System Analysis and Design of Physical Delay-Feedback Reservoir Computing Systems

Alexander Charles McDonnell

Doctor of Philosophy

University of York

School of Physics, Engineering and Technology

September 2024

Abstract

The Reservoir Computing Framework was first developed to aid with training complex recurrent neural networks that exhibit rich dynamical behaviours, and later expanded into a general model for dynamical systems. This allows a complex dynamical system to be treated as a "black box" by exploiting its input/output behaviour, which can be trained by a simple training algorithm to apply output weights, making them very computationally efficient. An interesting subset of reservoir computing is the delay-feedback reservoir, which has a hardware efficient architecture by trading space for complexity. A delay-feedback reservoir uses a single non-linear node, a delay-line, and a time-multiplexed input signal to produce a network of "virtual nodes", emulating a much larger spatial neural network. While delay-feedback reservoirs have been shown to compute temporal tasks exceptionally well, their design is unclear. Typically, a substrate is found that exhibits interesting dynamical properties, and the delay-feedback reservoir framework is applied, leading to an inefficient use of the substrate. This approach is common, as there is a lack of a general design framework; this is due to the effects of the internal parameters within a delay-feedback reservoir not being well understood. This thesis aims to better understand the function and parameters of a non-linear node within a delay-feedback reservoir so that the reservoir characteristics can be tuned to provide optimal performance for a particular computational task within a particular physical substrate. This thesis examines existing physical implementations of delay-feedback reservoirs to determine the key components within the non-linear node, which is shown to be a non-linear function and an integrator. The effect that the parameters within the key components of the non-linear node have on the performance and system dynamics are extensively investigated, showing that the performance can be significantly improved by changing the timescale of the integrator and by changing the exponent value within the Mackey-Glass non-linear function. An alternative non-linear node is proposed, where the non-linear and integration functions are combined within a high-order filter. An evolutionary algorithm was used to find the optimal filter parameters, which were then evaluated in terms of performance and system metrics. While it was found that a high-order filter can not replace a non-linear node, they can increase the memory of a system and can be used together with a non-linear function to supplement their dynamics.

Table of Contents

List of Tables

List of Figures

Dedication

First, I would like to thank my supervisor Martin for his continued feedback, support, and encouragement through this project. His passion and dedication to academia is inspiring.

I would also like to thank my parents, Joanne and Mark. I cannot thank them enough for the love and support they have given me over the years. I would not be where I am today without their guidance and understanding. Thank you for believing in me when I couldn't believe in myself.

I would also like to thank my siblings, Hannah and William, for their love, understanding, and comfort throughout my academic life.

Thank you to my grandparents, Ian, Teresa, and William, for your constant encouragement and always been there for me. To my grandmother Christine, I wish you were still here to see me finish this PhD. I will always remember your love and kindness.

I would also like to thank my friends Paul, Tom, Chris, and Simon. Thank you for always been there to have a chat and to help me relax, even if I needed "five more minutes".

Finally, thank you to my colleagues from BIST for the inspiring discussions we have.

Acknowledgements

Funding for this research was given through a Engineering and Physical Sciences Research Council (EPSRC) PhD studentship.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Publications

- A. C. McDonnell and M. A. Trefzer, "The effect of system timescale on virtual node connectivity within delay-feedback reservoirs," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8

Alexander Charles McDonnell

September 2024

Chapter 1

Introduction

1.1 Motivation

Since the creation of the general-purpose digital processors, countless work-hours have gone into perfecting the technology, ever pushing it to its limits. However, with the rise of intelligent hardware and artificial intelligence, is the digital general-purpose paradigm a fundamentally limited approach?

Vast improvements have been made within transistor technology in the last couple of decades, with expected sizes to reach 1.4nm by 2030. The reduction in transistor size has enabled high-density, high-performance, and low-power transistor devices to be fabricated at significantly reduced cost. This has enabled computationally expensive models, such as real-time object recognition, to be explored which were otherwise not feasible. It has also allowed the creation of intelligent hardware; a hardware architecture where hardware, sensors, and/or actuators are bundled together at the system level.

Although modern digital computing paradigms are capable of solving such complex problems, it is far from efficient. The inherent flaw is that a digital computing paradigm is applied to process real-world information, which is analogue. This conversion from the analogue to digital world carries many problems, from interfacing to encoding, and is a universal problem in real-time computational tasks; such as image and audio processing.

Another problem with modern digital computing paradigms is that they typically use general-purpose Turing-complete processors. This makes them highly effective at performing simple mathematical calculations, but struggle with anything complex. The problem lies in the information flow within the computing architecture. A typical Von Neumann architecture will process a series of instructions without any context to the information it is processing. It is this rigidity of the instructions that makes these devices unable to adapt to the information that is processed. This type of problem can be reduced when designing an artificial intelligence computational model by having semi-specialised hardware, such as a graphical processing unit, to better

match the native information flow within the computational model of the hardware. The specialised hardware reduces its ability to be fully general-purpose, allowing it to greatly improve the efficiency.

In order to address such problems, inspiration can be taken from alternative computing paradigms; such as unconventional computing.

1.2 Unconventional Computing

The concept of unconventional computing can be applied to any type of computing paradigm that does not compute using a traditional digital algorithmic approach to produce a well-defined output [12]. A large subset of unconventional computing systems are input-driven, where an input stimuli perturbs the dynamics of a system to produce a given behaviour. These dynamics could be mechanical, biological, or rely on the underlying physics of a substrate, and encompasses an enormous range of concepts, from quantum computing [13] to the behaviour of slime moulds [14]. Unconventional computing attempts to solve some of the limitations within traditional digital computing paradigms by attempting to exploit the dynamics and many-scale interactions of a substrate to perform computation. Although the term unconventional computing is relatively new, coined in 1998 by C. Calude and J. Casti [15], the earliest computers were analogue and mechanical in nature; classifying them as a subset of unconventional computing.

The oldest use of an analogue computer dates to the 1st century BC, where a geared system known as the Antikythera Mechanism was used for astronomical calculations [16]. As the mathematics behind gears were well defined, a mathematical model could be translated to a gear-based substrate to perform a computation. Information could then be fed into the substrate continuously and in real-time, producing an output that is similarly continuous and real-time; processing information in an analogue format. Analogue computers can solve a specific application very quickly but they

lack generalisation; an attribute which is very common within modern unconventional computing paradigms due to the nature of processing information within a specialised substrate. Although analogue computing has less prominence in today's world, the concept of processing information in its native format has long lived on.

A promising source of inspiration is to look towards biological systems, as many of the complex computational problems faced today rely on processing real-world data in real-time. Biological systems are uniquely adapted for performing a wide range of different tasks by processing a massive number of environmental stimuli concurrently. Unlike digital computing which follows an algorithmic approach, biological systems rely on learning to solve complex computational tasks. The human brain, our computing unit, is structured in a way that it contains billions of neurons and synapses that can form any number of networks to solve a task [17]. It is highly adaptable, both in terms of computing and fault tolerance, and learns from real-time real-world data; the ideal computing architecture to solve modern problems. Unfortunately, human type brain networks are very complex and not widely known or understood, but the fundamental concept of how biological systems process information can be applied to several unconventional computing paradigms; such as neuromorphic hardware [18] and Artificial Neural Networks [19].

1.3 Research Hypothesis

A particularly powerful approach to unconventional computing is the Reservoir Computing Framework. The reservoir computing framework is a machine learning paradigm inspired from the world of neurobiology. The foundation of the reservoir computing framework comes from the field of complex neural networks, specifically Recurrent Neural Networks, that exhibit interesting dynamical behaviours [20, 21]. Theoretically, this framework can be applied to any type of dynamical system that exhibits high-dimensionality, non-linearity, and fading memory. The reservoir computing

framework has already been successfully demonstrated on a wide range of traditional hardware, such as FPGAs and analogue circuitry [22, 10], to more unconventional systems, such as memristors and even a bucket of water [23, 24].

Many types of reservoir computing require a large amount of randomly connected recurrent neurons, which often requires a lot of space within hardware. As an alternative, Appeltant introduced an alternative type of reservoir computing based upon delay systems theory called a Delay-Feedback Reservoir [8]. The advantage of a delay-feedback reservoir is that it only requires a single non-linear neuron and a time delay to emulate many "virtual neurons" effectively emulating a much larger spatial recurrent neural network [25], allowing for a more compact and efficient use of hardware resources; allowing it to have a smaller silicon footprint than other traditional reservoir computing systems. Several interesting physical implementations of a delay-feedback reservoir have been realised with both commercial components, such as FPGAs and op-amps [26, 10], and more bespoke designs using photonic and optoelectronic techniques [27, 28]; both type of designs showing promising results in performing time series prediction and other temporal tasks.

Current physical implementations of delay-feedback reservoir systems are built from a top-down approach; a hardware setup is created that exhibits some of the key properties required for a delay-feedback reservoir system (for example: non-linearity, memory capacity, or high dimensionality) and the delay-feedback reservoir computing framework is applied to it. This approach is fundamentally flawed as it leads to an inefficient use of hardware resources; there is no clear design framework indicating how different internal parameters within a delay-feedback reservoir system affect the computational performance of the overall system.

In order to create efficient, low power, high-performance delay-feedback reservoir systems, the fundamental building blocks of the non-linear node need to be evaluated so that systems can be tuned to provide the required dynamics to solve a particular

computational task within a physical substrate; which can be identified in advance through signal processing techniques; such as a Fourier Transform.

This thesis aims to analyse the non-linear node within physical delay-feedback reservoir systems to better understand their function and internal parameters, so that the reservoir characteristics can be tuned to provide optimal performance for a particular computational task within a physical substrate.

Hypothesis: Tuning the non-linear node within a physical delay-feedback reservoir system provides optimal performance for a particular computational task while simplifying the hardware implementation of a typical delay-feedback reservoir system.

This main hypothesis can be broken down into the following sub-hypotheses.

Sub-Hypothesis 1: Computational tasks with particular characteristics perform best within distinct areas of the parameter space of the delay-feedback reservoir system model.

Sub-Hypothesis 2: Adjusting the timescale of the integrator within the non-linear node can lead to an increase in performance without a significant change in the hardware architecture.

Sub-Hypothesis 3: Changing the behaviour of the Mackey-Glass non-linear function can lead to an increase in performance within different computational tasks, with minimal changes to the hardware architecture.

Sub-Hypothesis 4: Replacing the non-linear function and integration stage of a delay-feedback reservoir with a higher order filter can lead to an increase in performance while simplifying the hardware architecture of a delay-feedback reservoir system model.

In order to test the validity of these hypotheses, the following objectives will be met:

Objective 1: Analyse the existing physical delay-feedback reservoir systems to determine a hardware-accurate simulation model.

Objective 2: Investigate the effect that the integrator timescale within a delay-feedback reservoir has on system performance and dynamics.

Objective 3: Investigate the effect that the non-linear function within a delay-feedback reservoir has on system performance and dynamics.

Objective 4: Determine if a high-order filter can be used as a non-linear node within a delay-feedback reservoir system.

1.4 Contributions

Throughout the completion of this thesis, the following contributions have been made to the academic field of physical Delay-Feedback Reservoir Computing:

- The creation of a generic parameterisable delay-feedback reservoir model, including the Mackey-Glass non-linear function and a generalised transfer function integration stage; developed within MATLAB and Simulink.
- The development of an evolutionary algorithm used to determine the optimal parameters of a complex high-order filter operating in the context of a delay-feedback reservoir system; developed within MATLAB and Simulink.
- The methodology to split the non-linear node into its independent subsequent key parts and optimise them independently.
- The methodology to tune the performance of the non-linear function and integration stage separately, based upon system performance or system metrics: kernel quality, generalisation rank, and linear memory capacity.
- The methodology to tune a delay-feedback reservoir to optimally solve a computational benchmark.

- The knowledge that "computational tasks that have a large dependency on memory" can gain additional performance by increasing the timescale of the system.
- The knowledge that the Mackey-Glass non-linear function has three operating modes that provide different non-linear behaviours and dynamics, which can be changed by setting the Mackey-Glass exponent to either one, odd, or even.
- The knowledge that a high-order filter can replace the integration stage within a typical non-linear node to supplement the dynamics of a non-linear function to achieve a greater performance.
- The knowledge that a computational task has a particular region of best performance within the parameter space of the delay-feedback reservoir system model.

1.5 Thesis Overview

This thesis contains eight chapters and is outlined as follows:

- Chapter 2 provides a background in Hybrid Computing Architectures, Artificial Neural Networks, and Reservoir Computing, with a focus on Delay-Feedback Reservoir Computing which is used within this work.
- Chapter 3 details how reservoir computing systems are trained and evaluated, with a focus on the computational benchmarks and system metrics used extensively within this work.
- Chapter 4 examines existing physical delay-feedback reservoir systems to create a typical hardware model, that is then expanded to create a hardware friendly simulation model, and outlines the common simulation methodology.

- Chapter 5 investigates the effect that the timescale of a first-order system has on a delay-feedback reservoir.
- Chapter 6 investigates the effect that the Mackey-Glass non-linear function has on a delay-feedback reservoir.
- Chapter 7 investigates the use of a high-order filter as a non-linear node replacement within a delay-feedback reservoir.
- Chapter 8 provides an overview of the thesis and discusses possible future work.

Chapter 2

Background

2.1 Hybrid Computing Architectures

In 1967, Gordon Moore posited that the number of components on an integrated chip will double approximately every two years, known as Moore's Law [29]. In today's world, this is not sustainable as we are reaching physical limits in transistor sizes and clock speeds, so alternative solutions must be found. An interesting solution is an attempt to bridge the gap between analogue and digital computers using modern technology, to create hybrid devices.

Analogue computers are much faster when interfacing with the real world, as data can be processed in its native analogue form, and can be processed at incredible speeds (in theory, data processing in analogue is limited only by the transmission mediums in which it is processed [30]); however, they are often complex, non-versatile, and lacking in precision. Digital computers are almost the opposite, they are very versatile, allowing for a wide range of functions to be performed, and can compute with great precision, but as they are discrete systems, they struggle to interface with the real world, and their speed is limited to how many instructions can be performed per second. Hybrid devices attempt to utilise the benefits of both digital and analogue computing together to create a device that outperforms their individual parts; one example of this is mixed analogue-digital hardware developed in Very Large-Scale Integration (VLSI).

Mixed analogue-digital VLSI devices are inspired from the original analogue computers that solved ODEs. These types of devices are often called "co-processors" as they are used in tangent with traditional digital systems. A general-purpose co-processor was outlined in 2005 by Cowan, Melville, and Tsividis, which contains integrators, programmable non-linear blocks, logarithmic and exponential blocks, and interconnect and fan-out hardware [31]. This was constructed in TSMC's 0.25-micron technology and shown to be 24 times faster in transient noise simulation when compared to a Sun Blade 1000 workstation [31]. In 2015, a smaller device aimed at solving up to 4th order ODEs was designed and constructed by Guo, et al [32]. The device was built on the more modern TSMC 25nm process, and tested on solving a model predictive

control (MPC) problem on a differential-drive robot [33]; once again showing promising results. While co-processors have shown promising results for solving ODEs, a more general-purpose substrate has been developed to provide the desired hybrid nature; these are known as Field Programmable Analogue Arrays (FPAAs).

Field Programmable Analogue Arrays.

The FPAA allows for real-time analogue signal processing to be performed on an integrated circuit. The key concept around an FPAA is that they contain Configurable Analogue Blocks (CABs) that allow for reconfigurable analogue hardware, this could be anywhere from integrators to filters. They traditionally operate in either the continuous or the discrete time domain.

When operating in continuous time, hardware resources within the CABs are simply connected together with programmable switches, and can operate at their full bandwidth; the downside to these designs is that they require very complex routing mechanisms, which often introduce parasitic behaviours. Continuous time FPAAs were first built in 1991 by Lee and Gulak and were used to solve simple neural network problems [34]; however, as the FPAA has poor configurable tolerances due to relying on varying the values of each transistor's transconductance, and requires a complex routing solution, the discrete time domain was explored.

One of the greatest advances of using the discrete time domain is that the FPAA could take advantage of switched capacitor technology. Switched capacitor technology allows for a resistance and/or capacitance to be set by switching a capacitor on and off at a specific frequency. Although the input data would have to be sampled in time, ultimately limiting bandwidth, it allows for programmable operation amplifiers to be used to implement many analogue designs. The use of programmable operation amplifiers and switch capacitors allows easy programming, as they only rely on a switching frequency, and it is simple to implement using CMOS technology. The first switched capacitor FPAA was proposed in 1996 by Kutuk and Kang and was used to implement a fourth-order band-pass filter [35]. There are a limited amount

of manufacturers of FPAAs, but one of note is Anadigm. Anadigm use switched capacitor technology to create integrated FPAA chips which can perform both analogue and digital processing capabilities. Further research is being performed with FPAA technology, with researchers developing a system-on-chip type of FPAA [36].

2.2 Artificial Neural Networks

While digital computers were gaining traction in the early 1940's, another method of computation inspired from biological system was proposed; the Artificial Neural Network (ANN). In today's world, ANNs can be broadly categorised into two types, feedforward networks, where data is applied at the input and the data propagates forward through a layered topology of neurons to an output, or Recurrent Neural Networks (RNNs), which allows data to be fed back into previous neurons, allowing data to propagate back and forward through a network, allowing a memory of previously processed data to linger within the network.

2.2.1 Feed-forward Neural Networks

Inspired from the world of Neuroscience, in 1943 McCulloch and Pitts developed the first computational model for neural networks, proposing that algorithms could be implemented using threshold logic [37]. Several simple networks were proposed using electrical circuits in order to describe how real neurons are modelled. These circuits had fixed weights, a scalar multiplier applied to an input of a neuron, and a fixed activation threshold; the activation threshold would then represent the output of a neuron. The idea was that different configurations could create "intelligence"; however, due to the fixed nature of the networks, they proved to be very application specific, for example, solving the Boolean logic for an AND or OR function.

Based upon the work of McCulloch and Pitts, in 1949 Hebbs proposed that in real brains, neurons are not fixed in nature and experience neuroplasticity, allowing

a neural network to reconfigure itself based upon growth and the input of previous stimuli [38]; this became known as Hebbian learning, the foundation of almost all learning techniques. The first Hebbian systems were developed by Clark and Farley in 1954, proving Hebbian learning could be applied practically [39].

The first great leap in practical neural networks was in 1958 when Rosenblatt proposed the idea of the perceptron [40]. The perceptron is based upon the neuron proposed by McCulloch and Pitts, however it employs a feedback system from the output to adjust the value of the weights against the desired value (shown in figure 2.1); this is known as supervised learning.

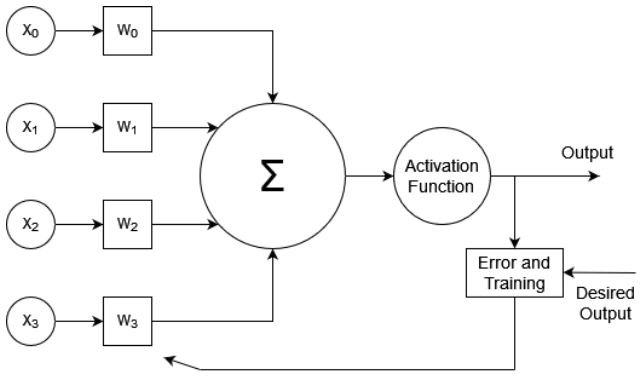

Figure 2.1 A 4-input perceptron neuron. The inputs of the perceptron are first weighted, and then the weighted sums are passed through an activation function. If there is a difference between the output of the perceptron and the desired output, then the input weights are adjusted to minimise the error.

The first perceptron machine, called the Mark I Perceptron, was built in 1960 and used to classify images. It contained 20x20 photocells so that a 400-pixel image could be processed, and had feedback-enabled programmable weights implemented through potentiometers and electric motors [41]. One issue that was found with the perceptron is that it converged to a solution very slowly. This was because the activation function that was used, called the Heaviside function, output a binary response. This meant that if a difference in input and output was found, the system had no way of finding out how great the error was, only that there was one. However, this problem was soon fixed with only a slight modification to the perceptron.

In 1960, Widrow and Hoff developed an alternative model to the perceptron with the hope of speeding up the convergence time, this model is called the Adaptive Linear Element; also known as ADALINE [42]. The main difference between a perceptron and ADALINE is that instead of using the neuron output after the activation function in the feedback, the summation of the inputs is taken just before the activation function (shown in figure 2.2). This allows for a cost function to be used to determine accurately how much of an error is present.

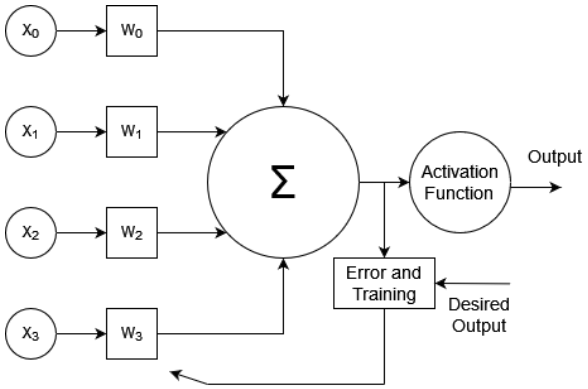

Figure 2.2 A 4-input ADALINE neuron. The inputs of the perceptron are first weighted, and then the weighted sums are passed through an activation function. The error is found by calculating the difference between the output of the summation node and a desired output. A training algorithm, set to minimise the error, then adjusts the input weights accordingly.

Although ADALINE was shown to outperform the perceptron in computation speed and was successfully physically implemented using memistors [42], a book wrote by Minsky and Papert in 1969 highlighted perhaps the biggest problem with single-layer networks; that they are limited to linearly separable problems [43].

A linearly separable system is one that can be bisected by a single linear function. An example of this would be to consider a square which has been divided in to two sections by a single line; one section is blue and the other is red. The only way to separate these sections is to place a line at the bisection point, this way only blue is present on one side and red on the other. The same logic can be applied to Boolean functions.

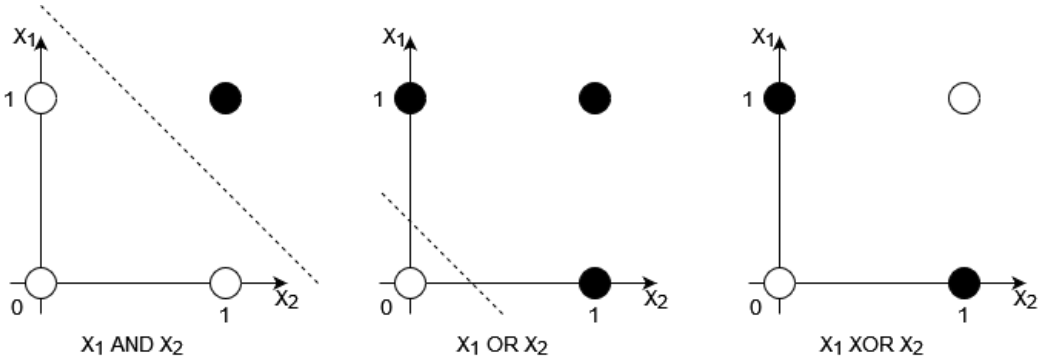

Figure 2.3 Three graphs showing the outputs of three Boolean functions: AND, OR, and XOR respectively. It shows that both the AND and OR functions are linearly separable, as indicated by the dotted line, while the XOR function is not.

As shown in Figure 2.3, both the AND and OR functions are linearly separable, but the XOR is not; this restricts single-layer networks as linear classifiers. It was known that having a multi-layered network would allow for non-linearly separable problems to be solved, however, the technology did not currently exist to perform learning algorithms on multi-layered networks [43]. A feedforward multi-layered network is a network that contains at least two layers, an input and output layer, but typically multiple "hidden layers" are placed between the input and output layers. Information within the network only moves from input to output, with each connection having their own weight; an example of a multi-layered network is shown in figure 2.4.

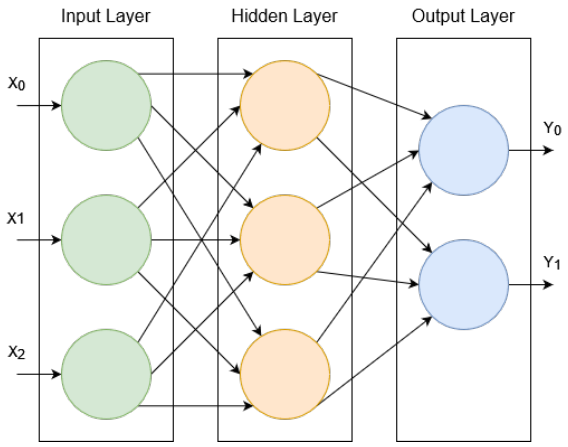

Figure 2.4 A 3-input and 2-output multi-layered network. The figure shows a fully connected system, with the flow of information moving from input to output.

The issue with this type of network is that traditional training algorithms are very computationally expensive, as on each update, every weight within the network needs to be updated, which is also dependent on its previous weight. This led to a gap of inactivity within ANN research until 1986, where an alternative method for training multi-layered networks was found.

The solution to training multi-layered networks was proposed by Rumelhart, Hinton, and Williams, named backpropagation [44], which was inspired from Kelly; who proposed it as a new method of computing gradients in flight control systems in the 1960's [45]. Consider a simple multi-layered network with three layers, an input, a hidden, and an output layer. Backpropagation can be described by [46]: first, calculating the error between the real and desired output; second, to calculate the new weights between the output and hidden layer based upon the error; third, apply the new weights and calculate the error terms between the hidden and input layer; finally, to calculate the new weights between the hidden and input layer based upon the error.

Backpropagation once again brought ANNs back into the spotlight as it was fast and easy to program, and as backpropagation uses chain rule to calculate the error, it can be extrapolated to any number of inputs, and requires little knowledge of the network; however, it does have some flaws. As the cost function relies on finding the global minimum of the error, it is possible for backpropagation to get stuck within a local minimum. Another flaw is that if the activation function of the neuron is chosen poorly [47], for example if the error has a small or large derivative, then the vanishing and/or the exploding gradient problem may occur, where the gradient of the error becomes too small or too big respectively. This is a huge problem as the gradient of the error function is directly proportional to the new weights (although it could be argued that with modern day GPU technology, the vanishing and the exploding gradient problem is no longer an issue [48]). Another flaw is that fully connected networks are prone to over-fitting data; although this can be avoided if a sufficiently sparse network is chosen [49]. Despite these issues, backpropagation has made ANNs take a huge leap forward in data processing. A particular interest in multi-layered

systems has often been in object classification; this focus has led to the creation of a subbranch of ANNs called Convolutional Neural Networks (CNNs) [50].

CNNs are specialised ANNs that focus on picking out features within data. They often utilise convolutional layers, hence their name, to down-sample input data; one early example of this is the polling layer [51]. In a polling layer, the input data matrix is divided into sections. Polling is then performed on each section, to either determine the maximum, minimum, or average value of the section. The output for each section creates a smaller matrix, which is then passed to the next layer; an example of a CNN is shown in figure 2.5

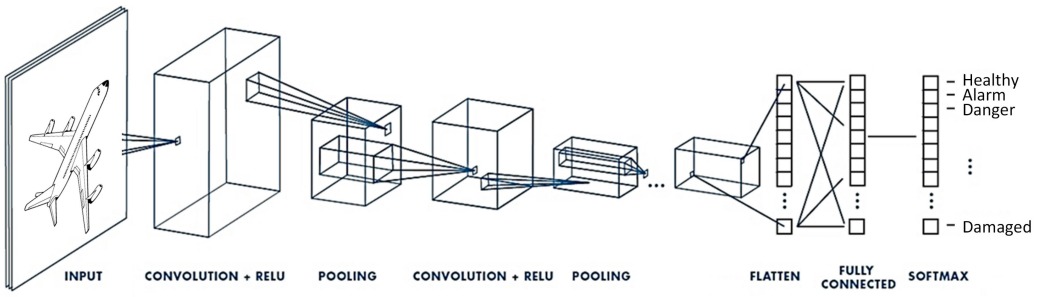

Figure 2.5 An example of a typical Convolutional Neural Network which is being used to monitor the structural health of an aeroplane, indicating any faults at its output layer [1].

Although feedforward neural networks have shown to be an effective tool as a classifier, they are not computationally universal [52].

2.2.2 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a variant of the feedforward neural network where processed data is allowed to travel backwards as well as forwards (shown in figure 2.6) [53]. This feedback between the output and input introduces two new key behaviours: first, as a signal can be propagated and/or looped for any number of times, RNNs become universally computational [54]; and second, as the output signal is now dependant upon its input in time, an RNN can be modelled as a dynamical

system [55]. This makes RNNs ideally suited to problems that rely on their previous input for classification or calculation, such as speech or image recognition, but have the downside of being difficult to train.

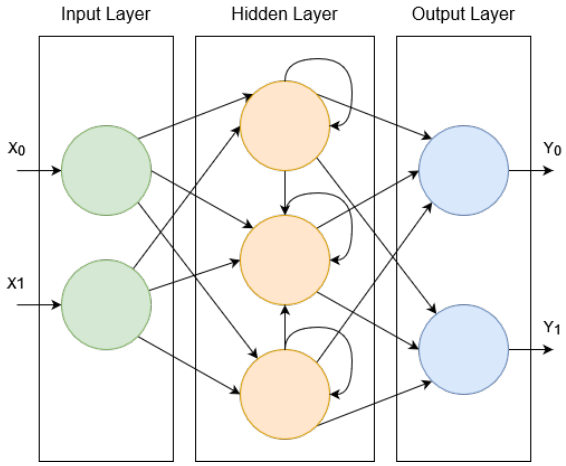

Figure 2.6 A 2-input and 2-output Recurrent Neural Network. The figure shows a fully connected network, with feedback and inter-connections at each node within the hidden layer.

Typically, RNNs are trained through a method called "backpropagation through time", which attempts to use standard backpropagation methods by first unfolding a RNN in time, so it appears to be a feedforward network [56]. However, this introduces the same errors as with standard backpropagation, with the vanishing gradient becoming more prevalent as data is passed through the same node multiple times with the same weights and threshold, decreasing the gradient of the error. In an attempt to minimise these issues, a standardised model for a RNN was developed.

The first, and most popular, standardised model for a RNN is called a Long Short-term Memory (LSTM) network, proposed by Hochreiter and Schmidhuber in 1997 to create a gated cell that can control the flow of information (shown in figure 2.7) [57]. A typical LSTM consists of a neuron, and three gates: an input gate, which controls new data being passed into the neuron; an output gate, that controls the output flow of the data; and a forget gate, that can remove the stored information within the neuron [57]. It has been shown that LSTM networks are much easier to train than

traditional RNNs as they minimise the vanishing gradient problem as gradients are unchanged between cell updates [56]; however, they are still prone to the exploding gradient problem. LSTM networks have been shown to solve a number of time domain problems, often in record time; such as Mackey-Glass time series prediction in 2005 [58], and suture knot tying in medical robots in 2006 [59]. Another interesting application of RNNs is in deep learning.

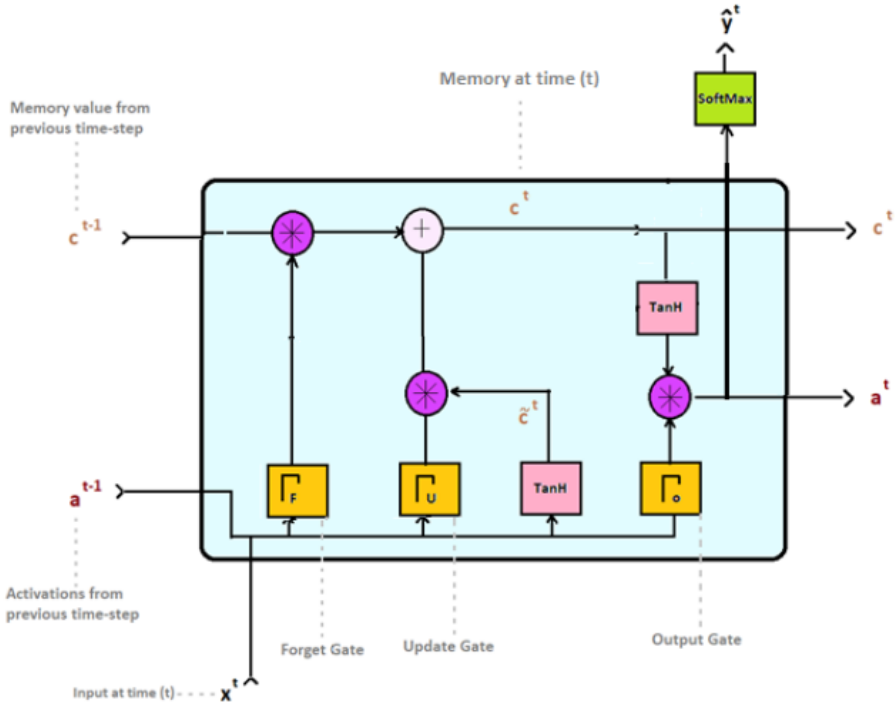

Figure 2.7 An internal diagram of a typical long short-term memory neuron. It contains three gates: an input gate, which controls new data being passed into the neuron; an output gate, that controls the output flow of the data; and a forget gate, that can remove the stored information within the neuron [2].

Deep learning [60] is a subset of machine learning [61], which is a type of learning where an algorithm adapts through experience; typically, without human intervention. Machine learning is often used to process large amounts of data in the hopes of extracting features and/or patterns, a task that is well-suited to ANNs; the use of ANNs (or sometimes any biological solution) to solve deep learning problems is called deep learning. As ANNs have roots within hardware, this has led to VLSI components

being developed to speed up tasks by attempting to learn their behaviours, this is known as neuromorphic engineering, although this technology, as seen in the public domain, is still within its infancy [62].

2.2.3 Spiking Neural Networks

Unlike other ANN models a Spiking Neural Network (SNN) utilises a spike train, which are discrete spikes that are produced within continuous time, to perform computation, rather than continuous input signals. This method is inspired from real biological neutrons [63], where their dynamics are modelled within each SNN neuron.

The dynamics of a neuron within a SNN normally utilises a leaky integrate-and-fire model, as this is most biologically accurate, with excitatory and inhibitory connections; where an excitatory signal increases the neuron potential, while an inhibitory signal will decrease it. Neurons may be further modelled to include a non-zero reset value, so that neurons can be stimulated above or below their resting potential, and refractory period, so that neurons do not become over saturated [64]. Figure 2.8 shows the dynamics of a real neuron.

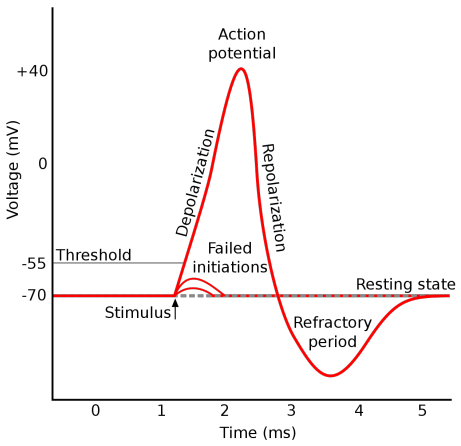

Figure 2.8 The dynamics of a biologically accurate neuron. The figure shows an excitatory stimulus being applied, and when its threshold value has been reached, the neuron depolarises towards its action potential. Once the neuron's action potential has been reached, it begins repolarisation, traveling past its resting state and settling back to its resting state after its refractory period. [3]

There are many methods of encoding a real valued number into a SNN spike train, usually using either rate or temporal coding schemes [65], and computation can be interpreted by using either deterministic or stochastic methods [66].

2.3 Reservoir Computing

The rise of Artificial Neural Networks (ANNs) has allowed once complex and expensive algorithms, such as classification and optimisation, to be applied with ease. Despite their success, a subsection of ANNs called the Recurrent Neural Network (RNN), is often tentatively approached due to their complex nature and difficulty with training. However, RNNs were seen in a new light when they were considered purely as complex dynamical systems [20], which lead to an alternative training approach; thus, creating the field of Reservoir Computing (RC).

The RC framework is a generalisation of three concepts: the Echo State Network (ESN), proposed by Jaeger in 2001 [21], and the Liquid State Machine (LSM), proposed by Maass in 2002 [67], which are two new computing paradigms that use sparsely connected RNNs at their heart; and the Backpropagation Decorrelation learning rule (BPDC), proposed by Schiller and Steil in 2005 [68], an on-line learning rule to efficiently train RNNs. This framework allows for a quick and simple way to train RC systems by considering the RNN as a dynamical "black-box", and only training the output of the system.

Although the RC framework was first derived from RNNs, it can be extended for use in any dynamical system that exhibits the following behaviours: a non-linear transform of the input signal, being able to represent the inputs in a high-dimensional state space, and to have a short-term memory of the previous inputs of the system. This has led to the RC network being applied to many non-typical systems, such as carbon nanotubes [69] and even a bucket of water [70].

2.3.1 Echo State Networks

The Echo State Network (ESN) is very similar in appearance to a traditional RNN, with both typically containing an input, output, and hidden layer (or more commonly known as the reservoir layer); several key differences are discussed below.

Unlike an RNN, an ESN operates in discrete-time [21, 71]. The network is updated in time-steps, notated as n, allowing for the network to be viewed as states at $x(n)$; where the function $x(n)$ is the state update equation. A network will have a dimensional input size of d and an dimensional output size of l, with a total number of neurons N. The system inputs, notated as $u(n)$, are connected to the reservoir layer with a weight matrix, $W^{in} \in R^{N \times d}$. Similarly, the system output, notated as $y(n)$, is coupled to the reservoir layer with a weight matrix, $W^{out} \in R^{l \times N}$. The reservoir layer contains a number of sparsely connected analogue neurons; notated as the matrix $W \in R^{N \times N}$. It is important that the neurons are analogue in nature, so that the inputs are transformed non-linearly [72], and that the neurons are connected recurrently, so that a fading memory (or echo) exists within the reservoir [72]. Within the reservoir layer, the inter-neuron connections have a randomly generated weight applied; these weights remain fixed. Typically, an ESN neuron is modelled using the leaky-integrator neuron model, where the state equation is:

$$x(n) = (1 - \alpha)x(n-1) + \alpha\, tanh(W^{in}u(n) + Wx(n-1)) \tag{2.1}$$

A decay rate is usually added to allow for control over the update dynamics of the reservoir; it is notated as α within equation 2.1. It should be noted that in this example the activation function is the hyperbolic tangent function, $tanh()$; however, other functions can be used to provide different activation dynamics. The structure of an ESN is shown in figure 2.9.

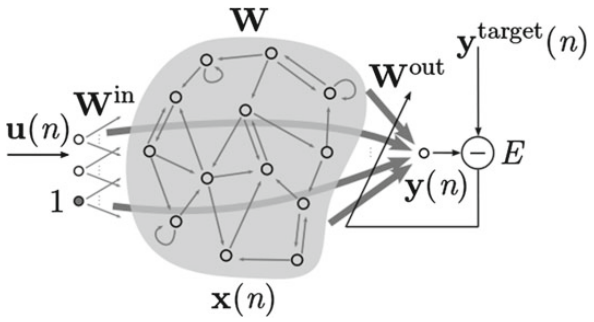

Figure 2.9 The structure of an echo state network. The input signal, $u(n)$, which is multiplied by the input weight matrix, $W^{in} \in R^{N \times d}$, propagates around the reservoir layer, $W \in R^{N \times N}$, as $x(n)$. The output states of the reservoir are taken, $y(n)$, which are multiplied by the output weight matrix, $W^{out} \in R^{l \times N}$. Training is then performed by minimising the error between the target output, $y^{target}(n)$, and the actual output, $y(n)$; the output weight matrix, W^{out}, is then updated according to this error. An additional 1 is applied to the input of the reservoir layer, which is used to bias the network. [4]

The system can then be simply trained by adjusting the output weights, W^{out}, until the error between $y(n)$ and the target output is minimised to zero [71].

