Universitat de les
Illes Balears

Reservoir computing based on delay-dynamical systems

Lennert Appeltant

Promoters: prof. dr. Jan Danckaert, prof. dr. Ingo Fischer,
dr. ir. Guy Van der Sande

Report writer: prof. dr. Claudio R. Mirasso

Joint PhD
Vrije Universiteit Brussel
Universitat de les Illes Balears
May 2012

Reservoir Computing based on Delay-dynamical Systems
PhD thesis by Lennert Appeltant
E-mail: Lennert.Appeltant@vub.ac.be

Vrije Universiteit Brussel
Pleinlaan 2
B-1050 Brussel
Belgium

Instituto de Física Interdiscplinar y Sistemas Complejos IFISC (UIB-CSIC)
Campus Universitat de les Illes Balears
E-07122 Palma de Mallorca
Spain

Proefschrift ingediend tot het behalen van de academische graad van Doctor in de Ingenieurswetenschappen
Thesis submitted in partial fulfilment of the requirements for the academic degree of Doctor in Engineering Sciences
Tesi presentada al Departament de Física de la Universitat de les Illes Balears per optar al grau de Doctor en Física

Promoters: prof. dr. Jan Danckaert, prof. dr. Ingo Fischer, dr. ir. Guy Van der Sande Jury members: prof. dr. ir. J. Tiberghien (chairman / Vrije Universiteit Brussel), prof. dr. ir. R. Pintelon (vice-chairman / Vrije Universiteit Brussel), prof. dr. A. Nowé (secretary / Vrije Universiteit Brussel), prof. dr. P. Colet (Universitat de les Illes Balears), prof. dr. L. Pesquera (Universidad de Cantabria), prof. dr. S. Massar (Université Libre de Bruxelles), prof. dr. ir. J. Van Campenhout (Universiteit Gent)

Print: Silhouet, Maldegem

2012 Uitgeverij VUBPRESS Brussels University Press
VUBPRESS is an imprint of ASP nv (Academic and Scientific Publishers nv)
Ravensteingalerij 28
B-1000 Brussels
Tel. +32 (0)2 289 26 50
Fax +32 (0)2 289 26 59
E-mail: info@vubpress.be
www.vubpress.be

ISBN 978 90 5718 120 7
NUR 925 / 926 / 928
Legal deposit D/2012/11.161/056

Acknowledgements

Writing an acknowledgement... How can I fulfill this job without forgetting someone? So many interested people asked me how my PhD was progressing. It was not always easy to explain that once again the simulation did not work or that everything I had been claiming the last months was not entirely correct because I had forgotten a square root somewhere... My PhD became a project in which so many people were involved that I am doomed to forget to mention someone. I could not have succeeded without the help of so many, but I would like to give some special, personal thanks.

First, I want to express my gratitude to prof. Tiberghien, prof. Pintelon, prof. Nowé, prof. Massar, prof. Colet, prof. Pesquera, and prof. Van Campenhout for accepting to be part of my jury. Next, I want to explicitly thank my promotors: prof. Jan Danckaert, prof. Ingo Fischer and dr. Guy Van der Sande.

Jan, bedankt voor de kans die je mij gegeven hebt om te werken op een onderwerp waarvan de wetenschappelijke output een lange tijd zeer hypothetisch is geweest. Ik ben er me ten volle van bewust dat het feit dat ik zolang kon doorwerken op dezelfde topic zonder enige vorm van publicatie te danken is aan het feit dat jij het voor mij hebt opgenomen. Bedankt voor het geloven in mij en mijn onderwerp, bedankt voor de fantastische werkomgeving die je creëert voor al je doctoraatsstudenten. Dat alles in een aangename, ongedwongen sfeer verliep, stimuleerde mij om te durven vragen, te zoeken, te proberen. Kortom, alle ingrediënten die nodig waren om positief werk te leveren waren in ruime mate aanwezig.

Ingo, many thanks for the scientific support, for the continuous interest in what I was doing and of course for the wonderful time in Palma. I thank you for putting your scientific shoulders under my research and for giving me the opportunity to spend part of my PhD at IFISC in the Universitat de les Illes Balears. I can only be grateful for the fact that someone with your

scientific experience wanted to accept a project with someone who had not proven anything yet. I will never forget the fruitful collaboration at work nor the enjoyment of the wonderful island of Mallorca after working hours.

Guy, mijn grootste dankbetuiging gaat naar jou. Het is nu ongeveer 7 jaar geleden dat ik van jou 'vaste stof en stralingsfysica' kreeg. Toen reeds moedigde je me aan om voor fotonica te kiezen. Je was een fantastische thesisbegeleider en een uitmuntende doctoraatsbegeleider. Jouw betrokkenheid bij dit project was intens voelbaar voor elke invalshoek van dit project. We hebben bijna elke weekdag van de laatste 4 jaar samen gewerkt. Of het nu ging om het uitdenken van een strategie, om te debuggen, om rapporten te schrijven of om mijn laptop te herinstalleren, ik kon steeds voor alles bij jou terecht. Jouw constructieve opmerkingen, jouw inzicht maar ook je richtlijnen op het vlak van communicatie zijn de wortels van het succes van deze studie.

Irina, ik had als jonge student het geluk mee door jou opgeleid te worden. Jouw enthousiasmerende werking is legendarisch. Het pumpernickel-virus heeft zijn werk gedaan en kan nooit meer uitgeroeid worden. Oprechte dank hiervoor.

One of the things I especially enjoyed about my PhD was the wide international collaboration. Thanks to prof. Mirasso, prof. Pesquera, prof. Larger, prof. Schrauwen and prof. Dambre for inviting me to their labs. It has been a pleasure to work with you. Our scientific discussions, your help, your encouragements, my deepest respect for that. I appreciated a lot that I, as a young beginner, was allowed to have a voice in your team of experienced scientists. Many thanks to all the people I met on my trips: Miguel, Daniel, Miguel-Angel, Konstantin, Jade, Xavi, Neus, Silvia, Clara, Ana, Romain, Sergei, Yanne, Maxime. Thanks for helping me getting through the ups and downs of reservoir computing.

Working on 'the 9th floor' has been very valuable on a professional level, but I especially appreciated the relaxed atmosphere. I thank you all for your enthusiasm, your support and for the wonderful years. The renovations have provided proof that working at TONA/APHY is great because of the people. It did not matter whether our office was on the 9th, somewhere in a student lab or even in the restaurant, the atmosphere was unforgettable. My deepest gratitude goes to Otti, Gordon, Lendert, Lydia, Vincent, Werner, Lilia, Sifeu, Modeste, Mulham, Guy I and Guy II, Stefano, Philippe, Lars, Nicky, Stijn, Stefaan, Pierre, Tom, Diane, Jana and Amani for their true friendship. Thank you for the scientifically completely irrelevant discussions. It provided a nice counterpart for all the simulation results that should have appeared on my computer screen, but did not.

Maybe also a small word of thanks to my own lap top. It needs to be said: it led a life of it's own and too early it decided to abruptly end our relationship...

A PhD comes with side-effects as well. Dragging around didactic material made me meet Lucia in the elevator. Since that moment I come to the VUB not only for pedagogical or scientific motives...

Niemand haalt de eindmeet zonder supporters. Het belang van een warm nest kan niet overschat worden. Welke richting het op wetenschappelijk vlak ook uitging, mijn trouwste fans, mijn ouders, mijn zus, Kevin, meter en de Staessens clan in Eppegem bleven op post. Of het nu ging om logistieke ondersteuning, even stoom afblazen of skype momenten vanuit Palma of conferenties, het deed deugd en het hielp me weer verder op weg. Jullie hebben me steeds gemotiveerd om het beste uit mezelf te halen.

Zovele jaren op de VUB leverde nog zoveel meer contacten op. Zowel in het restaurant, de voetbalcompetetie, het fitnesscentrum, het Kultuur Kaffee, de taallessen, de managementcursussen als in de Opinio kwam ik telkens weer soulmates tegen die zin hadden om de dagelijkse kommer door te spoelen... met een sprite. Aan allen, uitgesproken dank voor de memorabele momenten.

Thank you all for being persistent and encouraging, for believing in me, and for the many precious memories along the PhD journey.

Summary

Walking down a street, we are constantly bombarded with sensory impressions. Seeing a vehicle or a familiar face, hearing the ongoing traffic and conversations, smelling the food stalls ... All these external impulses instantly produce massive neural activity in our brain, so that we recognize the passing bus, a good friend or a car horn, or that the smell of freshly baked waffles makes us hungry. We can see a blurry photo and still recognize the scene in a fraction of a second, a task for which a computer takes minutes or even hours. Today, except for mathematical operations, our brain functions much faster and more efficient than any supercomputer. It is precisely this form of information processing in neural networks that inspires researchers to create systems that mimic the brain's information processing capabilities, in a way radically different from current computer based schemes. In this thesis we propose a novel approach to implement these alternative computer architectures, based on delayed feedback.

Time delays are intrinsic in many real systems. In engineering, time delays often arise in feedback loops involving sensors and actuators. In photonic systems, time delayed feedback plays an important role and arises due to unwanted external reflections. On the one hand, time delays tend to destabilize systems such as lasers, but, on the other hand, the chaotic output from e.g. a laser with feedback can put into use e.g. in chaotic communication systems. In general, systems subject to time-delayed feedback present a rich variety of dynamical regimes. We propose to exploit the rich dynamics of delayed feedback systems for information processing by using the system's transient response to an external input. We show that one single nonlinear node with delayed feedback can replace a large network of nonlinear nodes. Our results demonstrate that this new information processing architecture performs well in a variety of tasks, such as e.g. time series prediction and speech recognition.

We investigate whether applying this simple architecture in electronic, opto-electronic or photonics systems could potentially be more resource-efficient as hundreds or even thousands of artificial neurons could be replaced by only one single hardware node in combination with a delay line. Moreover, the fact that delay is easily implementable, sometimes even unavoidable, in photonic systems may lead to the implementation of ultra-fast all-optical computational units. First we numerically investigate the architecture and performance of delayed feedback systems as information processing units. Then we elaborate on electronic and opto-electronic implementations of the concept. Next to evaluating their performance for standard benchmarks, we also study task independent properties of the system, extracting information on how to further improve the initial scheme. Finally, some simple modifications are suggested, yielding improvements in terms of speed or performance.

Samenvatting

Wandelend door de straat word je aanhoudend gebombardeerd met indrukken uit je omgeving. Het waarnemen van een voertuig, een bekend gezicht, het horen van het verkeer en gesprekken van anderen, het ruiken van het voedsel in de eetstalletjes... Al die externe impulsen produceren instantaan een indrukwekkende neurale activiteit in onze hersenen, om op die manier de voorbijrijdende bus te herkennen, een goede vriend of een auto en om hongerig te worden door het ruiken van de geur van versgebakken wafels. We kunnen in een wazige foto vaak toch nog herkennen wat er op afgebeeld staat en dat in een fractie van een seconde, een taak die voor een computer verschillende minuten of misschien zelfs uren in beslag zou nemen, Op de dag van vandaag, behalve voor zuiver wiskundige operaties, functioneert ons brein veel sneller en vooral veel efficiënter dan eender welke supercomputer. Het is juist die vorm van informatieverwerking in neurale netwerken die onderzoekers inspireert om systemen te creëren die het brein nabootsen op het gebied van informatieverwerking, op een manier die radicaal verschillend is van de aanpak van de huidige computerarchitecturen. In deze thesis stellen we een nieuwe aanpak voor om deze alternatieve computerachitecturen te implementeren, gebaseerd op vertraagde koppelingen.

Tijdsvertragingen zijn intrinsiek aanwezig in een grote verscheidenheid aan systemen. In de toegepaste wetenschappen zijn vertragingen meestal afkomstig van terugkoppelingen van sensoren en actuatoren. Ook in fotonische systemen speelt tijdsvertraging een belangrijke rol en kan ze veroorzaakt worden door ongewilde reflecties. Terwijl aan de ene kant tijdsvertragingen een destabilizatie van een systeem zoals een laser kunnen teweeg brengen, kunnen ze aan de andere kant voordelig aangewend worden voor bv. chaos communicatie d.m.v. chaotische lasers met terugkoppeling. In het algemeen vertonen systemen die onderworpen zijn aan een tijdsvertraging een rijke varieteit aan dynamische regimes. We stellen voor om gebruik te maken van die rijke dynamica voor informatieverwerking door het transiente antwoord van

zo een systeem op een externe input aan te wenden. We demonstreren dat een enkele niet-lineaire node met vertraagde terugkoppeling een heel netwerk van niet-lineare nodes kan vervangen. Onze resultaten tonen aan dat deze nieuwe architectuur voor informatieverwerking goed presteert voor verschillende taken, zoals bv. het voorspellen van tijdreeksen en spraakherkenning.

We onderzoeken of het toepassen van deze simpele architectuur in elektronica, opto-elektronica of fotonische system potentieel efficiënter zou zijn op gebied van implementatie, aangezien honderden of zelfs duizenden artificiële neuronen vervangen kunnen worden door één fysiek aanwezige node met een vertraagde terugkoppeling. Daarenboven kan het feit dat vertraging gemakkelijk implementeerbaar is, soms zelfs onvermijdelijk, in fotonische systemen leiden tot een implementatie van ultra-snelle fotonische verwerkingseenheden. Allereerst bestuderen we de architectuur en de performantie van met vertraging gekoppelde systemen als informatieverwerkende componenten d.m.v. numerieke simulaties Daarna gaan we dieper in op een elektronsiche en een opto-elektronische implementatie van het concept. Naast het evalueren van de performantie op standaardtaken bekijken we ook taakonafhankelijke eigenschappen van het systeem en trachten we daaruit suggesties af te leiden voor het verbeteren van die initiële configuratie. Uiteindelijk stellen we enkele simpele modificaties voor die het systeem kunnen verbeteren op het gebied van snelheid en performantie.

Resumen

Cuando caminamos por la calle nos bombardean constantemente multiples impresiones sensoriales. Ver un vehículo o un rostro familiar, oír el tráfico o las conversaciones, oler la comida en los puestos,... Todos estos impulsos externos producen al instante una masiva actividad neuronal en nuestro cerebro para que así podamos reconocer el autobús que pasa, un buen amigo o el claxon de un coche, o que el olor de los gofres recién hechos nos de hambre. También podemos ver una foto desenfocada e immediamente reconocer la escena en una fracción de segundo, una tarea para la cual un ordenador tarda minutos o incluso horas. Hoy en día, excepto para operaciones matemáticas, nuestro cerebro funciona mucho más rápido y eficientemente que cualquier supercomputador. Es precisamente este tipo de procesamiento de la información en redes neuronales el que inspira a investigadores a crear sistemas que mimeticen las capacidades cerebrales de procesamiento, de manera totalmente diferente a los actuales esquemas basados en ordenadores. En esta tesis proponemos un nuevo enfoque para implementar estas arquitecturas de ordenador alternativas, basandonos en sistema con retroalimentación retrasada.

La aparición de un retraso temporal es intrínseca a muchos sistemas reales. En ingeniería, este retraso temporal surge generalmente en lazos de retroalimentación que implican sensores y accionadores. En sistemas fotónicos, la retroalimentación retrasada temporalmente juega un papel importante y surge debido a reflexiones externas indeseadas. Por una parte, los tiempos de retraso tienden a desestabilizar sistemas con láseres y pueden llegar a ser un problema. Por otra parte, la emisión caótica de un láser con retroalimentación, puede ser utilizada en sistemas de comunicaciones caóticas. En general, los sistemas sujetos a retroalimentación retrasada temporalmente presentan una amplia variedad de regímenes dinámicos. Proponemos explotar esta dinámica de los sistemas de retroalimentación retrasada para el procesamiento de información utilizando la respuesta transitoria del sistema

a una entrada externa. Mostramos que un simple nodo no-lineal con retroalimentación retrasada puede reemplazar una red con un gran número de nodos no-lineales. Nuestros resultados demuestran que esta nueva arquitectura de procesamiento tiene un buen rendimiento en una variedad de tareas, como por ejemplo, predicción de series temporales y reconocimiento del habla.

Investigamos si la aplicación de esta simple arquitectura en sistemas electrónicos, opto-electrónicos o fotónicos podría ser potencialmente más eficiente en términos de puesta en práctica, ya que cientos o incluso miles de neuronas artificiales podrían ser reemplazadas por una realización física con único nodo no-lineal y una línea de retraso. Además, el hecho de que el retraso es fácilmente implementable en sistemas fotónicos, incluso inevitable en ciertas ocasiones, puede llevar a implementar unidades computacionales ultrarápidas completemante fotónicas. Primero investigamos numéricamente la arquitectura y rendimiento de los sistemas con retroalimentación retrasada como unidades de procesamiento de información. A continuación evaluamos implementaciones electrónicas y opto- electrónicas del concepto. Después de evaluar el rendimiento para tareas estándares, también estudiamos propiedades del sistema independientes de la tarea, extrayendo información de cómo mejorar el esquema inicial. Finalemente sugerimos algunas simples modificaciones revelando mejoras en términos de velocidad o rendimiento.

Publications

Journal Publications

Lennert Appeltant, Miguel Cornelles Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, Joni Dambre, Benjamin Schrauwen, Claudio Mirasso, Ingo Fischer. **Information processing using a single dynamical node as complex system.** *Nat. Commun.* 2:468 doi: 10.1038/ncomms1476 (2011).

Laurent Larger, Miguel C. Soriano, Daniel Brunner, Lennert Appeltant, Jose M. Gutierrez, Luis Pesquera, Claudio R. Mirasso, and Ingo Fischer. **Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing.** *Opt. Express*, 20/3: 3241-3249, (2012) .

Conference proceedings

Romain Martinenghi, Sergei Rybalko, Lennert Appeltant, Guy Van der Sande, Jan Danckaert, Maxime Jacquot, Yanne Chembo, Laurent Larger. **Dynamique integro-differentielle en longueur d'onde optique, a retards multiples, pour le "Reservoir Computing".** 14e Rencontre du Non-lineaire Paris, Paris, France (2011).

Lennert Appeltant, Guy Van der Sande, Sergei Rybalko, Romain Martinenghi, Maxime Jacquot, Yanne Chembo, Laurent Larger, Ingo Fischer, Jan Danckaert. **Computational performance of a single bandpass electro-optic delay oscillator.** European Conference on Lasers and Electro-Optics

and the XIIth European Quantum Electronics Conference, Munich, Germany (2011).

Romain Martinenghi, Lennert Appeltant, Sergei Rybalko, Guy Van der Sande, Jan Danckaert, Maxime Jacquot, Yanne Chembo, Laurent Larger. **Multiple delay nonlinear wavelength dynamics for photonic Reservoir Computing.** European Conference on Lasers and Electro-Optics and the XIIth European Quantum Electronics Conference, Munich, Germany (2011).

Lennert Appeltant, Guy Van der Sande, Sergei Rybalko, Romain Martinenghi, Maxime Jacquot, Yanne Chembo, Laurent Larger, Ingo Fischer, Jan Danckaert. **Computational performance of a single bandpass electro-optic delay oscillator.** Osnabruck Computational Cognition Alliance Meeting on "Natural Computation in Hierarchies", Osnabruck, Germany (2011).

Lennert Appeltant, Miguel Cornelles Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, Joni Dambre, Benjamin Schrauwen, Claudio Mirasso, Ingo Fischer. **Single delay element as a computational unit.** International Symposium on Nonlinear Theory and its Applications, Kobe, Japan (2011).

Lennert Appeltant, Miguel Cornelles Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, Joni Dambre, Benjamin Schrauwen, Claudio Mirasso, Ingo Fischer. **Reservoir computing using a delayed feedback system: towards photonics.** Proc. Ann. Symp. IEEE/LEOS Benelux Chapter 2011, Ghent, Belgium, pp. 125-128,2011

Contents

1 Introduction

1.1 Learning vs programming

Novel methods for information processing are highly desired in our information driven society. While traditional Von Neumann computer architectures or Turing approaches [1] work very efficiently when it comes to executing basic mathematical instructions, in terms of efficiency they run into trouble for highly complex computational tasks such as, e.g., speech recognition or facial recognition. Our brain functions in a different way and seems to be optimally designed for these kinds of tasks. Walking down a street, we are constantly fed with sensory impressions. Seeing a vehicle or a familiar face, hearing the ongoing traffic and conversations, smelling the food stalls ... All these external impulses instantly produce large neural activity in your brain, so that we can recognize the passing bus, a good friend or a car horn or that the smell of freshly baked waffles makes us hungry.

When recognizing faces in a crowd, we are unaware that our brain not only analyzes each trait, but it also classifies these faces and compares them with known ones. Differentiating a tree from a car, the sound of a piano from the vacuum cleaner are types of activity we do constantly. The neural network system that constitutes our brain is constantly categorizing stimuli in different patterns and using these structures to interpret reality. In this, the human brain is very efficient. You can see a blurry photo and from that partial image still recognize the scene in a fraction of a second, a task for which a computer takes minutes or even hours. Today, except for mathematical operations, our brain functions much faster and more efficient than any supercomputer.

It is precisely this form of information processing in neural networks that has inspired researchers to create systems that mimic the brain's information pro-

cessing capabilities, radically different from current computer based schemes, see Fig. 1.1. Although many fundamental aspects of the brain's functioning are still unknown, we can get inspiration from some insights or the architecture of the brain. In Fig. 1.1(a) a Von Neumann computer architecture is depicted. An analog input coming from the outside world is digitized and subsequently processed by a preprogrammed computational unit. The instructions on how to treat the input bitstream are predefined in the program and they are usually executed by a combination of logical blocks. After the processing the output bitstream is converted back to an analog signal. In Fig. 1.1(b) another approach is used to solve the task. Instead of a traditional Von Neumann architecture, a neural network that learns by examples is used. For a considerable amount of problems it can be advantageous to *learn* how to solve them instead of executing a set of preprogrammed instructions. Machine learning is the scientific discipline that focusses on designing and implementing algorithms that allow to optimize learning behavior. The 'machine' or in many cases the network is being fed with some examples of possible inputs for the task to be solved. For a while the machine learns how to interpret and how to classify different inputs into different categories. When the learning phase is over, we can provide a previously unseen example to the machine for processing. The unseen input can be interpreted as long as it belongs to a general class of inputs that was present during the training phase. The training data have an unknown probability distribution, which can nevertheless reveal some underlying features when an appropriate training algorithm is used to interpret the data. The difficulty relies in the fact that not every single input pattern can be matched exactly with a different target class, hence the system and the algorithm need to have the capability to generalize. Consider for example the task where the idea is to classify pictures of people into two categories: man or woman. Although in most cases this should be an easy job for the human brain, formulating the rules in order to program this is far from straightforward. A more feasible approach consists of feeding examples to the system and providing it with the correct solution. The system should extract the classification rules without explicitly formulating them and then apply them to a test sample for which the solution is not given by the supervisor. Different women have different characteristics, but a well-trained system should be able to generalize them into one class.

Learning algorithms can be supervised or unsupervised. The former means that every input should be mapped to a certain target class, as defined by the (human) supervisor, during the training phase. When applying unsupervised learning there is no external control on the different categories of inputs to be distinguished. The machine itself needs to discover different features in the data during the processing, by identifying clusters of data inputs with similar

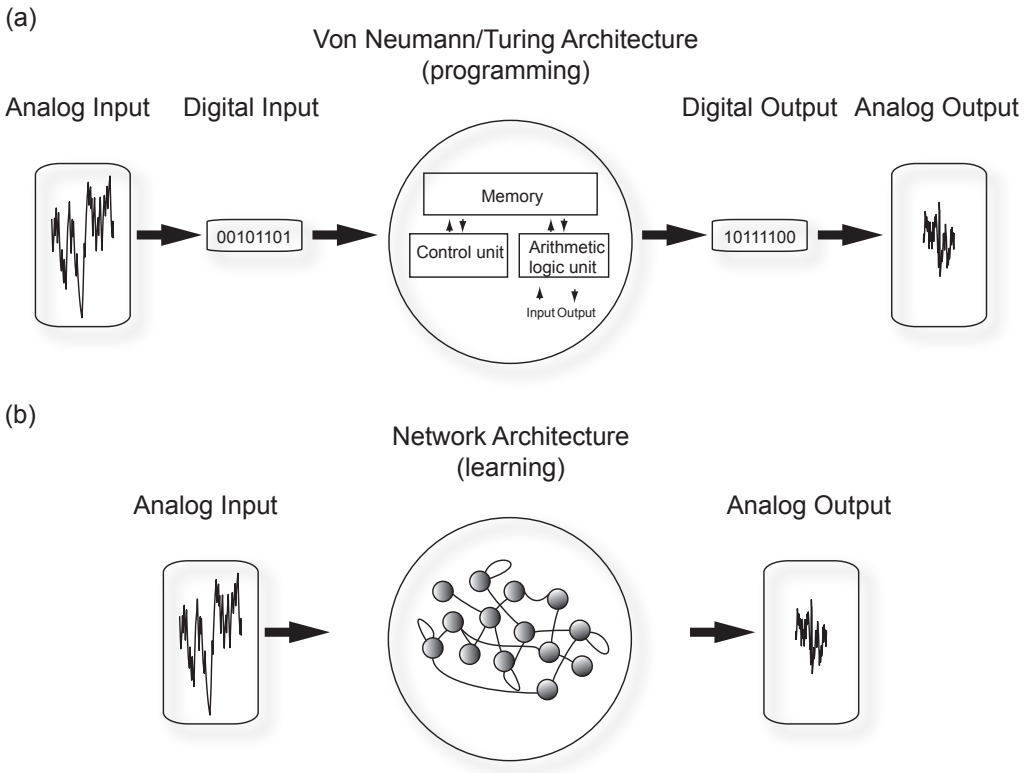


Fig. 1.1: Von Neumann versus network architecture. (a) Von Neumann architecture with pre-programmed instructions, executed by logical blocks. (b) Network architecture with a neural network that learns by examples.

properties. For problems where an agent, this could, e.g., be a robot, needs to interact with its environment, sometimes a technique called reinforcement learning is used. When using this approach, the goal is to make an agent follow a certain behavioral pattern in an environment, by rewarding him for beneficial actions and punishing for wrong choices. Characteristic for reinforcement learning is that a trade-off needs to be made, comparing long term with short term decisions.

As mentioned before, a system that relies on learning by examples is the human brain. In essence, the brain is a network of spiking neurons or nodes, connected to each other within a certain configuration. Each neuron is an electrically excitable cell that releases a spike when the voltage gradient over its membrane reaches a certain threshold. Trying to mimic and to understand the computational abilities of the brain, a lot of effort has been put into creating artificial neural networks, which refers to models of the brain structure, where the mathematical principle of what happens in the brain is applied without necessarily having to go into the biological details of a neuron. These structures allow for solving tasks such as pattern recognition, time series prediction and system identification in a computationally efficient way.

1.2 Artificial neural networks

The idea behind artificial neural networks is that a network is built from neurons[1] or nodes. These are in many cases treated as black boxes with a certain nonlinear transfer function, sometimes referred to as activation function. Some or even all of these nodes are connected with each other, resulting in a more global connectivity structure. A signal input is fed to the network and, while propagating through the connected nodes, it gets (nonlinearly) transformed. In the end, the signal reaches the output nodes, which send the output signal to the outside world. The shape of this output signal is not only dependent on the exact shape of the activation function of the nodes, but it is also sensitive to the way the nodes are interconnected with each other. Hence, by appropriately adapting the strength of the connections among all the nodes, the shape of the resulting output signal can be manipulated. This process is generally referred to as training and allows for a network to map input values on their corresponding target values [2, 3, 4].

[1] The terminology 'neuron' is not to be understood literally. It refers to the rough functionality, but it is not necessarily biology-related.

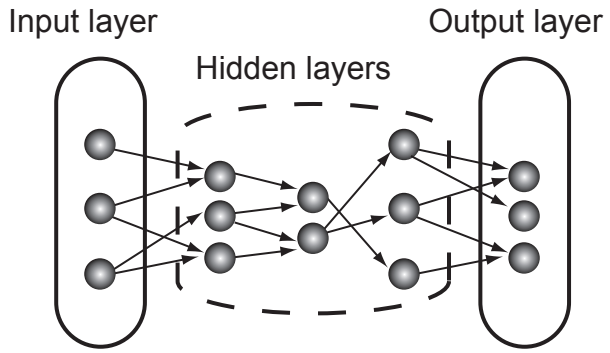


Fig. 1.2: Network topology: feedforward. A feedforward network with several layers is shown. The information only travels forward through the network and never enters in a loop.

1.2.1 Feedforward neural networks

In the most simple case, artificial neural networks consist of a structure in which no internal loops can be found, meaning that the propagating signal will never pass the same neuron twice, as depicted in Fig. 1.2. These structures are called feedforward neural networks and can be trained using linear algorithms such as backpropagation until the examples are correctly classified or a stopping criterion is satisfied [5]. All the neurons are divided into separate sequential layers and the signal only goes forward, one layer per discrete time step. The interaction between the neurons and the outside world only happens in the first and the last layer of the network. The first layer is an input layer that injects the input signals into the network, while the output layer provides the resulting signal coming out of the network structure. The layers in between are sometimes referred to as hidden layers. Because of their feedforward structure these kinds of networks are not capable of processing temporal information. The only values available for read out and interpretation of the signal are the ones in the last layer, which only contain information about a single moment of the input history. They were originally designed to process static spatial patterns of inputs.

1.2.2 Recurrent neural networks

When recurrent connections are added to the network, the training procedure becomes drastically more complicated. Because of the recurrence within the network, input data can remain present in the network for a much longer time and a certain dynamical memory is created. Then, the state of the network

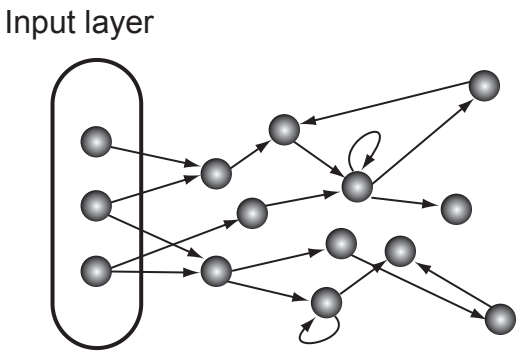


Fig. 1.3: Network topology: recurrent. A recurrent network is shown. Some connections couple the signal back to nodes belonging to previous layers, therefore making the layer structure obsolete.

does not only depend on the current input value, but also on the past one(s). The network is a dynamical system, the state of which can possibly depend on all previous input values that were ever presented to the system. This enables the processing of temporal information, necessary for tasks such as speech recognition, time series prediction etc. In recurrent neural networks, splitting up the system into different layers makes less sense since the signals can, in principle, endlessly loop around through all layers of the network. Even the output nodes can be fed back to the previous layers. An example of a recurrent network is shown in Fig. 1.3.

While training a feedforward network is a relatively easy task, doing this for a recurrent network is much harder. For a feedforward network a linear training algorithm, such as a least squares fit, can be used to determine the weights for all the connections. The training of a recurrent neural network becomes highly nonlinear and requires a lot of computational power. It does not even necessarily converge. It is exactly this problem that is avoided in the recently introduced concept of *reservoir computing*. The output layer is explicitly separated from the rest of the network and only the connections from the network to the output layer are trained. The connections within the network itself remain unaltered. By using this procedure, the training becomes linear. The untrained network implements a transformation of the input, which can be interpreted by the output layer using linear algorithms.

1.3 Reservoir computing

1.3.1 General concepts

Reservoir computing [6, 7, 8, 9, 10, 11, 12] is a recently introduced, bio-inspired, paradigm in machine-learning. With this approach, state-of-the-art performance has been found for processing empirical data. Even for tasks that are deemed computationally hard, such as chaotic time series prediction [9], or speech recognition [13, 14], amongst others, good results are obtained with a computationally efficient process. The main inspiration underlying reservoir computing is the insight that the brain processes information by generating patterns of transient neuronal activity excited by input sensory signals [15]. The electrical discharges of billions of neurons are organized in such a way that our brain can deliver the correct response to an external stimulus in a very short time. An analogy that is often brought up by the machine learning community is the one with waves emerging in a bucket of water when small pebbles are thrown into it. With the naked eye it might be tricky to make an estimation on the weight of the pebbles. The key idea is to transform this original question to another one that is much easier to solve. When the pebbles are thrown into a bucket of water wave patterns will emerge. The wave is a transient phenomenon because if no more perturbations are introduced, eventually it will fade out. By studying the wave pattern, one could deduce where the pebble hit the water surface or when it happened. The magnitude of the wave could even give an indication about the size and weight of the stone or about the velocity with which it was thrown. The water serves as a reservoir that will not solve the original problem, but it translates it into another form, allowing other methods to be used to interpret the information. Although just an analogy, the bucket of water provides an insight into some of the crucial elements of a potentially successful reservoir.

The objective of reservoir computing is to implement a specific nonlinear transformation of the input signal or to classify the inputs. Classification involves the discrimination between a set of input data, e.g., identifying features of images, voices, time series, etc. In order to perform the task, neural networks require a training procedure. Since recurrent networks are notoriously difficult to train, they were not widely used until the advent of reservoir computing. Another layer is added and the only part of the system that is trained are the connections from the reservoir to this extra layer. Thus, the training does not affect the dynamics of the reservoir itself. The situation is depicted in Fig. 1.4.

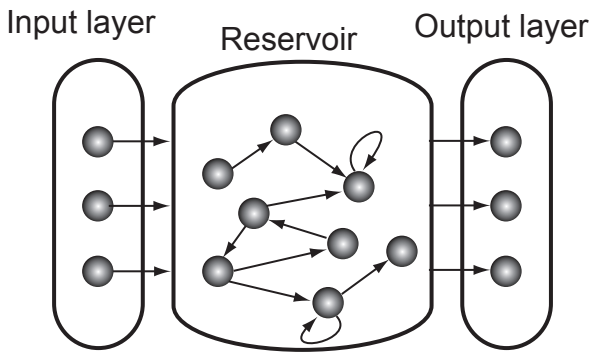


Fig. 1.4: Network topology: reservoir computing. A reservoir computing network is shown. The reservoir is a recurrent network, explicitly separated from the output layer.

To efficiently solve its tasks, a reservoir should satisfy several key properties. Firstly, it should nonlinearly transform the input signal into a high-dimensional state space in which the signal is represented. In machine learning this is achieved through the use of a large number of reservoir nodes which are connected to each other through the recurrent connections of the reservoir. In practice, traditional reservoir computing architectures employ several hundreds/thousands of nonlinear reservoir nodes to obtain good performance. In Fig. 1.5, we illustrate how such a nonlinear mapping to a high-dimensional state space facilitates separation (classification) of states with the example of an XOR. Consider the situation depicted in Fig. 1.5(a). Two binary input variables, x and y, lead to a target that corresponds to an XOR logical function. If x and y both have the same value this results in a 0, represented by a star. If x and y have different values, the result is a 1, represented by a sphere. The goal is to separate the red stars from the yellow spheres, but this cannot be achieved with one straight line. If this would have been the case, the problem would have been linearly separable. Linearly separable problems are regarded as easy, since they can be solved with a linear training algorithm. When smartly mapping this problem from a two-dimensional space onto a three-dimensional one, the nature of the separability changes. In Fig. 1.5(b) both variables kept their initial x- and y-positions, but the yellow spheres were given a different position along the z-axis compared to the red stars. It suffices to introduce one straight plane to separate the two types of variables. The 2D plane in 3D space is the equivalent of a straight line in 2D space. The nonlinear transformation to high-dimensional space does not construct the hyperplane itself, but it allows its existence by reshaping the nonlinear separation problem onto a linear one. Reservoir computing implements this idea: the input signal is

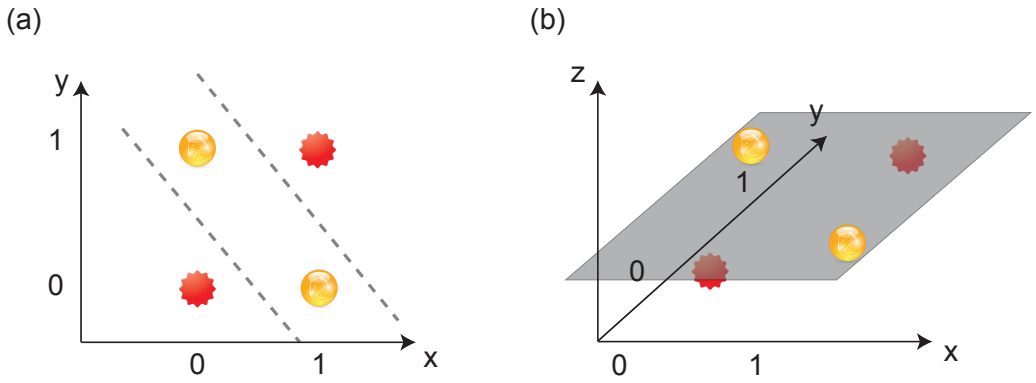


Fig. 1.5: Illustration of linear separability. In (a) The XOR problem in a two-dimensional input space: a 0 corresponds to a star and a 1 to a sphere. The yellow spheres and the red stars cannot be separated by a single straight line. (b) With a nonlinear mapping into a three-dimensional space the spheres and stars can be separated by a single linear 2D plane. Figure taken from Appeltant *et al.* [17].

nonlinearly mapped into the high-dimensional reservoir state represented by a large number of nodes. It can be shown that, the higher the dimension of the space is, the more likely it is that the data become linearly separable, see e.g. [16] .

Secondly the dynamics of the reservoir should be such that it exhibits a fading memory (i.e., a short term memory): the reservoir state is influenced by inputs from the recent past, but it is independent of the inputs from the remote past. This property is essential for processing temporal sequences (such as speech) for which the history of the signal is important. Additionally, the results of reservoir computing must be reproducible and robust against noise. For this, the reservoir should exhibit sufficiently different dynamical responses to inputs belonging to different classes. At the same time, the reservoir should not be too sensitive: similar inputs should not be associated to different classes. These competing requirements define when a reservoir performs well. Typically reservoirs depend on a few parameters, which must be adjusted to satisfy the above constraints. Experience shows that these requirements are satisfied when the reservoir operates (in the absence of input) in a steady regime. However, many aspects of dynamics leading to good performance are not yet known. For the reader interested in a more in-depth presentation of reservoir computing, we refer to the recent review articles [18, 19, 20, 21].

Fig. 1.6: Reservoir computing applications. (a) Modeling the movements of a robot arm based on sensory inputs, picture by [40] (b) Predicting and explaining traffic jam situations, picture by [41] (c) Speech recognition, picture by [42] (d) Handwriting recognition, picture by [43].

1.3.2 Applications

Nowadays, applications of neural networks are found in a large variety of fields. They are very commonly used in robotics [22, 23, 24, 25, 26, 27, 28, 29] where the main goal is usually to induce a movement after a sensory input. While accurate models exist to organize the movement in a traditional way, these often lack the flexibility to be used in a practical implementation of the robot. Some sensors might have slightly different parameters than the ones used in the model, or some parts of the robot design might simply be unknown. Neural networks are more suitable to estimate parameters from the system's behavior itself. Also, in all kinds of pattern recognition neural networks represent the state-of-the-art. They are used for automatic detection of credit card fraud [30], optical character recognition [31, 32] or grammar modeling [33]. Furthermore, reservoir computing has been contributing to speech recognition [17, 13, 34, 35, 36], noise modeling [9] and the generation and prediction of chaotic time series [37, 38, 39]. While numerous numerical implementations of this concept exist, competitive technical implementations are still scarce.

1.3.3 Different views on reservoir computing

1.3.3.1 Machine learning

From the viewpoint of machine learning, the techniques used in reservoir computing are related to those implemented in support vector machines, originally introduced in the nineties by Vapnik [44]. Support vector machines have proven to be able to attain state-of-the-art performance on a number of tasks. They also rely on a mapping of a low-dimensional input onto high-dimensional states with the goal to construct a hyperplane that separates different classes of data. The implementation, however, is fundamentally different. The main difference with reservoir computing relies in the exact realization of the high-dimensional mapping. While in reservoir computing the mapping is explicit -the dynamical response resulting in the reservoir states-, in support vector machines the high-dimensional space does not need to be formulated explicitly. A technique called the kernel trick is employed [45] and standard kernels are used for the mapping. A second difference is that in reservoir computing the mapping onto feature space is explicitly temporal. This is implemented by reservoirs exhibiting fading memory.

