMSE < C$ when using a scaled weight matrix for the MSO5 problem as a function of connectivity.

tion we employed a very simple and computational inexpensive way of regularizing the states of the network, in which connectivity does not matter anymore. For each time step (training and testing) we divide each value of the neuron by its index number in the network, such that every neuron should decay with a factor of $index^{time-step}$. As such, every neuron has a different exponential decaying factor, according to its index (e.g. neuron number 20 is decaying with 20^{400} after 400 training steps). This enables the network to be driven much more, and in a regular fashion, by the input signal. In Figures 3.8 and 3.9 we show examples of a neuron ($index = 20$) when using this method and when not using it respectively, for a network of size 20 which is fully connected (the signal used as input is the MSO2). Note the different scales on the y-axis in the two conditions. Even though the network was much more stable when using this regularization method, the minimum error achieved was worse with a factor of 10^{-1} to 10^{-2}. So even though we have a good network almost everywhere in the solution space defined by size and connectivity, this comes at the cost: we cannot reach a NRMSE as small as when not using regularization. Depending on the task at hand, one behavior might be preferred over the other.

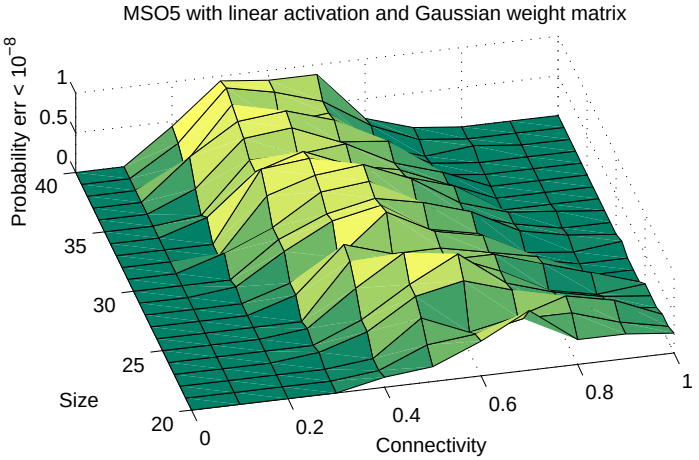
In Figure 3.10 we show the difference in the mean NRMSE between the two conditions: with regularization and without regularization as a function of connectivity and network size. We observe that without using regularization the scale of the plot is with a huge factor bigger than when using regularization. We even get a mean of over 10^{50} for one configuration of the parameters (the yellow peak in 3.10(b)), while regularization reduces this unstable behavior almost completely (we see that the biggest mean NRMSE in 3.10(a) is 1) and makes the space defined by connectivity and size seem almost convex. More experiments need to be performed to investigate if it really is convex. We leave this for future research, as well as finding a similar, but better regularization method, which does not have any precision cost as we observed when using this method. However, depending on the task, one might prefer consistent behavior than high precision, so this simple regularization method enables minimum variability of the ESN at a quite small precision cost compared to the overall magnitude of the error.



(a) Using a linear activation and an orthonormal weight matrix.

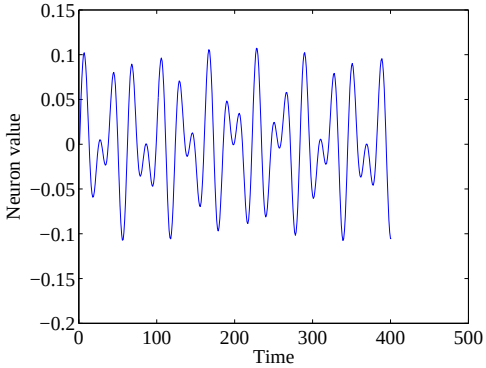


(b) Using a linear activation and a scaled weight matrix.

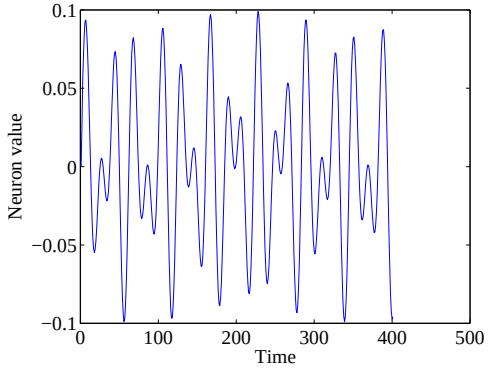


(c) Using a linear activation and a weight matrix drawn from a Gaussian distribution with mean 0 and standard deviation 0.2.

Figure 3.7: Probability that the error obtained is less than 10^{-8} using three types of matrices for the MSO5 problem as a function of size and connectivity.

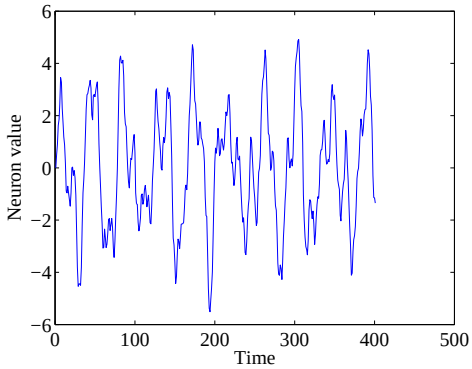


(a) Example neuron signal when using regularization

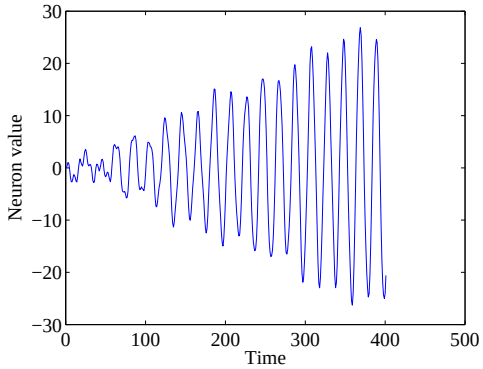


(b) Example neuron signal when using regularization

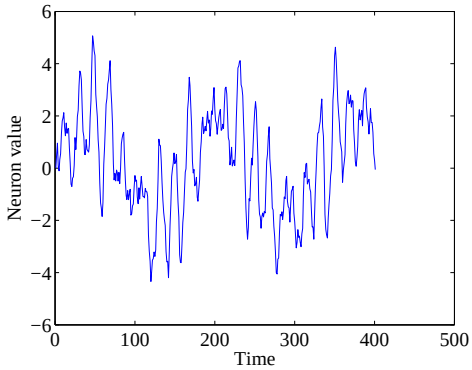
Figure 3.8: Examples of the signal in neuron with index 20 with regularization for an orthonormal weight matrix and a linear activation function.



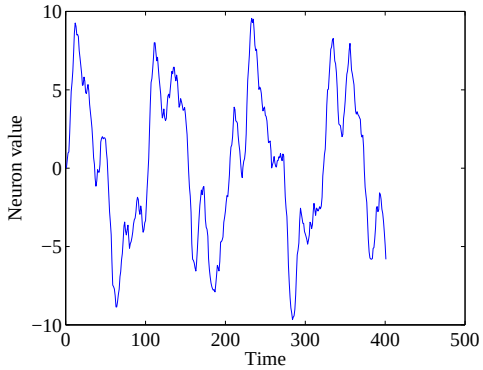
(a) Example neuron signal without using regularization



(b) Example neuron signal without using regularization

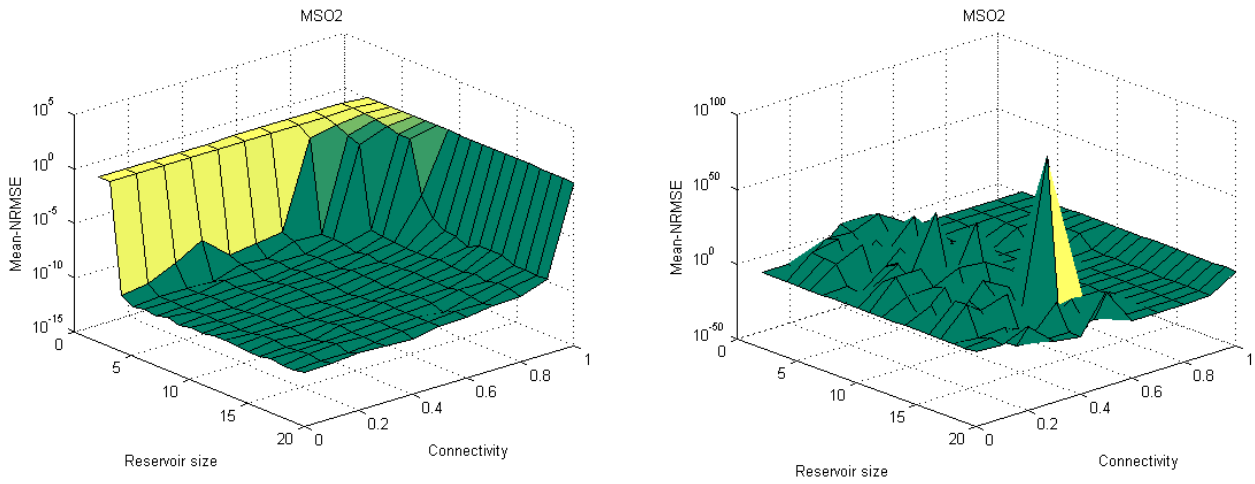


(c) Example neuron signal without using regularization



(d) Example neuron signal without using regularization

Figure 3.9: Examples of the signal in neuron with index 20 with no regularization for an orthonormal weight matrix and a linear activation function.



(a) With regularization

(b) Without regularization

Figure 3.10: Mean NRMSE.

3.4 Related Work: Compressed Sensing (Compressive Sampling)

Compressed sensing (CS) is a relatively new field in the area of signal processing and applied mathematics [18, 21, 34, 20, 101] which builds on the idea that a high dimensional signal can be represented in a lower dimensional space with some transformation. The well known Nyquist frequency can be surpassed (and even to a high degree) for some signals for which a sparse representation exists in some basis. A sparse representation means that for a signal with length n, compression by a signal of length $k << n$, where k are nonzero coefficients, whilst the other coefficients are 0 can recover the signal to a sufficiently accurate degree. Nonlinear optimization techniques can then be employed to accurately recover the original signal. It is easy to argue why fewer measurements are preferred to be able to recover the correct signal of interest: sometimes, more samples might not be available, or the computational overhead of acquiring them would be too big or simply because a small sample of measurements is sufficient for acquiring the correct signal. So instead of acquiring the signal at a high sampling rate and then compressing it to a lower dimensional signal, CS is concerned with directly acquiring the signal through some transformation of the original signal which requires much less samples. Thus it can be considered as a new framework for signal acquisition and sensor design. Of course the theory is much more complex and we won't give here a complete description, we just outline the basic principles and show the connection with the ESN paradigm. First, finding a sparse representation of a signal involves a sensing matrix (this is the name given in the CS literature) which has some desirable properties when dealing with a certain signal. There are some matrices which are generally good for this task independent of the signal structure [3, 36], however, in general one such matrix must be constructed to fit some specific properties of the signal acquired. We present below some fundamental mathematical principles of the CS framework.

3.4.1 Normed Vector Spaces

We talked in section 2.4.1 about p-norms. P-norms are a way of assigning a length to some vector which is real-valued and resides in $\mathbb{R}^n$.

$$\|\mathbf{x}\|_p = \begin{cases} (\sum_{i=1}^{n} |x_i|^p)^{1/p}, & p \in [1, \infty) \\ max_{1,2,\ldots,n} |x_i|, & p = \infty \end{cases} \tag{3.3}$$

$\langle \mathbf{x}, \mathbf{z} \rangle = \mathbf{z}^T \mathbf{x} = \sum_{i=1}^{n} x_i z_i$

The inner product helps defining the l_2 norm such that: $\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$. Different norms have different properties according to the choice of p. We show this in Figure 3.11 of unit-spheres induced by each of these norms.

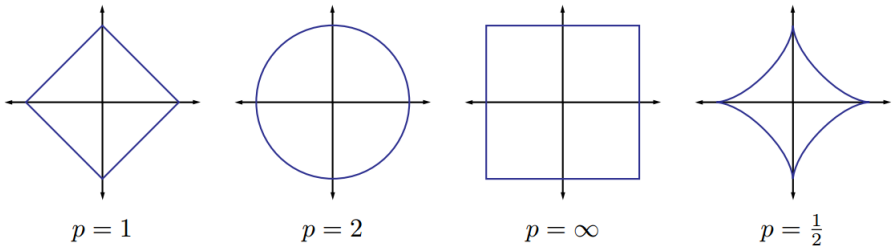


Figure 3.11: Unit spheres in $\mathbb{R}^2$ for the l_p norms. [Figure taken from [30]]

Norms are usually used to measure the strength of a signal or to measure some magnitude of deviation, or error from another signal. For example, suppose we have a signal $\mathbf{x} \in \mathbb{R}^2$ and we want to approximate it using a point residing in a one-dimensional affine space A. Then we want to measure the error achieved, and thus our task becomes to find a $\widehat{\mathbf{x}} \in A$ such that $\|\mathbf{x} - \widehat{\mathbf{x}}\|_p$ is minimal. In this case, our choice of the p-norm will have a significant outcome on the approximation. So to compute the closest point in A to $\mathbf{x}$ using each of the aforementioned norms, we can imagine growing a l_p sphere with the center in $\mathbf{x}$ until it intersects with A. This will be the point $\widehat{\mathbf{x}} \in A$ which is closest to our initial $\mathbf{x}$ according to the specific l_p norm. We show in Figure 3.12 the nature of the approximation for each of these norms.

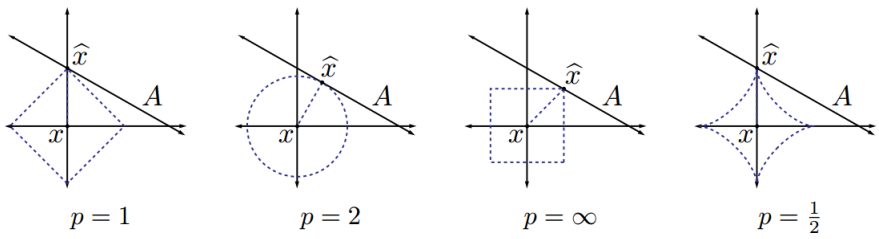


Figure 3.12: Best approximation of a point in $\mathbb{R}^2$ by a one-dimensional subspace for the l_p norms. Figure taken from [30].

We can observe that larger p spreads out the error more evenly among the two coefficients, while smaller p gives an error which is more unevenly distributed, and tends to be sparse. This example shows the behavior of the l_p norms in $\mathbb{R}^2$ but the same principles apply also for higher dimensions. This aspect of l_p norms plays a very important role in developments in compressed sensing.

3.4.2 Bases and Frames

To further understand the bigger mathematical framework of which CS is part we need to define some basic mathematical concepts. A set $\{\phi_i\}_{i=1}^n$ is called a basis for $\mathbb{R}^n$ if the vectors in this set span the whole space of $\mathbb{R}^n$ and are also linearly independent. Actually, in any n-dimensional vector space, a basis will consist of exactly n vectors; less than n vectors are not sufficient to span the whole space, while more are surely to be linearly dependent. So a basis, in fact, means that any vector which resides in the space spanned by this set can be represented (uniquely) by a linear combination of these basis vectors. Formally, for any $x \in \mathbb{R}^n$, there exist unique coefficients $\{c_i\}_{i=1}^n$ such that

$$\mathbf{x} = \sum_{i=1}^{n} c_i \phi_i \tag{3.4}$$

In matrix form this becomes:

$$\mathbf{x} = \mathbf{\Phi c} \tag{3.5}$$

where $\mathbf{\Phi}$ is the $n \times n$ matrix whose columns are given by ϕ_i and $\mathbf{c}$ is the vector of length n with elements c_i. An important type of basis is an *orthonormal* basis, which is defined again as a set $\{\phi_i\}_{i=1}^n$, but this time satisfying the following condition:

$$\langle \phi_i, \phi_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{3.6}$$

For an orthonormal basis the coefficients c can be computed as:

$$c_i = \langle x, \phi_i \rangle \,, \tag{3.7}$$

or in matrix form:

$$\mathbf{c} = \mathbf{\Phi}^T \mathbf{x} \tag{3.8}$$

This can be verified with the property of the orthonormality of the matrix $\mathbf{\Phi}$, meaning $\mathbf{\Phi}^T\mathbf{\Phi} = I$, where I is the $n \times n$ identity matrix. Now, considering the same properties of a set, but with the column vectors being linearly dependent, we get what is refereed to as a $frame$ [22, 24]. Formally, a frame is a set $\{\phi_i\}_{i=1}^n \in \mathbb{R}^d$, with $d < n$ which constitutes the matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times n}$ such that for all vectors $\mathbf{x} \in \mathbb{R}^d$ we have:

$$A \left\|\mathbf{x}\right\|_2^2 \leq \left\|\mathbf{\Phi}^T \mathbf{x}\right\|_2^2 \leq B \left\|\mathbf{x}\right\|_2^2 \tag{3.9}$$

with $0 < A \leq B < \infty$. When we have A as the largest possible value for this inequality to hold, and B as the smallest, we call them optimal frame bounds. The concept of frame can be defined also for infinite-dimensional spaces, but when $\mathbf{\Phi}$ is a $d \times n$ matrix, A and B correspond to the smallest and largest eigenvalues of $\mathbf{\Phi}^T\mathbf{\Phi}$. When working in the CS framework, or even sparse approximation, frames are often called *dictionaries* or *over-complete dictionaries* with their elements being called *atoms*.

3.4.3 Sparse Models

Signal processing is generally concerned with acquiring information from different types of signals or data. In order to have efficient algorithms which do this well for a specific signal or problem, it is desirable to have informative models for those particular signals. These can be generative models [45], graphical models or probabilistic Bayesian models [72]. Models are useful for using knowledge we already have about a specific signal to differentiate between interesting or probable signals from improbable signals. Usually signals are modeled as a vector lying in some vector space. However, not all vectors lying in a vector space represent valid signals. The actual problem of interest in many fields is considering the much lower number of degrees of freedom one such high dimensional signal has, compared to its ambient dimensionality. The models which take this into account are called low-dimensional signal models. Even though they are dealing with high-dimensional signals, these models try to approximate the signal with some low-dimensional model.
Real-world signals are generally well-approximated by a linear combination of base elements from a basis or dictionary. When such a representation is exact (the error is almost nonexistent) we say that such a signal is *sparse* in that basis. This type of sparse signal models capture the intuitive fact that even though a signal is high-dimensional, the information it contains is little compared to the ambient dimensionality (the actual dimension of the space in which the signals resides). Sparsity can be thought of a manifestation of Occam's razor, when having multiple ways of representing a signal choose the one with the lowest dimensionality, the simplest one. Formally, we say that a signal is $k - sparse$ when it has at most k nonzero elements, meaning that $\|x\|_0 \leq k$. We denote:

$$\sum_k = \{\mathbf{x} : \|\mathbf{x}\|_0 \leq k\} \tag{3.10}$$

as the set of all k-sparse signals. Usually these signals are not sparse in their original residing space, but admit a sparse representation in some basis $\mathbf{\Phi}$, so we can express $\mathbf{x}$ as $\mathbf{x} = \mathbf{\Phi c}$ where $\|\mathbf{c}\|_0 \leq k$. Sparsity has been useful in a lot of domains such as compression [32, 79, 102], denoising [33], statistics and learning [61] as a way of avoiding over-fitting and performing model selection [106], also in the study of human brain processing systems [13, 83] and in image processing using the wavelet transform which finds nearly sparse representations for natural images [25, 110]. An example is shown in Figure 3.13.
We describe shortly how this is achieved. The majority of natural images have large textured regions, or smooth regions, but just a few sharp edges. The multiscale wavelet transform recursively divides the image into its high and low frequency components. The lowest frequency components provide a coarse approximation of the image, while the high frequency ones fill in the remaining details by resolving edges. When computing a wavelet transform (Figure 3.13), we see that the majority of the coefficients have very low values, and thus by taking only the high valued coefficients (above a certain threshold) we can obtain a k-sparse approximation of the image, using only k basis elements. And because we know that natural images have sparse wavelet representations we can filter out some common types of noise which generally do not have a sparse wavelet representation.

3.4.4 Geometry of Sparse Signals

Sparse models are highly non-linear models because the choice of the dictionary used changes for each specific signal [87]. For example, given two k-sparse signals, a linear combination of the two signals will usually not be k-sparse anymore, even though it will be $2k$-sparse. We illustrate this in Figure 3.14, which shows $\sum_2$ embedded in $\mathbb{R}^3$, that is the set of all 2-sparse signals residing in $\mathbb{R}^3$.

(a) (b)

Figure 3.13: Sparse approximation of a natural image (a) Original image. (b) Approximation of the image obtained by keeping the largest 10% of the wavelet coefficients.

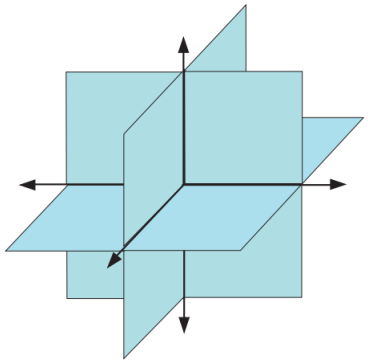


Figure 3.14: Union of subspaces defined by $\sum_2 \in \mathbb{R}^3$, the set of all 2-sparse signals in $\mathbb{R}^3$. Figure taken from [30].

3.4.5 Sensing Matrices and Incoherence

We have explained what a sparse representation of a signal is, but we shall now see how this is integrated into the framework of CS. So the basic idea is to be able to get this sparse representation with minimum computational resources. To this end we introduce the concept of random projection (RP). A random projection is a technique used in approximation algorithms by allowing the problem dimensionality to be reduced, while still retaining the problem structure. Formally, given n points in $\mathbb{R}^n$ we can project these points to a random d-dimensional subspace with $d << n$ such that [9]:
1. if $d = O(1/\gamma^2 \log n)$ then with the help of Johnson-Lindenstrauss lemma [28](which is an excellent result in the fields of compressed sensing, manifold learning, dimensionality reduction and graph embedding; we will give its exact description later) we can show that with high probability the (relative) distances and angles between points are preserved up to $1 \pm \gamma$
2. if $d = 1$, meaning we project just to a random line, we can generally still get something useful. As we can see this is a very important result, which is a direct consequence of the Johnson-Lindenstrauss lemma, which we give next.