ESNs have shown to exhibit exceptional performance in time-series prediction and temporal pattern detection with minimal training due to their dynamics. This was shown in the original publication by Jaeger when the concept of an ESN was proposed, and is shown in subsequent work [73, 74]. Two of the greatest strengths of these types of system is that they are remarkably easy to train with off-line supervised learning, and because the dynamics of the reservoir are fixed, with only the output weights changing during training, the inputs to the reservoir shape the dynamics of the reservoir to become a solution [21]; however, they are not without fault and require some careful consideration.

One of the most important factors within an ESN is how many neurons should the reservoir layer contain. As each neuron adds additional short-term memory into the system, it may seem intuitive to want to create a reservoir as big as computationally, or physically, possible. However, this will create a system that is prone to overfitting, unless measures are taken within the training algorithm to combat this, and will often

have diminishing returns after a given threshold. Instead, Lukoševičius suggests to start with smaller reservoirs, and to increase the size of the reservoir if required [4].

Another important factor to consider is the spectral radius of the reservoir matrix, W [4]. This parameter defines the weights between neurons, and how sparsely connected they are; this effectively sets the time constant of the reservoir and defines how much non-linear behaviour is present [75]. Mathematically, the spectral radius is the largest eigenvalue of W, and should be set so that its value is less than one so that the echo state property is guaranteed; although Lukoševičius argues this is not always necessary [4].

A fundamental aspect of the dynamical systems, and therefore ESNs, is that they have a natural timescale, set by the system dynamics. This is a crucial consideration as the timescale of the system needs to be fast enough so that an input signal can perturb the network, and allow the transients to move through the system; although it is often debated if it's best to let a system settle to its steady state, or utilise the transients of the system. Alternatively, if a system is slower than an input signal, then the system will not converge [21]. Usually, a parameter exists within the state equation, such as α in the leaky-integrator equation (see equation 2.1), to modify the system's timescale; however, this is only possible within a simulated system, whereas within a physical system, only the input can be adjusted.

Typically, most ESNs are implemented in software as they are easy to program, as the ESN model boils down to matrix manipulation, and are simple to train. Not many attempts have been made to implement such systems in hardware as ESNs are often quite large in neuron size, thus requiring lots of space on silicon; however, advancements in technology have opened up a new possibility for on-chip ESNs using memristors [76]. Although memristor technology is relatively new, several types of neural networks have been constructed using memristors, such as spiking neural networks [77], RNNs [78], and now ESNs [79].

2.3.2 Liquid State Machines

The Liquid State Machine (LSM) was proposed both simultaneously and independently of the ESN by Maass in 2002 [67]. Unlike the ESN, the LSM takes a neuroscientific approach to computing with the hope that it can model computational problems within cortical microcircuits more accurately than "previously proposed Turing machines and attractor-based dynamical systems" [80]. This is achieved by using the Spiking Neural Network (SNN) model, which is analogous to how real neurons function [80].

The structure of a LSM is similar to that of an ESN, with the only difference being that additional encoding/decoding hardware must be added to the input/output layers to convert between a spike train and a real valued number (the structure of a LSM is shown in figure 2.10). When using a LSM, the same parameters must be considered as when using an ESN, such as the size of reservoir and the spectral radius.

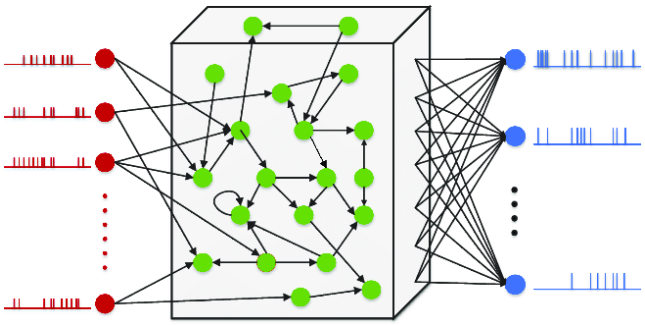

Figure 2.10 The structure of a LSM. The input of the system is a spike train, which passes through the reservoir layer which is fixed and randomly generated. The output layer, which is also a spike train, collects data from the reservoir layer and interprets the spike train based upon its design. [5]

One of the greatest strengths of the LSM is that they are very adaptive; and has in fact been shown to be a universal function approximator by the Stone–Weierstrass theorem, if the reservoir network exhibits input separability and fading memory [81]. This adaptability is due to how the LSM computes information; instead of simply implementing an algorithm, it maps input spike trains onto output spike trains. As a

result of this, the output is highly dependent upon the type of readout performed and the type of training performed [67]; therefore this is said to have multi-dimensional inputs and contextual outputs [80]. Consequently, multiple readout layers can be implemented, all extracting different information from the input data, allowing for parallel processing; however, this is often one of its greatest downsides.

The complexity of the LSM system means it's not always obvious what readout should be performed, hence why the standard is usually a linear readout; and the encoding scheme at the input layer can also have a huge effect on what the data is representing. These are additional complexities that must be considered when designing a LSM that are not present in other RC networks; although input scaling will always be a factor in any RC network.

Despite these flaws, LSMs have been applied to various tasks in hardware, such as: speech recognition [82], neural accelerators [5], and pattern recognition [83]. As the model of a LSM is digital in nature, due to their spike behaviour, LSMs have been implemented into FPGAs and VLSI technology.

2.3.3 Physical Reservoir Computing

As the interest within the field of Reservoir Computing has increased, the field has expanded to consider applying the Reservoir Computing framework to physical substrates rather than using software computational models. The Reservoir Computing framework states that it can be applied to any object or substrate which exhibits non-linearity, high-dimensionality, and short-term memory, which almost all physical substrates exhibit to some degree [70, 84, 85]. This means, that in theory, almost all substrates can be used to perform some form of computation; but the question is how well it can compute, how easy is it to interface with, and how energy efficient is it. The most popular physical devices utilise magnetic and optoelectronic substrates as they offer promising solutions in both strong computational dynamics and power efficiency.

The use of magnetic substrates has shown promising results as they are often dynamically rich, exploiting the physics of a substrate, such as artificial spin ice [86], spin-transfer torque [87, 88], or giant magnetoresistance [89], to provide these dynamics. As these substrates exploit physical phenomena, they can be low power and operate within a few milliwatts in some cases. An example of a magnetic nano-ring array reservoir computing system can be seen in figure 2.11.

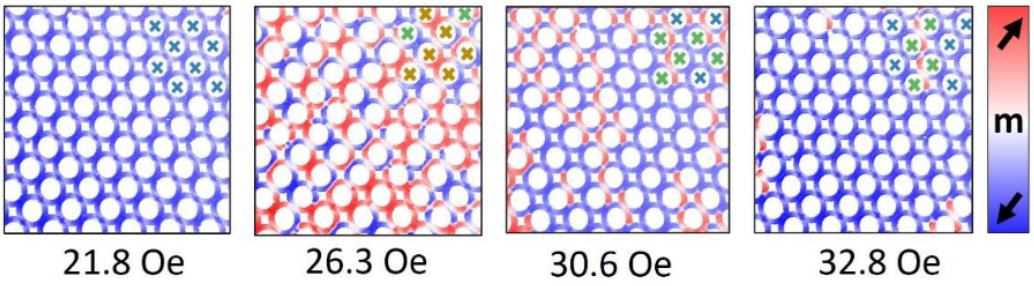

Figure 2.11 X-Ray photo-emission electron microscopy images of a magnetic nano-ring array showing the magnetisation when applied with different strengths magnetic fields, measured in Oersted's. [6]

Although there has been some success in using magnetic substrates for Reservoir Computing, their greatest weakness is within interfacing. Interfacing with these devices typically requires large, isolated electromagnets to input data and specialised equipment to read data from the device, making it impractical for real-world use. As the field of magnetic substrates mature, fabrication and interfacing equipment will become less specialised, increasing its practicability and decreasing its cost. An alternative approach is to use already commercially available electronic components and apply the Reservoir Computing framework to it; one such promising approach is the use of optoelectronic devices.

Optoelectronic and optical devices can be used to create an optical substrate, a type of substrate typically created using optical fibres or waveguides to exploit the dynamics of light in order to perform computation [90, 91, 28]. They have the advantage of offering high-speed processing with a low power consumption. An example of an optoelectronic implementation is shown in figure 2.12.

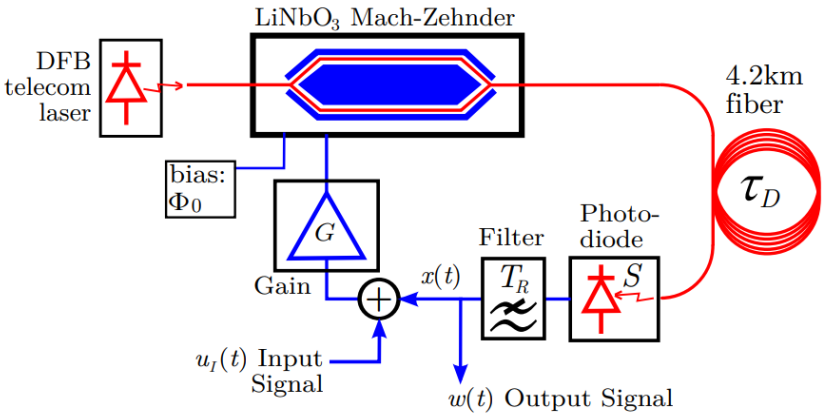

Figure 2.12 A block diagram of an optoelectronic implementation of a reservoir computing device utilising delayed feedback. [7]

However, much like magnetic substrates, interfacing can be a significant problem in optical substrates as information must be converted to light and manipulated within a controlled manner, which can be difficult at optical speeds.

In an attempt to solve the interfacing problems that most physical Reservoir Computing systems have, traditional electronic components have been used with new, unconventional devices [92], such as memristive devices [23, 93] or Micro-Electromechanical Systems (MEMS) [94], to create hybrid computing substrates. These types of devices have the advantage of simple interfacing, as interfacing is done through traditional analogue or digital components.

2.4 Delay-Feedback Reservoir Computing

Delayed Feedback Reservoir Computing, proposed by Appeltant in 2011 [8], is fundamentally different to other classical RC systems. While other RC systems have a reservoir layer that exhibit the properties of a RNN, a delay-feedback reservoir system contains only a single non-linear neuron and a delay-line. The system takes advantage of delay systems theory to emulate a much larger RNN network [25]. These emulated neurons, called virtual nodes, are created by time-multiplexing the input signal with a

higher frequency masking signal; however, this emulation comes at the cost of requiring a complex input layer.

The input signal, continuous $u(t)$ or discrete $u(k)$, undergoes a sample-and-hold operation to create $I(t)$. This signal is then time-multiplexed by a masking signal, $m(t)$, to create the input sequence, $J(t)$, which is the input to the non-linear node. The time period of the sample-and-hold operation, notated as τ, that typically defines the time delay within the delay-line; this process can be seen within figure 2.13. The number of virtual nodes, N, within the delay-feedback reservoir system can be calculated by:

$$N = \frac{\tau}{\theta} \tag{2.2}$$

Where θ is the time period of the masking signal. It should be noted that the first type of masking signals used were randomly generated; however Gan, et al has shown that masking signals can be optimised to increase performance in computation tasks [95]. While traditionally the time delay within the delay-line is equal to τ, this does not always have to be the case. When the time delay is equal to τ, each virtual node is connected to itself through the feedback loop. However when the time delay is not equal to τ, each virtual node is connected to a different virtual node within the system, allowing for the creation of more complex connection topologies between virtual nodes [96].

The general equation for a delay-feedback reservoir system, as stated by Appeltant [8], can be expressed as:

$$\dot{x}(t) = F(x(t), x(t-\tau) + J(t)) \tag{2.3}$$

Where $x(t)$ is the output of the non-linear node, $x(t-\tau)$ is the delayed output, and $J(t)$ is the masked input signal.

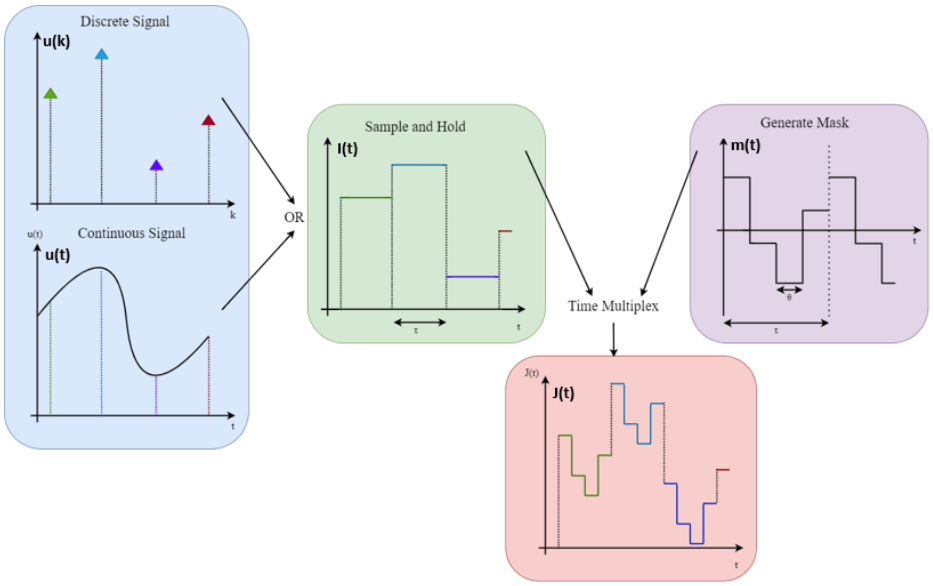

Figure 2.13 The input signal, continuous $u(t)$ or discrete $u(k)$, undergoes a sample-and-hold operation to create $I(t)$. This is then time-multiplexed by a masking signal, $m(t)$, to create the input sequence, $J(t)$. This input sequence is then fed into the non-linear node.

The output layer of the system collates the states of the delay-feedback reservoir system by linear combination. Linear combination is where the output layer collects the states within each of the virtual nodes, multiplies each node by a particular weight, and sums all the virtual nodes together within a single time step. This is can be mathematically written as:

$$\hat{y}_k = \sum_{i=1}^{N} w_i.x\left(k\tau - \frac{\tau}{N}(N-i)\right) \tag{2.4}$$

Where w_i is the weight attributed to each virtual node i, x is the output of the non-linear neuron at a given state, and $\hat{y}$ is the approximated output signal.

The readout weight matrix, typically referred to as W_{out}, is usually calculated as to minimise the mean square error between $|W_{out}X_{out} - Y_{target}|^2$, where X_{out} is a matrix containing the state space of the virtual nodes for several time steps, and Y_{out} is a vector containing the target output. Figure 2.14 shows the structure of a delay-feedback reservoir system.

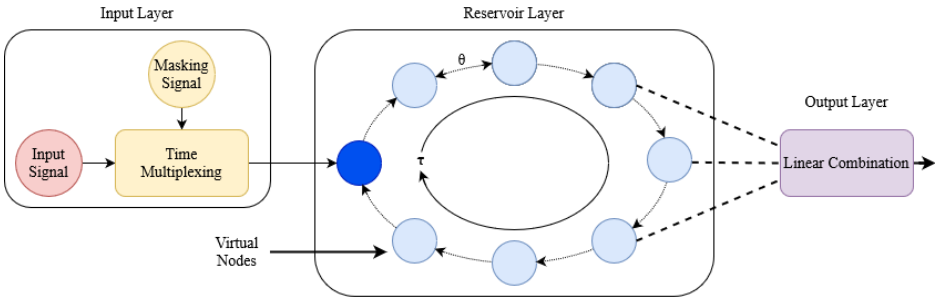

Figure 2.14 The structure of a Delayed Feedback Reservoir. A single non-linear node is used with a delay-line to create a topology of virtual nodes. The time period of the input signal, notated as τ, defines the time delay within the delay-line. The time period of the masking signal, θ, defines the spacing between the virtual nodes. The number of virtual nodes, N, can be calculated by $\frac{\tau}{\theta}$ [8].

The choice of θ defines the amount of virtual nodes present within the delay-feedback reservoir system for a fixed τ; the greater the virtual nodes, the greater the theoretical memory capability of the delay-feedback reservoir system could be, which is useful for solving computational tasks that rely on the memory of their past inputs. However, careful consideration must be taken when choosing θ as it also greatly affects the dynamics of the system, as well as the topology of the virtual nodes.

One of the major limitations of the delay-feedback reservoir is that it only supports a single input that must first undergo extensive signal processing; although efforts by Gan, et al have attempted to mitigate this single input problem by interleaving or sequentially applying inputs in order to solve a multiple input control problem [97].

A lot of physical implementations are either digital in nature [26], or used in a digital-analogue hybrid approach [10], where the non-linear node is analogue and the delay-line and pre-processing is done in digital; both of which have produced exceptional results in time-series prediction tasks. An alternative physical implementation is to use photonic and optoelectronic techniques, allowing for very fast and precise timings [27, 28]; far surpassing the speeds of traditional digital implementations. Additional care has to be taken with physical delay-feedback reservoirs as any propagation delays, which are added due to physical hardware, effectively increase τ. This can cause

numerous problems; such as timing mismatches when reading the output states of the virtual nodes, and the increase in τ may cause N to no longer be an integer, causing a time-dependant network topology.

Chapter 3

Training and Evaluation of Reservoir Systems

3.1 Training Reservoirs

There are two main methods of training the outputs of a reservoir. The first is "off-line" training, where the reservoir is run over a given training period, notated as T, with the states of the reservoir been collected into a matrix, notated as X, which is then used for training; the weights can then be calculated by using simple linear or ridge regression techniques. The second is "on-line" training, where states of the reservoir are collected continuously during the training period; the weights are then typically calculated using a gradient descent-based algorithm such as Recursive Least Squares (RLS).

Within this work only off-line training is used. However, the most popular off-line and on-line training algorithms are discussed here for reference.

3.1.1 Off-line Training

Off-line training is a supervised learning technique where the input of the reservoir, $u(t)$, is coupled together with a desired target output of the reservoir so that the system can learn the desired input-output behaviour by minimising the error, by calculating the mean squared error (MSE), between the output of the reservoir, $y(t)$, and the desired target output, $y_{target}(t)$. The equation for the MSE shown is expressed as:

$$E(y, y_{target}) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\frac{(y(n) - y_{target}(n))^2}{\sigma^2 y_{target}(n)}} \tag{3.1}$$

Off-line training occurs within a given training cycle, also known as batches, with the aim of producing an output weight matrix, W_{out}, that can be applied to the collected states of the reservoir to minimise the error between $y_{target}(t)$ and $y(t)$. This is done by driving the reservoir with the input signal, $u(t)$, and recording the reservoir states at each time step for the duration of the training period, T, to produce the reservoir state vector X. A vector of the desired output signal, $y_{target}(t)$, is also collated

over the training period of T to produce a target output vector of Y_{target}. These are defined as the following matrices:

$$X = [x(1), x(2), x(3), ..., x(T)] \tag{3.2}$$

$$Y_{target} = [y_{target}(1), y_{target}(2), y_{target}(3), ..., y_{target}(T)] \tag{3.3}$$

With the reservoir state vector X and desired target vector Y_{target} collected, the output weight matrix, W_{out}, can be calculated by solving the following overdetermined system:

$$Y_{target} = W_{out}X \tag{3.4}$$

The equation 3.4 can then be solved using linear regression, typically using Ordinary Least Squares. This can be expressed as:

$$W_{out} = Y_{target}X^T(XX^T)^{-1} \tag{3.5}$$

Although this is the simplest method of calculating W_{out}, attempting to invert the matrix XX^T typically results in stability issues, producing an output weight matrix which is unable to generalise and contains greatly amplified errors. In order to avoid stability issues, an alternative method is required to calculate the output weight matrix; a popular alternative is to use Ridge Regression.

Ridge Regression is a popular method of calculating W_{out}, as shown by the following formula:

$$W_{out} = Y_{target}X^T(XX^T - \lambda I)^{-1} \tag{3.6}$$

The advantage of ridge regression is the inclusion of the regression parameter λ. This addition can greatly increase the accuracy of the solution when the calculated weights are very large; which could indicate either an overly sensitive or unstable solution [4]. When the regression parameter is set to zero, ridge regression becomes the same as linear regression as shown in equation 3.5. Therefore it is recommend to set the regression parameter using a logarithmic scale for best performance [98].

An alternative to ridge regression is to use the Moore-Penrose pseudo-inverse function. This has the advantage of being present as a function in many modern-day programming environments, but it can produce an overdetermined result for large networks [4]. The pseudo-inverse training formula is given as:

$$W_{out} = Y_{target} X^{+} \tag{3.7}$$

Where + represents the use of the pseudo-inverse function.

While the Moore-Penrose pseudo-inverse function is often convenient, care should be taken when the reservoir state vector is large as it becomes computational expensive and often produces a overdetermined result.

3.1.2 On-line Training

While the off-line training algorithm uses the reservoir states and target output over a given training period to generate an output weight matrix, on-line training algorithms are iterative. An on-line training algorithm periodically calculates the output weight matrix based on the current state of the reservoir and current target output, then updates the output weight matrix on the next reservoir time step. This type of training is particularly useful when the system model to be trained is constantly adapting with time and being trained from non-stationary data; an off-line training algorithm using linear regression would not be able to calculate a weight matrix as the

reservoir is constantly changing. The two most popular on-line training algorithms are Recursive Least Squares (RLS) and Adaptive Moment Estimation (ADAM).

At its core, the RLS algorithm shares some similarities with the off-line training as the MSE is minimised between the predicted output of the reservoir, $y(t)$, and the desired output, $y_{target}(t)$. However, with the RLS algorithm, the output weight matrix, W_{out}, is calculated at each reservoir time step given the current reservoir state, $x(t)$, and desired output. The equation for the RLS is given as:

$$E(y, y_{target}, n) = \frac{1}{N_y} \sum_{i=1}^{N_y} \sum_{j=1}^{n} \lambda^{n-j} (y_{[i]}(j) - y_{target[i]}(j))^2 \tag{3.8}$$

The RLS algorithm utilises a forgetting parameter, λ, which controls how much of the previous input is included in the calculation, and an autocorrelation matrix, $\rho(t)$, which preserves information from previous time steps. The RLS algorithm is described as follows [99, 100]:

Initialisation: Set the forgetting parameter, λ, to a value near, but not equal to 1. Initialise the autocorrelation matrix to $\rho(0) = \delta I$, where δ is a large scalar and I is the identity matrix. Initialise the output weight matrix, W_{out}, with either random or zero values.

Step 1: Calculate the current reservoir state, $x(n)$, and predicted output value, $\hat{y}(n)$ as follows:

$$\hat{y}(n) = W_{out}(n-1)x(n) \tag{3.9}$$

Step 2: Calculate the error between the target output value, $y_{target}(n)$, and the predicted output value, $\hat{y}(n)$:

$$e(n) = y_{target}(n) - \hat{y}(n) \tag{3.10}$$

Step 3: Compute the Kalman gain vector, $K(n)$:

$$K(n) = \frac{\rho(n-1)x(n)}{\lambda + x(n)^T \rho(n-1)x(n)} \tag{3.11}$$

Step 4: Calculate the inverse correlation matrix, $\rho(n)$:

$$\rho(n) = \frac{1}{\lambda}(\rho(n-1) - K(n)x(n)^T \rho(n-1))) \tag{3.12}$$

Step 5: Update output weight matrix, W_{out}:

$$W_{out}(n) = W_{out}(n-1) + K(n)e(n) \tag{3.13}$$

Steps 1-5 repeat for the entire duration of the training length, T.

Although RLS is able to successfully train adaptive reservoirs, it comes at a computational cost as the number of weights scale quadratically rather than linearly, can be numerically unstable in convergence, and is highly sensitive to noise. An alternative approach is to use ADAM, an on-line training algorithm based upon gradient descent.

ADAM is a combination of two stochastic gradient descent methodologies, Root Mean Square Propagation (RMSProp) and Momentum, introduced by Kingma [101]. The algorithm utilises an adaptive learning rate and momentum based optimisation to allow for fast, stable convergence and provide robustness to noise and sparse gradients. Instead of using the loss function as the error like in RLS, ADAM calculates the gradient of the loss function at each time step. The gradient is then used to calculate the learning rate of the algorithm and a moving average of the gradient, known as its momentum.

While ADAM has become the increasingly popular in large-scale systems due to its robustness, it does suffer from a sensitivity to hyper-parameters which have to be manually tuned before peak performance can be reached.

3.1.3 Training Evaluation

In order to be able to compare the performance of the reservoir between different experimental runs and benchmark tasks, a measurable quantity of how well the reservoir has performed needs to be established.

Once the weight matrix has been calculated, W_{out} can then simply be multiplied by the output of the reservoir, $x(t)$, to determine the approximated output signal, $\hat{y}$; this can be expressed as:

$$\hat{y}(t) = W_{out}x(t) \tag{3.14}$$

The error between the trained reservoir output and its target can then be calculated, and is then often normalised against some statistical attribute. This is known as the normalised root mean square error (NRMSE), which is a common measurement of calculating the testing and training error. The NRMSE is expressed as:

$$NRMSE = \sqrt{\frac{1}{m}\frac{\sum_{k=1}^{m}(\hat{y}_{[k]} - y_{target[k]})^2}{\sigma^2(y_{target[k]})}} \tag{3.15}$$

Where, m is the number of data samples within the experiment, $\hat{y}_{[k]}$ is the trained reservoir output, $y_{target[k]}$ is the desired target function, and σ is the standard deviation.

A NRMSE result of 0 implies a perfect match between the trained reservoir output and the target function, whereas a result of 1 indicates the trained reservoir is approximating the mean value of the target output. Generally, the lower the NRMSE is, the better the reservoir is at the executed computational task.

3.2 Computational Benchmarks

In order to evaluate the performance of a reservoir computing system, computational benchmarks can be used to test how capable a system is and allows a comparison between systems. Benchmarks tasks can be categorised into four types: imitation, predictive, classification, and control.

An imitation task aims to replicate the input-output sequence of its target system, therefore *imitating* the target. This relationship tends to be superficial, as only a mapping of the inputs and outputs are trained. A predicative task attempts to emulate the dynamics of a target system with its own dynamics so that the next term in a sequence can be *predicted*. A classification task aims to identify a particular feature within an dataset. An example would be an attempt to detect a spoken phrase within auditory information or a particular object, such as a human, within visual information. Finally, the control task aims to provide feedback to an external system with the aim of changing its dynamics. This could take many forms, such as a controller within a closed-loop feedback system, or provide the driving signals for a chaotic system.

Within this work two types of tasks, imitation and prediction, are used extensively to test the performance of the modified Delay-Feedback Reservoirs.

3.2.1 NARMA

The non-linear autoregressive moving average task, or NARMA, is an imitation computational benchmark that is widely used to evaluate the performance of neural and reservoir networks [102]. The challenge of the benchmark is that it requires both non-linearity, albeit it weak, and a long-term dependency on previous input stimuli to evaluate the dynamical capability of a reservoir. The discrete-time NARMA-n benchmark is nth order, meaning it relies on memory of n previous inputs from a lag of n time steps.

The most popular variations are NARMA-5 and NARMA-10, which can be used to evaluate the memory performance of a Delay-Feedback Reservoir with different memory requirements.

The equation for the discrete-time NARMA-5 [103] model is given by:

$$y_{n+1} = 0.3y_n + 0.05y_n\left(\sum_{i=0}^{4} y_{n-i}\right) + 1.5u_n u_{n-4} + 0.1 \tag{3.16}$$

The discrete-time NARMA-10 [103] model is given by:

$$y_{n+1} = 0.3y_n + 0.05y_n\left(\sum_{i=0}^{9} y_{n-i}\right) + 1.5u_n u_{n-9} + 0.1 \tag{3.17}$$

When applying the NARMA as a benchmark to a reservoir computing system, the goal is to train a system on the input-output dynamics of the NARMA equation so that it can be recreated with the same input stimuli. The trained output sequence and NARMA equation output are then both compared so that the system performance can be evaluated, typically with the NRMSE.

3.2.2 Santa Fe Laser (A)

The Santa Fe Laser dataset A is a predictive highly non-linear computational benchmark derived from observing a far-infrared laser in a chaotic state [104]. The aim of the Santa Fe Laser benchmark is to attempt to train the reservoir system on the dynamics of the dataset so that the next n number of observations can be predicted. Typically, only the next observation is predicted, greatly reducing the dependency on previous input stimuli. This results in the Santa Fe Laser benchmark requiring a reservoir system to have a stronger non-linear high-dimensionality rather than fading memory.

3.3 System Metrics

Often, applying different computational benchmarks to a reservoir system gives very different results. Although benchmarks are useful in determining how well a reservoir system can handle a specific type of task, it offers little insight into the dynamical behaviours of the reservoir system. To allow a reservoir system to be more accurately evaluated and characterised, a set of metrics can be calculated for a particular system which is task agnostic. Here we discuss three of the most popular metrics.

3.3.1 Linear Memory Capacity

The linear memory capacity, or LMC, of a system is a measure of how well previous input stimuli can be recalled from a reservoir system. In terms of the LMC, it is a measure of how long a particular input can be stored within the reservoir before it degrades. This is an extremely useful tool as it is not often clear why a particular task has performed poorly on a reservoir system; measuring the LMC gives some indication if there is sufficient memory available within the system for a particular task.

The LMC of a system is calculated by injecting a random uniform distribution of numbers into the reservoir and then training the output to recover the previous inputs $u(k-i)$, for $i = 1, 2, 3..., 2N$ where N is the number of nodes within the reservoir, resulting in i outputs. The LMC is then measured by calculating the variance between the output of the reservoir and the delayed input, summed over all delays; the maximum memory capacity of a system is always $MC \leq N$ [105]. The equation for calculating LMC can be expressed as:

$$MC = \sum_{i=1}^{2N} \frac{cov^2\Big(u(k-i), y(k)\Big)}{\sigma^2\Big(u(k)\Big)\sigma^2\Big(y(k)\Big)} \tag{3.18}$$

This metric is called *linear* memory capacity as the reservoir is trained to recover its previous inputs, which are linearly related to one another. Another type of memory capacity that could be considered is non-linear memory capacity. This is a measure of a reservoir's ability to compute past inputs of a non-linear function [106], giving some insight into how non-linear functions transform over time. However, this type of memory capacity is extremely computationally expensive as the variance between the delayed non-linear functions must be computed for many different non-linear functions and is often not clear which non-linear functions should be included [107]; for example several degrees of polynomials or sinusoidal functions. Given that the linear memory capacity is generally a better metric for delay-feedback reservoirs [8] and the computational complexity of the non-linear memory capacity, it is decided not to use non-linear memory capacity within this work.

3.3.2 Kernel Quality

The kernel quality, or KQ, is a measure of how well a reservoir system can create a non-linear representation of different input streams. This can be viewed as how much dimensionality the system has, or how well the system is able to separate distinct input patterns. This measurement helps predict the heterogeneity of non-linear operations a system can perform, allowing the states of the reservoir to be linearly separable [108]. The KQ of a reservoir system can be calculated by the following steps:

- Generate m (ideally greater than the amount of nodes within the reservoir system) number of random input vectors, each with a length of k;
  $U = [u_1, u_2, u_3, ..., u_m]$, where u_i is a series of k random data points.
- Inject the set of input vectors, U, into the reservoir system.
- As the k data points are injected into the reservoir, collect the states of the reservoir system; creating a matrix of size nxm.

- Compute the singular values of the state matrix using singular value decomposition.
- The KQ is equal to the non-zero elements within the singular value matrix.

The maximum value the KQ is $KQ \leq N$, which implies that all of the injected set of input vectors have been independently mapped within the reservoir. The closer KQ is to N, the better the performance of the reservoir system will be.

3.3.3 Generalisation Rank

The generalisation rank, or GR, of a reservoir system is a measure of how well the system is able to generalise similar input streams. In a real system there is often external factors which degrade the input signal quality, such as noise. The GR allows some insight into how sensitive a reservoir system is and how well it can handle small variations within its input. Calculating the GR of a reservoir system is very similar to calculating KQ, the steps are as follows:

- Generate a single set of input vectors, notated as u, with k random data points.
- Create m copies of u (ideally greater than the amount of nodes within the reservoir system) and apply a random amount of noise to each data point within each input vector; $U' = [u'_1, u'_2, u'_3, ..., u'_m]$.
- Inject the set of input vectors, U', into the reservoir system.
- As the k data points are injected into the reservoir, collect the states of the reservoir system; creating a matrix of size nxm.
- Compute the singular values of the state matrix using singular value decomposition.
- The GR is equal to the non-zero elements within the singular value matrix.

The maximum value the GR is $GR \leq N$, but unlike the KQ, this implies a poor performance as every noisy input has been classified differently. An ideal GR is that of 1, where the system is able to generalise all input vectors; the closer the GR is to 1, the more robust the reservoir will be at handling input signal variances.

3.3.4 Computational Ability

An ideal reservoir system would have a KQ which is equal to N, and have a GR of 1. While it is useful to interpret these metrics independently, it can be useful to have a metric that allows for a quick insight into how well a reservoir system may perform; this is called the computational ability (CA) and can be expressed as:

$$CA = KQ - GR \tag{3.19}$$

The CA is at its maximum when the KQ is equal to N and the GR to 1, giving a maximum CA of N - 1, and the minimum occurs when KQ is equal to 1 and the GR to N, giving a minimum CA of 1 - N. However, the CA is often given as $CA \geq 1$ as negative values always indicate a poor performance.

Chapter 4

Model Derivation and Methodology

4.1 Evaluation of a Delay-Feedback Reservoir

A common approach to implementing Reservoir Computing within physical hardware is to apply delayed feedback to a system in an attempt to realise a Delay-Feedback Reservoir Computing (DFRC) system. As discussed in section 2.4, a DFRC is a promising choice for hardware implementation as it only requires a single non-linear neuron, also called "non-linear node", and a delay to create the reservoir layer. This allows for a much smaller reservoir layer than alternative reservoir computing paradigms as the inputs are time-multiplexed and only a single non-linear node is required, saving space in I/O and in silicon respectively. The downside to using a DFRC system is that it requires a complex input layer; as it must time-multiplex an input signal with a faster masking signal, shown in figure 2.13. However, as the process of time-multiplexing is a well known and understood process, this input layer can be implemented with simple, off the shelf components. Given that DFRC systems are ideal for hardware implementation and have shown to provide excellent performance in solving temporal computational tasks, why are they not used more often? The problem lies with the lack of a general design methodology.

Currently when a DFRC system is built, an interesting substrate is created and then the DFRC framework is applied to it in an ad-hoc fashion; leading to a top-down design approach [109]. This top-down approach often leads to an inefficient use of the substrate, as there are likely unused dynamics within the substrate that are not exploited, and an inefficient computing paradigm, as the dynamics of the substrate have not been tuned so that they match the dynamics required to solve a particular computational task. To create a DFRC system with the highest efficiency and performance, a bottom-up design approach must be used; however, this is not currently generally possible due to the absence of a common design methodology.

Within the literature surrounding DFRC, there is no indication on what a substrate must provide to make a good DFRC; this is largely due to a lack of a complete generic DFRC model which describes the fundamentals of what a DFRC system requires to

operate. Currently, the literature describes a DFRC system needing an input masking layer, a delay, a non-linear node, and an output layer [8]; this is typically shown as a block diagram within figure 4.1.

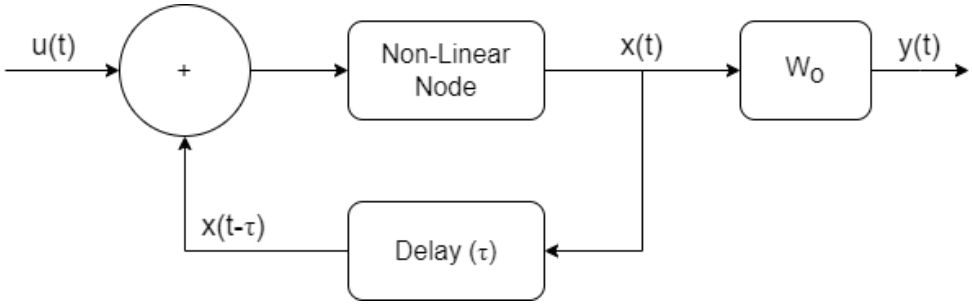

Figure 4.1 A block diagram showing the structure of a DFRC in terms of the current literature. Where $u(t)$ is the pre-masked input signal, $x(t)$ is the reservoir state vector, $x(t-\tau)$ is the delayed reservoir state vector, $y(t)$ is the weighed output.

The input masking layer is well described, consisting of a time-multiplexed signal combining the input and masking signals. The reservoir layer contains a non-linear node, a delay of time length τ, and a feedback loop. The non-linear node is described as a neuron that must exhibit the following dynamical behaviours; non-linearity, fading memory, and high dimensionality.

As discussed in section 2.3.3, almost all physical substrates exhibit these behaviours, with many types of these physical substrates being used as the non-linear node within a DFRC system. Therefore in order to obtain a better understanding of the functionality of the non-linear node in generic terms, several existing physical implementations of DFRC systems must be analysed.

4.1.1 Analysis of Current Physical Implementations

There are several physical implementations of DFRC architectures, most of which are very different in nature, with the most common types being; photonic, digital, analogue, or a hybrid of these types [110–113]. Although each of these types of physical implementations utilise a different computing paradigm to realise a DFRC

system, they all follow the same structure described in figure 4.1. In order to further understand the purpose and functionality of the non-linear node, three of the most popular and successful DFRC systems, in terms of cutting-edge performance and hardware translation feasibility, are analysed. In order to see if a generalised model can be obtained, the chosen physical implementations utilise different physical concepts to realise a DFRC; these are optolectronic, a digital photonic hybrid, and an analogue. The first DFRC system to be investigated is an optoelectronic implementation proposed by Larger, et al [7].

Optoelectronic Reservoir Layer

By far, the most common type of DFRC architecture uses photonic and optoelectronic techniques and devices. This is mainly twofold; the processing speeds at which photonic systems can operate at are highly attractive for processing real-time data, and they can be built using off the shelf telecommunication components that have been highly optimised by industry and are readily available. These advantages have led to a number of high-performance optoelectronic DFRC systems to be realised at a relatively low cost.

A particular architecture of interest is an optoelectronic implementation demonstrated by Larger, et al, which has been shown to have excellent performance in solving time series prediction tasks, while being based on a simple and efficient delay-coupled photonic system [7]. A block diagram depicting the architecture of the reservoir layer is shown in figure 4.2.

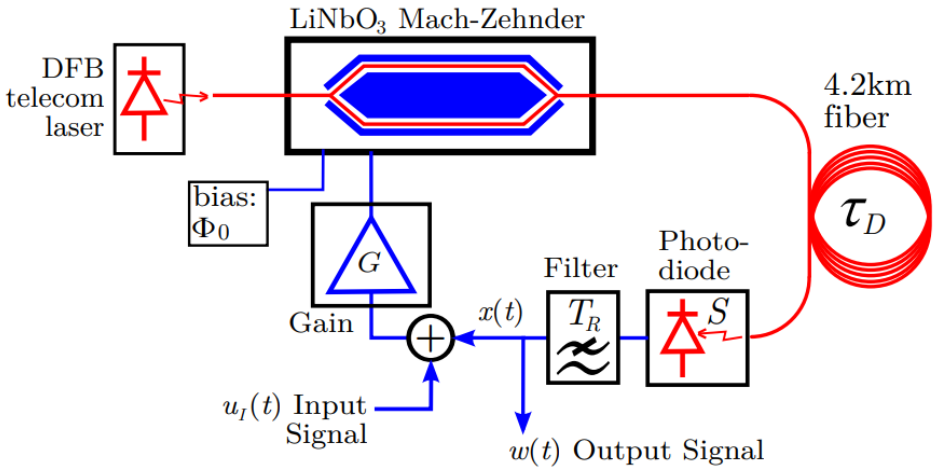

Figure 4.2 A block diagram of the reservoir layer of an optoelectronic delay-feedback reservoir. The red lines indicate the photonic section of the reservoir, consisting of a laser source that is modulated using a MZM, which is then delayed via a long length of optical fibre, then converted into an electronic signal by a photodiode. The blue lines indicate the electronic section realises a low-pass filter and a feedback loop, where the delayed output and pre-masked input are combined. [7]

The block diagram shown in figure 4.2 indicates that this architecture has two sections; a photonic and electronic section. The photonic section, indicated with a red line, carries information in the form of a modulated telecommunication laser source. This laser source is modulated using a Mach-Zehnder modulator (MZM), an interferometric device with a strong electro-optic effect that produces phase modulation by applying an electric field [114], which provides a non-linear transform of sin^2 to the modulating signal; which is a summation of the pre-masked input signal and delayed filtered output of the MZM. A long length of optical fibre is then used to implement the time delay of τ, then the output is turned into an electronic signal using a photodiode.