Another class of traditional computation machines closely related to reservoir computing are Hopfield networks [46, 47]. This approach is based on a set of attractors each one representing a certain class in which inputs can be categorized. Each state of the network can be related to a scalar value referred to as the *energy* of the network. If the activation functions of some nodes in the network are randomly updated, the network will evolve to a new state that corresponds to a local minimum of the energy function. Training the network means lowering the local minima of interest. When a perturbed input is fed to the system, it will be related to one of the local minima, based on similarity. A disadvantage of this concept is that it lacks the ability to process information at any moment in time. When a certain input is fed to the network, the state has to evolve to an attractor and this attractor represents the final computation result. No information is extracted from the evolution in state-space towards a certain attractor, only the final result is interpreted. In the case of reservoir computing also the excursion in phase-space to reach the new steady state is used to process the input. The states of the nodes represent transient behavior and they contain information of both the present and past inputs, thus creating memory.

1.3.3.2 Neuroscience

From the viewpoint of neuroscience, reservoir computing aims at mimicking the way the brain does information processing. In this context, reservoir

computing assumes that the neurons are embedded in a complex network whose intrinsic activity is modified by external stimuli. The persistent neuronal network activity makes the information processing of a given stimulus occur in the context of the response to previous excitations. The generated network activity is then projected into other cortical areas that interpret or classify the outputs. It was this bio-inspired view that motivated one of the original reservoir computing concepts, the liquid state machine [12].

1.3.3.3 Dynamical systems

From the point of view of complex system studies, the reservoir can be regarded as a complex dynamical system that operates optimally in a certain dynamical regime. Three basic properties, linked to the dynamical properties of the network should be fulfilled for a network to perform as reservoir [12]. Firstly, different inputs should be mapped onto different reservoir states. This is generally referred to as the *separation property*. Secondly, reservoir states that are only slightly different should be mapped onto identical targets. If not, noise would suffice to map identical inputs onto different target values. This is called the *approximation property*. Finally, *fading memory* is desired. In many tasks, the information is stored in the temporal behavior of the input (e.g. speech recognition). It does not suffice to process the present input values, also previous values have to be taken into account. Usually, only recent inputs are relevant while those from the far past do not need to be taken into account. These three properties can be realized by the dynamical system, provided that the system resides in a proper dynamical regime. When the system operates in a chaotic regime, it is highly sensitive to small input variations and therefore has very good separation properties. The separation might, however, become so high that the approximation property no longer holds. In the reservoir community it is sometimes claimed that the edge of chaos is an optimal operating point [18], since it offers a compromise between a stable system, with good approximation properties and fading memory, and a chaotic system, with excellent separation capability. This is not always the case and we have identified the steady regime as appropriate operating point, without input.

This viewpoint, relating reservoir computing to complex dynamics, suggests that reservoir computing can be implemented in a wide variety of physical systems, provided that separation, approximation and fading memory properties are fulfilled. This has led to a few proof-of-principle demonstrations using different systems such as a bucket of water [48], the cerebral cortex of a cat [49], a VLSI chip [50]. However, in all these implementations the

tasks performed have been rather simple and the performances did not reach those of digital implementations.

In 2006, during a talk at the IAP doctoral school in Couvin (B), prof. J. Van Campenhout (U. Gent) proposed the idea of investigating photonic implementations of reservoir computing [51]. This path has been actively pursued in the 6th phase (2007-2011) of the IAP network "photonics@be" [52, 53] by groups at U. Gent, ULB and VUB. In our work we will show that delay-coupled optical systems represent excellent substrates for reservoir computers They allow achieving sufficiently high-dimensional dynamics and thus the required mapping with only a few elements. When considering a practical application, the delayed feedback system can be implemented using photonics. Photonic systems have proven to be robust and well-controllable, offering high processing speed and low power consumption. In order to succeed, we have modeled several photonic systems to implement the reservoir computer. In collaboration with several groups, numerical simulations have been complemented by electronic and opto-electronic implementations, on which we will report in Chapters 3 and 4.

1.4 Delayed feedback systems as reservoirs

1.4.1 Delayed feedback systems

Nonlinear systems with delayed feedback and/or delayed coupling, often simply put as 'delay systems', are a class of dynamical systems that have attracted considerable attention, because they arise in a variety of real life systems [54]. They are commonly found in, e.g., traffic dynamics due to the reaction time of a driver [55], chaos control [56, 57] or gene regulation networks where delay originates from transcription, translation, and translocation processes [58]. Also in predator-prey models they occur with the time delay representing a gestation period or reaction time of the predators. Sometimes the delay in the system originates from the fact that the previous number of predators has an influence on the current rate of change of the predators [59]. In the brain, delay occurs because of the axonal conduction delay between two neurons [60]. Remote cerebral cortical areas are subject to an entire series of these axonal conduction delays. The total connection delay between these areas could even be tens of milliseconds, but still zero time lag synchronization between them was observed [61, 62, 63]. Delay is found in networks of semiconductor lasers [64] when the signal travels from

one laser to the other. Whether it is through free space or via, e.g., an optical fiber, the light needs to cover a certain distance and that requires time. In control systems the time-delayed feedback originates from the fact that there is a finite time between the sensing of the information and the subsequent reaction of the system under the influence of a control signal. Another example taken from daily life is temperature control of the water coming from a shower. Because of the fact that the water needs to travel a certain distance along the tube between the heating element and the shower head the response to any temperature adjustment of the system is not immediate from the perspective of the user. This could lead to an unstable behavior where the controller increases or decreases the temperature of the water too much due to apparent non-responsivity of the system.

It has been shown that delay has an ambivalent impact on the dynamical behavior of systems, either stabilizing or destabilizing them [57], with possible emergence of complex dynamics. This has been observed in e.g. biological systems [65] or laser networks [66]. Often it is sufficient to tune a single parameter (e.g., the feedback strength) to access a variety of behaviors, ranging from stable via periodic and quasi-periodic oscillations to deterministic chaos [67]. In photonics a normally stable laser source can become chaotic when subjected to feedback even for small feedback strengths. As an example we take one of the most simple delay systems, given by the equation

$$\dot{x}(t) = -\alpha x(t-\tau), \tag{1.1}$$

where we choose $\alpha = 0.2$. In Fig. 1.7 we show the solution of this equation for three different values of τ. In Fig. 1.7(a) $\tau = 7$. When looking at the time trace some damped oscillations can be observed in the transient before the system reaches a constant output value. However, when the delay time is increased to $\tau = 8$, as in Fig. 1.7(b) the oscillations are no longer exponentially damped. They increase in amplitude with time. For an even larger τ, equal to 10, this behavior is confirmed with an even stronger growth in amplitude. For this system the delay clearly has a destabilizing effect.

From the application point of view the dynamics of delay systems is gaining more and more interest: whereas initially it was considered more as a nuisance, it is now viewed as a resource that can be beneficially exploited. It found applications in chaos-communication [68] and also the results presented in the next chapter are an example of benefitting from the delay in the system [17, 69]. One of the simplest possible delay systems consists of a single nonlinear node whose dynamics is influenced by its own output a delay time in the past. Such a system is easy to implement, since it comprises only two elements: a nonlinear node and a delay loop. When going to more complex

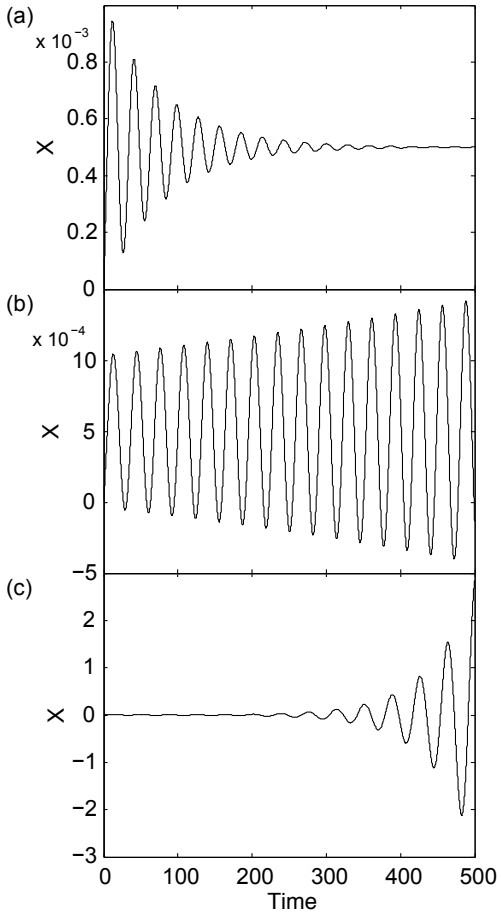


Fig. 1.7: Destabilizing effect of delay. Time trace originating from the system given by Eq. (1.1) (a) $\tau = 7$, (b) $\tau = 8$, (c) $\tau = 10$.

situations of several nonlinear nodes being coupled with delay, these systems have successfully been used to describe the properties of complex networks in general. They allow a better understanding of e.g. synchronization and resonance phenomena [70, 71, 72]. Of particular interest for this thesis is the situation in which only a few dynamical elements are coupled with delay within a certain configuration, e.g., a ring of delay-coupled elements [66].

Mathematically, delay systems are described by delay differential equations (DDE) that differ fundamentally from ordinary differential equations (ODE) as the time-dependent solution of a DDE is not uniquely determined by its initial state at a given moment. For a DDE the continuous solution on an interval of one delay time needs to be provided in order to define the initial conditions correctly. The general form of a DDE is given by

$$\dot{x} = F\left[x(t), x(t-\tau)\right].$$

with F any given linear or nonlinear function and with τ being the delay time. Mathematically, a key feature of time-continuous delay systems is that their state space becomes infinite dimensional. This is because their state at time t depends on the output of the nonlinear node during the continuous time interval $[t-\tau, t[$. Another interpretation is that a delayed feedback equation leads to a non-rational transfer function, resulting in an infinite number of poles. The dynamics of the delay system remains finite dimensional in practice [73], but exhibits the properties of high dimensionality and short-term memory. Since two key ingredients for computational processing are nonlinear transformation and high-dimensional mapping, delay systems are suitable candidates.

1.4.2 Can delay systems be used as reservoirs?

1.4.2.1 Topology of the network approach

Reservoir computing is an implementation of a recurrent neural network with the general idea that the network is split up into several parts. The recurrent part is difficult to train, therefore another layer is added, which is no more than a series of simple linear nodes. Traditional reservoir computing implementations are generally composed of three distinct parts: an input layer, the reservoir, and an output layer, as illustrated in Fig. 1.8.

The input layer feeds the input signals to the reservoir via fixed random weighted connections. These weights will scale the input that is given to the

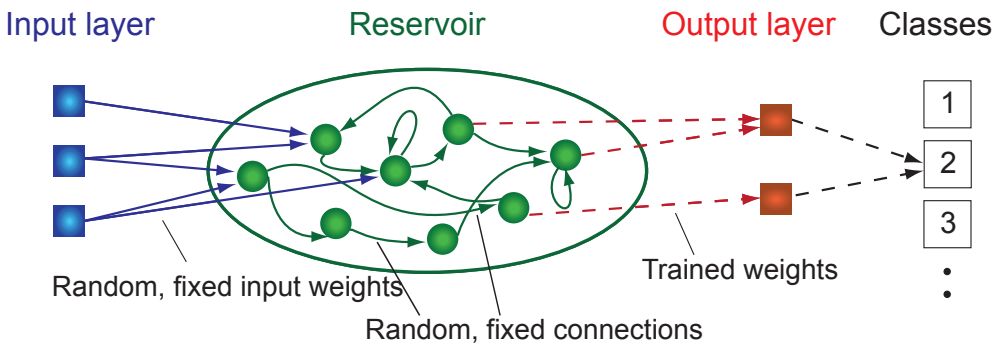


Fig. 1.8: Classical reservoir computing scheme. The input is coupled into the reservoir via a randomly connected input layer to the N nodes in the reservoir. The connections between reservoir nodes are randomly chosen and kept fixed, that is, the reservoir is left untrained. The reservoir' s transient dynamical response is read out by an output layer, which are linear weighted sums of the reservoir node states. Figure taken from Appeltant *et al.* [17].

nodes, creating a different input scaling factor for every individual node. The second layer, which is called reservoir or liquid, usually consists of a large number of randomly interconnected nonlinear nodes, constituting a recurrent network. The nodes are driven by random linear combinations of input signals. Since every node state can be seen as an excursion in another state space direction, the original input signal is thus projected onto a high dimensional state space. The emerging reservoir state is given by the combined states of all the individual nodes. Contrary to what happens in traditional recurrent neural networks, the coupling weights in the reservoir itself are not trained. They are usually chosen in a random way, globally scaled in order for the network to operate in a certain dynamical regime. Under the influence of input signals the network exhibits transient responses. These transient responses are read out by the output layer via a linear weighted sum of the individual node states, with no additional nonlinear transformation happening in the last layer. The training algorithm can thus be drastically simplified to a linear classifier.

The reservoir computing implementation we work with is closely related to echo state networks [9]. In echo state networks the node states at time step k are computed according to the following equation:

$$\mathbf{x}(k) = f\left[W_{res}^{res} \cdot \mathbf{x}(k-1) + W_{in}^{res} \cdot \mathbf{u}(k-1)\right]. \tag{1.2}$$

In this equation, $\mathbf{x}(k)$ is the vector of new node states at time step k, $\mathbf{u}(k-1)$ is the input matrix, which is considered at time step $k-1$, for causality reasons. The matrices W_{res}^{res} and W_{in}^{res} contain the (generally random) reservoir and input connection weights. The weight matrices are scaled by multiplicative factors in order to get good performance. For the nonlinear function f often a sigmoidal function, e.g. $f(x) = \tanh(x)$ is chosen. In some cases, feedback from the output to the reservoir nodes is also included. This is not used in our approach[2]. In a simplified formulation, the output is a weighted linear combination of the node states, a constant bias value and the input signals themselves.

$$\hat{\mathbf{y}}(k) = W_{res}^{out} \cdot \mathbf{x}(k) + W_{in}^{out} \cdot \mathbf{u}(k-1) + W_{bias}^{out}. \tag{1.3}$$

In reservoir computing only the matrices in Eq. (1.3) are optimized (trained) to minimize the mean square error between the calculated output values $\hat{\mathbf{y}}(k)$ and the required output values $\mathbf{y}(k)$.

1.4.2.2 Topology of the delayed feedback approach

In this section we introduce the general principle how delay dynamical systems can perform efficient computation. We succeed in replacing an entire network of connected nonlinear nodes by one single nonlinear node subjected to delayed feedback. This approach does not only provide a drastic simplification of the experimental implementation of artificial neural networks for computing purposes, it also demonstrates the huge computational processing power hidden in even the simplest delay-dynamical system. In Fig. 1.9 the delayed feedback equivalent of Fig. 1.8 is shown. Contrary to the parallel input feeding to several nodes in the case of the neural network, in the delayed feedback system all inputs need to be injected in one nonlinear node. To compensate for the loss of parallelism the input is pre-processed. This pre-processing will from now on be referred to as the masking procedure. It combines time-multiplexing of the input with imprinting different scaling factors on the input ensuring that the system always resides in the transient regime. It can be seen as a convolution between a masking function and the input to be injected in the system. The entire process is explained in detail in Chapter 2. After injection of the input into the node, the transformed signal resides in the delay line for a time τ before it is re-injected into the nonlinear node. Since the high-dimensionality of the system can be found

[2]When connections from the output layer back to the reservoir are included Eq. (1.2) becomes: $\mathbf{x}(k) = f\left[W_{res}^{res} \cdot \mathbf{x}(k-1) + W_{in}^{res} \cdot \mathbf{u}(k-1) + W_{out}^{res} \cdot \hat{\mathbf{y}}(k-1)\right]$.

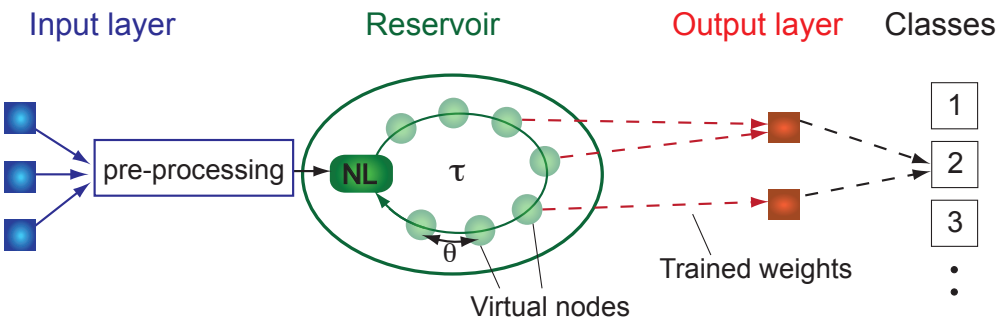


Fig. 1.9: Delayed feedback reservoirs scheme. All input channels are sent in via the one nonlinear node. Figure taken from Appeltant *et al.* [17].

along the delay line, different states residing in it are regarded as the neurons or nodes of the system. Because they do not represent physical nodes, we refer to them as virtual nodes. The states they contain do represent a nonlinearly transformed version of the input, but the transformation happened earlier in the real nonlinear node. The temporal separation of the different virtual nodes, θ, which is in fact the interval with which we read out states of the delay line, plays an important role for the performance of the system. We elaborate on this in Chapter 2. The node's transient dynamical response is read out along the delay line by an output layer, which combines them linearly in a weighted sum.

1.4.2.3 An example: chaotic time series prediction

To compare the approaches of traditional reservoir computing and our delayed feedback system, we demonstrate their function by means of a commonly used benchmark task: chaotic time series prediction. Without going into detail about the exact data processing, we illustrate the different steps and compare the performance. The test originates from a time series prediction competition, organized as a survey to compare different time series forecasting methods. At that time many new and innovative methods, such as artificial neural networks, emerged to compete with standard prediction methods. In May 1993 in Santa Fe, New Mexico the NATO Advanced Research Workshop on Comparative Time Series Analysis was held to have an overview of existing methods and their performance [74]. Several time series coming from different systems were provided as a challenge:

- A physics laboratory experiment (NH_3 laser)

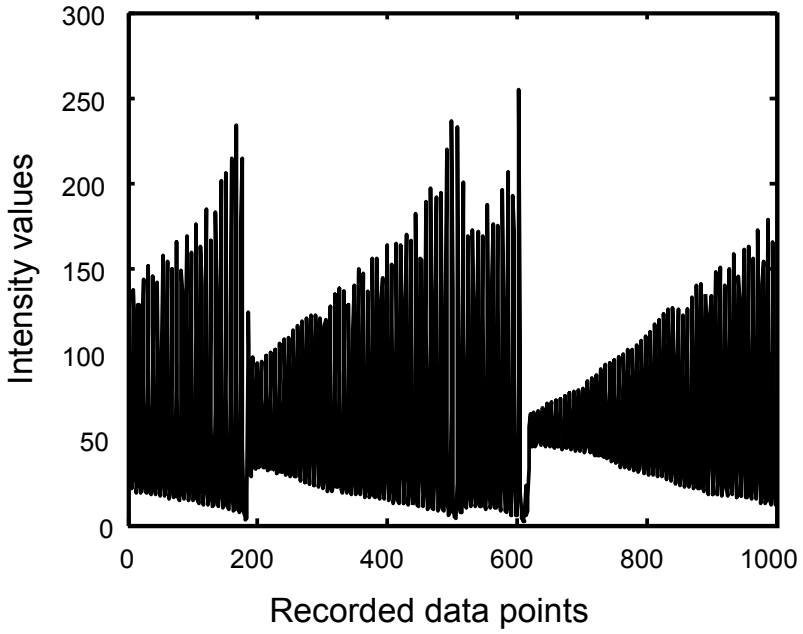


Fig. 1.10: Santa Fe input data. The input data series for the Santa Fe time series prediction of a chaotic laser is shown. The y-value denotes the measured intensity of the laser, while the x-axis indicates the index of the recorded sample.

- Physiological data from a patient with sleep apnea
- Tick-by-tick currency exchange rate data
- A computer-generated series designed specifically for the competition
- Astrophysical data from a variable white dwarf star
- J. S. Bach's last (unfinished) fugue from "Die Kunst der Fuge."

In this thesis we only consider the first set, coming from the chaotic laser exhibiting dynamics related to Lorenz chaos. The input data series is depicted in Fig. 1.10, with the laser intensity shown on the y-axis versus the index of the sampled data point.

The goal is to make a one-step ahead prediction, based on the present value of the system and this for all values of the time trace. In our training procedure several time series as the one depicted in Fig. 1.10 are fed to the system, for the case of a neural network with many nodes and a delayed feedback system, as examples. The systems will process the input data and

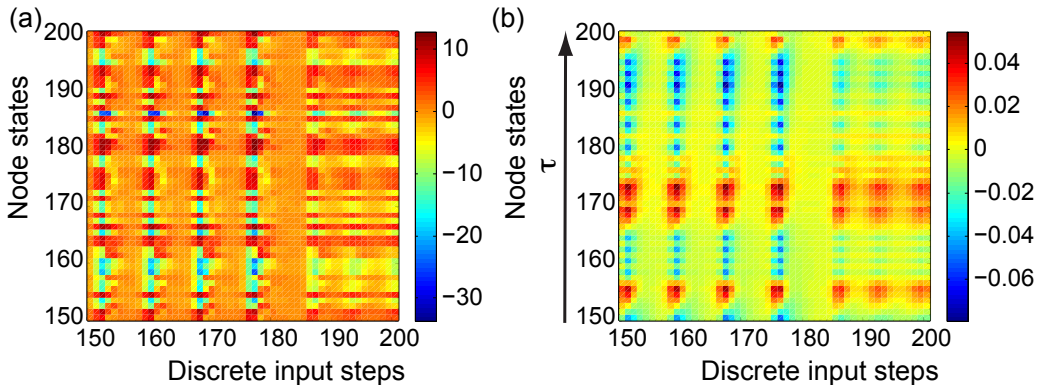


Fig. 1.11: Spatio-temporal representation Santa Fe. A zoom is presented of the evolution of the reservoir states of nodes. Feeding in 1000 input steps leads to the construction of 400 reservoir states of each 1000 steps. Here only 50 steps are shown for 50 nodes. The state values are shown in color code. (a) network reservoir approach, (b) delayed feedback reservoir.

nonlinearly transform it. In Fig. 1.11 the reservoir states are shown both for a network of randomly connected nodes and for a delayed feedback system, where we consider 400 states in both cases. One time series realization consists of 1000 measurement points. Every point that is fed to the reservoir leads to a change in all 400 node states of the reservoir, hence 400 series of 1000 points are recorded and plotted as reservoir states. Both systems rely on a different configuration, but for both the same nonlinearity type, a Mackey-Glass nonlinearity, was taken, with identical parameters. This type of nonlinearity will be extensively discussed in Chapter 3.

Both in the situation of Fig. 1.11(a) and the one of Fig. 1.11(b) 400 nodes were used, but only 50 node states are plotted. In Fig. 1.11(a) the reservoir states of a traditional network are depicted. The different node states are plotted along the y-axis and their evolution in discrete time is given by moving along the x-axis. Fig. 1.11(b) shows the states we can obtain with a delayed feedback setup. What is plotted corresponds to the spatio-temporal mapping carried out by the system [75]. On the y-axis one interval of τ is depicted, containing all the states in the delay line (in Fig. 1.11(b) only a part of the interval τ is shown). Moving along the x-axis gives the evolution in time of the entire delay line. Every discrete input step in Fig. 1.11(b) corresponds to a jump in time of τ. The general trend of the reservoir states is quite similar for the network and the delayed feedback response. The fact that they both respond in a similar way to identical inputs already gives a first indication that both are able to extract information in an equivalent way. The details of the input signal flowing through the different reservoirs and

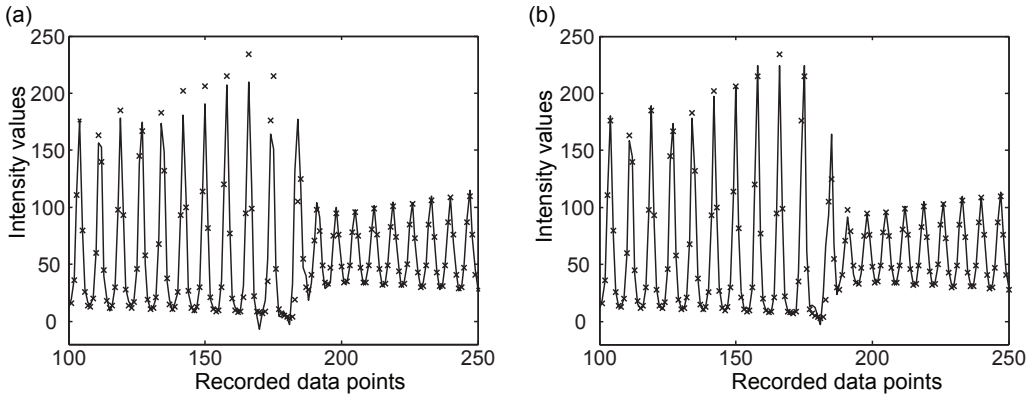


Fig. 1.12: Target reconstruction Santa Fe. The crosses represent the sample points of the original target series. The full line connects the approximation of the target. (a) The network reservoir. (b) The delayed feedback system reservoir.

the construction the reservoir states are not given in this introduction. In Chapter 2 we elaborate on the exact procedure of feeding and pre-processing the input.

In Fig. 1.12 the result of the training procedure on these reservoir states is depicted. The crosses correspond to the original target and the black curve is the approximation. Please note that the approximation of the target is also a discrete time series with the same number of samples as the original target. The full lines are present only for clarity and do not mean that we only sampled some points of the input or target. The error, expressed as a normalized mean square error, see Chapter 2, section 2.2.1, in this example is 0.0651 for the network approach and 0.0225 for the delayed feedback approach.

We have developed a way to drive and train these kind of delayed feedback reservoirs and have achieved performance comparable to state-of-the-art digital numerical simulations of reservoir computing. Moreover, based on our numerical results the first experimental implementations of delayed feedback reservoirs have been realized by members within the PHOCUS consortium[3].

[3]PHOCUS is an acronym and stands for: towards a PHOtonic liquid state machine based on delay-CoUpled Systems. This research project, funded by the European Commission within the Seventh Framework Programme (FP7) in the domain of Future and Emerging Technologies (FET-Open), brings together seven groups from four European countries. The first experimental implementation, an electronic setup, was realized by dr. M.C. Soriano, prof. C.R. Mirasso and prof. I. Fischer at IFISC at the Universitat de les Illes Balears in Palma de Mallorca. The second hardware setup, an opto-electronic delayed feedback reservoir was accomplished by prof. L. Larger, dr. D. Brunner, dr. M.C.

These implementations are, as far as we know, the first experimental reservoirs able to compete with numerical implementations of reservoirs in terms of performance. Our results, providing the proof of principle, were published in Nature Communications [17], and opened an entirely new range of possibilities to implement artificial neural networks in a resource-efficient way. Delayed feedback systems are relatively easy to implement in electronics, and even more so in photonic systems. Delayed feedback can be easily provided by a fiber loop holding the light for a certain amount of time. In the meanwhile, an opto-electronic extension of the originally electronic implementation has been realized [69]. We elaborate on them in Chapter 3 and Chapter 4.

1.5 Overview of this thesis

The main goal of this thesis is to demonstrate that delay dynamical systems can be used to successfully process information. We develop suitable methods to drive and train these systems and compare their performance with conventional reservoir computing systems.

- In Chapter 2 we go into the details of how delay dynamical systems can be used to perform reservoir computing. We elaborate on the masking procedure required to pre-process the input data and we introduce a formalization of the introduced interconnection structure between the virtual nodes.

- In Chapters 3 and 4 two different nonlinearities, one of Mackey-Glass type and the other of Ikeda type, are studied in terms of performance by evaluating several benchmarks. We present both numerical and experimental results, demonstrating an electronic and an opto-electronic implementation of the concept.

- Chapter 5 focusses on properties of the reservoir that are benchmark-independent. By studying the memory and the computational ability of a system, one is able to predict regions of good performance with respect to the requirements of the used benchmark task.

Soriano, prof. C.R. Mirasso and prof. I. Fischer in the same lab. During my stays at IFISC, I have had the opportunity to contribute to both experimental implementations. In the meanwhile, also another opto-electronic implementation of reservoir computing was realized at the ULB by a consortium of ULB and UGent. See Paquot *et al. [76].*

- Next to the basic configuration of a delayed feedback system, several modifications can be made in order to increase the system's performance. After studying the key properties of a reservoir configuration in Chapter 5, we use this knowledge in Chapter 6 to compensate for possible weaknesses concerning certain tests. By creating multiple delayed feedback structures we boost the memory capacity and by introducing a band-pass filter we can widen the region of operation. Next to structural changes of the system comprising one nonlinear node, we present first steps towards studying small network motifs.

2

Single node with delay: input, training and testing

In Chapter 1 we already proposed delayed feedback systems as suitable candidates for reservoir computing inspired problem solving. Here, we show that this intuition is correct and that, by adapting the reservoir computing paradigm to dlelay systems, properties of simple dynamical systems can be associated with information processing capabilities[1]. We go into detail on how to drive and train these systems. Because of the radically different architecture the input feeding requires a pre-processing procedure. We explain how this approach allows us to create and optimize the interconnection structure of the virtual nodes present in the delay line. The states are collected from the delay line and are interpreted by a training algorithm to construct a desired target function. To evaluate the performance of the system we introduce benchmark tasks that are commonly used in the field of machine learning and which we will use to quantify the performance of the delayed feedback system.

Parts of the work described in this chapter are published in Appeltant, L. *et al.* Information processing using a single dynamical node as complex system. *Nat. Commun.* 2:468 doi: 10.1038/ncomms1476 (2011).

[1]The presented architecture is the result of a tedious process of matching the requirements for a well-performing reservoir with the constraints imposed by a single node system. We evolved stepwise to an approach were the dynamics of the system provide the necessary ingredients for reservoir computing. The present approach came to life because of several discussions with dr. G. Van der Sande, prof. J. Danckaert, prof. J. Dambre, prof. B. Schrauwen, S. Massar, dr. M.C. Soriano, prof. C.R. Mirasso and prof. I. Fischer. For the training of the system we based ourselves on an existing framework for traditional reservoir computing systems, the RCToolbox developed by the ELIS department in UGent. Before these results were published some elements of the strategy were already mentioned in [77, 78].

2.1 Single delayed feedback for reservoir computing

2.1.1 Basic setup

Delayed feedback systems possess key properties to be used as a reservoir, nevertheless their architecture is radically different from the one of traditional neural networks. While in networks of randomly connected nonlinear nodes all nodes are accessible from the outside world, delayed feedback systems have only one nonlinear element. In order to provide input to all the virtual nodes that reside in the delay line one has to pass via the nonlinear node at the beginning of the delay line. We start by presenting in Fig. 2.1(a) the basic principle of our scheme. Within one delay interval of length τ we define N equidistant points separated in time by $\theta = \tau/N$. These N equidistant points represent the virtual nodes. The values of the delayed variable at each of the N points define the states of the virtual nodes. The term virtual originates from the idea that no additional transformation happens in the delay line, the states are just time shifted. The separation time θ among virtual nodes plays an important role and can be used to optimize the reservoir performance. The exact choice of this value and the consequences will be explained thoroughly in section 2.1.3.

2.1.2 Input driving

The virtual nodes are subjected to the time-continuous input stream $u(t)$ or time-discrete input $u(k)$, see Fig. 2.2, which can be a time-varying scalar variable or vector of any dimension Q. The feeding to the individual virtual nodes is achieved by serializing the input using time-multiplexing. For this, the input stream $u(t)$ or $u(k)$ undergoes a sample and hold operation to define a stream $I(t)$ which is constant during one delay interval τ, before it is updated. In our approach, every time interval of τ represents another discrete time step. The resulting continuous function $I(t)$ is related to the discrete input signal $u(k)$ by $I(t) = u(k)$ for $\tau k \leq t < \tau(k+1)$. This procedure is illustrated in the first part of Fig. 2.2 and the function $I(t)$ is depicted in Fig. 2.1(i). Thus, in our approach, the input to the reservoir is always discretized in time first, no matter whether it stems from a time-continuous or time-discrete input stream. What is actually injected into the

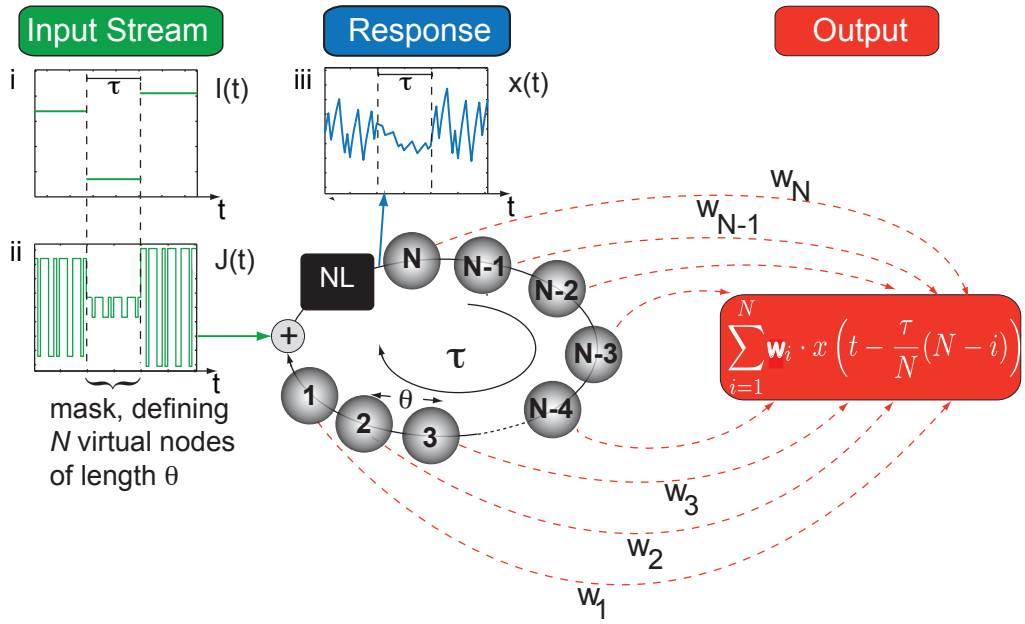


Fig. 2.1: Scheme of single nonlinear node reservoir computer. Along the delay line N states, separated a distance θ from each other, are chosen to represent virtual nodes. (i) 3 discrete samples are each kept constant for a time interval τ to reach all the states in the delay line. (ii) To create more diversity in the states a mask is superimposed on the input. (iii) The transient response of the node. Figure taken from supplementary material of Appeltant *et al.* [17].

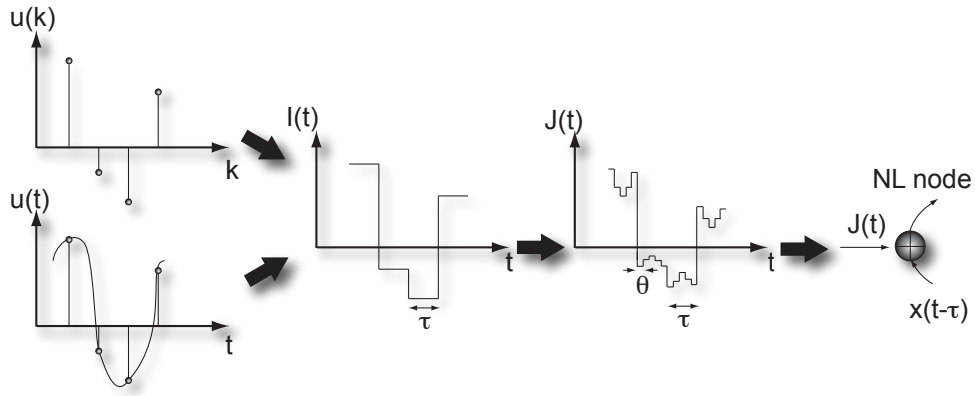


Fig. 2.2: Masking procedure. A time-continuous input stream $\mathbf{u}(t)$ or time-discrete input $\mathbf{u}(k)$ undergoes a sample-and-hold operation, resulting in a stream $I(t)$ that is constant during one delay interval τ before it is updated. The temporal input sequence, feeding the input stream to the virtual nodes, is then given by $J(t) = M \cdot I(t)$. Figure taken from Appeltant *et al.* [17].

nonlinear node is time-continuous again, but from this signal no distinction can be made whether the original data points where coming from a discrete or time-continuous signal.

The challenge of the input feeding gets even bigger when we want to introduce a specific input connection structure. In accordance to what happens in traditional neural network reservoirs, every single virtual node can have its proper input scaling factor. In terms of a 'classical' reservoir setup, these values correspond to the weights of the connections between the input layer and the reservoir layer. In Eq.(1.2), which we repeat here for convenience,

$$\mathbf{x}(k) = F\left[W_{res}^{res} \cdot \mathbf{x}(k-1) + W_{in}^{res} \cdot \mathbf{u}(k-1)\right]. \tag{2.1}$$

These weights were referred to as W_{in}^{res}, which is a random $(N \times Q)$ matrix (we recall that N is the number of virtual nodes and Q the dimension of the input). Every input value sent to a node is firstly multiplied with the factor related to that node. This is done to increase variability in the network. However, the delayed feedback system comprises only one physically present nonlinear node that drives all the virtual nodes in the delay line. Hence all

node states originate from the same transformation and there is no possibility to implement a scaling factor in the virtual node itself. The only option is to imprint coupling weights from the stream $I(t)$ to the virtual nodes by introducing a function $M(t)$, from now on referred to as the mask, as follows: $M(t) = W_{in,i}^{res}$ for $(i-1)\theta \leq t \leq i\theta$ and $M(t+\tau) = M(t)$.
This mask function is a piecewise constant function, constant over an interval of θ and periodic, with period τ. The values of the mask function during each interval of length θ are chosen independently at random from some probability distribution. When the input signal is one-dimensional, the values to be injected are given by

$$J(t) = I(t) \cdot M(t). \tag{2.2}$$

The function $J(t)$ is the product of the input and the mask function and is represented in Fig. 2.1(a) panel (ii). When the input consists of Q values $I^j(t)$, we generate a separate mask $M^j(t)$ for each input j and subsequently they are all summed together. The value to be injected is then given by:

$$J(t) = \sum_{j=1}^{Q} I^j(t) \cdot M^j(t) \tag{2.3}$$

The resulting evolution equations are thus

$$\dot{x}(t) = F\left(x(t), x(t-\tau) + \gamma J(t)\right) \tag{2.4}$$

where γ is an adjustable parameter (usually referred to as input gain).
The output of the nonlinear node is depicted in Fig. 2.1(iii). Because of the mask fluctuations in the input, the node is constantly in a transient regime. To relate the states in the delay line to reservoir states corresponding with an input step, the signal needs to be discretized again. Only then can we map every discrete input step $\mathbf{u}(k)$ is onto a discrete target value $\hat{\mathbf{y}}(k)$ and this for every k. The reservoir state comprises the virtual node states, i.e. the values at the end of each interval θ in the delay line. For the i^{th} virtual node the k^{th} discrete reservoir state is given by

$$x_i^{discr}(k) = x^{cont}\left(k\tau - (N-i)\theta\right). \tag{2.5}$$

Note that this definition implies that the virtual node state is always read out at the end of the interval θ. Although this is the common procedure

throughout this dissertation, other choices of sampling position can also yield valid results.

2.1.3 Interconnection structure

In the delayed feedback system with external input as described in the previous section, we can identify three time scales: the separation of the virtual nodes θ, the delay time τ, and the timescale T of the nonlinear node. We assume that the computational power is hidden in the diversity of the states, therefore we want to keep the system in a transient regime all the time. Therefore we predict good performance when the time scales are related by $\theta \lesssim T \ll \tau$.