Johnson-Lindenstrauss lemma:
Given $0 < \epsilon < 1$, a set X of m points in $\mathbb{R}^n$ and a number $n > 8 \ln m/\epsilon^2$, there is a Lipschitz function

$f : \mathbb{R}^n \to \mathbb{R}^n$ such that:

$$(1-\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \leq \left\|f(\mathbf{u})-f(\mathbf{v})\right\|^2 \leq (1+\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \textit{for all} \quad \mathbf{u},\mathbf{v} \in X. \tag{3.11}$$

We will not show the proof here, as this is out of the scope of this thesis, however we mention that one version of the proof involves taking f as a suitable multiple of the orthogonal projection on a random subspace of dimension n in $\mathbb{R}^n$ and exploits the phenomenon of concentration of measure. Because this thesis is concerned with machine learning as its primary goal, we need to mention here that a random projection (of some specific type) can be used for giving an interesting insight into kernel functions and also can help convert a kernel function into an explicit feature space [9, 82]. We show now how all of the above fits into the CS framework. We take $\mathbf{u}$ to be an N-dimensional signal that we wish to measure. So $\mathbf{u}$ is a vector with components u_i, with $i = 1, ..., N$ where each u_i can take a real value. We now take M measurements of the form $x_\mu = \mathbf{b}^\mu \cdot \mathbf{u}$ for $\mu = 1, ..., M$. We can consider each x_μ as an outcome of the measurement μ obtained by computing the dot product between the unknown signal $\mathbf{u}$ and the measurement vector $\mathbf{b}^\mu$. In matrix form this becomes $\mathbf{x} = \mathbf{B}\mathbf{u}$. Here $\mathbf{B}$ is called the measurement matrix (MxN), and each μth row is the vector $\mathbf{b}^\mu$ and $\mathbf{x}$ is a measurement vector whose μth component is x_μ. Let's say that the true signal $\mathbf{u}$ is sparse in some basis given by the columns of a NxN matrix $\mathbf{C}$. This means $\mathbf{u} = \mathbf{C}\mathbf{s}$, where $\mathbf{s}$ is a sparse N-dimensional vector, meaning that it has a small number K of nonzero elements, even though we don't know which K out of N elements are nonzero. For a wavelet transform as mentioned above, the K elements would be those coefficients we choose above the specified threshold. Finally, the relation between the sparse coefficients and the measurements is given by: $\mathbf{x} = \mathbf{A}\mathbf{s}$, where $\mathbf{A} = \mathbf{B}\mathbf{C}$. In CS literature $\mathbf{A}$ is called the *sensing matrix*. Now, an important problem is, given $\mathbf{C}$ (the sparsity basis) how should we choose our measurement basis $\mathbf{B}$? For instance, let's say we're dealing with an image and we choose our M measurements to be the randomly chosen wavelet coefficients, so our measurement basis is the wavelet basis (with M just a little larger than K). We see now that it is highly improbable that out of the M measurements we will take the whole set of K relevant coefficients (above the desired threshold). So to be sure we get the relevant data we need to make M as large as possible, approaching N the dimensionality of the signal. So we can easily conclude that a right choice of measurement basis is very important for the successful reconstruction. From the example above we see that the measurement basis should be very different than the sparsity basis, but how much different, and in what way ? For starters, our measurement basis should have many nonzero elements in the domain in which the signal is sparse. This notion is captured in the mathematical definition of *incoherence*. This entails a low value for the maximum inner product between rows of $\mathbf{B}$ and columns of $\mathbf{C}$, such that no measurement vector is similar to any sparsity vector. CS guarantees good recovery of the unknown signal, with M just a little bigger than K as long as the measurement vectors are sufficiently incoherent with respect to the sparsity domain [15]. We give next the formal definition of *mutual coherence*, which when this value is low, the two signals compared are called incoherent. The definition is with respect to a single matrix, however the same principles apply to rows and columns of different matrices (as stated above).

Mutual coherence:

Let $a_1, ..., a_m$ be the columns of a matrix A, which are assumed to be normalized such that $a_i^T a_i = 1$. The mutual coherence of A is then defined as:

$$M = \underset{1 \leq i \neq j \leq m}{max} \left|a_i^T a_j\right|.$$

This is where the random projection comes in. It is highly unlikely that a randomly projected vector is aligned with a sparsity vector, in fact in CS there exists a result which specifies exactly this. With random measurements only

$$M > O(K \log(N/K)) \tag{3.12}$$

measurements are needed to be almost sure (with high probability) of perfect signal reconstruction [19, 17]. Very important to note here is that this limit is not dependent at all on the sparsity domain, or the nature of the signal. So random projections are generally very good for dimensionality reduction *independent* of the signal.

3.4.6 Nonlinear Recovery Algorithm

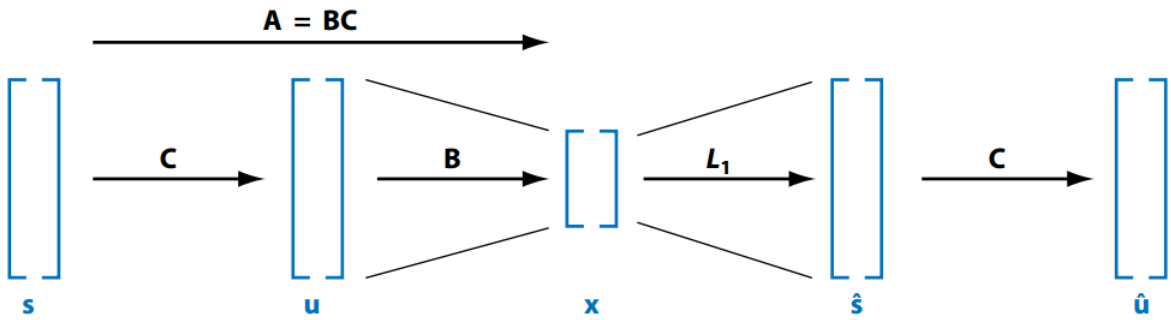


Figure 3.15: CS Framework. Figure taken from [42].

So even though we have a good set of measurements on our unknown signal, one problem still remains: how can we reconstruct the signal [42] ? We remind the reader of the equation between the measurement vector and the unknown signal: $\mathbf{x} = \mathbf{Bu}$. This is a set of M equations with N unknowns. If $M > N$ then the system is overdetermined and the problem is solved, however the very basic idea of CS is to have $M << N$, in which case the problem becomes under-determined, meaning we have a set of M equations with N unknowns. However there is still a way of solving this problem. We know that $\mathbf{u} = \mathbf{Cs}$ and $\mathbf{s}$ is sparse. Then we can write $\mathbf{x} = \mathbf{BCs}$, where $\mathbf{BC} = \mathbf{A}$. So the equation of the reconstruction becomes $\mathbf{x} = \mathbf{As}$, where $\mathbf{s}$ is sparse. Now we are facing a much simpler problem, finding a sparse vector in a set of candidate vectors. Formally, we have to estimate $\mathbf{s}$ by solving the following optimization problem:

$$(\widehat{s}) = \underset{s}{argmin} \sum_{i=1}^{n} V(s_i) \text{ subject to } \mathbf{x} = \mathbf{As} \tag{3.13}$$

where $V(s)$ is a cost function which penalizes non-zero values of $\mathbf{s}$. One possible choice is the function which is called in CS literature the $zero - norm$ and is actually $V(s) = 0$ if $s = 0$ and $V(s) = 1$ if $s \neq 0$. However this problem is a combinatorial optimization problem, computationally very expensive, which is intractable for many signals. The main choice in CS is the function $V(s) = |s|$. This quantity, $\sum_{i=1}^{N} s_i$ is called the L_1 norm of s, and as such, this method is called L_1 minimization. The L_1 norm is a convex function, and the function has just one local minimum which is also global. Efficient methods exist for finding this minimum through linear programming [12] or message passing [35]. So CS guarantees to recover the true signal, with a choice of $\mathbf{A}$ with the properties mentioned above, and with a number of measurements M which is proportional to the number of nonzero elements (K) in the sparse representation of the signal ($\mathbf{s}$), which can be much smaller than the dimensionality of

the original unknown signal ($\mathbf{u}$). We show in Figure 3.15 a schematic which describes the process just explained.

3.4.7 Dynamical CS Matrices

The framework of Compressed Sensing and the Reservoir Computing paradigm, which the Echo State Network is part of, are relatively recent developments in their respective fields. However, very recent developments exist in which these two sub-fields are connected through some very interesting methods. The two major groups contributing to this new connective field are Ganguli and Sompolinsky [42, 41, 113, 40] and Yap, Charles and Rozell [116]. Actually, the author found just the above cited literature regarding the connection between CS and ESN. So we consider this as a very new and exciting field with tremendous potential. First of all, to be able to do this, our transfer function needs to be the identity function, such that the system in equation 3.1. at time $t+1$ becomes (we change the notation from 3.1. such that $\mathbf{W}^{fb}$ is denoted by $\mathbf{v}$ to better reflect the fact that it is a vector, not a matrix):

$$\mathbf{x}(t+1) = \mathbf{W}^0\mathbf{v}\mathbf{y}(t+1) + \mathbf{W}^1\mathbf{v}\mathbf{y}(t) + ...\mathbf{W}^t\mathbf{v}\mathbf{y}(1) \tag{3.14}$$

$$= \left[\mathbf{v}\,|...|\,\mathbf{W}^t\mathbf{v}\right] \begin{pmatrix} \mathbf{y}(t+1) \\ . \\ . \\ . \\ \mathbf{y}(1) \end{pmatrix} \tag{3.15}$$

This is just unfolding the state equation in time (which is possible since the activation function is the identity function), and thus at each step incrementing the maximum power of $\mathbf{W}$ with one, to reach a maximum of $\mathbf{W}^t$ after t timesteps (from 2 to $t+1$, since $\mathbf{x}(1)$ is the initial condition for $\mathbf{x}$, usually a vector of zeros). We denote the first matrix with $\mathbf{A}$ and the second column vector with $\mathbf{u}$. Thus, our system at time $t+1$ becomes (we omit the $t+1$): $\mathbf{x} = \mathbf{A}\mathbf{u}$. We see that $\mathbf{u}$ is our input vector, the past $t+1$ values of it and $\mathbf{A}$ is a MxT measurement matrix. Following [116] we show the further developing of the analysis of $\mathbf{A}$, using the eigenvalue decomposition of the connectivity matrix $\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1}$, thus $\mathbf{A}$ becomes:

$$\mathbf{A} = \mathbf{U}\left[\widehat{\mathbf{v}}\,|\mathbf{D}\widehat{\mathbf{v}}|\,\mathbf{D}^2\widehat{\mathbf{v}}\,|...\,|\mathbf{D}^{t-2}\widehat{\mathbf{v}}|\,|\,\mathbf{D}^{t-1}\right] \tag{3.16}$$

where $\widehat{\mathbf{v}} = \mathbf{U}^{-1}\mathbf{v}$. Rewriting this equation, with $\mathbf{d} = diag(\mathbf{D})$ being the column vector constituted of the eigenvalues of $\mathbf{W}$, $\mathbf{V} = diag(\widehat{\mathbf{v}})$, we then get:

$$\mathbf{A} = \mathbf{U}\mathbf{V}\left[\mathbf{d}^0\,|\mathbf{d}|\,\mathbf{d}^2\,|...|\,\mathbf{d}^{t-1}\right] = \mathbf{U}\mathbf{V}\mathbf{F} \tag{3.17}$$

and as expected the exponentiation of the vector $\mathbf{d}$ is the element wise exponentiation. They denoted by $\mathbf{F}$ the matrix comprised of the concatenation of all the powers of $\mathbf{d}$. To be able to reach some CS settings, the authors made some specific assumptions on the nature of the network connectivity matrix $\mathbf{W}$, i.e. that it is a random orthonormal matrix, which, when large enough, has the eigenvalues distributed uniformly on the complex unit circle. Thus, the matrix $\mathbf{F}$ becomes a sub-sampled discrete-time Fourier transform (DTFT) matrix. They also assume that they have control over the input vector $\mathbf{v}$ (which in the ESN settings they actually do), and they take $\mathbf{v} = \mathbf{U}\mathbf{1}$, where $\mathbf{1}= [1, ...1]^T$.

This makes $\mathbf{V}$ the identity matrix. After this, we can see that $\mathbf{A}$=$\mathbf{UF}$, and because $\|\mathbf{UFs}\|_2^2 = \|\mathbf{Fs}\|_2^2$ for any $\mathbf{s}$, a Restricted Isometry Property (RIP) for the ESN can be drawn based on previous results for sub-sampled DTFT matrices [84]:

Theorem [116]:

Let $\mathbf{W}$ be an orthogonal $M \times M$ matrix with eigenvalues distributed uniformly on the unit complex circle, and $\mathbf{v}$=$\mathbf{U1}$ the length M feed-forward vector. Then, with probability at least $1 - O(N^{-1})$ for any input sequence $\mathbf{s} \in \mathbb{R}^N$ that is compressible in a basis ψ, $\mathbf{A}\psi$ satisfies RIP-(2k,δ) whenever

$$M \geq CK\delta^{-2}\mu^{-2}(\psi)\log^4(N) \tag{3.18}$$

The quantity $\mu(\psi)$ is the incoherence of the basis ψ with the sub-sampled DTFT $\mathbf{F}$, defined in this case as:

$$\mu(\psi) = \underset{n=1,..,N}{max} \; \underset{t\in[0,2\pi]}{sup} \left| \sum_{m=0}^{N-1} \psi_{m,n} \exp^{-jtm} \right|, \tag{3.19}$$

where $\psi_{m,n}$ is the (m,n)-th entry of the matrix ψ and j is as usual, when working with DTFTs, the imaginary unit ($\sqrt{-1} = j$). The satisfiability of the RIP ensures that we can recover the original signal through the next theorem [116]:

Theorem:

If a matrix $\mathbf{A}$ satisfies the RIP-(2K,δ) with $\delta < 0.4627$, and let $\mathbf{s} \in \mathbb{C}^N$ be any input vector, suppose we acquire noisy measurements $\mathbf{x}$=$\mathbf{As}$+ϵ with $\|\epsilon\|_2 < \eta$, then if $\widehat{\mathbf{s}}$ is the unique solution of:

$$\underset{\mathbf{s}}{min} \|\mathbf{s}\|_1 \text{ subject to } \|\mathbf{As} - \mathbf{x}\|_2 \leq \eta \tag{3.20}$$

Then

$$\|\mathbf{s} - \widehat{\mathbf{s}}\|_2 \leq \alpha\eta + \beta\frac{\|\mathbf{s} - \mathbf{s}_K\|_2}{\sqrt{K}} \tag{3.21}$$

where $\mathbf{s}_K$ is the best K-term approximation of $\mathbf{s}$ and α, β are constants that depend only on η. So, for an operator $\mathbf{A}$ satisfying the RIP, solving the l_1 minimization program above, guarantees an estimate $\widehat{\mathbf{s}}$, which has a distance from $\mathbf{s}$ bounded by the measurement noise level and by the distance from $\widehat{\mathbf{s}}$ of its best K-term approximation. The greatest power of the RIP, and this is a big enhancement of the usual ESN approach, is that if we get a matrix $\mathbf{A}$, which satisfies the RIP of order $2K$, distances between the projections of any $2K$-sparse signals are preserved in the measurements space, meaning $\|\mathbf{As}_1 - \mathbf{As}_2\|_2 \approx \|\mathbf{s}_1 - \mathbf{s}_2\|_2$. This stable embedding allows signal processing algorithms to work directly in the measurement space, without requiring a previous recovery step [29]. This is very useful in the ESN paradigm, because the ultimate goal of the neural network is *prediction* of the future signal, which means that we don't actually have an original signal to compare with; thus, knowing that the distance should be preserved in the measurement space, i.e. the activations of network nodes, we know also when a network is more likely to predict the correct signal or not. We show in Figure 3.16 the memory capacity obtained in [116] when using different sparsity bases, with different levels of incoherence with the sub-sampled DTFT.

3.4.8 Orthogonal Dynamical Systems

In another somehow similar approach, the authors in [41] have a very interesting theoretical insight into memory capacity. We don't give here their entire mathematical formulation, however we outline their

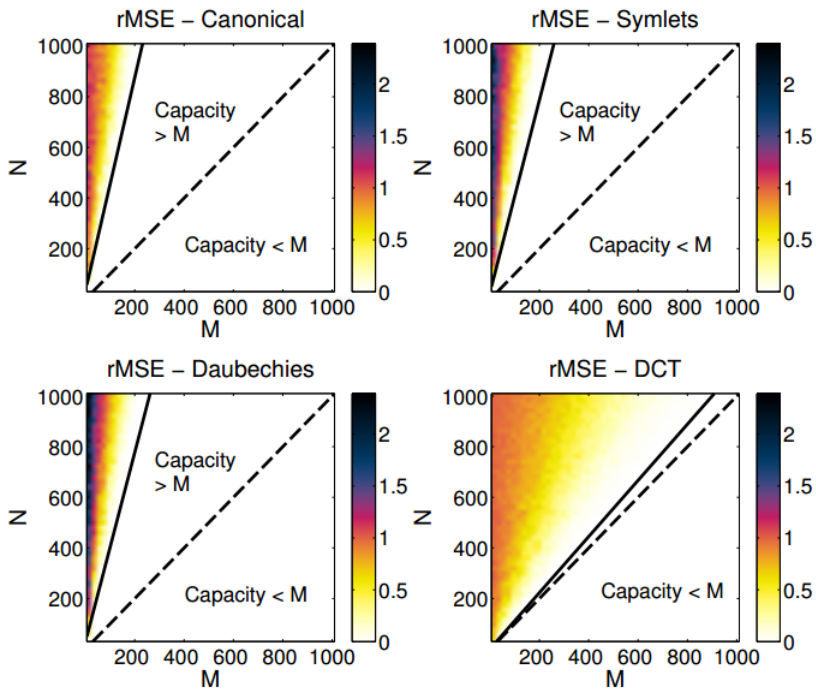


Figure 3.16: rMSE (relative MSE) for input of length N and M numbers of neurons, where the input is ρN-sparse in a basis ψ with $\rho = 0.05$. Between the dashed line ($M = N$) and the solid line (recovery error $= 0.1\%$) shows where the short term memory exceeds the number of network nodes. This is large for canonical, Symlets and Daubechies-10 wavelet basis, because the sub-sampled DFT is highly incoherent with the respective basis and is small for the discrete cosine transform (DCT) because the DCT is highly coherent with the sub-sampled DFT. Figure taken from [116].

achievements. They construct some annealed CS matrix which is a theoretical analogue to orthogonal echo state networks, in the sense that this type of matrix is a theoretical construct which does not have any correlations between its columns as the power of the matrix increases (W^k), however they are infinite in time and their columns decay with higher powers of W, two critical features of CS matrices. Using this approach they are able to derive analytically memory curves for the dynamical system under investigation and then confirm them through numerical simulations using l_1 minimization (when dealing with a sparse input signal). Then, when computing experimentally the memory capacity for the orthogonal echo state network (or orthogonal dynamical system) these curves are very similar with the previously obtained curves for the annealed system. We have to mention briefly some characteristics of the modeling used by the authors. We could say in short that σ is a parameter of the model, which if it is too small, then the measurements on the signal decay too quickly, thus preventing large memory capacity, and if σ is too large, signals from the distant past do not decay away, interfering with the measurements of more recent steps, and degrading memory. So there is an optimum σ for which the memory is maximum. As $t \longrightarrow \infty$, l_1 minimization always yields a zero signal estimate, so the memory curve asymptotically approaches the sparsity value for large t. A convenient measure of memory capacity, they say, is the time $T_{1/2}$ at which the memory curve reaches half its asymptotic error value. We show in Figure 3.17 and 3.18 their results. This is the first analytical approach to short-term memory in ESNs when dealing with a sparse signal in some basis (which most real-world signals are), and the fact that the theoretical predictions match so well with the experimental results is remarkable. Moreover they showed that the memory capacity for an orthogonal echo state network can be greater than the size of the network (as is stated in [52]) for sparse input statistics.

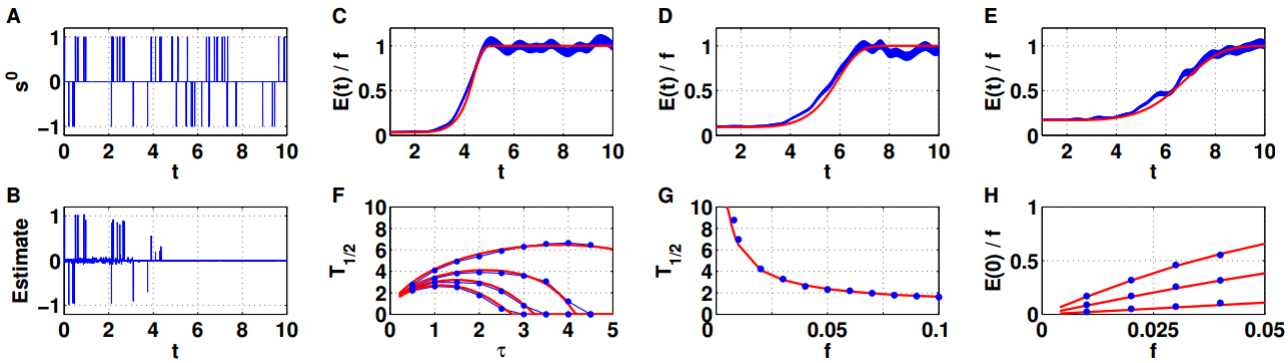


Figure 3.17: Memory in the annealed approximation (A) A plus-minus signal with sparsity 0.01 that lasts 10N time-steps, with N=500. (B) Reconstruction of the signal from the output of an annealed measurement matrix with N=500, $\tau = 1$. (C,D,E) Example memory curves for $\tau = 1$(C),2(D),3(E). In (F) we see $T_{1/2}$ as a function of τ. The curves from top to bottom are for different sparsity: 0.01,0.02,0.03,0.04. (G) $T_{1/2}$ optimized over τ for each sparsity value. (H) The initial error as a function of the sparsity; from bottom to top: $\tau = 1, 2, 3$. The red curves are theoretical predictions, while the blue curves and points are from numerical simulations using l_1 minimization with N=100, averaged over 300 trials; the width of the blue curves show the standard error [Figure from [41]].