The electronic section, indicated with blue lines, implements the feedback loop within the delay-feedback reservoir. The feedback is implemented by a summation block, where the pre-masked input signal and delayed output are combined to produce the modulation signal for the photonic section. A gain is then applied to the modulation signal so that the signal operates within the optimal MZM non-linear range. A bias

can also be added to the MZM electronically via Φ_0 to change the operating behaviour of the non-linearity. The final part of the electronic section is the filter, which is stated to be implemented as a low pass filter used to set the timescale of the system to T_R. Although not explicitly stated within the paper, the low pass filter also mathematically performs a leaky integration in order to compute $\dot{x}(t)$ to $x(t)$.

There are many similarities between the generic model shown in figure 4.1 and the optoelectronic architecture from figure 4.2, both sharing a summation node and a signal delay. As an initial hypothesis, it would appear that the non-linear node contains a device which can non-linearly transform a signal, and a filter which is able to integrate it. In order to check if this estimation is general for all types of DFRC systems, a digital photonic hybrid DFRC system proposed by Kumar [9] is investigated.

Digital Photonic Hybrid Reservoir Layer

While optoelectronic systems have been shown to provide excellent performance in solving real-time tasks, the analogue electronic section and the output layer can require complex analogue circuitry when running at photonic speeds to create a self contained system. In an attempt to create an all-in-one solution, the photonic layer can be combined with a high-speed FPGA to create a full three-layer DFRC system; this type of architecture is called a digital photonic hybrid DFRC system. A block diagram depicting a digital photonic hybrid reservoir layer is shown in figure 4.3.

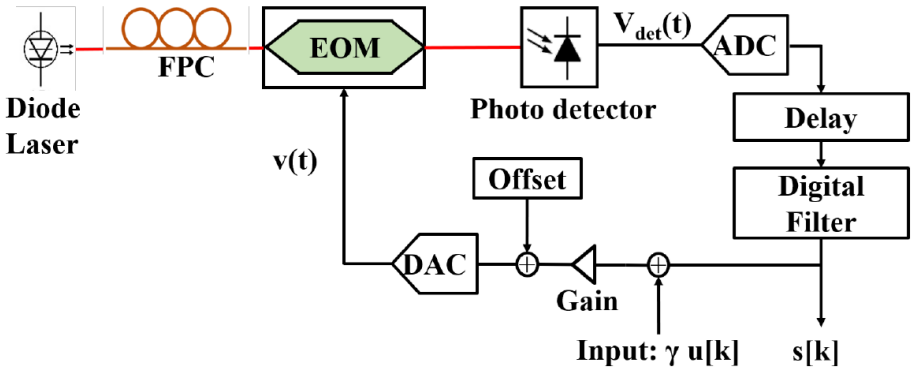

Figure 4.3 A block diagram of the reservoir layer of a digital photonic hybrid delay-feedback reservoir, implemented using optoelectronic hardware and an FPGA. The photonic section of the reservoir is indicated with red lines, which consists of a diode laser source, a fibre polarisation controller (FPC) to control the polarisation of the laser beam, and an electro-optic modulator (EOM) to provide a non-linear transform of sin^2. [9]

Similar to the previously discussed optoelectronic architecture, the block diagram of the digital photonic hybrid DFRC system, shown in figure 4.3, contains a photonic and electronic section.

This architecture has a relatively simple photonic section, as indicated with red lines, containing only a few components. The diode laser generates a laser beam which is passed through a fibre polarisation controller (FPC) to alter the polarisation of the laser for modulation. The laser is then modulated by an electro-optic modulator (EOM), which is functionally identical to the MZM discussed previously, while also providing a non-linear transform of sin^2 to the modulating signal. A photo detector is then used to convert the intensity of the light into an electronic signal, which is then converted from analogue to digital for processing within an FPGA.

The electronic section of this architecture is performed within an FPGA, therefore information must be converted between the analogue and digital domains in order to process the modulated laser signal, and to generate the modulation signal for the EOM. The feedback loop is similar to that of the optoelectronic architecture shown in figure 4.2, with the exception that the delay is performed digitally rather than optically

which is realised using block memory as a shift register. The pre-masked input signal is multiplied by a scaling factor of δ, then summed together with the delayed output to generate a modulation signal for the EOM. A gain factor is then applied to the signal so that the EOM operates within the correct non-linear regime; an additional DC offset can also be applied to the modulation signal if required through a separate summation block. The digital filter implemented is a finite impulse filter (FIR), which takes advantage of the block memory of delays to generate the state information of the DFRC system, whilst also setting the timescale of the system.

In structure, there is very little difference between the optoelectronic and digital photonic hybrid DFRC architecture. In order to realise the non-linear node, both used photonic devices to provide a non-linear transform, while the optoelectronic architecture had a simple resistor-capacitor (RC) low-pass filter and the digital photonic hybrid architecture used a FIR filter. Careful consideration must be given when using digital filters as they add additional latency to the system, which needs to be accounted for within the overall delay of the feedback loop, τ; this will only be a consideration within analogue filters if the group delay (the average time delay introduced by the filter) is within an order of magnitude of the masking period, θ. Although digital filters are slower, they have the advantage of being able to natively collect the state space of the reservoir in the filter coefficients. Another important factor to consider is that the transient responses of digital and analogue filters are also different; consideration needs to be given to ensure the correct transient response is generated to appropriately perturb the substrate. The choice of filter implementation is likely due to the nature of the computing substrate; an RC filter is easy to implement in analogue, while a FIR filter is easy in digital. The initial hypothesis that the fundamental building blocks of a non-linear node consist of two parts, a device which can non-linearly transform a signal and a filter which is able to integrate, appears to hold true for both optoelectronic and digital photonic hybrid type DFRC systems. To confirm further whether this hypothesis can be fully generalised, an analogue reservoir layer proposed by Soriano [10] is investigated.

Analogue Reservoir Layer

While previously discussed architectures have utilised a photonic section to achieve fast processing speeds, they suffer from interfacing problems with electronic systems. The optoelectronic architecture requires high speed analogue components and a complex design, while the digital photonic hybrid architecture requires a high speed FPGA and conversions between the analogue and digital domains making it a discrete-time system. An alternative approach is to have an all electronic reservoir layer that is entirely analogue. This allows for a continuous signal within the feedback loop with reduced hardware complexity, as there are no conversions between digital and analogue. An example of an all analogue reservoir layer is shown as a block diagram in figure 4.4.

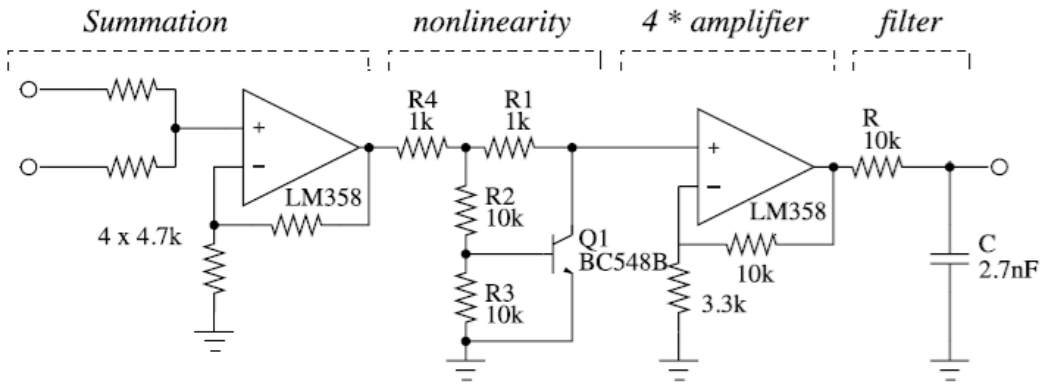

Figure 4.4 A circuit diagram depicting the four key stages within an analogue implementation of the reservoir layer within a delay-feedback reservoir without delayed feedback applied. The summation stage mixes a pre-masked input with the delayed feedback, the non-linearity stage applies a non-linear transform to the signal, the amplifier stage amplifies the output of non-linearly to account for attenuation, and finally the filter stage which integrates the signal. [10]

The circuit shown in figure 4.4 shows the core components of the reservoir layer before delayed feedback is applied. Delayed feedback can be applied by adding an inductor-capacitor (LC) ladder circuit, between the output and input to create an analogue delay [115].

The architecture consists of four main blocks; summation, non-linearity, amplification, and filtering. The summation block consists of a non-inverting summing amplifier, which sums the two input signals together. Although configured to provide a gain of two, the ratio of the resistors can be modified to apply an additional gain to the summed signal if required. The non-linearity section is realised by biasing a bipolar junction transistor (BJT) to operate as a non-linear amplifier. The behaviour of the non-linearity can be modified by changing the bias point of the BJT, which is done by changing the values of the resistors $R_1 - R_4$. The resistor values for $R_1 - R_4$, as shown in figure 4.4, are chosen to represent a Mackey-Glass non-linearity with an exponent of 6 [10]. The amplification stage applies a gain factor of four to the signal. This is done to account for any losses from the non-linearity stage and to scale the signal so that it is within the minimum and maximum ranges of the components. The final stage is filtering, which is realised by a RC low-pass filter. This filter sets the timescale of the system and performs a leaky integration, which is the same as the previous architectures.

Although different hardware was used within the analogue architecture, it has the same functional structure as the previously discussed architectures; a device to provide a non-linear transform, and a filter to integrate. As all three of the discussed architectures require these components to realise a physical non-linear node, it would suggest that the initial hypothesis of *the non-linear node consists of a device to provide a non-linear transform and a filter* is to be expected.

4.1.2 A Generic Computational Model

The first section of this chapter aims to investigate existing physical implementations of DFRC systems in order to determine the requirements for a general model. While some of the model components are apparent from the literature, i.e. the positive feedback loop and time delay, there is no definitive answer as to what the minimum requirements are to create a non-linear node. By investigating different DFRC archi-

tectures, each of which use different computing paradigms, it is discovered that the minimum requirements for a non-linear node are a non-linear transform and a filter to integrate the information. Hence, a general model for all DFRC systems can now be determined in terms of their functional blocks; this is shown in figure 4.5.

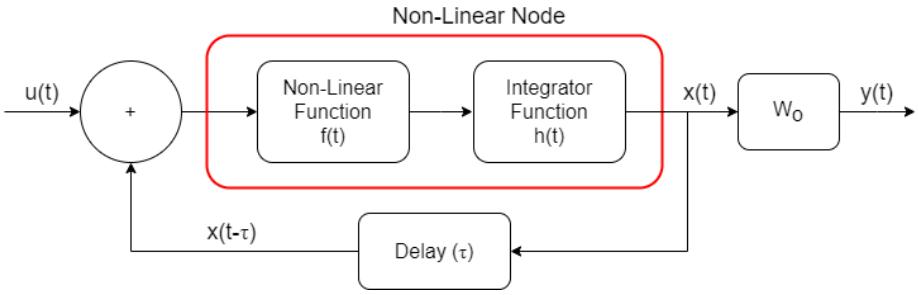

Figure 4.5 A block diagram showing the general structure of a DFRC system in terms of its functional blocks; $u(t)$ is the pre-masked input signal, $x(t)$ is the reservoir state vector, $x(t-\tau)$ is the delayed reservoir state vector, the function $f(t)$ is a non-linear function, the function $h(t)$ is a integration function, and $y(t)$ is the weighted output.

Where $u(t)$ is the pre-masked input signal, $x(t)$ is the reservoir state vector, $x(t-\tau)$ is the delayed reservoir state vector, the function $f(t)$ is a non-linear function, the function $h(t)$ is a integration function, and $y(t)$ is the weighted output.

From the generalised model shown in figure 4.5, the mathematical equation for the model can be expressed as:

$$\begin{aligned} x(t) &= h(f(u(t) + x(t-\tau))) \\ y(t) &= x(t)W_o \end{aligned} \tag{4.1}$$

While it is useful to have a generalised model with a generic integration function, it was found during the analysis of several physical architectures of DFRC systems that a first-order RC filter was typically used. As the focus of this work is on realistic and feasible hardware architectures, a typical model for a DFRC implementation using a first-order RC integrator can be created; this is shown in figure 4.6.

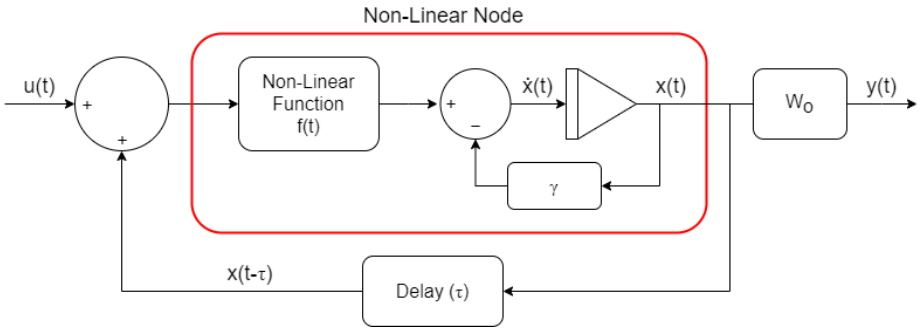

Figure 4.6 A block diagram showing a typical structure of a DFRC system with a first-order RC filter. Where $u(t)$ is the pre-masked input signal, $x(t)$ is the reservoir state vector, $\dot{x}(t)$ is the derivative of the reservoir state vector, $x(t-\tau)$ is the delayed reservoir state vector, γ is the feedback strength for the leaky integrator, the function $f(t)$ is a non-linear function, and $y(t)$ is the weighted output.

To better express the RC circuit in the time domain, a leaky integrator model is used which is mathematically equivalent. The strength of the leakiness is controlled by a feedback gain of γ, where $\gamma < 1$, which also sets the timescale of the system. The relationship between the leak strength and timescale is given by:

$$T = \frac{1}{\gamma} \tag{4.2}$$

From the first-order filter model shown in figure 4.6, the mathematical equation for the model can be expressed as:

$$\begin{aligned} \dot{x}(t) &= f(u(t) + x(t-\tau)) - \gamma x(t) \\ y(t) &= x(t) W_o \end{aligned} \tag{4.3}$$

Where $u(t)$ is the pre-masked input signal, the function $f(t)$ is a non-linear function, γ is the feedback strength for the leaky integrator and is typically less than one, $x(t)$ is the reservoir state vector, $\dot{x}(t)$ is the derivative of the reservoir state vector, $x(t-\tau)$ is the delayed reservoir state vector, $y(t)$ is the weighted output, and W_o is the trained output weight matrix.

4.2 Experimental Model

In order to test whether the non-linear node can be optimised for a particular computational task, a hardware accurate, as in represents a realistic and feasible hardware implementation, experimental model of a DFRC system must be created. In order to create the realistic hardware model, the typical structure of a DFRC system with a first-order filter (figure 4.6), which was derived in section 4.1.2, is used as a template. To complete this template, an appropriate non-linear function must be chosen. The Mackey-Glass dynamical system is chosen as it was used within the original DFRC system developed by Appeltant [116], as it has been shown in many cases to provide rich dynamics [117–119], and it can be easily realised in hardware [8, 10].

4.2.1 Mackey-Glass Dynamical System

The Mackey-Glass model is a first-order delay-differential equation created by Mackey and Leon Glass to model the respiratory system and hematopoietic diseases, which are biological functions that are subject to a time delay [120]. The original Mackey-Glass equation is given as:

$$\dot{P}(t) = \frac{\beta\ \theta^n P(t-\tau)}{\theta^n + P(t-\tau)^n} - \gamma P(t) \tag{4.4}$$

Where $P(t)$ is the homogeneous population density of mature red blood cells, $\dot{P}(t)$ is the derivative of the homogeneous population density, τ represents the time delay in blood cell production, γ is the death rate of blood cells, and β, θ, and n are parameters which define the blood cell production rate of the system.

Although initially used to model biological processes, the Mackey-Glass system has shown to exhibit a variety of rich dynamical behaviours, such as strong non-linearity, and high dimensionality; making it ideal for a wide range of applications. The Mackey-Glass model was first applied to a DFRC system by Appeltant within their original

DFRC work [116]. The original Mackey-Glass equation was slightly modified to make it simpler, so that it could be realised into hardware. The modified Mackey-Glass equation is given as:

$$\dot{x}(t) = \frac{\beta\Big(x(t-\tau) + \delta I_t\Big)}{1 + \Big((x(t-\tau) + \delta I_t\Big)^n} - \gamma x(t) \tag{4.5}$$

Where $I(t)$ is the pre-masked input signal, $x(t)$ is the reservoir state vector, $\dot{x}(t)$ is the derivative of the reservoir state vector, $x(t-\tau)$ is the delayed reservoir state vector, β is the coupling gain between the non-linear function and integrator, γ is the feedback strength for the leaky integrator, δ is a input scaling factor, and n is the non-linearity factor.

The Mackey-Glass time delay-differential equation can be separated into two parts, the Mackey-Glass non-linear function and a leaky integrator. This can be expressed as:

$$x(t) = \frac{\beta\Big(x(t-\tau) + \delta I_t\Big)}{1 + \Big((x(t-\tau) + \delta I_t\Big)^n} \tag{4.6}$$
$$\dot{x}(t) = -\gamma x(t)$$

This equation can also be expressed as a block diagram using the typical hardware structure of a DFRC system, as shown in figure 4.6, as a template. This is shown in figure 4.7, where f_t represents $x(t-\tau) + \delta I_t$.

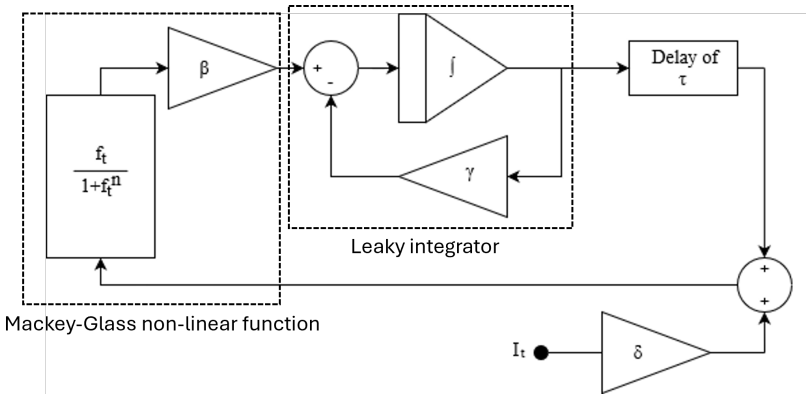

Figure 4.7 A block diagram of the DFRC hardware architecture implementing the Mackey-Glass non-linear time delay-differential equation separated into its subsequent parts. Where I_t is the pre-masked input signal, τ is the amount of time the signal is delayed for, β is the coupling gain between the non-linear function and integrator, γ is the feedback strength for the leaky integrator, δ is a input scaling factor, n is the non-linearity factor, and f_t represents $x(t-\tau)+\delta I_t$.

The Mackey-Glass DFRC hardware architecture shown in figure 4.7 illustrates the separation of the functional blocks within the non-linear node. This separation allows for testing both the non-linear function and integrator independently, so that their effect on system performance, for a particular computational task, can be individually determined.

4.2.2 Integrator

In order to allow for direct control over the timescale of the system, the current leaky integrator is expressed as a Laplace transfer function so that any filter can be used within the model. To convert the leaky integrator into the Laplace domain, the integrator is treated as a simple feedback loop with a feedback gain of γ. Closing the loop gives the equation:

$$H(s) = \frac{1}{s+\gamma} \tag{4.7}$$

As it is desired to set the timescale of the DFRC system directly using a single parameter, the equation 4.7 is rearranged to give:

$$H(s) = \frac{1}{Timescale \cdot s + 1} \tag{4.8}$$

Where $Timescale$ is the time constant of the system in seconds.

This modification also allows the model to accept any integrator that is expressible within the Laplace domain, generalising the model so that more complex filters can be used, allowing for further experiments to be performed if needed.

4.2.3 Simulation Model

With the non-linear function and integrator stages chosen, the DFRC model is constructed within Simulink, a MathWorks simulation tool (version 23.2 within MATLAB 2023b) [121]. Simulink allows for a near physical system to be modelled and executed with real-time transient analysis. Within MATLAB, the input and output layers of the DFRC system are implemented in software. The input data is generated within the MATLAB workspace and injected into the Simulink model, from where the output of the DFRC system is returned into the MATLAB workspace for reservoir training and evaluation. The Simulink model is shown in figure 4.8.

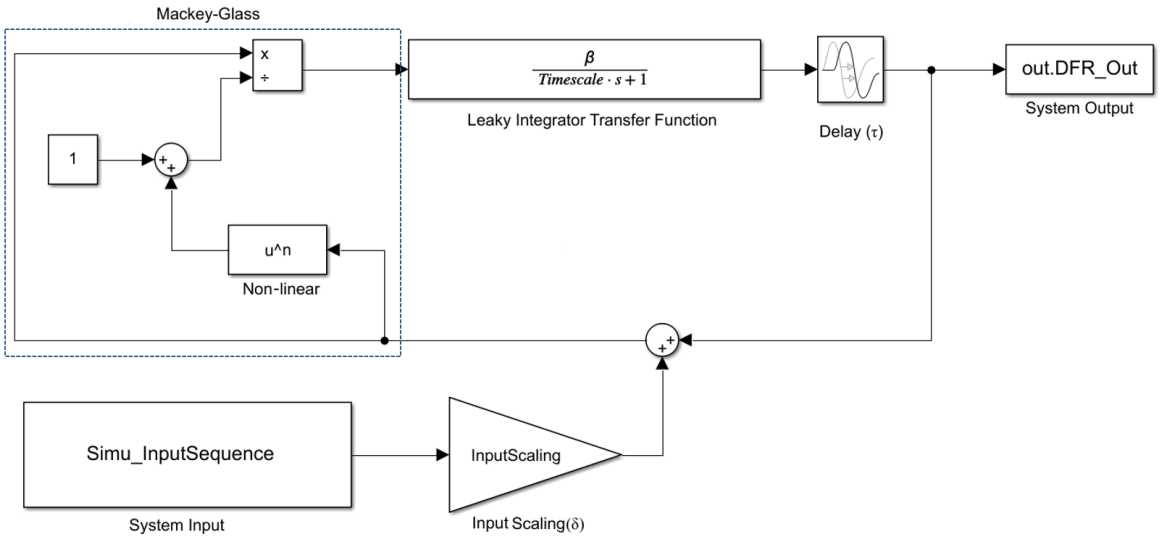

Figure 4.8 A schematic of a Delay-Feedback Reservoir, using the Mackey-Glass non-linear function as the general non-linear function, and a first-order transfer function as the integration stage, created within Simulink 23.2.

The Mackey-Glass non-linear function is built from Simulink numeric model blocks, as shown within the dotted rectangle within figure 4.8. The non-linearity factor can be modified by changing the n parameter within the "Non-linear" block. The integration stage is realised by combining the coupling gain, which was between the non-linear function and integrator, and the first-order transfer function given in equation 4.8. The delay is implemented using the Simulink "Transport Delay" block, which adds a signal delay of time length τ.

The input signal, generated within MATLAB, is injected into the reservoir using a "From Workspace" block, named "`Simu_InputSequence`". The states of the reservoir are read from the model using a "To Workspace" block, named "`out.DFR_Out`", at a sample rate of θ; this allows the states of each virtual node to be sampled at the correct time. These states are then returned into the MATLAB workspace, where the training and system evaluation take place.

Table 4.1 contains a list of the parameters and their functions within the Simulink model.

Name	Symbol	Function
Timescale	*Timescale*	Sets the timescale of the system
Coupling Gain	β	Gain between the non-linear function and integrator
Input Scaling Factor	δ	Gain factor applied to new input signal
Mackey-Glass Exponent	n	Sets the strength of the non-linearity within the non-linear function
Time Delay	τ	Sets the time delay within the feedback loop

Table 4.1 A list of model parameters which are available within the Mackey-Glass Delay-Feedback Reservoir Simulink model.

4.3 Common Methodology

The generalised and parameterisable model from section 4.2 is used throughout this work to carry out several different experiments to investigate the functions of the internal parameters, and their effect on the performance and metrics, within a DFRC system.

Although each experiment will be different, some methodologies are common amongst all experiments. As all experiments within this work are to be carried out by modifying the reservoir layer of the DFRC system, the common methodology is centred around the setup of the input and output layers of the DFRC system which are implemented in software within MATLAB: the same applies to the number of input samples and the masking, training, and testing procedures. These are kept constant so that variations in the input and output layers do not affect the results.

4.3.1 Input Layer

Masking Procedure

In order to thoroughly test the effect that each parameter within the DFRC model has on performance, two differently sized reservoirs are used. A DFRC system with a small number of 20-nodes is used to see how the internal parameters of the non-linear node affect a system which has limited dynamics. Conversely, a DFRC system with a large number of 200-nodes is used to emulate a system which can reach greater dynamics [105]. A random-weighted no-offset masking signal is generated once, for both the 20- and 200-node systems, and is used for masking all input data; figure 4.9 shows the masking signal generated for the 20-node system, while figure 4.10 shows the masking signal for the 200-node system.

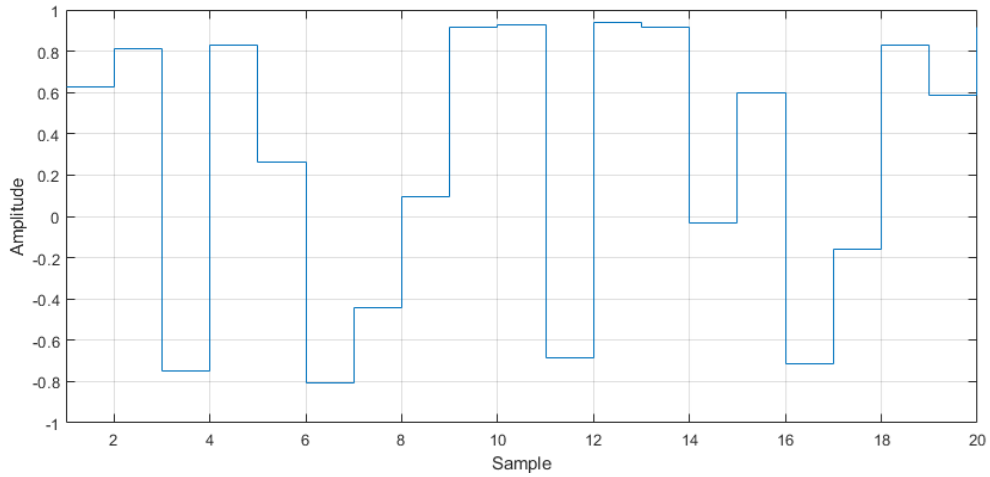

Figure 4.9 A random-weighted no-offset masking signal used to create a DFRC system with 20 virtual nodes.

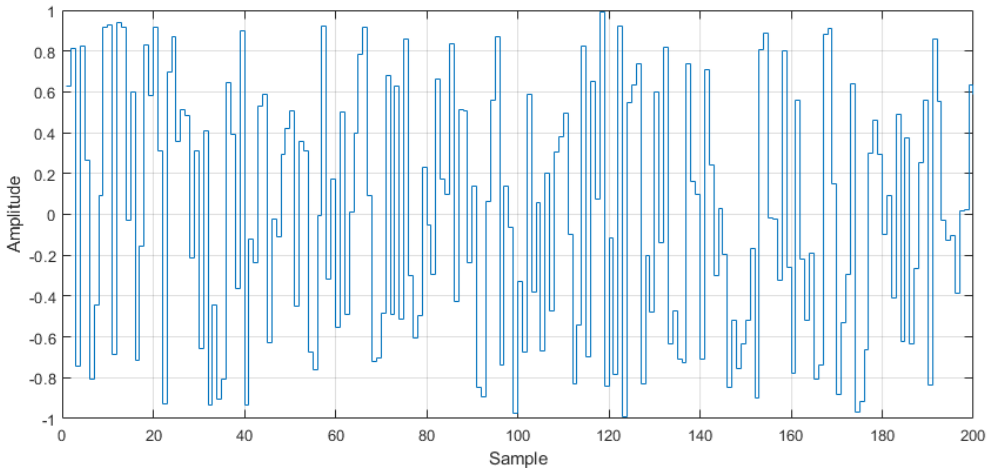

Figure 4.10 A random-weighted no-offset masking signal used to create a DFRC system with 200 virtual nodes.

It should be noted that the time delay around the feedback loop is set to $\tau = 80$ s; giving a θ for the 20- and 200-node systems of 4 s and 0.4 s respectively.

Generating Input Data

In order to evaluate the performance of a DFRC system as the systems dynamics change, two benchmarks are used; NARMA-10 and Santa Fe Laser. It is possible to evaluate the performance of a DFRC system using only these two benchmarks as they require opposite dynamics within a reservoir to achieve the best performance. The NARMA-10 benchmark has a large requirement of memory capacity, as it requires remembering n previous input steps in order to recreate the input-output dynamics, but requires little non-linearity. While the Santa Fe Laser benchmark is the opposite, it requires a reservoir with a large amount of non-linearity to predict its next time step, but requires little memory of its previous inputs (see section 3.2 for further details). By analysing the results of these benchmarks, it is easy to see which dynamics are present and which are missing.

The procedure for generating the input sequences of both the NARMA-n and Santa Fe benchmarks within MATLAB are the same. A sequence of 6000 data points, either generated from the NARMA-n equation or copied from the Santa Fe Laser

dataset, is time-multiplexed by the corresponding random-weighted no-offset, mask discussed in section 4.3.1, producing a sequence with amplitude range of between 1 and -1. To remove any influence from the initial zero state of the DFRC system, a washout period consisting of 100 data points is used; with the training and test datasets being split 80/20 respectively.

4.3.2 Output Layer

Training Strategy

One of the main reasons reservoir computing has become so desired as a computing paradigm is that the training process is exclusive to the output layer. Without having to train multiple weight matrices within the reservoir model, training typically becomes a simple case of linearly reading the outputs of a reservoir system to train a single output weight matrix. Linear combination, which is typically used to obtain the states within a DFRC system, can be expressed as:

$$\hat{y}_k = \sum_{i=1}^{N} w_i . x\Big(k\tau - \frac{\tau}{N}(N-i)\Big) \tag{4.9}$$

Where w_i is the weight attributed to each virtual node i, x is the output of the non-linear neuron at a given state, and $\hat{y}$ is the approximated output signal.

This produces a reservoir state vector of X, which has the dimensions of $[N \times L]$; where N is the number of virtual nodes within the DFRC system and L is the number of post-masked data inputs. The reservoir state vector is then used with the desired target vector, Y_{target}, to determine the output weight matrix, W_{out}, by solving the following overdetermined system:

$$Y_{target} = W_{out} X \tag{4.10}$$

In order to calculate the output weight matrix, W_{out}, several training techniques can be applied. These can be broadly separated into two training groups: on-line, and off-line; these are discussed in within section 3.1. For this work, the Moore-Penrose pseudo-inverse function is used as it has an efficient implementation within MATLAB; and as much of this work contains large parameters sweeps, the Moore-Penrose pseudo-inverse function will save on computational resources when compared to alternative training methods such as ridge regression. The equation for training a weight matrix with the Moore-Penrose pseudo-inverse function is given as:

$$W_{out} = Y_{target} X^{+} \tag{4.11}$$

Where + represents the use of the pseudo-inverse function.

Although using the Moore-Penrose pseudo-inverse function can produce a overdetermined solution for large or unstable systems, the DFRC systems used within this work only contain 20- and 200-nodes, which is not large enough to cause problems. Another factor to consider is that as there are a lot of big parameter sweeps within this work, it is likely that the system will become unstable within the parameter space of the DFRC system. A training algorithm like ridge regression may force some solutions on the boundaries of stability, but in reality these solutions will almost never generalise as variations within the random mask and input data will cause the system to become unstable. Using a training algorithm such as the Moore-Penrose pseudo-inverse function gives large variations within the training and testing errors, which gives insight into the boundaries of stability for a DFRC system.

Performance Evaluation

To compare the performance of a DFRC system between different experiment runs, a measurable quantity of how well the DFRC system has performed must be defined. Within this work, the performance is evaluated using normalised root mean square error (NRMSE), which represents the error between the trained DFRC system output and its desired target, which is normalised against the square of the standard

deviation (see section 3.1.3 for further details). The lower the value of the NRMSE, the better the DFRC system is at the executed computational task; with a value of zero implying a perfect match between the trained DFRC system and its target, and a value of one indicating that the DFRC system output is no better than the mean value of the target output.

4.3.3 Experimental Expectations

Within this work, several experiments will be run to evaluate the performance and system metrics of a delay-feedback reservoir while modifying the parameters of the non-linear node. However, in order to compare experimental results, a definition of what an "improvement" is needs to be established in terms of performance and system dynamics.

The performance of a delay-feedback reservoir is to be measured by NRMSE and the system metrics, as defined within section 3.3, used within this work are: Kernel Quality (KQ), Generalisation Rank (GR), and Linear Memory Capacity (LMC); the Computational Ability (CA) is also referenced, which is the difference between KQ and GR.

An ideal system has the following system metrics: high KQ, CA, LMC, and a low GR, with a system performance of a low NRMSE value. An ideal system indicates that the system is able to map inputs into high-dimensional space (high KQ), retain memory of its previous inputs (high LMC), generalise noisy inputs (low GR), and is able to recreate a target output signal from the output of the system (low NRMSE). Therefore, if a system has improved, it is implied that the system has become closer to the ideal system.

It should be noted that achieving a near state-of-the-art NRMSE performance of a delay-feedback reservoir is not a desired metric within this work, the focus is to analyse and understand the overall effect that changing the internal parameters within the

delay-feedback reservoir model has on its performance. Therefore, the same benchmark and masking sequences are used to directly compare results, which may have been generated from a poor seed. Any experiments where the NRMSE performance of the reservoir is an important factor, a statistical analysis will be performed using a randomly seeded benchmark and masking sequences, as performed within section 5.5.3.

From chapter 1, the following specific experimental objectives were defined:

Objective 2: Investigate the effect that the integrator timescale within a delay-feedback reservoir has on system performance and dynamics.

Objective 3: Investigate the effect that the non-linear function within a delay-feedback reservoir has on system performance and dynamics.

For these objectives, an increase in performance is to be expected as the timescale and non-linear functions are tuned. The system dynamics are also likely to change, which will lead to different system metrics. Given that the systems are to be evaluated using the NARMA-10 and Santa Fe benchmark, as discussed within section 3.2, the NARMA-10 evaluated system will have an improvement when the system is tuned to have a higher LMC, whereas Santa Fe will have an improvement when system is tuned to have a higher KQ and lower GR. It is likely that these are mutually exclusive, with the system being tunable for either higher memory or higher non-linearity.

It should be noted that the mutual exclusivity between memory and non-linearity can be often lessened in practise through external modifications to the reservoir system, such as additional post-processing at the readout [122], to increase the memory of the system. However, external modification to the delay-feedback reservoir system is out of the scope of this work.

4.4 Summary

This chapter investigates several different methods of implementing a DFRC system within hardware, proposes a generalised model for physical DFRC systems, creates a hardware accurate simulation model derived from this generalised model, and discusses some of the common methodology applied across all experiments within chapters 5, 6, and 7.

Within section 4.1, three of the most promising physical DFRC architectures from the current literature are analysed in order to determine the fundamental building blocks required for a DFRC system to function.

It was found that in all designs, the non-linear node within a DFRC system must at least include a non-linear function and a filter; which was later identified as an integrator.

These building blocks were put together to create a generalised model of a DFRC system, which can be used with any non-linear or integration function (shown in figure 4.5). A typical hardware model is developed, where a leaky integrator is used as the integration function, and a general non-linear function can be used (shown in figure 4.6) for the non-linear function.

Section 4.2 utilises the typical hardware model shown in figure 4.6 as a template to create a hardware accurate simulation model for a physical DFRC system within Simulink. The Mackey-Glass dynamical system was chosen as the non-linear function as it has been shown to exhibit rich dynamical properties, making it ideal for use within a DFRC substrate. The design was modified slightly, with the leaky integrator replaced by a Laplace transfer function equivalent, so that the model could accept any integrator expressible within the Laplace domain. This allows direct control over the timescale of the system, generalises the model to accept any integration function, and allows for a further range of experiments to be performed if needed. The final Simulink model is shown in figure 4.7, with the key parameters listed in table 4.1.

Finally, in section 4.3, the common methodologies that are shared throughout all of the experiments within this work are discussed; with the focus being on the operation of the input and output layers of the DFRC system, which are realised in software.

Chapter 5

Integrator Timescale

5.1 Manipulation of Timescales

All physical systems have a natural timescale that determines how long the transient activity within the system lasts when an input stimulus is applied. However, in many cases, the natural timescale of a substrate is much too small for either computation to be performed or practically harnessed; for example, in magnetic systems where the substrate reaches a steady state faster than most electronic equipment can detect [123]. Therefore, it is desired to configure a system to operate at different timescales. This can be achieved in a discrete system by modifying the time step of the system; this increases the fading memory window, allowing for more of the input to be retained between time steps. Alternatively, if the system has a defined integration stage, then the time constant of the integrator can be changed. If the integration stage is not well defined, most analogue systems can have their timescale adjusted by adding an integrator or filter to the architecture.

5.1.1 Timescales within Delay-Feedback Reservoir Computing

The timescale within the delay-feedback reservoir is particularly important as not only does it define the speed at which information can be processed, but the topology the virtual nodes will take. The purpose of the masking signal is to perturb the non-linear node in a way so that it remains within its transient stage [124]. If the timescale of the non-linear nodes, T, is much smaller than the time period of the masking signal ,θ, then the transient response will happen so quickly relative to the period of the masking signal, that there will be no bleed-over of the transients to their neighbouring virtual nodes; this creates a network topology where the virtual nodes are not connected. Alternatively, if the non-linear node has a timescale much greater than the time period of the masking signal, then the transient response will persist for multiple masking signal periods; creating a virtual node topology where each virtual

node is connected to several other virtual nodes, and not just its neighbour. This effect is shown in figure 5.1.

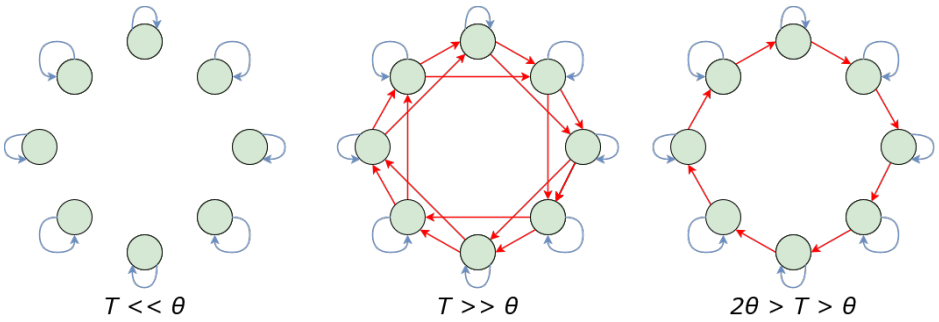

Figure 5.1 The effect of the topology of virtual nodes within a delay-feedback reservoir when the timescale of the non-linear nodes, T, and the period of the masking signal ,θ, changes. The blue arrows show the connections due to the delay-line within the system, and the red arrows show the connections due to the timescale the non-linear node.