If $T \ll \theta$, the nonlinear node reaches its steady-state for each virtual node. In this case the reservoir state $x(t)$ is only determined by the instantaneous value of the input $J(t)$ and the delayed reservoir state $x(t-\tau)$. There is no coupling between virtual nodes and the diversity of the reservoir states goes down. The behavior in this case is illustrated in Fig. 2.3. Fig. 2.3a shows the injected input (blue) and the corresponding output of the nonlinear node that is sent in the delay line (red). The part of the time trace shown here corresponds to one time-multiplexed input value with a binary mask imprinted on it. Because every mask value is kept constant long enough for the system to reach the steady-state, all node states with equal mask values are identical. Regardless of the number of virtual nodes that are tapped from the delay line, with this mask only two different reservoir state values can be used for computation. Fig. 2.3b illustrates the equivalent traditional network of nodes in terms of connectivity. All nodes have a self-coupling, induced by the delayed feedback, but they are not influenced by the states of the other nodes in the network.

When $\theta < T$, the state $x(t)$ of the system at time t depends on the states of the previous virtual nodes. The strength of this dependency is an exponentially decaying function of the separation of the virtual nodes. However, when T/θ is too large, the nonlinear system is essentially not capable of responding to the instantaneous value of the feedback and input. Empirically we have found that for $N = 400$ virtual nodes, $T/\theta = 5$ is the best choice[2]. This leads to significant coupling between virtual nodes, but without too much averaging. A more rigorous approach is applied in Chapter 6, section 6.3 where the effect of the number of nodes, given a certain virtual node separation, is studied. The regime is illustrated in Fig. 2.4. In Fig. 2.4(a) the node output

[2]This optimal value of θ has been determined using the NARMA10 task, as illustrated in Fig. 3.6 in Chapter 3, section 3.3.1.4.

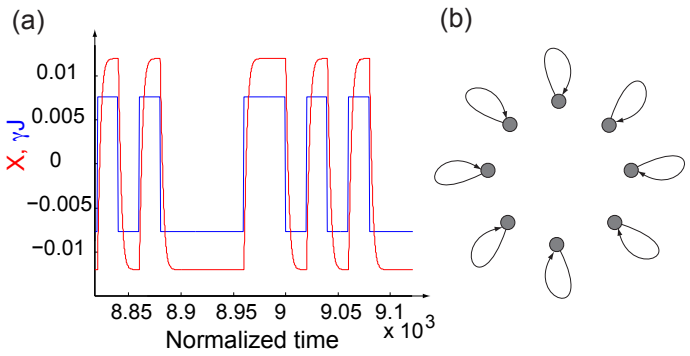


Fig. 2.3: Input time trace for large θ and corresponding interaction graph (a) Input time trace $\gamma J(t)$ (blue) and oscillator output $x(t)$ (red) of our system when the time scale T of the nonlinear system is much smaller than the separation θ of the virtual nodes $T \ll \theta$. Here we choose $T/\theta = 0.05$. The values on both the x- and y-axis are dimensionless. The mask $M(t)$ takes two possible values. For this choice of parameters, the system rapidly reaches a steady-state that is independent of previous inputs. (b) In this regime the system behaves like N independent nodes, each of which is coupled only to itself at the previous time step. Figure taken from supplementary material of Appeltant *et al.* [17]

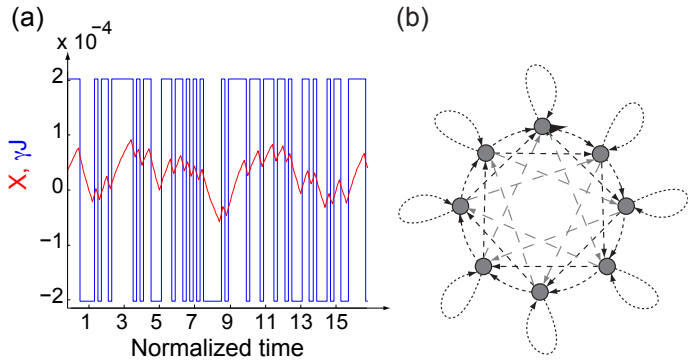


Fig. 2.4: Input time trace for small θ and corresponding interaction structure (a) Input time trace $\gamma J(t)$ (blue) and oscillator output $x(t)$ (red) of our system when the time scale T of the nonlinear system is larger than the separation θ of the virtual nodes $T \gg \theta$. Here we choose $T/\theta = 5$. The values on both the x- and y-axis are dimensionless. The mask $M(t)$ takes two possible values. (b) In this case the system does not have the time to reach an asymptotic value. Therefore, the dynamics of the nonlinear node couples neighboring virtual nodes with each other. Figure taken from supplementary material of Appeltant *et al.* [17]

never leaves the transient regime. This significantly increases the richness of the reservoir states. Since the state of one virtual node depends on the state of the previous ones, because of inertia, the equivalent connectivity graph is given by Fig. 2.4(b). All nodes still experience the self-coupling, but they are also connected to adjacent nodes, with the connection weights exponentially decreasing as we move further back in time.

This relation between the timescales can be used to establish a more formal link between the traditional formulation of reservoir computing, such as given in section 1.3.1 and the interconnection graphs presented in Fig. 2.3(a) and 2.4(b). In contrast to traditional reservoirs where all communication between nodes takes place from one discrete time step to another, in our concept interaction between nodes occurs through inertia of the nonlinear system and through the feedback line. For this reason, the interaction graphs shown in Figs. 2.3(b) and 2.4(b) do not quite correspond to the interconnection matrix W_{res}^{res} used for traditional reservoirs in Eq.(1.2). In what follows, we will derive an approximate interconnection matrix W_{res}^{res} describing the coupling between virtual nodes processing information from different input time steps. For simplicity of notation, in the following we normalize all times with respect to the intrinsic time scale of the nonlinear system T, that is we

work in units where $T = 1$. In this dissertation the we consider nonlinear equations of the form:

$$\dot{x}(t) = -x(t) + F\left[x(t-\tau), J(t)\right] \tag{2.6}$$

with F any nonlinear function and $J(t)$ given by Eq.(2.2). We recall that $J(t)$ is constant over each segment with duration θ and equals $w_{in,i}u(k)$ over the segment containing virtual node i, with $w_{in,i}$ the specific input scaling factor of node i. Assuming a constant value of $F\left[x(t-\tau), J(t)\right]$ during the duration θ, solving Eq.(2.6) yields:

$$x(t) = x_0 e^{-t} + \left(1 - e^{-t}\right) F\left[x(t-\tau), J(t)\right] \tag{2.7}$$

where x_0 is the initial value at the beginning of each interval θ, i.e., the value for the previous virtual node. In particular, the values of the virtual nodes are given by Eq.(2.7) with t replaced by θ. Now returning to the discrete time of input signal $u(k)$, the state of the i^{th} virtual node ($i \in [1, N]$) is reached after a time θ , denoted by $x_{i,k}$. The input to virtual node i at time step k equals $w_{in,i}u_k$. Eq.(2.7) can be written as

$$\begin{aligned}
x_{1,k} &= x_{n,k-1}e^{-\theta} + \left(1 - e^{-\theta}\right) F\left(x_{1,k-1}, w_{in,1}u_k\right) \\
&\cdots \\
x_{i,k} &= x_{i-1,k}e^{-\theta} + \left(1 - e^{-\theta}\right) F\left(x_{i,k-1}, w_{in,i}u_k\right) \\
&\cdots \\
x_{n,k} &= x_{n-1,k}e^{-\theta} + \left(1 - e^{-\theta}\right) F\left(x_{n,k-1}, w_{in,n}u_k\right)
\end{aligned} \tag{2.8}$$

where θ is the separation of the virtual nodes. This equation allows us to recursively compute each virtual node state at time step k only as a function of the input at the same time step k and virtual node states at time step $k-1$:

$$x_{i,k} = \Omega_i x_{n,k-1} + \sum_{j=1}^{i} \Delta_{ij} F\left(x_{j,k-1}, w_{in,j}u_k\right) \tag{2.9}$$

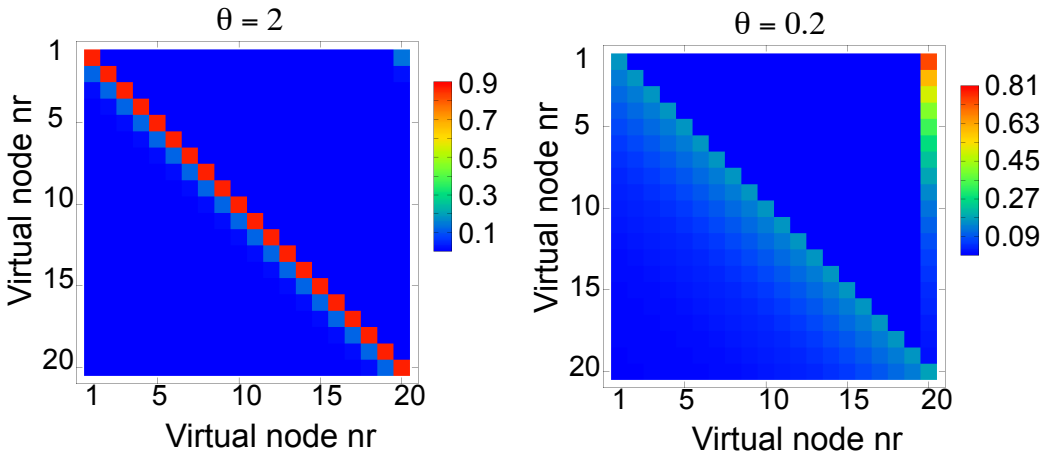


Fig. 2.5: Analytical interaction graphs for large and small θ Interaction graphs for different virtual node separation where we plot the coefficients Ω_i and Δ_{ij} of Eq.(2.9) as a matrix using color coding. For large values of θ (left), the diagonal elements are significantly larger than all others, but when θ decreases (right), the exponential tail of the off-diagonal elements and also the connection to the last virtual node of the previous input step become dominant. Figure taken from supplementary material of Appeltant *et al.* [17]

with

$$\Omega_i = e^{-i\theta},$$

$$\Delta_{ij} = \left(1 - e^{-\theta}\right) e^{-(i-j)\theta}, \text{ with } i \geq j.$$

This equation is our analogue of Eq.(1.2), representing classical reservoirs and it explicitly describes the state coupling between consecutive time steps. However, it differs from traditional reservoirs because the nonlinear functions are applied to the states before the summation is taken. The interaction topology encoded in Eq.(2.9) is similar to that in the recently proposed cycle reservoir [79]. Figure 2.5 illustrates this interaction topology by showing interaction strength matrices for two values of θ. The coefficients Ω_i correspond to the values found in the last column, while the diagonal and off-diagonal elements are given by Δ_{ij} . In terms of traditional reservoirs, this can be related to W_{res}^{res} .

The strongest assumption in this analytical derivation is the fact that the function F is treated as a constant value over the interval θ. To verify whether

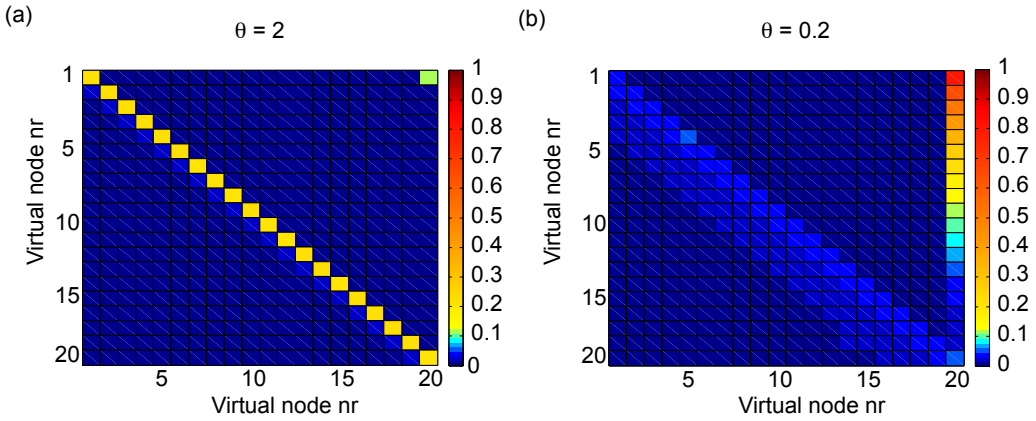


Fig. 2.6: Numerical interaction graphs for large and small θ Interaction graphs for different virtual node separation where we plot the coupling strength between the virtual nodes as a matrix using color coding. For large values of θ (left), the diagonal elements are significantly larger than all others, but when θ decreases (right), the exponential tail of the off-diagonal elements and the also the connection to the last virtual node of the previous input step become dominant.

this approximation is valid we perform a numerical check. While running the reservoir for some random input samples we perturb one of the virtual nodes with a pulse of amplitude 1 and observe how this perturbation is being passed on to other virtual nodes. In this numerical experiment we choose a Mackey-Glass nonlinearity type to fulfill the role of the function F. For further details on this particular nonlinearity type we refer to Chapter 3 where the Mackey-Glass oscillator is used for an electronic implementation of the single node delayed feedback reservoir. Fig. 2.6 shows the interaction strength matrices obtained from numerical simulations. The scaling is expressed in arbitrary units since the obtained values depend on the strength of the pulse and the exact shape of the nonlinear transfer function.

Qualitatively, a confirmation of the analytical result is found. For large values of θ ($\theta = 2$) the self-feedback is the strongest coupling contribution for all virtual nodes. This results in a strong main diagonal in Fig. 2.6(a). When setting θ to a small value ($\theta = 0.2$) the effect of the inertia becomes more important and the off-diagonal elements are more pronounced. Also the coupling with the last virtual node (last column) is strongly present.

2.1.4 Training

The reservoir states themselves are not the desired outcome of the entire system. A training algorithm assigns an output weight to each virtual node, such that the weighted sum of the states approximates the desired target value as closely as possible:

$$\hat{y}(k) = \sum_{i=1}^{N} w_i \cdot x\left[k\tau - \frac{\tau}{N}(N-i)\right],$$

with w_i the weight assigned to the node state of virtual node i, x the output of the nonlinear node and $\hat{y}$ the calculated approximation of the target. the values of the w_i are determined by a linear training algorithm. The training of the read-out follows the standard procedure for reservoir computing [6, 9]. The testing is then performed using previously unseen input data of the same kind of those used for training.

2.1.4.1 Determination of the weights

During the whole process, all weight matrices in Eq.(1.2) remain unchanged. The determination of optimal weight values, the process referred to as training, can be performed either in one-shot (offline) learning or by gradually adapting the weights (online learning). The former approach has been applied in our work. It consists of driving the reservoir with a sufficient number of input samples and recording the node states for each time step. For N nodes and k time steps, the result is a $(N \times k)$-dimensional reservoir state matrix. To this matrix, we add a constant signal to be able to generate a bias for the required output signal. We will refer to the resulting $((N+1) \times k)$ matrix as S, and to the concatenation of all readout weight matrices as W, being a $R \times (N+1)$ matrix, where R is the number of outputs. y designates the $R \times k$ matrix corresponding to the desired output. The aim is to minimize the mean square error $\parallel WS - \mathbf{y} \parallel^2$. This can be obtained by choosing

$$W = \left(\mathbf{y}S^{\dagger}\right)^{T} \tag{2.10}$$

Here † denotes the Moore-Penrose pseudo-inverse[80], which allows to avoid problems with ill-conditioned matrices. After the training stage, the performance of the system is evaluated by applying previously unseen input signals to the reservoir (the testing stage).

2.1.4.2 Overfitting

In order to avoid overfitting to the training data, regularization is commonly used. This is a technique to avoid complexity by implicitly or explicitly penalizing models with a large number of parameters. Regularization is performed either by adding some Gaussian noise to the node states during training, or by using so-called *Tikhonov regularization* or *ridge regression*, which minimizes $\| WS - \mathbf{y} \|^2 + \| \lambda W \|^2$ instead. The second term serves the purpose of keeping the weights as small as possible, while still minimizing the error. Smaller weights make the system less sensitive to the specific shape of a training example, allowing a different realization of the same input class to be mapped into the same target category. Both approaches can be used and are expected to yield equivalent results [81]. Regularization complicates the training because the parameter λ, or if Gaussian noise is added in the training procedure, the amplitude of the noise, needs to be optimized first. Therefore, yet another data set is used than the ones used in training and testing.

2.1.4.3 Unbalanced data sets

When a problem consists of a series of inputs that should be categorized into their corresponding target classes, the problem might be unbalanced. This refers to the fact that more examples are present from one class than from the other. When considering a purely binary problem - let's say deciding whether a certain piece of fruit is an apple or an orange - this corresponds to the situation where there are more pieces of one fruit than of the other. However, the problem might also occur in a multi-class situation, even when all classes have an equal number of elements. In that case the definition of the classifier becomes of importance. When going back to the fruit example, this corresponds to having M apples, M oranges and M lemons. Even though all the classes have M examples, the set could still be unbalanced when multiple classifiers are used. There is no problem when one classifier maps all examples directly on their target. However, in some cases, including the isolated spoken digit recognition used in this thesis, many different classifiers will be defined, each one stating whether a particular piece of fruit is a positive (is a member of) or a negative (is not a member of) example for a particular class. Hence for the apple-case there will be M positive examples and $2M$ negative ones. The unequal number of examples for both decision possibilities will cause the linear hyperplane that separates the two categories in feature space to shift closer to the class with many examples. Ideally, the hyperplane should lie in the middle between the realizations of the two classes. A technique to avoid this problem is Fisher relabeling [82], where for a two-class case the positive

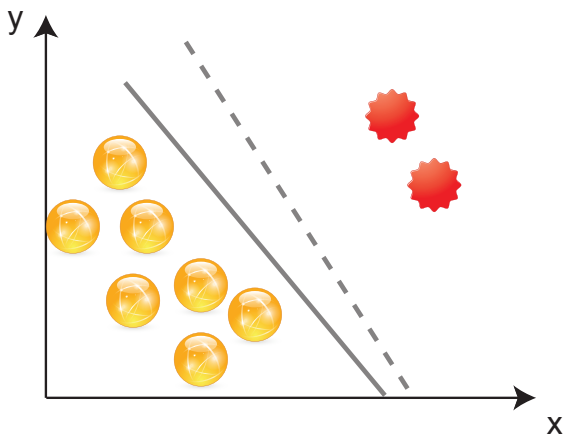


Fig. 2.7: Fisher relabeling. In the case of an unbalanced dataset the separating hyperplane tends to shift towards the class with the most examples present in the dataset. By relabeling the data it is possible to construct a separation closer to the ideal separation line.

and negative targets $\{-1, 1\}$ are replaced by $\left\{\frac{n_1+n_2}{n_1}, -\frac{n_1+n_2}{n_2}\right\}$, with n_1 the number of examples of class 1 and n_2 the number of realizations belonging to class 2. This formula can be extended for multiple classes. By introducing this scaling the unbalance is compensated for and the separating hyperplane shifts more to the middle between the two classes.

2.1.4.4 Framework for simulation and training

For performing numerical simulations that evaluate the performance of delayed feedback systems we can distinguish two major parts. The first part consists of a collection of functions to integrate delayed feedback systems with external input driving. In this dissertation we use a simple Euler or improved Euler integration algorithm, but other methods suitable for delay systems exist. The simulation code preprocesses the input as described above and simulates the reservoir states. When these reservoir states have been generated, they are sent to the second part: training. All training methods and mechanisms we use are independent of the reservoir configuration and they rely on the discrete time reservoir states. We analyzed and adapted existing training methods present in the reservoir computing toolbox, RCToolbox in Matlab v1.0., developed by the ELIS department in Ghent University [83].

2.2 Benchmark tasks

To give an indication on the processing power of a reservoir a wide range of benchmark tasks are available. Each of these tasks requires different key properties to make a correct estimation of the target function. While some are strongly dependent on good linear memory for good performance, others require a strong nonlinear transformation. In this dissertation we limit ourselves to four benchmarks: NARMA, Santa Fe laser data series prediction, sunspot prediction and isolated spoken digit recognition[3].

2.2.1 NARMA

The NARMA task is one of the most widely used benchmarks in reservoir computing. It is an acronym for Non-Linear Auto-Regressive Moving Average and was originally introduced in [84]. Later, it has been used in many other publications in the context of reservoir computing, for instance in [85, 79]. For the NARMA task, the input $u(k)$ of the system consists of scalar random numbers, drawn from a uniform distribution in the interval $[0, 0.5]$ and the target $y(k+1)$ is given by a recursive formula. In literature, two versions frequently occur: NARMA10 and NARMA30, with the number at the end referring to the order of the system to be identified. For NARMA10 the system is defined as

$$y_{k+1} = 0.3y_k + 0.05y_k \left[\sum_{i=0}^{9} y_{k-i}\right] + 1.5u_k u_{k-9} + 0.1 \tag{2.11}$$

and for NARMA30 the equation is given by

$$y_{k+1} = 0.2y_k + 0.04y_k \left[\sum_{i=0}^{29} y_{k-i}\right] + 1.5u_k u_{k-29} + 0.001. \tag{2.12}$$

Here, we will only make use of the NARMA10 task. The input stream $J(t)$ for the NARMA10 test is obtained from u_k according to the procedure discussed

[3]Some tasks that are commonly used in the field of machine learning, but that have not been addressed in this dissertation are: delayed bit parity check and Lorenz data series prediction.

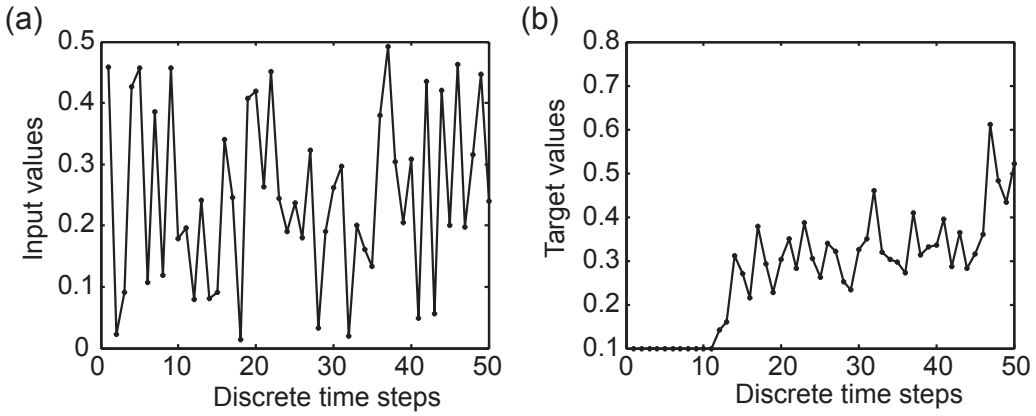


Fig. 2.8: NARMA10 input and target. (a) Discrete points drawn from a uniform distribution within the interval $[0, 0.5]$. (b) Target points calculated from the input points using Eq. (2.11), with the first 10 steps of the target equal to 0.

in section 2.1.2. The input scaling values for the mask consist of a random series of amplitudes of 0.1 and -0.1. The input signal, multiplied with the mask and the input scaling factor γ, feeds the nonlinear node. The term $1.5u_k u_{k-9}$ indicates that memory will play a crucial role in order to obtain good performance on this task. Since in the calculation of step $k+1$ the term u_{k-9} appears, the system needs to have a memory of at least 10 steps. One can still benefit from a longer memory, because of the fact that earlier targets are also used to calculate the next target value. However, such terms are of less importance. The error is expressed as a Normalized Root Mean Square Error (NRMSE), defined as

$$NRMSE = \sqrt{\frac{1}{m}\frac{\sum_{k=1}^{m}\left(\hat{\mathbf{y}}_k - \mathbf{y}_k\right)^2}{\sigma^2\left(\mathbf{y}_k\right)}}, \tag{2.13}$$

where m is the number of time steps in the target function and σ denotes the standard deviation. In Fig. 2.8 an example is given of the input and target for a NARMA10 realization. Fig. 2.8(a) shows some discrete points drawn from a uniform distribution within the interval $[0, 0.5]$. In Fig. 2.8(b) the corresponding target points are shown, calculated according to the tenth-order system from Eq.(2.11). Because of the need of the term u_{k-9} the first target point with a non-zero value is the tenth one. Because of the fact that the NARMA10 target is defined as a sum of only positive inputs, longer time series can diverge. Practically we keep on generating random input series until a realization remains stable within the considered time interval.

2.2.2 Isolated spoken digit recognition

In the isolated spoken digit recognition task, the input dataset consists of a subset of the NIST TI-46 corpus [86, 87] with ten spoken digits (0...9), each one recorded ten times by five different female speakers. Hence, we have 500 spoken words, all sampled at 12.5 kHz. These are all preprocessed using a standard cochlear ear model [88]. The input $u(k)$ (with k the discretized time) for the reservoir then becomes a set of 86-dimensional state vectors (channels). The duration in time of each of these pre-processed spoken digit series can be up to 130 time steps. To construct an appropriate target function, ten linear classifiers are trained, each representing a different digit of the dataset. The target function is -1 if the spoken word does not correspond to the sought digit and +1 if it does. For every approximation of the target the time trace is averaged in time and a winner-takes-all approach is applied to select the actual digit. An example is given in Fig. 2.9 where the averages are shown for the ten classifiers. Ideally, nine of these classifiers would have an average of -1 and one would have +1 as an average. However, the targets have been altered using Fisher relabeling to compensate for the unbalance of the dataset. The one with the highest average is taken as the winner. To quantify the performance, two measures are used: the word error rate (WER) and the margin. While the WER simply characterizes the number of misclassified sample over the total number of samples, the margin denotes the difference in average between the best and the second best guess.

To eliminate the impact of the specific division of the available data samples between regularization, training and testing, we use n-fold cross validation. This means that the entire process of regularization, training and testing is repeated n times on the same data, but each time with a different assignment of data samples to each of the three stages. The reported performances are the average across these n runs.

For the spoken digit recognition task, the mask consists of a random assignment of three values: 0.59, 0.41 and 0. The first two values have each 5% probability of being selected, while the third one has 90% probability of being chosen. Using a zero mask value implies that some nodes are insensitive to certain channels, thus avoiding averaging of all the channels. In terms of traditional reservoir computing this corresponds to a 10% connection fraction of the input weights.

2.2.3 Santa Fe laser data prediction

The Santa Fe laser data prediction task is an example of a one-step time series prediction. The data set we use consists of 4000 points, divided into 4

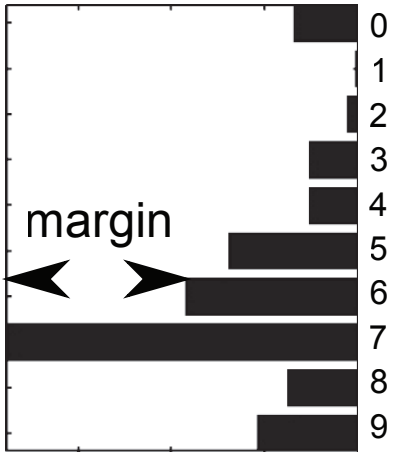


Fig. 2.9: Isolated spoken digit recognition. Classifier result: the approximation of the target time trace is averaged for all time steps and subsequently a winner-take-all approach is applied. The margin is defined as the difference in average between the best guess and the second best guess.

different samples of 1000 points each. The measurements were made on an 81.5-micron $14NH_3$ cw (FIR) laser, using a LeCroy oscilloscope. The setup can be found in ref.[89]. The error is expressed as a Normalized Mean Square Error, defined as

$$NMSE = \frac{1}{m} \frac{\sum_{k=1}^{m} \left(\hat{\mathbf{y}}_k - \mathbf{y}_k\right)^2}{\sigma^2\left(\mathbf{y}_k\right)}. \tag{2.14}$$

In Fig. 2.10 both the input and the target series are plotted. The target equals the input version except for the fact that it is shifted one time step, hence one-step prediction. The discrete input points are connected with a full line and the discrete target points with a dotted one.

2.2.4 Sunspot prediction

Similarly to the Santa Fe laser data task, this is also a one-step time series prediction task. The data set is provided by [90] and consists of 3100 counted sunspots collected from Jan 1749 to April 2007. The error is expressed as an NMSE. The sunspot data series is depicted in Fig. 2.11.

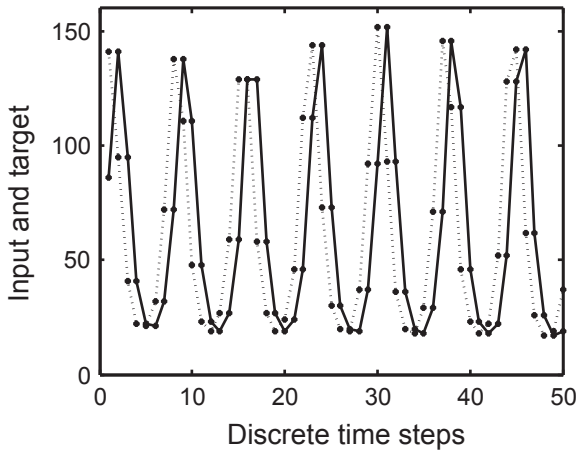


Fig. 2.10: Santa Fe laser data input and target. The discrete points from the original laser data are connected with a full line. The target can be constructed by simply shifting the input series one step forward. The target points are connected with a dotted line.

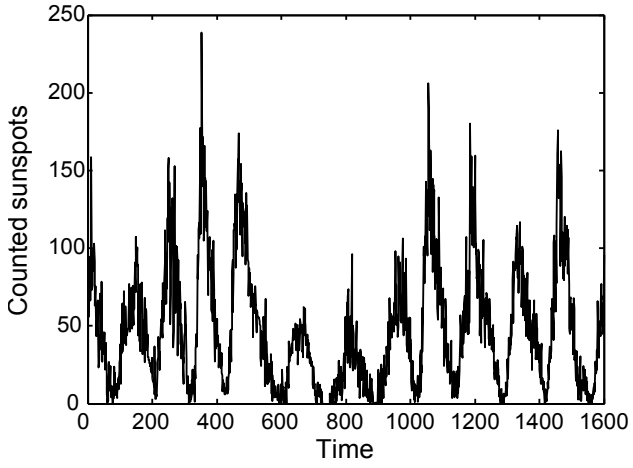


Fig. 2.11: Sunspot data series. The number of counted sunspots is depicted versus time.

2.3 Conclusion

We have proposed delayed feedback systems as reservoir computer. By using only one nonlinear node and a delay line we succeed in creating an analogy with a vast network of nodes.

A first step required to successfully process in put by using delayed feedback systems is an input pre-processing stage. We have outlined a procedure that combines time-multiplexing of the input with imprinting a mask on every input value. This enables us to provide a proper scaling factor for each virtual node.

By defining the mask correctly we can ensure that the node output is always in a transient regime. Moreover, the chosen separation distance between two virtual nodes, defined by the mask, is an important tool to manipulate the interconnectivity between virtual nodes. We have deduced an analytical expression for the interconnectivity weights and validated this numerically.

Finally, we have decided on a way to read out reservoir states from the system. These states can be used for training, using linear training algorithms.

In Chapters 3 and 4 we will model practical implementations of the delayed feedback reservoir concept, evaluating their performance using several benchmark tasks. Next to elaborating on numerical results, the experimentally obtained performance of an electronic and an opto-electronic system are shown.

3 Modeling an electronic implementation

After explaining the general concept in Chapter 2, we now model a practical implementation of the single node delayed feedback reservoir. To demonstrate our concept, we have chosen the widely studied Mackey-Glass oscillator [91], which was originally introduced as a model of blood cell regulation. It has already been used extensively in the characterization of chaotic systems [92]. This choice originates from the fact that this nonlinearity can be implemented in an easy and noise robust way using a simple electronic circuit. Moreover, the shape of the nonlinearity and the strength of the nonlinear contribution can be tuned, allowing for exploration of the optimal settings depending on the task at hand. The numerical simulations have been performed by the author. The experiment was carried out by dr. M.C. Soriano, prof. C.R. Mirasso and prof. I. Fischer at IFISC, Palma de Mallorca, Spain. The author assisted in the input driving and the training of the experimental system. The work reported in this chapter is partly covered in Appeltant *et al.* [17]

3.1 Mackey-Glass delayed feedback oscillator

The equation of the system is given by

$$\frac{dx(t')}{dt'} = \frac{1}{T}\left[-x(t') + \frac{C \cdot x(t'-\tau')}{1 + b^p \left[x(t'-\tau')\right]^p}\right], \tag{3.1}$$

with C being the coupling factor, p the nonlinearity exponent, b a nonlinearity coefficient, T the intrinsic timescale and τ' the delay time. The intrinsic time

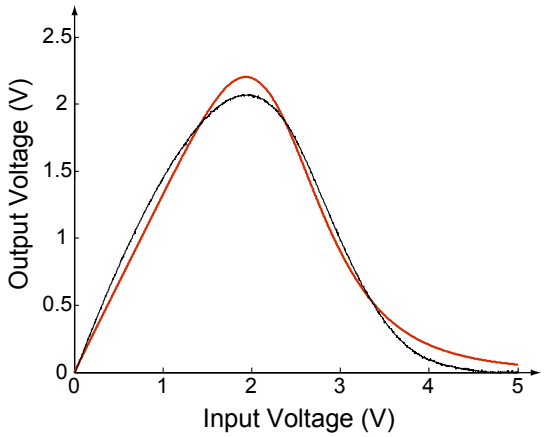


Fig. 3.1: Mackey-Glass nonlinearity shape and experimental fit. Experimental transfer function (black) compared to a fit using the Mackey-Glass equation (red). Fit parameters correspond to $C = 1.33, b = 0.4$ and $p = 6.88$ in Eq.(3.4). Figure taken from supplementary material of Appeltant *et al.* [17].

scale is a measure for the response time of the system and will be used as a reference time scale in the stage where input is added to the equation. The delayed state of x appears in the nonlinearity term on the right hand side, implying that we have nonlinear feedback. The second term between the brackets is the nonlinear term and an example of the shape of this transfer function is given in Fig. 3.1.

When extending Eq.(3.1) to the case where also an external input is injected into the system this equation becomes

$$\frac{dx(t')}{dt'} = \frac{1}{T}\left[-x(t') + \frac{C \cdot [\alpha \cdot x(t'-\tau') + \beta \cdot J(t')]}{1 + b^p\,[\alpha \cdot x(t'-\tau') + \beta \cdot J(t')]^p}\right]. \tag{3.2}$$

The factor α determines how much of the feedback signal is mixed with the input, while the factor β scales the magnitude of the input signal. The mixing of input and feedback signal happens just before the re-injection into the nonlinear node. We have rescaled the variables and parameters in the previous equation to obtain a minimum number of significant parameters, as follows: $\eta = C\alpha$, $\gamma = b\beta$, $X = b\alpha \cdot x$ and $t = t'/T$. This transforms Eq.(3.2) to

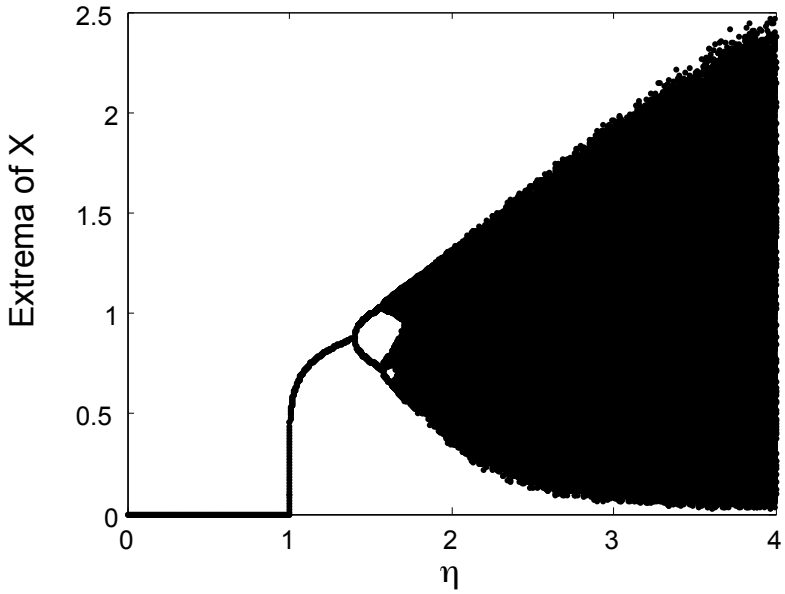


Fig. 3.2: Orbit diagram for Mackey-Glass system given by Eq.(3.3) and $\gamma = 0$. The feedback strength η is varied, while the extrema of the X values are plotted for every value of η. Different dynamical regimes can be observed, for $0 \leq \eta \geq 1$ we find a zero fixed-point that for larger values of η evolves to a non-zero fixed-point, limit cycles and finally deterministic chaos. The delay time was kept constant at $\tau = 80$.

$$\frac{dX(t)}{dt} = -X(t) + \frac{\eta \cdot [X(t-\tau) + \gamma J(t)]}{1 + [X(t-\tau) + \gamma J(t)]^p}. \tag{3.3}$$

with X denoting the rescaled dynamical variable, t a dimensionless time, and τ the delay in the feedback loop. The characteristic time scale of the oscillator, determining the decay in the absence of the delayed feedback term, has now become equal to 1 because of the time normalization. The parameters η and γ represent feedback strength and input scaling, respectively. When numerically simulating the system without input ($\gamma = 0$) we can construct an orbit diagram as the one shown in Fig. 3.2. We investigate the dependence on the feedback strength η, while plotting the minima and maxima of the X variable. This representation allows to easily identify the different dynamical regimes that can be addressed when scanning one parameter of the system.

By adjusting the value of η we can guarantee that the system operates in

a stable fixed-point in the absence of external input ($\gamma = 0, p = 7$). With input, however, the system might exhibit complex dynamics. The choice of this particular nonlinearity has two main advantages. Firstly, it can be easily implemented by an analogue electronic circuit [93], which allows for fine parameter tuning [94]. Secondly, the exponent p can be used to tune the nonlinearity if needed. When p is chosen to be very small, the system becomes very weakly nonlinear. For $p = 0$ the nonlinear contribution disappears completely.

3.2 Experimental implementation

A block diagram of the experimental implementation of reservoir computing, as used in a spoken digit recognition experiment, is shown in Fig. 3.3. Following [93], the Mackey-Glass nonlinearity itself is constructed according to the scheme depicted in Fig. 3.4. The circuit consists of four parts: the nonlinearity, an amplifier and an RC-filter. The nonlinearity was constructed using two field effect transistors, one p-channel and one n-channel, and a resistor. Both of them are coupled with the gate of each transistor connected to the source of the other, resulting in a transfer function that can be fitted to the next Mackey-Glass equation

$$X_{out} = \frac{C \cdot X_{in}}{1 + b^p \, (X_{in})^p}. \tag{3.4}$$

The RC filter is used to determine the time constant of the system, which is 10 ms ($R_4 \cdot C_1$). Connected to the circuit of Fig. 3.4, we added a PC controlled A/D D/A converter (National Instruments 6025E, 200 kSamples/s, 12-Bit A/D conversion). The delay line and the combination with the external input are both implemented digitally in the PC via LabView code. The continuously acquired data are delayed for a time corresponding to the feedback time. The input stream $u(k)$ is converted into the function $J(t)$ by imprinting the mask. Finally, the sum of external input $J(t)$ and delayed output of the circuit are fed into the nonlinearity (FET transistors). In Fig. 3.1 both the experimentally observed transfer curve and the fit to the Mackey-Glass Eq.(3.4) are depicted.

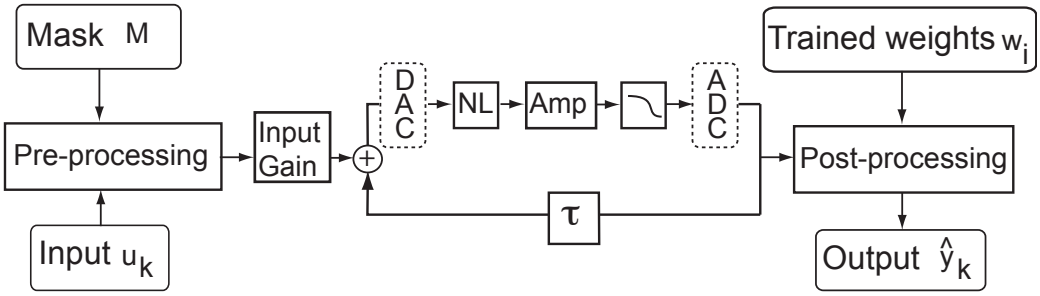


Fig. 3.3: Schematic of the experimental reservoir computer. The Mackey–Glass type nonlinear node is realized as in [93]. The time constant of the system is $T = 10$ ms. The delay loop is implemented digitally by means of Analog to Digital and Digital to Analog Converters (ADC and DAC). The preprocessing to create the input stream $\gamma J(t)$, with γ the adjustable input gain described in Eq.(2.3) , and the post-processing to create the output $\hat{y}(t)$ are also realized digitally. Figure taken from supplementary material of Appeltant *et al.* [17].