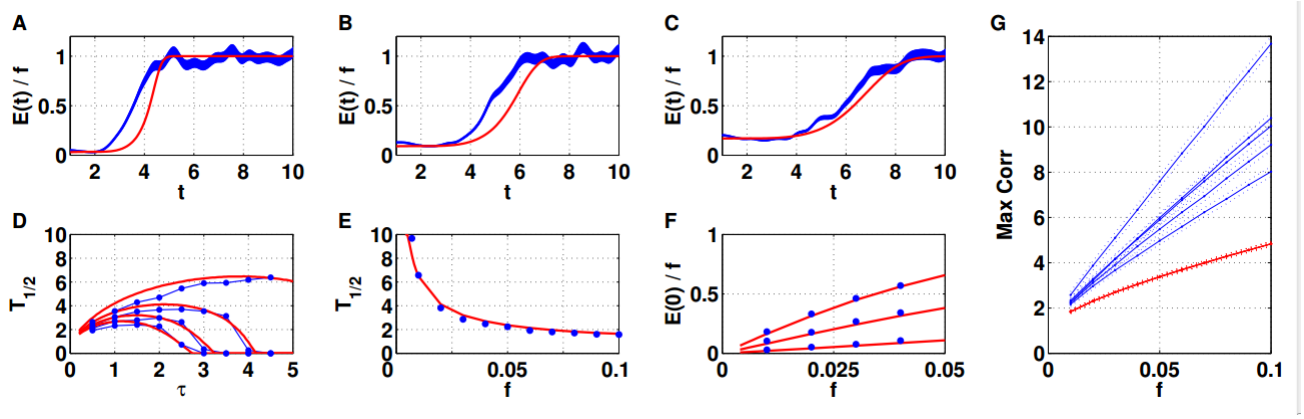


Figure 3.18: Memory in orthogonal dynamical systems. Panels A-F are identical to panes C-H in Figure 3.16, but now the blue curves are obtained when using measurement matrices derived from an orthogonal neural network. (G) The mean and standard deviation of σ_f for 5 annealed (red) and 5 orthogonal matrices (blue) with N=200 and T=3000 [Figure from [41]].

An interesting problem, related to the Restricted Isometry Property [16] is how many time-steps do we actually need to predict the next 300 time-steps of the MSO problem. We deal with the MSO2 for this and show a $4D$ plot next. It turns out we actually need very little time-steps to predict the future signal.

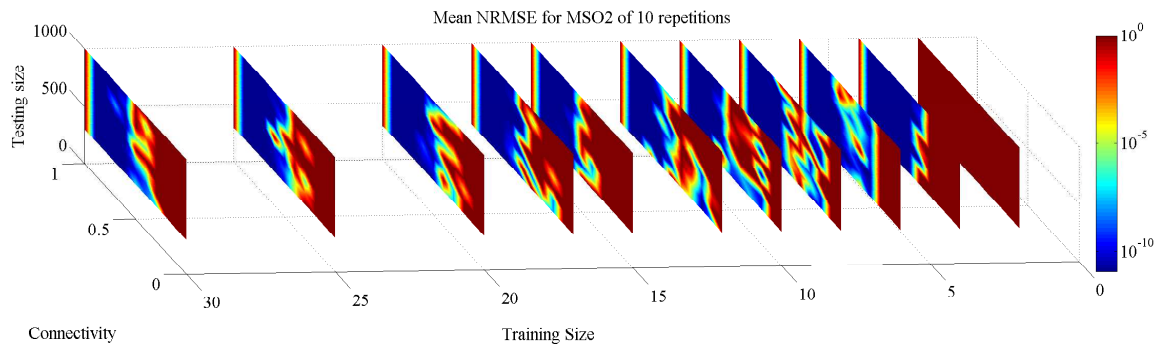


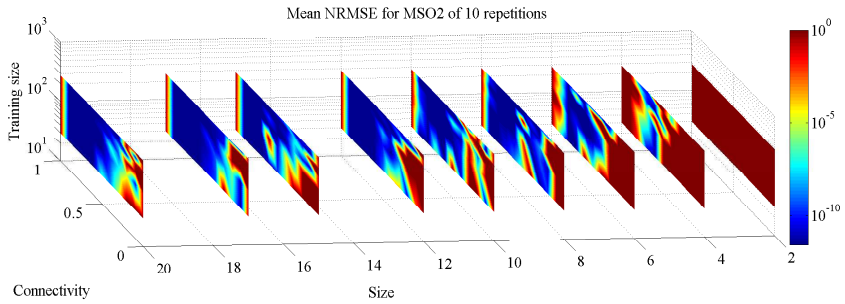
Figure 3.19: Plot showing the NRMSE as a function of connectivity, testing size and training size. Network size = 20.

We were interested in finding out how the error surface looks like when considering also the training size and the testing size, as the RIP gives some conditions on the amount of measurements one has to make. It seems that the 300 steps usually used in the literature for the MSO problem are way too many for the ESN to catch the signal dynamics. We did not find much of a difference when considering a training sequence between 30 and 300 steps. The training size axis is logarithmic. In Figure 3.19 we show the NRMSE for training size from 1 to 30 and testing size from 300 to 1000. As we can see the minimum training size is around 3-5 steps, which is impressive considering the general trend in the literature. Also the error does not seem to be affected too much when increasing the testing size. We tested until 1000 steps and the error remained small (figure 3.19 and 3.20(c)). In Figure 3.20(a) we see the error dependent on connectivity, network size and training size.

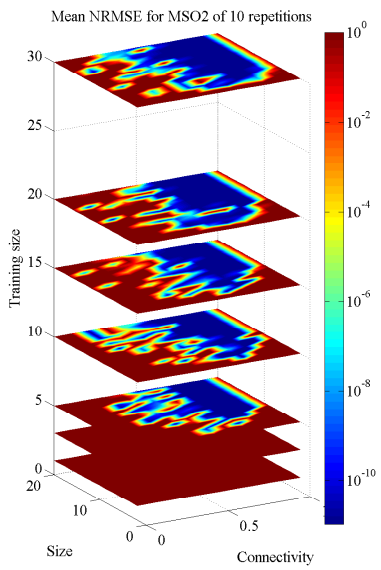
3.5 Discussion

In this chapter we tried to give a comprehensive overview of the current connections existent in the literature between CS and ESNs. Profound theoretical insights can be gained on the ESN and its inner functioning by applying results and methods from CS. As mentioned in [41] several questions remain to be answered, like for example, why do the orthogonal linear neural networks perform as good as their theoretical counterparts, even though the columns of the state matrix have correlations between them (they are not linearly independent as in theory). An interesting problem not tackled in the two approaches is the problem of prediction with such orthogonal linear networks: How many steps can be predicted ? With what precision ? How many steps do we need for training to catch the signal dynamics ? We tried to bridge the gap and showed that indeed for prediction of periodic signals, like the MSO problem, linear orthonormal (unit-norm orthogonal) echo state networks perform incredibly well compared to their non-linear counterparts. A possible future direction would be to investigate to what extent the compression-reconstruction processes and the prediction process are related. What is the relation between compression and prediction when dealing with quasiperiodic signals, or highly chaotic ones ? Or how well do these orthonormal echo state networks perform when having to predict more complex signals ? Can we still use an identity activation function ? Nothing has been said

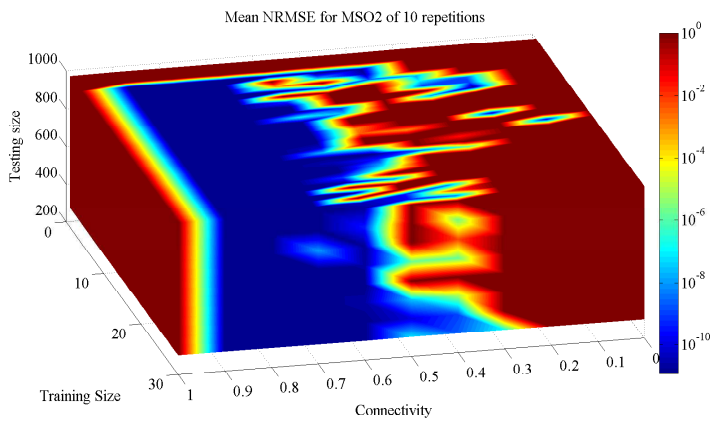
about non-linear orthogonal networks (using a *tanh* activation function for example). We will try to investigate in the next chapters various aspects of the orthonormal echo state networks when trying to predict much more complex signals, including a few real-world signals.



(a) Plot showing the NRMSE as a function of connectivity, size and training size. Training size from 30 to 300.



(b) Plot showing the NRMSE as a function of connectivity, size and training size. Training size from 1 to 30.



(c) Plot showing the NRMSE as a function of connectivity, testing size and training size. Network size = 20.

Figure 3.20: Plots showing the NRMSE dependency on connectivity, training size and testing size.

Chapter 4

Exploring the ESN

4.1 Activation Function

4.1.1 Tanh Neurons

Usually the ESN is used in conjunction with *tanh* neurons, especially for predicting chaotic time-series, like the Mackey-Glass attractor. We show below the equation for generating the time-series; the Mackey-Glass equation is a nonlinear time delay differential equation:

$$\frac{dx}{dt} = \beta \frac{x_\tau}{1 + {x_\tau}^n} - \gamma x, \quad \gamma, \beta, n > 0, \tag{4.1}$$

with $\beta = 0.2$, $\gamma = 0.1$,$n = 10$ and with x_τ representing the value of x at the time-step $t - \tau$. The behavior of the time-series displays a range of chaotic and periodic dynamics for different values of the parameters; in the literature on ESNs, usually $\tau = 17$ is used for a mildly chaotic behavior and $\tau = 30$ is used for a strongly chaotic behavior. We will use for our experiments exactly these two resulting time-series. We will try to predict at first the easier one ($\tau = 17$) and we will leave the harder one ($\tau = 30$) for later. We use the same training steps and testing steps as used in general on Mackey-Glass, 3000 (with a washout of 1000 steps) and 84 respectively. The noise added to the argument of the activation function (all versions of it) is drawn from a uniform distribution between $(-0.00001, 0.00001)$. For almost all experiments we used this type of noise, unless mentioned otherwise. We tried also using individual noise for every neuron, but the results were worse and so we used just one value of the noise at each iteration, common to all neurons. We show in Figure 4.1 a sample of the Mackey-Glass time series with $\tau = 17$. First we use *tanh* neurons, with an orthonormal weight matrix. The default network size for these experiments was $nsize = 400$ as this is also the size used by [51].

4.1.2 Linear Neurons

We then used linear neurons to predict the same Mackey-Glass with $\tau = 17$. We again used an orthonormal weight matrix and a feedback vector of ones. In general, we are exploring the ESN with an orthonormal weight matrix, but for the sake of completeness we show also comparisons with the random weight matrix, rescaled, such that the spectral radius is 0.9, as generally used in the literature. For this dataset the linear neurons perform poorly, so we don't show the results of the experiments

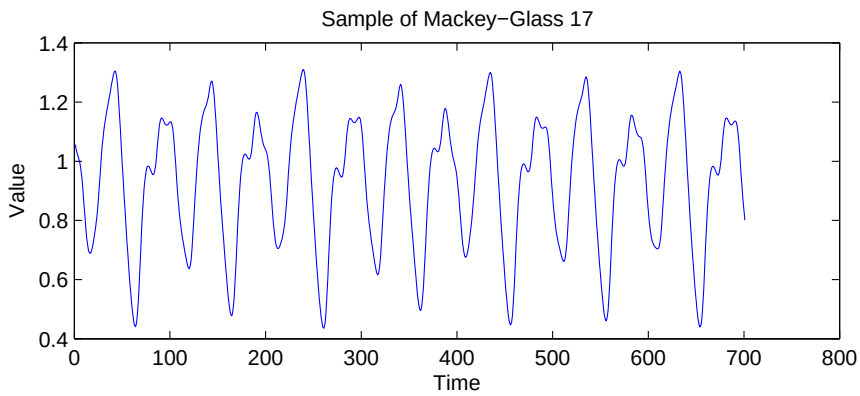


Figure 4.1: Sample from the Mackey-Glass 17

here. We will see later that for some datasets the linear version of the ESN performs better than the non-linear one, as is also the case in the MSO problems, as shown in chapter 3.

4.1.3 Mixing Non-linear and Linear

We then tried a mix of the two approaches. We thought that if we want to have a stronger effect of the input on each state of the neurons we could change the update equation from one step to the other in:

$$\mathbf{x}(\mathbf{t}+\mathbf{1}) = tanh(W \cdot \mathbf{x}(\mathbf{t})) + W^{fb} \cdot \mathbf{y}(t) + v \tag{4.2}$$

We remind the reader the usual form of the equation:

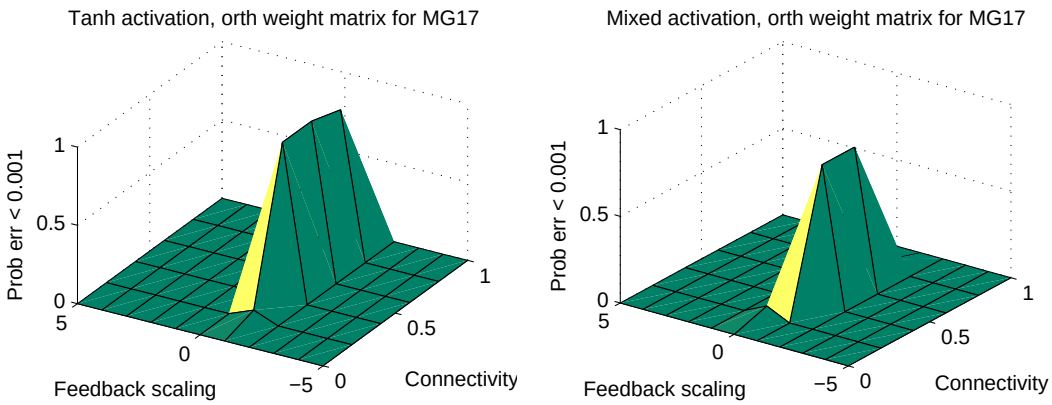
$$\mathbf{x}(\mathbf{t}+\mathbf{1}) = tanh(W \cdot \mathbf{x}(\mathbf{t}) + W^{fb} \cdot \mathbf{y}(t) + v) \qquad \text{(equation 2.1)}$$

This (equation 4.2) gave reasonable results, but worse than with the usual *tanh* activation function (equation 2.1). This version of the equation has a similar effect to regularization, adding the current input at every time step with a bigger weight than when using the usual form of the equation. We will see that in another setting this actually enables us to perform some perturbation on the structure of the matrix which normally does not work with the usual *tanh* activation. We should mention here that adding noise to the activation function is critical for finding good solutions. Also the magnitude of the noise is critical, if we lower the noise, the variability from one repetition to the other increases, however the lowest achieved NRMSE is lower than when using more noise. So we can consider the noise as a kind of transient perturbation which directly influences the variability and the lowest NRMSE possible, with the addition that increasing the noise stabilizes the solution with a precision trade-off (worse minimum NRMSE). So finding another perturbation mechanism would be ideal, to be able to lower the noise and still perturb the network enough as to get lower variability, and thus a more stable solution.

4.2 Feedback Scaling

As suggested by [62] we also tested different scalings for the output feedback connections (W^{fb}) and we show the results in Figure 4.2 for an orthonormal weight matrix and in Figure 4.3 for a random

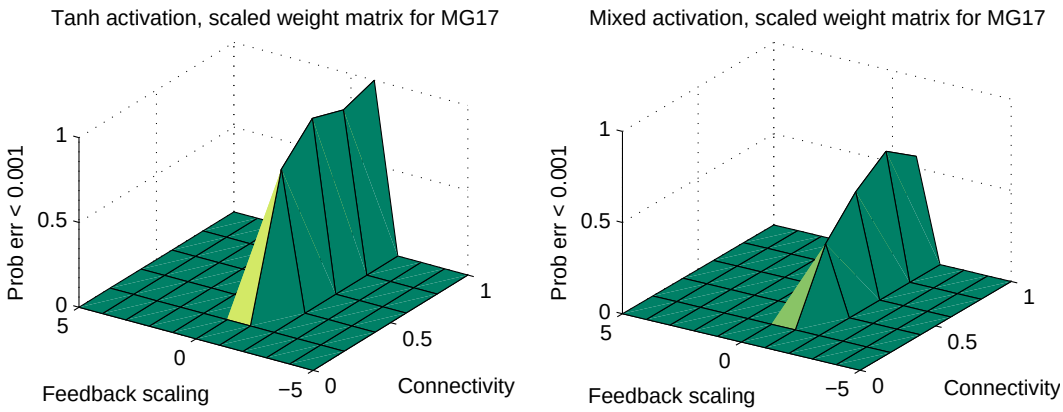
scaled weight matrix. We test this for both versions of the equation (eqs. 2.1 in Figure 4.2(a) and Figure 4.3(a) and 4.2 in Figure 4.2(b) and Figure 4.3(b)). We use for this a scalar of the form 10^s, where s is the scaling and goes from -5 to 5 in increments of 1. We multiply this value with the original feedback vector (vector of ones) and we get the final version of the feedback scaling. We use in our experiments 16 iterations for each value of the connectivity (which goes from 0 to 1 in increments of 0.2) and the scaling value s. We then tested also for the usual matrix used in the ESN,



(a) Feedback scaling results for eq. 2.1 (b) Feedback scaling results for eq. 4.2

Figure 4.2: Results when scaling the feedback vector for both versions of the equation. Network size = 400.

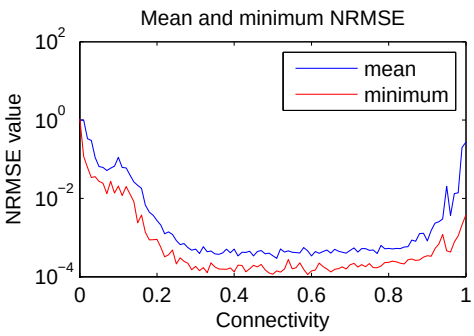
that is a random weight matrix with weights drawn from a uniform distribution between -1 and 1 and the spectral radius set to 0.9. The difference in performance is almost not noticeable. We see



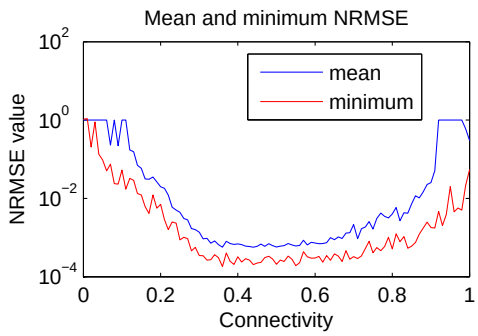
(a) Feedback scaling results for eq. 2.1 (b) Feedback scaling results for eq. 4.2

Figure 4.3: Results when scaling the feedback vector for both versions of the ESN equation, this time with a scaled weight matrix. Network size = 400.

that for both equations we find a critical value for the feedback scaling. We then scale the feedback vector to all the ESN versions ($s = -1$), and perform the experiments again, this time varying just the connectivity parameter from 0 to 1 in increments of 0.01 (Figures 4.4(a) and 4.5(a)). We then tried the same experiments but this time with a mixed activation function as described in section 4.1.3 (Figures 4.4(b) and 4.5(b)). The reason we are investigating this form of the equation is because by using this equation we manage to achieve the smallest error on the MG17 dataset, with a factor of 10^{-3} smaller than the previous best result achieved in the literature. The details will follow in section 4.4. We then perform the same series of experiments for a scaled weight matrix (Figures 4.6(a), 4.7(a) and Figures 4.6(b), 4.7(b)). We see a similar performance as for an orthonormal matrix.



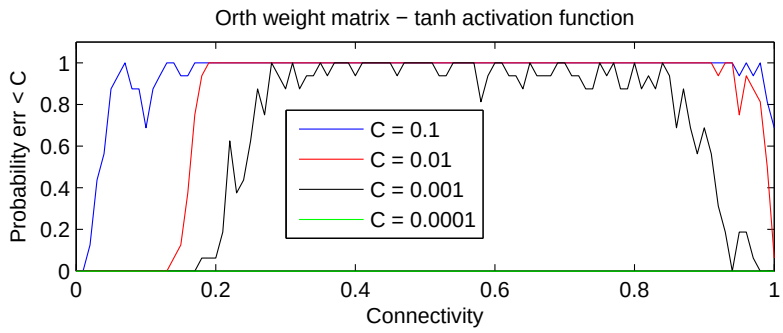
(a) Mean and minimum NRMSE when using an orthonormal weight matrix and a *tanh* activation function



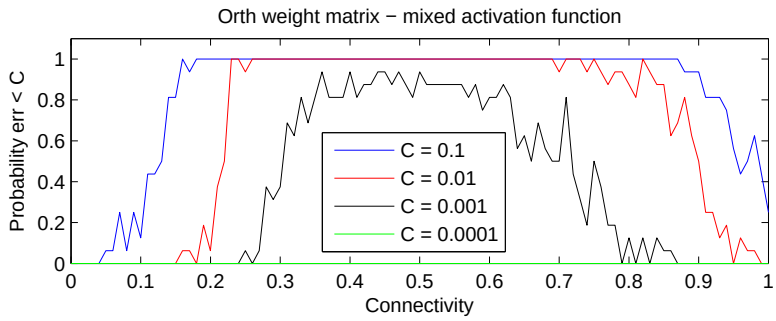
(b) Mean and minimum NRMSE when using an orthonormal weight matrix and a *mixed* activation function

Figure 4.4: Mean and minimum NRMSE as a function of connectivity, when fixing the scaling of the feedback vector to the optimal value found in the previous experiments ($s = -1$); using an orthonormal weight matrix and a *tanh* and *mixed* activation function. Network size = 400.

We can conclude that the difference between an orthonormal and the usual matrix is insignificant, while the usual *tanh* activation function is better than the *mixed* activation function, at least when performing these Monte Carlo experiments. In some plots the default mean NRSME was sometimes much bigger than 1, so we truncated it to 1 to better visualize the NRMSE.