As the timescale of the non-linear node increases, more information is passed on to the next virtual node, thus creating a stronger connection between virtual nodes. Therefore, a new parameter of ρ is introduced due to the effect of the timescale being so prominent, allowing for an indication of how strong the connection is between each virtual node. It is defined as:

$$\rho = \frac{T}{\theta} \tag{5.1}$$

The effect of the timescale and masking period of a first-order integration stage, with its corresponding ρ value, can be seen within figure 5.2.

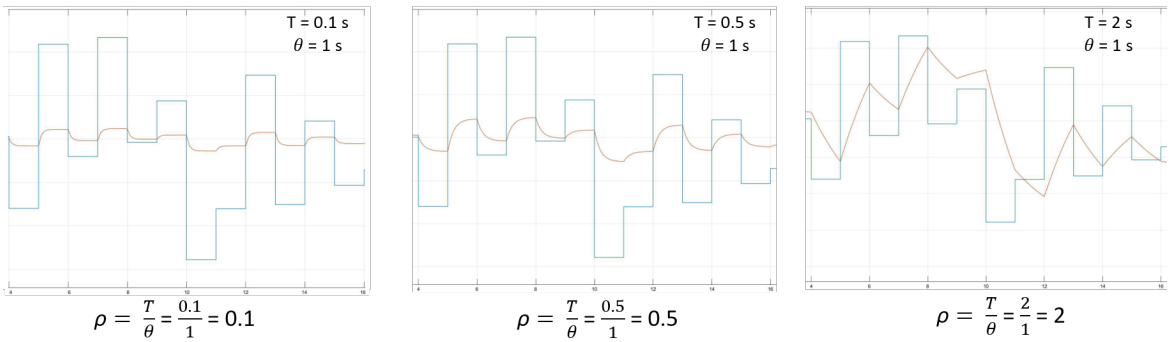

Figure 5.2 The effect of the timescale and masking period of a first-order integration stage within a delay-feedback reservoir for three timescales, 0.1 s, 0.5 s, and 2 s, with their calculated ρ values. The blue line represents the masked input signal, while the orange line represents the response of a first-order integration stage.

Given that the timescale of a delay-feedback reservoir system fundamentally changes the behaviour of the reservoir, the effect of the relationship between the timescale and the masking period, θ, is to be explored. This chapter aims to investigate how the timescale within a hardware accurate delay-feedback simulation model affects both the computational performance and dynamics of the reservoir system.

5.2 Experimental Methodology

The effect that the integrator timescale has on system dynamics and computational performance can now be investigated using the experimental model from chapter 4. Two reservoir systems will be tested (a 20-node and a 200-node system with a τ of 80 s) at four different timescales (10ms, 100ms, 200ms, and 400ms). This allows a comparison between a smaller and larger virtual node system, while allowing for larger values of ρ. It was decided that, in order to keep the processing speed of each reservoir the same, the time delay, τ, will remain constant while the timescale of the first-order system and the period of the masking signal will be modified. The ρ values of the two systems and their timescales are shown in table 5.1.

System Timescale	20-Node (ρ)	200-Node (ρ)
10ms	0.0025	0.025
100ms	0.025	0.25
200ms	0.05	0.5
400ms	0.1	1

Table 5.1 The values of ρ for two delay-feedback reservoir Simulink models with different virtual nodes, when tested at four different timescales. As shown in figure 5.1, a smaller ρ suggests a lower virtual node connectivity.

As the input scaling and coupling gain have a large effect on the computational performance of the reservoir, a parameter sweep (sweeping both the input scaling and coupling gain) is performed for each reservoir node and timescale permutation, for both the NARMA-10 and Santa Fe benchmarks as well as the system metrics discussed in chapter 3. Table 5.2 shows the model parameter values and ranges for the proposed parameter sweep:

Name	Symbol	Value (20-Node)	Value (200-Node)
Coupling Gain	β	0.05-2	0.05-2
Input Scaling Factor	δ	0.05-2	0.05-2
Mackey-Glass Exponent	n	9	9
Time Delay	τ	80 s	80 s
Masking Signal Period	θ	4 s	0.4 s

Table 5.2 Parameter values of the delay-feedback reservoir Simulink model during the input scaling and coupling gain parameter sweep.

The training and testing of the output, and the methodology for generating the input data of the delay-feedback reservoir, is described in section 4.3.

5.3 Experimental Results

Within this section, the results from the parameter sweep defined in section 5.2 are analysed.

The parameter sweeps show the performance of the reservoir evaluated at each step within the parameter sweep, as a colour spectrum; with the lowest calculated NRMSE showing as blue and the highest calculated NRMSE showing as yellow. It should be noted that as a wide range of values are used for the input scaling and coupling gain, it was common for the reservoir to go into saturation and become unstable, giving very high NRMSE values. Therefore, the maximum displayed value of NRMSE is 1; as an NRMSE above 1 always indicates poor performance.

The parameter sweeps display both of the training and testing NRMSE values as it is important to consider both values independently. Although in this case both of the training and testing NRMSE values are similar, this may not always be true as the Moore-Penrose pseudo-inverse function is used to calculate the weight matrix. This training algorithm sometimes calculates a weight matrix with very large values, which may result in an overdetermined or unstable solution.

5.3.1 NARMA-10 Benchmark

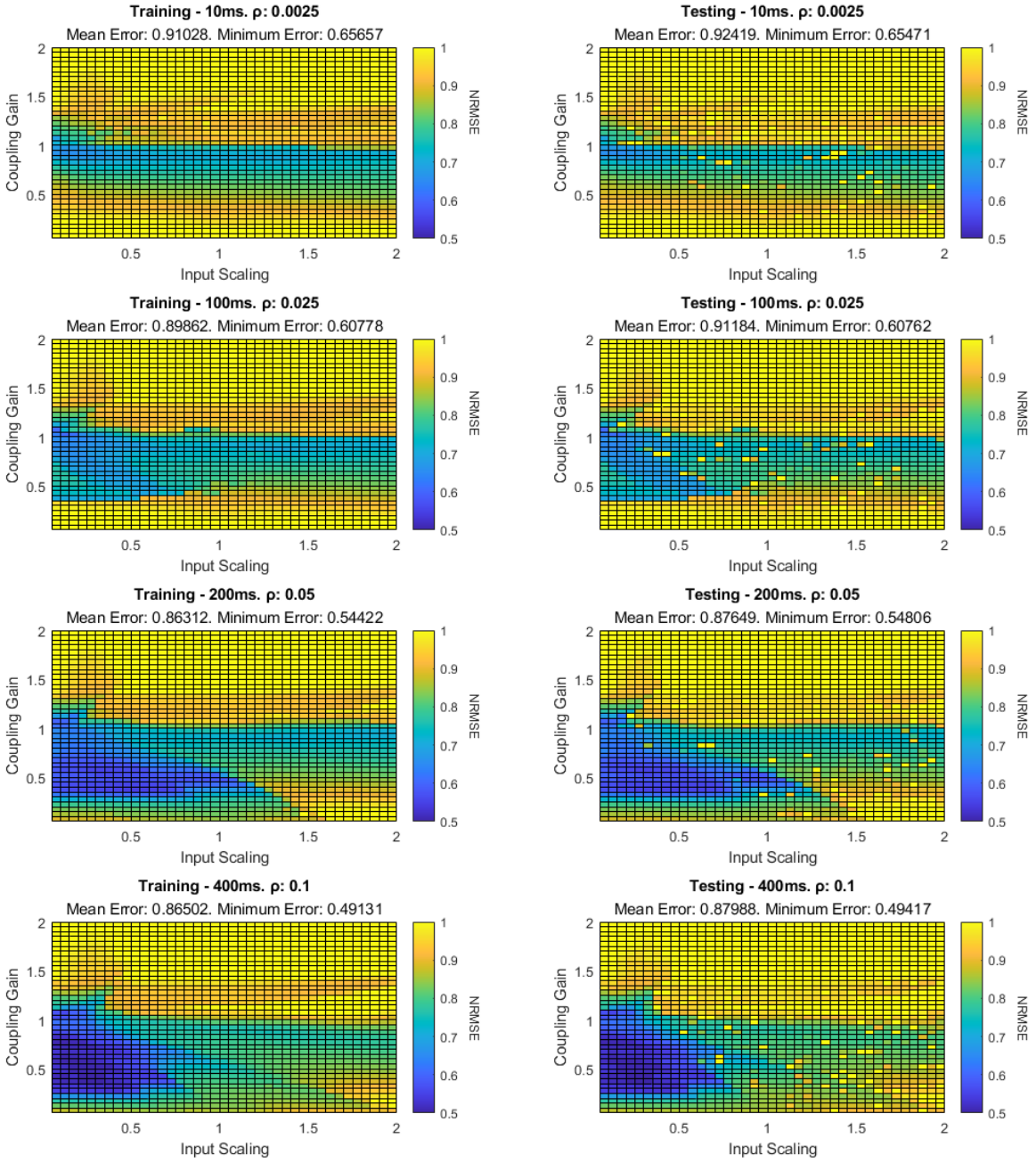

Figure 5.3 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20-node system running the NARMA-10 benchmark, at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

Timescale		10ms	100ms	200ms	400ms
20-Node	ρ	0.0025	0.025	0.05	0.1
	NRMSE	0.655	0.608	0.548	0.494
	Input Scaling	0.1	0.05	0.45	0.3
	Coupling Gain	0.9	1.05	0.35	0.35

Table 5.3 Table of the best performing sweep parameters on the test dataset for the 20-node system with the NARMA-10 benchmark.

20-Node System. Figure 5.3 shows the effects of the input scaling and coupling gain within a 20-node system evaluating the NARMA-10 benchmark at four different reservoir timescales.

When the timescale is small, a few interesting observations can be made. First, the operating region of NARMA-10 is spread over a large range of input scaling values, indicating that the reservoir system is not greatly sensitive to changes in input scaling. Second, the reservoir system is greatly sensitive to the coupling gain, with the best performance being within a coupling gain range of 0.8 – 1. Finally, even though the operating region is quite large, the best testing NRMSE value is 0.655 indicating a poor training performance.

This result is to be expected as the 20-node delay-feedback reservoir has a timescale of 10ms with a θ of 4, giving the system a ρ of 0.0025. This indicates that as the transient exists for such a small amount of time, there is little information to bleed-over to the next virtual node. This means that it would be very hard to saturate the system by applying additional inputs, no matter what scaling factor they may be, as the transient response would decay before propagating to the next virtual node. This is also why the reservoir operates best with a coupling gain range of 0.8 – 1, as anything lower than 0.8 reduces the output of the integrator to be much lower than that of the input signal; reducing the effectiveness of the feedback loop.

When the timescale increases, further observations can be made: the operating region of NARMA-10 becomes more compact, the sensitivity to the input scaling

Timescale		10ms	100ms	200ms	400ms
200-Node	ρ	0.025	0.25	0.5	1.0
	NRMSE	0.586	0.411	0.340	0.308
	Input Scaling	0.05	0.25	0.15	0.15
	Coupling Gain	1.05	0.4	0.45	0.8

Table 5.4 Table of the best performing sweep parameters on the test dataset for the 200-node system with the NARMA-10 benchmark.

increases, the sensitivity to the coupling gain decreases, and the average training/testing NRMSE value decreases. This can once again can be explained in terms of ρ.

As the timescale increases, ρ increases. This means that more of the transient response of the non-linear node is being passed on to the next virtual node creating a greater bleed-over, which increases the reservoir's ability to mix information, and thus increasing its performance resulting in a lower NRMSE. If an unusually large signal is continuously fed into the integrator, it will become saturated, causing the system to become unstable. As there is a larger signal within the feedback loop, due to the increase in ρ, the tolerance for how large the input signal can be must decrease; causing an increased sensitivity to the input scaling and allowing for a greater range between 0 and 1 for the coupling gain.

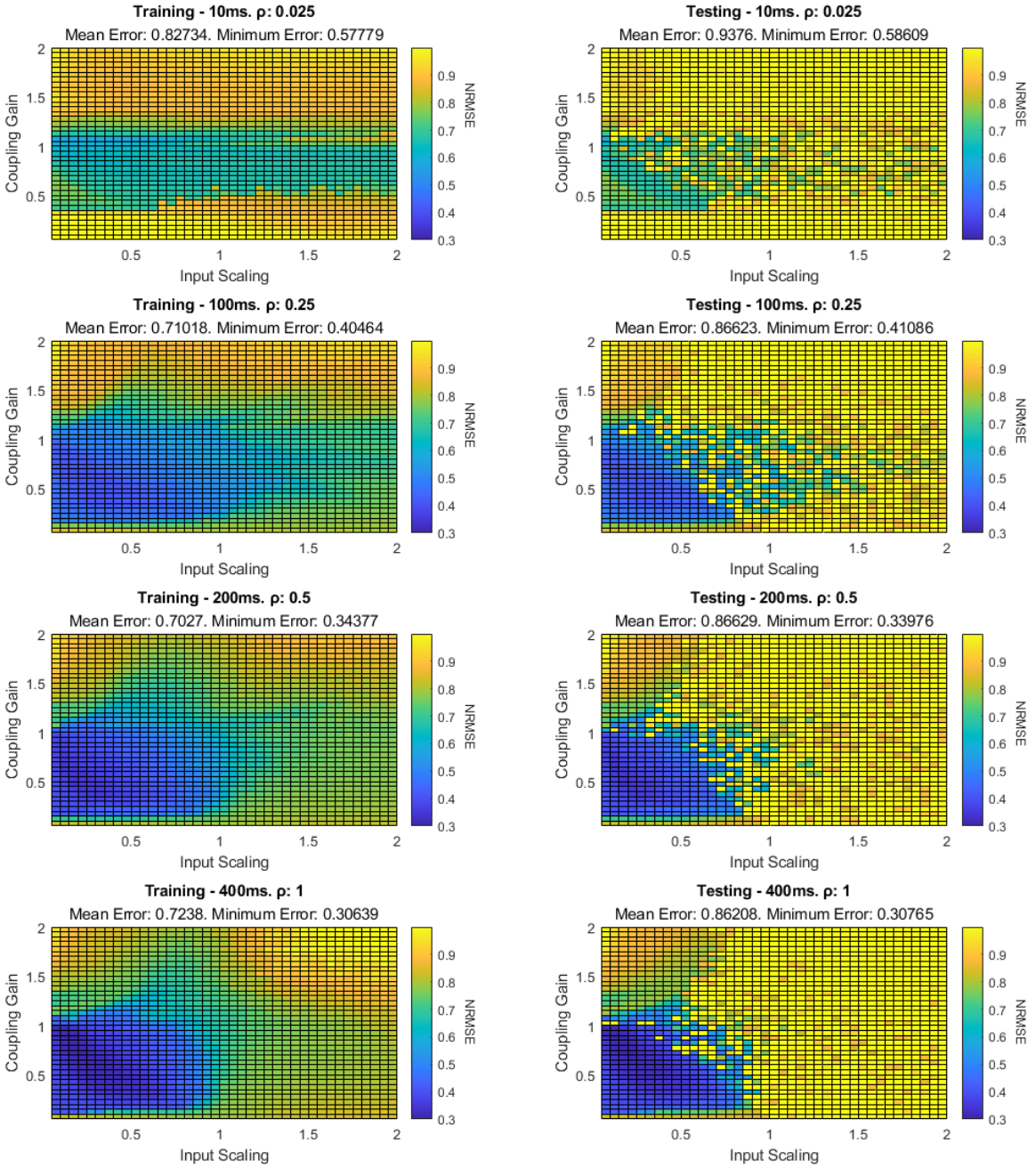

Figure 5.4 Graphs showing the training and testing NRMSE values of a parameter sweep for a 200-node system running the NARMA-10 benchmark, at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

200-Node System. Figure 5.4 shows the effects of the input scaling and coupling gain within a 200-node system evaluating the NARMA-10 benchmark at four different system timescales. Many of the observations made in the 20-node system can be applied to the 200-node system, but with a few exceptions.

The operating region of the NARMA-10 benchmark is much larger within the 200-node system given the same timescales; this is due to the number of the virtual nodes within the 200-node system being an order of magnitude greater than the number of the virtual nodes within the 20-node system. The increase in virtual nodes not only increases the value of ρ, but also gives the delay-feedback reservoir much richer dynamics. It would appear that the increase in performance is due to ρ (at least for the case of running the NARMA-10 benchmark). This is evident by looking at the 100ms 20-node system, and the 10ms 200-node system as they both have a ρ of 0.025. The 20-node system has a minimum testing NRMSE of 0.608, while the 200-node system has a minimum testing NRMSE of 0.586. This would indicate that the full system dynamics of the 200-node system are not being utilised, as the 20-node system performs almost as well; this can also be visually confirmed by examining the corresponding parameter sweeps.

Another observation can be made in regards to the difference between the training and testing NRMSE values. The training NRMSE value exhibits a boundary gradient (moving from good to poor performance) around the operating region, whereas the testing NRMSE value is much more sporadic, with a much less defined gradient. This is due to the training algorithm causing the weight matrix to be overdetermined and unstable, as the weight matrix for a 200-node system has 10 times the amount of values than the 20-node system.

Despite these differences, it is clear that there is a specific operating region in which the NARMA-10 benchmark performs best. Increasing the timescale within both systems appears to condense the operating region towards the bottom-left of the parameter sweep i.e. lower values of input scaling and coupling gain. The next step is

to investigate if this is true for other computational tasks or a specific feature of the NARMA-10 benchmark. Hence, a second benchmark needs to be run which is selected to compliment NARMA. The Santa Fe benchmark requires less memory, but a greater degree of non-linearity, as discussed within chapter 3,

5.3.2 Santa Fe Benchmark

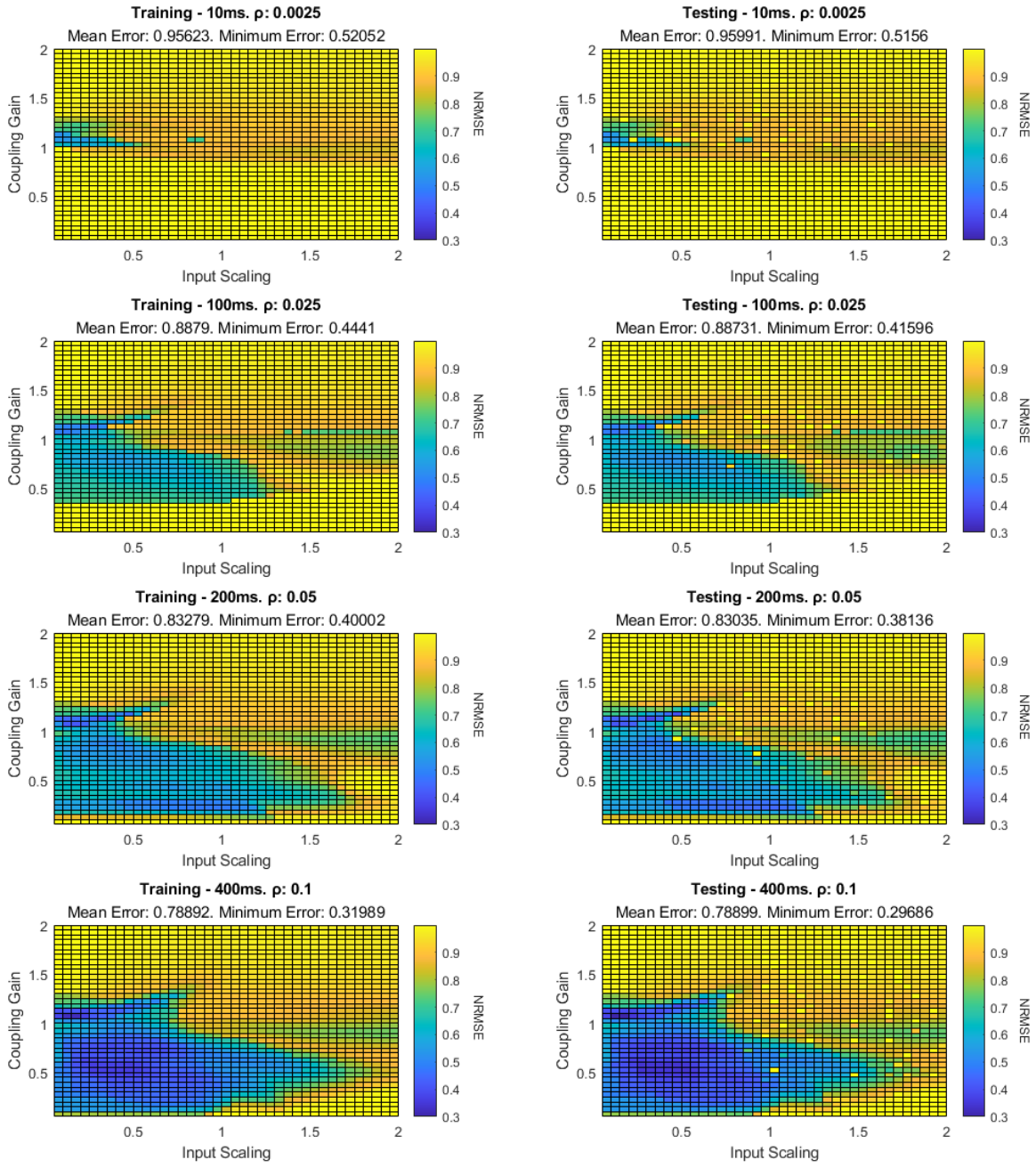

Figure 5.5 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20-node system running the Santa Fe benchmark, at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

Timescale		10ms	100ms	200ms	400ms
20-Node	ρ	0.0025	0.025	0.05	0.1
	NRMSE	0.516	0.416	0.381	0.297
	Input Scaling	0.05	0.25	0.30	0.15
	Coupling Gain	1.1	1.1	1.1	1.05

Table 5.5 Table of the best performing sweep parameters within the testing dataset for the 20-node system with the Santa Fe benchmark.

20-Node System. Figure 5.5 shows the effects of the input scaling and coupling gain within a 20-node system evaluating the Santa Fe benchmark at four different system timescales.

Even though the same 20-node reservoir system was used for both the NARMA-10 and Santa Fe benchmark, the results look very different. The only common observation is that as the timescale increases, the region of best performance expands; but how the region of best performance expands is different. In the case of the NARMA-10 benchmark, the region of best performance tended to become more concentrated; moving towards the bottom-left of the parameter sweep. In contrast, the Santa Fe benchmark region of best performance starts very small at a timescale of 10ms, but then expands out towards the right of the parameter sweep, i.e. increasing values of input scaling, as the timescale increases; with the computational performance increasing within the previously stated region of best performance (as shown as a decrease in NRMSE). However, the most important distinction between the operating regions of the NARMA-10 and Santa Fe benchmarks is that the Santa Fe benchmark has two distinct regions of best performance rather than one.

The region of best performance within the bottom-left quadrant of the parameter sweep is somewhat similar to that of the NARMA-10 system; showing good performance. This implies that there are some mutual dynamics within the region of this reservoir system that both benchmark tasks benefit from. However, the second region of best performance is a much smaller area, within the middle-left of the parameter sweep, where the best NRMSE results are observed. This implies that there is a particular

Timescale		10ms	100ms	200ms	400ms
200-Node	ρ	0.025	0.25	0.5	1.0
	NRMSE	0.374	0.138	0.112	0.098
	Input Scaling	0.25	0.30	0.65	0.25
	Coupling Gain	1.1	1.1	1.15	1.1

Table 5.6 Table of the best performing sweep parameters within the testing dataset for the 200-node system with the Santa Fe benchmark.

quality within the dynamics of the reservoir system that the Santa Fe benchmark can take advantage of; this is evident as the minimum NRMSE value is lower in all of the Santa Fe timescales compared to their NARMA-10 counterpart.

200-Node System. Figure 5.6 shows the effects of the input scaling and coupling gain within a 200-node system evaluating the Santa Fe benchmark at four different system timescales.

The observations made within the 20-node system are mirrored within the 200-node system; where the two regions of best performance have greatly expanded due to the increase in virtual nodes. Even with a timescale of 10ms and a ρ of 0.025, almost half of the parameter space results in having a low-to-moderate NRMSE value.

During the evaluation of the NARMA-10 benchmark, it was noticed that the 100ms 20-node system and the 10ms 200-node system performed almost the same, having an NRMSE difference of only 0.022. It was hypothesised that for the NARMA-10 benchmark the value of ρ was more important as the NARMA-10 benchmark could not utilise the full dynamics of the 200-node system. To check if this is benchmark agnostic or a perk of the NARMA-10 benchmark, the minimum testing NRMSE value was checked for the 100ms 20-node system and the 10ms 200-node system while evaluating the Santa Fe benchmark; giving an testing NRMSE value of 0.444 and 0.374 respectively. This implies that for the case of the Santa Fe benchmark, the other dynamics of the reservoir system dominate over the dynamics related to the value of ρ.

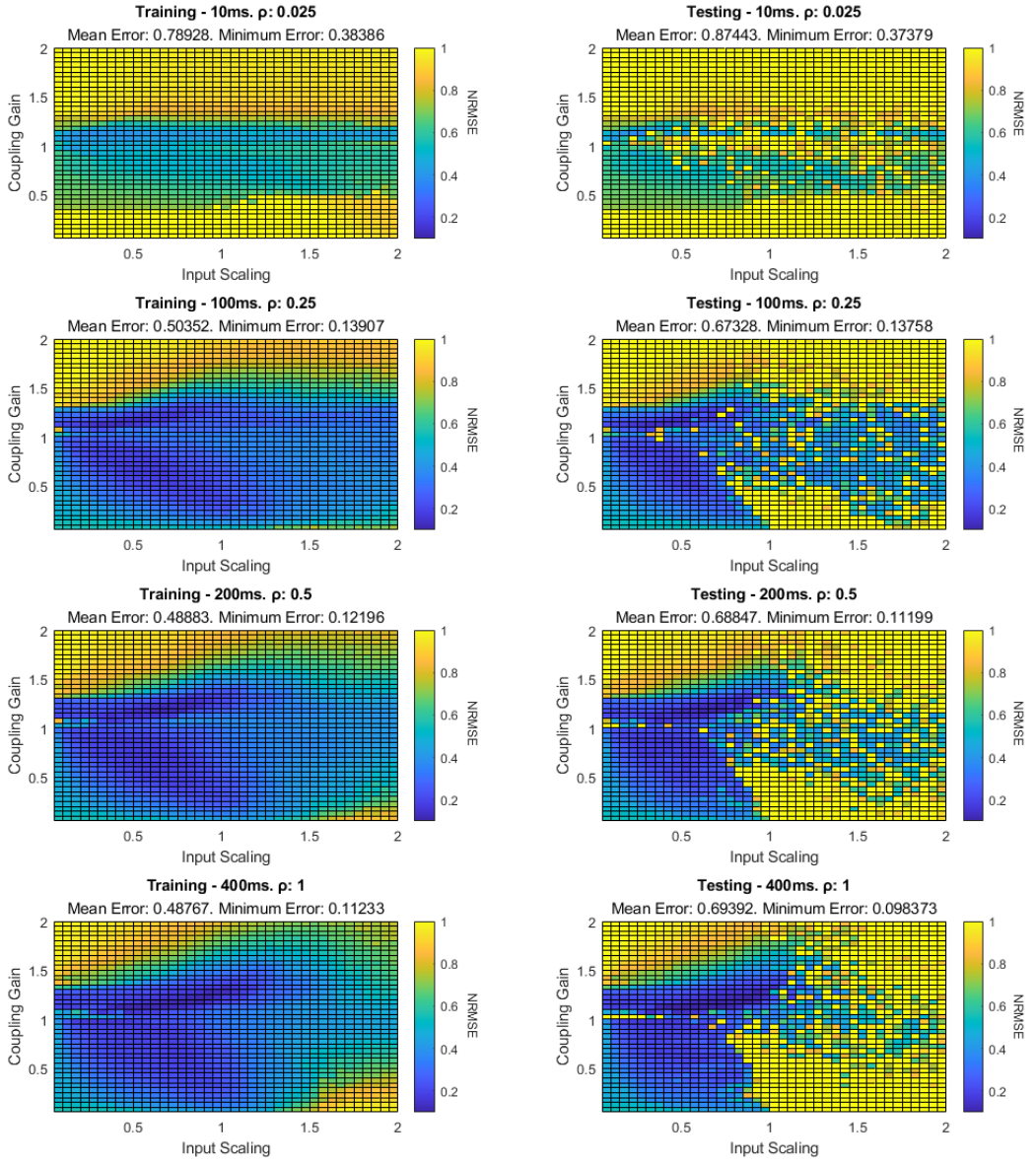

Figure 5.6 Graphs showing the training and testing NRMSE values of a parameter sweep for a 200-node system running the Santa Fe benchmark, at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

Much like the 200-node NARMA-10 system, the same discrepancies within the training and testing NRMSE values can be observed. Despite this, it is clear that the delay-feedback reservoir model is better suited to solve the Santa Fe benchmark as the best minimum testing NRMSE value was 0.098, while the best minimum testing NRMSE value for the NARMA-10 benchmark was 0.308. In order to establish why the Santa Fe benchmark performed much better, the metrics of the delay-feedback reservoir are analysed in the following sections.

5.3.3 System Metrics

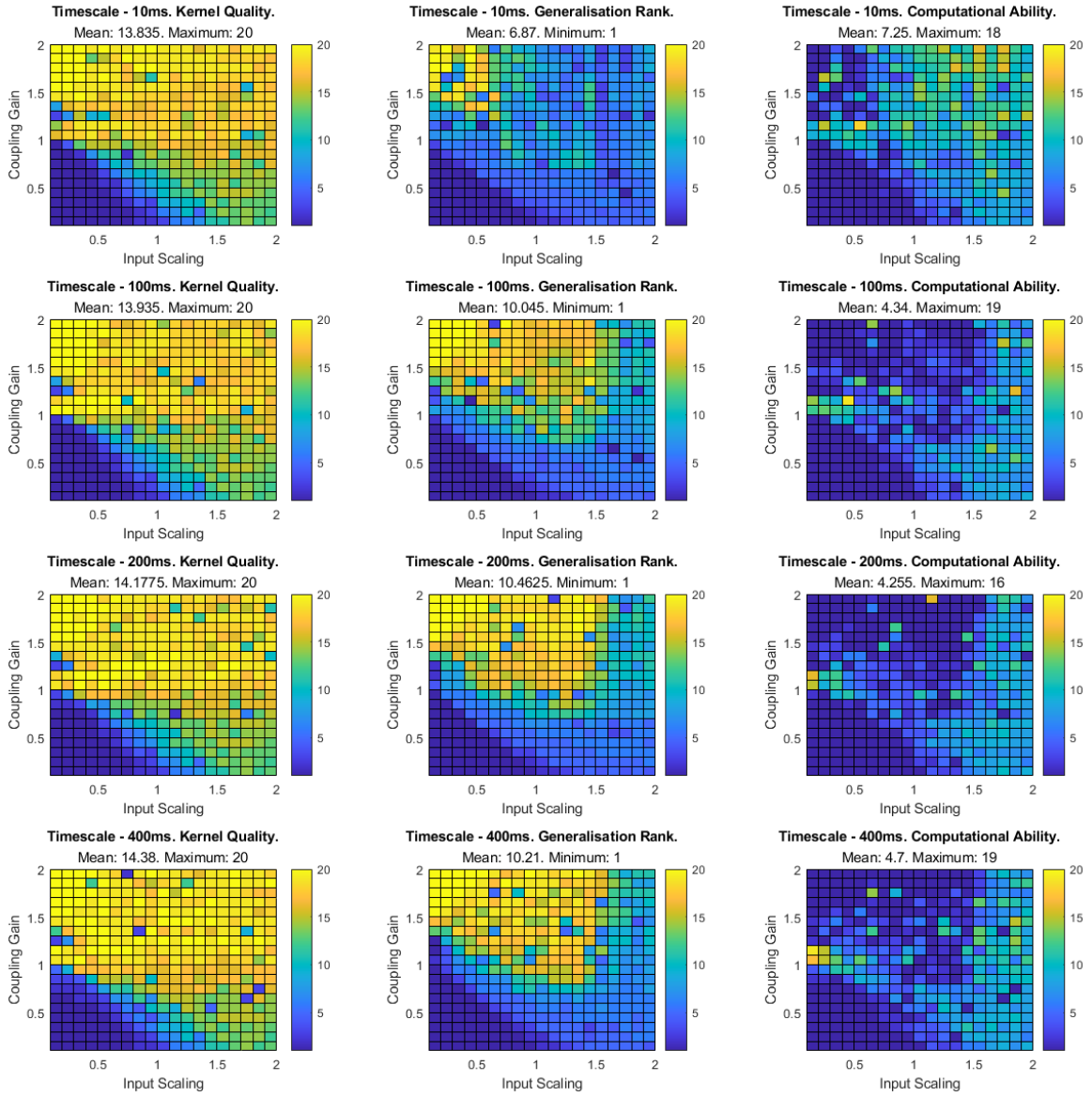

Figure 5.7 Graphs showing the Kernel Quality, Generalisation Rank, and Computational Ability of a parameter sweep for a 20-node system, ran at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

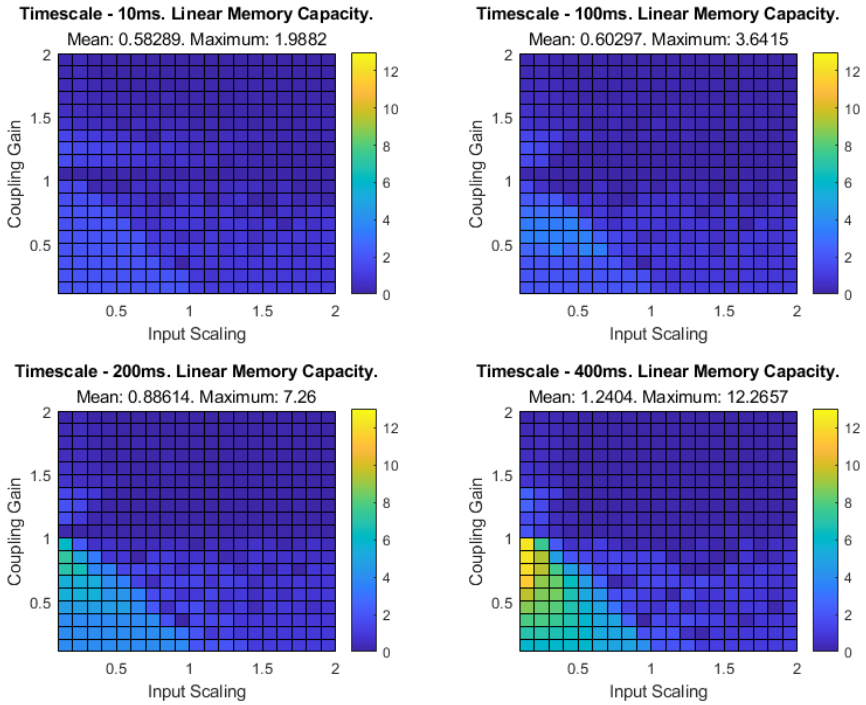

Figure 5.8 Graphs showing the Linear Memory Capacity of a parameter sweep for a 20-node system, ran at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

20-Node System. Figures 5.7 and 5.8 show how the system metrics of a 20-node delay-feedback reservoir system change during a parameter sweep of the input scaling and coupling gain. Before any comparison between the system metrics and the computational benchmarks are made, the effect of the timescale on the system metrics is first determined.

System Metric Evaluation. Within figure 5.7, a relatively consistent pattern of the Kernel Quality (KQ), Generalisation Rank (GR), and Computational Ability (CA) can be observed across all timescales apart from the 10ms timescale; which is due to system instability and saturation. As previously discussed, when the timescale is set to 10ms it is much harder for the system to become saturated and unstable. However, as the timescale increases (and therefore ρ increases), it becomes easier to force the system to become unstable and saturate; this is evident by looking at the KQ and GR metrics. When the value of KQ is high, the system is able to map input vectors onto

a high-dimensional non-linear space, but when the system goes into saturation, the KQ will still report a high value as a saturated reservoir has a strong non-linearity, as values are converging to the same saturation point. The full picture can be observed when GR is also taken into account. As GR is a measure of how well a system can generalise input vectors, when there is a low GR, the system can detect similar inputs and generalise them. However, when GR is high, the system is unable to generalise and recognise similar inputs, and one of the reasons for this is an unstable or saturated system. This can be visually confirmed by looking at the CA of the system at different timescales; as the timescale increases, pockets appear where the system is in saturation.

The relationship between the Linear Memory Capacity (LMC) of the system (shown in figure 5.8) and the timescale shows that as the timescale increases, the LMC also increases. This is to be expected as stronger connections between the virtual nodes are made as ρ increases, allowing for more of the previous input stimuli to be passed between virtual nodes.

The NARMA-10 Benchmark. Comparing the region of best performance of the 20-node NARMA-10 system (shown in figure 5.3) with the CA and LMC system metrics, shows that the area in which the best testing NRMSE values are observed is where the KQ is at its lowest and the LMC is at its highest. This finding is not unexpected as the NARMA-10 benchmark relies on inputs that have a lag of 10 time steps, and therefore requires the LMC to be at least 10 to sufficiently recall past inputs. Also, the NARMA-10 benchmark has a relatively weak non-linearity, and therefore does not require a very high CA to produce a good NRMSE value.

The Santa Fe Benchmark. The comparison between the region of best performance within the 20-node Santa Fe system (shown in figure 5.5) and the system metrics give some insight as to why the Santa Fe benchmark has two regions of best performance, while the NARMA-10 benchmark has one. The first region of best performance is within the bottom-left quadrant of the parameter sweep where the moderate NRMSE results occur, and the second region of best performance is within the middle-left of

the parameter sweep where the best performing NRMSE results occur. As the Santa Fe benchmark is highly non-linear with the reliance on only a single input delay, an ideal reservoir would have a high CA and a low-to-moderate LMC. This requirement is perceived within the graphs as the first best performing operating region exhibits a high LMC and a low CA, and the second best performing operating region exhibits a low LMC and a high CA; this is true for all timescales.