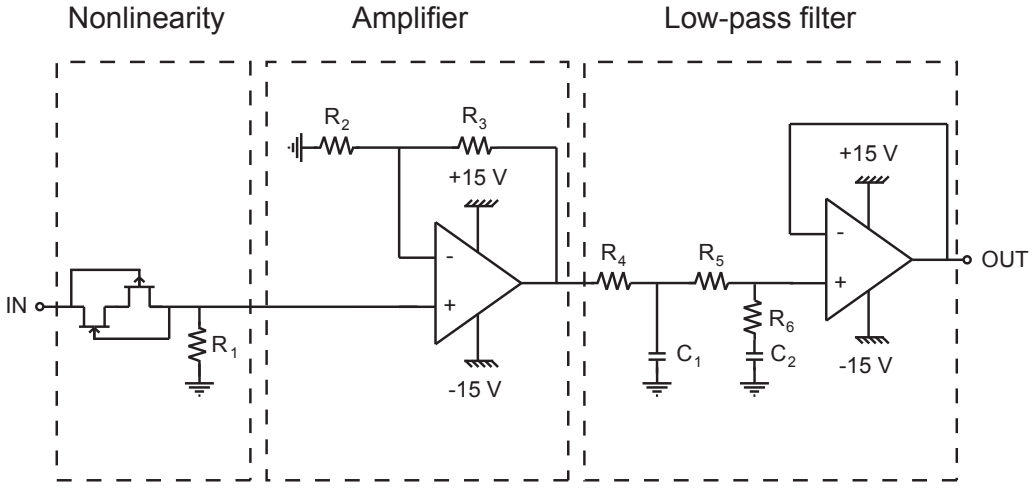


Fig. 3.4: Schematic representation of the hardware node. Two FET transistors, one n-channel (Fairchild 2N5457) and one p-channel (Fairchild 2N5460), and a resistor generate the nonlinear function itself. An amplifier (LM741) provides the desired magnification factor and two RC-circuits allow to set the time constant of the circuit. $R_1 = 507\,\Omega$, $R_2 = 1\,\mathrm{k}\Omega$, $R_3 = 3.7\,\mathrm{k}\Omega$, $R_4 = 100\,\mathrm{k}\Omega$, $R_5 = 5.7\,\mathrm{k}\Omega$, $R_6 = 1.2\,\mathrm{k}\Omega$, $C_1 = 0.1\,\mu\mathrm{F}$, $C_2 = 47$ pF.

3.3 Results

3.3.1 NARMA10

3.3.1.1 Numerically obtained performance

In this paragraph results from numerical simulations are presented, demonstrating the computational capabilities of the Mackey-Glass delay-dynamical node for the NARMA10 benchmark (see Chapter 2, section 2.2.1), a task commonly used in reservoir computing literature [85, 95]. To quantify the performance of the reservoir, the normalized root mean square error (NRMSE) of the predicted versus the value obtained from the NARMA10 model is used. Up to now, the best performance reported in traditional reservoir computing for a reservoir of $N = 400$ nodes, is NRMSE $= 0.099$ [85]. If the reservoir is replaced by a shift register that contains the input, the minimal NRMSE is 0.4. Also for a purely linear reservoir this is the lowest error found for the NARMA10 task. From now on we use this value as an upper bound for good performance. Only when the reservoir performs better than NRMSE $= 0.4$ its implementation makes sense for us. NRMSE values below this level require a nonlinear reservoir. Here, a nonlinear exponent of $p = 1$ is chosen, resulting in a weak nonlinearity. Fig. 3.5 depicts the numerical results in the $\gamma - \eta$ plane. A large region with NRMSE < 0.2 (dark green) has been obtained. The minimal normalized root mean square error obtained is as low as NRMSE $= 0.12$. Numerically, we therefore achieve comparable performance to conventional state-of-the-art RC, but with a much simpler architecture. The light green region ($0.2 <$ NRMSE < 0.3) represents the region where performance is reasonable and the yellow region ($0.3 <$ NRMSE < 0.4) is situated around the performance of a linear reservoir. Finally, the red region depicts the area where reconstruction of the target signal completely fails.

When looking at the left hand side of the figure, a red strip of bad performance can be noted. This region corresponds to very low values of the feedback strength. By re-normalizing the model as was done in Eq.(3.3), the parameter η scales the entire nonlinearity, meaning that for very small η no information is being fed to the nonlinear node. When moving to the right we find a valley of better up to very good performance. It can be seen that the error decreases for smaller values of the input scaling γ. A smaller input scaling results in a smaller region of the nonlinearity that is scanned by the information signal. However, nonlinearity is still present, when a purely linear system is employed the error rises to 0.4. We would like to point out that in order to have the combination of high memory and nonlinear transformation

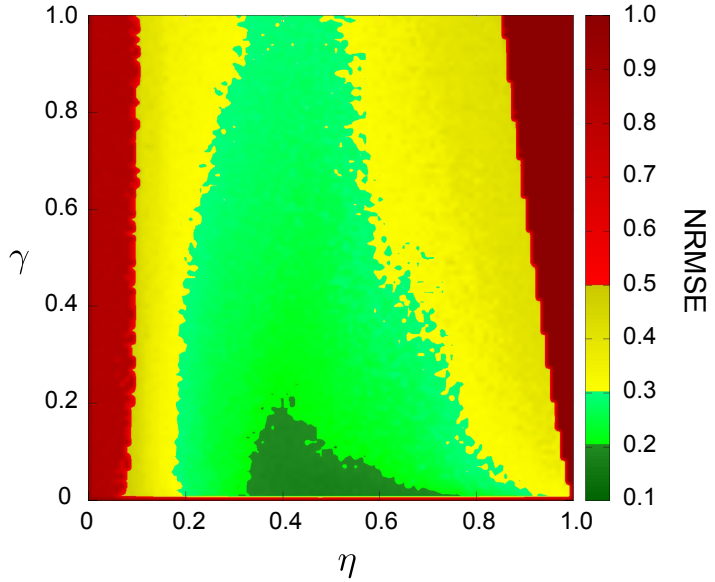


Fig. 3.5: Simulation results for the NARMA10 task. The two scanned parameters are γ (input scaling) and η (feedback strength). The exponent in Eq.(3.3) is set to $p = 1$. Other characteristics of the reservoir are $\tau = 80$, $N = 400$, $\theta = 0.2$). The obtained performance for the NARMA-10 task, expressed as a normalized root mean square error, is encoded in color. Figure taken from Appeltant *et al.* [17]

an extremely high precision is required. For the very optimal point of this parameter scan it goes up to 16 decimal digits. We go more into details on noise effects in Chapter 5. The attentive reader has noticed that the smallest values of γ lead to high errors. This is trivial, when $\gamma = 0$ this implies that no input is being fed to the system at all. When the precision is driven up, the small red region on the bottom of the scan becomes narrower and narrower. On the right hand side another abrupt transition to bad performance occurs. This is caused by a bifurcation where the zero-fixed point solution becomes unstable and a non-zero- fixed point solution appears. When γ increases the bifurcation point shifts steadily to values of η lower than 1. If we repeat this parameter scan for a high exponent, e.g. $p = 7$, the lowest error value found is 0.4, indicating that the reservoir cannot exceed a linear shift register in terms of performance. Hence to solve the NARMA10 task with a delayed feedback reservoir that uses a Mackey-Glass nonlinearity type, we need to work with a low nonlinearity exponent.

3.3.1.2 Experimentally obtained performance

For the NARMA10 task no experiment has been carried out using the Mackey-Glass system. The available setup allows for a tunable nonlinearity, where the exponent can be altered using a potentiometer. However, for lower exponents the quality of the fit between the Mackey-Glass function and the experimentally realized function degrades significantly. Reaching an exponent of $p = 1$ was not possible with the electronic circuit we used. Another reason to only study the numerical model is noise. As will be explained in Chapter 5, this task is extremely noise sensitive. Even numerical noise in simulations can cause an increase in error.

3.3.1.3 Comparison with state of the art

To place the obtained results in perspective, we indicate some results on this benchmark task found in literature. Table 3.1 compares the results found with a delayed feedback reservoir with the performance of network reservoirs from literature.

3.3.1.4 Optimal virtual node separation width

Next to the parameters varied in the scan, also θ plays a crucial role. In Fig. 3.6 we show the numerically obtained performance of the Mackey-Glass system for the NARMA10 test when scanning θ. The optimal point is found for virtual node separations of $\theta = 0.2$, in units of the characteristic time scale of

Table 3.1: NARMA10 performance literature review. For several sets of reservoirs sizes the performance is given as an NRMSE. The first performance column gives the results found by Jaeger *et al. [85]* and the second column the ones obtained by Rodan *et al.* [79]. The final column shows the results found with a delayed feedback reservoir. All are results coming from numerical simulations.

Res. size	NRMSE [85]	NRMSE[79]	NRMSE Our system
20	0.56	-	0.53
50	0.29	0.41	0.35
100	-	0.31	0.27
150	-	0.23	0.21
200	-	0.21	0.18
400	0.099	-	0.12

the nonlinear node. For shorter separations the Mackey-Glass system might not be able to generate a sufficient response to the external input. For larger separations the connectivity among virtual nodes is lost and consequently any memory with respect to previous input. For node separations $\theta > 3$, the NRMSE reaches a level of 0.4, which is the performance of a linear shift-register.

From now on we always use $\theta = 0.2$ in our delayed feedback reservoir. The motivation for this is not only the performance plot from Fig. 3.6, but also the processing speed of the system. Using time-multiplexing to sequentially feed all the virtual nodes in the delay line implies that every discrete input step requires a time τ to be processed. In a setup consisting of N virtual nodes, the next relation holds: $\tau = N\theta$. By choosing a smaller value of θ we also decrease the total delay time and as a consequence we increase the processing speed. Of course, there is also a lower limit for the size of θ. This lower limit can be set by the performance of the system on a certain task, e.g. because smaller θ make the node state dependent on a large range of adjacent virtual nodes. Another issue can be noise robustness. Indeed, when θ is very small, the corresponding excursion of the system becomes smaller as well, bringing the signal closer to the noise level.

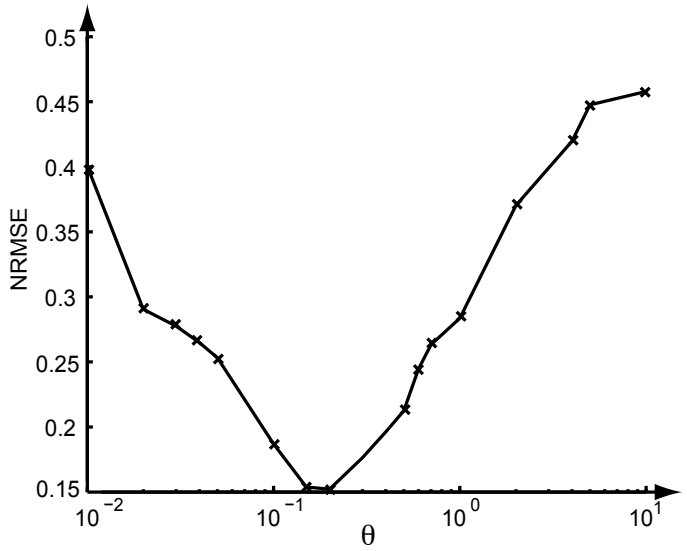


Fig. 3.6: Performance of NARMA10 task as a function of node separation. Plot of the normalized NRMSE for the NARMA10 task (simulations) versus the separation θ of the virtual nodes. Parameters are: $\eta = 0.5, \gamma = 0.05, p = 1, \tau = 400\theta$. Figure taken from Appeltant *et al.* [17]

3.3.2 Santa Fe time series prediction

3.3.2.1 Numerically obtained performance

Here, we show numerical results for the Santa Fe laser task, which was already introduced in Chapter 1, section 1.4.2.3 and in Chapter 2, section 2.2.3. Usually, in literature, performance on this task is expressed as the normalized mean square error (NMSE) of the predicted versus the actual value. In Fig. 3.7 the NMSE values are shown in color coding, while scanning the input scaling (γ) and the feedback strength (η). The exponent is chosen as $p = 1$. Because now in the calculation of the error values the square root is omitted, the reference values of the error are much lower. In the plot of Fig. 3.7 the lowest value is NMSE = 0.019 (corresponding to an NRMSE = 0.137) and the region of good performance is for low values of γ and η. When η increases, the valley of good performance becomes narrower. Note that, although the values of γ are smaller than the ones considered in the case of NARMA10, the part of the nonlinearity that is explored is still reasonably large. While a typical input of NARMA10 belongs to the interval [0,0.5], a Santa Fe data point corresponds to a normalized intensity between 0 and 250. The resulting value injected in the Santa Fe case is therefore still larger than in the case

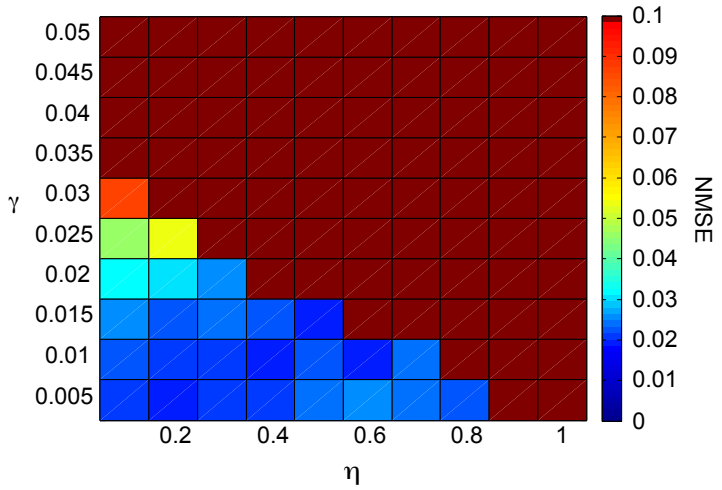


Fig. 3.7: Performance for the Santa Fe time series prediction task. The two scanned parameters are γ (input scaling) and η (feedback strength) and the error is plotted in color code as an NMSE. The nonlinearity is of the Mackey-Glass type and the exponent in Eq.(3.3) is set to $p = 1$. Other characteristics of the reservoir are $\tau = 80$, $N = 400$, $\theta = 0.2$).

of our NARMA10 scan. For the virtual node separation we choose $\theta = 0.2$, which was previously found to be optimal for the NARMA10 task. Also for the Santa Fe time series prediction this seems to be a suitable value.

3.3.2.2 Comparison with state of the art

In Table 3.2 a comparison with literature is made, comparing the delayed feedback approach with a cycle reservoir as described in [79]. Rodan *et al.* achieve slightly better results, but in the same order of magnitude.

3.3.3 Isolated spoken digit recognition

3.3.3.1 Performance: numerical simulations and experiments

For spoken digit recognition, memory is not as crucial as for the NARMA10 task, allowing us to use a higher exponent in the Mackey-Glass equation. This is beneficial since it is easier to implement in an experimental setup. We have verified through numerical simulations that a broad range of values of p yields similar results. The virtual node separation is set at $\theta = 0.2$, while the total number of virtual nodes is $N = 400$. Larger values of θ also yield

Table 3.2: Santa Fe performance literature review. For several sets of reservoirs sizes the performance is given as an NMSE. The first performance column gives the results found by by Rodan *et al.* [79]. The last column shows the results found with a delayed feedback reservoir. Both are results coming from numerical simulations.

Res. size	NMSE [79]	NMSE Our system
50	0.0184	0.0228
100	0.0125	0.0214
150	0.00945	0.0212
200	0.00819	0.0210
400	-	0.0190

good results for this task, but the shortest one is chosen because of speed considerations. The specifics of the training procedure are detailed in section 2.2.2. Fig. 3.8 depicts the numerically obtained classification performance of unknown samples as a function of η for $\gamma = 0.5$, which has been chosen such that input and feedback signals are of the same order of magnitude. The classification performance is expressed in two ways: the word error rate (WER) that shows the percentage of words that have been wrongly classified, and the margin (distance) between the reservoir's best guess of the target and the closest competitor. It can be seen that an increase in margin corresponds to a decrease in WER. Our results show that there is a broad parameter range in η with good performance, with an optimum for both, margin and WER, around $\eta = 0.8$. Note that the performance breaks down when η approaches 1. This is expected, as it corresponds to the threshold of instability of the Mackey-Glass oscillator when there is no input ($\gamma = 0$). At the optimum value of η, we obtain a WER as low as 0.14%. This corresponds to less than one misclassification in 500 words. These performance levels are comparable to or even better than those obtained with traditional reservoir computing composed of more than 1200 nodes for which a WER of 4.3% was reported [13], with a reservoir of 308 nodes for which more recently a WER of 0.2% was obtained [14] and also with alternative approaches based on Hidden Markov Models which achieved a WER of 0.55% [96].

3.3.3.2 Speaker identification: numerical results

A side task that we implemented is the identification of the speaker uttering the digit. The processing of the signal in the reservoir is exactly the same, In the considered data set there are five female speakers, hence identifying them

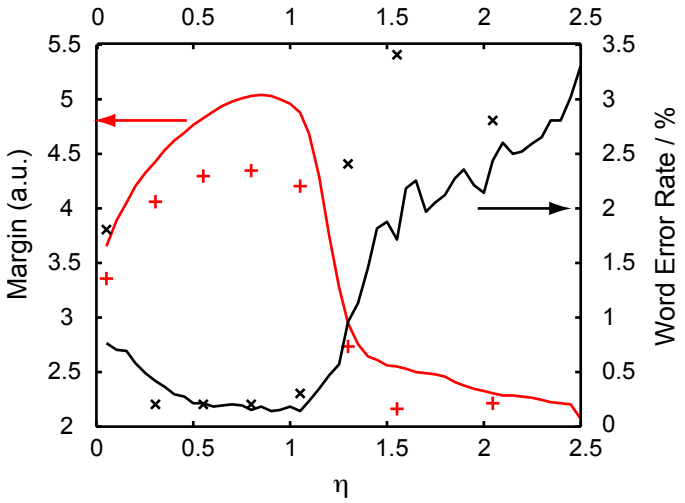


Fig. 3.8: Numerical and experimental results for spoken digit recognition. The y-axis on the left-hand side denotes the margin, whereas the y-axis on the right-hand side denotes the word error rate. The abscissa represents the parameter η. γ has been kept fixed at 0.5 and the exponent is set to $p = 7$. The delay time is set at $\tau = 80$, with $N = 400$ nodes of $\tau = 0.2$ separation. The red line represents results for the numerically obtained margin and the black line represents the numerically obtained word error rate. The red and black crosses denote the corresponding experimental results. Figure taken from Appeltant *et al.* [17].

Table 3.3: Isolated Spoken Digit Recognition performance literature review. For several sets of reservoirs sizes the performance is given as an WER in %. The first performance column gives the results found by Verstraeten *et al. [13, 14]* and the second column the ones obtained with a delayed feedback reservoir. Both are results coming from numerical simulations.

Res. size	WER (%) [13] and [14]	WER (%) Our system
50	7.32	3.0
100	2.96	1.6
150	1.82	1.2
200	1.38	0.8
308	0.2	0.3
400	-	0.14

only requires five extra classifiers. This way the system will process the data, while the readout layer will solve two different tasks on the same reservoir states. When training purely on the cochleagram (before feeding it into the reservoir) the WER in terms of speaker identification is 2%. Although this is already a relatively good result, we can still improve significantly by letting the reservoir process the cochleagram. The obtained error then becomes 0.4% for the speaker task.

3.3.3.3 Comparison with state of the art

As an indication of the quality of the obtained results we give some performance data for this test from literature. In Table 3.3 the first performance column are results listed by Verstraeten *et al.* [13, 14] while the last column shows our results using delayed feedback systems. Again, one can conclude that the single node with delayed feedback performs as well or even better than tradition reservoir simulations.

3.3.4 Sunspot Prediction

3.3.4.1 Numerically obtained performance

Another time series often used as prediction task is the sunspot data series, described in Chapter 2, section 2.2.4. The best result we found on this test corresponds to NMSE = 0.11. This value is similar to what is found in

literature [79]. Usually, in literature, linear nodes perform best on this task and also in our case the results gets better as we explore smaller and smaller parts of the nonlinearity. We note that the performance that can be obtained for this benchmark is strongly dependent in the choice of training and test samples. In our case we used the first 1600 data points for training, the next 500 for validation and the last 1000 data points for testing. In case one would like to introduce cross-validation, the average performance degrades drastically because of the reversed role of training and test data. To avoid this strong dependence on the choice of input data, from now on we only use the Santa Fe laser data as a time series prediction task.

3.4 Conclusion

We have presented numerical evidence that a single nonlinear node with delayed feedback can perform equally well as standard implementations of reservoir computing on several benchmark tasks. The delayed feedback system has a number of parameters that can be adjusted, such as the feedback strength η, the input gain γ, the delay time τ, the separation of virtual nodes in the delay line θ, the type of nonlinearity (in this case the exponent p of the Mackey-Glass system), and the choice of input mask. When comparing this approach with traditional reservoir computing, we can identify some analogues with parameters used in the network approach. The feedback strength and input gain are used to determine the dynamical regime in which the non-perturbed system operates; the delay time is related to the number of nodes and the separation of the virtual nodes to the sparsity of the interconnection matrix; the type of nonlinearity can also be varied as in traditional reservoir computing.

We have successfully modeled an electronic implementation of a delayed feedback system that is used for reservoir computing. This modeling has enabled an experimental implementation, which represents the first hardware implementation of reservoir computing using the delayed feedback approach presented in Chapters 1 and 2. A good accordance between numerics and experiment for the isolated spoken digit recognition task and we obtain results comparable to those obtained with state-of-the-art digital realizations. We found that this task is noise robust. A decrease in margin was observed, but the word error rate remains the same compared to numerical simulations. This experiment, using a Mackey-Glass nonlinearity type, demonstrated that electronic implementations are feasible, even rather simple, and very cost-efficient in terms of components.

In the next chapter we model another experimental implementation. Instead of using only electronics, we extend the concept to opto-electronic systems. This is a step towards an all-optical approach that could enable us to to all-optical information processing in a resource-efficient way.

4

Modeling an opto-electronic implementation

Optics could potentially play an important rule in computing [97, 98, 99]. Also in the field of reservoir computing efforts have been made to combine the computational power of the reservoir computing concept with the potentially very fast signal handling in photonics. For instance, a photonic reservoir approach based on a network of coupled semiconductor optical amplifiers has recently been proposed and simulated at Ghent University [52, 53]. However, considering the physical complexity of the reservoir, employing many nodes is technologically highly demanding and can potentially be a drawback. The delayed feedback approach introduced in the previous chapters is easily implementable using optics. In this chapter, we elaborate on an opto-electronic system, implementing an Ikeda delayed feedback oscillator. We present both numerical and experimental results. The building and testing of the experimental implementation has been done by prof. L. Larger, dr. M.C. Soriano, dr. D. Brunner, prof. C.R. Mirasso and prof. I. Fischer, with theoretical support from prof. J.M. Gutierrez and prof. L. Pesquera. Besides providing the numerical results, the author has been involved in the input pre-processing and the training procedure. The experimental results presented in this chapter have been published in Optics Express [69].

4.1 Experimental implementation

In Fig. 4.1 the scheme we use for the experimental implementation of the opto-electronic delayed feedback system is depicted. Fig. 4.1(a) shows the schematic representation and Fig. 4.1(b) is a picture of the actual setup. The configuration is based on a simple and efficient delay coupled photonic system, which was originally proposed for the exploration of optical chaos [100, 101, 102], as exhibited by an Ikeda ring cavity [103]. Later on Larger

et al. made slight modifications to the setup to use it in the framework of broadband optical chaos communications [101], and highlighted as a system for studying fundamental characteristics and applications of complex dynamics including reservoir computing [104]. The nonlinear element consists of an electronic circuit, a standard telecommunication wavelength diode laser plus an integrated Mach-Zehnder modulator (MZM) providing an electro-optic nonlinear transformation. This transformation can be a $\sin^2$-function or a $\cos^2$-function.

A DFB telecom laser ($1.5\,\mu$m, 20 mW, polarization maintaining fiber pigtail) serves as the optical source, seeding the optical power P_0 into an integrated $LiNbO_3$ Mach-Zehnder electro-optic modulator. This electro-optic modulator is used as a electrically tunable two-wave interference modulation transfer function, $P(v) = P_0 \sin^2[\pi v/(2V_\pi) + \phi]$, according to the voltage $v(t)$ applied to its RF-electrodes. Here V_π is the electro-optic efficiency of the MZM, corresponding to the voltage to be applied for achieving a π phase shift (e.g. leading to the change from a destructive to a constructive interference condition in the optical output intensity). The function $P(v)$ performs the nonlinear transformation of our dynamical reservoir. After the MZM, the signal propagates through a $L = 4.2$km optical fiber delay line, introducing the time delay $\tau = nL/c = 20.87\,\mu$s, where c is the speed of light in vacuum, and n is the refractive index of the SMF28 telecom fiber.

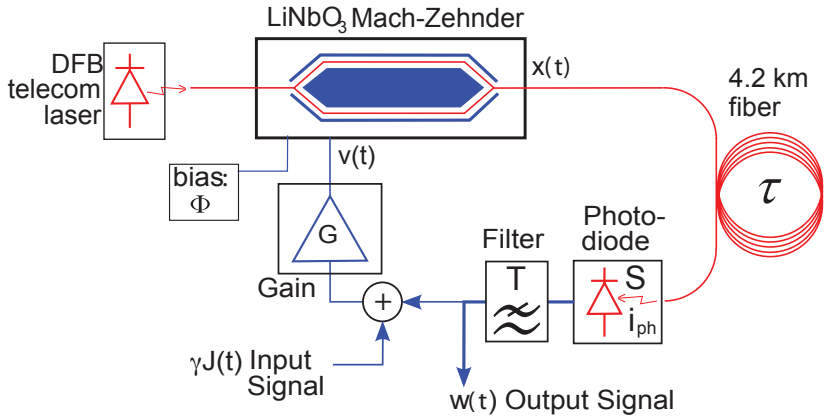
To understand why the Mach-Zehnder has this particular transfer function, we look at Fig. 4.2. The input port of the MZM receives a signal coming from the laser, with power P_0. The power is proportional to the magnitude of the electric field, squared. The signal is split equally over the two arms, leading to the next expression for the field injected in each of the two arms:

$$\mathbf{E}_i = \frac{|E_0|}{\sqrt{2}} \mathbf{1}_E e^{i\omega t_i},$$

with ω the pulsation of the electro-magnetic field and $\mathbf{1}_E$ the polarization. The index i refers to the initial time. The wave traveling along path 1, as indicated in the figure, remains unaffected by the modulation. When it arrives at the end of the separate path it is of the next form:

$$\mathbf{E}_{f,path1} = \frac{|E_0|}{\sqrt{2}} \mathbf{1}_E e^{i\omega t_f},$$

(a)



(b)

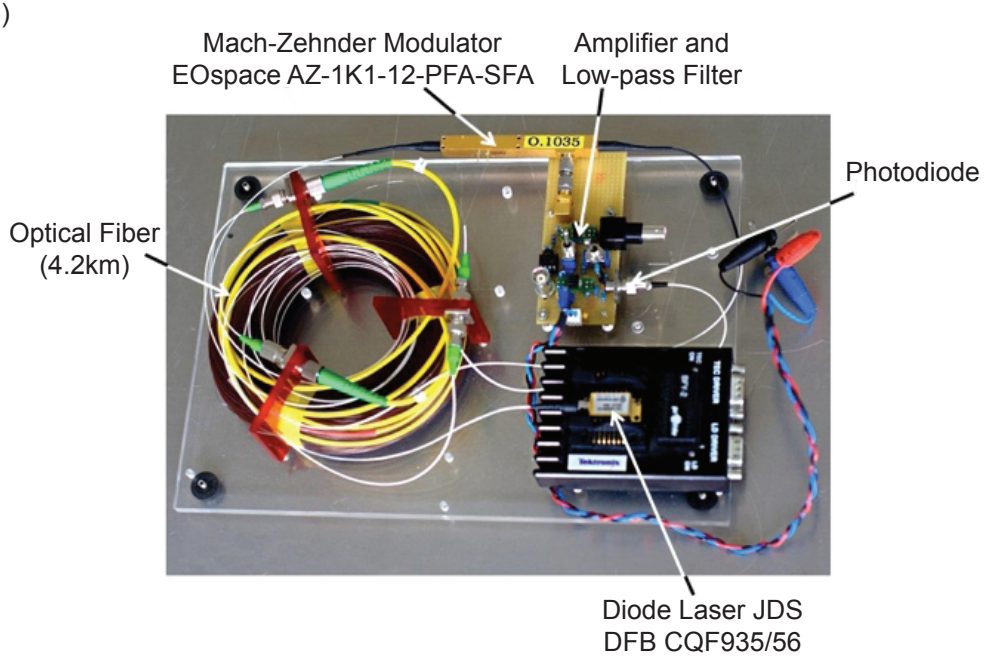


Fig. 4.1: Opto-electronic implementation of reservoir computing. The nonlinearity is realized optically using a Mach-Zehnder interferometer. Subsequently the signal enters the fiber loop that forms the delay line, to be read out in the end with a photodiode. (a) schematic representation of the setup, figure taken from [69] (b) picture of the setup.

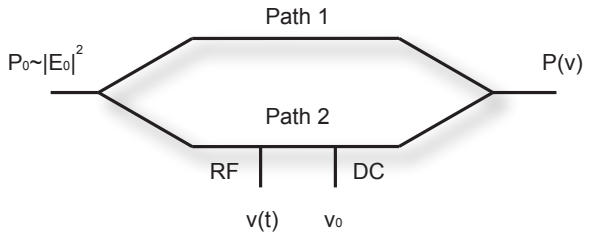


Fig. 4.2: Scheme of Mach-Zehnder modulator. One arm of the modulator is modulated by 2 electrodes, an RF electrode and a DC electrode. The former contains the input and the feedback signal, the latter sets the offset phase of the nonlinearity.

with the index f referring to the final time. The wave traveling along path 2, however, undergoes a modulation by both the RF electrode and the DC electrode. These modulation terms are present in the propagation term of the field:

$$\mathbf{E}_{f,path2} = \frac{|E_0|}{\sqrt{2}} \mathbf{1}_E e^{i\left(\omega t_f + \pi v/(2V_\pi) + \phi'\right)},$$

with ϕ' being the bias introduced by the DC voltage. When combining the two fields again like $\mathbf{E}_f = \frac{\mathbf{E}_{f,path1} + \mathbf{E}_{f,path2}}{\sqrt{2}}$ we get the next expression:

$$\mathbf{E}_f = \frac{|E_0|}{2} \mathbf{1}_E e^{i\omega t_f} \left[1 + e^{i(\pi v/(2V_\pi) + \phi')}\right].$$

What is detected in the end is not the electric field, but the power, which is proportional to the product of the electric field and its complex conjugate, $P(v) \sim E_f \cdot E_f^*$. This leads to the expression:

$$\frac{|E_0|^2}{4} \left[2 + 2\cos\left(\pi v/(2V_\pi) + \phi'\right)\right] = |E_0|^2 \sin^2\left(\pi v/(2V_\pi) + \phi\right), \quad (4.1)$$

with $\phi = \phi' + \pi/2$.

The opto-electronic circuit that closes the loop in Fig. 4.1(a) performs several basic signal processing tasks:

- It converts the optical intensity modulation into a photocurrent modulation according to the photo diode conversion efficiency S. The photo-

detection current thus reads $i_{ph}(t) = S\alpha P\left(v(t-\tau)\right)$ where α accounts for all the optical losses from the MZM output to the photodiode input.

- It acts as a low-pass filter, imposing the time scale T of the system which is normalized to 1 in Eq.(4.2). It filters the corresponding time variation, via a passive RC filter (a capacitor C is connected with a resistor R in parallel), leading to a response time $T = RC$. The voltage across R, $w(t) = R\left[i_{ph}(t) - C\frac{dw(t)}{dt}\right]$ corresponds to the filtered nonlinear delayed feedback signal detected by the setup, also used as the readout signal.

- It adds the input signal $\gamma J(t)$ to the feedback signal.

- It amplifies the sum of the feedback and input data to drive the MZM to allow for sufficient nonlinear operation. This results in a voltage $v = G\left[w + \gamma J(t)\right]$ of sufficient amplitude to achieve dynamical modulation of the MZM of order V_π (thus able to modulate the MZM transmission in a nonlinear fashion).

With the aim of mimicking the nodes of traditional reservoir computing we define virtual nodes along the delay line (4.2 km optical fiber). For a parallel readout one can simply tap the delay line at the nodes' positions to read their states. With the proposed scheme a sequential readout is also possible, making it more practical and ideally suited for an experimental realization. The experimental system provides direct access to key parameters, e.g. the feedback strength η and the phase offset of the MZM ϕ, enabling easy tunability of nonlinearity and dynamical behaviors. Parameter η is controlled via the laser diode power, while ϕ is controlled by the DC bias voltage of the MZM. For the experiments we chose a number of $N = 400$ virtual nodes, a delay time of $\tau = 20.87\,\mu$s, i.e. $\theta = \tau/N = 52.18\,$ns. With the internal system timescale of $T = 240$ ns, a ratio of $T/\theta \approx 4.6$ between the system response time and the virtual node separation is obtained. It is worth mentioning that other values of N and τ yield similar results, as long as the indicated relative scaling is fulfilled. In Fig. 4.3, a typical time trace, acquired from the experimental system, is shown.

In parallel with the experimental work at IFISC, another opto-electronic implementation was experimentally realized at ULB [77, 76]. In this work we will concentrate on the numerical simulations of such a systems.

To evaluate the performance of our system we perform three benchmark tasks, relevant in the machine learning field, of which two experimentally: NARMA10, spoken digit recognition and time series prediction.

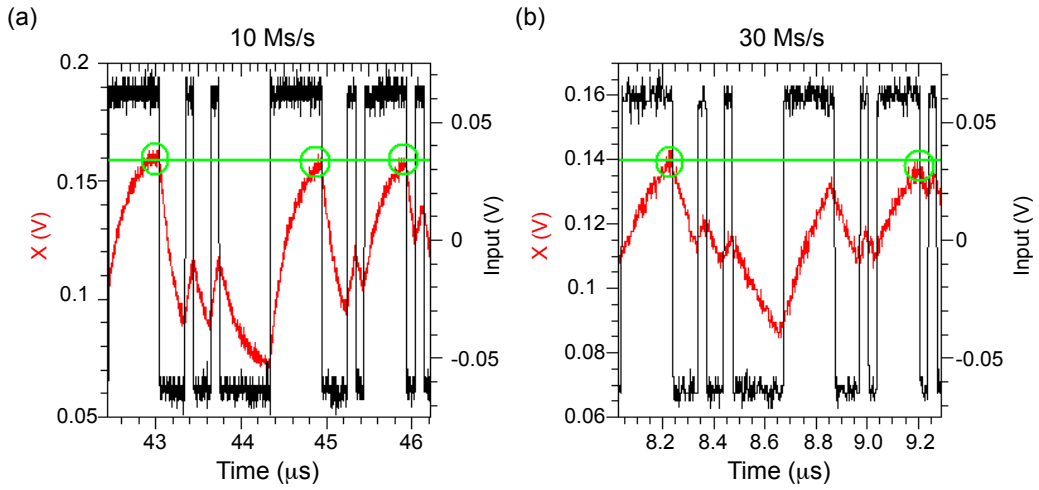


Fig. 4.3: Experimental recording of the injected input (black) and the response of the nonlinear node (red). (a) By varying the mask at 10Ms/s the system is close to reaching the steady-state for some virtual nodes. (b) When varying the mask values at 30 Ms/s the system typically resides in the transient regime. The green circles denote the virtual nodes for which the nonlinear node has approached the steady-state value.

4.2 Ikeda delayed feedback oscillator

The function derived in Eq.(4.1) is known as the Ikeda nonlinearity, a well-studied system [103, 67] that is often encountered in electro-optical systems. When defining a new variable $x(t)$ as $x(t) = \frac{\pi G}{2V_\pi} w(t)$ and introducing $\eta = \frac{\pi GRS\alpha |E|_0^2}{2V_\pi}$, we obtain the following dynamical equation for the system:

$$\dot{x}(t) = -x(t) + \eta \sin^2 \left[x(t-\tau) + \phi\right] \tag{4.2}$$

with η the feedback strength. The nonlinearity can be tuned by biasing the system to a different operating point depending on the phase parameter ϕ. In Fig. 4.4 some experimental measurements of the orbit diagram of the system are shown, scanning the feedback strength. This bifurcation scheme has already been studied in detail in Ref.[105]. Very rich dynamical behavior has been observed depending on the parameters η and ϕ in the large delay regime ($\tau \gg 1$) . The destabilization of the steady-state typically occurs at $\eta = 1$, when the steady-state is at the linear operating point of the $\sin^2$

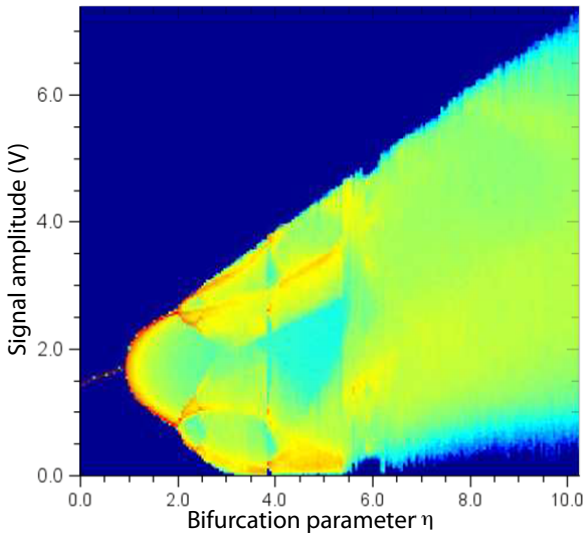


Fig. 4.4: Experimentally obtained orbit diagram of the Ikeda nonlinearity. The orbit diagram when scanning the feedback strength,η, illustrating the possibility of rich dynamical behavior. Courtesy of L. Larger.

nonlinearity in a closed loop configuration ($\phi = -1/2 \pm \pi/4$), usually followed by a period-doubling route to chaos, as η is increased. A high-dimensional chaotic motion is finally obtained for $\eta \gg 1$ (chaotic attractor dimension scaling with η).

In Fig. 4.5 the steady-state is plotted for $\eta = 0.3$ and $\eta = 0.5$ as a function of the offset phase of the nonlinearity. The steady-state solution is obtained by solving Eq.(4.2) in case the derivative is equal to 0. The solution is found by solving the equation:

$$x_{st}(t) = \eta \sin^2 \left[x_{st}(t) + \phi \right].$$

When increasing η towards higher values, the asymmetry of the steady-state solution becomes stronger.

If the systems is driven with an external input $J(t)$, scaled with an input scaling factor γ, the equation becomes

$$\dot{x}(t) = -x(t) + \eta \sin^2 \left[x(t-\tau) + \gamma J(t) + \phi \right]. \tag{4.3}$$

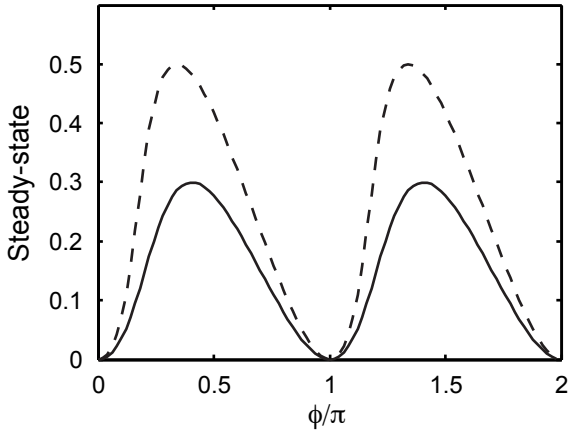


Fig. 4.5: Steady-state solution of the Ikeda nonlinearity of Eq.(4.2). The full line corresponds to the steady-state value for $\eta = 0.3$, while the dashed line shows the steady-state for $\eta = 0.5$

The input $J(t)$ is a time-multiplexed, masked version of the discrete input points originating from the task that needs to be solved.