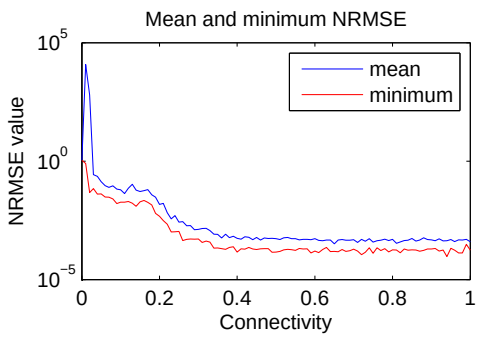


(a) Probability of finding a network which gives an error < C when using an orthonormal weight matrix and a *tanh* activation function

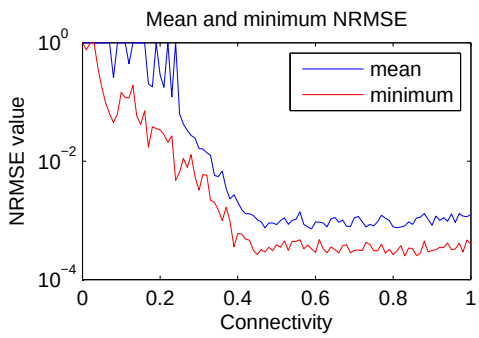


(b) Probability of finding a network which gives an error < C when using an orthonormal weight matrix and a *mixed* activation function

Figure 4.5: Probability of finding a network which gives an error < C when using an orthonormal weight matrix.

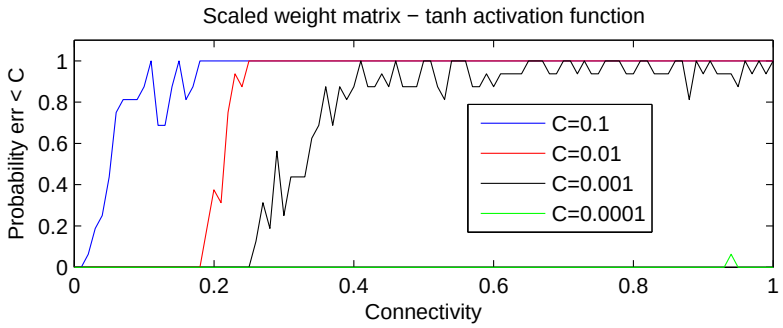


(a) Mean and minimum NRMSE when using a scaled weight matrix and a *tanh* activation function

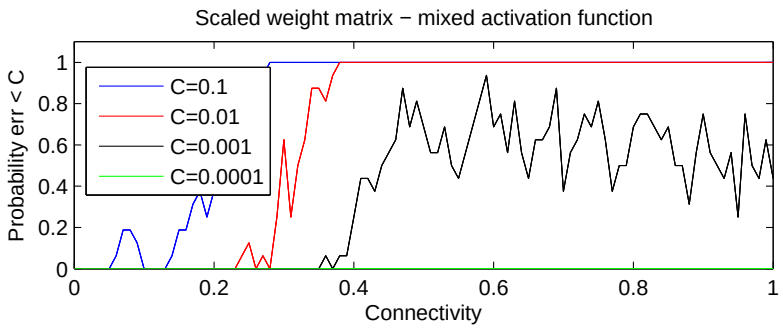


(b) Mean and minimum NRMSE when using a scaled weight matrix and a *mixed* activation function

Figure 4.6: Mean and minimum NRMSE when scaling the feedback vector for a scaled weight matrix and a *tanh* and *mixed* activation function. Network size = 400.



(a) Probability of finding a network which gives an error < C when using a scaled weight matrix and a *tanh* activation function



(b) Probability of finding a network which gives an error < C when using a scaled weight matrix and a *mixed* activation function

Figure 4.7: Scaled weight matrix results on Mackey-Glass with $\tau = 17$.

4.3 Different Read-outs

When considering the read-out method, many have reported improvement over the usual regression by using non-linear techniques. We wondered with what time requirements are we able to improve on the performance of the ESN employing linear regression. We investigate for this the Multi-Layer Perceptron, Ridge Regression and Support Vector Machines on the Mackey-Glass with $\tau = 17$ and 84-steps prediction.

4.3.1 Multi Layer Perceptron

When using a Multi-layer perceptron (MLP) the architecture (number of hidden layers and number of neurons in each layer) is critical for reaching a small error. The computational cost is much bigger than any regression, however, using an MLP readout enables the ESN to reach a small error in many more cases than when using regression. The variability problem doesn't seem to exist anymore, of course at the cost of increased computation time. Because the training of even a small MLP (we use the fast Levenberg-Marquardt backpropagation) takes quite a lot, we do not have the same exhaustive search as for regression, instead we take two networks, one which performs good when using a linear regression read-out (Figure 4.8) and one which performs bad when using the regression read-out (Figure 4.9), and then train the MLP on these two to compare with the linear regression. We see an interesting result. Low NRMSE can be achieved by the MLP in the case where linear regression cannot (NRMSE $\approx 10^{-4}$ compared to NRMSE $\approx 10^{15}$ for linear regression). Moreover, we are able to reach a NRMSE of 2.8×10^{-6} in the case where linear regression reaches 2.2×10^{-5}. So we can safely conclude that the MLP is a more powerful read-out technique than linear regression but with an added computational cost. Also the size of the MLP seems critical for good performance.

4.3.2 Ridge Regression

When using linear regression we minimize the sum of squared differences between the desired output (y_i, i = 1,..,N where N is the number of observations, in time in our case) and the predictions $\mathbf{w}^T\mathbf{x}_i$. The vector $\mathbf{w}$ satisfies $y_i = \mathbf{w}^T\mathbf{x}_i$. Thus the loss function which we are trying to minimize is:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 \tag{4.3}$$

In matrix form, this becomes:

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) \tag{4.4}$$

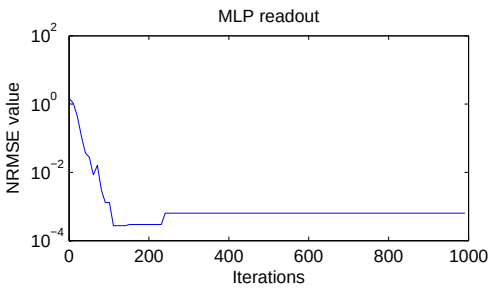
Now, taking the derivative w.r.t. $\mathbf{w}$ and equating to zero, gives us:

$$\nabla E(\mathbf{w}) = -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) = 0 \Rightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{Y} \tag{4.5}$$

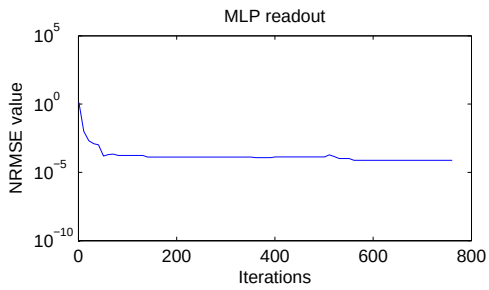
And taking the inverse on both sides we then get:

$$\widehat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{4.6}$$

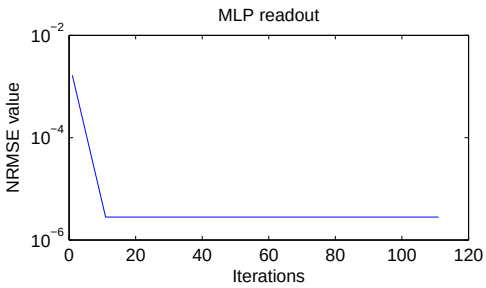
The equation above is often called the least-squares solution. However often the elements of $\mathbf{w}$ can get large while trying to fit the given data, and so the model becomes more complex and this can also



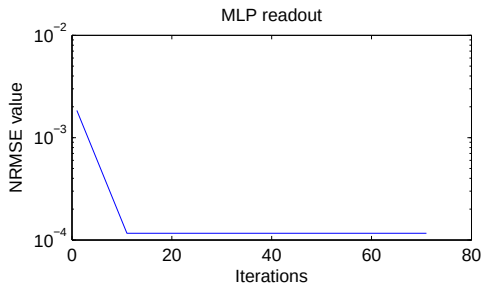
(a) MLP read-out. 1 layer of 3 neurons.



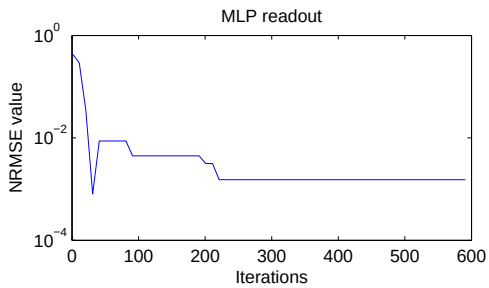
(b) MLP read-out. 2 layers of 5 and 3 neurons.



(c) MLP read-out. 1 layer of 10 neurons.



(d) MLP read-out. 1 layer of 20 neurons.



(e) MLP read-out. 2 layers of 3 and 5 neurons.

Figure 4.8: MLP read-out performance on an ESN which yields **good** performance (NRMSE $< 10^{-4}$) when using a linear regression read-out. Network size = 400.

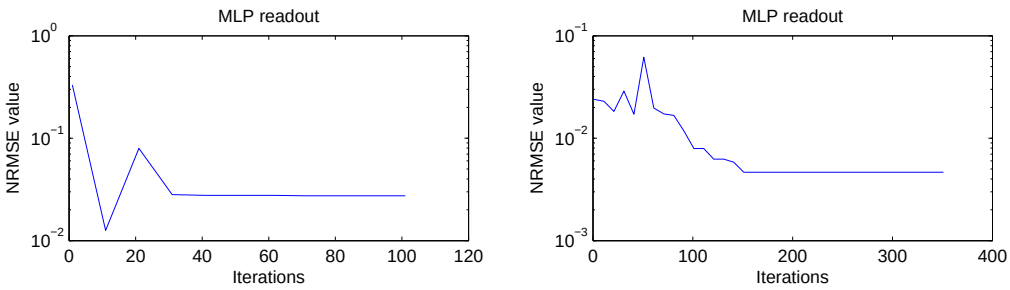
lead to over-fitting. One solution, which ridge regression employs is to add a penalizing term, like for example the squared norm $\mathbf{w}^T\mathbf{w}$. So, the equation becomes:

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \tag{4.7}$$

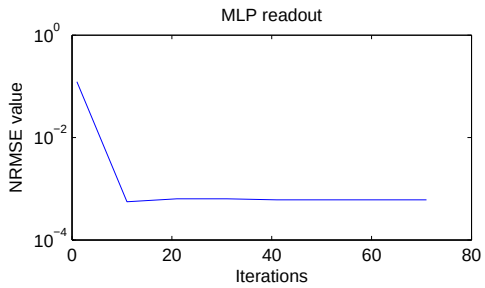
λ is called a hyper-parameter and controls the amount of regularization. Taking again the derivative w.r.t. $\mathbf{w}$, equating with 0 and then taking inverse on both sides gives us the final form of the solution:

$$\widehat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y} \tag{4.8}$$

When testing for λ we employ a series of Monte Carlo experiments, with K taking the values between -35 to 5 in increments of 2, and $\lambda = 10^K$. The weight matrix is as usual an orthonormal matrix, but this time 30 values from the first row are replaced with values from a uniform distribution between -1 and 1. This substitution enables the network to achieve high performance as we will see in more detail in section 4.5. We show in Figure 4.10 the probability of getting an error smaller than 0.01 from a series of 16 repetitions for each value of K and the connectivity parameter (that goes from 0 to 1 in increments of 0.2). We see the optimal value for λ is -9. We show in Figures 4.11 and 4.12 the comparison between linear regression and ridge regression on Mackey-Glass with $\tau = 17$. We see



(a) MLP read-out. 1 layer of 5 neurons. (b) MLP read-out. 2 layers of 5 and 3 neurons.



(c) MLP read-out. 1 layer of 10 neurons.

Figure 4.9: MLP read-out performance on an ESN which yields **bad** (NRMSE $> 10^{15}$) performance when using linear regression read-out.

Figure 4.10: Plot showing the NRMSE obtained by ridge regression when varying λ and the connectivity. Network size = 400.

that even if ridge regression is much more stable giving relatively good results in a wider range of the connectivity parameter, i.e. the probability of getting an error smaller than C is 1, for many values of C, the minimum NRMSE is still achieved by the usual linear regression with no regularization term (in Figure 4.11 the yellow line, for C=0.00001, has a peak close to connectivity 1, compared to Figure 4.12 where is always 0).

4.3.3 Support Vector Machines

As before for the MLP read-out, the added computational cost of running an SVM, constrains us to run the SVM on just two networks: one good network which yields a small error when using the linear regression read-out and a bad network with linear regression. The purpose is the same as for the

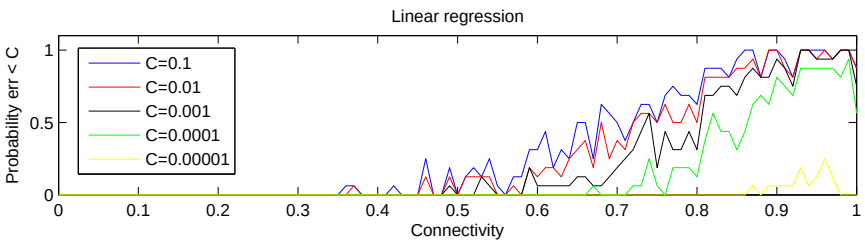


Figure 4.11: Plot showing the NRMSE obtained by linear regression on MG17 (the network is perturbed, see text for details). Network size = 400.

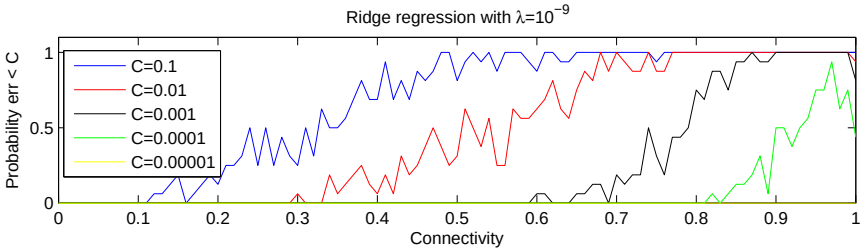


Figure 4.12: Plot showing the NRMSE obtained by ridge regression on MG17 with $\lambda = 10^{-9}$ (the network is perturbed, see text for details). Network size = 400.

MLP, to compare the SVM with linear-regression in both cases. We use the LibSVM library, which has also a wrapper for Matlab. After 5-fold crossvalidation we get the following parameters: '-s 3 -t 2 -c 0.87595 -g 0.0625 -p 1e-012' for the good network and '-s 3 -t 2 -c 0.87595 -g 2 -p 1e-007' for the bad network. Where the parameters are the usual LibSVM parameters: s is the SVM type, ϵ-SVR in our case, t is the kernel type, radial basis kernel in our case, c is the C parameter of the SVM, g is the γ in the kernel function, and p is the ϵ in the loss function of the ϵ-SVR. For the crossvalidation we varied just γ and ϵ. The NRMSE reached was 0.007 for the good case and 0.006 for the bad case, which confirms our conclusion from the MLP experiments, i.e. using non-linear read-outs improves the general performance of the ESN, apparently eliminating the variability problem, but with a significant computational cost added. However for the SVM the lower NRMSE reached was not as low as for the MLP or even the linear regression case. But this might be due to further tuning needed of the SVM.

4.4 Adding Randomness on a Row

Applying Particle Swarm Optimization (PSO) to a row or column of the ESN yielded interesting results (details of the experiments are given in the next chapter), and so we wanted to investigate what happens if we set some values on a row to random numbers (we choose to order the chapters like this and not in the order of development, to group different methods under the same title, i.e. efficient methods). We do this based also on the fact that randomness usually helps in the ESN paradigm. We choose a uniform distribution with values between -1 and 1. So we start with the usual orthonormal matrix and then set a number (r) of values from a row to random values in [-1,1]. We perform 16 repetitions for each value of r, as each repetition chooses some other individual values to change. We see a very interesting behavior. Adding randomness to one row stabilizes the network, meaning that it is more probable to find a good network when adding more random values to the respective row (row number one in our experiments). This is the most unexpected result. Reaching a lower value of the

NRMSE from many repetitions would be understandable, as randomness helps a lot when searching for good ESNs, as we saw in multiple experiments previously in this chapter, but lowering the variability of the ESN this much by introducing randomness is a most surprising fact, at least for the author. We show in Figure 4.13 the effect of using a uniform distribution [-1,1] on the probability of getting a certain NRMSE, in Figure 4.14 we show the effect of using a Gaussian distribution (mean 0, standard deviation 1), and in Figure 4.15 we show the effect of using a constant value on all positions replaced (in this case the value of 1 is used). We then select a good network from the above experiments and perform PCA on the network states, to have a better insight on the inner dynamics of the network. We also show in Figure 4.16 the eigenvalues of this modified weight matrix. In Figures 4.17 we show the first 4 principal components (their projection in the original space), in Figure 4.18 the variance explained by the respective number of components, and in Figure 4.19 we show the network trajectory in the space defined by the first 3 principal components.

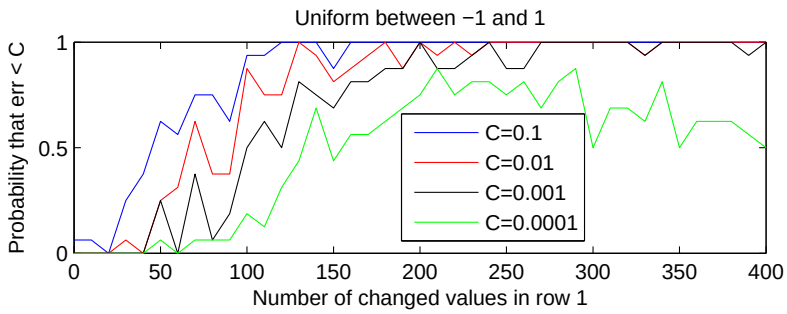


Figure 4.13: Replacing random values of row 1 from a uniform distribution [-1,1] on Mackey-Glass 17. Network size = 400.

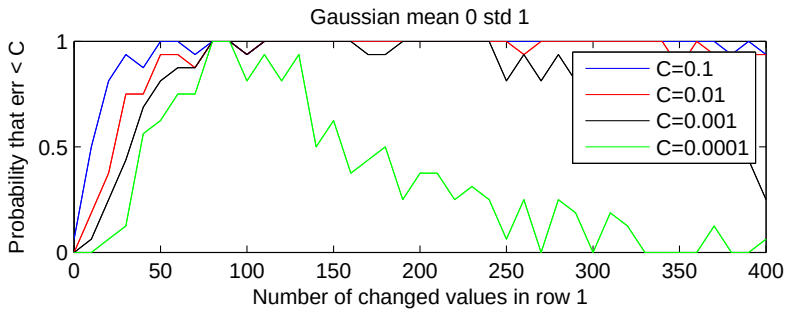


Figure 4.14: Replacing random values of row 1 from a Gaussian distribution with mean 0 and std 1 on Mackey-Glass 17. Network size = 400.

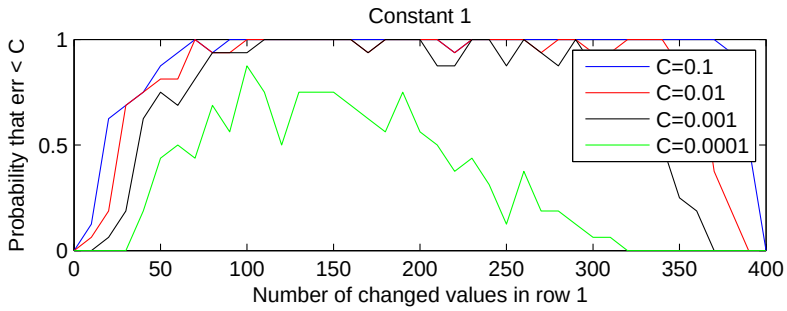


Figure 4.15: Replacing random values of row 1 with a constant 1, on Mackey-Glass 17. Network size = 400.

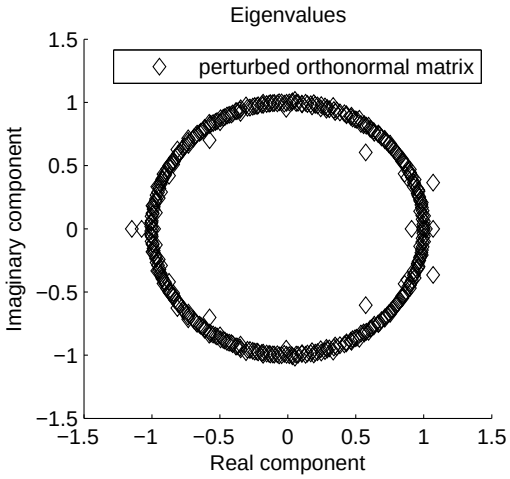


Figure 4.16: Eigenvalues of a perturbed orthonormal weight matrix, on Mackey-Glass 17. Network size = 400.

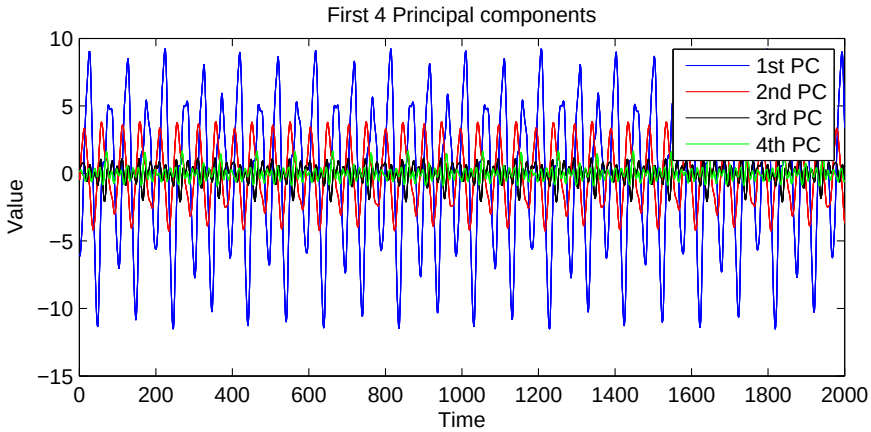


Figure 4.17: The four principal components.