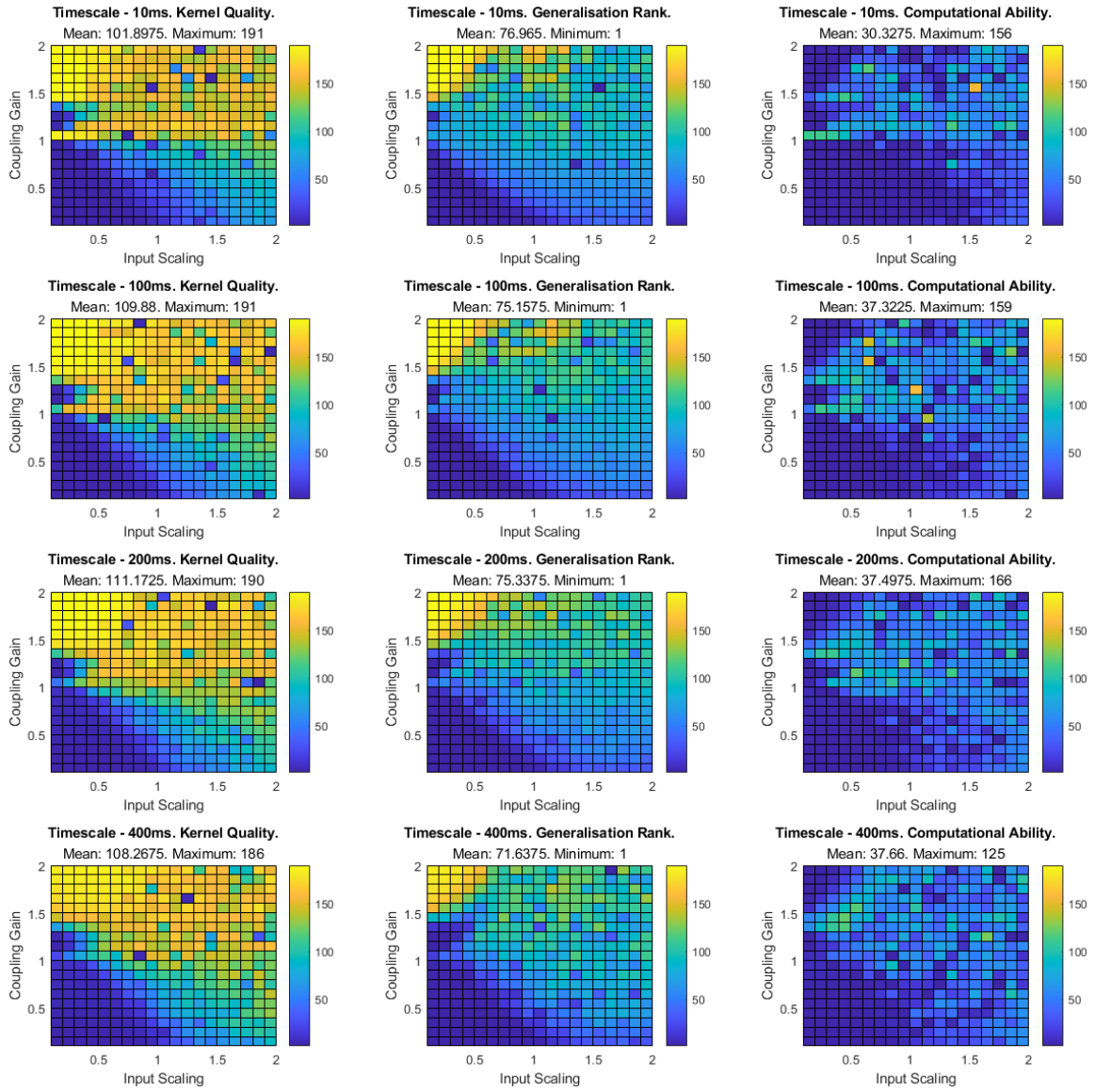

Figure 5.9 Graphs showing the Kernel Quality, Generalisation Rank, and Computational Ability of a parameter sweep for a 200-node system, ran at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

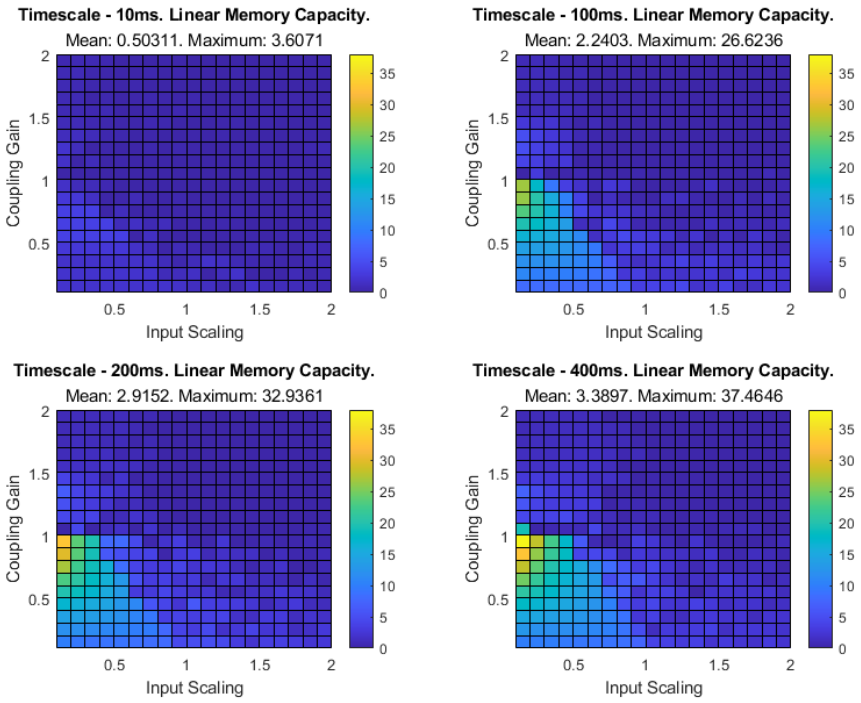

Figure 5.10 Graphs showing the Linear Memory Capacity of a parameter sweep for a 200-node system, ran at four different timescales, with the input scaling and coupling gain ranging between 0.05 and 2.

200-Node System. Figures 5.9 and 5.10 show how the system metrics of a 200-node delay-feedback reservoir system changes during a parameter sweep of the input scaling and coupling gain. Before the system metrics and computational benchmarks are compared, the system metrics for the 200-node system are evaluated, and compared to the 20-node system.

System Metric Evaluation. Just as with the 20-node system, there is little change within the mean of the KQ, GR, and CA across all timescales as observed within figure 5.9. However, when compared to the 20-node system (shown in figure 5.7), the mean and maximum values of the KQ, GR, and CA have greatly increased. This is to be expected as it was previously discussed within section 3.3, that the maximum possible value for the KQ is equal the number of nodes within the reservoir system; which has increased tenfold over the 20-node system.

From figure 5.10, a proportional relationship between the LMC of the system and the timescale can be observed; which is once again mirrored by the 20-node system (figure 5.8). An interesting observation is that the maximum value and distribution of the LMC within the 100ms 20-node system and the 10ms 200-node system, the systems with the same value of ρ, are very similar. This once again indicates that in systems that share the same value of ρ, they share some dynamical properties which are independent of the number of virtual nodes within the system.

The NARMA-10 Benchmark. A comparison between the region of best performance of the 200-node NARMA-10 benchmark and the 200-node system metrics gave the same conclusions that were made within the 20-node case; the best NRMSE values are observed where the KQ is at its lowest and the LMC is at its highest. This is to be expected as the region in which the LMC is the greatest is in the same location within the parameter sweep in both the 20- and 200-node systems.

As the CA is much higher within the 200-node system, some moderate-to-poor NRMSE values appear on the fringes of the LMC region. This lack of LMC allows the training algorithm to calculate a weight matrix which overfits the training dataset. This would produce a moderate-to-poor training NRMSE value, but as the reservoir system has no LMC, it cannot recreate the dynamics of the NARMA-10 signal, which produces a high testing NRMSE value. This explains the discrepancy between the training and testing NRMSE values within figure 5.4.

The Santa Fe Benchmark.

Comparing the region of best performance between the 200-node Santa Fe benchmark and the 200-node system metrics, results in similar conclusions made within the 200-node case, with the exception of both regions of best performance larger due to the increase in the area and magnitude of the CA. The area of greatest computational performance is still within the middle-left region, as this is where the maximum CA is.

Despite the large increase of the CA across the entire parameter space of the 200-node system, the testing NRMSE value is still very sporadic after the input scaling

passes a value of 1; this is because the GR within the bottom-right of the quadrant is poor. When the input scaling becomes greater than 1, the signal within the delay-feedback reservoir is bound to be large due to the addition of an input signal, which provides a greater than unity positive feedback, resulting in saturation. If there are extreme peaks or fluctuations within the feedback loop of the reservoir, these values need to be non-linearly mapped to the virtual nodes of the system; requiring the KQ of the reservoir to be large. The greater the non-linear fluctuations are, the greater the KQ has to be to support it; providing the system has not gone into saturation.

5.4 Discussion

From the results shown in section 5.3, some conclusions can be drawn regarding: the effect of the timescale on a delay-feedback reservoir, the region of best performance that computational tasks occupy within the coupling gain and input scaling parameter space, and how a computational task with a high memory requirement can be emulated on a smaller virtual node system. This section aims to explore these conclusions further, so that sub-hypothesis 1 and sub-hypothesis 2 can be tested.

5.4.1 Timescale vs Virtual Nodes

From the results shown in section 5.3.3 it can be established that the timescale has little effect on the KQ, GR, and CA of a system, but has a large effect on the LMC; as shown in both the 20- and 200-node systems. Conversely, it found that the number of virtual nodes within a system has little effect on the LMC of the system, but has a large effect on the KQ, GR, and CA of the system; as is observed across all four timescales. Given that the KQ, GR, and CA affects the dimensionality and how well non-linearity is processed within a system; and given how the LMC affects how long previous input stimuli are present within the system, the following conclusion can be made: if a particular computational task has a large dependency on previous input stimuli, but

does not exhibit strong non-linearity, then additional performance may be gained by increasing the timescale of the system. Alternatively, if a particular computational task exhibits strong non-linearity and has a small dependency on previous input stimuli, additional performance may be gained by increasing the number of virtual nodes within the system.

5.4.2 Region of Best Performance

Within section 5.3 it was observed that both the NARMA-10 and Santa Fe benchmark tasks had a specific regions within the parameter sweep space where the greatest computational performance is observed; this was termed as the region of best performance of the benchmark. After evaluating the system metrics, it was found that the NARMA-10 task favoured a region which had a high LMC, and that the Santa Fe task favoured a region with a high CA.

Therefore, if a computational task is well characterised in terms of non-linearity, dimensionality, and its dependency on previous input stimuli, which it often is, then a region of best performance can be predicted within the parameter space of the system. To see if this can be applied as a general statement, additional experiments need to be performed when changing other key parameters within the delay-feedback reservoir system model, to see if computational tasks still have a region of best performance. If this is proven to be a general statement across all delay-feedback reservoir systems, then it will allow for an alternative approach to designing delay-feedback reservoirs, as systems can be tuned to solve a particular computational task if the parameter space of the reservoir is known.

5.4.3 System Emulation

Within section 5.3, it is hypothesised that a computational task within a 200-node system could be emulated within a 20-node system by only changing the value of ρ with

no performance decrease. It is found that when running the NARMA-10 benchmark, the 100ms 20-node system and the 10ms 200-node system have similar performance, having a testing NRMSE difference of 0.022; implying that the 200-node NARMA-10 benchmark could be emulated within a 20-node system. Conversely, for the Santa Fe benchmark run, the 100ms 20-node system and the 10ms 200-node system have a testing NRMSE value of 0.416 and 0.374 respectively; as there is a greater range in error between the 20- and 200-node systems, it is unclear if the Santa Fe benchmark can be emulated within a smaller system.

To test this hypothesis, the best performing input scaling and coupling gain parameters, determined in the parameter sweeps within section 5.3, are simulated 100 times using different random seeds, so that a statistical analysis can be performed. Each set of 100 simulations is set to the same input scaling, coupling gain, and value of ρ, but to a different, randomly generated random-weighted no-offset mask; a new input sequence for the NARMA-10 benchmark is also generated for every input. To better test the hypothesis, an additional 200-node system with a timescale of 40ms, giving a ρ value of 0.1, is also simulated so an additional comparison can be made. A coarse parameter sweep finds a best performing input scaling and coupling gain to be 0.3 and 0.45 for the NARMA-10 benchmark and 0.2 and 1.1 for the Santa Fe benchmark, with the same 100 sample statistical analysis also performed The testing NRMSE values are calculated from the resulting simulations and plotted as a box-plot; this is shown in figure 5.11.

The NARMA-10 results shown in figure indicate 5.11 that the value of ρ is the dominating factor when it comes to the performance of the systems, and that is it independent of the number of virtual nodes within the reservoir. The 20- and 200-node systems (which share a ρ value of 0.025 and 0.1) have almost the same performance. This is to be expected, because the NARMA-10 benchmark heavily relies on the dependency of the previous input stimuli and a low reliance on the non-linearity and dimensionality of the reservoir. Providing there is enough non-linearity and dimensionality within the

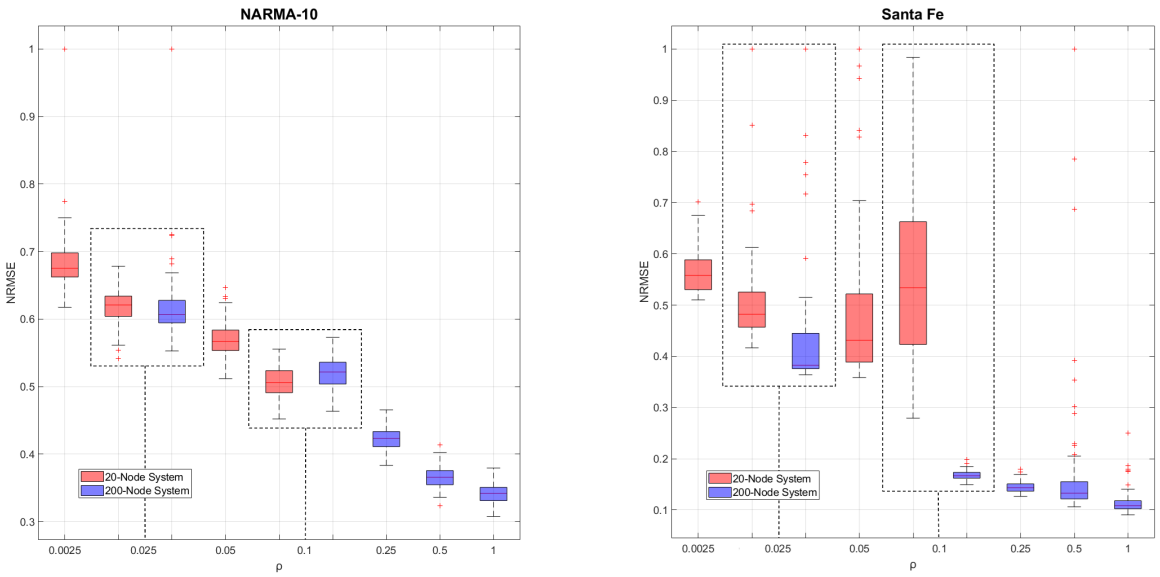

Figure 5.11 Graphs showing the training NRMSE results of 100 simulation runs for several values of ρ. Each box-plot is a reservoir configured with the best performing input scaling and coupling gain parameters found in section 5.3 and 5.4.3. Reservoir systems sharing the same value of ρ are shown within a dotted box for comparison.

20-node system, an increase of the number of nodes does not increase the performance as the system already exhibits the necessary non-linear dynamics for computation.

The Santa Fe results, once again shown in figure 5.11, indicate that the number of virtual nodes within the reservoir is the dominating factor in terms of the performance of the systems in this case. However, unlike the NARMA-10 benchmark, the value of ρ has an effect on the 20-node Santa Fe benchmark, with the interquartile ranges increasing as ρ increases. This is likely because the chosen input scaling and coupling gain are already on the edge of stability, meaning the random deviations within the random mask may have forced the reservoir to become unstable and perform worse. The effect is much more prominent within the 20-node reservoir as the 200-node reservoir is more robust. These results confirm that the following hypothesis, *if a computational task has a large dependency on previous input stimuli and is being run on a many-node system, it is possible to emulate the same performance within a smaller-node system providing that the smaller-node system has a value of ρ equal to that of the previous system* can be accepted. The limiting factor of how small the new smaller-node system

can be is that it exhibits the minimum non-linearity and dimensionality required to run the computational task.

5.5 Summary

Within this chapter, the effect that the system timescale has on the performance of a delay-feedback reservoir system is investigated with the aim of addressing sub-hypothesis 2: *adjusting the timescale of the integrator within the non-linear node can lead to an increase in performance without a significant change in the hardware architecture.*

Two delay-feedback reservoir systems, one with 20-nodes and the other with 200-nodes, were tested at four different system timescales: 10ms, 100ms, 200ms and 400ms. Each system configuration was evaluated with both the NARMA-10 and Santa Fe benchmarks in order to see how the delay-feedback reservoir system dynamics change. The two benchmarks were chosen because the NARMA-10 benchmark requires a delay-feedback reservoir system with a high memory capacity and a low non-linearity, while the Santa Fe benchmark requires the opposite high non-linearity and low memory capacity. By running these opposite benchmarks and analysing the results, it is possible to see which dynamics are present and which are missing.

To confirm these changes in dynamics, each delay-feedback reservoir system configuration was also analysed purely in terms of its system metrics: Kernel Quality (KQ), Generalisation Rank (GR), and Linear Memory Capacity (LMC). As the delay-feedback reservoir system is highly susceptible to input scaling and coupling gain, a parameter sweep was performed with parameters for each system configuration between the values of 0 and 2. The results of these parameter sweeps allow for several conclusions to be made, which are discussed within this section.

5.5.1 Effect of Timescale on System Performance

The results from the parameter sweeps show that the timescale of the delay-feedback reservoir system is proportional to the linear memory capacity. This relationship is best explained in terms of ρ; the new parameter introduced to measure the strength between virtual nodes. When ρ is low, the connection between virtual nodes is weak. This weak connection means that only a part of the the information from the previous node is being passed on to the next node; this can be thought of as a type of memory. As ρ increases, the connection between virtual nodes increases, allowing for more information from the previous nodes to be passed on to the next node. This relationship appears to be independent of the number of virtual nodes within the system (as shown in the 20-node 100ms and 200-node 10ms cases where the LMC is about 3.6), however, this relationship only holds if there is a sufficient number of virtual nodes within the system, as the maximum linear memory capacity is always equal to the number of virtual nodes within the system.

While there was a strong relationship between the timescale and the LMC of the system, the timescale has virtually no effect on the KQ or GR of the system. This is to be expected as when the connectivity between virtual nodes increases, more of the information from the previous node is passed on, which adds no additional non-linearity or dimensionality to the system dynamics. This can be confirmed when comparing the performance of the NARMA-10 and Santa Fe parameter sweeps. The Santa Fe benchmark performed universally better when the amount of nodes within the system increased, whereas the NARMA-10 benchmark performed better as the value of ρ increased. This shows that the Santa Fe benchmark depends more on the number of nodes rather than the change in dynamics when the value of ρ is changed; which is to be expected given Santa Fe requires non-linearity rather than memory.

5.5.2 Region of Best Performance

Another important conclusion that can be made when comparing the performance of the NARMA-10 and Santa Fe parameter sweeps is that each benchmark has a region of best performance where it has optimal performance. Throughout the analysis of the parameter sweeps of the NARMA-10 and Santa Fe benchmarks, it was found that there was a specific region within the parameter space of the system where optimal performance could be achieved. When comparing these regions of best performance to the parameter sweeps of the system metrics, it shows that the region of best performance of the NARMA-10 benchmark is within an area with a high LMC and low KQ, while the Santa Fe benchmark is within an area with a high KQ but low LMC.

Given the nature of the benchmark tasks chosen, this is an expected result as each benchmark performs best within a region of the parameter space where the required dynamics are the strongest. This provides some evidence to support sub-hypothesis 1: *computational tasks with particular characteristics perform best within distinct areas of the parameter space of the delay-feedback reservoir system model.* However, before this can be fully confirmed to be a general case statement, additional experiments need to be performed to determine if the computational tasks still have a region of best performance when other parameters within the experimental model are changed.

5.5.3 System Emulation

During the analysis of the parameter sweeps of the NARMA-10 benchmark it was noticed that the 20-node 100ms system and the 200-node 10ms system had similar performance, having a testing NRMSE difference of only 0.022. It was then hypothesised that it may be possible to emulate the performance of a 200-node system running the NARMA-10 benchmark within a 20-node system, given the same value of ρ. This allows delay-feedback reservoirs to be optimised by reducing the amount of virtual nodes, reducing the overall bandwidth of the system, without affecting its performance.

In order to test if this was a possibility or a statistical anomaly, 100 simulations were run with both the NARMA-10 and Santa Fe benchmarks. The best performing input scaling and coupling gain were chosen from each parameter sweep, with a different randomly generated random-weighted no-offset mask for each benchmark and a new input sequence for the NARMA-10 benchmark. The testing NRMSE values were calculated, as per the common methodology, and the resulting simulations are plotted within a box-plot for statistical analysis.

The results confirm that for the NARMA-10 benchmark, the value of ρ is dominant over the number of nodes within the delay-feedback reservoir system, with both 20- and 200-node delay-feedback reservoir systems performing almost identical when they both have the same ρ value. In contrast, the results for the Santa Fe benchmark show that the number of nodes within the delay-feedback reservoir system is the dominant factor, showing that the Santa Fe benchmark cannot be emulated. This proves that it is possible to emulate a larger node system within a smaller node system given that the computational benchmark has a strong dependency on memory, but a weak dependency on non-linearity.

5.5.4 Conclusions

The work completed within this chapter confirms acceptance of sub-hypothesis 2, as it was shown on multiple accounts that the performance of a delay-feedback reservoir can be increased by only changing the timescale of the system. Within the hardware accurate simulation model used, this timescale is changed by setting the time constant of the first-order filter. It is possible to change this value in physical hardware by adjusting the capacitance or resistance of the filter, which can be achieved in real time by using variable resistors and capacitors. The key conclusions of this chapter:

- If a particular computational task has a large dependency on previous input stimuli, but does not exhibit strong non-linearity, then additional performance may be gained by increasing the timescale of the system.

- If a particular computational task exhibits strong non-linearity and has a small dependency on previous input stimuli, additional performance may be gained by increasing the number of virtual nodes within the system.
- If a computational task has a large dependency on previous input stimuli and is being run on a many-node system, it is possible to emulate the same performance within a smaller-node system providing that the new system has a value of ρ equal to the previous system, providing the new system exhibits the minimum non-linearity and dimensionality required to run the computational task.
- The experiments within this chapter support sub-hypothesis 1: *computational tasks with particular characteristics perform best within distinct areas of the parameter space of the delay-feedback reservoir system model.*

Chapter 6

The Mackey-Glass Non-Linear Function

6.1 The Non-Linear Function

A desired property of a reservoir computing substrate is to provide a non-linear transform of the input stimuli. This non-linear transform serves two purposes within a system; the ability to map inputs to a high dimensional non-linear space, and to constrain information within an attractor region.

If a system was purely linear, it would not be able to solve non-linear problems due to a linear mapping of the inputs into the state space of the neural network; as found within early artificial neural networks [125]. The non-linear reservoir layer allows the network to learn non-linear behaviours as inputs are mapped into a high dimensional, non-linearly transformed state space within the network.

Another important feature that non-linearity provides is the ability to constrain information within a specific operating range by acting as an attractor. This can greatly help stabilise a system by mapping very large or small information values, that may have been produced by other components within a network, to values centred around zero. This is particularly useful within physical systems as there are often constraints, such as a maximum or minimum voltage, that could saturate or even cause damage to a system [23, 92, 126, 127].

Almost all physical substrates exhibit some degree of non-linearity. Many physical substrates that are considered to be linear, are either only assumed to be approximately linear, or behave linearly within a specific region of best performance. Exploiting the physics of a substrate to behave as a non-linear function has the advantage of not requiring a mathematical non-linear transform, which is typically implemented within software. This typically leads to a more efficient hardware architecture in terms of space within silicon and power consumption as the non-linear transform is an intrinsic part of the substrate.

6.1.1 The Mackey-Glass Non-Linear Function

Within section 4.2.3, it was decided to use the Mackey-Glass non-linear time delay differential equation to realise the non-linear function within the simulation model. This was chosen as the Mackey-Glass non-linear function has been shown to provide excellent performance in both simulated and physical delay-feedback reservoir architectures, and there are known, albeit complex, analogue implementations of the Mackey-Glass non-linear function making it ideal for hardware translation [128, 10, 129]. Despite the wide use of the Mackey-Glass equation, research has focused on applying the Mackey-Glass equation within hardware or simulation models, with the Mackey-Glass exponent assumed to be an approximate level of "how" non-linear the function is; the effect that the strength of the non-linearity has on computational performance and system dynamics has not been investigated. This chapter therefore aims to investigate the effects of the Mackey-Glass non-linear function, and how it affects the computational performance of benchmark tasks and the system dynamics.

Within the computational model, the Mackey-Glass non-linear function is given by the following equation:

$$f(x) = \frac{x}{x^n + 1} \tag{6.1}$$

Where n is the strength of the non-linearity of the equation.

To understand the effect that n has on the shape of the non-linearity, equation 6.1 is plotted with several values of n; this is shown in figure 6.1.

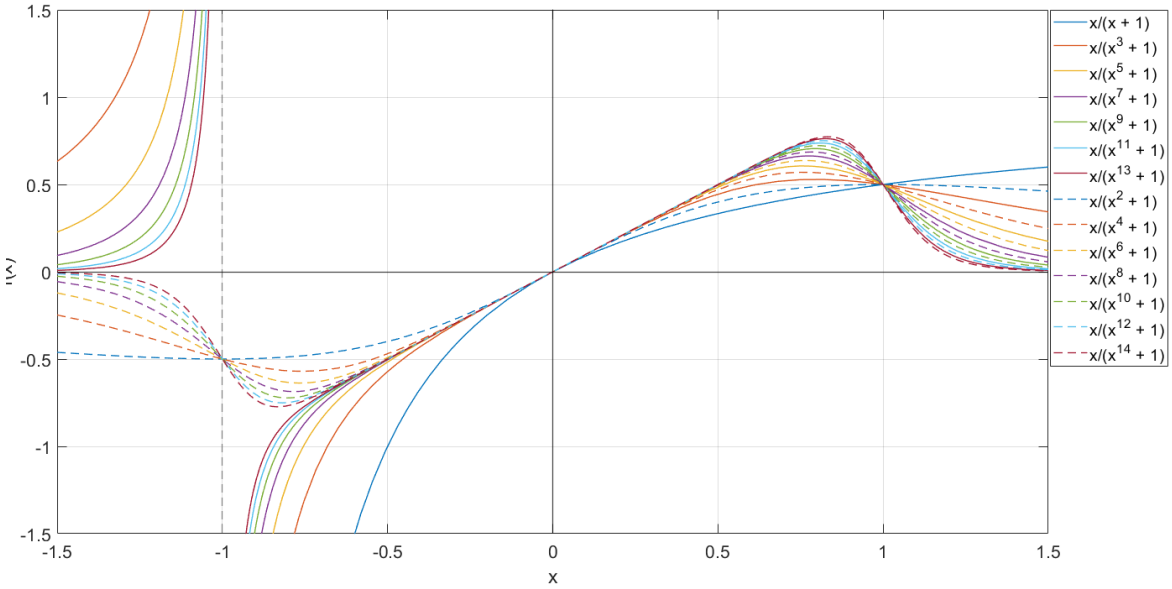

Figure 6.1 A parameter sweep of different values of n within the Mackey-Glass non-linear equation. The value of n is swept between 1 and 14. Odd values (solid lines) of n can be seen within the graph with an asymptote at $x = -1$, while even values (dashed lines) of n are $y = x$ symmetrical.

The graph in figure 6.1 shows several parameter sweeps of the Mackey-Glass non-linear function where the exponent n is set between 1 and 14. This figure shows three key features that the exponent has on the shape on the non-linear function: the function has a linear region of unity gradient centred around the origin that becomes more linear as the value of the exponent increases, the function has an asymptote at $x = -1$ for odd values of the exponent, and when considering the function within the feedback loop of the delay-feedback reservoir, the attractor behaviour it will exhibit.

For small values of n, the shape of the function is non-linear with a strong attractor behaviour where x is positive. However for negative values of x the function becomes asymptotic, mapping negative x values to positive y where $x < -1$.

As the value of n increases, the linear region around the origin becomes wider. This reduces the attractor range as more of the function is mapped to $y = x$, but the attractor behaviour becomes stronger outside the linear range. This is best seen by

looking at the derivative of the Mackey-Glass non-linear function; this is shown in figure 6.2 for n values 1, 2, 5, 6, 9, and 10.

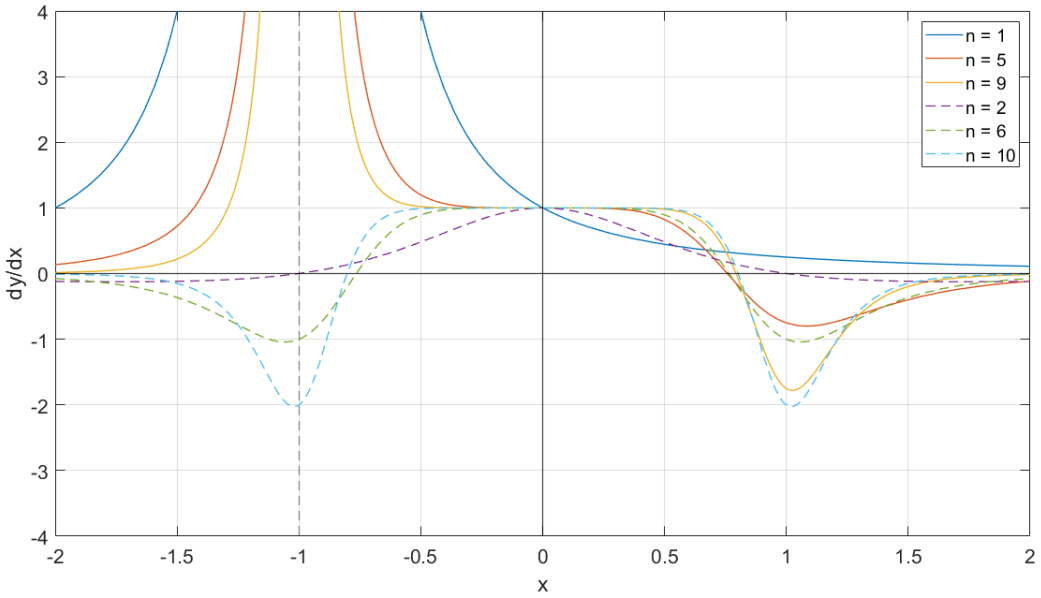

Figure 6.2 A parameter sweep of six different values of n within the derivative of the Mackey-Glass non-linear equation. The odd values are plotted with solid lines, while the even values are plotted with dashed lines.

Figure 6.2 clearly shows that as the value of n increases, the linear region within the function becomes wider, with the attractor behaviour becoming stronger at the edges of the linear region.

Another feature of the Mackey-Glass non-linear function is that for odd values of n, the function becomes asymptotic as $x = -1$. When n is even, the function becomes odd; mapping $f(-x) = -f(x)$. This means that when negative values exist within the Mackey-Glass delay-feedback reservoir system, they are simply mapped the same way the positive values are. However when n is odd, negative values which are $x > -1$ are drawn towards the asymptote, then mapped positive when $x < -1$; causing negative values to be mapped to positive within an odd n system.

Therefore it can be said that for an even valued n system, the attractor behaviour maps values outside of the linear range of the function towards zero. Whereas for an

odd valued n system, the attractor maps positive and large negative values towards zero, and maps small negative values toward positive values when outside of the linear range of the function.

6.2 Experimental Methodology

Using the experimental platform created within section 4.8, the effect that the value of the exponent, within the Mackey-Glass non-linear equation, has on the performance and dynamics of a delay-feedback reservoir system can be investigated. In order to test how the system dynamics change with different values of the Mackey-Glass exponent, n, two differently sized reservoir systems are used, one with 20-nodes and one with 200-nodes, with different values of n. The values of n to be tested are $1, 2, 5, 6, 9, 10, 13,$ and 14. These values were chosen as they exhibit the most significant changes within the function shape, as seen in figure 6.1, and the odd and even pairs are chosen to allow comparison of the different attractor behaviours in order to compare the experiments with those from section 5. Each reservoir is given a τ of 80 s and a timescale of 400ms, as this timescale was shown to provide the best results for all benchmarks within chapter 5.

Results from chapter 5 suggest that the input scaling and coupling gain greatly affect the computational performance of a reservoir system. Hence, a parameter sweep of the input scaling and coupling gains is performed for each reservoir type and n value permutation, for both the NARMA-10 and Santa Fe benchmarks as well as the system metrics discussed in section 3.3. Table 6.1 shows the parameter values and ranges for the performed parameter sweeps:

Name	Symbol	Value (20-Node)	Value (200-Node)
Coupling Gain	β	0.05-2	0.05-2
Input Scaling Factor	δ	0.05-2	0.05-2
System Timescale	T	400ms	400ms
Time Delay	τ	80 s	80 s
Masking Signal Period	θ	4 s	0.4 s

Table 6.1 Parameter values of the delay-feedback reservoir Simulink model during the input scaling and coupling gain parameter sweep.

The training and testing of the output, and the methodology for generating the input data of the delay-feedback reservoir, is described in section 4.3.

6.3 Experimental Results

The parameter sweeps display the NRMSE of the reservoir system at each sweep point as a colour spectrum; the lowest calculated NRMSE value is shown as blue, with the highest shown as yellow. As a system may be drawn into saturation or become unstable, the maximum NRMSE value within all of the parameter sweeps is set to 1. Both of the training and testing NRMSE values are plotted as there will likely be differences within the parameter sweep due to the Moore-Penrose pseudo-inverse function calculating the weight matrix, which can sometimes cause an overdetermined or unstable solution.

6.3.1 NARMA-10 Benchmark

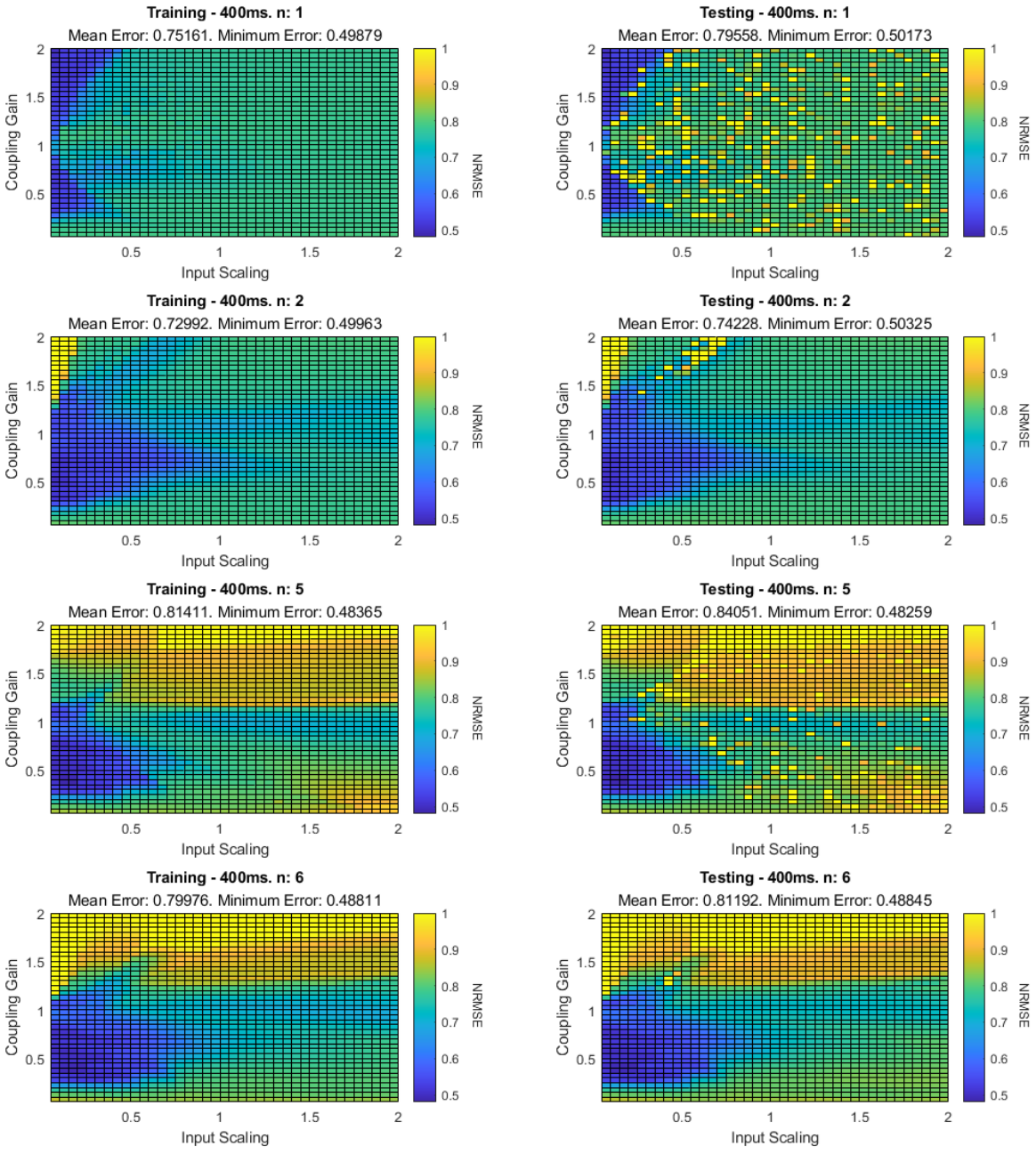

Figure 6.3 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20-node system running the NARMA-10 benchmark, using four different small Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

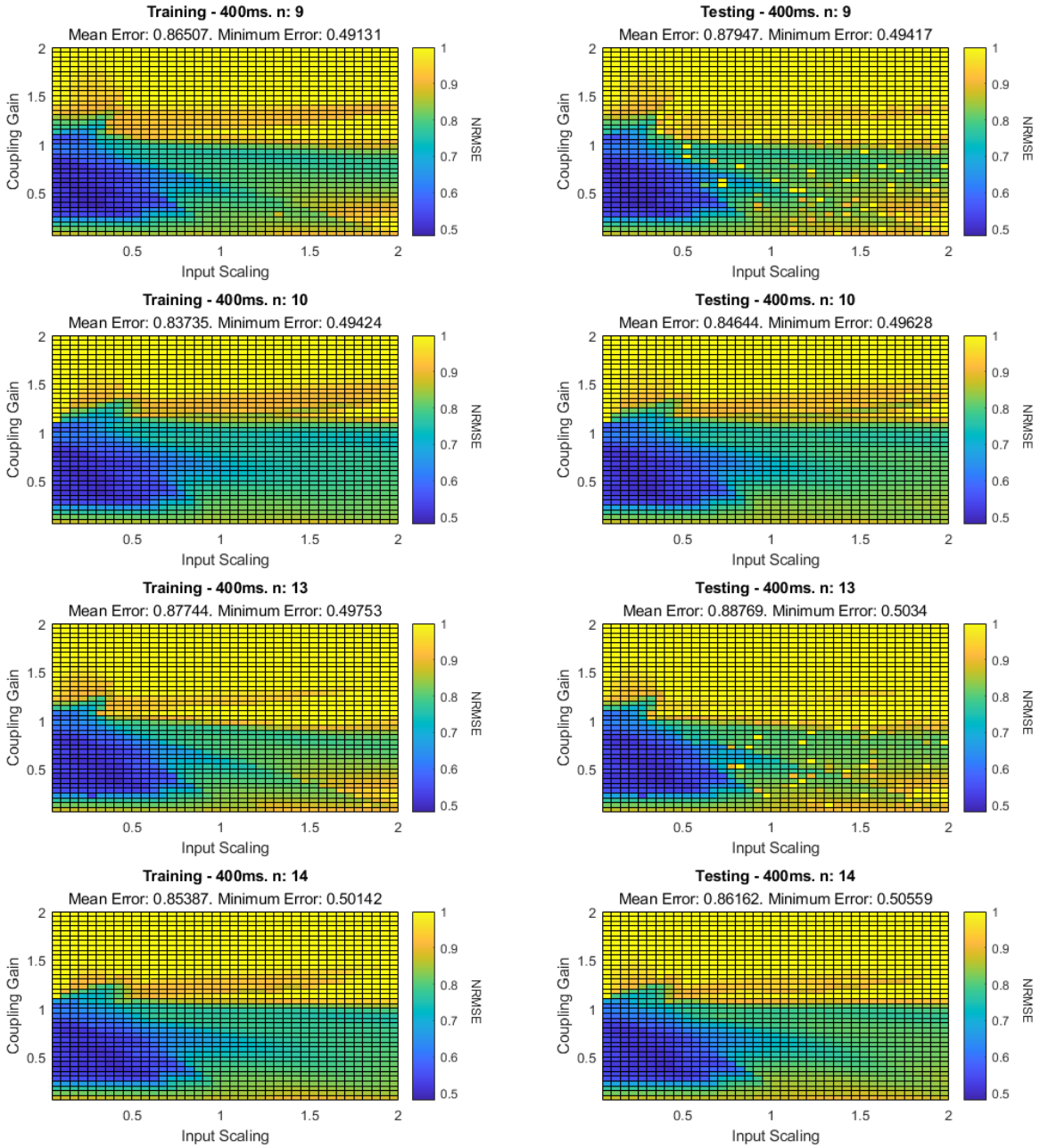

Figure 6.4 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20-node system running the NARMA-10 benchmark, using four different large Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

Mackey-Glass Exponent Value	NRMSE	Input Scaling	Coupling Gain
1	0.502	0.05	2
2	0.503	0.05	0.5
5	0.483	0.15	0.35
6	0.488	0.15	0.4
9	0.494	0.3	0.35
10	0.496	0.3	0.35
13	0.503	0.4	0.45
14	0.506	0.4	0.45

Table 6.2 Table of the best performing sweep parameters within the testing dataset for the 20-node system with the NARMA-10 benchmark for different values of the Mackey-Glass exponent.