4.3 Results

4.3.1 NARMA10

4.3.1.1 Numerically obtained performance

When scanning the nonlinearity phase against the feedback strength we obtain the NARMA10 results shown in Fig. 4.6. The input scaling γ is kept constant at 0.01. A clear phase dependence can be observed. The best region can be found for the offset phase chosen between $\pi/20$ and $\pi/4$, where the minimum reached NRMSE is 0.22.

When comparing the results found for the NARMA10 task with the steady-state value as a function of the offset phase and with the nonlinearity of the system evaluated at the steady-state (see Fig. 4.5), we see that the best performance is found on the positive slope of the nonlinearity. The effect of an increasing η on the steady-state is also reflected in the NARMA10 results, where for higher values of η the region of good performance on the positive slope becomes narrower.

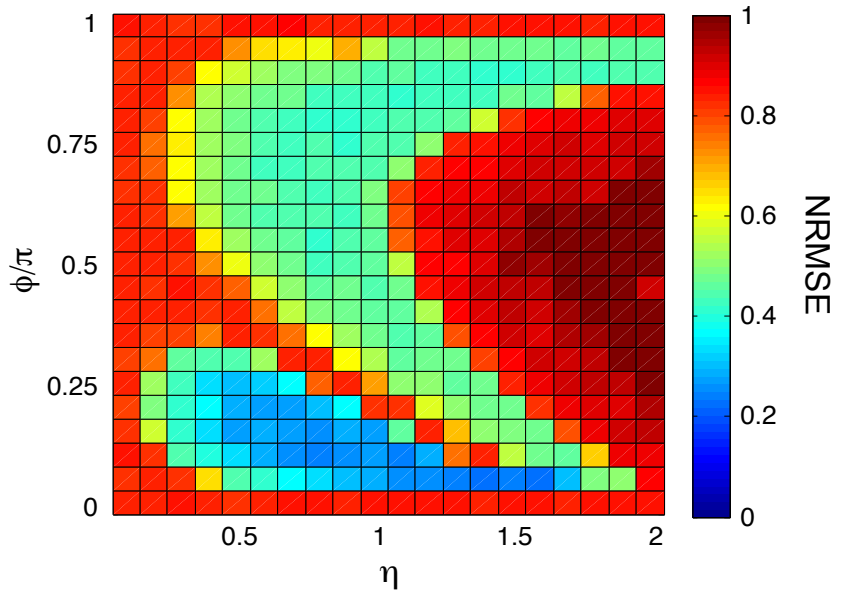


Fig. 4.6: Numerical results on NARMA10 task for Ikeda nonlinearity. The offset phase of the nonlinearity is scanned versus the feedback strength of the system. The input scaling is set to $\gamma = 0.01$.

Note that we do not elaborate on an experimental implementation of the NARMA10 task for this nonlinearity type. The fact that good results can only be found for very small values of the input scaling implies that the experimental implementation suffers too much from noise in the system to obtain a good performance. The effect of noise is studied in detail in Chapter 5.

4.3.1.2 Comparison with state of the art

For quantitative performance results described in literature we refer to Table 4.1. The best performance we obtained in numerical simulations is an NRMSE of 0.22 for 400 virtual nodes. This error is higher than what we found for the Mackey-Glass nonlinearity type. We conclude that in the single node delayed feedback configuration, the Mackey-Glass nonlinearity type yields slightly better results than the Ikeda nonlinearity type.

Table 4.1: NARMA10 performance literature review. The first performance column gives the results found by Jaeger *et al. [85]* and the second column the ones obtained by Rodan *et al.* [79]. The final column shows the results found with a delayed feedback reservoir. using an Ikeda nonlinearity type. All are results coming from numerical simulations.

Res. size	NRMSE [85]	NRMSE [79]	NRMSE Our system
200	-	0.21	-
400	0.099	-	0.22

4.3.2 Santa Fe laser data

4.3.2.1 Numerically obtained performance

Next, the Santa Fe time series prediction task is used to evaluate the performance of this opto-electronic delayed feedback reservoir. The error is expressed as the NMSE between the predicted point and the target. Numerical results are shown in Fig. 4.7.

We vary the input scaling versus the offset phase of the nonlinearity. The performance is clearly phase dependent, however, for small values of γ the phase dependence almost disappears. The region of the nonlinearity that is scanned decreases with γ, making the system less sensitive to the exact shape of the nonlinear transfer function. Given the fact that smaller input scalings will explore only a more linear part of the nonlinearity, one can conclude that to tackle the Santa Fe time series prediction task only a weakly nonlinear processing is required.

4.3.2.2 Experimentally obtained performance

This benchmark has also been solved experimentally. The corresponding results are depicted in Fig. 4.8. For an intermediate feedback strength $\eta = 0.2$ (blue points), a strong dependence of the NMSE on ϕ is found. For $\phi = 0.1\pi$ we experimentally obtain the lowest prediction error with a NMSE $= 0.124 \pm 4 \cdot 10^{-4}$. The experimental value of γ is not explicitly known, but it is estimated to be around 0.02. We note that the numerically achieved results are up to one order of magnitude better, as can be seen in Fig. 4.7 or in ref. [15, 74]. When including quantization noise as is present in the experiment, the performance is of a similar level as the experimental one. To provide evidence that the performance indeed stems from the interplay of high-dimensional mapping and nonlinearity, and not from the nonlinearity

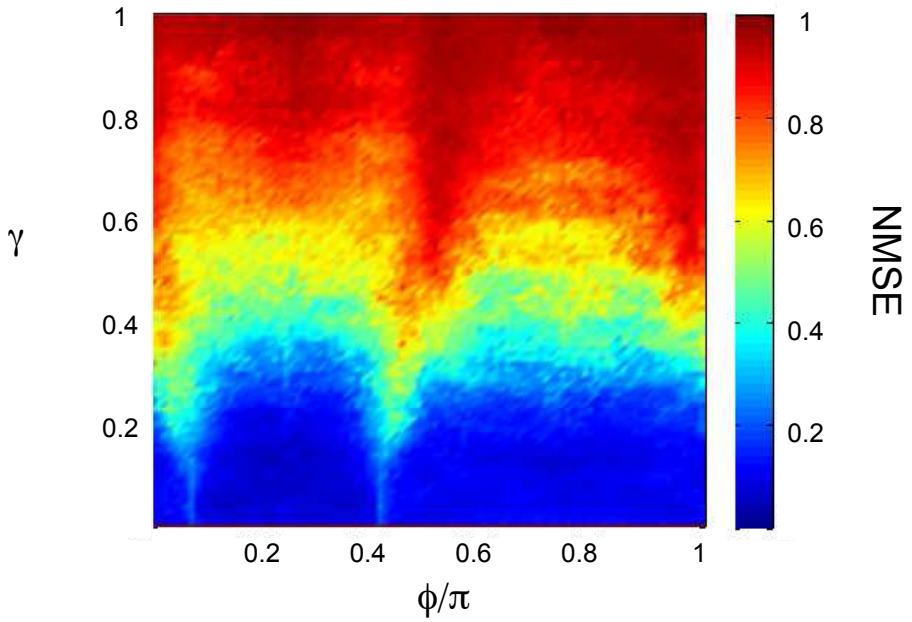


Fig. 4.7: Numerical results for the Santa Fe laser prediction task. The used nonlinearity is of the Ikeda type. The dependence on the nonlinearity offset phase and the input scaling on the reservoir computing performance on the Santa-Fe laser data set is investigated. Best performance for $\eta = 0.2$ is found for several values of ϕ as long as γ is taken small enough. The minimum NMSE is found to be 0.04, comparable to values found in literature when using traditional reservoir computing approaches on this benchmark.

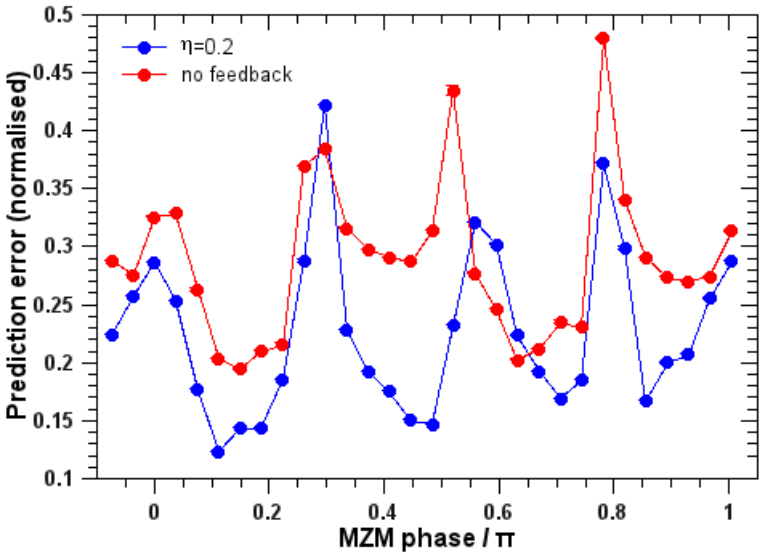


Fig. 4.8: MZM phase dependence of the reservoir computing performance using the Santa-Fe data set. Best experimental performance for $\eta = 0.2$ is found around $\phi_0 = 0.1\pi, \phi_0 = 0.5\pi, \phi_0 = 0.7\pi$ and $\phi_0 = 0.85\pi$ phase values in the vicinity of local extrema of the transfer function of the MZM (see Figs. 4.10(d), 4.10(a), and 4.10(b). Figure taken from [69].

alone, we in addition plot the data obtained when disconnecting the feedback line (red points). The output value of the nonlinear node is sent into the delay line and is used for training. However, they are not fed back into the nonlinear node. The lower performance without feedback loop is clearly visible.

4.3.2.3 Comparison with state of the art

For quantitative performance results described in literature we refer to Table 4.2. Numerically we obtain a slightly worse performance for the Ikeda system than for the Mackey-Glass delayed feedback oscillator. The experimental implementation performs significantly worse. We attribute this mainly to the 10 bit resolution of the experimental setup. Effects of quantization noise are studied in Chapter 5, section 5.4.2.

Table 4.2: Santa Fe performance literature review. The first performance column gives the results found by by Rodan *et al.* [79]. The second and the third column show the results found with a delayed feedback reservoir numerically and experimentally, respectively.

Res. size	NMSE [79]	NMSE Num.	NMSE Exp.
200	0.00819	-	-
400	-	0.04	0.124

4.3.3 Isolated spoken digit recognition

4.3.3.1 Numerically obtained performance

As we did for the case of the Mackey-Glass nonlinearity the TI46 speech corpus is used, see Chapter 2, section 2.2.2. The performance for this task is characterized by the word error rate (WER) as well as a margin to the closest competitor digit. In Fig. 4.9 two simulation results are shown. In Fig. 4.9(a) the input scaling is scanned versus the offset phase, while in Fig. 4.9(b) the feedback strength is scanned versus the offset phase. Contrary to what was observed for the NARMA10 benchmark, for the isolated spoken digit recognition the best operating point is not found at the lowest values of γ. For this test, memory is of less importance and nonlinear transformation plays a more crucial role. For spoken digit recognition the performance is clearly phase dependent and some relatively wide regions are found were the error corresponds to less then 1 mistake out of 500 on average.

In Fig. 4.9(a) we chose $\eta = 0.3$. The region of good performance is widest for $\gamma = 0.4$ or $\gamma = 0.5$. In Fig. 4.9(b) the input scaling is chosen as $\gamma = 0.5$. The offset phase is scanned from 0 to π. In this figure, instead of the WER we plot the log(WER) to have a more clear distinction between regions with good performance and parameter regions that perform less well. We see that the best performance is obtained when the bias point is situated close to a maximum or a minimum of the nonlinearity shape. In this region the nonlinear shape is strongly nonlinear, which implies less memory, but more nonlinear mixing of the input signals. These results confirm that isolated spoken digit recognition relies more on identification of the general shape than on memory of previous inputs.

4.3.3.2 Experimentally obtained performance

The experimental performance of the system can be found in Fig. 4.10. Fig. 4.10(a) and Fig. 4.10(b) show the dependence of WER and margin in the

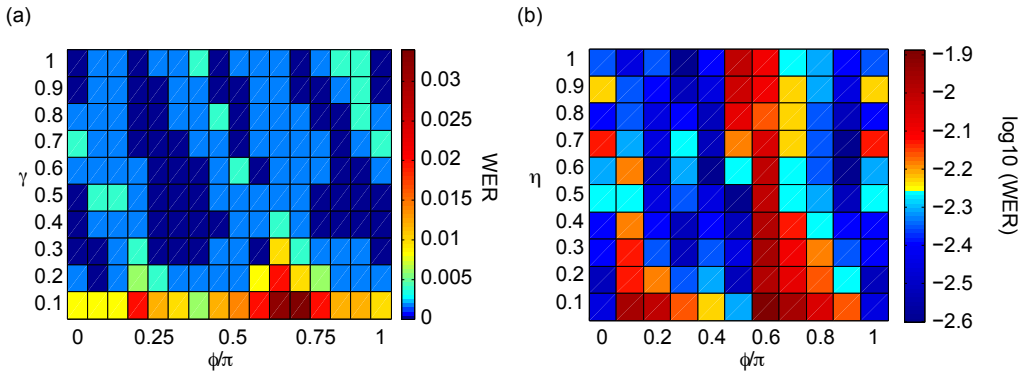


Fig. 4.9: Numerical results on the isolated spoken digit recognition task. An Ikeda nonlinearity is used. The WER is shown in color code. (a) γ is scanned versus ϕ for a constant feedback strength $\eta = 0.3$, (b) η is scanned versus ϕ for a constant input scaling $\gamma = 0.5$.

η-ϕ plane. The best operating point is found to be around $\eta = 0.3$ and $\phi = 0.89\pi$, where after repeated measurements the WER $< 0.02\%$ is reached. Fig. 4.10(c) shows the WER and margin as a function of ϕ for $\eta = 0.3$. It can be seen that good performance is not limited to a single point with the WER remaining below 0.2% for the range $0.75 \leq \phi \leq 0.95$. Fig. 4.10(d) shows the MZM transmission as a function of ϕ. A comparison between Figs. 4.10(c) and 4.10(d) allows an interpretation of the ϕ dependence. At values of ϕ not far from the points of strongest nonlinearity in the MZM response is where spoken digit recognition works best. In contrast the performance dramatically decreases in proximity to a linear response of the MZM.

The numerical results in Fig. 4.9(b) can be compared to the experimental results in Fig. 4.10(a). Both have the WER expressed in logarithmic scale to make the general trend more visible. In both situations valleys of good performance are found at the same positions, $\phi \approx \pi/2$ and $\phi \approx 9\pi/10$. The small differences between the two performance plots can be explained by two points. Firstly, there is the fact that the exact value of γ cannot easily be matched between numerics and experiment. Secondly, in the experimental setup there is the presence of noise, both system noise and quantization noise. As was observed for the Mackey-Glass nonlinearity in Chapter 3, noise decreases the margin. In simulations, small irregularities in the shape of the valley of good performance can be observed because no noise was taken into account. In the experiment the margin is lower and we observe the general trends of good or bad performance.

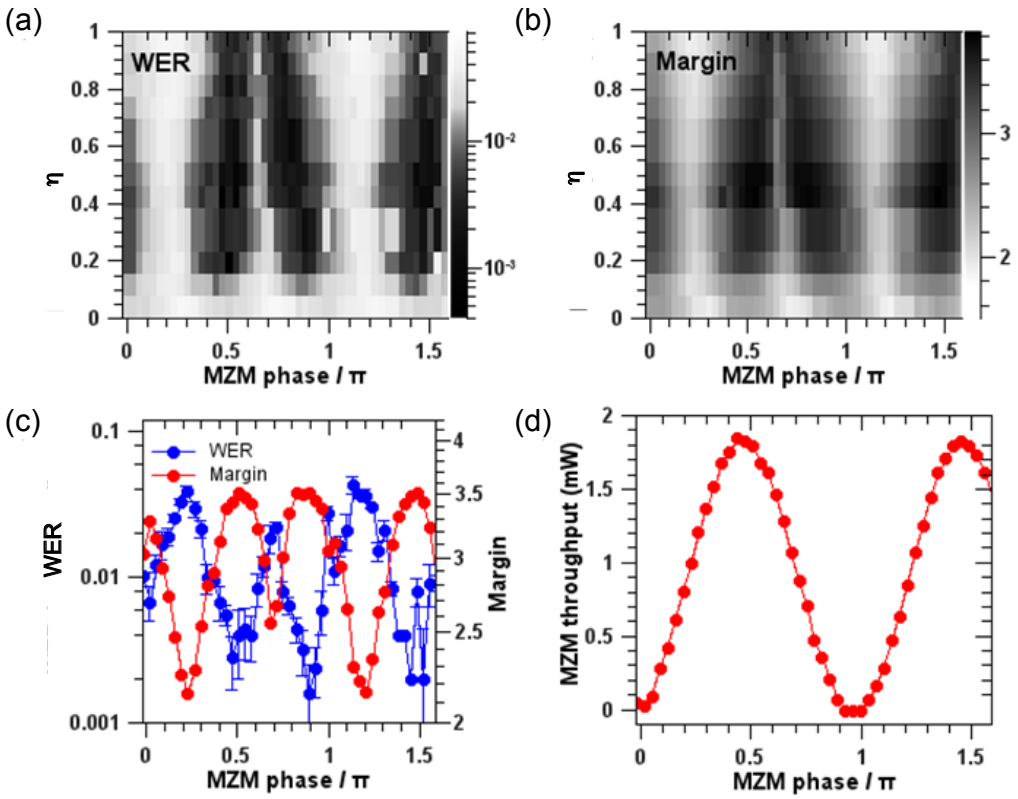


Fig. 4.10: Experimental results on Isolated Spoken Digit Recognition task. (a) and (b) show the WER and margin, respectively, for spoken digit recognition in the $(\eta - \phi)$-plane (feedback strength vs. MZM phase). The two figures of merit show a similar dependency on both parameters, with excellent performance at $\eta = 0.3$ and $\phi = 0.89\pi$. (c) Detailed dependence of the performance on the MZM phase at $\eta = 0.3$. (d) MZM transmission function as a function of phase ϕ. Figure taken from [69].

Table 4.3: Isolated Spoken Digit Recognition performance literature review. The used nonlinearity is of the Ikeda type. The first performance column gives the results found by Paquot *et al. [76]* and the second column the ones obtained with our delayed feedback reservoir. Both results are experimental.

Res. size	WER (%) [76]	WER (%) Our system
200	0.4	-
400	-	0.02

4.3.3.3 Comparison with state of the art

For quantitative performance results described in literature we refer to Table 4.3. The best performance found here is similar to the result found for a Mackey-Glass type nonlinearity, indicating that this task is not so sensitive to the exact shape of the nonlinearity. In the similar experiment performed by Paquot *et al.* at the ULB [76], the obtained performance on the isolated spoken digit recognition task is WER = 0.4% or 200 nodes.

4.4 Bandpass filtering

The DDEs of the nonlinearities studied so far could all be written in the next form:

$$\dot{x}(t) = -x(t) + F\left(x(t-\tau), J(t)\right), \tag{4.4}$$

with $\dot{x}$ the derivative to time, F any nonlinear function, $x(t-\tau)$ the delayed feedback term and $J(t)$ the masked input to the system. The derivative on the left hand side of the equation represents a low-pass filter, which attenuates all frequencies above a certain cut-off frequency. The timescale related to this low-pass filter has been normalized to 1 and it plays a crucial role in achieving good performance on reservoir computing tasks. We have found that the system needs to be driven on a time scale faster than the response time of the low-pass filter, optimally 5 times faster. The richness in different time scales corresponds more variety in connections of the virtual nodes.

In experimental setups another filtering effects, attenuating low frequencies, frequently occurs [106, 107]. This extra high-pass filtering term, in combination with the low-pass filter that was already described, creates a band-pass

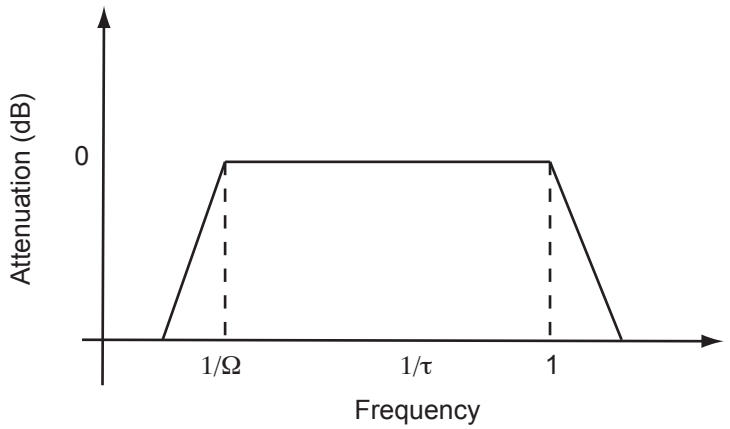


Fig. 4.11: Frequency domain representation. The low cut-off frequency (high-pass filter) is given by Ω^{-1}, while the high cut-off frequency (low-pass filter) is normalized to 1. The inverse of the delay length τ^{-1} is situated in between the two in the above situation.

filter and can have a significant influence on the dynamics of the system. In a delayed feedback reservoir we can benefit form this extra time scale present in the system. It can increase the variation of node states and enhance the mixing among them. To model this effect we extend Eq.(4.4) by adding a high-pass filtering term. The general equation then reads

$$\dot{x}(t) = -x(t) + -\frac{1}{\Omega}\int_{-\infty}^{t} x(s)ds + F\left[x(t-\tau), J(t)\right], \tag{4.5}$$

with Ω the ratio of the timescales of the high-and low-pass filters. Since the response time of the low-pass filter is normalized to 1 in our scheme, Ω represents the normalized response time of the high-pass filter. A schematic representation of the situation in the frequency domain is given in Fig. 4.11. $1/\Omega$ denotes the high-pass cut-off frequency, $1/\tau$ indicates the frequency scale introduced by the delay line and the 1 corresponds to the normalized low-pass frequency. When a high-pass filter is added to the previously used Ikeda delayed feedback system with low-pass filter, described by Eq.(4.3), the dynamical equation becomes:

$$\dot{x}(t) = -x(t) - \frac{1}{\Omega}\int x(s)ds + \eta \sin^2\left[x(t-\tau) + \gamma J(t) + \phi\right], \tag{4.6}$$

where Ω is a new timescale introduced by the high-pass filtering. Adding

a high-pass filter term to the equation implies that the lowest frequencies are eliminated from the system. Hence also the DC terms disappears, resulting in the removal of the non-zero fixed point solution of the low-pass Ikeda Eq.(4.3). Only the zero-fixed point solution remains. To evaluate the effect of high-pass filtering on performance, we now study the example of the NARMA10 task.

4.4.1 NARMA10

We have explored the effect of the Ω on the performance for the NARMA10 task in more detail in Fig. 4.12 where we plot the Normalized Root Mean Square Error (NRMSE) in the phase vs. feedback strength plane. We find that good performance can be achieved when operating close to the inflection points of the nonlinearity around $\phi = \pi/4$ and $\phi = 3\pi/4$. Still the best operation is with a NRMSE of 0.22 is at $\phi = \pi/4$ corresponding to the positive slope of the nonlinearity. The sub-figures correspond to different high-pass time scales in the system.

In Fig. 4.12(a) an asymmetrical performance curve can be seen, resulting from the offset phase dependency of the steady-state solution in case of the low-pass Ikeda system. In the absence of a high-pass filter, the steady-state solutions of the unperturbed system are asymmetrically dependent on η, see Fig. 4.5, what explains the same effect in the performance plot of the low-pass. In Fig. 4.12(b) and Fig. 4.12(c) the non-zero steady state solution no longer exists, hence the symmetry is restored. For $\Omega = 100$, the performance is determined by the nonlinearity shape. The two slopes of the nonlinearity (the positive and the negative one) give rise to an extended region of good performance, with equal results for both slopes. When evolving to the limit situation of $\Omega = 1$, the bias points on the positive slope yield even better performance, while the operating points on the negative slope can no longer perform accurate calculation. By making the high- and low-pass cut-off filters coincide, a very narrow pass-band is created. The parameter region of optimal performance is still centered at both inflection points ($\phi = \pi/4$ and $\phi = 3\pi/4$), but the best error around $\phi = \pi/4$ has improved to a value of 0.17. We conclude that we find better performance for the Ikeda nonlinearity delayed feedback system when band-pass filtering is used. The addition of a high-pass filtering term removes the non-zero fixed point steady-state solution and this way it alters the performance.

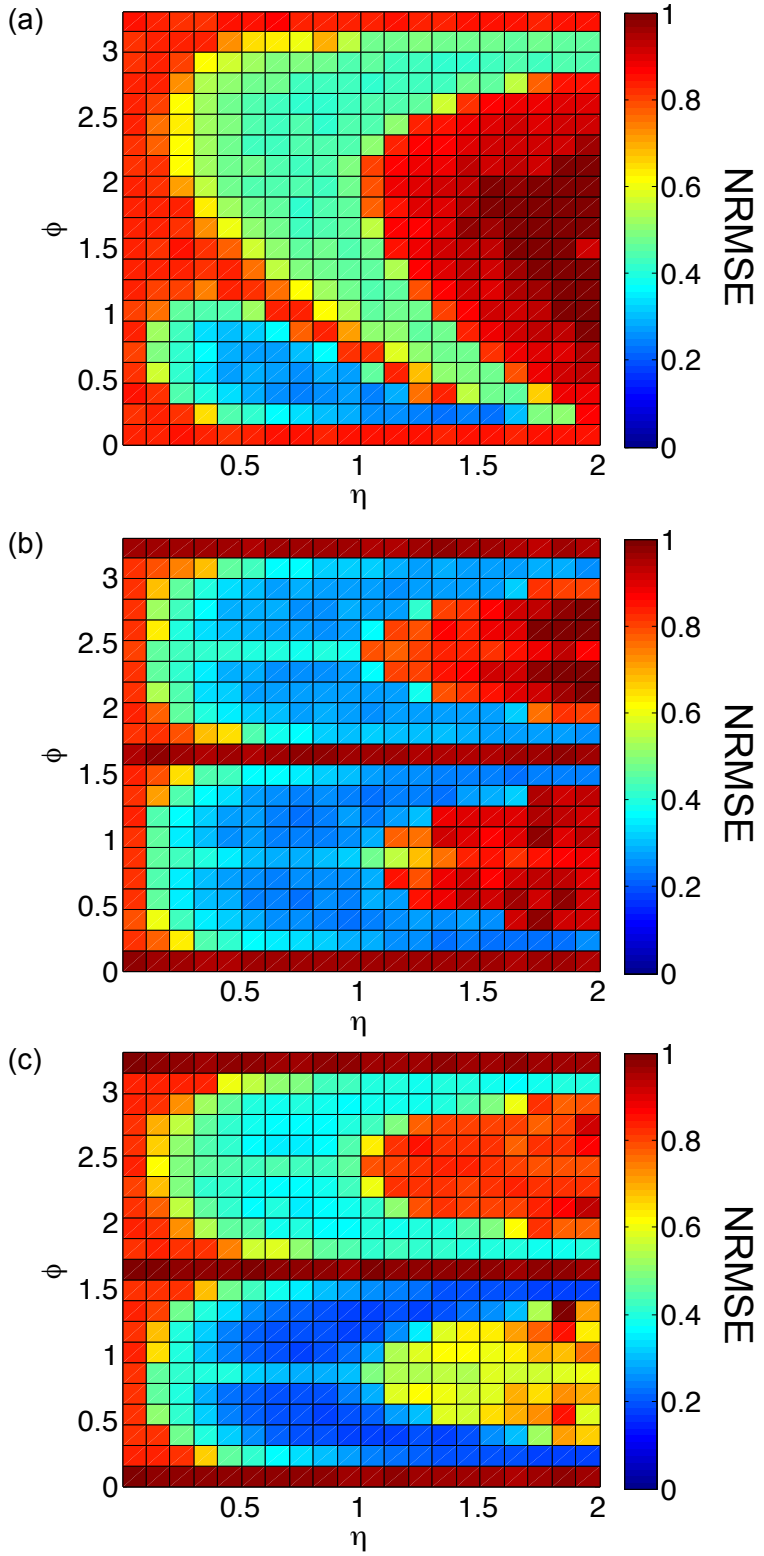


Fig. 4.12: Performance on NARMA-10 benchmark for an Ikeda nonlinearity. The error is expressed in color code as an NRMSE. The number of nodes equals 400 and all nodes are separated by a distance $\theta = 0.2$. γ is set to 0.01. (a) Low-pass situation, (b) band-pass situation with $\Omega = 100$, (c) band-pass situation with $\Omega = 1$.

4.4.2 Interconnection structure

The introduction of an extra high-pass filter introduces another time scale in the system and adds complexity to the dynamics. The node response is integrated over several virtual nodes and this changes the virtual interconnectivity. As depicted in Fig. 4.13, the value of Ω influences especially the connection to the last virtual node in the previous τ interval. While the situation is still very similar for the low-pass case and the band-pass case with $\Omega = 100$, shown in Fig. 4.13(a) and Fig. 4.13(b) respectively, the interconnectivity structure for $\Omega = 1$ shows clear differences.

The connectivity with neighboring virtual nodes can become negative, while in the low-pass situation only positive relation were found. In Fig. 4.14 a time trace of the injected input and the corresponding node response are shown.

In blue the injected input, scaled with γ, is shown, while the red curve represents the response of the nonlinear node. In this simple example only 20 virtual nodes were used, separated by $\theta = 0.2$. For large inputs, the response looks qualitatively very similar to what is obtained with the low-pass filtering. However, in Fig. 4.14 we notice for the second input interval (smaller input amplitude) that the response does not simply follow the mask variations. The contribution of the integral in Eq.(4.5) becomes significantly more important than the input driving.

4.5 Conclusion

The results demonstrated in this chapter prove that the delayed feedback approach as described in Chapters 1 and 2 can be implemented efficiently using an opto-electronic system. First of all, with numerical simulations we have shown that benchmark tasks such as NARMA10, Santa Fe time series prediction and isolated spoken digit recognition can be solved using an delayed feedback Ikeda oscillator. This confirms that the good performance of the Mackey-Glass nonlinearity described in Chapter 3 is not only valid for that particular shape of the transfer function of the nonlinear node. As long the nonlinearity possesses different regions, weakly linear and strongly nonlinear, several tasks can be solved. Depending on the requirements for good performance on a particular task the operating point needs to be chosen carefully.

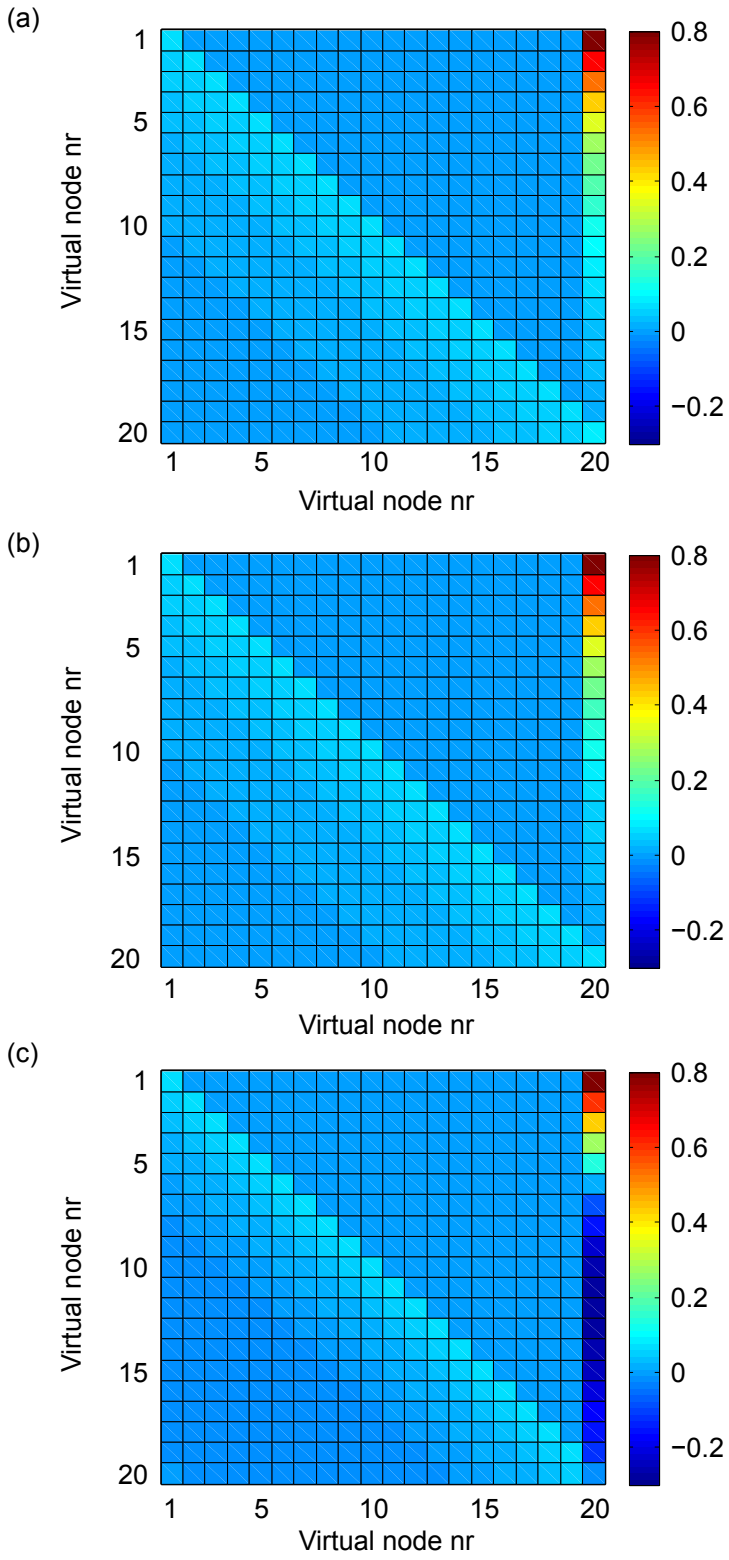


Fig. 4.13: Interaction graph for an Ikeda band-pass system. The interaction graph is shown for a band-pass Ikeda delayed feedback system. The coupling strength is expressed in color code. The virtual nodes are separated by a distance $\theta = 0.2$. γ is set to 0.1 and ϕ is chosen to be 0.5. (a) Low-pass situation, (b) band-pass situation with $\Omega = 100$, (c) band-pass situation with $\Omega = 1$.

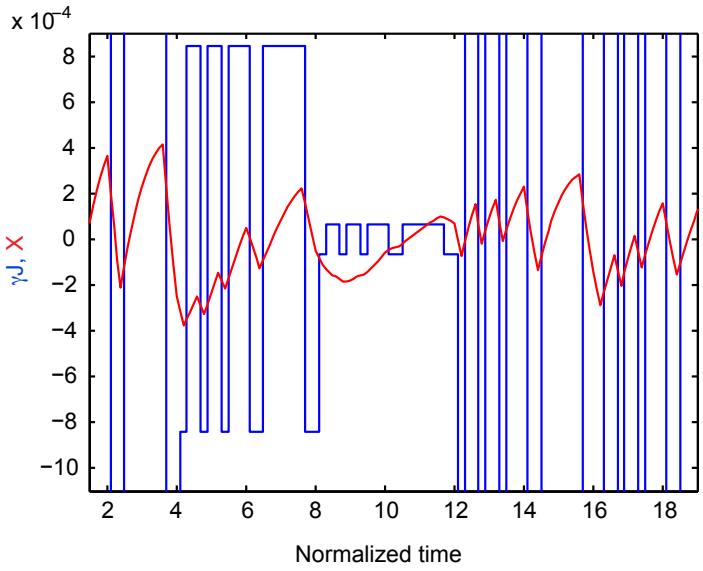


Fig. 4.14: Time trace for an Ikeda band-pass system. The blue curve denotes the injected input multiplied with the input scaling and the red curve shows the response of the nonlinear node The virtual nodes are separated by a distance $\theta = 0.2$ and for this example τ was chosen equal to 20θ. γ is set to 0.1, Ω is equal to 0.1 and ϕ is chosen to be 0.5.

Secondly, the experimental results that have been obtained at IFISC in the Universitat de les Illes Balears demonstrate the first experimental optoelectronic implementation of reservoir computing, able to compete with digital reservoirs in terms of performance. Also the similar experiment at the ULB confirms this[76]. The experiments encourage investigation in more depth of these approaches to optical information processing. They represent a flexible, efficient and potentially low power-consuming device with excellent computational performance. These results should not be limited to an optoelectronic oscillator and might be transferred to an all-optical implementation. This is one of the main goals within the PHOCUS project, however, that is beyond the scope of this dissertation.

Finally, we have shown that a band-pass filtering effect in the setup can be used to increase the complexity of the dynamic response of the node, with beneficial effect on the available region of operation in parameter space.

In the next chapter we will elaborate on general properties of the reservoir that are not benchmark dependent. Instead of evaluating the performance in terms of an obtained error value, we measure the memory and the computational ability of a delayed feedback reservoir.

5

Task-independent properties of delayed feedback reservoirs

As was discussed in Chapters 3 and 4, different benchmark tasks require different properties of a reservoir. While tasks such as NARMA10 and Santa Fe laser data prediction seems to be optimally solved by weakly nonlinear reservoirs, isolated spoken digit recognition benefits from stronger nonlinearity types. Given the specific details of the task in question, this can be explained by e.g. the capability of a reservoir to separate different inputs, to generalize different realizations of the same input to the same class and to recall previous inputs. In this chapter we study some task-independent measures of reservoir performance in the case of a delayed feedback system and apply that knowledge to explain its performance on the NARMA10 task[1]. For this reason we always specifically investigate the parameter region where best performance is found for NARMA10. To characterize regions of good computational ability we make use of quantities such as the kernel quality and the generalization rank. These properties were originally introduced in [18] and have been used to characterize reservoir computing systems in terms of processing power [109]. We remark that all results shown in this chapter have been obtained with a Mackey-Glass nonlinearity type. Similar results can be found for an Ikeda nonlinearity type [108].

[1]This approach was originally proposed by dr. S. Ortin and prof. L. Pesquera from the Universidad de Cantabria [108].

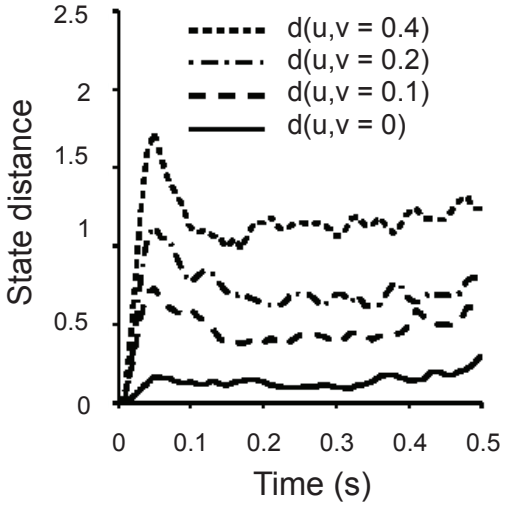


Fig. 5.1: Separation property. The state distance $d(X_u, X_v)$ is plotted as a function of time for several input spike trains with a different distance $d(u, v)$. The full line denotes the distance between two identical spike trains, but different initial conditions of the reservoir. Figure taken from [7]

5.1 Separation property and kernel quality

5.1.1 Separation property

To distinguish different input signals using reservoir computing, they need to be mapped onto different reservoir states. The linear separation property (SP) is a measure for this, describing the distance between two reservoir states that are driven with different inputs, possibly as a function of time. It was originally introduced by Maass *et al.* in 2002 [7]. In their work a computer model for neural microcircuits and spike train inputs is employed to calculate the SP. As inputs a large set of pairs of Poisson spike trains u and v were randomly generated and injected into a reservoir with random initial conditions. For each of these time-varying inputs u and v the response of the reservoir, respectively X_u and X_v, was recorded and for every time step. The distance was measured using the Euclidean distance in the reservoir state space. In Fig. 5.1 the result is shown after averaging for a large amount of different initial conditions.