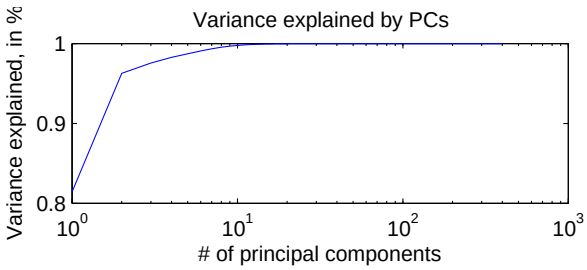


Figure 4.18: Variance explained by the respective number of PCs.

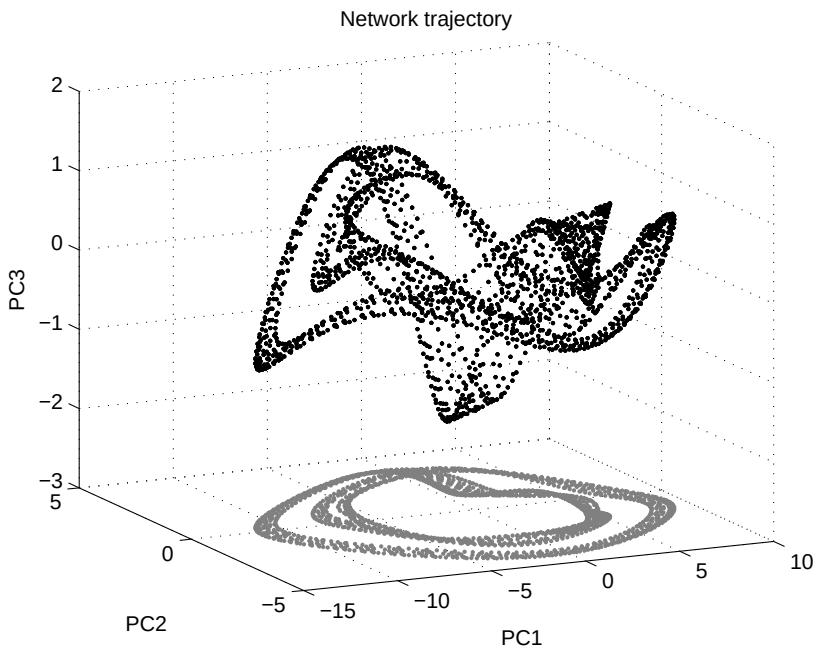


Figure 4.19: Network trajectory in the space defined by the first 3 PCs.

We then wondered what happens if we add randomness on a second row. We take for this row number 2 and perform 16 repetitions for each value of r of the two rows, $r1$ and $r2$. We increment r with 20 at each iteration, as we consider this is still a good description of what the solution space defined by the two values of r looks like.

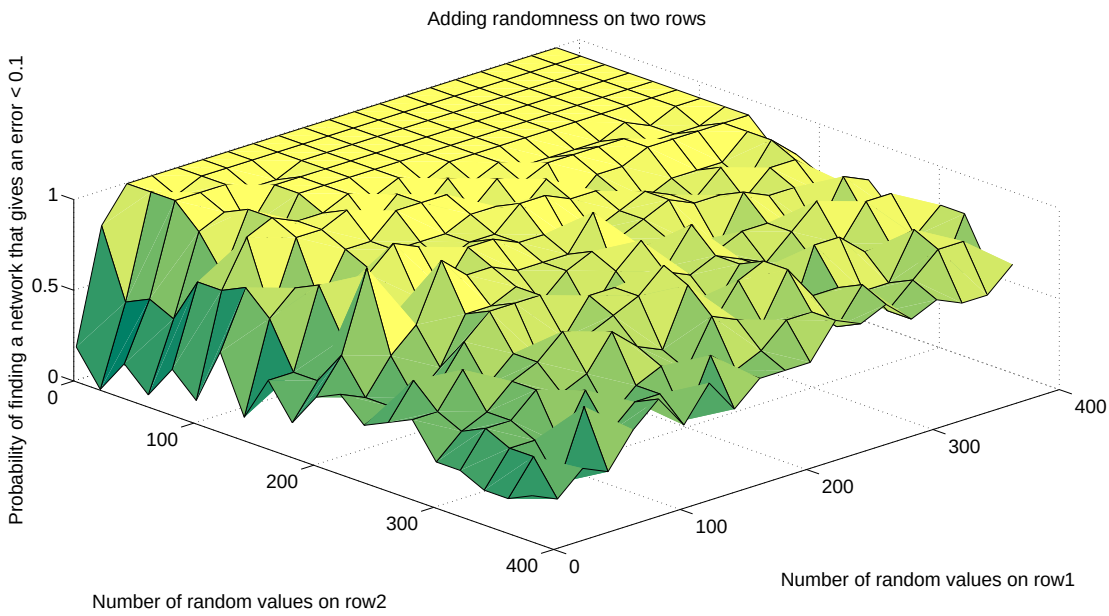


Figure 4.20: Plot showing the probability of finding a network which gives an error < 0.1 when adding randomness on two rows. Network size $= 400$.

We expected to have the same effect for the second row also, meaning that the surface plot should have been symmetrical, however we see in Figures 4.20 and 4.21, to our surprise, that this is not so. The randomness in the second row has a different effect than the randomness in the first one. Thus, we need another set of experiments, one investigating each row individually, to see the behavior of

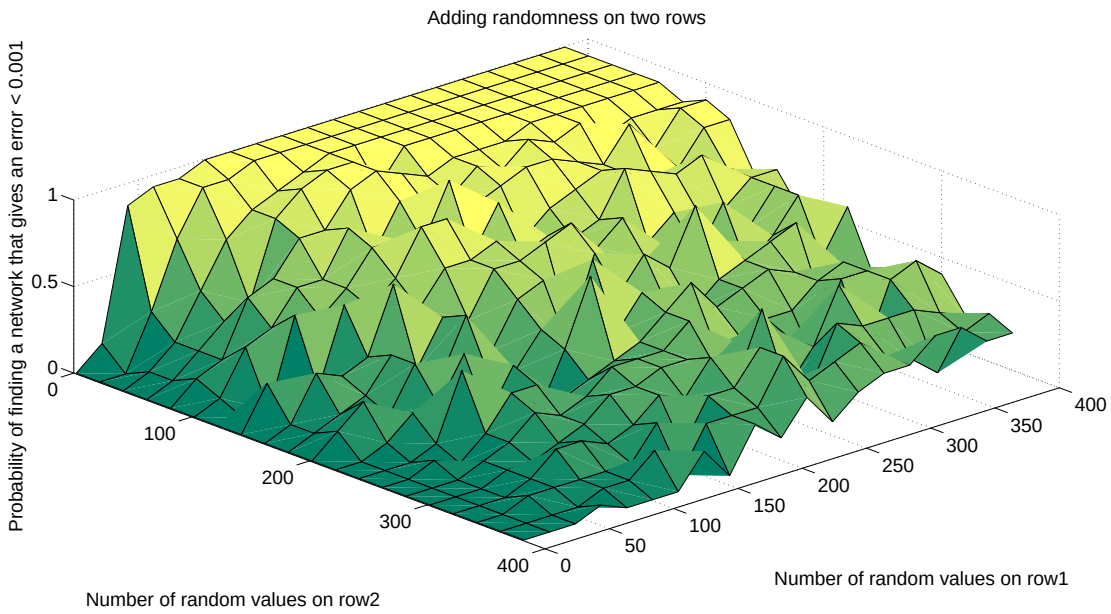


Figure 4.21: Plot showing the probability of finding a network which gives an error < 0.001 when adding randomness on two rows. Network size $= 400$.

the network when each of the 100 rows is modified as we described earlier. We perform the same experiment for the columns. We start from 0 and with increments of 10, advance to 100. We perform 16 repetitions for every value. We see the results showing the mean and standard deviation of the probability of getting a NRMSE smaller than 0.1, for all columns (Figure 4.22) and rows (Figure 4.23).

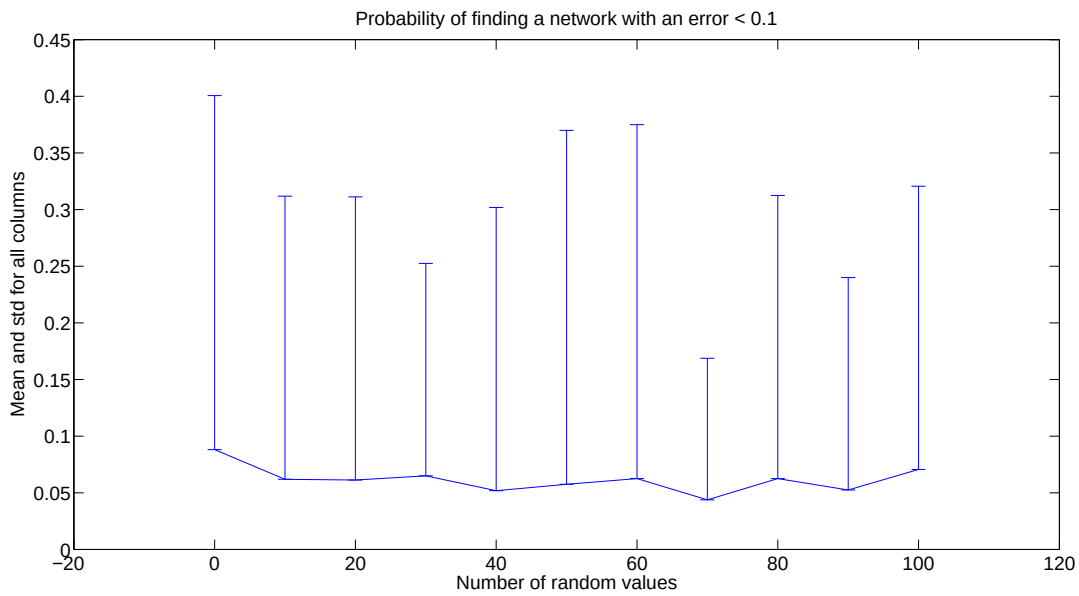


Figure 4.22: Plot showing the mean and standard deviation of the probability of finding a network with an error < 0.1 for all **columns** when adding randomness. Network size $= 100$.

We observe a very unusual result. Just the first row has the effect of stabilizing the network, all the other rows do not (we do not show the result here). We speculate that this happens because of the different distribution of weights in *column* number one, which is due to the orthogonalization process.

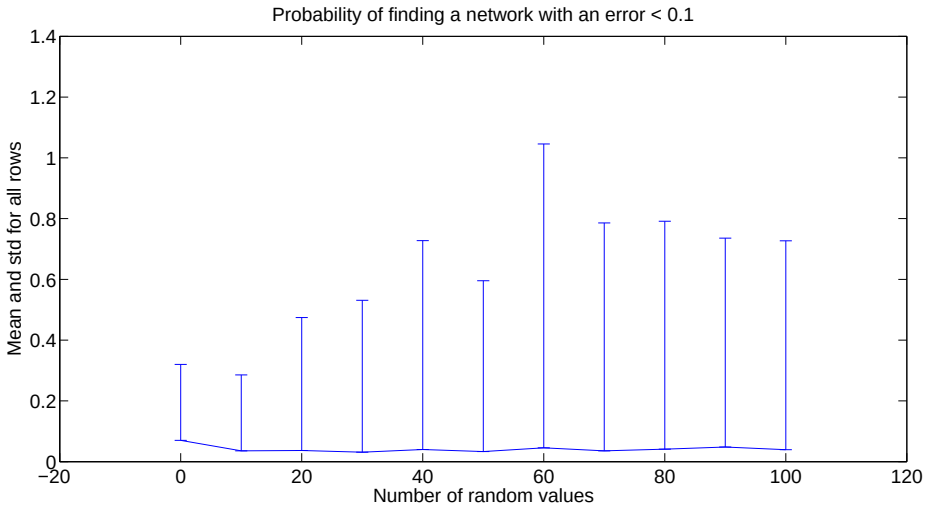
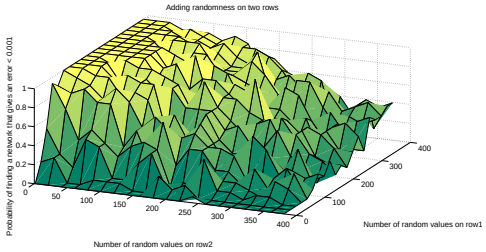
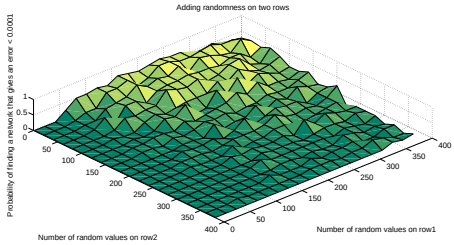


Figure 4.23: Plot showing the mean and standard deviation of the probability of finding a network with an error < 0.1 for all **rows** when adding randomness. Network size $= 100$.

To test this we replace column number one with another column (number 3) and then test the same process of adding randomness to that row (still 3). We conclude that our speculation was indeed correct, the probability to find a network with a small error when adding randomness to row 3 is indeed the same as for row number one when the column number one was in its default place. So the first column plays a major role in the dynamics of the network, as well as the first row. The values on a column are the inputs to one specific neuron, while the values on a row are the outputs to other neurons. So, in a way, it makes sense that if a neuron has an unusual activation (very different than the others), then filtering the output which the neuron gives to the other neurons is quite important indeed. And that is what the randomness added to the first row is doing, filtering (to be more precise random projecting) the signal of neuron number one to the other neurons. Following this train of thought we then realized that if we replace one more column of the matrix with a column which would have a distribution similar to column number one, then, by adding randomness to this second row also, we might improve even more on the stability of the network. We show these experiments in Figures 4.25, compared to when using the default second column in Figure 4.24. We see that the probability of getting a NRMSE $< 10^{-3}$ in Figure 4.24 is 1 almost everywhere in the space defined by $r1$ and $r2$, when column number two is replaced; this is very different than when using the default values for column number two. The column that we choose to replace is irrelevant, however, it is obvious that the row modified ($r2$) should then map to the respective column replaced. We could consider the process of adding different values to the row(s) as a kind of perturbation, or alteration to the original orthonormal weight matrix. The effect is similar to that of the noise added to the activation function, but instead of transient noise affecting also the observations of each neuron, this is a type of structural noise, added to the whole structure of the network. Maybe the effect is so strong also because the values changed affect all neurons, as the network is fully connected. We should mention that this technique yielded by far the best results we obtained with many datasets, among extensive experiments we performed throughout the development of this thesis. Further investigations we didn't have time for, might include constructing weight matrices which are mixtures of various distributions, or having individual rows or columns being such mixtures. Needless to say that we performed the same experiments with the usual random scaled weight matrix with a spectral radius set to 0.9 but we obtained no good results.

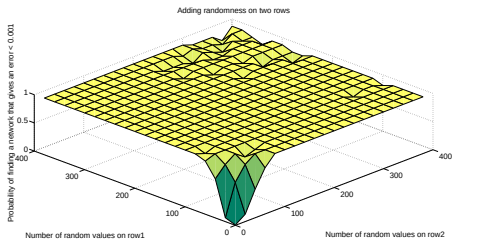


(a) Mackey-Glass 17 when adding randomness on two rows. Network size = 400. The Z axis is showing the probability of finding a network which gives an error $< 10^{-3}$

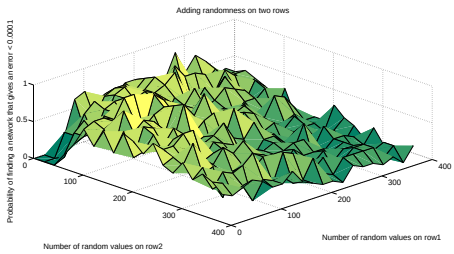


(b) Mackey-Glass 17 when adding randomness on two rows. Network size = 400. The Z axis is showing the probability of finding a network which gives an error $< 10^{-4}$

Figure 4.24: Plot showing the probability of finding a good network when adding randomness on two rows.



(a) Mackey-Glass 17 when adding randomness on two rows. Network size = 400. The Z axis is showing the probability of finding a network which gives an error $< 10^{-3}$



(b) Mackey-Glass 17 when adding randomness on two rows. Network size = 400. The Z axis is showing the probability of finding a network which gives an error $< 10^{-4}$

Figure 4.25: As Figure 4.24 but this time the second column has a distribution similar to the first column.

4.5 Other Datasets

4.5.1 Santa Fe Laser

To test our final approach, we want to see how our echo state network performs dealing with other datasets (time-series). We select for this the Santa Fe Laser dataset, used in [55, 85] and available from [1]. This dataset is a mixture of chaotic and periodic pulsation behavior of a real laser. The task is to predict the next laser activation value $y(t+1)$ given all the values until time t (including). We show below a sample of the dataset. We use 200 washout steps, 3000 training steps and 3000 testing steps. We tested the last approach which worked best for Mackey-Glass, which involves modifying

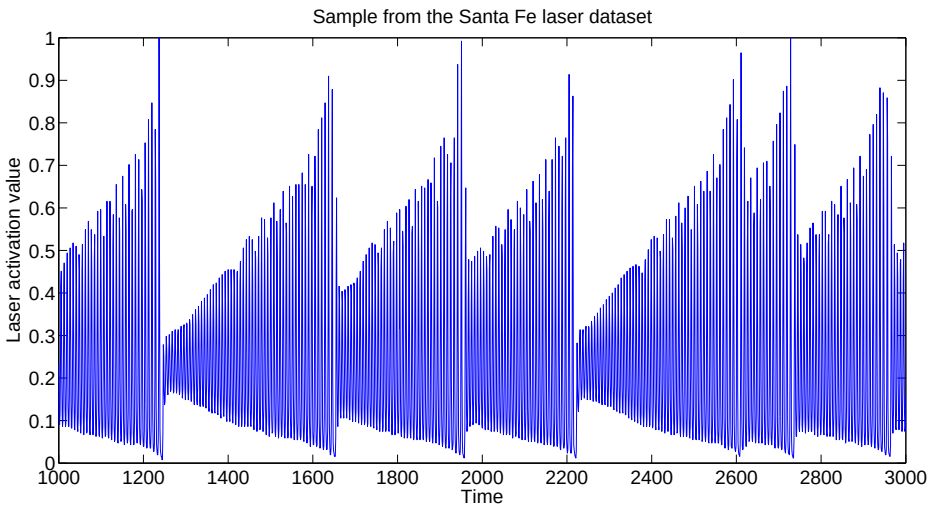


Figure 4.26: Sample from the laser dataset

the first row (we take the network size to be 200). We test with a uniform distribution between -1 and 1, a Gaussian distribution with mean 0 and standard deviation 1, and a constant of 1. The constant seems to work much worse for the Laser dataset so we don't show the plot. The other two work better, reaching an NMSE of 0.0021 when using a Gaussian distribution and 0.0027 when using a uniform distribution. What we notice from the plots, is that the variability of the ESN is much less than in Mackey-Glass and it is increasing when adding random values to the first row, even though the minimum is decreasing as shown in Figure 4.27. In Figure 4.28 we show the probability plot already familiar to the reader by now. The NMSE is smaller than the usual NRMSE. We give the NMSE, because usually the literature we found dealing with the Santa Fe laser dataset (as well as the Sunspots dataset) computes the NMSE, instead of the NRMSE. The NMSE is defined as (where $\langle\rangle$ denotes the expectation operator):

$$NMSE = \frac{\left\langle \|\widehat{y}(t) - y(t)\|^2 \right\rangle}{\left\langle \|y(t) - \langle y(t)\rangle\|^2 \right\rangle} \tag{4.9}$$

[1] http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html

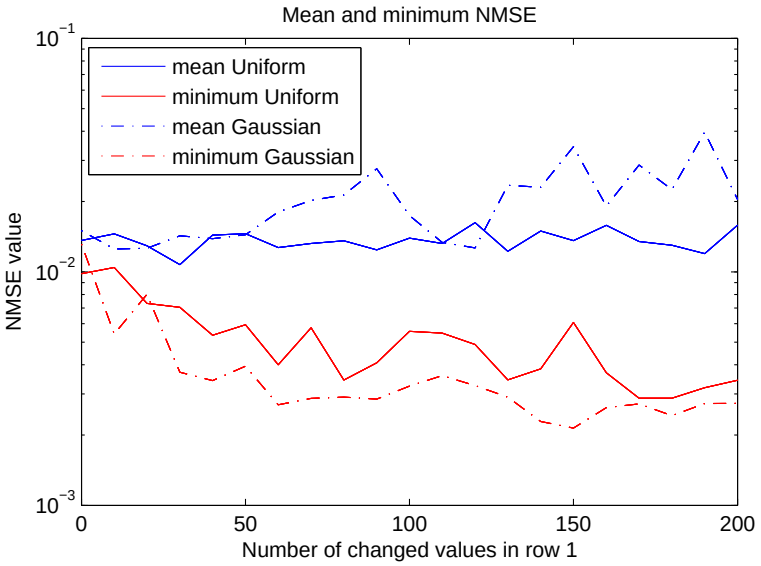


Figure 4.27: Mean and minimum NMSE for an orthonormal matrix with a *tanh* activation function on the laser dataset. The matrix has a modified first row.

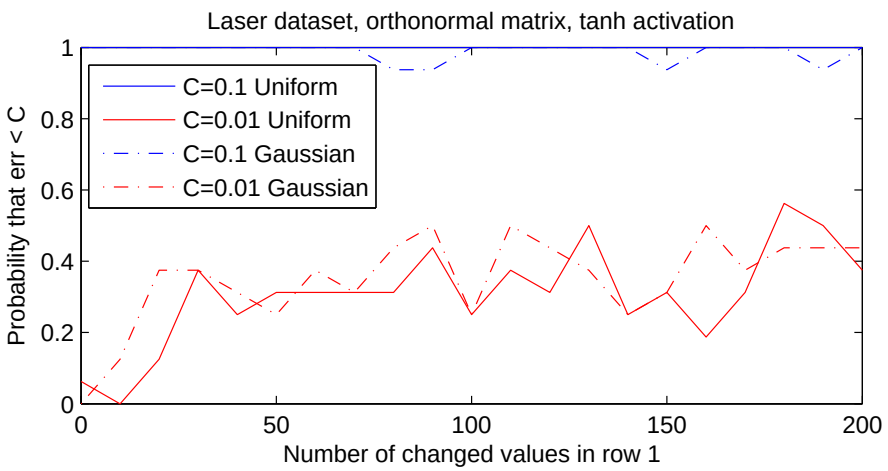


Figure 4.28: Probability that we get an NMSE $<$ C when using an orthonormal matrix with a *tanh* activation function on the laser dataset. The matrix has a modified first row.