20-Node System. Figures 6.3 and 6.4 show the effects of the input scaling and coupling gain on a 20-node delay-feedback reservoir system, evaluating the NARMA-10 computational benchmark, at several different Mackey-Glass exponent values, n.

When the Mackey-Glass exponent is set to values which indicate high non-linearity, as shown in figure 6.3, some interesting observations can be made. First, when the exponent is set to 1, the region of best performance for the NARMA-10 benchmark is small and shows a very high sensitivity to input scaling, showing a reasonable NRMSE value for an input scaling of only 0.05-0.5, and a low sensitivity to the coupling gain. Second, the overall performance of the system is poor, with a maximum training and testing NRMSE value of approximately 0.5, within this small region of best performance. Third, it is likely that this region of best performance has a strong localised Linear Memory Capacity (LMC) but poor Computational Ability (CA) as indicated by the sharp gradient moving from average to poor performance.

A sensitivity to input scaling is to be expected due to the behaviour of the attractor. When the Mackey-Glass exponent is set to a value of 1, the Mackey-Glass non-linear equation has a very small linear region and a large weak attractor area, almost all values within the system, except for the very small values, will be transformed and

pushed towards zero. When the input scaling is below 1, the signal is attenuated so it is less likely to be affected by the attractor behaviour. As the integrator also attenuates, a wider range of coupling gains can be utilised before the system is either negatively affected by the attractor, or before the output of the integrator becomes too large to overpower the attenuated input signal.

As the value of the Mackey-Glass exponent increases, as shown in figure 6.3 and figure 6.4, the shape of the region of best performance exhibits a large change for an n value of 1 and 2, but exhibits only smaller changes as n increases. This once again is most likely due to the behaviour of the attractor and linear region within the non-linear function as n increases.

When the Mackey-Glass exponent, n, increases, the linear region of the non-linear function becomes wider and the attractor behaviour becomes stronger. This larger linear region allows for a larger signal to exist within the feedback loop before being drawn towards the attractor, and the stronger attractor means that it takes fewer iterations around the feedback loop before being pushed towards zero. Together, this leads to a decreased sensitivity to input scaling as the linear mapping increases, but an increased sensitivity to coupling gain as signals are more likely to become larger in value, causing the integrator to saturate. However, if there is a particular dynamic that a computational task requires (such as LMC for the NARMA-10 computational benchmark) within a particular region of the parameter space, then the computational task gains no advantage having a reduced sensitivity to input gain as the dynamics are localised in a particular spot. This would result in no additional performance increase as the value of n increases; which is displayed in figures 6.3 and 6.4.

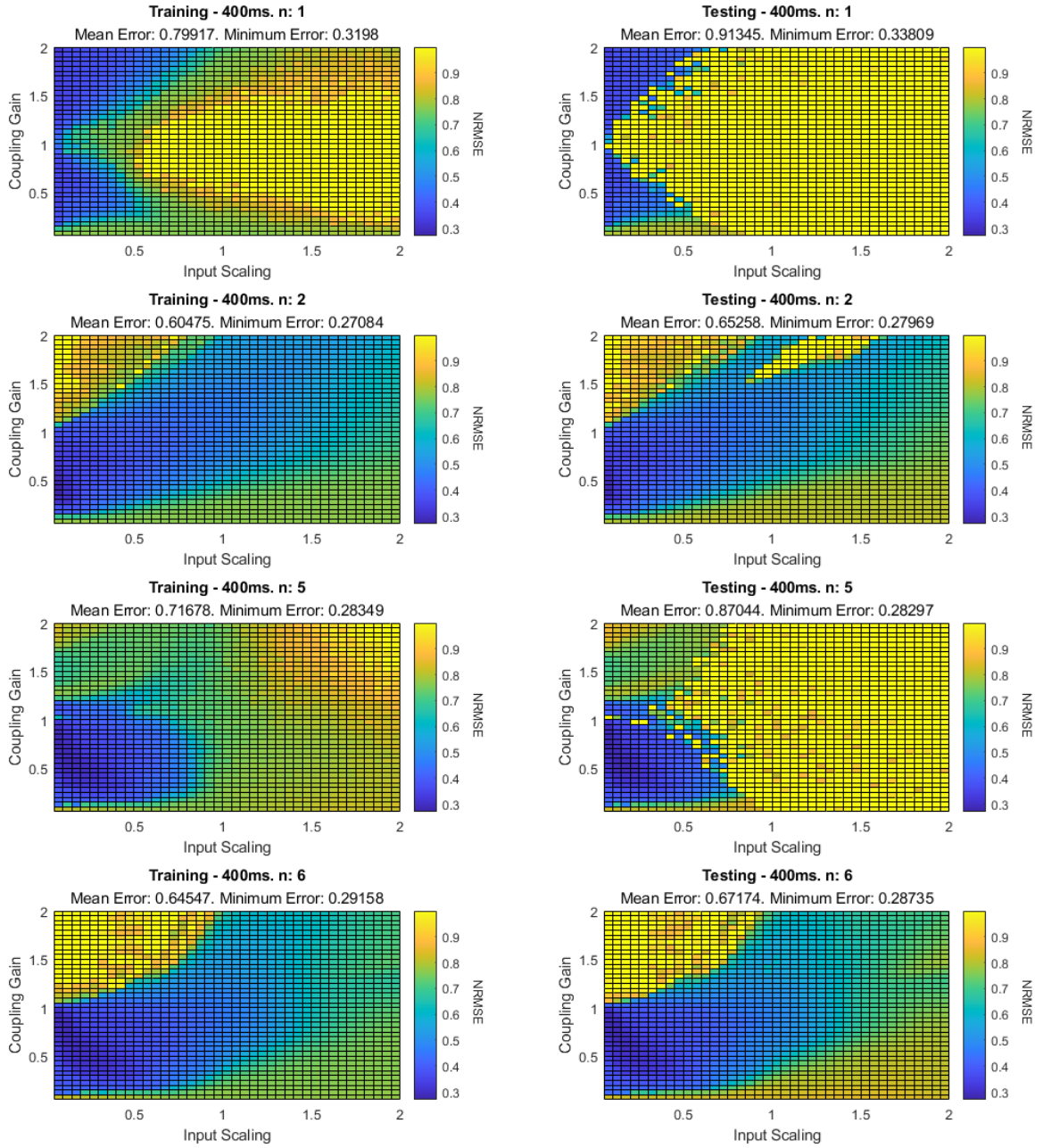

Figure 6.5 Graphs showing the training and testing NRMSE values of a parameter sweep for a 200-node system running the NARMA-10 benchmark, at four different small Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

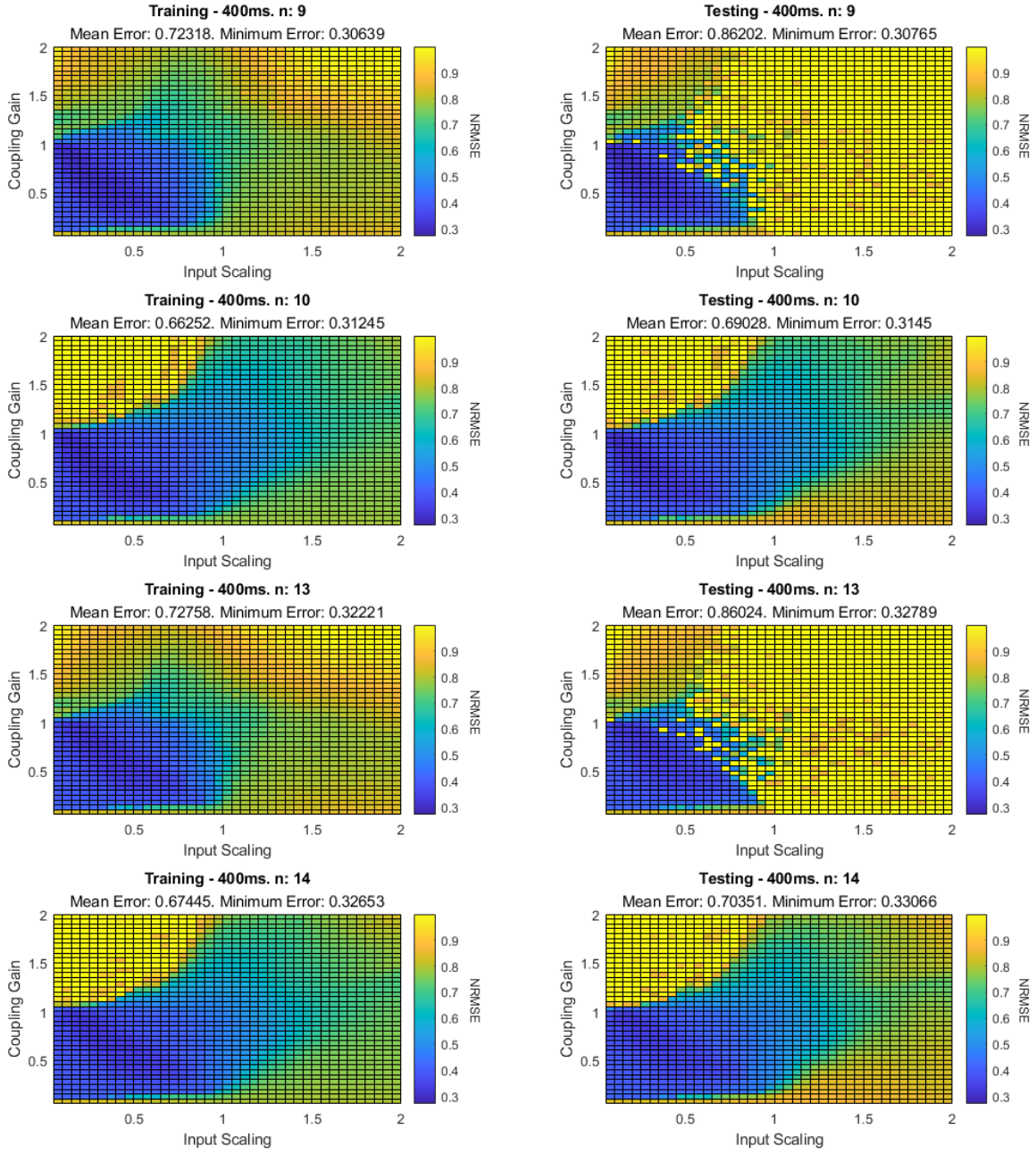

Figure 6.6 Graphs showing the training and testing NRMSE values of a parameter sweep for a 200-node system running the NARMA-10 benchmark, at four different large Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

Mackey-Glass Exponent Value	NRMSE	Input Scaling	Coupling Gain
1	0.338	0.05	2
2	0.28	0.05	0.35
5	0.283	0.05	0.75
6	0.287	0.1	0.7
9	0.308	0.15	0.8
10	0.315	0.15	0.8
13	0.328	0.25	0.8
14	0.331	0.15	0.95

Table 6.3 Table of the best performing sweep parameters within the testing dataset for the 200-node system with the NARMA-10 benchmark for different values of the Mackey-Glass exponent.

200-Node System. Figures 6.5 and 6.6 show the effects of the input scaling and coupling gain on a 200-node delay-feedback reservoir system, evaluating the NARMA-10 computational benchmark, at several different Mackey-Glass exponent values, n. Similar observations can be made across the 20- and 200-node systems in terms of performance and the location of the region of best performance, but there are some key differences as well.

First, the region of best performance is much larger within the 200-node system given the same values of n. This region expansion is to be expected as the 200-node system has a greater number of virtual nodes by an order of magnitude, allowing the 200-node system to achieve richer dynamics, such as a larger region of LMC within the parameter space that the NARMA-10 task can utilise.

Second, for even values of n, the region of best performance spreads diagonally from the origin and has a much more pronounced region of poor operation within the top left of the parameter space. While for odd values, the shape of the region of best performance is similar to the 20-node cases, albeit wider. This behaviour is likely due to the mapping of values within the negative side of the attractor.

When the coupling gain is high and the input scaling is low, information may be lost if the scaled input signal is considerably smaller than the delayed reservoir state vector. This effect can be seen in both figures for the 20- and 200-node systems as a region of poor performance in the top right of the parameter sweeps. An additional effect that occurs within the region is when there is no loss of scaled input data: the signal becomes large enough to be affected by the attractor behaviour of the non-linear function.

With an odd value of n, if a signal exceeds the linear range within the non-linear function then the signal is attenuated towards zero proportional to its magnitude. Whereas with even values of n, only large positive signals are attenuated towards zero, with large negative signals being mapped to attenuated positive values, and small negative signals that exceed the linear range of the non-linear function being mapped to large negative numbers; this is due to the asymptotic behaviour of the negative side of the attractor, as discussed in section 6.1.1. The mapping of negative values to positive allows for some information to be retained, while with odd values of n, information would be simply lost.

6.3.2 Santa Fe Benchmark

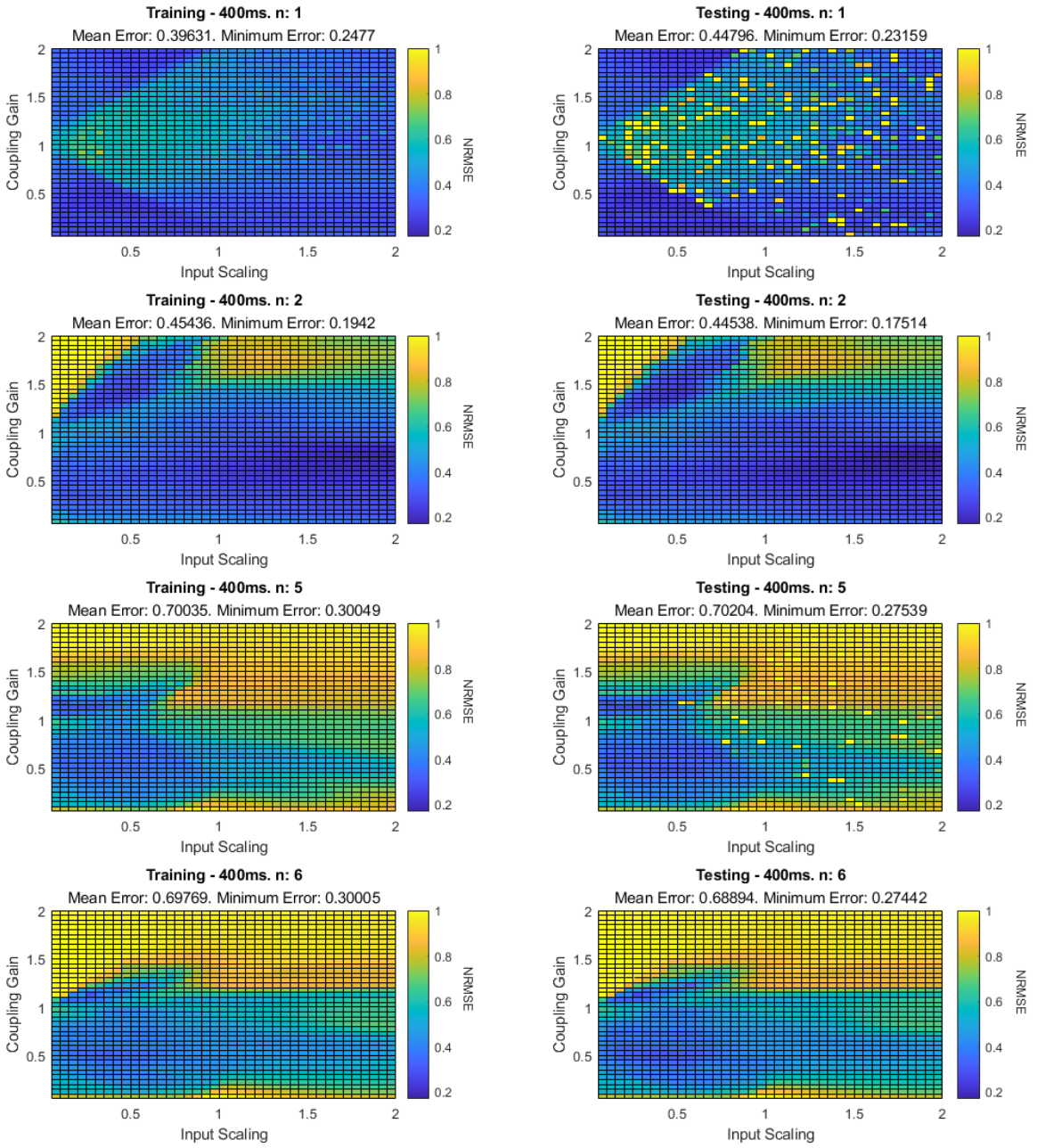

Figure 6.7 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20-node system running the Santa Fe benchmark, using four different small Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

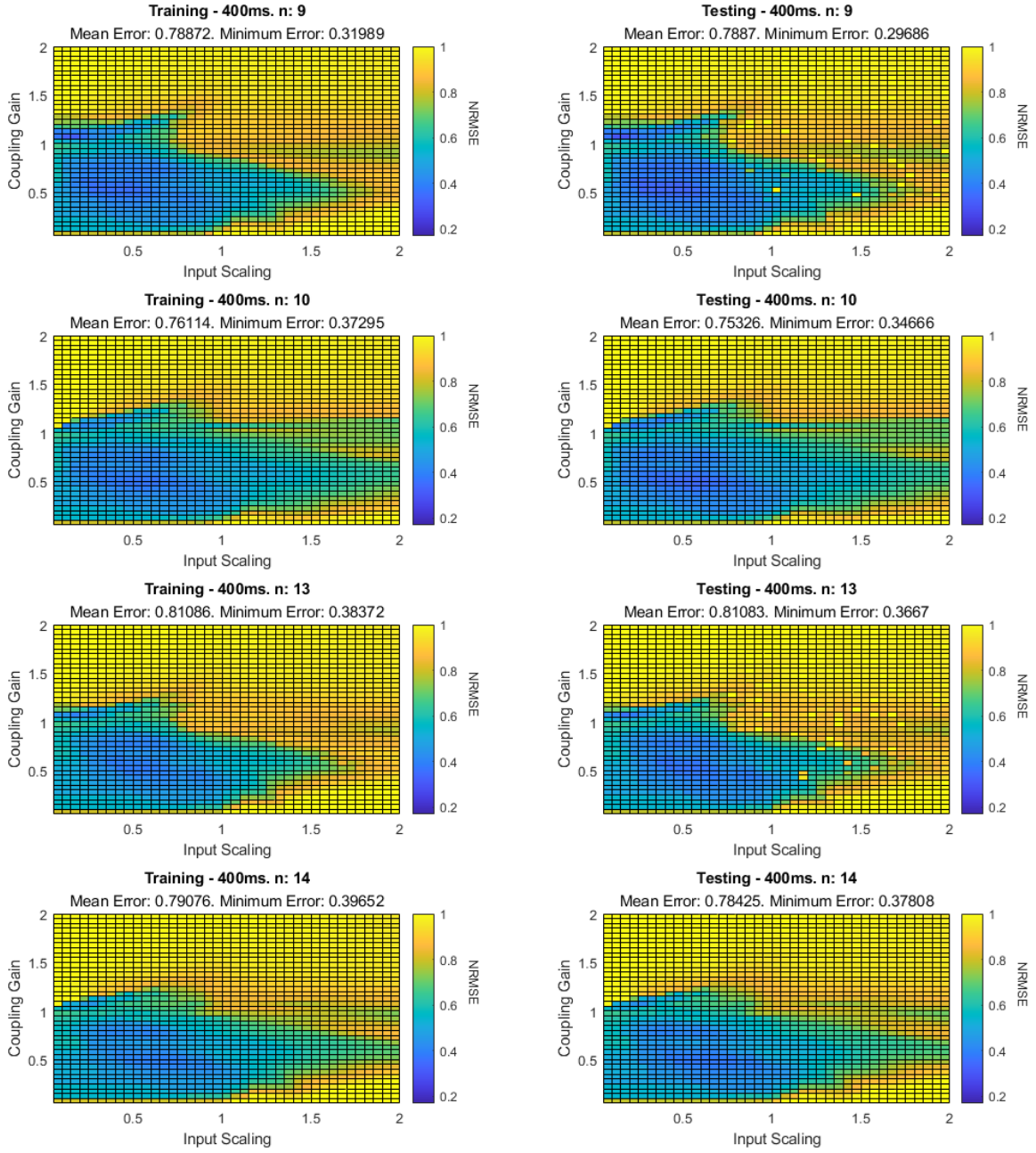

Figure 6.8 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20-node system running the Santa Fe benchmark, using four different large Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

Mackey-Glass Exponent Value	NRMSE	Input Scaling	Coupling Gain
1	0.232	0.6	0.3
2	0.175	2	0.75
5	0.275	0.25	1.1
6	0.274	0.25	1.1
9	0.297	0.15	1.05
10	0.347	0.6	0.45
13	0.36	0.15	1.05
14	0.378	0.75	0.45

Table 6.4 Table of the best performing sweep parameters within the testing dataset for the 20-node system with the Santa Fe benchmark for different values of the Mackey-Glass exponent.

20-Node System. Figures 6.7 and 6.8 show the effects of the input scaling and coupling gain on a 20-node delay-feedback reservoir system, evaluating the Santa Fe Laser computational benchmark, at several different Mackey-Glass exponent values, n.

Like in chapter 5, where the timescale of the delay-feedback reservoir was tested, there is little in common between the NARMA-10 and Santa Fe benchmark tasks within the 20-node reservoirs; with the exception of when n is equal to 1. The region of best performance for the Santa Fe benchmark is much wider, exhibiting a larger distribution of test errors with fair performance across the entire input scaling parameter range. The presence of two regions of best performance still exist, a large region within the bottom left of the parameter space, and a smaller region in the middle left of the parameter space, just above the line where the coupling gain is 1. As the value of n increases, the operating region recedes, becoming more sensitive to both input scaling and coupling gain. This receding behaviour initially happens quickly as n increases to 5, specifically the sensitivity to coupling gain, but appears to recede much slower after n reaches 6. The second smaller operating region also becomes less defined, almost disappearing when n is equal to 14; which is most likely due to the loss of strong non-linear dynamics within this area as n increases.

Mackey-Glass Exponent Value	NRMSE	Input Scaling	Coupling Gain
1	0.0913	0.6	0.3
2	0.0868	2	1
5	0.092	0.7	1.2
6	0.134	1.5	1.2
9	0.103	0.55	1.1
10	0.171	0.35	0.95
13	0.128	0.6	1.1
14	0.193	0.85	0.85

Table 6.5 Table of the best performing sweep parameters within the testing dataset for the 200-node system with the Santa Fe benchmark for different values of the Mackey-Glass exponent.

One of the most significant differences between the NARMA-10 and Santa Fe benchmarks is that as n increases, the performance decreases; with the exception of when n is equal to 1. Given that the Santa Fe benchmark requires strong non-linear dynamics within a reservoir to achieve its best performance, it would be logical to assume that the performance would decrease as the system becomes more linear as n increases. However when n is equal to 1, the minimum test error is larger than when n is equal to 2, but exhibits a much smaller mean error as the overall sensitivity to parameters has greatly decreased. This is similar to what was observed within the 20-node NARMA-10 benchmark parameter sweep (figure 6.3) in section 6.3.1, but with a much wider range. This is likely due to the Santa Fe benchmark utilising more of the non-linear dynamics within the reservoir system when n is equal to 1, which the NARMA-10 benchmark was unable to utilise due to it only requiring a small amount of non-linearity; therefore the additional non-linearity would not lead to an increase in performance.

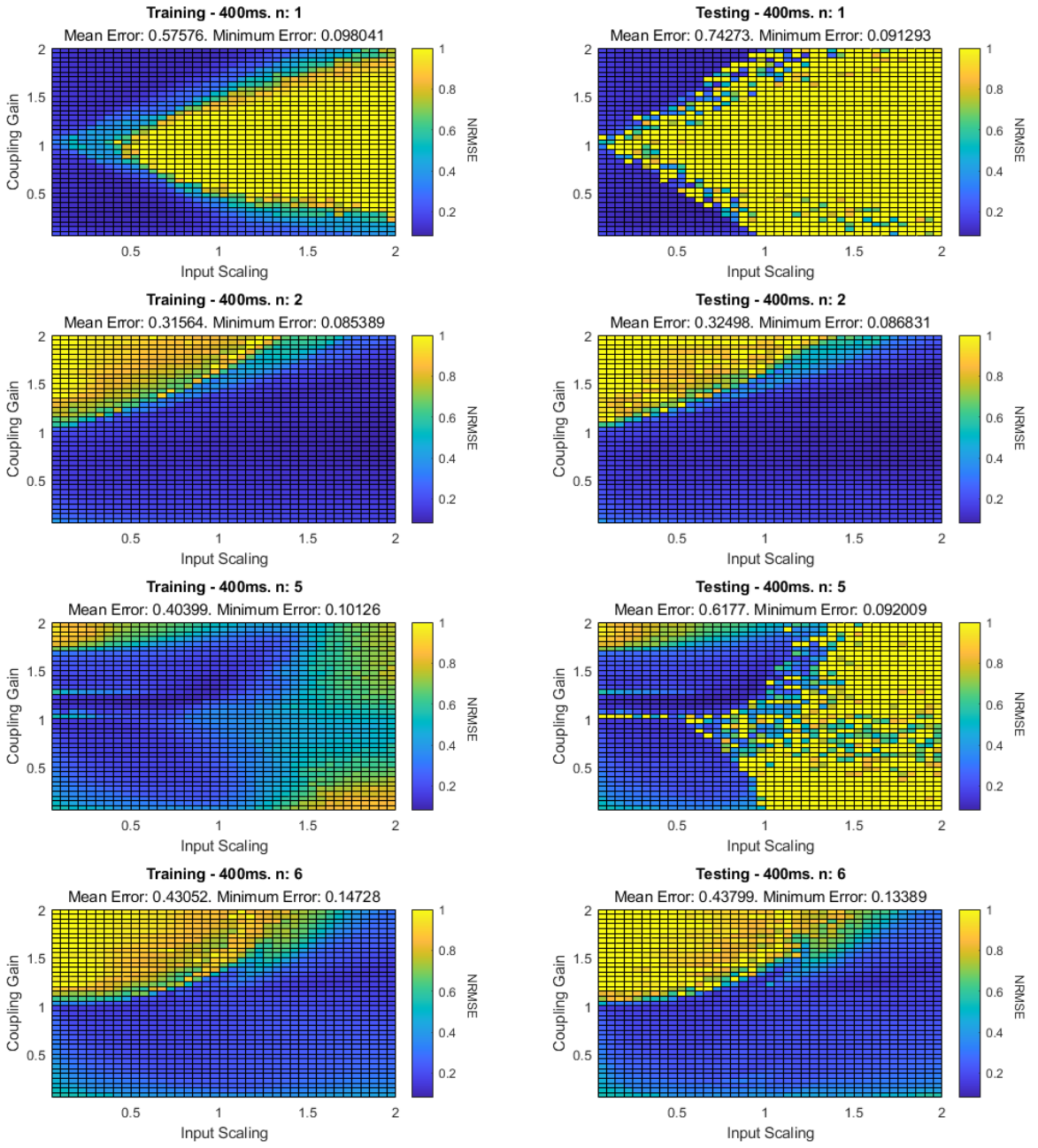

Figure 6.9 Graphs showing the training and testing NRMSE values of a parameter sweep for a 200-node system running the Santa Fe benchmark, using four different small Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

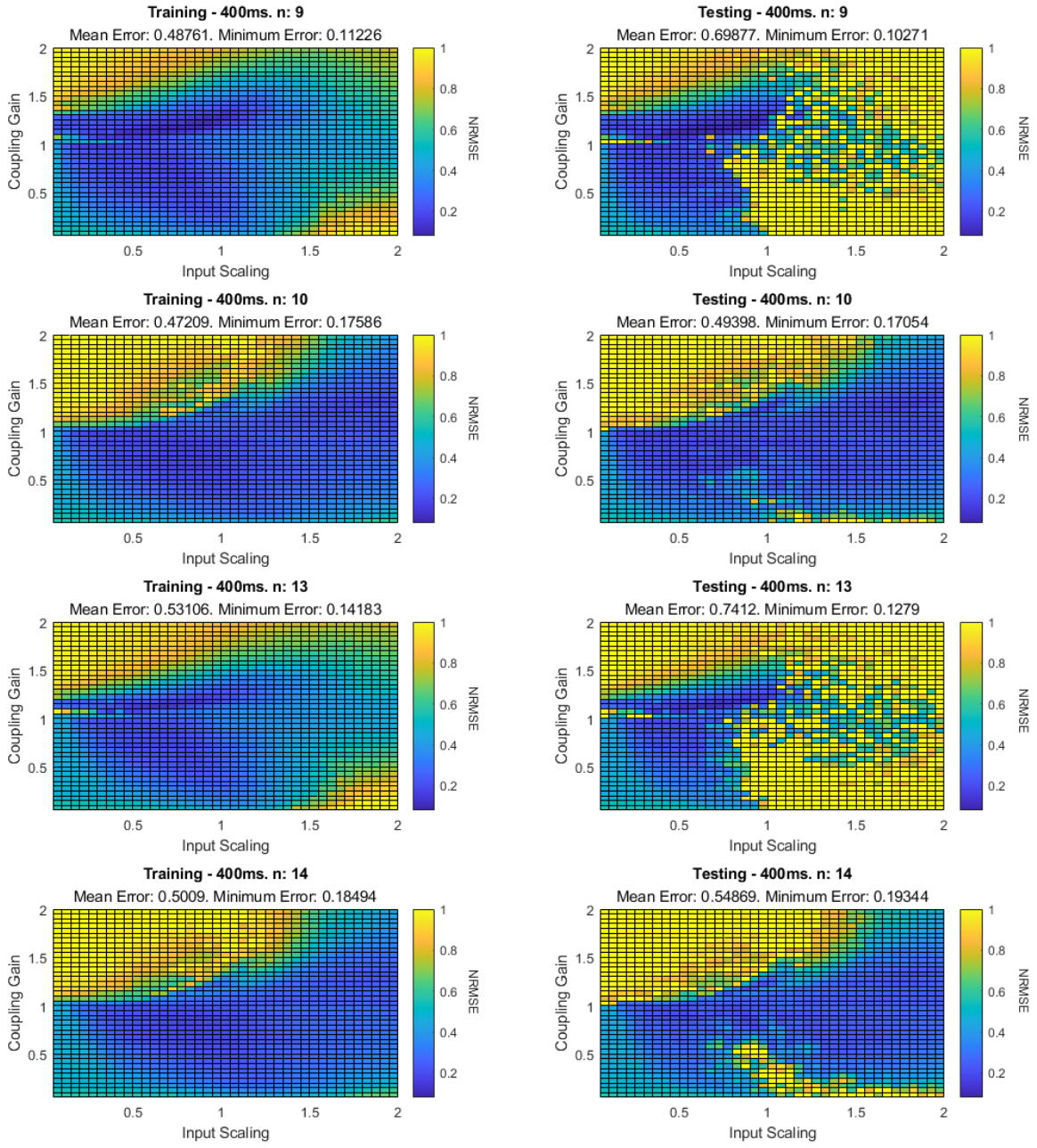

Figure 6.10 Graphs showing the training and testing NRMSE values of a parameter sweep for a 200-node system running the Santa Fe benchmark, using four different large Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2.

200-Node System. Figures 6.9 and 6.10 show the effects that input scaling and coupling gain have on a 200-node delay-feedback reservoir system, evaluating the Santa Fe Laser computational benchmark at several different Mackey-Glass exponent values, n.

While the effect of having an odd or even Mackey-Glass exponent value was subtle within the 20-node system, this effect is much more pronounced within the 200-node system.

When the value of n is small and even, there is a very wide region of best performance with virtually no sensitivity to input scaling, and a moderate sensitivity to coupling gain. The NRMSE values are small throughout most of the region of best performance, with no localised region of peak performance. As the even value of n increases, the sensitivity to input gain increases, with the distribution of low NRMSE values becoming more concentrated, creating a noticeable localised region of peak performance around the centre of the parameter space. What appears to be task agnostic is the pronounced region of poor operation, located within the top left of the parameter space which is present when n is even within the 200-node reservoir systems running the NARMA-10 and Santa Fe computational benchmarks. Given that the region becomes more prominent as the even value of n increases, this must be a property of the negative side of the attractor as discussed previously.

When the value of n is above 1 and odd, the reservoir system exhibits two regions of best performance; one in the bottom left and the other in the top left region of the parameter space. Unlike previously, the top region of best performance is larger than previously observed, with both regions having a low sensitivity to coupling gain and an input scaling sensitivity approximately between 0.05 - 1.2. The best performing NRMSE results are within the upper operating region, as seen within the other Santa Fe benchmark parameter sweeps within chapter 5.3.2. However as the odd value of n increases, the upper region of best performance becomes smaller; which is once again

due to the difference in signal sizes between the scaled input signal and the delayed reservoir state vector.

When the value of n is set to 1, the reservoir exhibits a similar region of best performance, as observed within the 20-node Santa Fe reservoir system and the 20- and 200-node NARMA-10 reservoir systems. Given that the shape of this region is different when n has increased and is task agnostic, this strongly indicates that there is a region of instability inherent to this non-linearity. This can be explained by referring to the shape of the non-linear function when n equals 1 in figure 6.1. Within this figure, it can be observed that there is a very small linear region near the origin, therefore any data exceeding this linear range is affected by the attractor. Given the nature of the negative side of the attractor, any signal that becomes slightly negative is drawn towards the asymptote and then mapped into the positive region. This may take several iterations around the feedback loop as the attractor behaviour is weak.

Another observation, with the exception of n being equal to 1, is that as the value of n increases, so do the NRMSE values of the reservoir, resulting in decreased performance when solving the Santa Fe benchmark; which mirrors what is found within the 20-node case. However, unlike the 20-node case, odd values of n perform significantly better than even values. The performance gain in odd values of n is most likely due to the lack of the upper region of best performance observed within even cases. Whereas the general decrease in performance as n increases, for both odd and even cases, is likely due to the decreased dynamics and shrinkage within the upper region of best performance within the even cases; as the best performing NRMSE results were found within this region.