The separation distance of the states increases with the separation distance of the inputs. The solid line represents a control distance, which should ideally

be equal to 0. This curve denotes the separation distance of states originating from identical inputs, but different initial conditions of the network. The distance found there is significantly lower than the separation distance of the states resulting from different inputs, implying that the distances found for the dashed lines are not simply originating from the initial conditions of the network. If one also wants to consider the timing differences between two spike trains, a Gaussian kernel $e^{-2(t/T)}$ with a width of typically 5 ms is convolved with the two input spike trains. The resulting distance $d(u, v)$ is defined as the distance of the resulting two continuous functions in the L_2-norm (divided by the maximal lengths of the spike trains u and v). This measure of the separation is not used in this dissertation. We use an extended measure of this, called the kernel quality.

5.1.2 Kernel quality

This measure indicates how well the reservoir represents different input streams and it can be used as a measure for the complexity and diversity of nonlinear operations performed. It allows to evaluate to which extent the reservoir facilitates classification performance of a linear decision-hyperplane [110]. While the SP describes the distance between two or more state vectors without considering the separability (see Fig. 1.5), the kernel quality measures how well the reservoir represents different input streams [18]. The state matrix is expected to contain a representation of the inputs that are linearly separable. In order to have an excellent processing capability, the elements of the state matrix should also be linearly separable. To study a reservoir of N nodes, a set of N different inputs is used, $U = \{u_1, u_2, ..., u_N\}$. Every input u_i is a series of k random data points to be injected into the reservoir. The first $k - 1$ points serve to warm up the reservoir, making sure that the system is no more in its quiescent state. A scheme of the situation is depicted in Fig. 5.2.

For the k^{th} point the states s_i of all N nodes are recorded and it forms the i^{th} column in the resulting $N \times N$ kernel matrix. Finally, the rank, which is the number of linearly independent eigenvectors, is used to determine how many input responses can be distinguished using a linear readout. When for the rank $r_{kq} = N$ holds, this implies that all desired target values y_i at step k for the inputs u_i can be returned by the reservoir trained with a linear readout. Hence, a high kernel quality rank is desired. If the rank of this matrix has a value $r_{kq} < N$, r_{kq} can still be viewed as a measure for the computational performance of a network. It gives an indication of the degrees of freedom that are available to map the states onto the targets. In Fig. 5.3 a two-dimensional parameter scan of the kernel quality is plotted for

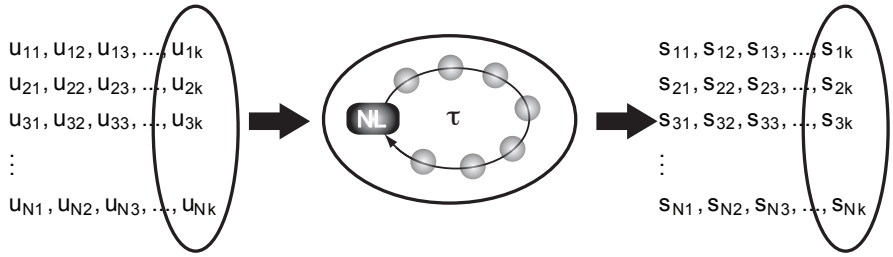


Fig. 5.2: Kernel quality scheme. N input series of k steps are fed into the nonlinear node. Only for the last step the reservoir states are read out and used to construct the kernel matrix.

the Mackey-Glass nonlinearity. We investigate the dependence both on the feedback strength (η) and the input scaling (γ).

In Figs. 5.3(a) and (c) the exponent is chosen equal to 1. Since in the system 400 virtual nodes are employed, the maximum achievable rank is 400 as well. Within the scanned range, a large region of maximum rank is found, especially for higher values of the feedback strength and the input scaling. When zooming to the NARMA10-range, we still observe a large region with a maximal rank. When considering the situation for $p = 7$, depicted in Figs. 5.3(b) and (d), the region of good separation is considerably wider. For low values of η and γ, within the NARMA10-range, the rank is higher for $p = 1$. Within the small window of good performance on the NARMA10 task, the kernel quality is lower for a higher exponent.

5.1.3 Generalization property

Next to kernel quality, another crucial property is the ability to generalize. Non-identical inputs do not imply that they should be classified in different categories. They might differ a little bit because of noise or because of previous inputs. Ideally, a reservoir computing system is able to neglect these very small differences as long as the general pattern belongs to the same category. The generalization rank is in fact the ability of the readout layer to read states from the reservoir and generalize from the training data to the test data. The construction of the generalization rank is similar to the kernel quality rank. Again, a set of N different inputs is used, $U' = \{u'_1, u'_2, ..., u'_N\}$. This time, every input u'_i is a series of $k - l$ random data points and l fixed points at the end of the series, denoted by X in Fig. 5.4.

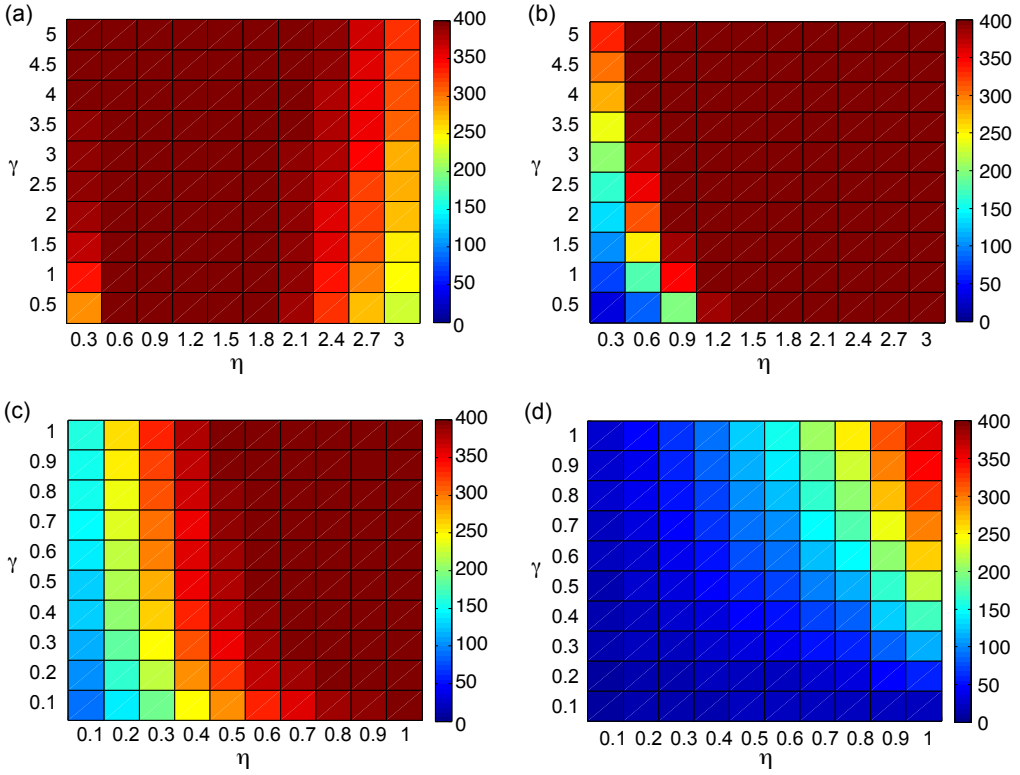


Fig. 5.3: Kernel quality for delayed feedback reservoir with Mackey-Glass nonlinearity. In (a) and (c) the kernel quality is plotted in color code for $p = 1$, scanning both η and γ. In (b) and (d) the same η-γ range is scanned, but for $p = 7$. In both cases 400 virtual nodes are considered, separated by $\theta = 0.2$.

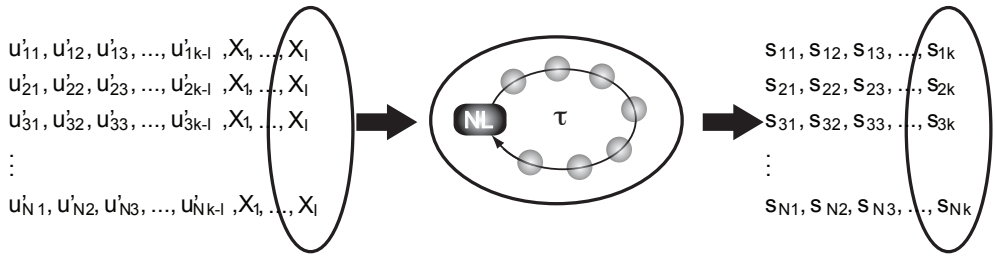


Fig. 5.4: Generalization scheme. M input series of k steps are fed into the nonlinear node. For every sample the last l steps are identical. Only for the last step the reservoir states are read out and used to construct the kernel matrix. A system with good generalization should map all inputs onto the same reservoir sates.

The l points are chosen randomly as well, but they are identical for all u'_i. In order to be able to generalize, the reservoir-readout combination should be able to map the several u'_i onto the same target. The fact that earlier inputs (the first $k-l$ steps of every u'_i) were different, should not affect the fact that the last inputs (the last l steps of every u'_i) are the same and should be mapped onto the same class. In this test the choice of k and, especially, l is crucial. The generalization rank is given by the rank of the $N \times N$ matrix, which is constructed as in section 5.1.2. The better a system is able to generalize and become insensitive to input steps earlier than l steps ago, the lower the rank will be. Hence, for the generalization a low rank is desired. Because of the relevance for the NARMA10 task, in this work l is usually chosen to be 9. The generalization property for a Mackey-Glass nonlinearity type delayed feedback reservoir is depicted in Fig. 5.5 where the influence of the input scaling (γ) and the feedback strength (η) are investigated.

In Figs. 5.5(a) and (c) the situation for $p = 1$ is shown, while Figs. 5.5(b) and (d) correspond to the case of $p = 7$. For $p = 1$, a region of high rank is found, but for higher values of η the generalization becomes better and the rank goes down to 0. The system makes no distinction between the different inputs. When the exponent is high, the generalization rank is maximum for quite a broad range. The rank becomes high at a slightly larger value of η, but contrary to the case of $p = 1$, it stays high for the large values of η within the investigated parameter range. Although both types of nonlinearity clearly have a different behavior in terms of generalization rank, within the parameter range that is important for the NARMA10 benchmark, they yield very similar results.

5.1.4 Computational ability

The kernel quality and the generalization are somehow contradicting properties. We desire a high kernel quality rank and a low generalization rank. However, in systems where a difference in inputs (present or past) is enlarged in the reservoir state space, both of them will usually be high. When shifting the operating point towards regions where the separation becomes stronger, the separation of past inputs becomes stronger as well, rendering the system less able to generalize. Therefore one needs to work in parameter regimes that keep the middle between strong separation and sufficient generalization. To have an indication of the optimal area for processing, the computational ability r_c is used, which is the difference between kernel quality rank r_{kq} and generalization rank r_g:

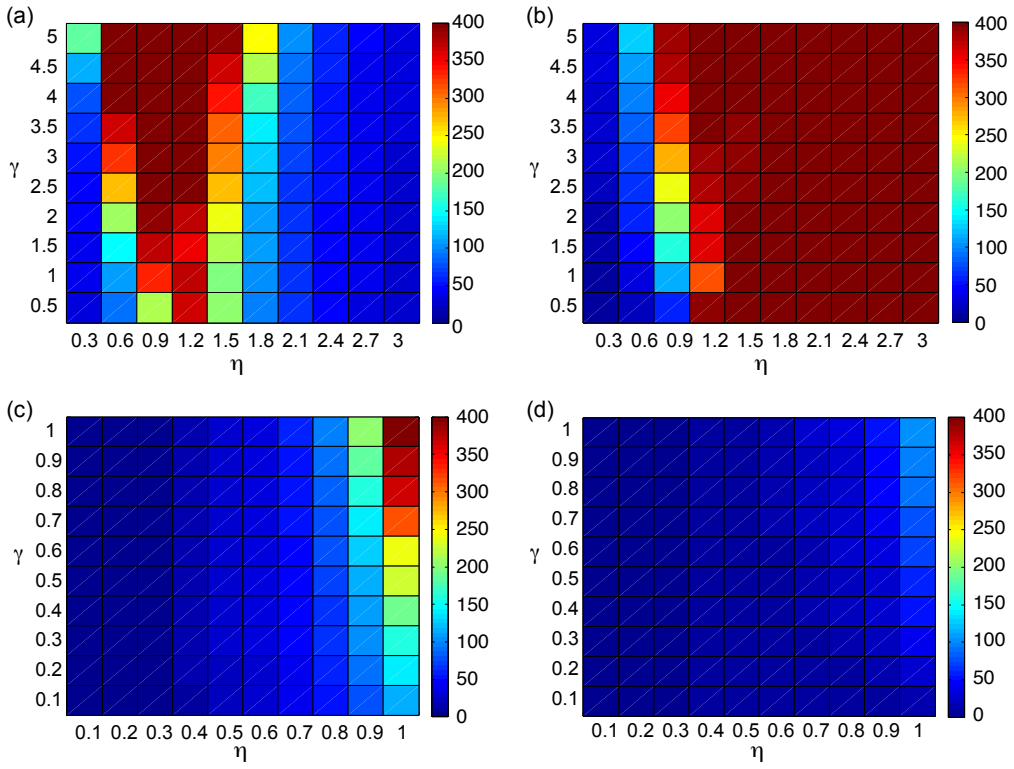


Fig. 5.5: Generalization rank for delayed feedback reservoir with Mackey-Glass nonlinearity. In (a) and (c) the generalization rank is plotted in color code for $p = 1$, scanning both η and γ. In (b) and (d) the same η-γ ranges are scanned, but $p = 7$. In both cases 400 virtual nodes separated by $\theta = 0.2$. We note that (c) and (d) are zooms of the situations depicted in (a) and (b), respectively.

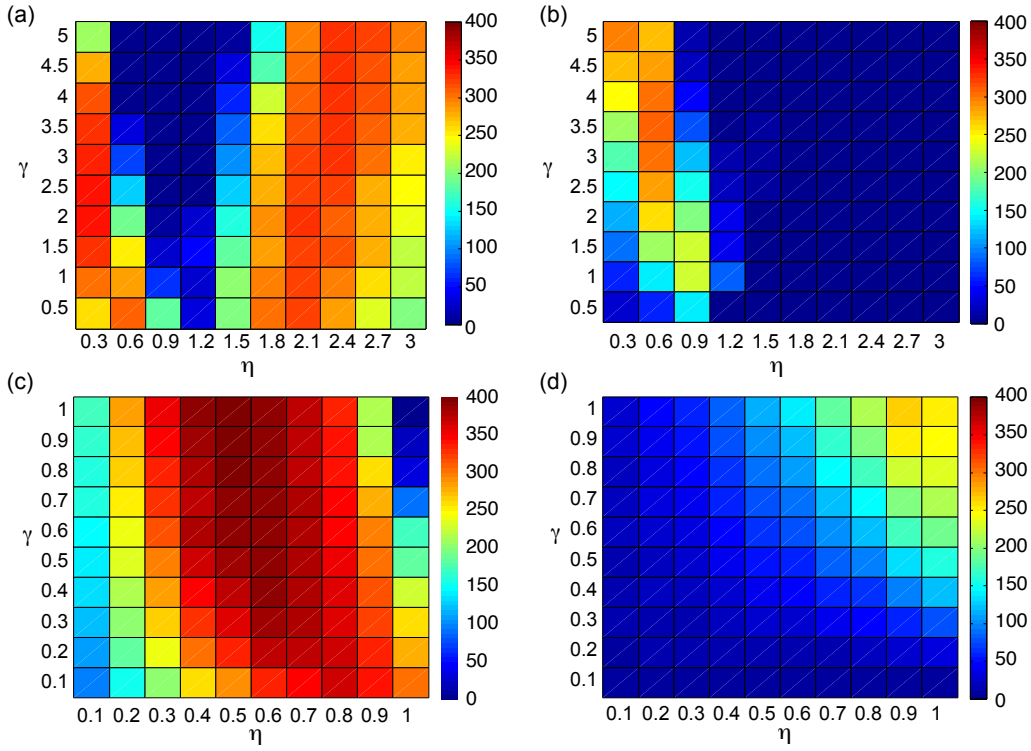


Fig. 5.6: Computational ability for delayed feedback reservoir with Mackey-Glass nonlinearity. In (a) and (c) the computational ability is plotted in color code for $p = 1$, scanning both η and γ . In (b) and (d) the same η-γ ranges are scanned, but $p = 7$. In both cases 400 virtual nodes separated by $\theta = 0.2$. We note that (c) and (d) are zooms of the situations depicted in (a) and (b), respectively.

$$r_c = r_{kq} - r_g. \tag{5.1}$$

Using the data obtained for the kernel quality and the generalization in Figs. 5.3 and 5.5, respectively, we obtain the computational ability using 5.1. The situation is given for $p = 1$ in Figs. 5.6(a) and (c) and for $p = 7$ in Figs. 5.6(b) and (d).

For $p = 1$ two regions of high computational ability are found. One is situated at lower values of η ($\eta < 1$) and the other one is situated at high values ($\eta > 2$). In between, the computational ability rank drops to 0. Also in the zoomed parameter region of the NARMA10 task we find mostly a high rank for $p = 1$. This means that an accurate distinction can be made between

reservoir states originating from different inputs, without being oversensitive to variations in the past. For $p = 7$ this region is less pronounced and much narrower. For values of $\eta > 1$ the computational ability rank decreases rapidly for both exponents and within the zoomed region the rank is low. Considering Figs. 5.6(a) and (b), we can decide that better computational properties are found for the configuration with the weak nonlinearity, $p = 1$. The strong nonlinearity is able to map different inputs onto different reservoir states, but it is not as efficient in generalizing.

This measure can yield an accurate indication of the parameter range for which the system is suitable for computation. However, it only represents one property required for successful processing of an input signal. Another crucial element is memory, the ability to retain previously injected inputs.

5.2 Memory

5.2.1 Memory capacity

Memory of previous inputs can be of significant importance when solving problems. Absence of memory can be sufficient to degrade performance so drastically that the reservoir loses all its prediction or processing abilities. An example of a task with very strict requirements for memory is NARMA10. Since we will focus on this benchmark, we repeat the recursive formula of Eq.(2.11), defining the target y_{k+1} as a function of the input u_k.

$$y_{k+1} = 0.3y_k + 0.05y_k \left[\sum_{i=0}^{9} y_{k-i}\right] + 1.5u_k u_{k-9} + 0.1 \tag{5.2}$$

To calculate the target value of discrete time step $k + 1$ the input of step $k - 9$ is used. This implies that the input value of 10 steps earlier needs to be available for a good target approximation. The recursive summation with earlier target values suggests that even more than 10 steps will be required.

In 2002, Jaeger proposed a test that addresses the memory capacity of a system [111]. Jaeger choose the input $u(k)$ of the reservoir to be points drawn from a uniform distribution in the interval $[-0.8, 0.8]$. However, in this work we will mostly evaluate the memory capacity in order to explain results obtained for the NARMA10 test and therefore the input interval will be $[0, 0.5]$. This scaling with offset has no effect on the intrinsic scoring on the memory capacity. It will only avoid that the input scaling needs to be

shifted in order to compare with NARMA10 performance. The outputs are constructed as an infinite number of output series, y_i, each being a copy of the time series u which is shifted by i steps, hence $y_i(k)$ is a reconstruction of $u(k-i)$ for $i = 1 \ldots \infty$. In practice, the maximum value for i is chosen high enough not make a significant difference in the scoring. It was shown theoretically in [111] that when using a reservoir of size N, the maximum possible memory capacity equals N, the value which can be reached when using a purely linear reservoir. The total memory capacity is defined as the normalized correlation between the approximation of the targets returned by the readout layer and their associated delayed inputs, denoted by the next equation:

$$\mu_c = \sum_{i=1}^{\infty} m_i, \tag{5.3}$$

with m_i being the memory function or normalized correlation between $\hat{y}_i$ and y_i, given by

$$m_i = \text{corr}[y_i(k), \hat{y}_i(k)] = \text{corr}\left[y_i(k), u(k-i)\right].$$

In Fig. 5.7 a memory curve is shown for a delayed feedback reservoir with a nonlinear node of the Mackey-Glass nonlinearity type. We investigate the memory for different values of feedback strength (η) and input scaling (γ). For a small number of delayed steps the reconstruction is excellent and the correlation is close to 1. Starting from a shift of 11 steps back in time, the memory gradually decreases to 0. When integrating the surface under the memory curve we obtain the memory capacity. By repeating this for a large number of parameters we obtain the result depicted in Fig. 5.8.

In Fig. 5.8 the memory capacity for both a delayed feedback Mackey-Glass system with node separation of 0.2, see Fig. 5.8(a), and one with separation of 1.0, see Fig. 5.8(b), is depicted. In both cases the exponent is chosen equal to 1, implying that the nonlinearity is only weak and the transfer function is monotonous. Again, the same parameters are varied, η and γ. For every set of parameters the memory curve is constructed and subsequently the memory capacity is calculated using Eq.(5.3). Both realizations look quite similar, which might indicate that the connections introduced by the inertia of the nodes do not have a significant contribution in creating memory. The memory capacity reaches its highest value for low γ, because in that case a smaller region of the nonlinearity is scanned. A drastic decrease in memory capacity

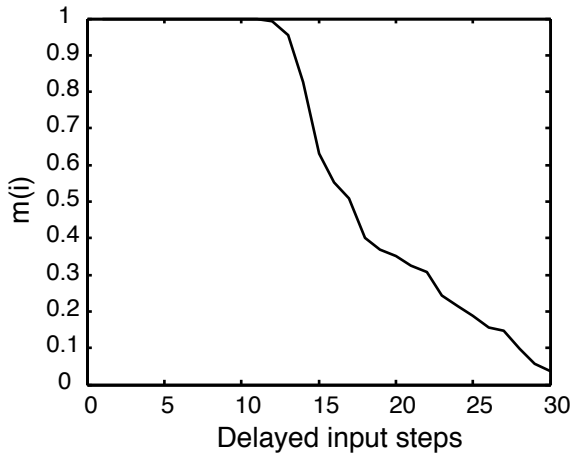


Fig. 5.7: Memory curve for for delayed feedback reservoir with Mackey-Glass nonlinearity. The system is able to reconstruct the delayed input signal perfectly up to 11 delayed input steps. After that the memory decays gradually. The parameters are $\eta = 0.48, \gamma = 0.02, p = 1, \tau = 80$ and 400 virtual nodes with separation distance $\theta = 0.2$.

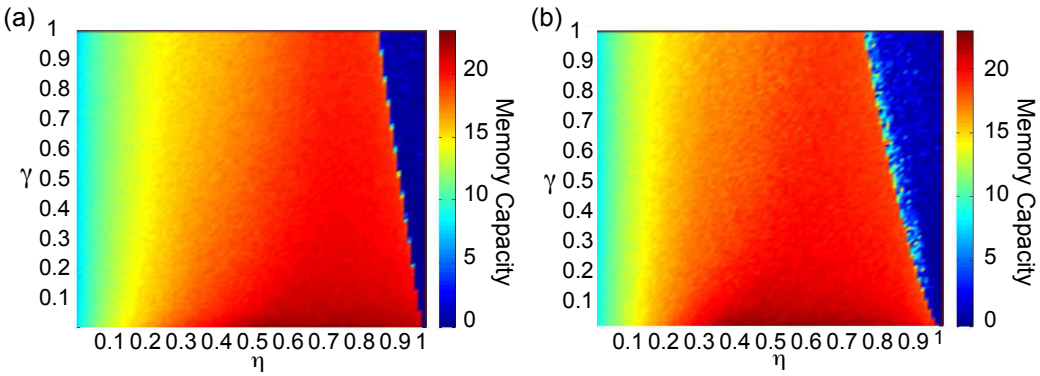


Fig. 5.8: Memory capacity for delayed feedback reservoir with Mackey-Glass nonlinearity, different θ. The memory capacity is shown in color coding, for different values of feedback strength (η) and input scaling (γ). Blue indicates low MC, while a red color corresponds to a high MC of 20 steps or more. The exponent is chosen $p = 1$ and 400 virtual nodes are used for training. (a) $\theta = 0.2$, (b) $\theta = 1$.

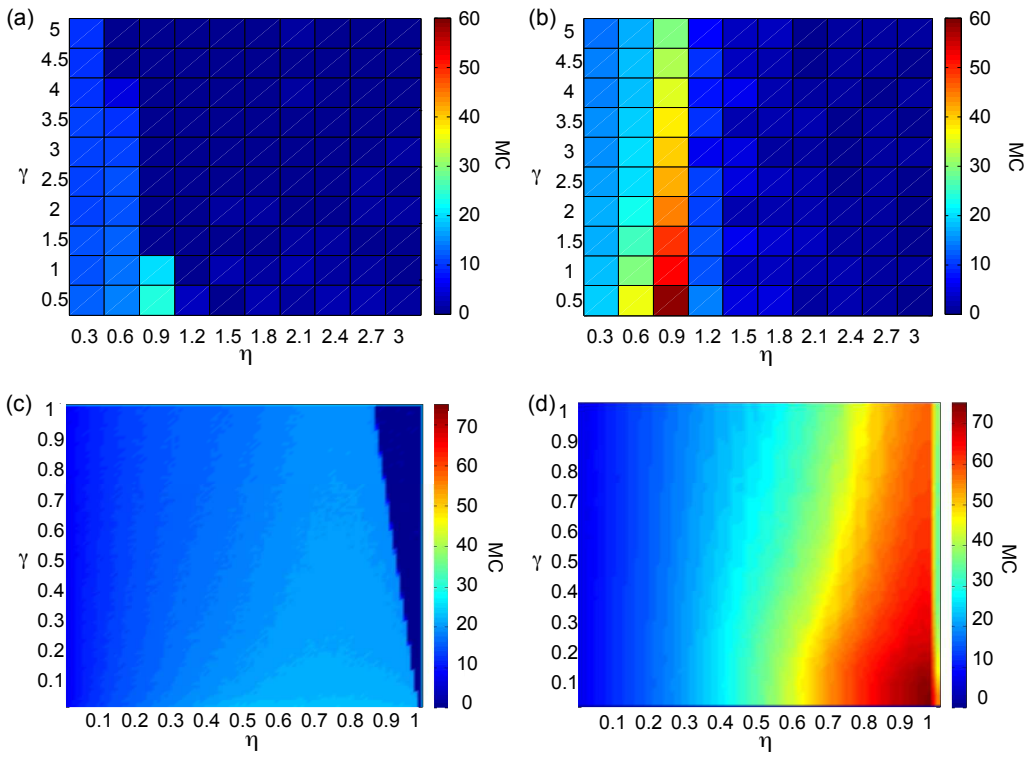


Fig. 5.9: Memory capacity for delayed feedback reservoir with Mackey-Glass nonlinearity, different p. The memory capacity is shown in color coding, while varying the feedback strength (η) and the input scaling (γ). The virtual node separation is chosen $\theta = 0.2$ and 400 virtual nodes are used for training. For (a) and (c) the exponent is chosen $p = 1$, while for (b) and (d) it is set to $p = 7$. We note that (c) and (d) are zooms of the situations depicted in (a) and (b), respectively.

is detected when crossing the bifurcation point to a non-zero fixed point. At this point the memory degrades drastically and it becomes insufficient for any task requiring a certain amount of knowledge from the previous inputs. This bifurcation point shifts to smaller values of η for larger values of θ.

When comparing the situation of $p = 1$ with the one of $p = 7$, we get the results depicted in Fig. 5.9. In Fig. 5.9(a) and (c) the case of $p = 1$ is shown, while Fig. 5.9(b) and (d) contain the memory capacity scan for $p = 7$.

For values of η larger than 1 the memory capacity is low ($MC < 10$) in both cases and for any value of the feedback strength the memory capacity is higher for lower values of input scaling. In the NARMA10 optimal parameter region the highest memory capacity values are found, but the ones for $p = 7$

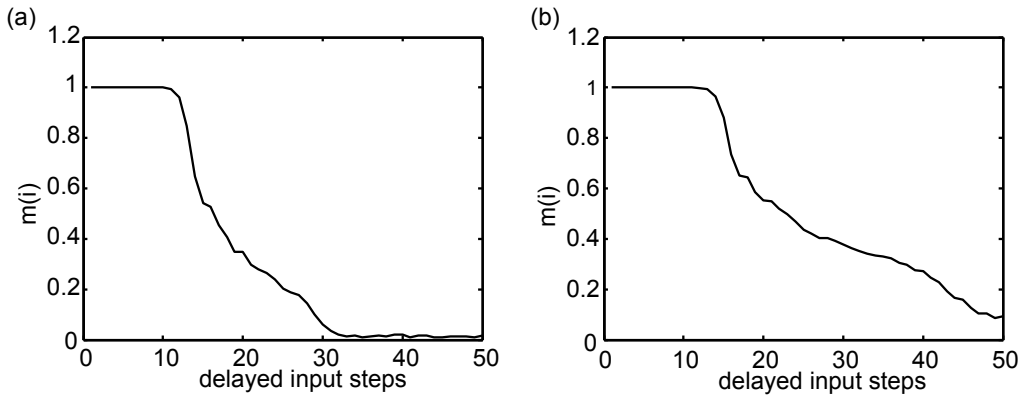


Fig. 5.10: Memory functions for delayed feedback reservoir with Mackey-Glass nonlinearity, different p. The memory function is shown for $p = 1$ and $p = 7$ in (a) and (b), respectively. In both figures the same parameter values are used: $\eta = 0.5, \gamma = 0.01$ and $\tau = 80$ consisting of 400 nodes separated by $\theta = 0.2$.

are significantly higher than the ones for $p = 1$. While the region of high memory capacity reaches values between 20 and 30 for $p = 1$, the system with $p = 7$ reaches values up to 75. It might strike as odd that the system with the higher exponent seems to have more memory than the system with the weaker nonlinear transfer function. However, the observed region consists of small values of γ scanning only a small part of the nonlinearity[2], hence the destructive power in terms of memory for the more nonlinear system cannot be seen yet. Around 0 the deviations from linear behavior are smaller for high exponents. The nonlinear contribution ϵ ($\epsilon < 1$) is taken to the power 7, hence it is smaller than for the exponent 1 case. Another aspect is the shape of the memory curve. For tasks such as NARMA10 is is not the total memory capacity that is important, it is the memory of a certain number of earlier steps. The individual memory curves are shown in Fig. 5.10.

In Fig. 5.10(a) the memory function of a Mackey-Glass nonlinearity with exponent 1 is depicted. When comparing the shape of this curve with the one of a Mackey-Glass nonlinearity with exponent 7, shown in Fig. 5.10(b), we remark that the memory is better for a higher exponent. Not only the region of very high correlation is more extended, but also the memory tail at the end is longer.

[2]Although $\gamma = 1$ might not seem a small value, the combination of the NARMA10 input and the mask already provide a small amplitude. For $\gamma = 1$ the maximum range that is explored is $0.5 \cdot (\pm 0.1) \cdot 1 = \pm 0.05$.

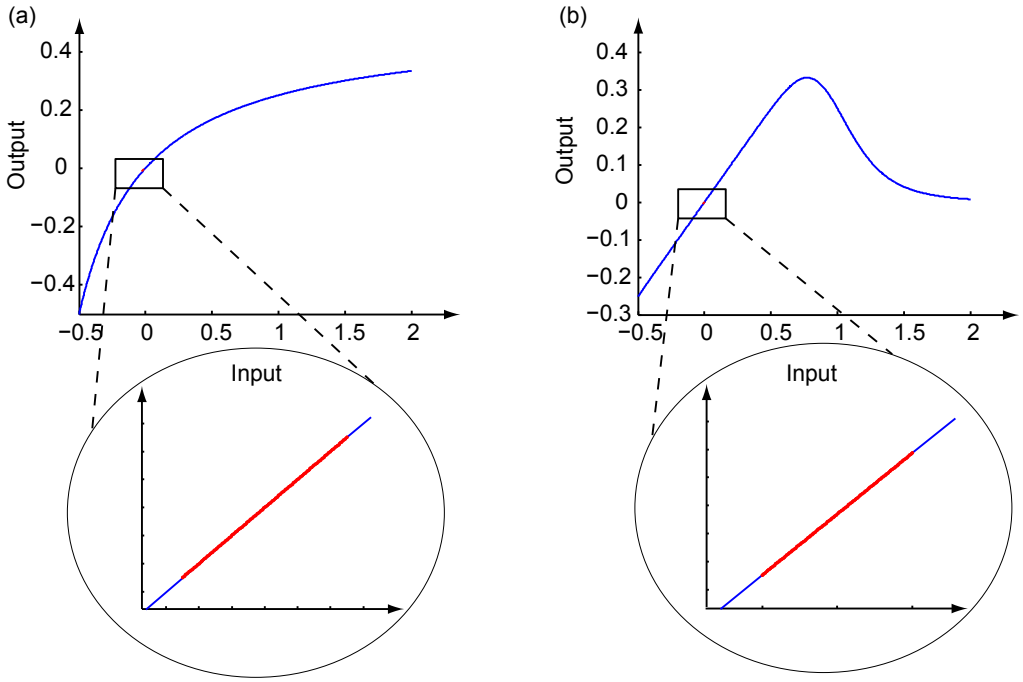


Fig. 5.11: Nonlinearity shapes for Mackey-Glass nonlinearity, different p. The Mackey-Glass nonlinearity transfer function is shown for $p = 1$ and $p = 7$ in (a) and (b) respectively. In both figures $\eta = 0.5$. Blue indicates the transfer function of the nonlinearity, while the red stars show the region that is actually scanned by the input. Although the global shape of the nonlinearity varies strongly with p, the local nonlinearity in the region where the input is scanned is quite similar for both.

Since for the investigated set of parameters ($\gamma = 0.01, \eta = 0.5$, NARMA10 input samples) only a small part of the nonlinearity is scanned by the input signal, the global shape of the nonlinearity becomes of less importance. In Fig. 5.11 the global nonlinearity shape is given as an illustration, but from the inset it can be seen that the regions scanned by the input are very similar. For small values of injected input the higher exponent even corresponds to a more linear function.

5.2.2 Memory quality

Another measure that is occasionally used in the RC community is the memory quality. The memory capacity represents the area under the memory curve depicted in Fig. 5.7, but it fails to capture all information about the

curve. It makes no distinction between the different shapes of the curves, as long as the total area underneath the curve remains the same. Hermans *et al.* therefore introduced another measure called the memory quality [112]. The memory quality is defined as

$$\mu_q = \frac{1}{\mu_c} \sum_{i=1}^{\mu_c} m_i, \tag{5.4}$$

with μ_c being the memory capacity as defined in Eq.(5.3).

In contrast to the memory capacity, this measure gives an indication of how rectangular the shape of the memory curve is. For some tasks the memory might be required to be very high at the first steps, as for example for the NARMA10 task. In Eq.(5.2) the term $u_k u_{k-9}$ implies that when calculating target y_{k+1} we still need to have the input of 10 steps earlier present in the memory to complete the computation. The other terms containing the combinations of previous targets add some extra steps to this requirements, but in total a memory of more or less 20 steps is all that is needed to achieve a good performance. A longer memory will not decrease the error any further. Any inputs further than 10 steps steps ago become almost irrelevant. In such a case a perfect memory is required up to 10 to 20 steps ago and then the memory can decrease rapidly.

5.3 Explaining performance: an example

The properties described in the previous sections provide us with tools to estimate the limits of performance on a certain task without explicitly testing the benchmark in question. Because of its simplicity we consider again the NARMA10 test using a Mackey-Glass nonlinearity. For convenience the performance curve shown in Fig. 3.5 is repeated here in Fig. 5.12.

The exponent of the Mackey-Glass nonlinearity is taken to be $p = 1$ and the delay line consists of 400 virtual nodes separated by a distance of $\theta = 0.2$. The best result is found for very low values of γ. By comparing with the results from the memory capacity (Fig. 5.8(a)) and the computational ability (Fig. 5.6(a)), we find that a compromise has to be found between both of them. Good computational ability is found for higher values of γ, while the memory capacity is maximal for the lowest values of γ. The valley of good performance on the NARMA10 task corresponds to the region where both, memory and computational ability, score well.

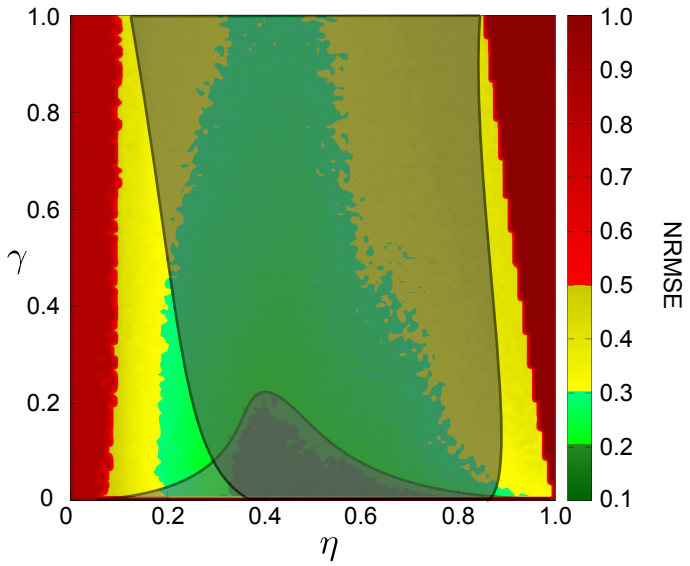


Fig. 5.12: Explaining performance on the NARMA10 task. The feedback strength (η) is scanned versus the input scaling (γ). The upper grey area represents the region of high computational ability as found in Fig. 5.6(a) and the lower grey shaded area denotes the region of good memory, coming from Fig. 5.8(a).

5.4 Noise

In a hardware implementation different types of noise can be present. Computational performance is mainly affected by noise within the system itself and by quantization noise due to the finite resolution of the analogue-to-digital (A/D) and digital-to-analogue (D/A) converters.

5.4.1 System noise

System noise can have a significant effect on the performance of delayed feedback systems as reservoir computers. The noisy reservoir state is fed back into the nonlinear node and this way keeps on propagating through the system. We model the noise by Gaussian white noise $\xi(t)$ with standard deviation σ_N. In the case the system is described by a single variable, the equation becomes

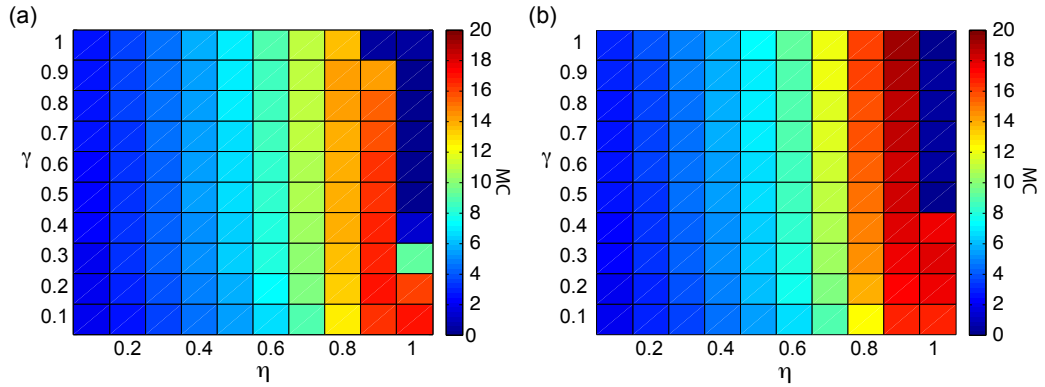


Fig. 5.13: Effect of noise on memory capacity for delayed feedback reservoir with Mackey-Glass nonlinearity.. Memory capacity is shown in color coding while scanning γ on the y-axis and η on the x-axis. Both for (a) $p = 1$ and (b) $p = 7$ the introduction of system noise leads to destruction of the memory of the system. The standard deviation of the used noise is $\sigma_N = 10^{-6}$.

$$x(t) = -x(t) + \eta F\left[x(t-\tau) + \gamma J(t)\right] + \xi(t). \tag{5.5}$$

The Signal to Noise Ratio (SNR) is given by

$$SNR = 20 \log\left(\frac{\sigma_{in}}{\sigma_N}\right), \tag{5.6}$$

where σ_{in} is the standard deviation of the masked input $J(t)$, already scaled with the input scaling factor γ. The effect of noise can be seen clearly in the memory capacity of the system, as is demonstrated in Fig. 5.13.