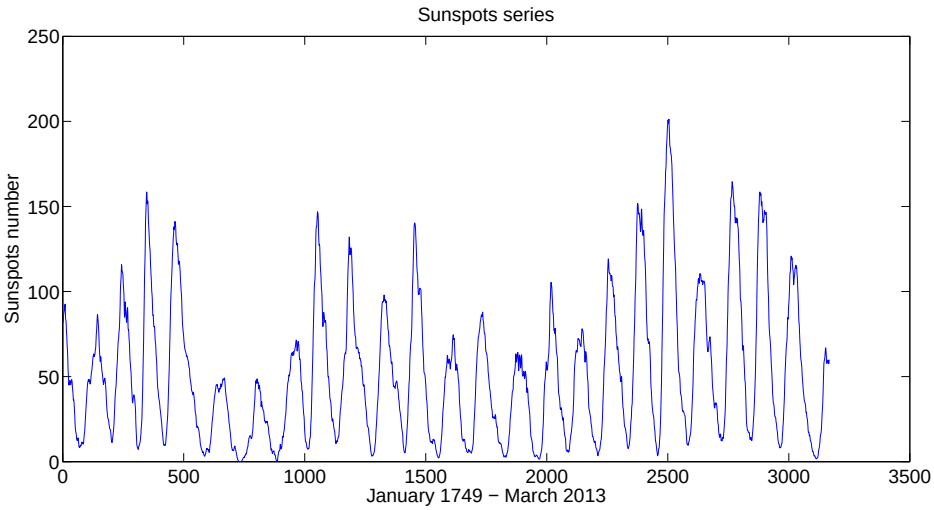


Figure 4.29: The sunspots series

4.5.2 Sunspots

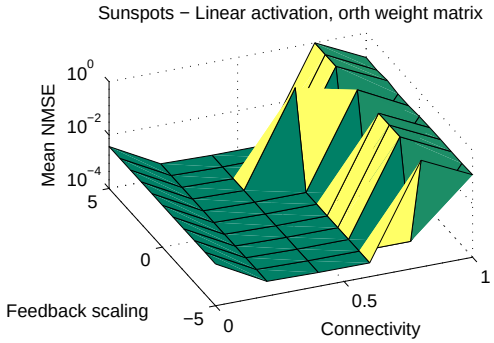
The sunspots series is another often used time-series in prediction, especially using ESNs. We take the time-series from [2]. The series contains 3167 values, representing the sunspots numbers from January 1749 to March 2013. The task is again to predict the step $y(t+1)$, given the values up to time t. We show in Figure 4.29 the sunspots time-series.

It seems that the one step prediction for the sunspots series is a relatively easy task for the echo state network. Even with two neurons it achieves an NMSE of 10^{-3} (we don't show the results here). When dealing with the sunspots time-series, the ESN functions more like a filter, a mapper, mapping the values of the current input, to the next value (it is just a one-step prediction), this is why it works reasonably well even with a 2-neuron network. Actually, even when the connectivity parameter is 0 (which means no synapses exist in the network), the ESN still manages to achieve an error of $\approx 10^{-3}$ which is pretty impressive. We see that the minimum NMSE we obtained ($\approx 10^{-4}$) is much smaller than for example [85], with $\approx 10^{-1}$ (keep in mind that we used 30 neurons for the sunspot series, while the authors in the article cited used 200 neurons). We note that we don't add any noise to the state equation when dealing with the sunspots dataset. We show in Figures 4.30(a), 4.30(b) and 4.30(c) the feedback scaling plots on the sunspots dataset when using an orthonormal matrix and the three types of activation function: linear, tanh and mixed respectively. In Figures 4.31(a), 4.31(b) and 4.31(c) we show the analogue plots but this time for a scaled weight matrix. The smallest NMSE was achieved with the linear activation function (very similar in magnitude in both cases of an orthonormal and scaled weight matrix).

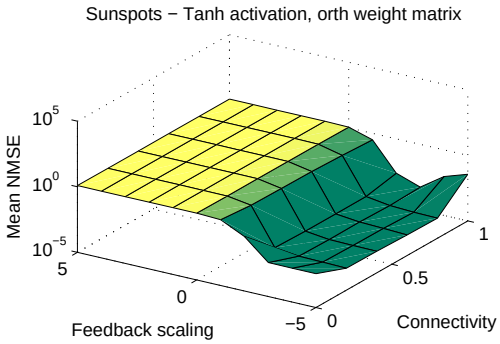
4.5.3 Mackey-Glass with $\tau = 30$

We show in Figure 4.32 a sample from the Mackey-Glass $\tau = 30$ time-series. We employ for it the same perturbation mechanism used before, but this time we add values from an exponential sequence, as the replacement for the first row. We show in Figure 4.34 the values in the replacement vector. In Figure 4.33 we show the results obtained when replacing the specified number of values from row number 1 (we go from 0 to 400 in increments of 10), from the respective distribution. For the

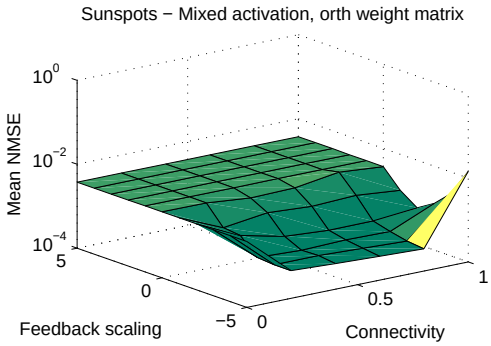
[2] `http://www.esrl.noaa.gov/psd/gcos_wgsp/Timeseries/SUNSPOT`



(a) The mean NMSE from 16 repetitions, for each specified value of the feedback scaling and connectivity.

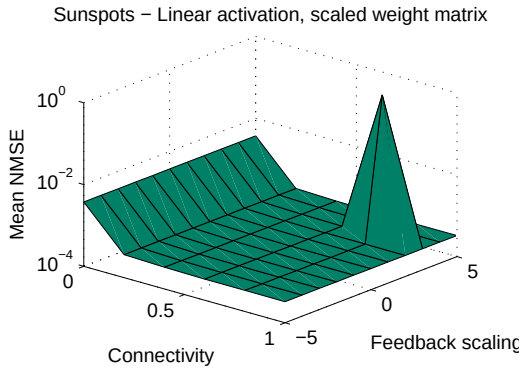


(b) The mean NMSE from 16 repetitions, for each specified value of the feedback scaling and connectivity.

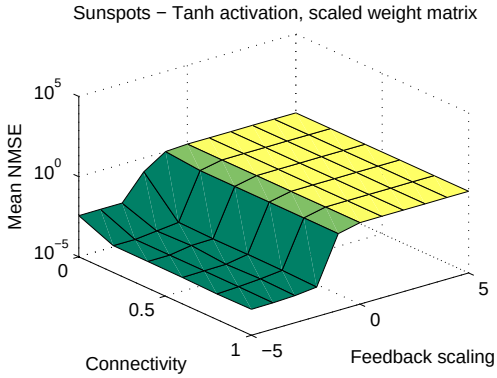


(c) The mean NMSE from 16 repetitions, for each specified value of the feedback scaling and connectivity.

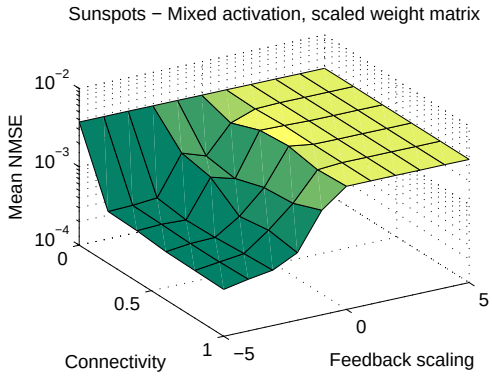
Figure 4.30: Orthonormal weight matrix on the sunspots series. Showing the mean NMSE as a function of feedback scaling and connectivity for the three activation functions.



(a) The mean NMSE from 16 repetitions, for each specified value of the feedback scaling and connectivity.



(b) The mean NMSE from 16 repetitions, for each specified value of the feedback scaling and connectivity.



(c) The mean NMSE from 16 repetitions, for each specified value of the feedback scaling and connectivity.

Figure 4.31: Scaled weight matrix on the sunspots series. Showing the mean NMSE as a function of feedback scaling and connectivity for the three activation functions.

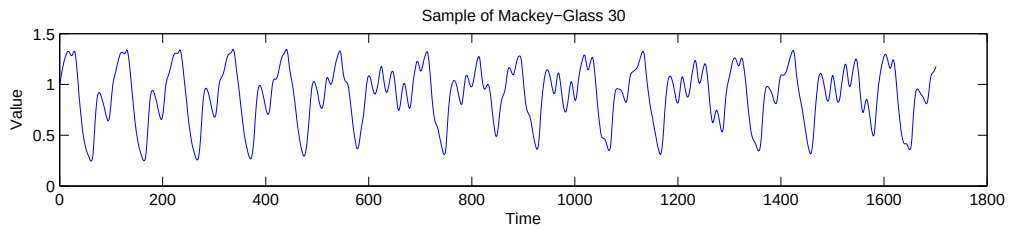


Figure 4.32: Sample from the Mackey-Glass 30

exponential distribution we test from 0 to 20 in increments of 1, as this seems to be the optimal interval. We manage to achieve a NRMSE of 0.0068. These results, confirm one more time, the fact that the perturbation technique is indeed highly performant with minimum computational resources.

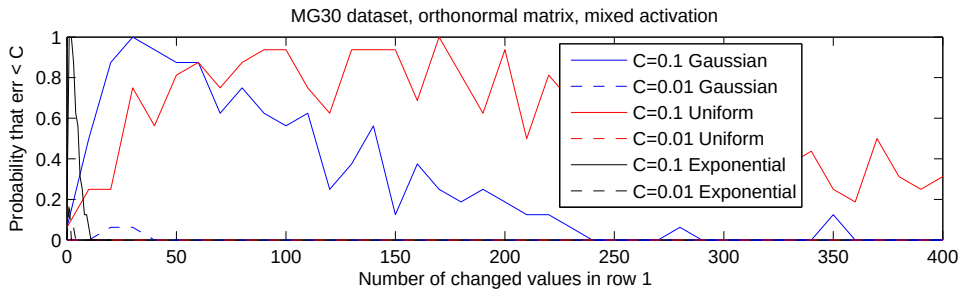


Figure 4.33: Results when replacing the first row with values from different distributions.

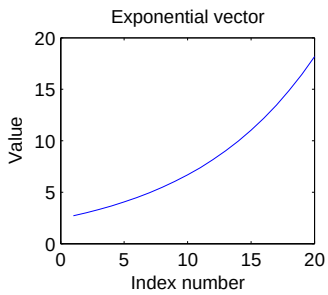


Figure 4.34: Vector replacing 20 values of the first row in the exponential case.

4.6 Discussion

We saw in this chapter how trying different versions of the activation function enable us to reach very low NRMSEs on many datasets. Investigating feedback scaling as well as connectivity is again critical for finding good ESNs for specific datasets. We then saw how different non-linear read-out techniques don't suffer from the variability problem as the usual linear regression does. With increased computational resources, the non-linear read-outs offer a more correct approximation of the true capabilities of the ESN. However for very low values of NRMSE it seems that the linear technique works best, but this conclusion might be also due to the relatively small number of experiments performed with the non-linear techniques compared to the linear one. We then saw a very interesting and simple modification of the orthonormal weight matrix which yields the best ESNs in terms of the minimum NRMSE achieved on many datasets. This same technique stabilizes the network to a

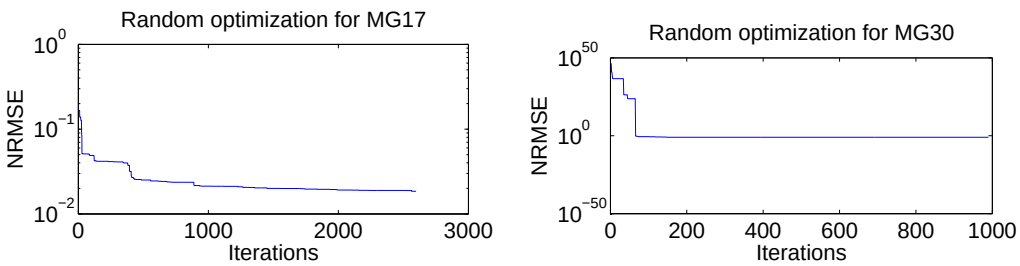
high degree without any added noise (as is usually done for ESN stability). Actually we interpret this weight matrix modification as a kind of structural perturbation of the matrix, similar to noise (which is a transient perturbation), but without the added computational cost and performance decrease associated with noise. What makes it even more interesting is that the computational resources of this technique are minimum and the process of training remains linear. Theoretical investigations are needed to unveil the process behind this high performance of the ESN constructed in such a way and maybe shed some light on the relationship between the distribution of the modified row and the statistics of the predicted signal.

Chapter 5

Efficient Methods for Finding Good ESNs

5.1 Random Optimization

In accordance with the ESN paradigm, meaning low computational time and exploration of randomness, we tried using random optimization [73], to improve network performance. Evaluating a small size ESN takes very little, a few hundred milliseconds with 3000 training steps. We saw in the previous chapter that training an MLP for a read-out takes a lot of time, the same for finding good parameters for an SVM. So we thought of performing Monte Carlo experiments in which at every repetition one (or more) weights are changed randomly and if it gives a smaller error, then the weight is saved, otherwise not. This is a very simple algorithm which gave very good results when tested on the Mackey-Glass with $\tau = 17$. The algorithm has two parameters: the first one, an integer is the number of weights which are changed with every repetition. To increase the variability of the search we made this parameter vary with every repetition, randomly, but we still have the maximum number of neurons changed with every repetition (the second one). We can see the pseudo-code of the algorithm in Algorithm 5.1. Some conclusions can be drawn from using this method. First, a local optimum is reached very fast, as probably they are very many, considering the big dimensionality of the system. Second, once a local optimum is reached, the solution cannot be improved much. However, this method is still a fast way of finding a suboptimal solution. We obtained 0.01 for MG 17 with a 30 neuron ESN and 0.09 for MG 30 with a 400 neurons ESN (Figure 5.1). This is not a very small NRMSE, however it is reached very fast.



(a) Mackey-Glass 17. Network size = 30.

(b) Mackey-Glass 30. Network size = 400.

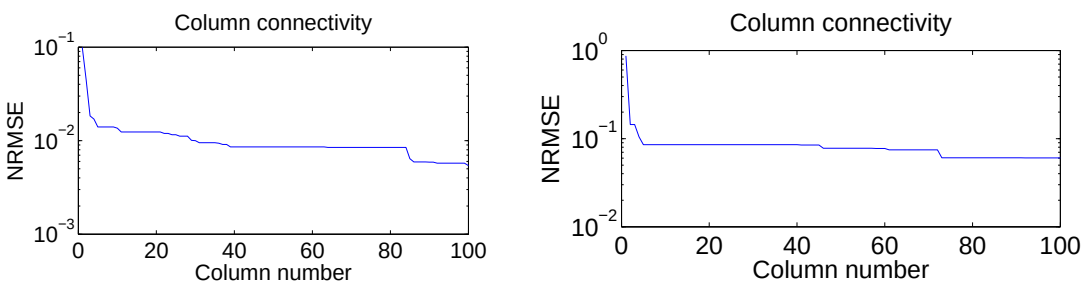
Figure 5.1: Plot showing the NRMSE when using random optimization for Mackey-Glass.

Algorithm 1 Random optimization

```
Init nsize, W_1, s_v, repetitions, minimum_error
W_1 ← orthonormal matrix, i ← 0
while i < number_of_iterations do
  W ← W_1
  rr,rc ← random vectors of integers of size s_v
  for j = 1 to repetitions do
    replace the weights W(rr, rc) with random values
    current_error ← compute_error(W)
    if current_error < minimum_error then
      W_1 ← W
      minimum_error ← current_error
    end if
  end for
  i ← i+1
end while
```

5.2 Column Connectivity

Seeing that connectivity is such an important parameter of the echo state network, we wondered what happens if we investigate individual column connectivity. To our surprise this turned out to be even a better method of finding a good echo state network for a specific time-series fast. We proceed by showing the results we got when iterating through the columns with different values of the column connectivity parameter at each repetition. We tried two approaches: first iterating sequentially through columns, and second, iterating randomly through them. For every value of the column connectivity we select randomly the respective number of weights to set to 0, and as we have many combinations of weights, we execute this procedure 10 times. The weights of the initial, fully connected weight matrix, come from an orthonormal matrix. We show also the pseudo-code in Algorithm 5.2. We have to mention that this method is computationally much less expensive than random optimization; to iterate through all columns of a small network (30 neurons), even though we have 10 values of the connectivity parameter and 10 repetitions, takes about 10 minutes and we're sure to find a reasonably good network.



(a) Mackey-Glass 17. Network size = 100.

(b) Mackey-Glass 30. Network size = 400.

Figure 5.2: Plot showing the NRMSE when exploring individual column connectivity values for Mackey-Glass.

Algorithm 2 Column connectivity

```
Init nsize, W_1, repetitions, minimum_error
W_1 ← orthonormal matrix, c ← 1, W_2 ← W_1
while c < number_of_columns do
  W ← W_1
  for column_connectivity = 0 : 0.1 : 1 do
    W(:, c) = W_2(:, c) // reset column to the initial fully connected column
    for j = 1 to repetitions do
      replace column_connectivity*nsize random weights with 0
      current_error ← compute_error(W)
      if current_error < minimum_error then
        W_1 ← W
        minimum_error ← current_error
      end if
    end for
  end for
  c ← c+1
end while
```

What we notice here is that after we iterate through the first few columns, the error does not improve much. Because the network is fully connected at first, changing some column values affects the whole network to a high degree. We see that when we increase the size, the same behavior remains, but the error reached is indeed much smaller. So the algorithm scales up with the network size, meaning that we are sure to find a good network according to the capabilities given by the network size by using this method. As we mentioned earlier we tested for sequential iteration and also random iteration through the column connectivity parameter and even though we did not notice a significant difference, the random approach seemed better, so we show in Figure 5.2 the error plot of the random iterating procedure. The pseudo-code is written for the sequential version, but we can easily imagine replacing the column number in each iteration with a random integer from 1 to the maximum size of the network. The plot for MG30 shows just 100 iterations because when having a 400 neuron network it takes a lot more time to go through all columns. However, the algorithm managed to achieve a NRMSE of 0.0605.

5.3 Particle Swarm Optimization (PSO) on a Column

Having concluded so far that a column is very important in the general behavior of the network, as the network is initially fully connected, it almost functions as a controller for the ESN, being able to switch from a general bad behavior (meaning an initial NRMSE of more than 10^{30}, similar to what we saw in Chapter 3 when not using regularization, Figure 3.9) to a reasonable behavior (NRMSE < 0.1) with just a few zero values in the column, leaving the rest of the network unchanged. This means that using PSO [58] (or another meta-optimization algorithm, we choose PSO for speed) could help in optimizing just one column of the ESN, at least to a good-enough local optimum, a fast not-so-bad solution. Optimizing the whole network with PSO would be unfeasible, as even for a small network we have too many parameters. We used 9 particles and 100 iterations for each topology. The parameters

for the moving dynamics of the particles are taken from [78]. The initial weight matrix is as before an orthonormal matrix which spans the range of a random matrix with weights between 0 and 1. We start with a fully connected network.

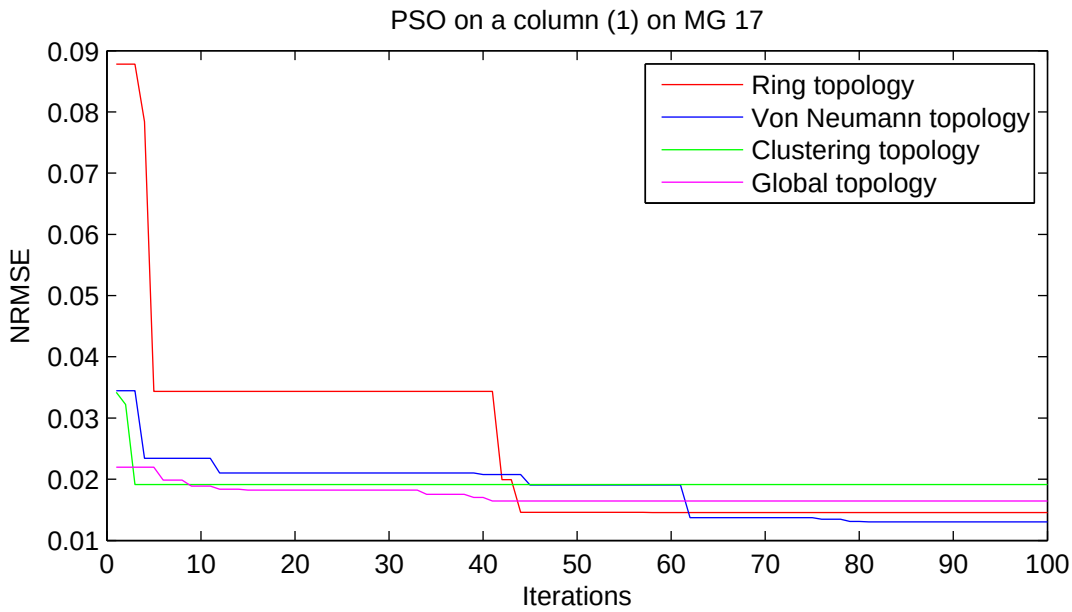


Figure 5.3: Plot showing the NRMSE when using PSO on the first column on MG 17. Network size = 100.