6.3.3 System Metrics

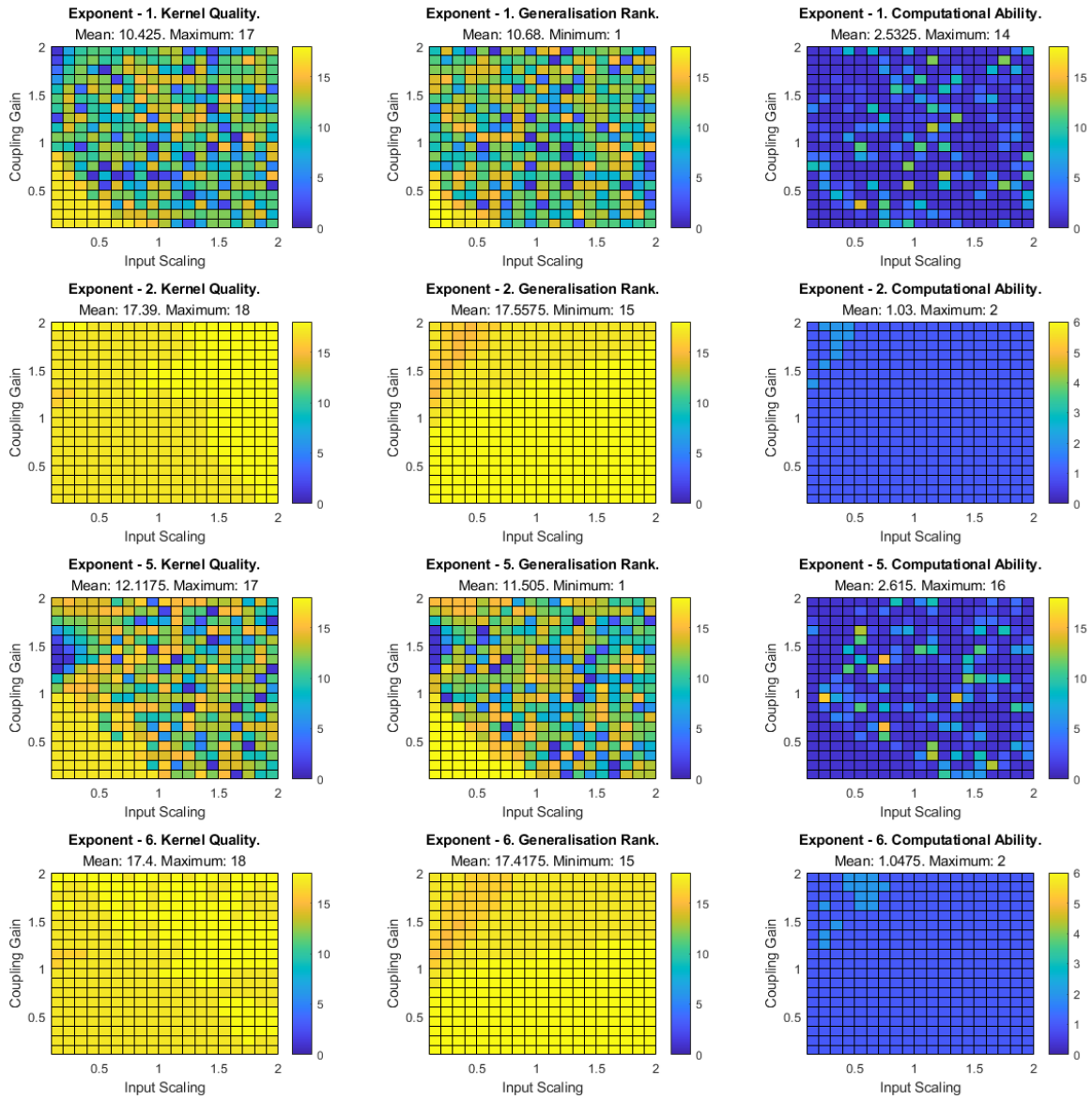

Figure 6.11 Graphs showing a parameter sweep for a 20-node system, at four different high non-linear Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2, for the following the system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

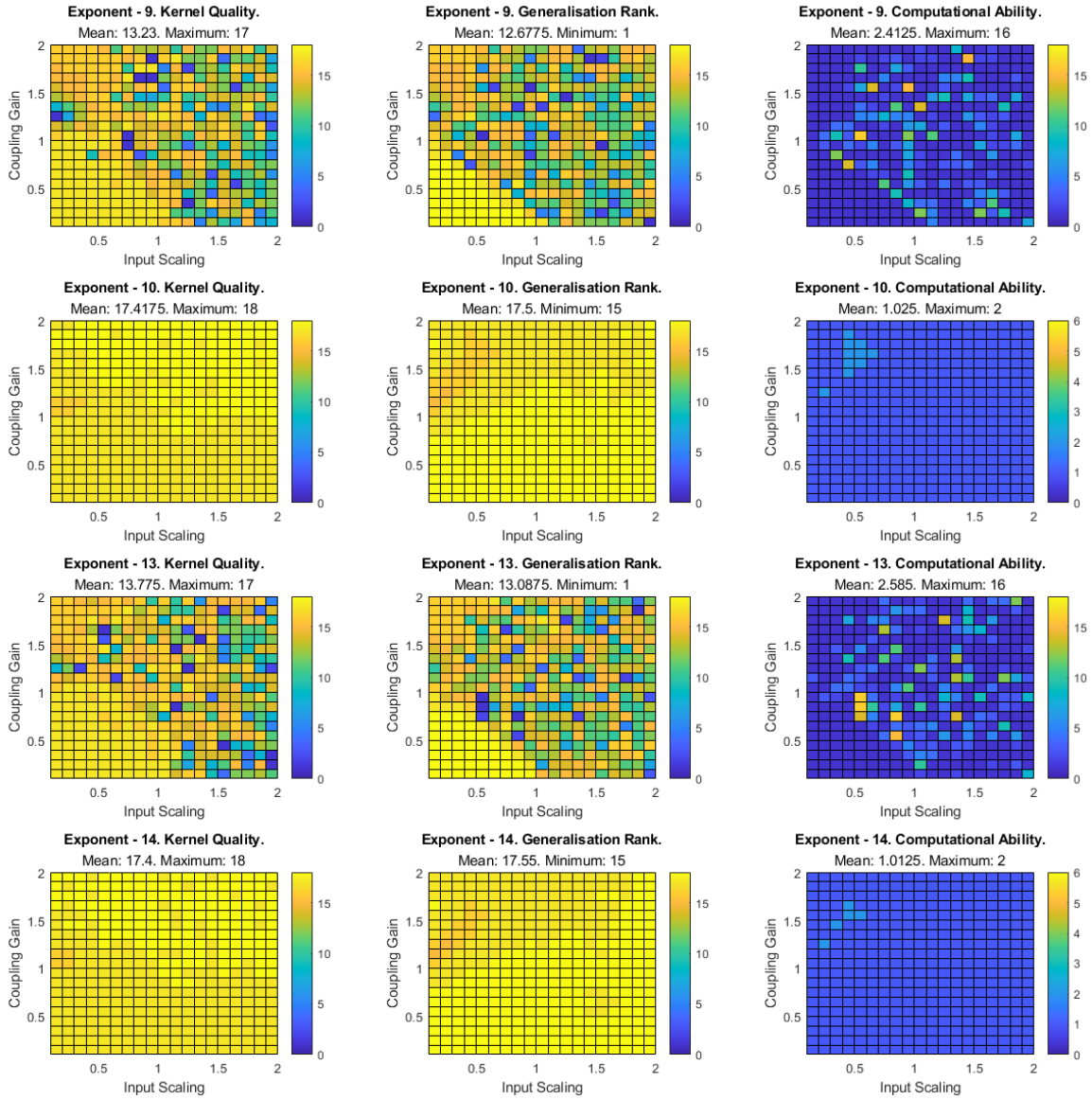

Figure 6.12 Graphs showing a parameter sweep for a 20-node system, at four different low non-linear Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2, for the following the system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

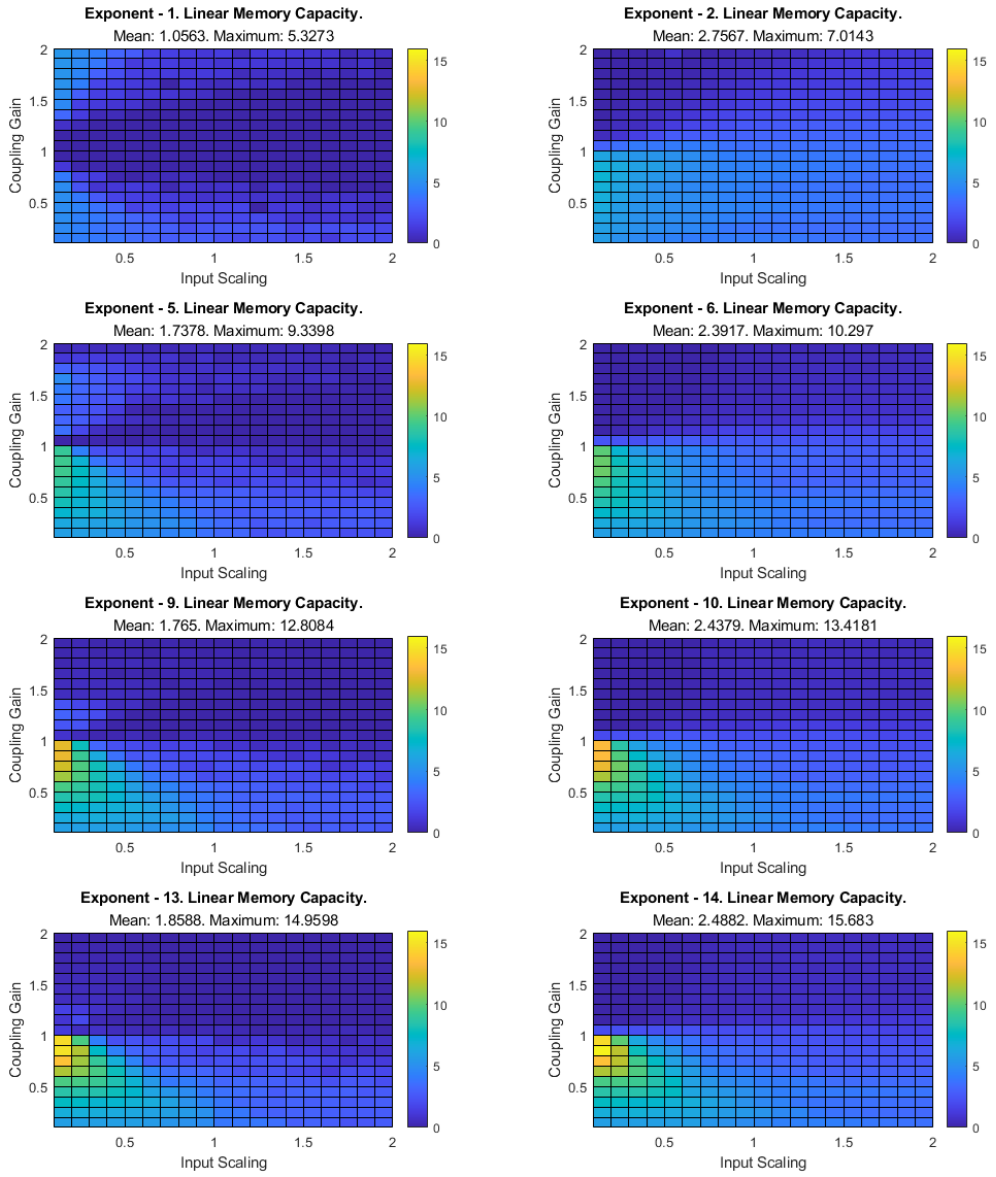

Figure 6.13 Graphs showing a parameter sweep for a 20-node system, ran with eight different Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2, for the system metric Linear Memory Capacity.

20-Node System. Figures 6.11 and 6.12 show the Kernel Quality (KQ), Generalisation Rank (GR), and Computational Ability (CA), while figure 6.13 shows the Linear Memory Capacity (LMC) of a 20-node delay-feedback reservoir system during a parameter sweep of the input scaling and coupling gain. Before the comparison between system metrics and computational benchmarks are made, the effect that the Mackey-Glass exponent has on the system metrics is first determined.

System Metric Evaluation. Figures 6.11 and 6.12 show very different dynamics for the odd and even values of n. A significant difference is to be expected given the large variations observed throughout the benchmark parameter sweeps.

For even values of n, both the KQ and GR exhibit a relatively consistent pattern as the value of n increases. The value of KQ appears to be high throughout all of the parameter sweeps, showing only a slight sensitivity to coupling gain as the value of n increases; indicating that the system is able to map input vectors onto a high-dimensional non-linear space. However, the GR parameter sweep shows a minor sensitivity to parameter changes, but indicates that the system's ability to generalise input vectors is poor within the majority of the parameter space; with the expectation of the top left area of the parameter sweep. The distribution of the values of KQ and GR were fairly uniform throughout all of the parameter sweeps, with the range between the minimum and maximum values for KQ and GR being 2 and 3 respectively. This can be seen within the CA parameter sweep, with the even values of n showing a small region in the upper right exhibiting weak CA.

Although the CA of the even systems indicate only a small local area of good performance, it is important to consider KQ and GR independently, as a computational task can utilise certain system metrics independently of others.

In the case of odd values of n, the parameter plots for KQ and GR appear to be much more sporadic, indicating richer dynamics. These richer dynamics are most likely due to the negative section of the attractor when n is odd. The signal mapping of large negative numbers being drawn towards the asymptote, and then mapped into

the positive region, gives the system a much stronger non-linear behaviour than for even values of n. These stronger behaviours can be observed within the CA parameter sweeps, with the maximum value of CA significantly exceeding that of the even values of n. Another interesting observation within the CA is that as the odd values of n increases, the distribution takes the shape of a cone with low CA regions in the top and bottom left of the parameter sweeps. The mean of the CA only increases slightly as n increases, implying the distribution is being drawn towards a cone shaped region exhibiting good CA.

A key observation, for both even and odd values of n, is that although the shape of the parameter plots do change with an increase of n, the values of KQ and GR seem to be independent of the value of n. This initially is a surprising result as the Santa Fe benchmark performed significantly better with lower values of n, it was expected that an increase in n would lead to a lower KQ and a higher GR. However, on reflection, these parameter sweeps show the true behaviour of the Mackey-Glass non-linear function.

The shape of the Mackey-Glass non-linear function only really changes in three cases: when n is odd, even, or equal to 1. Increasing the n value increases the width of the linear space around the origin and increases the attractor strength. This means that the non-linear transform is not dependent on the value of n, only if it is odd or even. The exception to this is when n is equal to 1, where the shape of the function is inherently unstable; giving a system different non-linear dynamics to when $n > 1$.

In figure 6.13, the relationship between the value of the Mackey-Glass exponent and the LMC shows that as the value of n increases, so does the maximum memory capacity. This result is to be expected as the greater the value of n, and the wider the linear region within the non-linear function is, the more information can be passed between virtual nodes as the connection strength increases.

Unlike within the KQ and GR parameter sweeps, the LMC shows a minor difference between odd and even values of n. For odd values of n, a second region of memory

appears in the middle left side of the parameter space, but decreases in size as n increases. However for even values of n, there is only a single region of strong LMC, but it appears to be less sensitive to input scaling. This behaviour is due to the shape of the non-linear function for odd and even values of n; as previously discussed in sections 6.3.1 and 6.3.2.

The NARMA-10 Benchmark. The region of best performance within the 20-node system running the NARMA-10 benchmark, as shown in figures 6.3 and 6.4, are now compared with the system metrics computed within this section; showing that the region where NARMA-10 performs best matches with the regions of strongest LMC within the LMC parameter sweep. The region of best performance of the NARMA-10 benchmark is only slightly influenced by the KQ and GR as the region of best performance is within areas of relatively low KQ and high GR. This is to be expected as NARMA-10 has a heavy reliance on memory and weak reliance on the non-linearity of the reservoir system.

The Santa Fe Benchmark. The region of best performance within the 20-node system running the Santa Fe benchmark, as shown in figures 6.3 and 6.4, is compared with the system metrics computed within this section.

There are two regions of best performance where the 20-node Santa Fe benchmark performs well, a large area within the bottom left of the parameter space (referred to as the lower region of best performance) and a smaller but better performing area within the middle left of the parameter space (referred to as the upper region of best performance). The upper region of best performance correlates strongly to the KQ and GR plots, as the upper region of best performance is placed in the locations where KQ is high, and where GR is low; this is shown more clearly within the CA plot. The lower region of best performance correlates stronger to areas with high LMC rather than KQ or GR; most noticeably within the odd cases of n, where the lower region of best performance is much wider due to the greater CA within the parameter space. This result is to be expected given the required dynamics for optimal performance

to perform the Santa Fe computational benchmark; requiring high non-linearity and low-to-moderate memory.

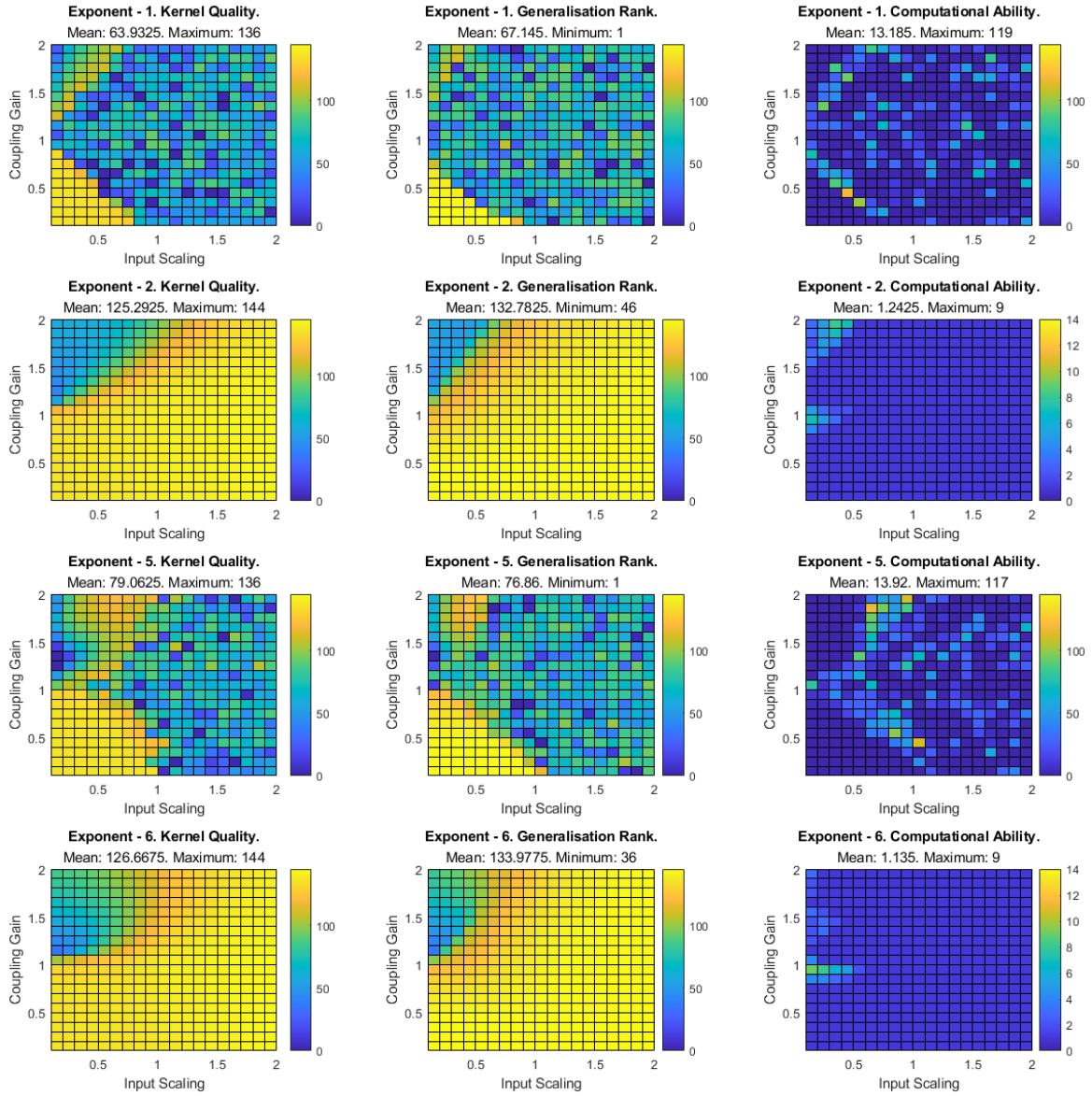

Figure 6.14 Graphs showing a parameter sweep for a 200-node system, at four different high non-linear Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2, for the following system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

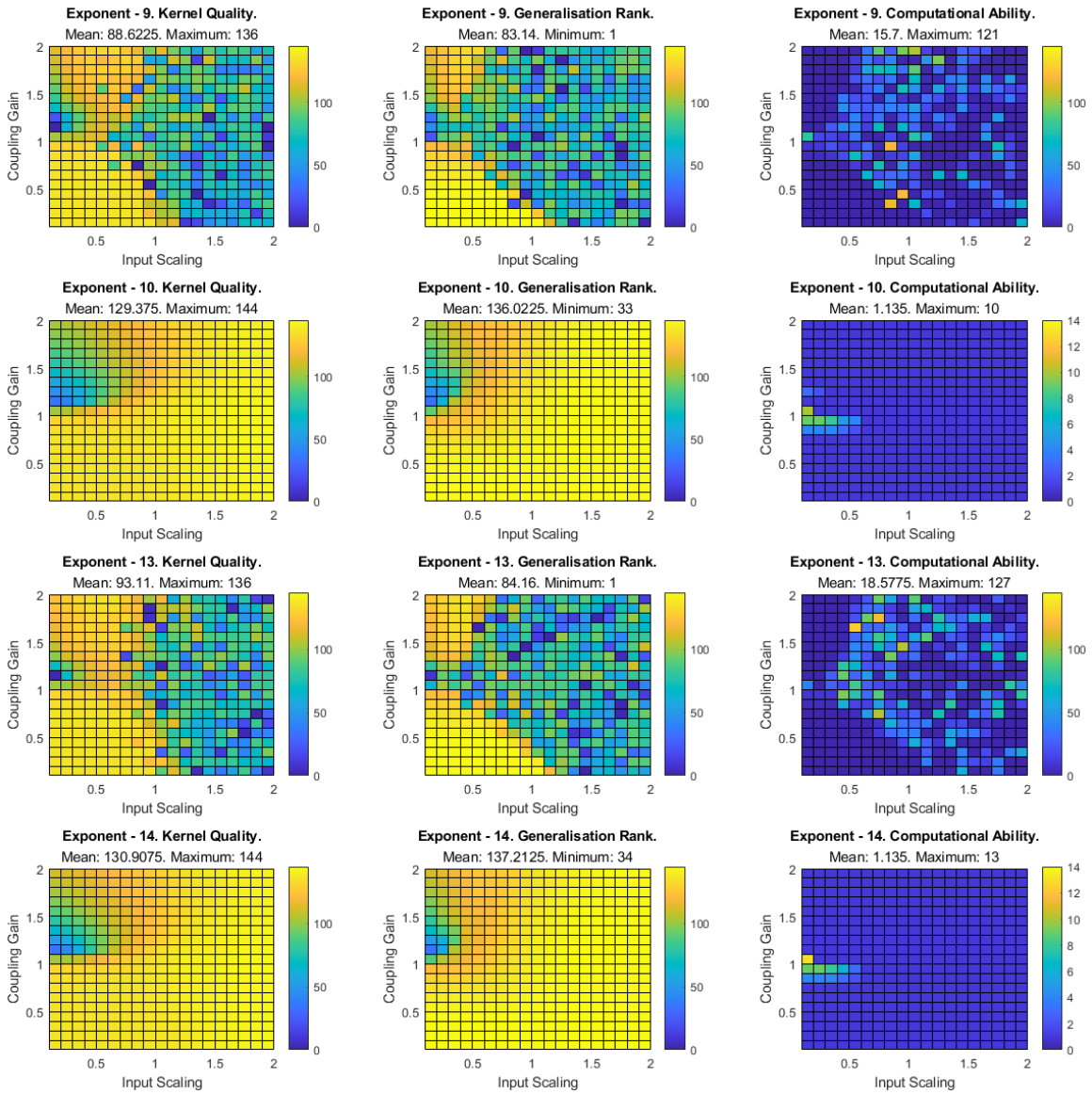

Figure 6.15 Graphs showing a parameter sweep for a 200-node system, at four different low non-linear Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2, for the following system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

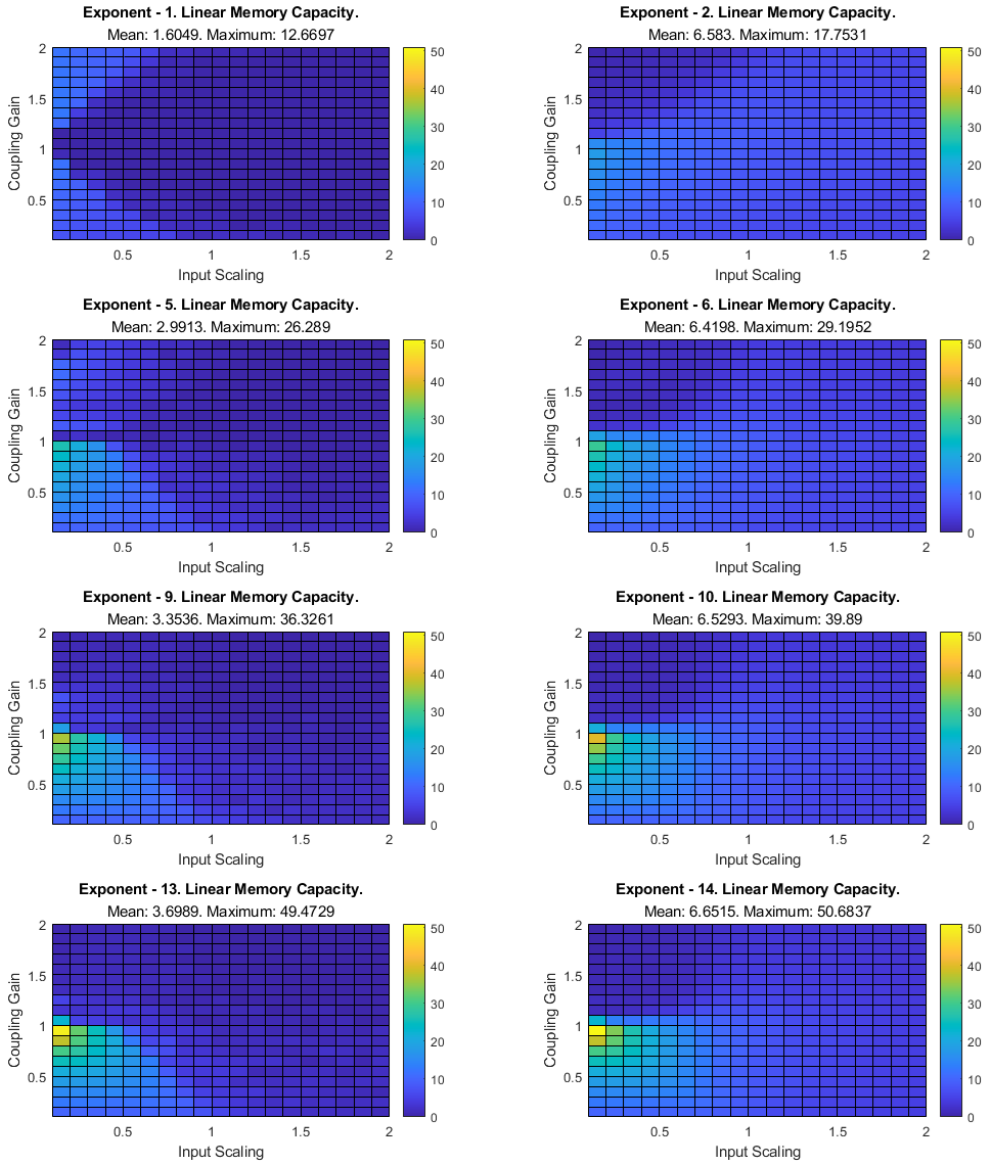

Figure 6.16 Graphs showing a parameter sweep for a 200-node system, ran with eight different Mackey-Glass exponent values, with the input scaling and coupling gain ranging between 0.05 and 2, for the system metric Linear Memory Capacity.

200-Node System. Figures 6.14 and 6.15 show the KQ, GR, and CA, while figure 6.16 shows LMC of a 200-node delay-feedback reservoir system during a parameter sweep of the input scaling and coupling gain. Before any comparison between the system metrics and computational benchmarks are made, the system metrics for the 200-node system are first evaluated, and then compared to the results of the 20-node system.

System Metric Evaluation. Many of the same observations can be made within the 200-node system as with the 20-node system.

Just as with the 20-node system metrics, there are significant differences in the distribution of KQ and GR between odd and even values of the Mackey-Glass exponent, n. As the value of n increases, there is virtually no change within the maximum and minimum values of KQ and GR for both the even and odd cases of n, however the mean does increase slightly.

When comparing the previous 20-node system metrics to the 200-node system metrics, two key observations can be made: First, the mean and maximum values of the KQ, GR, and CA have greatly increased. This is due to having an order of magnitude more virtual nodes within the system, which allows the reservoir system to achieve greater dynamics by allowing an increase in the theoretical maximum possible value for the system metrics by a factor of 10. Second, the distribution of KQ and GR is similar to the 20-node system metrics, but with the GR and KQ in the odd cases of n having a higher sensitivity to input scaling than the previous system. The most significant difference in the distribution of KQ is the area of poor performance in the top left of the parameter space within the even 200-node system. This region of poor performance was noticed previously, and is present within both the 200-node NARMA-10 and Santa Fe benchmark tasks. This region of poor performance, which is most likely due to loss of information from the difference in signal sizes between the scaled input signal and delayed reservoir state vector, is confirmed within the system metrics as a region where the system is unable to map input vectors onto a high-dimensional non-linear space.

There is little difference between the LMC parameter sweeps for the 200-node system, shown in figure 6.16, and the 20-node system; with the only exception being that the mean and maximum values for the LMC are significantly larger in the 200-node system due to the larger number of virtual nodes. The same proportional relationship between LMC and the value of n, and the distribution of LMC values, can be observed for both the 20- and 200-node parameter sweeps.

The NARMA-10 Benchmark. Comparing the region of best performance of the 200-node system running the NARMA-10 benchmark and the system metrics of the 200-node system leads to many of the same conclusions that are drawn for the 20-node system.

The region of best performing NRMSE values correlate to the areas with the highest memory within the LMC system metrics. This can be seen by observing the distribution of LMC values within the parameter sweep plots for the even values of n, which are less sensitive to input scaling and coupling gain than the odd values, having a diagonal component to its distribution which is visible within the shape of the region of best performance within the NARMA-10 benchmark.

An interesting observation is that the KQ does have an effect on the NARMA-10 benchmark, as the region of poor KQ performance shown in the top left of the parameter space negatively impacts the performance of the benchmark. With the KQ being so low, the reservoir is unable to map the input vectors onto a high-dimensional non-linear space, therefore this becomes the dominant metric in terms of performance as the reservoir system is unable to recreate the input-output dynamics of the NARMA-10 computational benchmark.

The Santa Fe Benchmark. The region of best performance within the 200-node system running the Santa Fe benchmark, as shown in figures 6.9 and 6.10, is compared with the 200-node system metrics computed within this section. Many of the conclusions discussed when comparing the 20-node systems can be applied to the 200-node sections; with a few exceptions.

The effect that the KQ has on the region of best performance within the 200-node Santa Fe benchmark task is much more prominent than in the 20-node system; with the distribution of KQ closely matching the regions of best performance within the parameter sweeps, for all values of n. For the even values of n, the upper and lower regions of best performance are visible within the KQ plots, and for the odd values of n, the wider lower region of best performance is visible.

The reason why the upper region of best performance has the best NRMSE values are clear when comparing the Santa Fe parameter sweeps to the system metric parameter sweeps. The upper region of best performance strongly correlates with CA and medium-to-low LMC, while the lower region of best performance strongly correlates with medium-to-low CA and high GR. This once again confirms the desired system metrics for optimal performance when running the Santa Fe benchmark task.

6.4 Discussion

From the results shown in section 6.3, some interesting conclusions can be drawn regarding: the relationship between the Mackey-Glass exponent and the number of virtual nodes within a system; the differences between the odd and even values of the Mackey-Glass exponent; the region of best performance for computational benchmarks; and the effectiveness of the Mackey-Glass non-linear function. This section explores these conclusions further, so that sub-hypotheses 2 and 3 can be tested.

6.4.1 Mackey-Glass Exponent vs Virtual Nodes

The results within section 6.3 show that increasing the Mackey-Glass exponent, n, has little effect on the KQ and GR, with a moderate effect on the LMC of the simulated reservoir systems. Increasing the amount of virtual nodes within the system did have some effect within the KQ, GR, and LMC, as an increase in virtual nodes also increases the theoretical maximum KQ, CA, and LMC of a system. This allows

the operating range to become generally larger, as the system becomes less sensitive to input scaling and coupling gain, therefore it would be expected for the 200-node system to outperform the 20-node system.

It is initially expected that the maximum KQ would decrease as n increases, given that the shape of the non-linear function is modified. However, it is found that the Mackey-Glass non-linear function does not change the non-linearity of the system by simply increasing the value of n. As the value of n increases, the linear region within the non-linear function becomes wider, and the attractor strength at either end of the linear region becomes stronger. The increase in the linear region and attractor strength does not increase the non-linearity of the system, just the sensitivity to system parameters, particularly input scaling.

6.4.2 Mackey-Glass Operating Modes

It is observed within section 6.3 that increasing the Mackey-Glass exponent, n, has little effect on the system metrics within a particular system. While this is true, one observation which sees a huge effect within the system metrics is whether the value of n is odd or even.

During the analysis of the parameter space of the NARMA-10 and Santa Fe computational benchmarks, it is found that a system has very different behaviours, in terms of region of best performance and distribution of system metrics, when n is odd, even, or set to a value of 1. When the value of n is even, there is a region of poor operation within the top left area of the parameter space which is universal in all benchmark tasks and system metrics, with a reduced sensitivity to both input scaling and coupling gain. However, when the value of n is odd, it is common for an additional region of best performance to appear within the top left area of the parameter space, in a similar location to the region of poor performance for even values of n, and shows a greater sensitivity to coupling gain.

The shape of the Mackey-Glass non-linear function when n is even exhibits a linear region at the origin of the function, with a positive and negative attractor at both ends of the linear region to attenuate values exceeding the linear range, and map the attenuated values towards 0. When operating within a region with a high attenuation of input signal gain but a large amplification of the delayed feedback, two effects can occur. First, the integrator is likely to become saturated as the output is always amplified, and second, a significant loss of new input information will be lost if the input signal is significantly smaller than the delayed feedback signal. In a non-linear function where outlying values are simply attenuated and forced towards 0, these effects are likely to occur. However, in the case of an odd value of n, the shape of the Mackey-Glass non-linear function also exhibits a linear region at the origin of the function, but has a positive attractor that attenuates values and maps them towards 0, but within the negative attractor, values are drawn towards the asymptote and then mapped into the positive region. This remapping of negative values to positive allows for information to be maintained within the system, as large negative values are attenuated and are then mapped to a positive value rather than towards zero. The change from a negative value to a positive value also allows the integrator to force itself out of negative saturation, as strong negative values become positive, forcing the integrator to become more positive rather than being stuck within a negative region.

The final case is where the value of n is equal to 1. In this system, the non-linear function is inherently unstable as virtually no linear region exists, therefore the signal is always non-linearly transformed by a weak attractor; all positive values are drawn towards zero, and all negative values are eventually drawn towards the asymptote before becoming positive values.

A particularly interesting observation within the testing computational benchmark parameter sweeps for odd and even values of n show that the Moore-Penrose pseudo-inverse training algorithm performs more reliably for even values of n, with odd values of n producing more sporadic results. This is most likely due to the pseudo-inverse training algorithm struggling to find solutions within a system with a discontinuous

non-linear function. To reduce this sporadic behaviour, a ridge-regression training algorithm may perform better as the addition of a regression parameter may reduce the chances of calculating an unstable solution. However, this is a separate investigation and not necessary for this work.

It should be noted that although integer Mackey-Glass exponent values were used within this work, it is possible to use rational numbers; which often gives a complex result. Evaluating these values would be similar to evaluating a filter, as they can be represented within a magnitude and phase plot.

6.4.3 Region of Best Performance

Within section 5.5.2, the concept of a computational benchmark having a specific region of best performance within the parameter space where optimal performance is achieved has been introduced. There is strong evidence within chapter 5 to confirm that sub-hypothesis 1: *computational tasks with particular characteristics perform best within distinct areas of the parameter space of the delay-feedback reservoir system model* can be accepted. However, it is decided that additional experiments must be performed to determine if the computational tasks still have a region of best performance when other key parameters within the experimental model are changed.

Additional experiments are performed within this chapter, testing the effect that the Mackey-Glass non-linear function has on the performance of a delay-feedback reservoir system. These experiments once again confirm the concept of a region of best performance, as it has been observed that the NARMA-10 benchmark achieved best computational performance within a region of high LMC and low-to-moderate GR, while the Santa Fe benchmark achieved best computational performance within a region of high GR but low LMC. Therefore it can be said, if a computational task is well characterised in terms of non-linearity, dimensionality, and its dependency on previous input stimuli, then it will be possible to predict a region of best performance within the parameter space of the system. This allows for a more robust and methodical approach

to designing delay-feedback reservoirs, as it will be possible to tune a system to solve a particular computational task if the parameter space of the reservoir is known.

6.4.4 Utilisation of the Mackey-Glass Non-Linear Function

Throughout the analysis of the benchmark parameter sweeps within section 6.3, three key features are observed: the non-linearity of the Mackey-Glass non-linear function only changes when the value of the Mackey-Glass exponent is odd, even, or equal to 1; increasing the value of the Mackey-Glass exponent focuses the region of the best performing NRMSE values to a particular point, often moving the previously best region; the Santa Fe computational benchmark is affected by changing the value of n, with the performance decreasing as n increases, while the NARMA-10 benchmark remained unaffected. This poses the question, how much of the non-linear function is actually being utilised by the benchmark tasks?

To answer this question, a heat-map is used to measure the input and output behaviour of the non-linear function during run-time. The existing experimental model, shown in figure 4.8, is modified to enable the input and output values of the Mackey-Glass non-linear function to be recorded at a sample rate of θ. A sample rate of θ is chosen as this matches the period of the masking signal, which ensures that the input and output behaviour can be sampled when new values enter the non-linear function. The modified experimental model is shown in figure 6.17, created within Simulink 23.2.

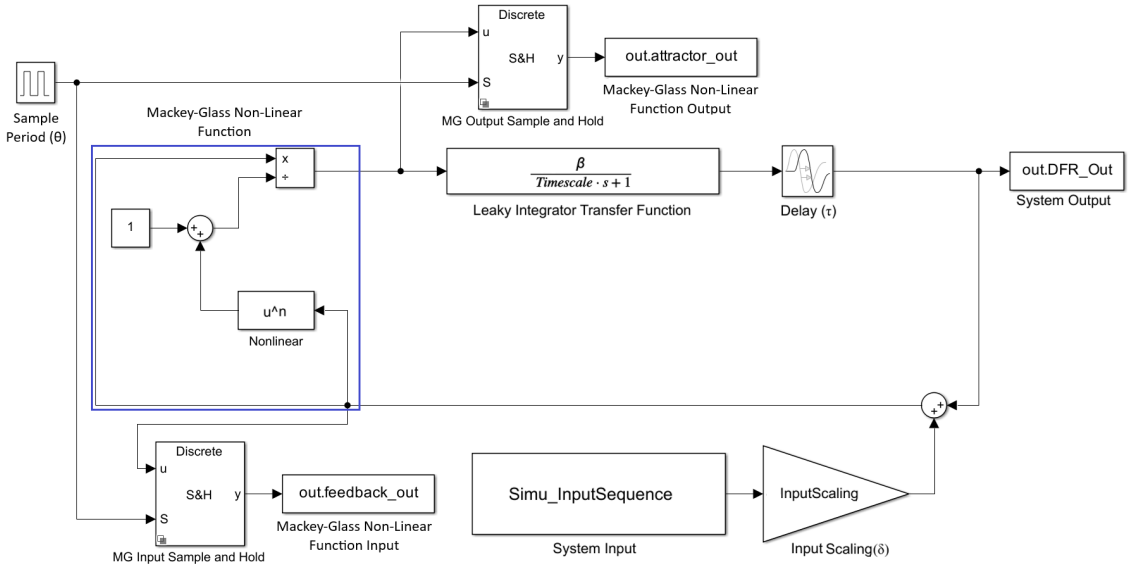

Figure 6.17 A schematic of a Delay-Feedback Reservoir, using the Mackey-Glass dynamical system as a non-linear function, and a first-order transfer function as the integration stage, modified to record the input and output of the Mackey-Glass non-linear equation, using a sample and hold function, at a sample period of θ. Model created within Simulink 23.2.

In order to sample the input and output of the Mackey-Glass non-linear function at the correct sample period, a sample and hold function, implemented by the Simulink "Sample and Hold" block, is used to capture the data at a sample period of θ. The output of the sample and hold functions are returned to the MATLAB workspace using the two "To Workspace" blocks, named "`out.feedback_out`" to record the input of the non-linear function and "`out.attractor_out`" to record the output of the non-linear function. The clock of the sample and hold blocks are driven using a Simulink "Pulse Generator", which generates a clock with a duty cycle of 50%, with a period of θ.

The best performing sweep parameters within the testing dataset, for the 20- and 200-node systems running the NARMA-10 computational benchmarks, are chosen for each of the different values of the Mackey-Glass exponent; these values are shown in table 6.2. A heat-map is then generated for each of the best performing parameters, to determine how much of the non-linear function is being utilised; this is shown in figure 6.18.

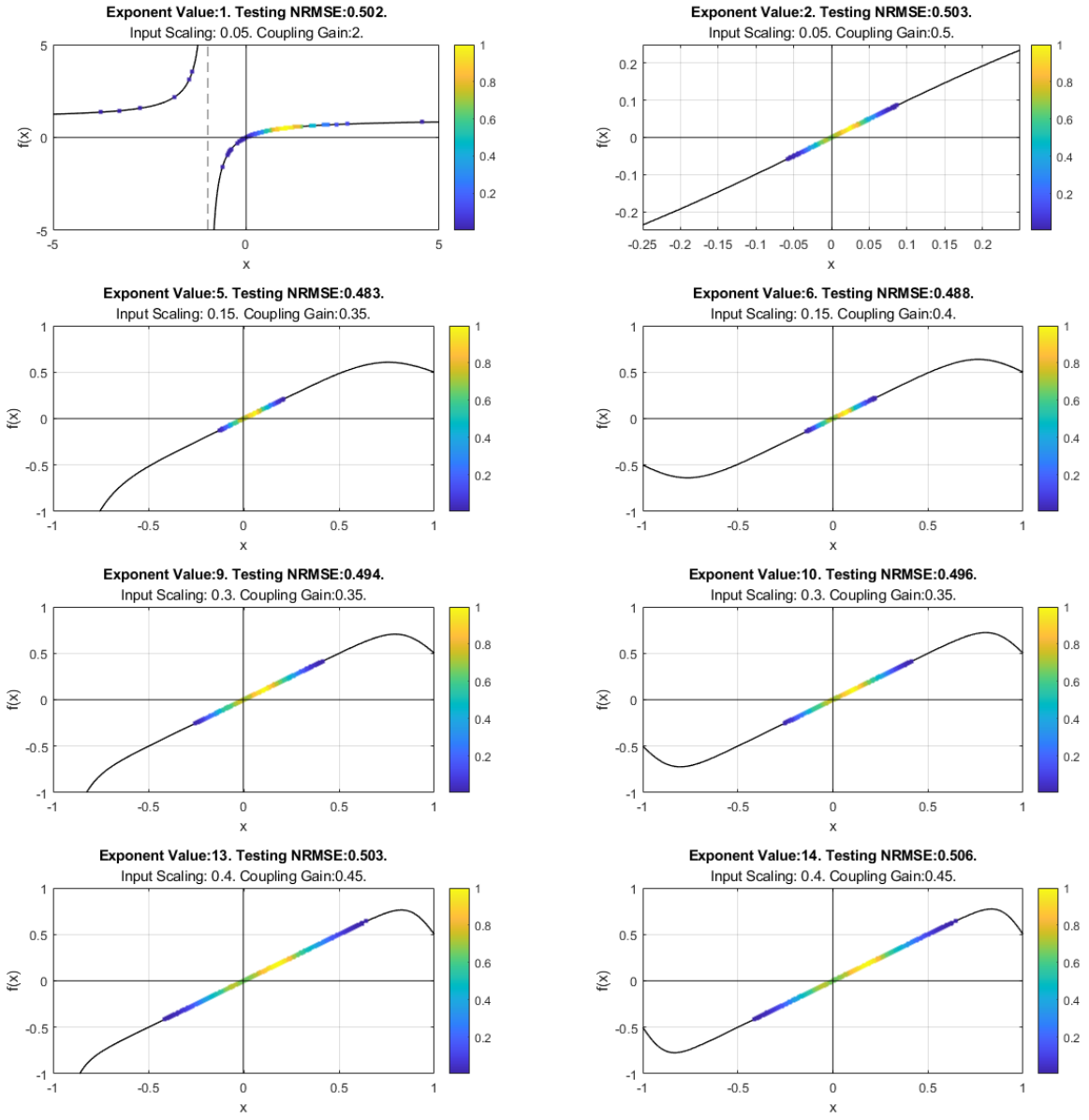

Figure 6.18 Heat-maps of the input and output behaviour of the Mackey-Glass non-linear node, for different values of the Mackey-Glass exponent, running on a 20-node delay-feedback reservoir system while evaluating the NARMA-10 benchmark, with the best performing sweep parameters determined in section 6.3.1.