A two-dimensional parameter scan varying the feedback strength and the input scaling shows the effect of noise on the memory capacity. Compared to Fig. 5.8(a) the memory has decreased significantly. Especially the region of small input scaling factors is affected. The signal becomes smaller and hence more sensitive to noise. The noise strength is kept constant at $\sigma_N = 10^{-6}$. The average standard deviation of a masked, time-multiplexed NARMA10 input is 0.0285. For constant noise strength the SNR varies with γ. For $\gamma = 0.1$ this becomes

$$SNR = 20\log\left(\frac{\sigma_I}{\sigma_N}\right) = 20\log\left(\frac{\gamma \cdot 0.0285}{10^{-6}}\right) = 69\,\text{dB},$$

while for $\gamma = 1$ the SNR is 69 dB. When looking at the computational ability, we would see an even more drastic effect. The computational ability is defined as the difference between the kernel quality rank and the generalization rank, see Eq.5.1. However, when adding noise to the system, the generalization matrix becomes of full rank. All realizations become linearly independent because all the noise contributions, regardless of their amplitude, are linearly independent. In that case the tolerance used in the rank calculation becomes an important factor. Even in the noiseless system the calculation of the rank suffers from numerical noise in the calculation. To avoid having a full rank all the time, a tolerance is built in. In all the results shown this tolerance level was set just high enough to eliminate the effect of numerical noise. Using the computational ability in noisy systems would require a re-definition of the tolerance level based on the amplitude of the noise. This also implies that transferring this concept to experimental implementations is not trivial. The exact choice of the threshold level and the implications of this choice are still an open question and are not considered in this dissertation.

In Fig. 5.14 the performance on the NARMA10 task is shown when this same white noise is applied. The best performance found is an NRMSE around 0.4, the performance of a linear shift register. Hence noise of this strength has degraded both the memory and the computational ability sufficiently to destroy the reservoir's ability to solve the NARMA10 task, both for $p = 1$ and $p = 7$.

5.4.2 Quantization noise

Another form of noise that occurs in the experimental implementation is quantization noise. The input values, whether originating from a discrete or a continuous interval, are in practice injected into the experimental setup with a limited precision. Also the states read out of the delay line and used for training are subjected to quantization. Fig. 5.15 demonstrates the evolution of the obtained error as a function of the number of bits for the Isolated Spoken Digit Recognition task. The simulations are performed on the Mackey-Glass system of Eq.(3.3), with the parameters settings which were found to be optimal in Fig. 3.8: $p = 7, \eta = 0.8$ and $\gamma = 0.5$. Once again the performance is expressed in two ways, the WER (in black) and the margin (in red). As shown in Fig. 3.8 the optimal result is 0.14% WER,

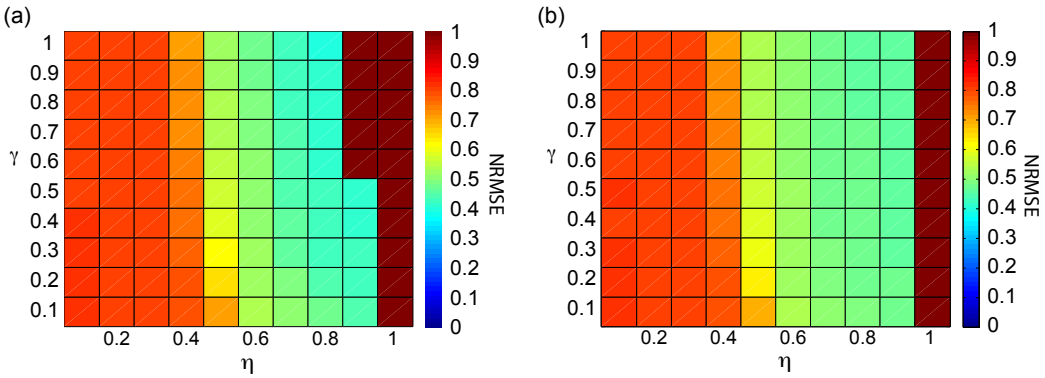


Fig. 5.14: Effect of noise on the NARMA10 task for delayed feedback reservoir with Mackey-Glass nonlinearity. The NRMSE is shown in color coding while scanning γ on the y-axis and η on the x-axis. Both for (a) $p = 1$ and (b) $p = 7$ the introduction of system noise leads to a significant increase in error. The standard deviation of the used noise is $\sigma_N = 10^{-6}$.

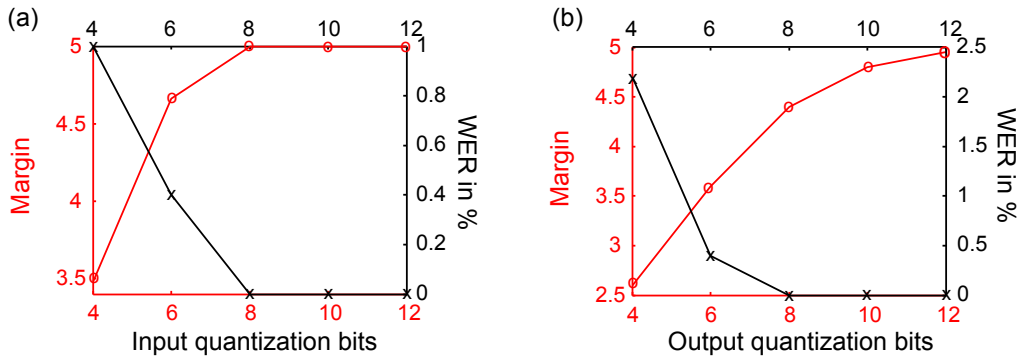


Fig. 5.15: Effect of quantization noise on the isolated spoken digit recognition task for delayed feedback reservoir with Mackey-Glass nonlinearity. The number of bits used in the quantization process, distributed over the entire dynamical range, is varied. For every quantization level the error is given as a margin (red) and as a WER (black).

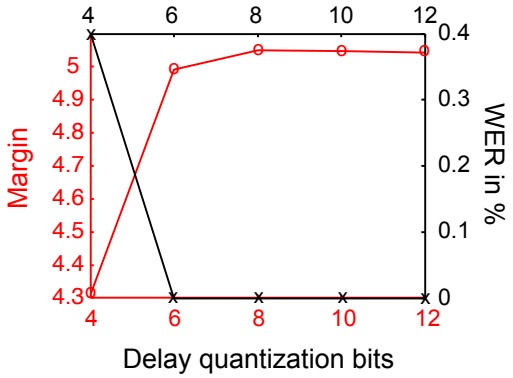


Fig. 5.16: Effect of quantization noise on delayed feedback reservoir with Mackey-Glass nonlinearity. We vary the number of bits used in the quantization process, distributed over the entire dynamical range. The red line denotes the margin, while the black line denotes the WER.

corresponding to a margin of around 5. The results here plotted in Fig. 5.15 show that as soon as the number of quantization bits is greater than or equal to 8, the performance remains unaffected. The error in a 20 fold cross-validation was found to be 0, indicating zero mistakes or one miss-classified word in this run. The fact that this task seems more robust to noise is because of the less strict requirements on precision in the reservoir states. In Fig. 5.16 the effect of the quantization of the nonlinearity itself is depicted. The quantized states are not only the ones used for training, but also they are fed back to the nonlinear node. Only for 4 bits a slight decrease is observed, while from 6 bits on the reservoir works optimally. The reason for this insensitivity for quantization is that in fact the nonlinear transfer function itself has changed. The nonlinear transfer function is no longer a smooth shape, but it contains different levels. In the previous case, where we quantized only the states for training, the nonlinearity makes a continuous mapping, but the state used for training is quantized. Small node output differences will then in the training procedure be enlarged by the quantization. When the nonlinear transfer function itself is quantized, the values trained on are consistent with the node values present in the system. This explains the good performance, even with a low number of bits.

5.5 Conclusion

We have discussed two important properties of the reservoir: computational ability and memory. Both of them are tools to evaluate the performance of a reservoir, independently from any benchmark task.

The computational ability can be calculated as the difference between the kernel quality and the generalization. We have applied this measure for a delayed feedback reservoir with a Mackey-Glass nonlinearity type, focussing on the parameter region where optimal performance for the NARMA10 task was found in Chapter 3. For a delayed feedback reservoir with a Mackey-Glass type nonlinearity with $p = 1$ the parameter region of high computational ability is significantly wider and more pronounced.

Next to computational ability we have also evaluated the memory. Although the memory capacity was found to be higher for a high nonlinearity exponent, the case of $p = 1$ has a relatively wide parameter region where the memory is sufficient for the NARMA10 task. Moreover, in this case we were able to find an overlap of the parameter region of high computational ability and the one with high memory capacity. It is in this overlap zone that good performance is found for the NARMA10 task. From these findings we can explain why good results are obtained for NARMA10 in the case of $p = 1$, and not in the case of $p = 7$.

Finally, we studied the effect of noise on the reservoir performance, making a distinction between system noise and quantization noise. The NARMA10 task is highly noise sensitive and the performance degrades drastically, even for small noise amplitudes. The memory capacity is severely affected and the use of the computational ability cannot be easily transferred to the case of a noisy system. When investigating the performance of quantization noise on the isolated spoken digit recognition, we find in general good noise robustness. The system is less sensitive to input quantization than to quantization of the reservoir states, probably because of the low-pass filtering effect of the nonlinear node. When the nonlinear transfer function itself is quantized the performance remains excellent, even for very few quantization levels. In general, we can roughly state that classification tasks are more noise robust than system identification tasks.

In the next chapter we will introduce some modifications to the single node delayed feedback architecture, with as a goal to extend the parameter region of good performance and to improve the noise robustness of the system.

6 System modifications

We have shown that single node delayed feedback systems have good computational properties. However, as was already demonstrated for the NARMA10 task, finding a compromise between long and good memory and strong computational power is far from straightforward. While the former can be obtained with weakly nonlinear systems, the latter usually requires strong nonlinearities that are able to transform and mix information. In order to satisfy both requirements simultaneously, one might be forced to work in parameter regimes that are hard to reach experimentally or that e.g. require high precision. Another option is to modify the architecture of the delayed feedback reservoir used so far. The basic configuration consists of one nonlinear node and a delay line that feeds back the signal after holding it for a time τ. Nothing restricts us from investigating adapted setups, where some of the capabilities of the system can be boosted in terms of performance or speed. Here, we suggest three improvements to the system: multiple delayed feedback, a network motif consisting of two bidirectionally delay coupled nodes and an optimal construction of the mask values in the pre-processing stage.

6.1 Multiple delayed feedback

6.1.1 Architecture

The configuration discussed so far always relied on the same input feeding procedure. Every input sample was held constant for a time τ equal to the delay time, representing one discrete input step. This way the input response was distributed over the entire delay time, ensuring the maximal use of the dimensionality introduced by the delay. Another implication is

that it becomes challenging to remember previous inputs. Every discrete step, a new input changes all the node states. The only memory mechanisms are the feedback line and the inertia of the nonlinear node and neither gives a direct link to states older than one step ago. These older states only remain present in the system when the nonlinearity is very weak.

Here, we outline another approach, based on the presence of multiple feedback lines in the system. Originally this was proposed by prof. L. Larger in the framework of the PHOCUS project and the work was initiated by dr. S. Ortin and prof. L. Pesquera. The basic setup is shown in Fig. 6.1. To preserve older information more explicitly, we can add several delay lines to the system. The longer the delay, the older the response that is being fed back. Even without explicitly reading the older states from the delay line, the information is re-injected into the system and extends its memory. In Fig. 6.1(a) the basic configuration is shown for the example of three delay lines. The delay of length τ corresponds to the situation as was considered so far. The extra delay lines correspond to delays of 2τ, 3τ etc.. The green nodes represent the responses to the last masked input sequence, the red ones to one discrete step before that etc. Similar to the procedure used in [17] the tapped states of the first interval are employed as virtual nodes and their value is used in the training procedure. In order to only take into account the effect of previous states being fed back and not including the fact that they are still present in the delay line, only the youngest set of nodes is used for the training. This set corresponds to the response to the present input. All the older states will remain present in the feedback line for a while, but they are not included in the training data.

In a practical implementation we can benefit from the fact that the longest delay line contains all the states that appear in the shorter one. From Fig. 6.1(a) it can be seen that the green nodes appear in all three delay lines hence we have a clear redundancy. In Fig. 6.1(b) a more resource efficient configuration is shown, where only the longest delay line is present. The shorter delay lines are introduced in the system by tapping the longer one at the desired interval and feeding back all the tapped states to the node.

6.1.2 Numerically obtained performance

6.1.2.1 Memory

In Fig. 6.2(a) the memory curve is depicted in the case of a Mackey-Glass nonlinearity with single feedback and in Fig. 6.2(b) we show the memory curve for the same nonlinearity, but using multiple feedback with 5 delay

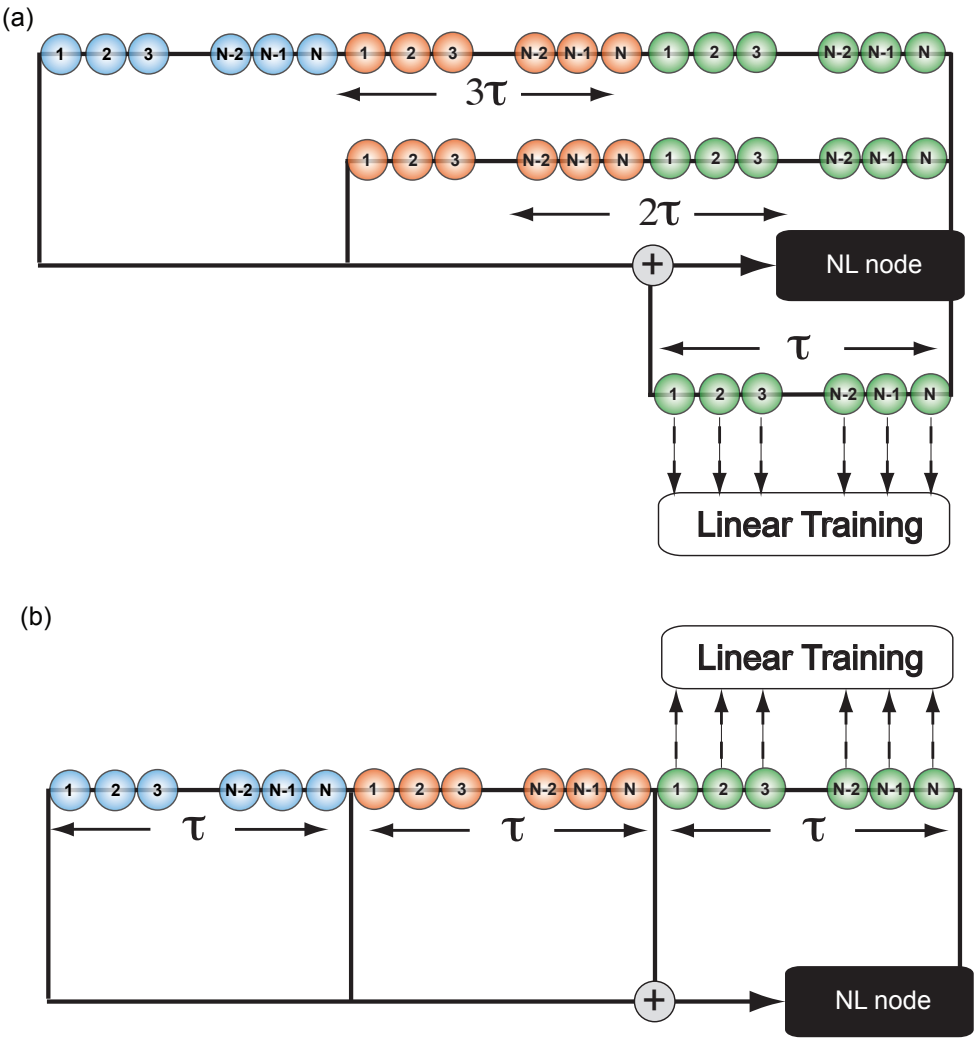


Fig. 6.1: Multiple feedback setup. In this example the delay time is chosen as 3τ, corresponding to the length of 3 time-multiplexed input samples. The training algorithm only operates on the youngest set of node states (depicted in green). (a) scheme of the multiple delay concept, (b) equivalent scheme with only 1 delay line and taps.

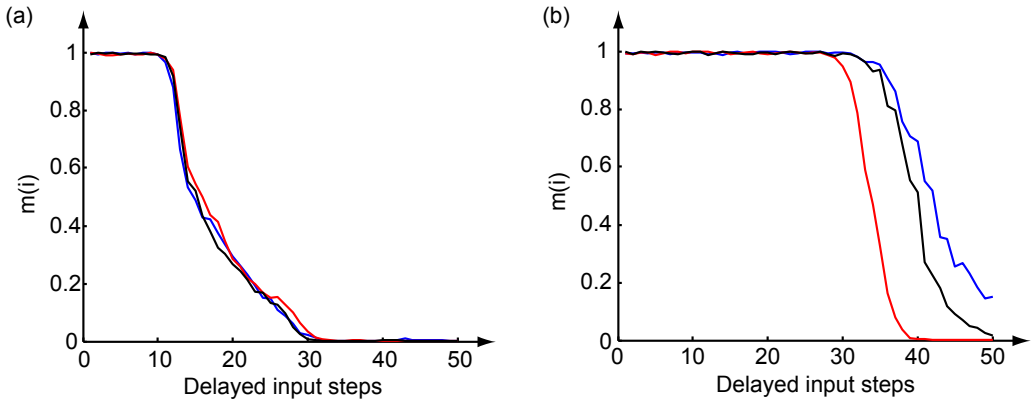


Fig. 6.2: Memory function for multiple delay lines. The memory function is depicted for several feedback weight distributions. The red line represents the oldest states having a smaller weight, blue the oldest states having a larger weight. In both cases the scaling is linear from old to young. The black line represents all weights to be equal. In all three cases the total sum of the weights is 1 and each delay line is chosen randomly within the corresponding τ-interval. Parameters are chosen as: $\gamma = 1, \eta = 0.5$ and $p = 1$. (a) 1 delay line (b) 10 delay lines.

lines. The different colors correspond to different feedback weight configurations. Blue corresponds to the configuration where the weights are linearly decreasing with the highest weight being assigned to the oldest response, red represents the opposite situation where the youngest states have the highest weight and black corresponds to all weights being equal. In all three cases the total sum of the feedback weights is kept constant and equal to 1. For the linearly increasing or decreasing weights the values are chosen as: $w_1 = \frac{1}{S}, w_2 = \frac{2}{S}, ..., w_n = \frac{n}{S}$, with S being the sum of the numerators of all weight values. It becomes clear that having feedback from older responses has a positive influence on the linear memory capacity. The more feedback lines are introduced (with the total delay length increasing correspondingly), the longer the memory.

Using explicit feedback of several previous input responses enables us to design the memory function [1]. While in traditional neural networks the memory function remains high for a certain amount of steps to fade out afterwards, here it becomes possible to have memory holes. In Fig. 6.3 we depict the situation where feedback lines omitted in the schedule. A total delay length of $100 \times \tau$ was used, but the responses belonging to the interval

[1] An idea proposed by dr. S. Ortin and prof. L. Pesquera in the framework of the PHOCUS project

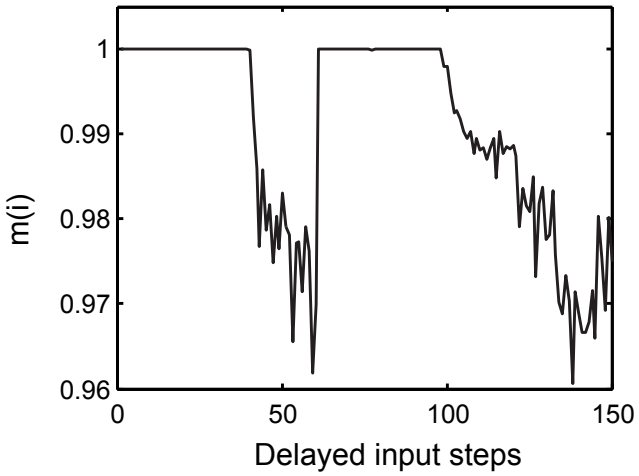


Fig. 6.3: Memory curve for the multiple delayed feedback case. The total delay is 100τ. Every τ-interval is being fed back to the nonlinear node from a random position within the interval, except for the part $[40 \times \tau, 60 \times \tau]$. All feedback weights are equal and their total sum is 1. A clear degradation in memory is observed for the inputs injected 40 to 60 steps ago. For steps 61 to 100 the memory returns to eventually degrade permanently for older inputs.

$[40 \times \tau, 60 \times \tau]$ were not being fed back to the nonlinear node. As can be seen in Fig. 6.3 the memory function is not monotonously degrading as was the case in all curves so far.

6.1.2.2 NARMA10

A crucial choice is the exact length of the feedback lines. They can be chosen to exactly match a multiple of τ, e.g. $l \cdot \tau$, with $l \in \mathbb{N}_0$, or to show a certain mismatch, $l \cdot \tau + \delta$, with $\delta < \tau$. In Fig. 6.4 we plot the NRMSE obtained for the NARMA10 benchmark when varying the number of delay lines used in the delayed feedback configuration. In all cases the parameters were chosen as: $\eta = 0.4, \gamma = 0.01, p = 1$ and $\theta = 0.2$, a configuration far from the optimal point for the single delayed feedback case. In that case the high input scaling degrades the memory sufficient to make the obtained error rise significantly in the single delayed feedback situation. The number of delay lines used for feedback is varied in the and for every value the obtained NRMSE for the NARMA10 task is given. In Fig. 6.4(a) the feedback positions in the delay line are chosen exactly at a multiple of τ. The first point of the curve corresponds to the single delayed feedback situation of Chapter 3. There the

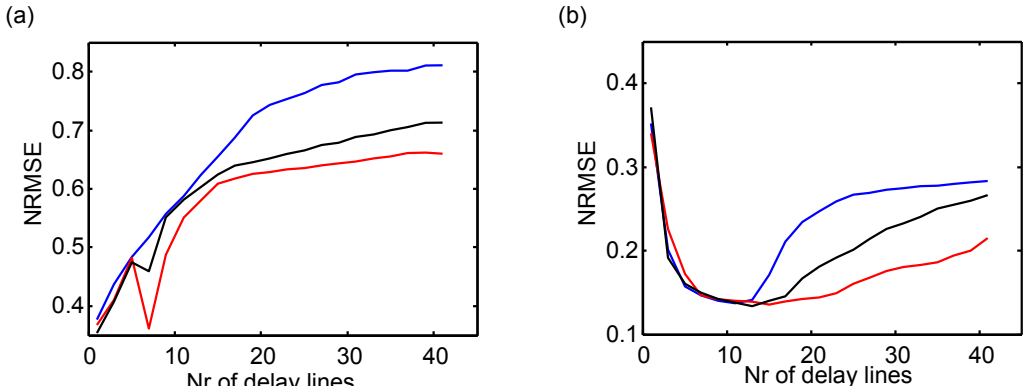


Fig. 6.4: NARMA10 performance curve for multiple delay lines. The NRMSE obtained for the NARMA10 task is depicted as a function of the number of feedback lines for several feedback weight distributions. (a) The length of the delay lines was chosen exactly at multiples of τ, the shortest delay line. (b) The length of the delay lines was chosen not to be a multiple of τ. The red line represents the oldest states having a smaller weight, blue the oldest states having a larger weight. In both cases the scaling is linear from old to young. The black line represent all weights to be equal. In all three cases the total sum of the weights is 1. For 1 delay line the three cases are identical. We used a Mackey-Glass type nonlinearity with parameters: $\gamma = 1, \eta = 0.4$ and $p = 1$.

performance is slightly below 0.4, due the high input scaling that was used ($\gamma = 1$). When adding extra delay lines, the performance only gets worse. In Fig. 6.4(b), where the lengths of the extra delay lines are not exact multiples of τ a clearly positive effect on the NRMSE is observed. In that case all fed back values correspond to different inputs, but corresponding virtual nodes, implying that all of them have states constructed with the same mask value. When the delay lines are chosen randomly within different τ intervals, making sure that none of them feeds back responses belonging to the same input step as another feedback line, the performance can be boosted significantly. NRMSEs as low as 0.12 can be achieved for input scaling factors one hundred times larger than the ones used in the single feedback case. This way will benefit from the computational ability of the system at higher input scaling, while compensating for the memory requirements using multiple feedback lines. Until 12 delay lines we hardly observe a difference between the different feedback weight distributions. Only for a much higher number the configuration with the highest weights corresponding to the longest delay lines seems to degrade more rapidly in terms of performance than the inverse case. The equally distributed weights keep the middle between the two. This observation can be explained by the fact that for the NARMA10 task enough memory is present in all three cases, as can be deducted from Fig. 6.2(b). For the case of higher weights being assigned to longer delays, more importance is given to inputs very far in the past, further than necessary for the approximation of the NARMA10 target. They become an increasingly disturbing part of the total feedback signal and weigh more and more as the number of delay lines increases. As a side remark we mention that in order to maintain the basic structure of the delay feedback system, both the shortest and the longest delay line had there feedback position at the exact τ interval. Like that that we did not alter the way the 400 virtual nodes used for training are constructed and we had control over the total delay of the system.

In Fig. 6.5 the input scaling and the feedback strength are varied and the performance on the NARMA10 task in the presence of noise is depicted. The standard deviation of the noise is set at 10^{-6}, as was done in Chapter 5, section 5.4.1. We employ 10 feedback lines with equal weights for all of them and their sum being equal to 1. The length of the delay lines were not matched exactly with an interval of τ. We observe that excellent performance is found even in the presence of noise. The region of very good performance reaches above $\gamma = 10$. When higher input scalings are used the response of the node is stronger and the variations in the output are larger. This could facilitate an experimental implementation significantly.

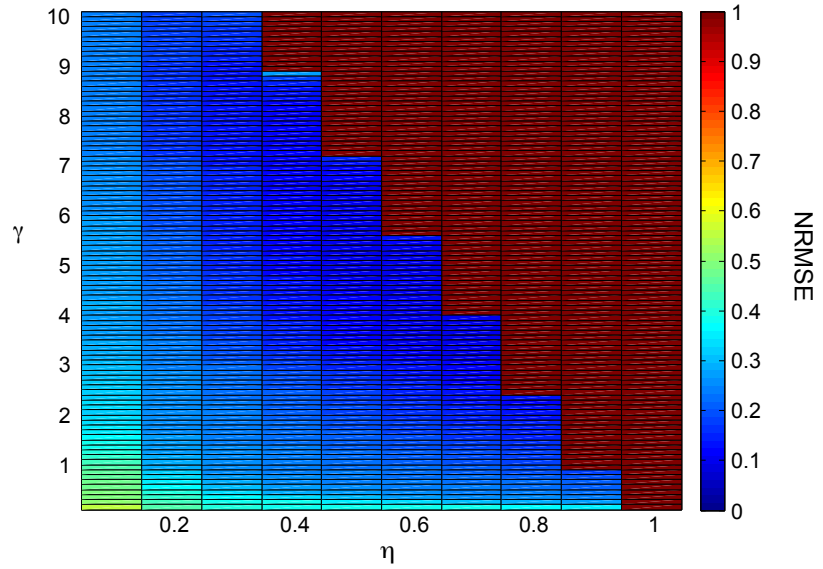


Fig. 6.5: NARMA10 performance curve for multiple delay lines in the presence of noise. The NRMSE obtained for the NARMA10 task is depicted as a function of the feedback strength and the input scaling. The delay lines have lengths not equal to multiples of τ. The blue line represents the oldest states having a smaller weight, red the oldest states having a larger weight. In both cases the scaling is linear from old to young. The black line represent all weights to be equal. In all three cases the total sum of the weights is 1. We used a Mackey-Glass type nonlinearity with parameters: $\tau = 80, \theta = 0.2$ and $p = 1$.

6.2 Network motifs

6.2.1 Architecture

In the case of a conventional reservoir computing setup, where the reservoir consists of a random network of nonlinear nodes, the input is fed in parallel to all nodes of the reservoir. While in terms of simplicity this concept suffers from the large number of nodes that need to be implemented, in terms of speed it benefits from its huge parallelism. On the other hand, the delayed feedback approach is far more resource efficient, but that implies that all the input data need to be serialized. The processing speed is limited by the delay length and can only be improved by using a faster sampling rate. It is reasonable to look for a compromise between these two extremes by using a few nonlinear elements instead of only one. This opens a wide range of possible configurations of which only one example is studied here. The dynamics of small networks of delay-coupled nonlinear nodes have already been discussed in several studies [64, 113, 114].

In Fig. 6.6 the situation of two bidirectionally coupled nodes is shown. They are separated from each other by half the delay time and are both driven by a masked input. To fully benefit from all available delay induced dimensions the mask is chosen to be different for both nodes. In the case of both nodes being identical and receiving exactly the same masked input, their output would be identical as well. Hence the virtual node states in the two delay lines would be redundant. When choosing a different mask we benefit from all dimensions in a total delay time of τ. Moreover, the time to process one input sample has now become $\tau/2$, only half of the single node delay system. The more nodes are added, the faster the processing can take place, because less serializing is needed. In the limit we reach a network of many nodes, similar to conventional neural networks. A trade-off has to be made between speed, simplicity of implementation and other factors such as power consumption.

6.2.2 Numerically obtained performance for NARMA10

In Fig. 6.7 the memory capacity of a bidirectionally coupled system of two Mackey-Glass nodes is depicted. Both nodes receives exactly the same input, but with a different mask. The parameters are identical for both nodes. The exponent is taken 1 in Fig. 6.7a and 7 in Fig. 6.7b.

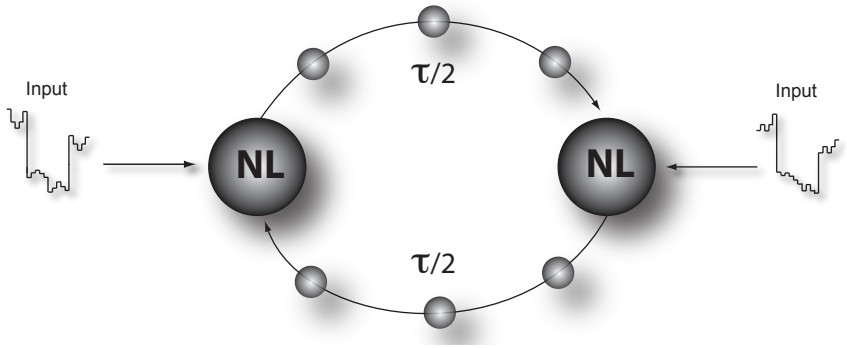


Fig. 6.6: Network of two bidirectionally coupled nodes. The two nodes are equally spaced along the delay line and receive the same input, but with a different masking function for each.

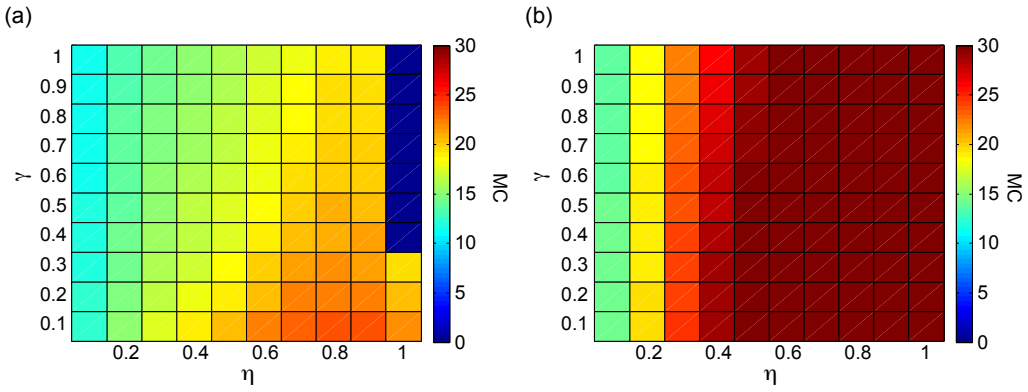


Fig. 6.7: Memory capacity for two coupled nodes. The memory capacity is plotted in color code for a network motif consisting of 2 bidirectionally coupled nodes, while scanning the feedback strength and the input scaling. Both nodes receive identical inputs, but preprocessed with a different mask. The two nodes are equally spaced within the τ-interval and each drive 200 virtual nodes with a separation of 0.2. (a) $p = 1$, (b) $p = 7$.

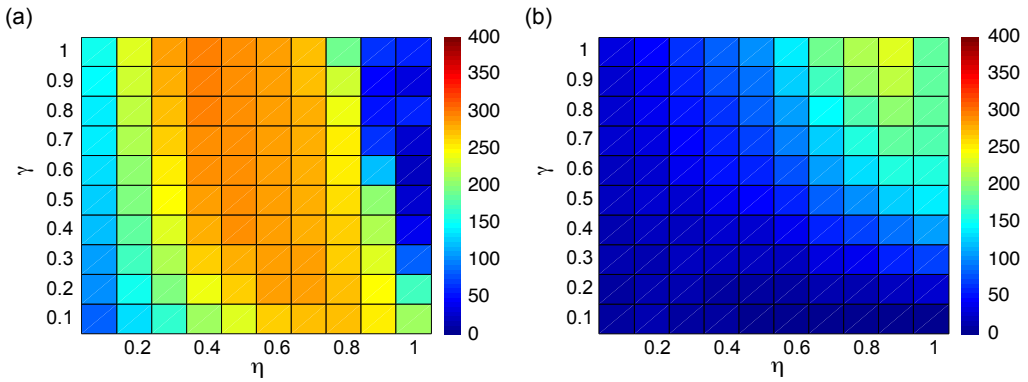


Fig. 6.8: Computational ability for two coupled nodes. The computational ability is plotted in color code for a network motif consisting of 2 bidirectionally coupled nodes, while scanning the feedback strength and the input scaling. Both nodes receive identical inputs, but preprocessed with a different mask. The two nodes are equally spaced within the τ-interval and each drive 200 virtual nodes with a separation of 0.2. (a) $p = 1$, (b) $p = 7$.

Next to memory capacity, we study the computational ability of the bidirectionally coupled system. The color coded plots in Fig. 6.8 shows the rank corresponding to the computational ability as defined in Eq.(5.1).

The scanned parameter region is identical to the one studied in detail for the single delayed-feedback situation. Also in terms of results a clear similarity can be seen. Although the parameter scan uses larger steps in this case, the same trends are observed. For a low exponent the memory is highest for small values of γ and a total degradation of memory is observed when crossing the bifurcation point. For the $p = 7$ the memory is higher in general (for $0 < \gamma < 1$) and the drastic degradation within the scanned interval disappears. When looking at the performance of the system on the NARMA10 benchmark (see Fig. 6.9), we observe for $p = 1$ (Fig. 6.9a) that the region of good performance has shifted towards higher values of γ. NRMSE values of 0.15 can be found for γ as high as 1. This relaxes the strict conditions that were imposed for good performance with the single delay element. For an exponent of $p = 7$ the system never outperforms a linear shift register and all errors are higher than 0.4, as shown in Fig. 6.9b.

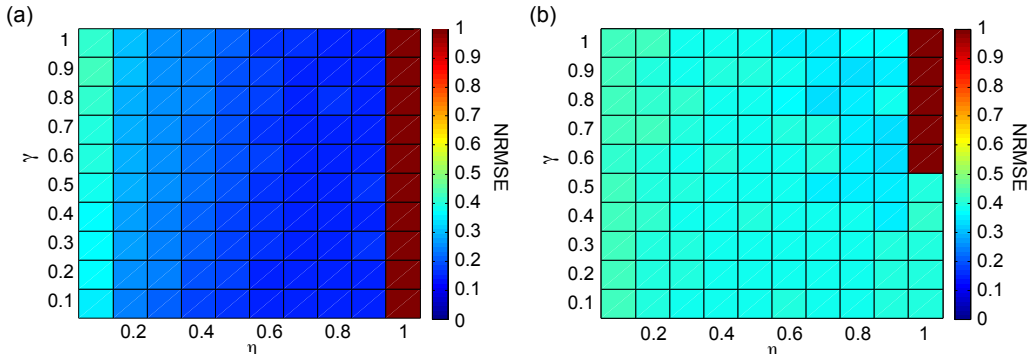


Fig. 6.9: Two coupled nodes, NARMA10. The NARMA10 performance expressed as an NRMSE is plotted in color code for a network motif consisting of 2 bidirectionally coupled nodes, while scanning the feedback strength and the input scaling. Both nodes receive identical inputs, but preprocessed with a different mask. The two nodes are equally spaced within the τ-interval and each node drives 200 virtual nodes with a separation of 0.2. (a) $p = 1$, the region of good performance - NRMSE ≈ 0.15 - reaches much higher values of γ compared to the single delayed-feedback case (b) $p = 7$, the performance never reaches better than what can be achieved by a linear shift register.

6.3 Construction of an optimal mask

6.3.1 Concept

In the results described throughout this dissertation the values of the mask were always chosen randomly. They represent the values with which an input is multiplied before it is fed into a node, as explained in Chapter 2, section 2.1.2. The masks used for the NARMA10 task and the Santa Fe time series prediction were always random drawings from the binary distribution $\{-0.1, 0.1\}$. When a large amount of virtual nodes is considered this is acceptable, since the probability of a high variety in occurring mask value sequences increases with the number of nodes. However, in terms of efficiency, it is desirable to construct a mask in the shortest possible way (the smallest possible number of nodes) that still yields good performance. When using a random mask, the possibility of having N equal mask values for N nodes is also included. In fact this corresponds to the absence of a mask and it results in a bad performance. In order to avoid this or any similar situation, we outline a method that guarantees an optimal choice for the mask values. Only binary masks are considered, an approach which is justified since for

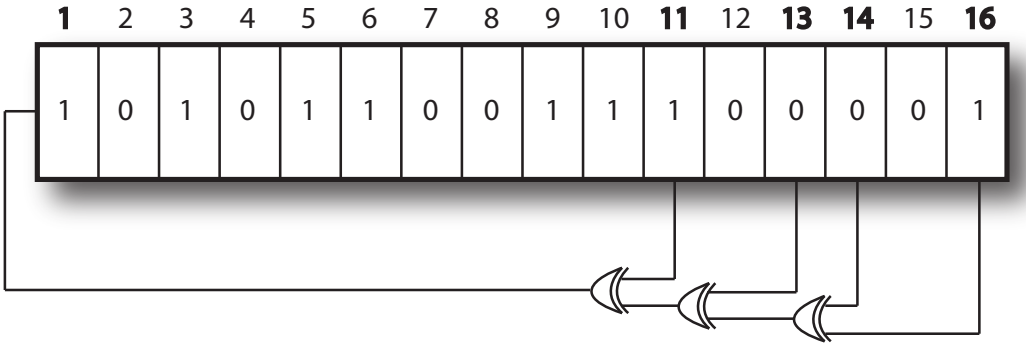


Fig. 6.10: Fibonacci linear feedback shift register for a 16-bit block. With the scheme depicted here a sequence of bits, containing all possible bit patterns of a 16-bit block, can be generated. When letting the system run freely the output is a series of bits with a period of 65535 bits. Bits 11, 13, 14 and 16 are tapped and XORed to generate the next bits of the register.

several tasks no improvement in performance was observed when multiple mask values were used to solve the task. We conjecture that the series of mask values should contain all possible patterns of these values, given a certain sequence length. Therefore, one needs to decide how long the sequence of mask values to be considered should be. Because of inertia, the output of the nonlinear node does not only depend on the input and the feedback values, but also on the most recent previous output values. If for example the node state is strongly dependent on the states that were obtained for the 6 previous adjacent virtual nodes, we believe no more information can be extracted then given by all possible length-6 sequences of the two mask values. Even more, in order to construct the most efficient mask, it seems that all of them should occur exactly once.

This can be done by using a modification of what is called *maximum length sequences* [115]. In maximum length sequences a series of values is generated that contains all possible bit patterns of an m-bit block in a ring structure. This means that all possible sequences of an m-bit block occur on condition that the bits are placed in a ring and that the bits at the end can make a combination with the bits at the beginning. The linear feedback shift register setup that generates such a maximum length sequence is depicted in Fig. 6.10. The example in Fig. 6.10 shows the linear feedback shift register for 16-bit blocks. The structure generates a sequence of bits containing all possible realizations of a 16-bit block when the bits are placed in a closed ring. The bit stream is periodical with period 65535 ($= 2^m - 1$). Some of the states in the 16-bit block are tapped and subsequently combined using

an XOR logical operation. The outcome of the XOR gates serves as the youngest bit to enter the shift register. The positions of the taps depend on the length of the block to be varied. Usually they are described by what is referred to as a polynomial mod 2, which implies that the coefficients of the polynomial are either 0 or 1. Tab. 6.1 gives an overview of the polynomials for blocks going from length 2 to 19.