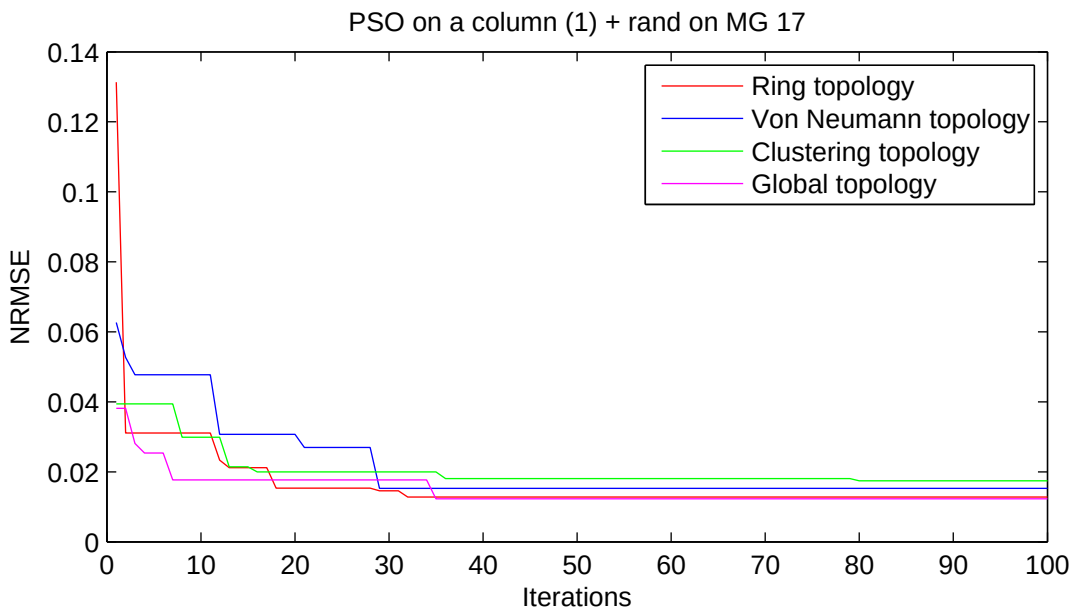


Figure 5.4: Plot showing the NRMSE when using PSO plus randomness on the first column on MG 17. Network size = 100.

The four topologies are given in Figures 5.5 and 5.6. The Clustering topology is similar to the global topology. The moving dynamics is as follows: we split the whole set of particles into clusters based on proximity; each member of a cluster goes to its cluster best, while the cluster best from each cluster goes to the gobal best. So we could say it is a hierarchical global topology, with particles going to the cluster best in the long run, but with an intermediate stop at the cluster best.

Figure 5.5: Topologies for PSO. Left - Global. Middle - von Neumann. Right - Ring.

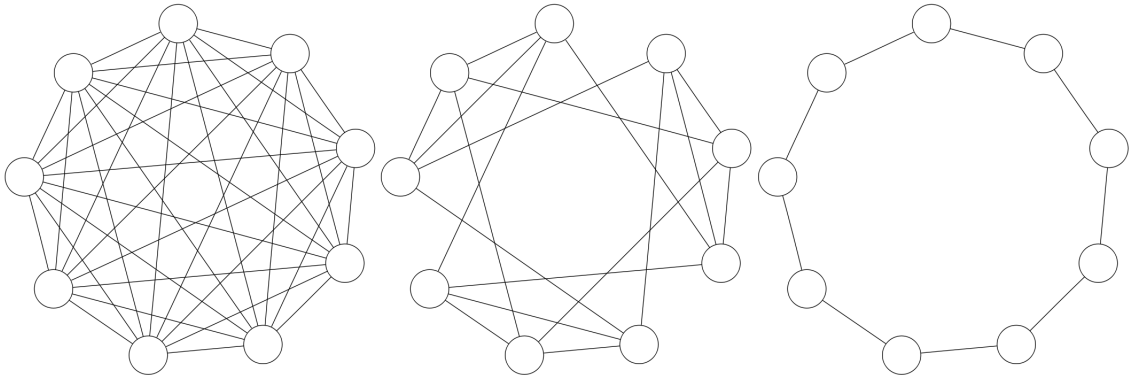
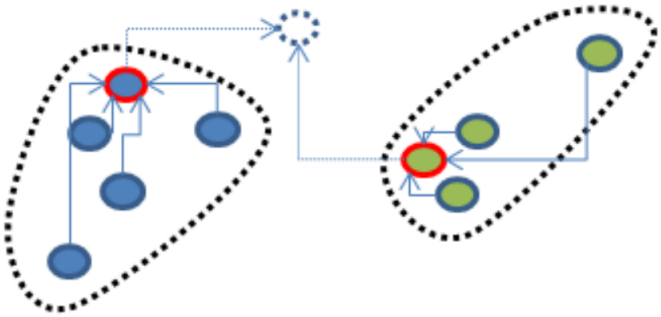


Figure 5.6: The clustering topology for PSO.



We thought that maybe adding randomness to the search process when using PSO will lower the NRMSE further, as we saw earlier randomness is usually beneficial to the search process. Thus, we reinitialize the whole network at each ESN evaluation, keeping only the column we are optimizing with PSO. We manage to achieve a smaller NRMSE with all four topologies when at each function evaluation we reinitialize the weight matrix (still to an orthonormal matrix), and we keep just the column we are trying to optimize, in this case, column number one.

5.4 Particle Swarm Optimization on a Row

We investigated earlier what happens when we use random optimization on a single column and we concluded that the process yields a worse error than when using the same process on a row. When using PSO we expected the same good result for a single row, so we were not surprised to see that the errors achieved were much lower than on a column. We even reach an error of 5×10^{-5} in just a few iterations (< 10) with a 400 neuron network which is remarkable. We start the same from a fully connected network, with the weight matrix being an orthonormal matrix which spans the space of a random matrix with weights between 0 and 1. We show in Figure 5.7 for comparison, the PSO on the first row with a 100 neuron network, as in the previous subsection, dealing with PSO on a column. We observe that the difference is significant, all topologies perform better.

We performed the same experiments for the row as we did for the column, one in which the matrix changes at every function evaluation (the +rand part in the Figures) and the normal one, in which the matrix remains the same. We expected to see the same improvement and in the previous subsection, however, this time the performance is inversed (except for the Clustering topology). The errors are

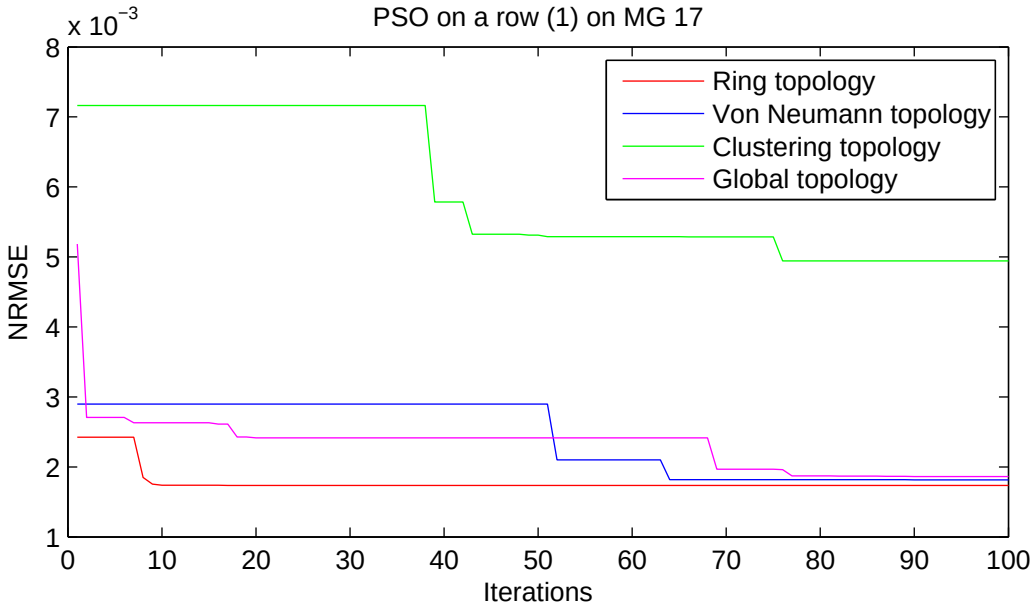


Figure 5.7: Plot showing the NRMSE when using PSO on the first column on MG 17. Network size = 100.

smaller without adding randomness.

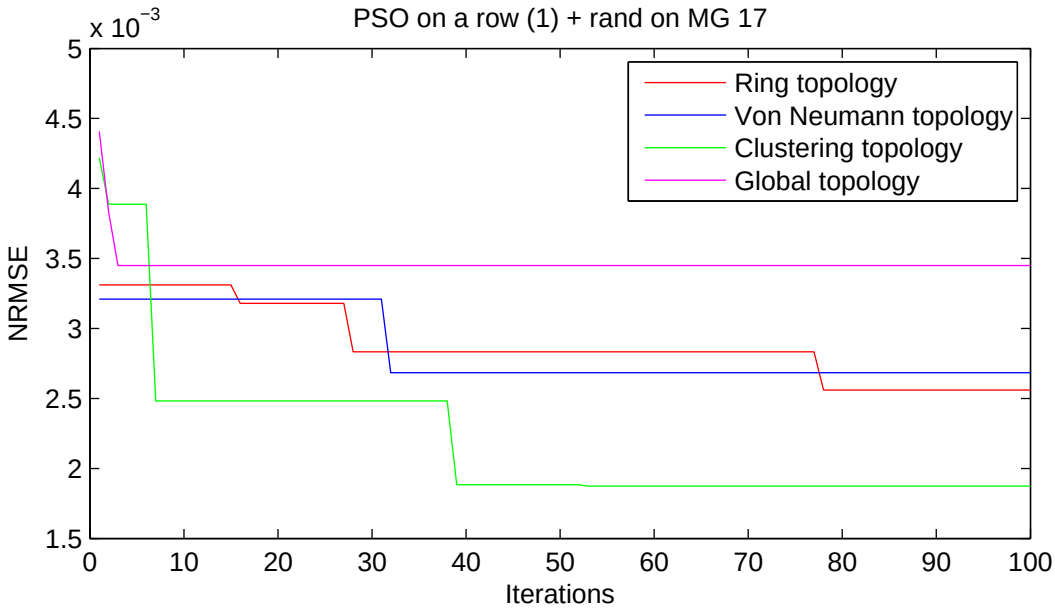


Figure 5.8: Plot showing the NRMSE when using PSO plus randomness on the first column on MG 17. Network size = 100.

5.5 Echo State Networks as Complex Networks

Complex networks is the name given to a multidisciplinary field on some networks of nodes. The applications which find themselves in this paradigm range from physics to geography, epidemiology, analysis of the internet, analysis of researchers' cooperation graph, the brain and more. Basically, any complex system which can make use of vertices (entities which have more or less the same structure)

and edges (connections between them, which can be cost, distance, or some other higher-order structure) can be modeled as a complex network. The field has a rich history and is still a developing field, with a profound theory underlying the description and analysis of such networks. However, in the ESN literature just one approach (to the author's current knowledge) finds itself in this field, namely [31]. We will try to deal with some of these models in the context of the echo state networks, namely build and analyze networks in ways specific to the complex network approach.

5.5.1 Scale-free Models

A scale-free network is a network whose degree distribution follows a power law, namely the fraction $P(k)$ of nodes having degree k, i.e. connections to other nodes scales as:

$$P(k) \approx k^{-\gamma} \tag{5.1}$$

where gamma typically ranges between 2 and 3. Many real-world networks have been conjectured to be scale-free, like the World Wide Web, social networks, brain networks, etc. We will start by building an echo state network constrained by this degree distribution and see how it performs when dealing with the Mackey-Glass chaotic time-series. A first experiment shows that for small networks (50 neurons) the network is too sparse and cannot catch the input dynamics. The first experiment was done with the weights drawn from an orthonormal matrix, i.e. if the weight should be present according to the above formula, then get the weight of an orthonormal matrix the same size as our matrix, which is on the same position. Trying with a 400 neuron network again gave no results whatsoever. However when inverting the probability such that, $P(k) \approx 1 - k^{-\gamma}$ we get decent results (Figure 5.9), the network even reaches an error of 5.47e-004, but this performance is most probably because of the big size of the network, as we saw in the previous chapters this is critical for some time-series. Next, we investigate the performance of more complex network models when dealing with the same Mackey-Glass with $\tau = 17$.

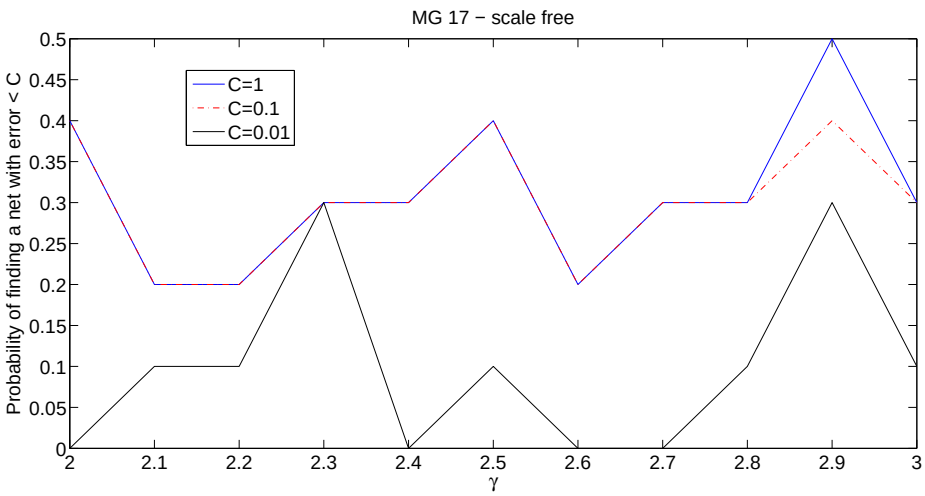


Figure 5.9: Scale free network. Network size = 400.

5.5.2 The Erdos-Renyi Graph

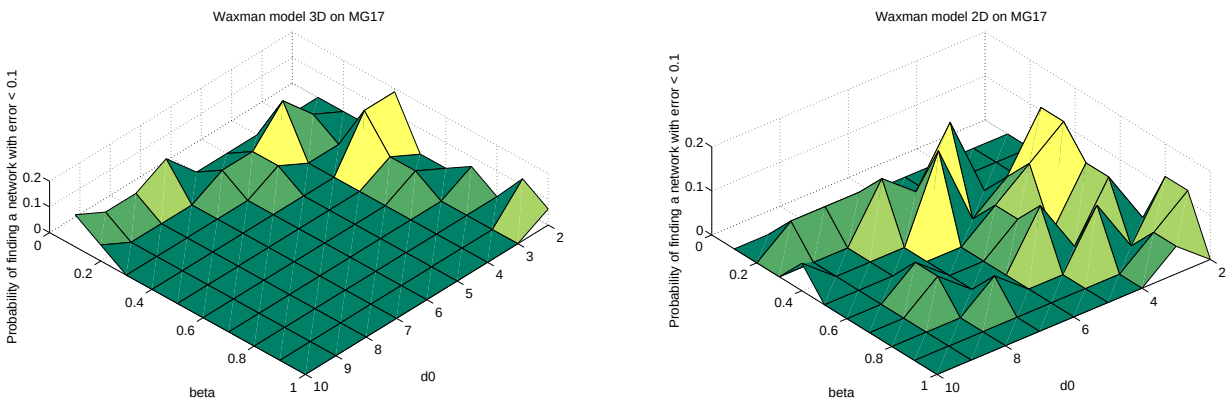
The Erdos-Renyi (ER) graph is usually used as the null model and is the basic approach to random graphs. The generation of an Erdos-Renyi graph involves going through all pairs of nodes and connecting them with probability p. This is usually how random matrices are generated in the ESN paradigm, and p is referred to as the connectivity value. We don't show the experiments as they would be very similar to the ones investigating the connectivity parameter in the previous chapter.

5.5.3 The Waxman Model

The Waxman model [112] is a random topology generator and is similar to the ER model, but with a spatial component. In this model the nodes are distributed uniformly in an n-dimensional space, usually n=2, and edges (or links, or connections) are then added to the model with probability proportional to the distance between nodes:

$$P(i,j) = \beta e^{-d_E(i,j)/d_0} \tag{5.2}$$

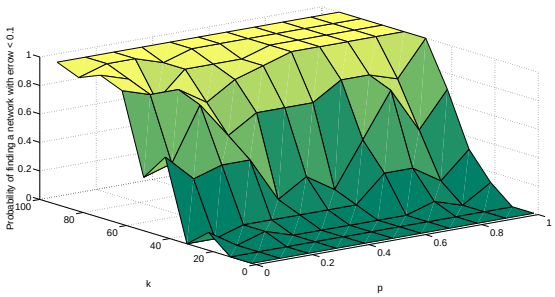
where $d_E(i,j)$ is the distance between node i and node j, β is a parameter which controls the density of the connections (or connectivity as it is known in the ESN literature) and d_0 describes the typical length of an added link. The minimum error achieved was 0.0055 for the 3D model (Figure 5.10(a)) and 0.0095 for the 2D model (Figure 5.10(b)). We again took the weights from an initial fully connected orthonormal matrix. We tried also the usual approach, getting the weights from a uniform distribution between -1 and 1 and then scaling the matrix by setting the spectral radius to 0.9, and then iterating through different values of the connectivity, but that gave absolutely no results. The network size used in both experiments are in agreement with a uniform lattice in 2D (15 by 15 = 225) and in 3D (6 by 6 by 6 = 216).



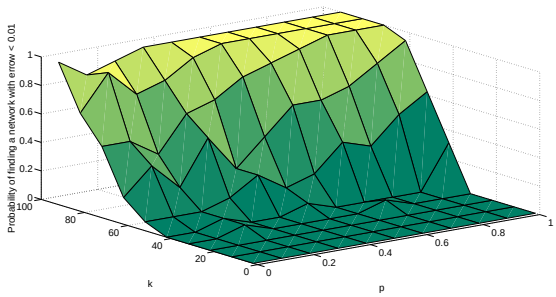
(a) Waxman model in 3D space on Mackey-Glass 17. Repetitions = 16. Network size = 216.

(b) Waxman model in 2D space on Mackey-Glass 17. Repetitions = 16. Network size = 225.

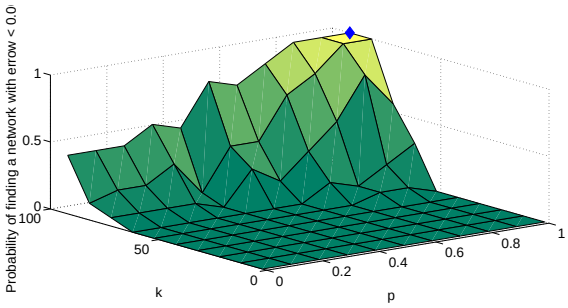
Figure 5.10: Plot showing the probability of finding a network constructed by the Waxman model which gives an error < 0.1 when dealing with Mackey-Glass 17.



(a) Probability that NRMSE < 0.1. Repetitions = 16. Network size = 1000.



(b) Probability that NRMSE < 0.01. Repetitions = 16. Network size = 1000.



(c) Probability that NRMSE < 0.001. Repetitions = 16. Network size = 1000.

Figure 5.11: Watts-Strogatz model on Mackey-Glass 17. Probability that a network has a NRMSE lower than C, where C = 0.1 (a), 0.01 (b), 0.001 (c).

5.5.4 The Watts-Strogatz Model

The Watts-Strogatz model [111] is a powerful network model which incorporates a spatial dependency as well as long range links. More specifically, the initial model is a regular periodic lattice, in which every node is connected to its k closest neighbors. Then, with probability p the already existent links are rewired randomly. We see that the solution space is relatively compact, if k is bigger, meaning each node has more neighbors to which it is connected, then a smaller p suffices for a similar performance with bigger p and smaller k. The minimum error achieved was 5.4×10^{-4}. So apparently (at least for network size = 400) after one certain value of k the performance went down, even if for a bit different values of p, it still went down, however it seemed that this model had big potential, given the NRMSE achieved and the shape of the solution space; and so we decided to investigate an even bigger size, 1000 neurons. We show in Figure 5.11 our results. The blue diamond represents the lowest NRMSE achieved: 2.9×10^{-5}. The previous best result was obtained by [54] and it was 6.3×10^{-5}. This is remarkable considering that no reservoir training takes place whatsoever, just the usual regression.

Table 5.1: Overview of results obtained in this chapter.

-	Min NRMSE on MG17	Min NRMSE on MG30	Size for MG17	Size for MG30
Random Optimization	10^{-2}	9×10^{-2}	30	400
Column Connectivity	5×10^{-3}	6×10^{-2}	100	400
PSO on a row	10^{-2}	-	100	-
PSO on a column	10^{-3}	-	100	-
Waxman model 2D	9×10^{-3}	-	225	-
Waxman model 3D	5×10^{-3}	-	216	-
Watts-Strogatz	2.9×10^{-5}	-	1000	-

5.6 Discussion

In this chapter we explored efficient methods of finding good ESNs for the Mackey-Glass (with $\tau = 17$ mostly) time-series. We can draw the conclusions that randomness usually helps in the searching process, that good enough networks can be found quite fast for some time-series and that methods which employ special topologies have some unique advantages: the construction process is very fast and if crossvalidation is used for the model parameters, one can be sure to find a good network between certain ranges of these parameters. We show in Table 5.1 an overview of the results obtained in this chapter. Many more models can be used for the echo state network, for a comprehensive overview the reader is refered to [4]. We have to mention that this approach to ESNs has just been started [31], and we expect many more developments to come that explore the relationship between signal statistics and network topology. Theoretical investigations could be of tremendous help here and should be of interest to physicists (the primary field of complex networks) and to theoretical computer scientists.
In this chapter we explored efficient methods of finding good ESNs for the Mackey-Glass (with $\tau = 17$ mostly) time-series. We can draw the conclusions that randomness usually helps in the searching process, that good enough networks can be found quite fast for some time-series and that methods which employ special topologies have some unique advantages: the construction process is very fast and if crossvalidation is used for the model parameters, one can be sure to find a good network between certain ranges of these parameters. We show in Table 5.1 an overview of the results obtained in this chapter. Many more models can be used for the echo state network, for a comprehensive overview the reader is refered to [4]. We have to mention that this approach to ESNs has just been started [31], and we expect many more developments to come that explore the relationship between signal statistics and network topology. Theoretical investigations could be of tremendous help here and should be of interest to physicists (the primary field of complex networks) and to theoretical computer scientists.

Chapter 6

Discussion

6.1 Summary

In Chapter 1 we briefly introduced the prediction problem and the ESN, the motivation for research and we gave the research questions we will try to answer. In Chapter 2 we described the ESN in detail, we gave its theoretical foundations and described some successful ESN methods for dealing with time-series. We saw there that understanding the inner working of the ESN can make use of Dynamical Systems Theory. In Chapter 3 we employed a linear ESN to deal with the MSO problem and achieved the highest level of performance in the literature with a huge factor better than the previous results. We then showed the connection with the recent field of Compressed Sensing. In Chapter 4 we investigated in detail the effect of the activation function on the general behavior of the ESN. We concluded that the mixed activation function acts like a regularization of the network and even though it does not improve the general performance, it has some desirable properties as we saw the lowest error achieved for Mackey-Glass with $\tau = 17$. Also in Chapter 4, we dealt with multiple datasets to show how different versions of the ESN deal with different datasets and we achieved the lowest error on Mackey-Glass with $\tau = 17$, again with a huge factor better then the previously best results found in the literature. In Chapter 5 we described efficient methods for finding a good ESN for some specific dataset. We also showed a few methods for constructing an ESN based on theory from the field of Complex Networks. We show in Table 6.1 an overview of the main results obtained.

Table 6.1: Overview of results obtained.

-	MSO5	MG $\tau = 17$	Sunspots	Laser	MG $\tau = 30$
Size	40	1000	30	200	400
Connectivity	0.5	1	1	1	1
Technique	Monte Carlo	Gaussian pert.	Monte Carlo	Gaussian pert.	Exp. pert.
Noise	No	No	No	No	No
Weight matrix	Orthonormal	Orthonormal	Random scaled	Orthonormal	Orthonormal
Activation fct	Linear	Mixed	Linear	Tanh	Mixed
Steps predicted	300	84	1	1	84
Min NRMSE	$1.02x10^{-11}$	$8.34x10^{-8}$	-	-	$6.8x10^{-3}$
Min NMSE	-	-	$2.02x10^{-4}$	$2.1x10^{-3}$	-

6.2 Conclusion

One of the main issues investigated in this thesis is the effect of different weight matrices for the echo state network. The connection with compressed sensing is most desirable as we can further draw theoretical and practical conclusions on the properties of the weight matrix, as well as on the number of steps needed for training. We also constructed weight matrices based on the topology of the network. The fact that we can draw from the field of complex networks as well, a much older field than reservoir computing, helps us in constructing sound hypotheses and then test these using the ESN paradigm. We also investigated the effect of the activation function of the ESN, showing that a linear activation works best for some datasets, while other need non-linearity to catch the dynamics of the time-series. The most important point, we believe, is the fact that we showed that the orthonormal weight matrix with a mixed activation function can be affected by randomness added to it in such a way that the ESN does not need noise to be able to catch the dynamics of the input signal and also can reach an unmatched level of performance. Even though we showed this only for the Mackey-Glass with $\tau = 17$, probably some modifications to our approach exist which would enhance the ESN for other time-series as well (see next section for possible extensions). We hypothesize that the randomness added to the weight matrix (the row replacement process) has the effect of a kind of structural perturbation to the initial network, this is why noise is not needed anymore. This is also due to the relative symmetry of the initial orthonormal matrix compared to the random scaled matrix. The mixed activation function acts like a regularization for the ESN, limiting the influence that the inner dynamics has on the overall behavior of the network. We showed that randomness can be used in various ways to reach a local optimum very fast, and that this is due to random projections, which have very desirable properties when dealing with unknown signals. The fact that the state space is so big (in general we use a high number of neurons with continuous values between -1 and 1) has of course major disadvantages, for example, it is very hard to find the global optimum, or to have a convergence guarantee, but it has also advantages, i.e. the function has many local optima, which can be found relatively fast using randomness in many ways, and also due to the high level of connectivity of the network. Because if the network is highly connected, modifying few values of the weight matrix has a major effect on the overall behavior of the network, due to the high level of interdependency between neurons. We reached unprecedented levels of performance for many datasets generally used in the ESN paradigm, namely MSO2, MSO3, MSO4, MSO5, MSO6, MSO7, MSO8, Mackey-Glass with $\tau = 17$ and the sunspots time-series. However further theoretical investigations need to be done to have a deeper understanding of the dynamics of the ESN and its applications.

The research questions mentioned in Chapter 1 can be answered in the following ways:

1. Can we construct some ESNs to minimize the variability between different initializations?

Yes, by using specific topologies and investigating the parameters through cross-validation, or by using non-linear read-outs the variability can be lowered to a minimum. The magnitude of the noise, if big enough, can also reduce the variability of the ESN to a sufficient degree. Perturbation mechanisms in general help stabilize the network.

2. Can we find (time) efficient methods for finding a good ESN for some specific dataset?

Yes. Many methods employing randomness can be used to find a local optimum very fast. Also constructing matrices based on local rules (complex networks) generates (very fast) networks with sufficient precision.

3. Is noise critical for finding good ESNs (as is previously suggested in the literature)? No. By employing other perturbation methods, noise can be eliminated from the activation function completely also reaching an unprecedented level of performance.

6.3 Future Research

With respect to the random perturbations added to the network (section 4.4), first of all, theoretical investigations need to be done, to show analytically what are the performance levels which can be reached and also the limitations involved in such an approach. For some other datasets, some different type of perturbation might do the trick, like exponential / logarithmic. It would be very interesting to see how different magnitudes and distributions of the perturbation can affect the overall functioning of the ESN when dealing with some specific dataset. We would approach this by using control theory for non-autonomous (input driven) dynamical system. Controlling dynamical systems has long been an endeavor of physicists and mathematicians, however for non-linear non-autonomous (i.e. input driven) systems, little theory exists (according to [27]). Extending the theory of linear-systems or of autonomous systems could prove worthwhile and some steps have been taken in this direction [27]. But for this, a deeper understanding of how memory is stored in ESNs is needed. We saw that the dynamical system needs to have some specific attractors for memory capacity. But some questions remain: what type of attractors ? how many ? how can we count and modify different attractors of a ESN ? Some theory exists also in this area, but again is from different fields (theoretical biology, physics) [2, 60]. From our perspective, the ultimate question is: what characteristics of the input signal should we extract and how to embed these in the process of constructing an echo state network tailored to this specific signal ?

When strictly considering the topology of the ESN, one can think of many extensions to section 5.5 There are many complex network models and many types of measures for characterizing the properties of such networks, like the clustering coefficient, the centrality of nodes, the average path between any two neurons, etc. Such measure might be informative, even critical for some behaviors of the ESN, while others might be irrelevant. To understand this dependency many experiments need to be performed, with different topologies and different datasets.

When considering the connection between ESNs and CS we see that we can use theory from CS to construct ESN weight matrices. More methods from CS can be employed here and adapted to the ESN paradigm, like for example adaptive dictionary construction [63], that is construction of the weight matrix based on the input statistics.

Bibliography

[1] Baddeley, R., Abbott, L. F., Booth, M. C., Sengpiel, F., Freeman, T., Wakeman, E. A., and Rolls, E. T. Responses of neurons in primary and inferior temporal visual cortices to natural scenes. *Proceedings of the Royal Society of London. Series B: Biological Sciences 264*, 1389 (1997), 1775–1783.

[2] Bagley, R., and Glass, L. Counting and classifying attractors in high dimensional dynamical systems. *Journal of Theoretical Biology 183*, 3 (1996), 269–284.

[3] Baraniuk, R., Davenport, M., DeVore, R., and Wakin, M. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation 28*, 3 (2008), 253–263.

[4] Barthélemy, M. Spatial networks. *Physics Reports 499*, 1 (2011), 1–101.

[5] Baum, L. E., and Petrie, T. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics 37*, 6 (1966), 1554–1563.

[6] Bell, A. J., and Sejnowski, T. J. An information-maximization approach to blind separation and blind deconvolution. *Neural computation 7*, 6 (1995), 1129–1159.

[7] Bellman, R. A markovian decision process. *J. Math. Mech. 6* (1957), 679–684.

[8] Bertschinger, N., and Natschläger, T. Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation 16*, 7 (2004), 1413–1436.

[9] Blum, A. Random projection, margins, kernels, and feature-selection. *Subspace, Latent Structure and Feature Selection* (2006), 52–68.

[10] Bollerslev, T. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics 31*, 3 (1986), 307–327.

[11] Box, G. E., and Jenkins, G. M. Time series analysis forecasting and control. Tech. rep., DTIC Document, 1970.

[12] Boyd, S., and Vandenberghe, L. *Convex optimization.* Cambridge university press, 2009.

[13] Brunel, N., Hakim, V., Isope, P., Nadal, J.-P., and Barbour, B. Optimal information storage and the distribution of synaptic weights: perceptron versus purkinje cell. *Neuron 43*, 5 (2004), 745–757.

[14] Buehner, M., and Young, P. A tighter bound for the echo state property. *Neural Networks, IEEE Transactions on 17*, 3 (2006), 820–824.

[15] Candes, E., and Romberg, J. Sparsity and incoherence in compressive sampling. *Inverse problems 23*, 3 (2007), 969.

[16] Candès, E. J. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique 346*, 9 (2008), 589–592.

[17] Candes, E. J., and Plan, Y. Matrix completion with noise. *Proceedings of the IEEE 98*, 6 (2010), 925–936.

[18] Candes, E. J., Romberg, J. K., and Tao, T. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics 59*, 8 (2006), 1207–1223.

[19] Candes, E. J., and Tao, T. Decoding by linear programming. *Information Theory, IEEE Transactions on 51*, 12 (2005), 4203–4215.

[20] Candes, E. J., and Tao, T. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on 52*, 12 (2006), 5406–5425.

[21] Candès, E. J., and Wakin, M. B. An introduction to compressive sampling. *Signal Processing Magazine, IEEE 25*, 2 (2008), 21–30.

[22] Casazza, P. G., and Kutyniok, G. *Finite frames.* Springer, 2012.

[23] Chen, S., Cowan, C., and Grant, P. Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, IEEE Transactions on 2*, 2 (1991), 302–309.

[24] Christensen, O. *An introduction to frames and Riesz bases.* Springer, 2002.

[25] Chui, C. K. *An introduction to wavelets*, vol. 1. Academic press, 1992.

[26] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[27] Dambre, J., Verstraeten, D., Schrauwen, B., and Massar, S. Information processing capacity of dynamical systems. *Scientific reports 2* (2012).

[28] Dasgupta, S., and Gupta, A. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms 22*, 1 (2003), 60–65.

[29] Davenport, M. A., Boufounos, P. T., Wakin, M. B., and Baraniuk, R. G. Signal processing with compressive measurements. *Selected Topics in Signal Processing, IEEE Journal of 4*, 2 (2010), 445–460.

[30] Davenport, M. A., Duarte, M. F., Eldar, Y. C., and Kutyniok, G. Introduction to compressed sensing. In *Eldar, Y.C., Kutyniok, G. (Eds.), Compressed Sensing: Theory and Applications, Cambridge University Press (2011).*

[31] Deng, Z., and Zhang, Y. Complex systems modeling using scale-free highly-clustered echo state network. IEEE, pp. 3128–3135.

[32] DeVore, R. A. Nonlinear approximation. *Acta numerica 7* (1998), 51–150.

[33] Donoho, D. L. De-noising by soft-thresholding. *Information Theory, IEEE Transactions on 41*, 3 (1995), 613–627.

[34] Donoho, D. L. Compressed sensing. *Information Theory, IEEE Transactions on 52*, 4 (2006), 1289–1306.

[35] Donoho, D. L., Maleki, A., and Montanari, A. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences 106*, 45 (2009), 18914–18919.

[36] Duarte, M. F., Wakin, M. B., Baron, D., and Baraniuk, R. G. Universal distributed sensing via random projections. *Proceedings of the 5th international conference on Information processing in sensor networks* (2006), 177–185.

[37] Eddy, S. R. Hidden Markov models. *Current opinion in structural biology 6*, 3 (1996), 361–365.

[38] Elman, J. L. Finding structure in time. *Cognitive science 14*, 2 (1990), 179–211.

[39] Engle, R. F. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society* (1982), 987–1007.

[40] Ganguli, S., Huh, D., and Sompolinsky, H. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences 105*, 48 (2008), 18970–18975.

[41] Ganguli, S., and Sompolinsky, H. Short-term memory in neuronal networks through dynamical compressed sensing. *Advances in neural information processing systems* (2010), 667–675.

[42] Ganguli, S., and Sompolinsky, H. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual review of neuroscience 35* (2012), 485–508.

[43] Ghahramani, Z. Learning dynamic bayesian networks. *Adaptive processing of sequences and data structures* (1998), 168–197.

[44] Hammer, B., and Steil, J. J. Tutorial: Perspectives on learning with RNNs. 357–368.

[45] Hinton, G. E., and Ghahramani, Z. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences 352*, 1358 (1997), 1177–1190.

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

[47] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[48] Holzmann, G. Echo state networks with filter neurons and a delay&sum readout with applications in audio signal processing. *Unpublished Master's thesis, Graz University of Technology* (2008).

[49] Holzmann, G., and Hauser, H. Echo state networks with filter neurons and a delay&sum readout. *Neural Networks 23*, 2 (2010), 244–256.

[50] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences 79*, 8 (1982), 2554–2558.

[51] Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148* (2001).

[52] Jaeger, H. Short term memory in echo state networks. Tech. rep., 2001.

[53] Jaeger, H. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. Tech. rep., 2002.

[54] Jaeger, H., and Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science 304*, 5667 (2004), 78–80.

[55] Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks 20*, 3 (2007), 335–352.

[56] Jarvis, S., Rotter, S., and Egert, U. Extending stability through hierarchical clusters in echo state networks. *Frontiers in neuroinformatics 4* (2010).

[57] Jordan, M. I. Attractor dynamics and parallellism in a connectionist sequential machine.

[58] Kennedy, J., Eberhart, R., et al. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (1995), vol. 4, Perth, Australia, pp. 1942–1948.

[59] Klampfl, S., and Maass, W. A theoretical basis for emergent pattern discrimination in neural systems through slow feature extraction. *Neural computation 22*, 12 (2010), 2979–3035.

[60] Klemm, K., and Bornholdt, S. Stable and unstable attractors in boolean networks. *Physical Review E 72*, 5 (2005), 055101.

[61] Koltchinskii, V., and Yuan, M. Sparsity in multiple kernel learning. *The Annals of Statistics 38*, 6 (2010), 3660–3695.

[62] Koryakin, D., Lohmann, J., and Butz, M. V. Balanced echo state networks. *Neural Networks* (2012).

[63] Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Kjersti, E., Lee, T.-W., and Sejnowski, T. J. Dictionary learning algorithms for sparse representation. *Neural computation 15*, 2 (2003), 349–396.

[64] Ku, C.-C., and Lee, K. Y. Diagonal recurrent neural networks for dynamic systems control. *Neural Networks, IEEE Transactions on 6*, 1 (1995), 144–156.

[65] Lee, C.-H., and Teng, C.-C. Identification and control of dynamic systems using recurrent fuzzy neural networks. *Fuzzy Systems, IEEE Transactions on 8*, 4 (2000), 349–366.

[66] Legenstein, R., Pecevski, D., and Maass, W. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology 4*, 10 (2008), e1000180.

[67] Lukoševičius, M. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 659–686.

[68] Lukoševičius, M., and Jaeger, H. Survey: Reservoir computing approaches to recurrent neural network training. *Computer Science Review 3*, 3 (2009), 127–149.

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

[70] Maciel, L., Gomide, F., Santos, D., and Ballini, R. Exchange rate forecasting using echo state networks for trading strategies. *Submitted to Computational Intelligence for Financial Engineering (CIFEr), 2014*.

[71] Makridakis, S., and Hibon, M. Arma models and the Box-Jenkins methodology. *Journal of Forecasting 16*, 3 (1997), 147–163.

[72] Mallat, S. *A wavelet tour of signal processing*. Academic press, 1999.

[73] Matyas, J. Random optimization. *Automation and Remote Control 26*, 2 (1965), 246–253.

[74] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[75] Murphy, K. P. Dynamic bayesian networks: Representation, inference and learning. *PhD thesis* (2002).

[76] Ozturk, M. C., Xu, D., and Príncipe, J. C. Analysis and design of echo state networks. *Neural Computation 19*, 1 (2007), 111–138.

[77] Pape, L., Ruessink, B. G., Wiering, M. A., and Turner, I. L. Recurrent neural network modeling of nearshore sandbar behavior. *Neural Networks 20*, 4 (2007), 509–518.

[78] Pedersen, M. E. H. Good parameters for particle swarm optimization. *Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001* (2010).

[79] Pennebaker, W. B., and Mitchell, J. L. *JPEG: Still image data compression standard*. Springer, 1993.

[80] Pineda, F. J. Generalization of back-propagation to recurrent neural networks. *Physical review letters 59*, 19 (1987), 2229–2232.

[81] Principe, J. C., Rathie, A., and Kuo, J.-M. Prediction of chaotic time series with neural networks and the issue of dynamic modeling. *International Journal of Bifurcation and Chaos 2*, 04 (1992), 989–996.

[82] Rahimi, A., and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems* (2007), 1177–1184.

[83] Rasmussen, P. M., Hansen, L. K., Madsen, K. H., Churchill, N. W., and Strother, S. C. Model sparsity and brain pattern interpretation of classification models in neuroimaging. *Pattern Recognition 45*, 6 (2012), 2085–2100.

[84] Rauhut, H. Compressive sensing and structured random matrices. *Theoretical foundations and numerical methods for sparse recovery 9* (2010), 1–92.

[85] Rodan, A., and Tino, P. Minimum complexity echo state network. *Neural Networks, IEEE Transactions on 22*, 1 (2011), 131–144.

[86] Roeschies, B., and Igel, C. Structure optimization of reservoir networks. *Logic Journal of IGPL 18*, 5 (2010), 635–669.

[87] Rubinstein, R., Bruckstein, A. M., and Elad, M. Dictionaries for sparse representation modeling. *Proceedings of the IEEE 98*, 6 (2010), 1045–1057.

[88] Rummelhart, D., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature 323*, 9 (1986), 533–535.

[89] Schmidhuber, J., Wierstra, D., and Gomez, F. Evolino: Hybrid neuroevolution/optimal linear search for sequence prediction. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)* (2005).

[90] Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J. J., and Stroobandt, D. Improving reservoirs using intrinsic plasticity. *Neurocomputing 71*, 7 (2008), 1159–1171.

[91] Shi, Z., and Han, M. Support vector echo-state machine for chaotic time-series prediction. *Neural Networks, IEEE Transactions on 18*, 2 (2007), 359–372.

[92] Siewert, U., and Wustlich, W. Echo-state networks with band-pass neurons: Towards generic time-scale-independent reservoir structures. *Internal status report, PLANET intelligent systems GmbH* (2007).

[93] Smith, S. W., et al. *The scientist and engineer's guide to digital signal processing.* 2nd ed. California Technical Publishing, 1997.

[94] Smolensky, P. Information processing in dynamical systems: Foundations of harmony theory. Tech. rep., 1986.

[95] Sondik, E. J. The optimal control of partially observable Markov processes. Tech. rep., DTIC Document, 1971.

[96] Song, Q., and Feng, Z. Effects of connectivity structure of complex echo state network on its prediction performance for nonlinear time series. *Neurocomputing 73*, 10 (2010), 2177–2185.

[97] Steil, J. J. Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning. *Neural Networks 20*, 3 (2007), 353–364.

[98] Stemmler, M., and Koch, C. How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate. *Nature neuroscience 2*, 6 (1999), 521–527.

[99] Sussillo, D., and Abbott, L. F. Generating coherent patterns of activity from chaotic neural networks. *Neuron 63*, 4 (2009), 544–557.

[100] Sussillo, D., and Barak, O. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation 25*, 3 (2013), 626–649.

[101] Tao, T. *Structure and Randomness: pages from year one of a mathematical blog.* American Mathematical Soc., 2008.

[102] Taubman, D., and Marcellin, M. Jpeg2000: Image compression fundamentals, practice and standards. *Massachusetts: Kluwer Academic Publishers* (2002), 255–258.

[103] Triesch, J. A gradient rule for the plasticity of a neuron's intrinsic excitability. Springer, 2005, pp. 65–70.

[104] Tsymbal, A. The problem of concept drift: definitions and related work. Tech. rep., 2004.

[105] van der Zant, T., Becanovic, V., Ishii, K., Kobialka, H.-U., and Plöger, P. Finding good echo state networks to control an underwater robot using evolutionary computations. In *Proceedings of the 5th IFAC symposium on Intelligent Autonomous Vehicles (IAV04)* (2004).

[106] Vapnik, V. *The nature of statistical learning theory.* springer, 2000.

[107] Verstraeten, D., Dambre, J., Dutoit, X., and Schrauwen, B. Memory versus non-linearity in reservoirs. *Neural Networks (IJCNN), The 2010 International Joint Conference on* (2010), 1–8.

[108] Verstraeten, D., Schrauwen, B., d'Haene, M., and Stroobandt, D. An experimental unification of reservoir computing methods. *Neural Networks 20*, 3 (2007), 391–403.

[109] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on 37*, 3 (1989), 328–339.

[110] Wainwright, M. J., Simoncelli, E. P., and Willsky, A. S. Random cascades on wavelet trees and their use in analyzing and modeling natural images. *Applied and Computational Harmonic Analysis 11*, 1 (2001), 89–123.

[111] Watts, D. J., and Strogatz, S. H. Collective dynamics of "small-world" networks. *Nature 393*, 6684 (1998), 440–442.

[112] Waxman, B. M. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on 6*, 9 (1988), 1617–1622.

[113] White, O. L., Lee, D. D., and Sompolinsky, H. Short-term memory in orthogonal neural networks. *arXiv preprint cond-mat/0402452* (2004).

[114] Whitle, P. *Hypothesis testing in time series analysis*, vol. 4. Almqvist & Wiksells, 1951.

[115] Xue, Y., Yang, L., and Haykin, S. Decoupled echo state networks with lateral inhibition. *Neural Networks 20*, 3 (2007), 365–376.

[116] Yap, H. L., Charles, A. S., and Rozell, C. J. The restricted isometry property for echo state networks with applications to sequence memory capacity. *Statistical Signal Processing Workshop (SSP), 2012 IEEE* (2012), 580–583.

[117] Yildiz, I. B., Jaeger, H., and Kiebel, S. J. Re-visiting the echo state property. *Neural Networks 35* (2012), 1–9.