Since in the case of the optimally constructed mask all bit patterns need to be present for one input step of length τ, the ring structure is not valid within one and the same input step and some bits need to be added. In general, when all possible realizations of m bits are required in the mask, the minimal mask length is exactly 2^m+m-1. This extra length of m bits originates from adding one 0 (the combination with m zeros is not present in a maximum length sequence) and from adding the last $m-1$ bits of the sequence to the beginning of the series (because the mask is not a ring structure for one input step).

6.3.2 Numerically obtained performance

6.3.2.1 NARMA10

By replacing 0 and 1 with the low and high value of the mask, respectively, we can construct a mask with all possible mask value sequences of m values present in the mask sequence. Using these optimally constructed masks, a performance plot is shown for the Mackey-Glass nonlinearity in Fig. 6.11, where the number of virtual nodes is scanned and the performance on the NARMA10 task is shown as an NRMSE. The parameters are chosen close to optimal according to Fig. 3.5, $\eta = 0.5, \gamma = 0.01, p = 1$ and $\theta = 0.2$, while the two mask values were ± 0.1.

The points in green represent the scoring of 100 optimally constructed masks, while the blue points mark the scoring of 100 randomly chosen binary masks. Theoretically the random masks could also include the sequence with all identical mask values for which the performance is known to be 0.40. Hence, the spread on the random masks is much larger than shown here. Firstly, it can be noted that for the optimal NRMSE a saturation can be observed when increasing the number of virtual nodes up to 134 or more. This complies with the fact that the θ is chosen to be 0.2, corresponding to 5 virtual nodes per time unit of the nonlinear node. The response time of the node corresponds to 5 virtual nodes, hence it is acceptable to assume that combinations with a total length of a little more than the response time are important in terms of performance. Any older state has no significant influence on the present node state and cannot create more variation in the reservoir states. The

Bits	**Characteristic polynomial**	**Length**
m		$2^m - 1$
2	$x^2 + x + 1$	3
3	$x^3 + x^2 + 1$	7
4	$x^4 + x^3 + 1$	15
5	$x^5 + x^3 + 1$	31
6	$x^6 + x^5 + 1$	63
7	$x^7 + x^6 + 1$	127
8	$x^8 + x^6 + x^5 + x^4 + 1$	255
9	$x^9 + x^5 + 1$	511
10	$x^{10} + x^7 + 1$	1023
11	$x^{11} + x^9 + 1$	2047
12	$x^{12} + x^{11} + x^{10} + x^4 + 1$	4095
13	$x^{13} + x^{12} + x^{11} + x^8 + 1$	8191
14	$x^{14} + x^{13} + x^{12} + x^2 + 1$	16383
15	$x^{15} + x^{14} + 1$	32767
16	$x^{16} + x^{14} + x^{13} + x^{11} + 1$	65535
17	$x^{17} + x^{14} + 1$	131071
18	$x^{18} + x^{11} + 1$	262143
19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$	524287

Table 6.1: Characteristic polynomials for the construction of a linear feedback shift register. In the column on the left the length of the bit block for which all possible patterns need to be constructed is given, with in the center column the polynomial indicating where the taps need to be placed in the setup of 6.10. The column on the right shows the length of the sequence that contains all possible variations of the m-bit block.

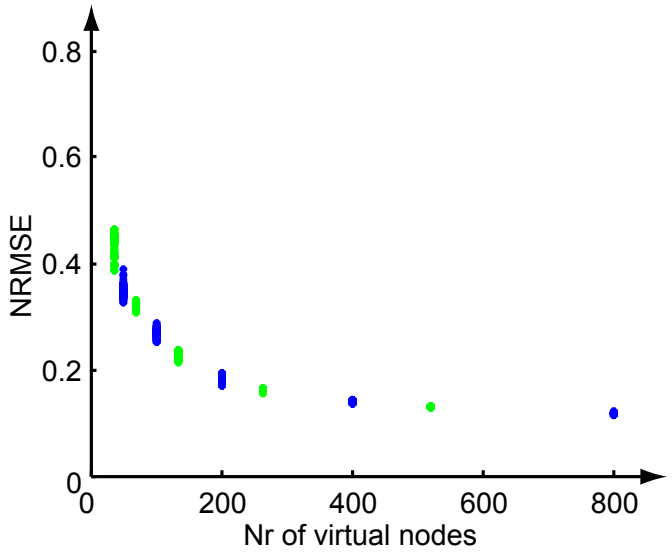


Fig. 6.11: Performance plot NARMA10 for random and optimally constructed masks. A Mackey-Glass nonlinearity type is used, with parameter settings: $\eta = 0.5, \gamma = 0.01, p = 1$ and $\theta = 0.2$. The blue points denote the scoring of the random masks, while the green points indicate the NRMSE obtained for optimally constructed masks. For every scanned node number 100 masks were generated.

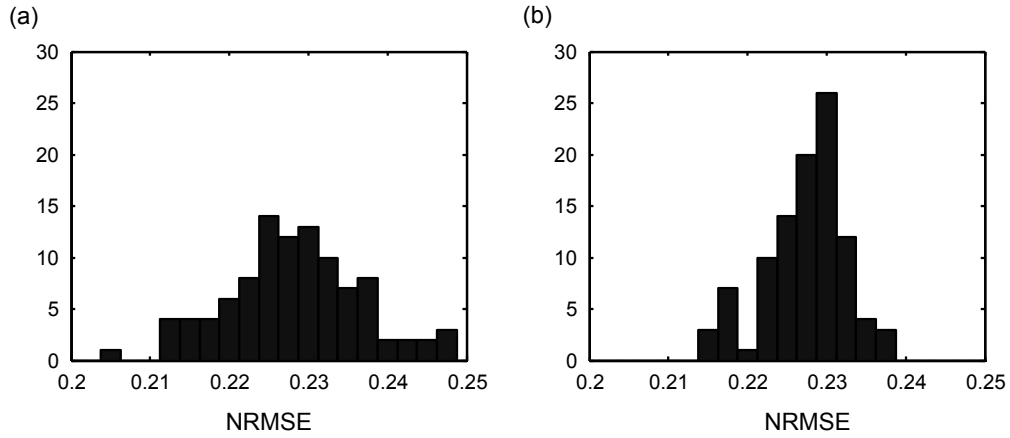


Fig. 6.12: Histogram of the performance for NARMA10 for random and optimally constructed masks. For both plots the employed nonlinearity is of the Mackey-Glass type, 134 virtual nodes were employed and the mask values are +0.1 and -0.1. (a) distribution for 100 randomly chosen masks, (b) distribution for optimally constructed masks.

spread on the performance of the different masks is higher for small node numbers and decreases when more virtual nodes are employed. For the small node numbers, not all possible bit combinations of 5 or more bits are present. For some of the randomly chosen masks there are more combinations lacking than for others, resulting in a spread on the error. When increasing the nodes to a higher number, all possible mask value patterns are included and even a certain redundancy is introduced, which can be necessary when the used training algorithm is not able to extract all information optimally from the reservoir states.

In Fig. 6.13 the performance of the optimally constructed masks is shown for two values of θ, the virtual node separation. The green points denote the score of the system for $\theta = 0.2$ and the blue points for $\theta = 1$. The spread on the values obtained for $\theta = 1$ is significantly lower. The node state is less dependent on the states of adjacent virtual nodes, hence variability in the mask is of less importance. The overall error is higher for $\theta = 1$.

6.3.2.2 Santa Fe laser data

When the same test is made for the Santa Fe Laser task, a similar performance-node relation is found, as shown in Fig. 6.14. Again, the number of virtual nodes is varied, but this time the performance is shown as an NMSE. The

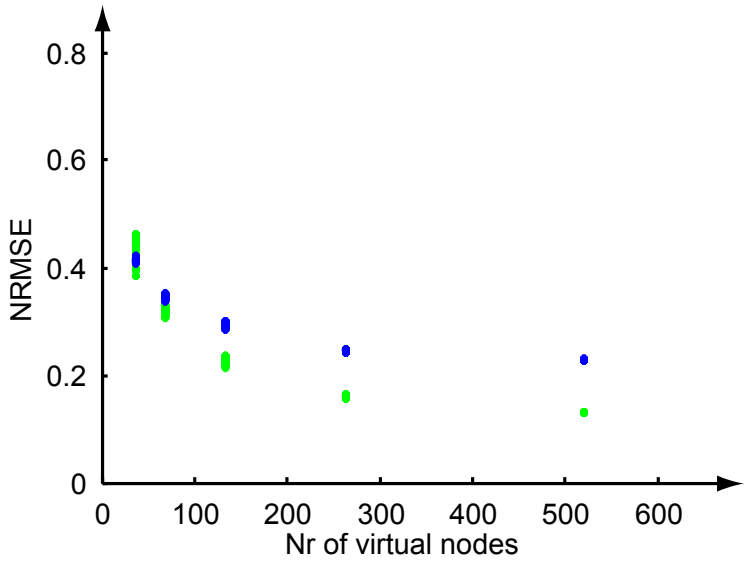


Fig. 6.13: Performance plot for NARMA10 for optimally constructed masks with different θ. A Mackey-Glass nonlinearity type is used, with parameter settings: $\eta = 0.5, \gamma = 0.01$ and $p = 1$. The blue points denote the scoring of the optimally constructed masks for $\theta = 0.2$, while the green points indicate the NRMSE obtained for the same optimally constructed masks for $\theta = 1$. For every scanned node number 100 masks were generated.

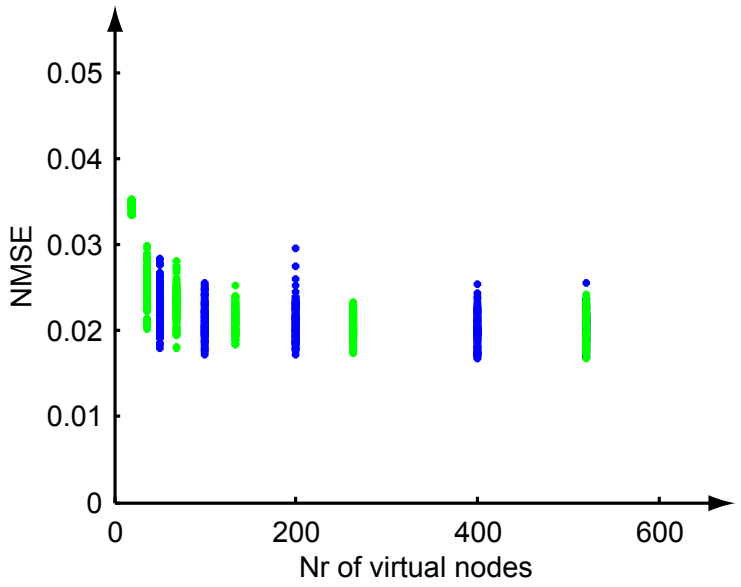


Fig. 6.14: Performance plot for the Santa Fe laser prediction task for random and optimally constructed masks. A Mackey-Glass nonlinearity type is used, with parameter settings: $\eta = 0.5, \gamma = 0.01, p = 1$ and $\theta = 0.2$. The blue points denote the scoring of the random masks, while the green points indicate the NMSE obtained for optimally constructed masks. For every scanned node number 100 masks were generated.

blue points denote the error of the randomly chosen masks and the green points represent the optimally constructed masks. Here, saturation of the performance already occurs at 36 nodes.

6.4 Conclusion

Even though the single node delayed feedback reservoir can compete with traditional reservoirs made of hundreds of connected nodes in terms of performance, still improvements can be made to the design of the system. In this chapter we have discussed three possible modifications: two modifications in the basic architecture and one in the pre-processing stage.

By adding several feedback lines to the single delayed feedback situation, the memory can be boosted significantly. This can be beneficial for tasks that require a very long memory, but it also allows for tasks with low or medium

memory requirements, to work in parameter regimes that were not suitable before. We have achieved excellent performance for the NARMA10 task for high values of the input scaling ($\gamma \approx 10$), even in the case with system noise. This could possibly facilitate experimental implementations of noise-sensitive tasks.

Another modification is the use of more than one nonlinear node. Some improvements are found in terms of performance, but the main advantage is the fact that the speed of the system can be increased. Although in this dissertation we have limited ourselves to one implementation, two bidirectionally delay-coupled nodes, there are many possible configurations that can be explored here.

Finally, we have proposed a modification in the pre-processing procedure. Instead of using a randomly chosen mask, we have outlined a procedure to construct a mask vector of which we believe it optimizes performances. Using a modified version of maximum length sequences, it becomes possible to create more diversity in the states of the virtual nodes. These masks exhibit the same average performance as a randomly chosen mask, but the spread becomes lower and we guarantee that the accidental choice of a 'bad' mask is avoided. Both for randomly chosen and for optimally constructed masks, when increasing the number of virtual nodes, for the NARMA10 task the performance saturates and the spread on the performance becomes smaller. For the Santa Fe laser data prediction this effect is not so pronounced.

7 Conclusion and future outlook

7.1 What we accomplished...

Reservoir computing is a recently introduced paradigm in machine learning, performing information processing in a computationally efficient way. By dividing a recurrent neural network into two layers, called the reservoir and the output layer, the training algorithm can be reduced to a mere linear algorithm. This approach yields excellent results for a variety of tasks, e.g., time series prediction, pattern recognition and robot control.

We have identified delayed feedback systems comprising a single nonlinear node and a delay line as suitable systems for reservoir computing. Delayed feedback systems are receiving an increasing amount of attention, with delay occurring in many physical and technological systems. Examples are the brain, traffic dynamics, networks of lasers, etc. While often it is considered to be a nuisance, some technological implementations such as chaos control and chaos communication exploit the presence of delay beneficially.

Along the spatial dimension of the delay line we have defined virtual nodes that fulfill a role comparable to the nodes in a traditional network approach. Taking into account the different architecture, the procedure of feeding the input to the reservoir has been drastically redesigned. The input signal needs to be sequentialized and a mask is imprinted on it before it is injected into the node. After processing the information in the reservoir and reading out the node states, they are fed to a linear training algorithm. The used training procedures are identical to the ones employed in traditional reservoir computing setups.

We have pinpointed the virtual node separation distance as one of the most important parameters in the delayed feedback reservoir setup. This distance

is defined via the mask function and enables us to keep the output of the nonlinear node constantly in a transient regime. Because of the inertia of the nonlinear node it determines the interconnectivity structure of the virtual nodes. We have succeeded in describing this structure, equivalent to what is used in reservoirs consisting of a vast network of nonlinear nodes. From a fundamental point of view, the simplicity of the delayed feedback architecture assists in gaining a deeper understanding of the interplay of dynamical properties and reservoir performance. The reduction of a complex network to a single hardware node facilitates implementations enormously, because only a few components are needed. Nevertheless, the use of delay dynamical systems imposes certain constraints, since the feeding of the virtual nodes is carried out serially, in contrast to the parallel feeding of the nodes in traditional reservoir computing. This serial feeding procedure implies that the speed of the information processing is limited by the delay time. This drawback is compensated for by the much simpler hardware architecture of the reservoir, and by the fact that the read-out can be taken at a single point of the delay line.

The performance of our approach has been evaluated numerically and has led to two experimental implementations. The first one is an electronic implementation with a Mackey-Glass nonlinear node type. This represents the first experimental implementation of delayed feedback reservoirs and it is able to compete with numerical simulations in terms of performance. Several benchmark tasks have been used to quantify the performance. For the NARMA10 task, a problem of system identification, an NRMSE of 0.12 was reached in numerical simulations and an NMSE of 0.019 was found for the Santa Fe laser data prediction task. Experimentally, the isolated spoken digit recognition task was solved, resulting in a WER of 0.14%. All these results are comparable to what is found for state-of-the-art traditional reservoirs.

The second implementation is opto-electronic and uses an Ikeda nonlinearity type. In this case excellent performance was also achieved for the same three tasks. Numerically, for the NARMA10 task an NRMSE of 0.22 was found, for the Santa Fe laser data prediction an NMSE of 0.04 and for the isolated spoken digit recognition a WER of less than 0.2% has been obtained. We have investigated the Santa Fe laser data prediction and the isolated spoken digit recognition experimentally as well. The performance on the Santa Fe task decreased drastically with increasing quantization noise, but for the isolated spoken digit the experimentally found WER remains below 0.2%. This proves that the exact realization of the system or the specific shape of the nonlinear transfer function are not crucial. Since, in practice, often a band-pass filtering effect is observed in the opto-electronic experimental setup, we have modeled this numerically. The band-pass effect can be beneficial to widen the available parameter region. From the two cases we can

conclude that a simple nonlinear dynamical system subject to delayed feedback can efficiently perform information processing. As a consequence, our simple scheme can replace the complex networks used in traditional reservoir computing.

Besides looking at the performance obtained on standard benchmark tasks, we have numerically investigated some task-independent properties of the reservoir. By measuring the kernel quality and the generalization we can characterize system performance using the computation ability. This enables us to predict in which parameter regions the delayed feedback reservoir can potentially process information in a correct way. When combining this with another measure, the memory capacity, performance results on, e.g., the NARMA10 benchmark can be explained. We have briefly studied the effect of system noise and quantization noise. It has been found that for the NARMA10 task noise can have a devastating effect on performance, even for small noise amplitudes. The isolated spoken digit recognition task seems more noise robust and very good results are obtained, even with quantization levels down to only 6 digits.

To allow expansion of the viable parameter region and as such to facilitate experimental implementation, we have proposed some system modifications. We have investigated a single node delayed feedback reservoir with multiple delayed feedback lines, a network motif consisting of two bi-directionally delay coupled nonlinear nodes and we outlined a method to construct an optimal mask. The modifications have been evaluated in terms of speed, performance and noise robustness. With multiple delayed feedback the memory of the reservoir can be boosted significantly, allowing for operation at very high input scalings. In this region the system is less noise sensitive. For the network motifs the parameter region of good operation widens, but not drastically. The main advantage here is that a trade-off can be made between temporal mapping using delay and spatial mapping using nonlinear nodes. When using a few physically present nonlinear nodes, the speed of information processing increases. Finally, we have suggested a method to construct the mask using a method based on maximum length sequences, instead of randomly choosing the mask values. This results in a smaller spread on the performance because of the employed mask and, more importantly, it avoids drastic failure due to a badly chosen mask realization.

The concept we have proposed could enable high-speed implementations of reservoir computing, using high-speed components that would be too demanding or expensive to be used for many nodes. In particular, realizations based on electronics and photonics systems are feasible using this simple scheme, including real-time processing capabilities.

7.2 Continuation of this work

7.2.1 Integrated all-optical approach

We have gained experience in driving and training delayed feedback systems by implementing this in an electronic and an opto-electronic way. For both of these approaches the input signals need to be fed in electronically. However, in a telecommunication application the signals are optical and it would be beneficial to skip the conversion to electronics. An all-optical implementation is highly desired and this is the main goal of the PHOCUS project. At the IFISC institute in the Universitat de les Illes Balears substantial progress has been made on both the numerical and the experimental implementation of a semiconductor laser with feedback, used as a reservoir in the framework of the PHOCUS project. One of the challenges in an all-optical approach is the influence of the optical phase. Although the phase can provide an enrichment of the complexity of the system, it is a difficult property to control experimentally. Also the masking procedure has to be optimized for optical systems, taking into account the several timescales present in the optical system.

As candidates for the implementation of the all-optical system, we too will consider these SLs with delayed optical feedback. SLs are fast, sensitive to feedback and can be integrated (Fig. 7.1). Interesting devices are distributed feedback (DFB) lasers, semiconductor ring lasers (SRLs) and wavelength tunable lasers. While DFBs are easily controllable and mature devices, SRLs and wavelength tunable lasers are particularly suited for integration. The delay can be constructed externally or in an integrated way. In most of the laser types described above several optical modes can be present simultaneously, with interactions between them. This offers the opportunity to drastically reduce the length of the delay line, as several input streams can be sent to different wavelengths, while several computational nodes belonging to different wavelengths can be placed at the same position in the delay line. Appropriate methods to input the data streams and to read out the data will need to be developed and tested. We will focus on problems relevant to information and telecommunication industry. This includes the implementation of hashing, flow recognition, transient data classification, sensor networks and channel equalization. The data streams in a telecom network are being sent around optically, hence an all-optical processing unit to analyze them on the fly is highly desirable. These devices will all be subject to an optical feedback signal. The delayed feedback can be implemented in

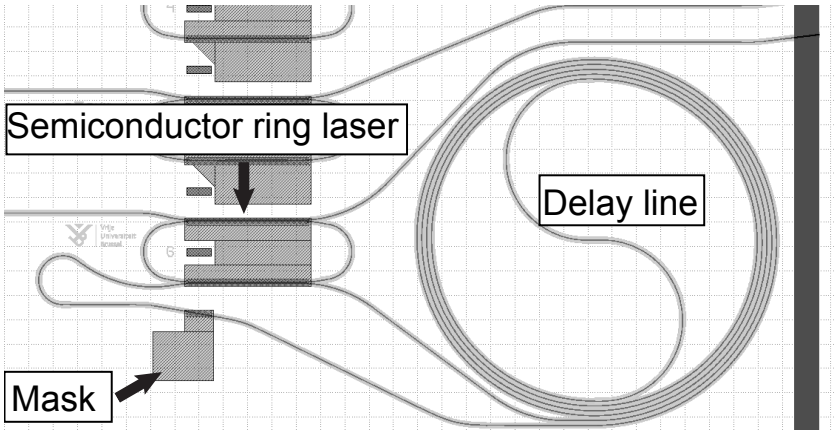


Fig. 7.1: Proposal of a reservoir computing setup using a semiconductor ring laser. The ring laser and the delay line can be realized in an integrated form. A separate contact is provided to imprint the mask on the input signal. Chip designed by Werner Coomans.

several ways: in integrated form or externally, e.g. through a pigtailed optical fiber. Ring laser devices have the practical advantage of having several input and output ports. This advantage can be exploited to feed the input to the system in one port and to re-inject the feedback in another port. Ring lasers have two counter-propagating modes, hence the feedback can be self-feedback (in the same directional mode) or cross-feedback (in the counter propagating mode). The effect of their interaction on the performance is one of the concepts to be studied. We can tune the feedback strength by placing a variable optical attenuator in the delay path. Input data is fed into the system either by injection of optical signals or by direct modulation of the electrical injection current. To excite sufficiently complex transient behavior in semiconductor lasers, the injected data rates need to be around 10 GHz. On the detection side, we need to analyze the complexity of the excited transient behavior by sampling the optical output signals at speeds exceeding this frequency. When working with optical injection, the opportunity to address several wavelengths arises. By sending in different streams of information at different wavelengths, parallelism is created without adding extra components. As a consequence, the constraint on the delay time can be decreased by the number of wavelength channels used.

We hope to have contributed to an emergent and exciting new field, combining physics, photonics and machine learning. We dare to hope that the work presented here will foster novel developments and photonic implementations of reservoir computing.

References

[1] Chouard, T. Turing at 100: Legacy of a universal mind. *Nature* **482**, 455 (2012).

[2] Rosenblatt, F. The perceptron–a perceiving and recognizing automaton. *Report 85-460-1, Cornell Aeronautical Laboratory* (1957).

[3] Minsky, M. L. & Papert, S. A. *Perceptrons* (MA: MIT Press, Cambridge, 1969).

[4] Kolmogorov, A. On the representation of continuous functions by several variables of superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk. SSSR* **953-956**, 669–681 (1957).

[5] Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).

[6] Jaeger, H. The 'echo state' approach to analyzing and training recurrent neural networks. *Technical Report GMD Report, German National Research Center for Information Technology* **148**, 2188–2191 (2001).

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

[8] Steil, J. J. Backpropagation-decorrelation: Online recurrent learning with o(n) complexity. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, vol. 1, 843–848 (2004).

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

[10] Verstraeten, D., Schrauwen, B., D′Haene, M. & Stroobandt, D. An experimental unification of reservoir computing methods. *Neural Netw.* **20**, 391–403 (2007).

[11] Buonomano, D. & Maass, W. State-dependent computations: spatiotemporal processing in cortical network. *Nat. Rev. Neurosci.* **10**, 113–125 (2009).

[12] Maass, W., Joshi, P. & Sontag, E. Computational aspects of feedback in neural circuits. *PLOS Comput. Biol.* **3**, 1–20 (2007).

[13] Verstraeten, D., Schrauwen, B., Stroobandt, D. & Van Campenhout, J. Isolated word recognition with the liquid state machine: a case study. *Inform. Process. Lett.* **95**, 521–528 (2005).

[14] Verstraeten, D., Schrauwen, B. & Stroobandt, D. Reservoir-based techniques for speech recognition. In *Proceedings of IJCNN06 , International Joint Conference on Neural Networks*, 1050–1053 (2006).

[15] Rabinovich, M., Huerta, R. & Laurent, G. Transient dynamics for neural processing. *Nat. Rev. Neurosci.* **321**, 48–50 (2008).

[16] Cover, T. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electron.* **14**, 326–334 (1965).

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

[18] Legenstein, R. & Maass, W. Edge of chaos and prediction of computational performance for neural microcircuit models. *Neural Networks* **16**, 323–333 (2007).

[19] Lukoševicius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review* **3**, 127–149 (2009).

[20] Hammer, B., Schrauwen, B. & Steil, J. Recent advances in efficient learning of recurrent networks. In *European Symposium on Artificial Neural Networks*, 213–226 (2009).

[21] Verstraeten, D. *Reservoir Computing: computation with dynamical systems.* Ph.D. thesis, Universiteit Gent (2010).

[22] Joshi, P. & Maass, W. Movement generation and control with generic neural microcircuits. In *Proceedings of the First International Workshop on Biologically Inspired Approaches to Advanced Information Technology (BioADIT)*, 258–273 (Lausanne, Switzerland, 2004).

[23] Burgsteiner, H. Training networks of biological realistic spiking neurons for real-time robot control. In *Proceedings of the 9th International Conference on Engineering Applications of Neural Networks*, 129–136 (Lille, France, 2005).

[24] Maass, W., Legenstein, R. & Markram, H. A new approach towards vision suggested by biologically realistic neural microcircuit models. In *Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision (BMCV)*, vol. 2525, 282–293 (Tubingen, Germany, 2002).

[25] Hertzberg, J., Jaeger, H. & Schonherr, F. Learning to ground fact symbols in behavior-based robots. In *Proceedings of the 15th European Conference on Artificial Intelligence*, 708–712 (Lyon, France, 2002).

[26] Jaeger, H. Reservoir riddles: suggestion for echo state network research (extended abstract). In *Proceedings of the IEEE international Joint Conference on Neural Networks (IJCNN)*, 1460–1462 (Montreal, Canada, 2005).

[27] Antonelo, E., Schrauwen, B. & Stroobandt, D. Event detection and localization for small mobile robots using reservoir computing. *Neural networks* .

[28] Antonelo, E., Schrauwen, B. & Stroobandt, D. Modeling multiple autonomous robot nehaviors and behavior switching with a single reservoir computing network. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 1843–1848 (Singapore, 2008).

[29] Antonelo, E., Schrauwen, B. & Stroobandt, D. Unsupervised learning in reservoir computing: modeling hippocampal place cells for small mobile robots. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)* (2009).

[30] VISA. *Counterfeit fraud* (2006).

[31] LeCun, Y. *et al.* Handwritten digit recognition: Applications of neural net chips and automatic learning. *IEEE Communication* 41–46 (1989).

[32] Simard, P., Steinkraus, D. & Platt, J. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, vol. 343, 958–963 (2003).

[33] Tong, M., A.D., B., Christiansen, E. & Cottrell, G. Learning grammatical structure with echo state networks. *Neural Networks* **20(3)**, 424 – 432 (2007).

[34] Skowronski, M. & Harris, J. Automatic speech recognitionusing a predictive echo state network classifier. *Neural Networks* **20(3)**, 414 – 423 (2007).

[35] Ghani, A., McGinnity, T., Maguire, L. & Harkin, J. Neuro-inspired speech recogition with recurrent spiking neurons. In *Proceedings of the International Conference on Artificial Neural Networks, Part I (ICANN)*, 513–522 (2008).

[36] Jaeger, H. Discovering multiscale dynamical features with hierarchical echo state networks. *Technical Report 10 GMD Report, German National Research Center for Information Technology* (2007).

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

[38] Crone, S., Nikolopoulis, K. & Hibon, M. Automatic modelling and forecasting with artificial neural networks - a forecasting competition evaluation. *technical Report, lancaster University management School* (2008).

[39] Wyffels, F., Schrauwen, B. & Stroobandt, D. Using reservoir computing in a decomposition approach for time series prediction. In *Proceedings of the European Symposium on Time Series Prediction*, 149–158 (2008).

[40] Greenhill, R. & Elias, H. http://www.shadowrobot.com/media/pictures.shtml [gfdl (www.gnu.org/copyleft/fdl.html) or cc-by-sa-3.0(www.creativecommons.org/licenses/by-sa/3.0/)]. *Wikimedia Commons* (2012).

[41] Duran Ortiz Mariordo, M. [gfdl (www.gnu.org/copyleft/fdl.html), cc-by-3.0 (www.creativecommons.org/licenses/by/3.0), gfdl (www.gnu.org/copyleft/fdl.html) or cc-by-3.0 (www.creativecommons.org/licenses/by/3.0)]. *Wikimedia Commons* (2012).

[42] Equationaudio. [cc-by-sa-3.0 (www.creativecommons.org/licenses/by-sa/3.0) or gfdl (www.gnu.org/copyleft/fdl.html)]. *Wikimedia Commons* (2012).

[43] Naylor, M. Image gallery, re-upload = ulamm 09:28, 16 november 2007 (utc) (transferred from en.wikipedia) [public domain]. *Wikimedia Commons* (2012).

[44] Vapnik, V. *The nature of statistical learning theory. Statistics for engineering and information science* (Springer, 1999), 2 edn.

[45] Aizerman, M., Braverman, E. & Rozonoer, L. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control* **25(6)**, 821–837 (1964).

[46] Hopfield, J. Neural networks and physical systems with emergent collective computational abilities. *Proc. NatL Acad. Sci. USA* **79**, 2554–2558 (1982).

[47] Hopfield, J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. NatL Acad. Sci. USA* **81**, 3088–3092 (1984).

[48] Fernando, C. & Sojakka, S. Pattern recognition in a bucket. *Lecture Notes in Computer Science* **2801/2003**, 588–597 (2003).

[49] Nikolic, D., Haeusler, S., Singer, W. & Maass, W. Temporal dynamics of information content carried by neurons in the primary visual cortex. In *Proceedings of NIPS, Advances in Neural Information Processing Systems*, vol. 19, 1041–1048 (2007).

[50] Schürmann, F., K., Meier & Schemmel, J. Edge of chaos computation in mixed-mode vlsi - a hard liquid. vol. 17, 1041–1048 (2004).

[51] Van Campenhout, J. Toward photonic reservoir computing: a tutorial. *IAP Doctoral School, Couvin* (2006).

[52] Vandoorne, K. *et al.* Toward optical signal processing using photonic reservoir computing. *Opt. Express* **16**, 11182–11192 (2008).

[53] Vandoorne, K., Dambre, J., Verstraeten, D., Schrauwen, B. & Bienstman, P. Parallel reservoir computing using optical amplifiers. *IEEE Trans. Neural Netw.* **22**, 1469–1481 (2011).

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

[55] Orosz, G., Wilson, R., Szalai, R. & Stépan, G. Exciting traffic jams: Nonlinear phenomena behind traffic jam formation on highways. *Phys. Rev. E* **80**, 046205 (2009).

[56] Pyragas, K. Continuous control of cahos by self-controlling feedback. *Phys. Lett. A* 170–421 (1992).

[57] Schöll, E. & Schuster, H. *Handbook of Chaos Control* (Wiley-VCH, Weinheim, 2008), 2 edn.

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

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

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

[61] Roelfsema, P. R., Engel, A., König, P. & Singer, W. Visuomotor integration is associated with zero time-lag synchronization among cortical areas. *Nature* **385**, 157–161 (1997).

[62] Fischer, I. *et al.* Zero-lag long-range synchronization via dynamical relaying. *Phys. Rev. Lett.* **97**, 123902 (2006).

[63] Vicente, R., Gollo, L., Mirasso, C., Fischer, I. & Pipa, G. Dynamical relaying can yield zero time lag neuronal synchrony despite long conduction delays. *P. Natl. Acad. Sci. USA* **105**, 17157–17162 (2008).

[64] Heil, T., Fischer, I., W., E., Mulet, J. & Mirasso, C. Chaos synchronization and spontaneous symmetry-breaking in symmetrically delay-coupled semiconductor lasers. *Phys. Rev. Lett.* **86**, 5 (2001).

[65] Takamatsu, A., Fujii, T. & Endo, I. Time delay effect in a living coupled oscillator system with the plasmodium of physarum polycephalum. *Phys. Rev. Lett.* **85**, 2026–2029 (2000).

[66] Van der Sande, G., Soriano, M. C. & Fischer, I. Dynamics, correlation scaling, and synchronization behavior in rings of delay-coupled oscillators. *Phys. Rev. E* **77** (2008).

[67] Ikeda, K. . & Matsumoto, K. . High-dimensional chaotic behavior in systems with time-delayed feedback. *Physica D* **29**, 223–235 (1987).

[68] Argyris, A. *et al.* Chaos-based communications at high bit rates using commercial fibre-optic links. *Nature* **438**, 343–346 (2005).

[69] Larger, L. *et al.* Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Opt. Express* **20**, 3241–3249 (2012).

[70] Boccaletti, S., Kurths, J., Osipov, G., Valladares, D. & Zhou, C. The synchronization of chaotic systems. *Phys. Rep.* **366**, 1 (2002).

[71] Pecora, L. M. & Carroll, T. Synchronization in chaotic systems. *Phys. Rev. Lett.* **64**, 821 (1990).

[72] Pikovsky, A., Rosenblum, M. & Kurths, J. *Synchronization: a universal concept in nonlinear sciences* (Cambridge University Press, Cambridge, UK, 2001).

[73] Le Berre, M. *et al.* Conjecture on the dimensions of chaotic attractors of delayed-feedback dynamical systems. *Phys. Rev. A* **35**, 4020–4022 (1987).

[74] Weigend, A. & Gershenfeld, N. *Time Series Prediction: Forecasting the Future and Understanding the Past*, vol. 80 (Addison-Wesley, 1993). URL `ftp://ftp.santafe.edu/pub/Time-Series/Competition`.

[75] Giacomelli, G., Meucci, R., Politi, A. & Arecchi, F. Defects and spacelike properties of delayed dynamical systems. *Phys. Rev. Lett.* .

[76] Paquot, Y. *et al.* Optoelectronic reservoircomputing. *Scientific Reports* **2** (2012).

[77] Paquot, Y. *et al.* Artificial intelligence at light speed : toward optoelectronic reservoir computing. *Belgian Pysical Society Magazine* 15–22 (2010).

[78] Paquot, Y., Dambre, J., Schrauwen, B., Haelterman, M. & Massar, S. Reservoir computing: a photonic neural network for information processing. In *Proceedings of SPIE Photonics Europe, Nonlinear Optics and Applications IV*, vol. 7728 (2010).

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

[80] Penrose, R. A generalized inverse for matrices. *Proc. Cambridge Phil. Soc.* **51**, 406–413 (1955).

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

[82] Duda, R. O., Hart, P. E. & Stork, D. G. *Pattern Classification* (John Wiley and Sons, 2001).

[83] URL `http://reslab.elis.ugent.be/software`.

[84] Atiya, A. & Parlos, A. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE T. Neural Netw.* **11**, 697–709 (2000).

[85] Jaeger, H. Adaptive nonlinear system identification with echo state networks. *Advance in Neural Information Processing Systems, MIT Press* **15**, 593–600 (2003).

[86] *Texas Instruments-Developed 46-Word Speaker-Dependent Isolated Word Corpus (TI46)* (NIST Speech Disc 7-1.1 (1 disc), September 1991).

[87] Doddington, G. & Schalk, T. Speech recognition: turning theory to practice. *IEEE Spectrum* **18**, 26–32 (1981).

[88] Lyon, R. A computational model of filtering, detection, and compression in the cochlea. In *Proceedings of IEEE-ICASSP*, vol. 7, 1282–1285 (1982).

[89] Huebner, U., Abraham, N. & Weiss, C. Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared nh3 laser. *Phys. Rev. A* .

[90] (National Geophysical Data Center (NGDC, 2007).

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

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

[93] Namajunas, A., Pyragas, K. & Tamasevicius, A. An electronic analog of the mackey-glass system. *Phys. Lett. A* **201**, 42–46 (1995).

[94] Sano, S., Uchida, A., Yoshimori, S. & Roy, R. Dual synchronization of chaos in mackey-glass electronic circuits with time-delayed feedback. *Phys. Rev. E* **75**, 016207 (2007).

[95] Steil, J. J. Memory in backpropagation-decorrelation o(n) efficient online recurrent learning. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)* (2005).

[96] Walker, W. Sphinx-4: A flexible open source framework for speech recognition. *Technical Report, Sun Microsystems* (2004).

[97] Caulfield, H. & Dolev, S. Why future supercomputing requires optics. *Nat. Photonics* **4**, 261 (2010).

[98] Miller, D. A. B. Correspondence to the editor. *Nat. Photonics* **4**, 406 (2010).

[99] Tucker, R. The role of optics in computing. *Nat. Photonics* **4**, 405 (2010).

[100] Neyer, A. & Voges, E. Dynamics of electrooptic bistable devices with delayed feedback. *IEEE J. Quantum Electron.* **18**, 2009–2015 (1982).

[101] Larger, L., Goedgebuer, J. & Udaltsov, V. S. Ikeda-based nonlinear delayed dynamics for application to secure optical transmission systems using chaos. *C. R. Phys.* **5**, 669–681 (2004).

[102] Callan, K., Illing, L., Gao, Z., Gauthier, D. J. & Schöll, E. Broadband chaos generated by an optoelectronic oscillator. *Phys. Rev. Lett.* **104**, 113901 (2010).

[103] Ikeda, K. Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system. *Optics Commun.* **30**, 257 (1979).

[104] Larger, L. & Dudley, J. M. Optoelectronic chaos. *Nature* **465**, 41–42 (2010).

[105] Erneux, T., Larger, L., Lee, M. W. & Goedgebuer, J. Ikeda hopf bifurcation revisited. *Physica D* **194**, 49–64 (2004).

[106] Udaltsov, V. *et al.* Bandpass chaotic dynamics of electronic oscillator operating with delayed nonlinear feedback. *IEEE T. Circuits-I* **49**, 1006–1009 (2002).

[107] Peil, M., Jacquot, M., Chembo, Y., Larger, L. & Erneux, T. Routes to chaos and multiple time scale dynamics in broadband bandpass nonlinear delay electro-optic oscillators. *Phys. Rev. E* **79** (2009).

[108] Ortin, S. & Pesquera, L. Deliverable 4: numerical evaluation of input/output methods. *PHOCUS Deliverable* (2010).

[109] Schrauwen, B., Buesing, L. & Legenstein, R. On computational power and the order-chaos phase transition in reservoir computing. In *Proceedings of NIPS, Advances in Neural Information Processing Systems*, vol. 21, 1425–1432 (Vancouver, 2009).

[110] Vapnik, V. *Statistical Learning Theory* (John Wiley, New York, 1999).

[111] Jaeger, H. Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the 'echo state network' approach. *Technical Report GMD Report 159, German National Research Center for Information Technology* (2002).

[112] Hermans, M. & Schrauwen, B. Memory in linear recurrent neural networks in continuous time. *Neural Networks* **23**, 341–355 (2010).

[113] D'Huys, O., Vicente, R., Erneux, T., Danckaert, J. & Fischer, I. Synchronization properties of network motifs: Influence of coupling delay and symmetry. *Chaos* **18** (2008).

[114] D'Huys, O., Fischer, I., Danckaert, J. & Vicente, R. Role of delay for the symmetry in the dynamics of networks. *Phys. Rev. E* **83** (2011).

[115] Davies, W. Generation and properties of maximum-length sequences. *Control* (1966).