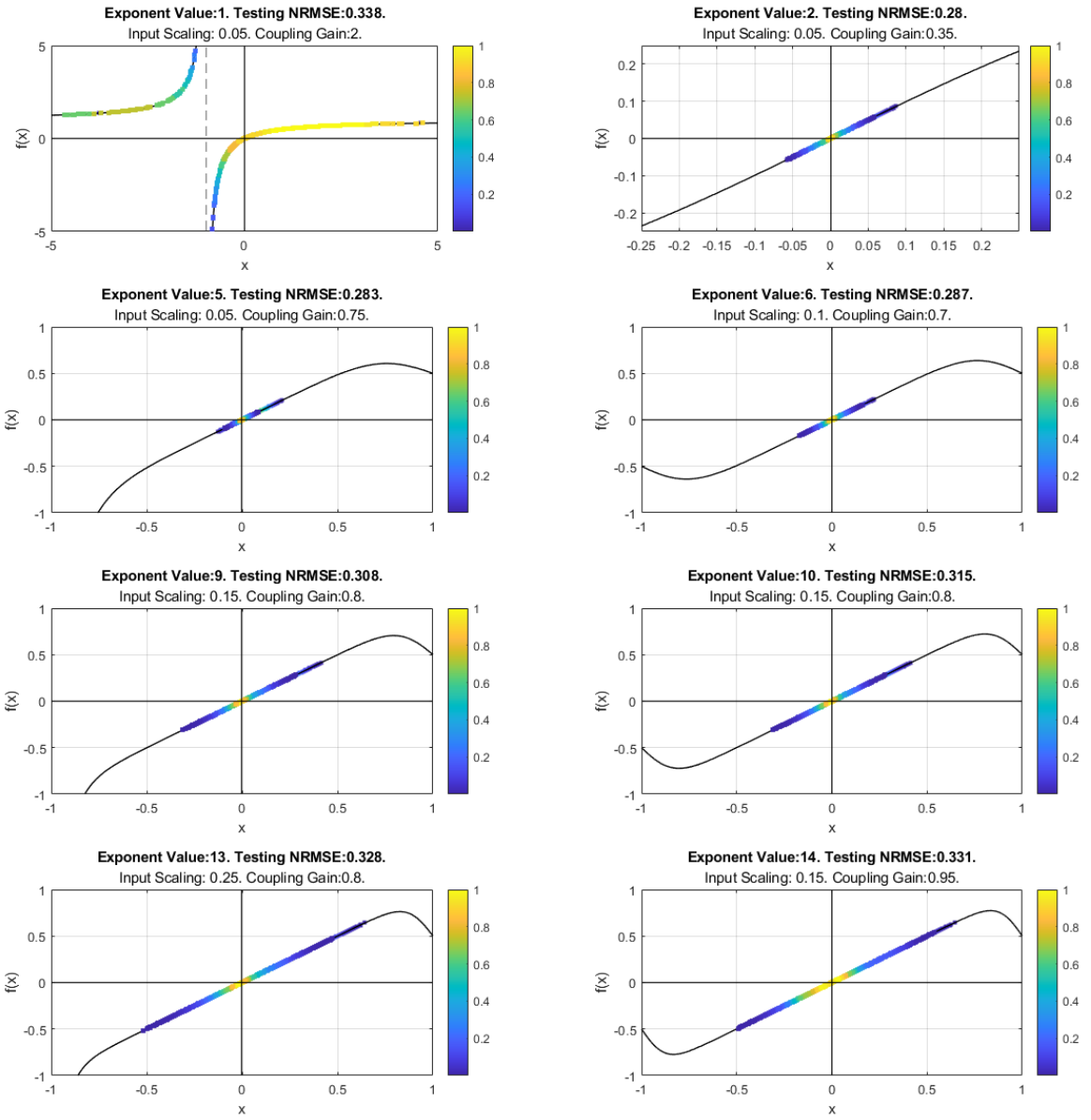

Figure 6.19 Heat-maps of the input and output behaviour of the Mackey-Glass non-linear node, for different values of the Mackey-Glass exponent, running on a 200-node delay-feedback reservoir system while evaluating the NARMA-10 benchmark, with the best performing sweep parameters determined in section 6.3.1.

Figures 6.18 and 6.19 show the heat-maps of the input and output behaviour of the Mackey-Glass non-linear function for different values of the Mackey-Glass exponent, for both a 20- and 200-node system, evaluating the NARMA-10 benchmark.

With the exception of the Mackey-Glass exponent, n, being equal to 1, the heat-maps show that the non-linear function operates exclusively within the linear range to achieve the best performance for the NARMA-10 benchmark. For the 20-node system, the area with the greatest point density is slightly to the right of the origin, while for the 200-node system, the area with the greatest point density is closer to the origin. This indicates that the 200-node system is utilising more of the system dynamics than the 20-node system in order to utilise both the positive and negative sides of the linear region. This is an expected result as the dynamics of the Mackey-Glass non-linear function have little effect on the performance of the system when running the NARMA-10 benchmark, but the number of virtual nodes do. As the NARMA-10 benchmark does require some non-linear dynamics in order for the reservoir to recreate the input/output behaviour of NARMA, it is likely that a linear function is not the optimal function to achieve the best computational performance; an alternative non-linear function with different dynamics may lead to better computational performance.

An interesting observation is that when n is equal to 1, the heat-maps confirm the inherent instability of the function. Within the 20-node system the instability is visible, as there are large values being remapped by the attractor behaviour of the non-linear function, but is mainly constrained within a fairly narrow region. However within the 200-node system, the instability is much more prominent, with the point density showing high utilisation within a large region of the non-linear function. The utilisation of the non-linear region when n is equal to 1 explains why the behaviour of this function is so different to the other values of n, as there is such a strong non-linear mapping of values within the function with very weak attractor behaviour.

With the heat-maps for the NARMA-10 systems investigated, the best performing sweep parameters within the testing dataset, for the 20- and 200-node systems running

the Santa Fe computational benchmark, are chosen for each of the different values of the Mackey-Glass exponent; these values are shown in the table 6.4. A heat-map is then generated for each of the best performing parameters, to determine how much of the non-linear function is being utilised; this is shown in figure 6.20.

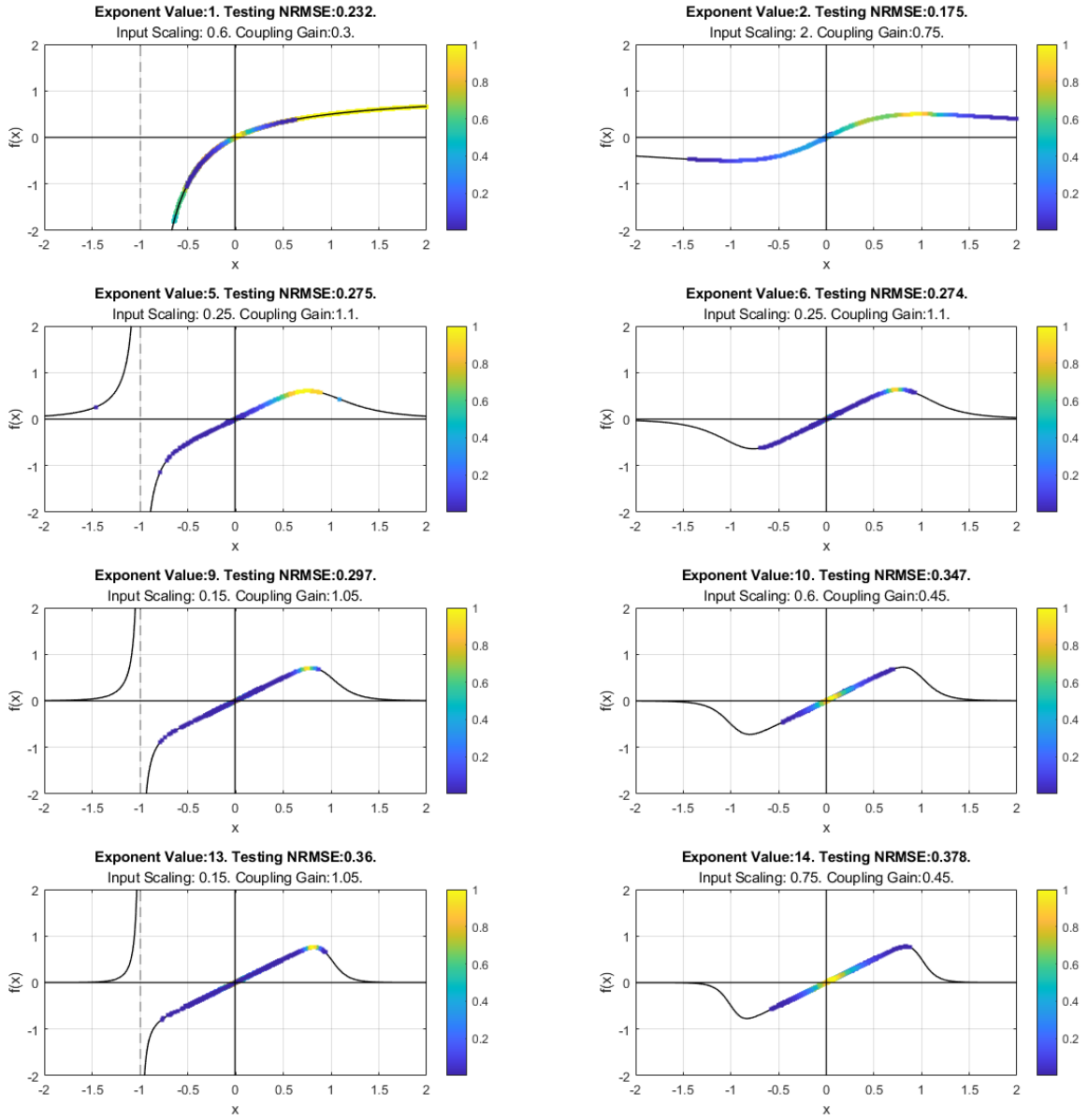

Figure 6.20 Heat-maps of the input and output behaviour of the Mackey-Glass non-linear node, for different values of the Mackey-Glass exponent, running on a 20-node delay-feedback reservoir system while evaluating the Santa Fe benchmark, with the best performing sweep parameters determined in section 6.3.1.

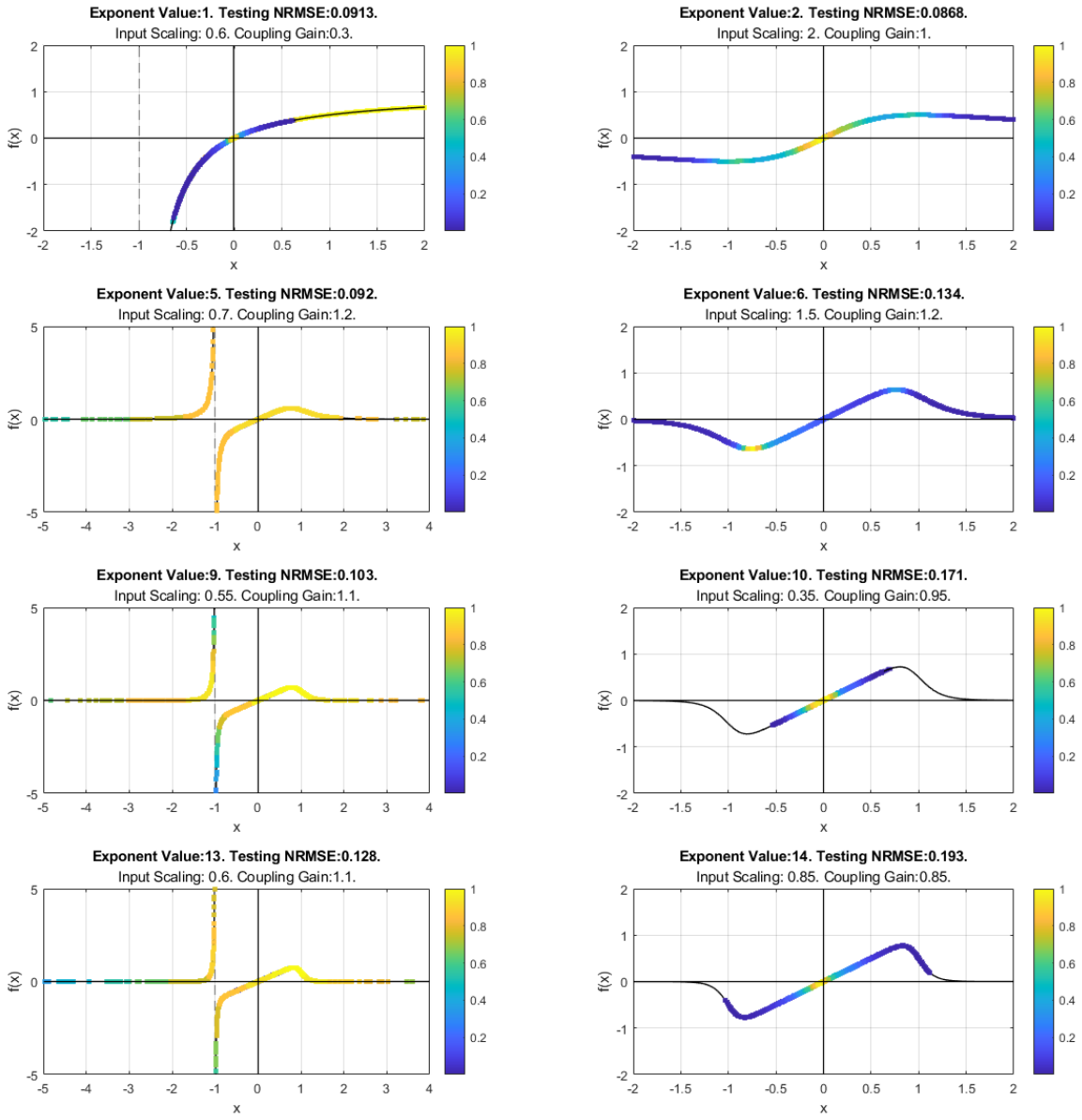

Figure 6.21 Heat-maps of the input and output behaviour of the Mackey-Glass non-linear node, for different values of the Mackey-Glass exponent, running on a 200-node delay-feedback reservoir system while evaluating the Santa Fe benchmark, with the best performing sweep parameters determined in section 6.3.2.

Figures 6.20 and 6.21 show the heat-maps of the input and output behaviour of the Mackey-Glass non-linear function for different values of the Mackey-Glass exponent, for both a 20- and 200-node system, evaluating the Santa Fe benchmark.

Unlike the NARMA-10 benchmark, which has the region with the greatest point density within the linear region of the Mackey-Glass non-linear function, the region with the greatest point density within the Santa Fe benchmark is at the peak of where the attractor behaviour is the strongest, for the majority of the heat-maps. Given the requirement for strong non-linearity to compute the Santa Fe benchmark, it is to be expected that the preferred region would be at the point of greatest non-linearity within the function.

The best performance is found when the Mackey-Glass exponent is at its lowest, where the linearity is at its highest and the attractor effect is at its weakest, giving the function a less aggressive non-linear transform, allowing for more of the non-linear function to be utilised. As the number of virtual nodes increases, more of the non-linear function is utilised. Typically, within the even values of the Mackey-Glass exponent there is a localised high density point, with the rest of the non-linear function being weakly utilised. However, within the odd values of the Mackey-Glass exponent, the majority of the non-linear function is utilised, with no clear localised usage area. The additional utilisation within the odd values of the Mackey-Glass exponent is most likely due to the strong negative attractor behaviour, mapping strong negative values to positive values rather than towards 0, as observed with the even Mackey-Glass exponent case.

6.5 Summary

Within this chapter, the effect that the Mackey-Glass non-linear node has on the performance and parameter sensitivity of a delay-feedback reservoir is investigated with the aim of answering sub-hypothesis 3: *Changing the behaviour of the Mackey-*

Glass non-linear function can lead to an increase in performance within different computational tasks, with minimal changes to the hardware architecture.

In order to answer this hypothesis, two delay-feedback reservoir systems, one with 20-nodes and the other with 200-nodes, were tested at eight different Mackey-Glass exponent values, $1, 2, 5, 6, 9, 10, 13$, and 14, while separately evaluating the NARMA-10 and Santa Fe computational benchmarks.

6.5.1 Effectiveness of the Mackey-Glass Non-Linear Function

As the input scaling and the coupling gain greatly affect the computational performance of a reservoir system, a parameter sweep is performed for each reservoir node and Mackey-Glass exponent permutation, for both the NARMA-10 and Santa Fe benchmarks, as well as the system metrics.

It was found during the parameter sweeps that increasing the Mackey-Glass exponent increases the width of the linear region and increases the strength of the attractor, it has no effect on the KQ and GR of the system, but it does have a moderate effect on the LMC.

The Mackey-Glass non-linear function was found to have three operating regions, these are when the exponent is: odd, there is an asymptote present within the function, that causes an attractor behaviour that is able to map negative values to positive values; even, the non-linear function is symmetrical, this causes the attractor behaviour to map negative values towards zero; equal to one, the system is chaotic and unstable.

Further experiments were performed to find how much of the Mackey-Glass non-linear function is being utilised. To check the utilisation, the input and output behaviour of the non-linear function is recorded during run-time to generate a heat-map for every reservoir node, Mackey-Glass exponent, and benchmark permutation.

The generated heat-maps show two interesting features: first, the 20-node systems utilise a smaller amount of the non-linear function, when compared to a 200-node

systems; second, the NARMA-10 benchmark has the greatest point density within the linear region, whereas the Santa Fe benchmark has the greatest point density within the peak of the positive attractor.

6.5.2 Conclusions

The work within this chapter confirms the validity of sub-hypothesis 3, as it was shown that the non-linear function can be tuned to create different non-linear behaviours and system dynamics, leading to an increase in performance. The changes in behaviour and dynamics are achieved by changing the exponent value within the Mackey-Glass non-linear function, which can be changed relatively easy within hardware without a hardware redesign, typically by changing the bias resistor values.

Additionally, sub-hypothesis 1 is accepted by once again observing that the NARMA-10 and Santa Fe computational benchmarks had a particular region of best performance within the parameter space of the delay-feedback reservoir system model, where optimal performance could be achieved. The following key conclusions are made within this chapter:

- If a particular computational task has a strong dependency on a non-linear transform, then additional performance may be gained by decreasing the exponent value within the Mackey-Glass non-linear function.
- The Mackey-Glass non-linear function has three operating modes that provide different non-linear behaviours and dynamics, which can be changed by setting the Mackey-Glass exponent to either one, odd, or even.
- The Mackey-Glass non-linear function is rather limited in terms of non-linear dynamics, exhibiting an adjustable region of linearity of unity gradient with an attractor present at the edges of the linear range.

- The Mackey-Glass non-linear function has a predictable region of best performance for both the NARMA-10 and Santa Fe benchmark tasks, with the NARMA-10 primarily utilising the linear region, while Santa Fe primarily utilises the non-linear regions.
- The experiments within this chapter further support sub-hypothesis 1; that a computational task has a particular region of best performance within the parameter space of the delay-feedback reservoir system model, where optimal performance can be achieved.

Chapter 7

High-Order Filters as Non-Linear Node

7.1 High-Order Filter Non-Linear Node

In chapter 4, it was found that the non-linear node for a delay-feedback reservoir system consisted of two functional blocks; a non-linear function, $f(t)$, and an integration function, $h(t)$. Typically, a simple first-order RC filter is used for the integration stage due to its simplicity to model and implement within hardware. The choice of which non-linear function to use is much more ambiguous, with several different hardware friendly implementations to choose from. The Mackey-Glass delay-differential equation is a promising function that is widely used as it already consists of two stages, a non-linear function and a leaky integrator, and the behaviour of the non-linear function can be relatively easily realised in hardware. As shown in chapters 5 and 6, the combination of a first-order RC filter and the Mackey-Glass non-linear function creates a powerful delay-feedback reservoir, with its many parameters able to be tuned to increase the efficiency when solving a particular computational task. However, as shown within chapter 6, the Mackey-Glass non-linear function only has three operating modes where it provides different dynamics to the system, which limits the dynamics it can produce.

An alternative approach could be to use a high-order filter to attempt to combine both the non-linear and integration functions together. This would create a high-order filter that can both provide additional dynamics to the system, such as oscillations, and perform the necessary integration in an all-in-one non-linear node. This would give the advantage of being hardware friendly, as filters are typically easily implemented within analogue circuity, and have the ability to adjust key parameters in real-time. The problem with the approach is that a high-order filter typically has many parameters, each of which can have a complex effect on the overall behaviour of the filter. A typical high-order filter, $H(s)$, can be expressed in the Laplace domain as:

$$H(s) = \frac{(s - z_0)(s - z_1)...(s - z_n)}{(s - p_0)(s - p_1)...(s - p_m)} \qquad m \geq n \tag{7.1}$$

Where n and m is the number of zeros and poles respectively and, z_n is the n^{th} zero location, p_m is the m^{th} pole location.

The parameter space for the proposed approach is large, and it is not obvious which parameters will work well. Therefore, in order to determine the parameters required to confirm if a high-order filter can be used as a non-linear node, the field of evolutionary algorithms is explored; an overview of which can be found within appendix A.

7.2 Experimental Implementation

A possible alternative approach to designing the non-linear node within a delay-feedback reservoir could be to use a high-order filter to provide both the non-linear dynamics and the integration behaviour required for a non-linear node. While it was shown in chapter 5 how the dynamics of a delay-feedback reservoir can be changed by changing the timescale of the first-order integration stage, within a high-order filter there are multiple timescales and additional variables that may affect the dynamics of the reservoir system. An ideal approach would be to have an in-the-loop evolutionary algorithm combined with an online learning technique so that both the high-order filter and delay-feedback reservoir system parameters can be optimised to solve a computational task with a particular set of dynamics. However, as there is no previous literature on using a high-order filter as a replacement for a non-linear node within delay-feedback reservoirs, it is not obvious what dynamics would be preferable within a high-order filter to realise a non-linear node; making an in-the-loop approach very difficult without the necessary insight. As it is not practical to perform parameter sweeps on every parameter within a high-order filter to gain this insight, a simpler evolutionary algorithm is used to determine and optimise the parameters required to create a high-order filter with strong dynamics to replace the non-linear node.

7.2.1 Genetic Representation of a High-Order Filter

In order to realise an evolutionary algorithm, a representation of a high-order filter must first be genetically encoded. As previously discussed, a high-order filter can be mathematically expressed in the Laplace domain as:

$$H(s) = \frac{(s - z_0)(s - z_1)...(s - z_n)}{(s - p_0)(s - p_1)...(s - p_m)} \qquad m \geq n \tag{7.2}$$

Where n is the number of zeros, m is the number of poles, z_n is the n^{th} zero location and p_m is the m^{th} pole location.

This allows a high-order filter to be expressed in terms of its poles and zeros, fully characterising the corresponding differential equation; where the poles of the transfer function defines the modes of the differential equation and the zeros change the residues [130]. For example, a second-order filter with one zero can be expressed as the following:

$$H(s) = \frac{s - z_0}{(s - p_0)(s - p_1)} = \frac{s - z_0}{s^2 - (p_0 + p_1)s + p_0 p_1} \tag{7.3}$$

While the monic form of a Laplace transfer function is often used within the literature, it has the problem of producing a different DC gain for different pole and zero values. This can be shown by using the final value theorem to calculate the DC gain:

$$\begin{gathered} H(s) = \frac{s - z_0}{s^2 - (p_0 + p_1)s + p_0 p_1} \\ H(\infty) = \lim_{s \to 0} [H(s)] \\ = \lim_{s \to 0} \left[\frac{s - z_0}{s^2 - (p_0 + p_1)s + p_0 p_1} \right] \\ = \frac{-z_0}{p_0 p_1} \end{gathered} \tag{7.4}$$

If this filter representation was used within the evolutionary algorithm, then an additional coupling gain factor would be applied to the high-order filter every time the values of the poles and zeros are changed. This would make it very difficult to compare the performance of high-order filters for different pole and zero values. An alternative approach is to use a non-monic filter representation, where the coefficient of the lowest power term is equal to 1, allowing for a unity DC gain which is independent of the pole and zero values. This can be expressed as:

$$H(s) = \frac{(1 - \frac{s}{z_0})(1 - \frac{s}{z_1})...(1 - \frac{s}{z_n})}{(1 - \frac{s}{p_0})(1 - \frac{s}{p_1})...(1 - \frac{s}{p_m})} \tag{7.5}$$

This allows a second-order filter with one zero to be expressed as the following:

$$H(s) = \frac{1 - \frac{s}{z_0}}{\frac{s^2}{p_0 p_1} - \frac{s}{p_0 + p_1} + 1} \tag{7.6}$$

This new mathematical representation allows for changes to be made to the pole and zero values without changing the DC gain of the system. This once again can be shown using the final value theorem:

$$\begin{aligned} H(s) &= \frac{1 - \frac{s}{z_0}}{\frac{s^2}{p_0 p_1} - \frac{s}{p_0 + p_1} + 1} \\ H(\infty) &= \lim_{s \to 0} [H(s)] \\ &= \lim_{s \to 0} \left[\frac{1 - \frac{s}{z_0}}{\frac{s^2}{p_0 p_1} - \frac{s}{p_0 + p_1} + 1} \right] \\ &= \frac{1}{1} = 1 \end{aligned} \tag{7.7}$$

With the mathematical representation determined, the genetic encoding can now be considered.

It was shown in equation 7.5 that a high-order filter can be represented by the product of its poles and zeros. As only real-valued poles and zeros are to be initially

considered, a simplistic genetic encoding can be used where a genome consists of the poles and zeros values required to characterise the dynamics of a high-order filter combined with an "activation" parameter that determines whether a given pole or zero is active. This allows for the dynamics of a high-order filter to be modified by independently changing the location of the poles and zeros, which can then be modified by a genetic operator during the evolutionary process. An example of this encoding for a third-order two zero filter is shown in figure 7.1.

	p0	p1	p2	p3	p4	p5
Pole Value	-1	-4	-3	-5	-5.5	-3.5
Pole Active	1	1	1	0	0	0

	z0	z1	z2	z3
Zero Value	-1	-4	-3	-5
Zero Active	1	1	0	0

Figure 7.1 The genetic encoding of a third-order two zero high-order filter. The values of the poles and zeros are stored within an array, with a boolean "activation" parameter used to determine if a pole or zero is active.

In order to generate the transfer function of the filter represented by the genetic encoding shown in figure 7.1, equation 7.5 is applied to give the following transfer function for use in simulation:

$$H(s) = \frac{(1+s)(1+\frac{s}{4})}{(1+s)(1+\frac{s}{4})(1+\frac{s}{3})} \tag{7.8}$$

7.2.2 Evolutionary Algorithm Implementation

As the aim is to use evolutionary algorithms as a proof of concept tool, it is important to keep the evolutionary algorithm simple. Therefore the microbial evolutionary algorithm developed by Harvey [131] is chosen due to its simple, lightweight, and versatile design. The main difference between the microbial evolutionary algorithm and the evolutionary algorithm used within this work, is that instead of two individuals being randomly selected from the population to be evaluated and placed back in the population, all individuals within the population are paired together and evaluated in

parallel. This adds a large genetic variation to the population in a single generation, rather than periodically, and keeps the population constant. This change allows the algorithm to be optimised in terms of speed; generally fewer generations are required to reach the optimisation goal [132], and computational resources and time can be saved by running simulations in parallel. A flow diagram of the implemented evolutionary algorithm is shown in figure 7.2.

The key sections within the evolutionary algorithm described in figure 7.2 will now be discussed.

Initialisation

During the initialisation stage of the evolutionary algorithm, the number of poles and zeros within an individual is randomly generated for all individuals within the population. The number of poles within an individual is first generated from a random uniform distribution, between a fixed minimum and maximum value, then the number of zeros is also generated from a random uniform distribution, up to a maximum value of the previously generated number of poles minus one. This ensures that an individual can represent a real physical high-order filter by ensuring the number of zeros is never greater or equal than the number of poles. With the number of poles and zeros within each individual randomly chosen, the initial pole and zero values for each individual are then generated.

The initial pole and zero values are generated from a Gaussian distribution. A Gaussian distribution is chosen over a uniform distribution for two reasons. First, the effect of moving the poles and zeros within a high-order filter alters the behaviour significantly. A Gaussian distribution is more likely to produce a gradual change in pole and zero values, while still having a possibility to generate values on the fringes of the distribution; how likely a pole or zero is to deviate from the mean can be controlled by the standard deviation of the distribution. Secondly, in order to get the strongest dynamics from a high-order filter, the poles need to be within an order of magnitude of each other. If a pole is much further away from the other, it may either dominate the

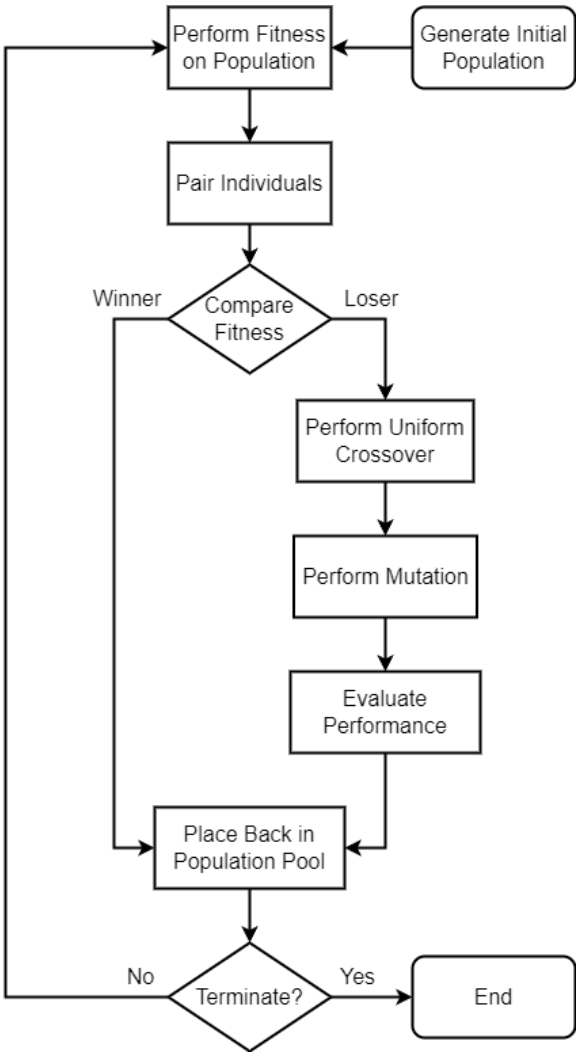

Figure 7.2 A flow diagram of the evolutionary algorithm used within chapter 7. The evolutionary process is based on the microbial evolutionary algorithm but optimised for parallel processing with MATLAB.

filter behaviour and act like a first-order filter or become dominated by other faster poles. An example of how a pole or zero is generated from a Gaussian distribution is shown in figure 7.3.

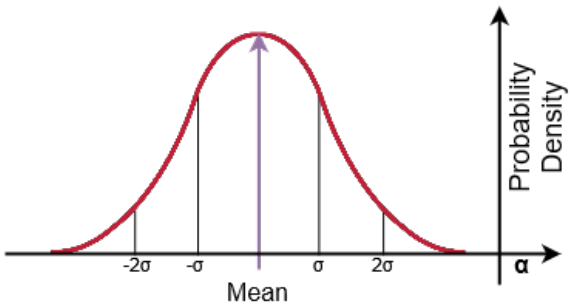

Figure 7.3 A Gaussian distribution showing the probability curve that the initial poles or zeros are generated from.

Evaluation

The fitness score of each individual will be determined by evaluating the performance of the corresponding filter, when running a computational benchmark, in terms of NRMSE. The mean of the NRMSE value for both the training and testing datasets is then used to represent the fitness score of the individual. Using the mean NRMSE value of the training and testing datasets allows the evolutionary algorithm to determine the best performing individual in terms of performance, but also finds the best solution in terms of stability as both the testing and training errors are considered.

As a high-order filter may also be sensitive to input scaling and coupling gain, a parameter sweep needs to be performed for each individual. As it is not feasible to perform an extensive parameter sweep for every individual, a coarse parameter sweep consisting of 16 points will be used. Given that there may be large variations in the test error between parameter sweep points, the geometric mean is taken to suppress any large fluctuations in the test error that would skew the arithmetic mean. The standard deviation of the test errors is also recorded to give an indication of the parameter sensitivity for each individual, but it is not used within the fitness calculation as it would provide a bias towards parameter sensitivity over overall performance.

Selection

The selection process is entirely random. The population is split into two equal parts and then shuffled. Individuals are then paired together and evaluated based upon the evaluation criteria. The individual with the lowest fitness score is designated the winner and is placed back into the population pool unchanged; this is an elitism strategy to keep the winning genetic traits in the next generation. The individual with the higher fitness score is designated the loser, and has the genetic operators, uniform crossover, and mutation applied to them; the modified loser is then placed back in the population pool.

Termination

The evolutionary loop terminates when either the maximum number of generations have been reached, or the fitness score has become stagnant. There is no target fitness termination criteria within this evolutionary algorithm as although the theoretical "ideal" fitness score would be zero, the practical limit to the fitness function is unknown.

7.2.3 Genetic Operators

There are two genetic operations present in the evolutionary algorithm used within this work; uniform crossover and mutation. The uniform crossover genetic operator takes a "winner" and a "loser" individual pair and stochastically mixes them together. This performs two crucial steps, first it allows the loser to inherit potentially beneficial pole or zero locations from the winner, and secondly it stochastically increases or decreases the number of active poles and zeros within the losing individual.

An example of an individual pair undergoing the uniform crossover operation is shown in figure 7.4, where the blue boxes represent genes from the winning individual and the orange boxes represent genes from the losing individual.

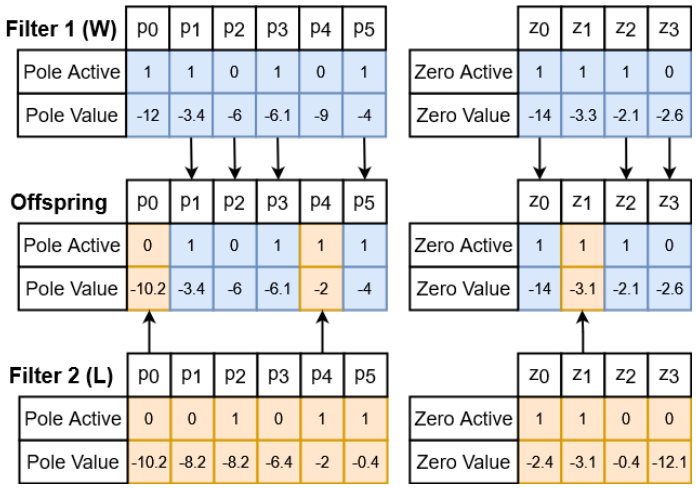

Figure 7.4 A "winner" and "loser" individual pair undergoing the uniform crossover genetic operation. The blue boxes represent genes from the winning individual, while the orange boxes represent genes from the losing individual.

Figure 7.4 shows how the winning fourth-order three zero filter is mixed with the losing third-order two zero filter using the uniform crossover genetic operation to create a fourth-order three zero filter.

After the uniform crossover has produced an offspring, the mutation genetic operator is then applied to it. The mutation genetic operation randomly mutates an active pole or zero within an individual. If a pole or zero is selected for mutation, a new value is generated from a Gaussian distribution, with the mean value of the distribution set to the current value and the standard deviation set as a hyper-parameter. A Gaussian distribution is once gain chosen over a uniform distribution to mutate the pole and zero values as it is more likely to produce a gradual change in values rather than compared to the uniform distribution.

An example of the mutation genetic operation acting on a fourth-order two zero filter shown in figure 7.5; the green and red boxes represent which pole and zero values are active or inactive respectively, the purple boxes represent which genes are selected for mutation, and a graph shows how the poles and zeros move during the mutation process.

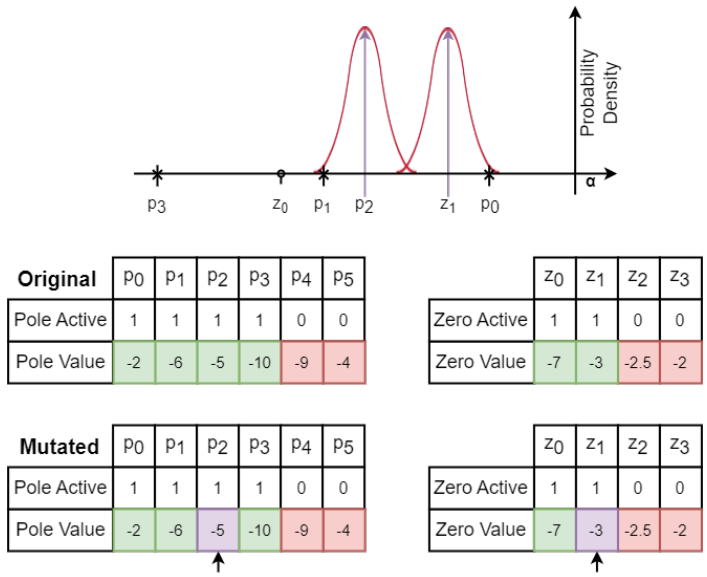

Figure 7.5 An example of a mutation genetic operation being performed in a fourth-order two zero filter. The green and red boxes indicate active and inactive pole and zero values respectively, with the purple boxes representing a gene that has been selected for mutation. A graph displaying how the poles and zeros move during the mutation process is also shown above.

7.2.4 Simulation Methodology

With the behaviour of the evolutionary algorithm outlined, the functionality is implemented into software. The MATLAB computing environment is used to implement the evolutionary algorithm as Simulink can be used to accurately simulate the effect of an high-order non-linear node as a physical system. The existing experimental model, as shown in figure 4.8, is modified to remove the Mackey-Glass non-linear function and to extend the functionality of the "Transfer Function" block to accept an arbitrary transfer function; the modified experimental model is shown in figure 7.6.

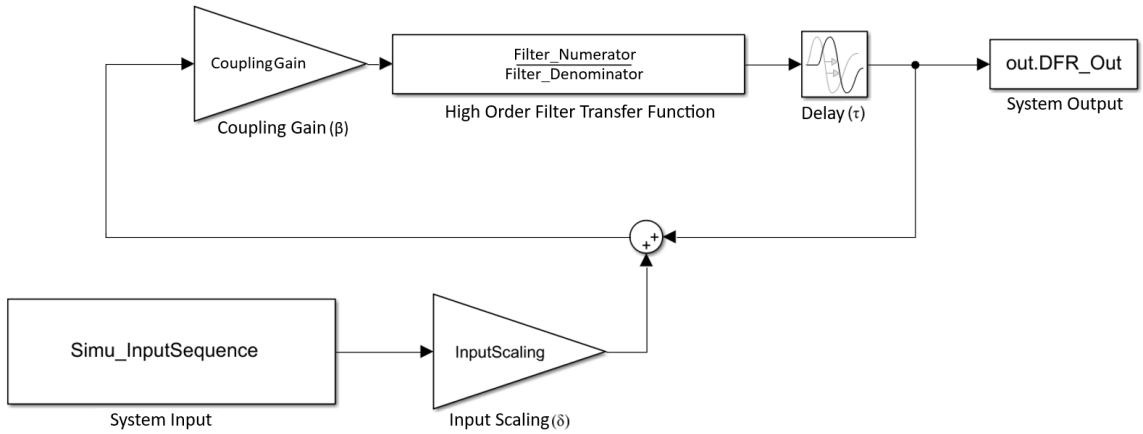

Figure 7.6 A schematic of a Delay-Feedback Reservoir, using a high-order transfer function as the non-linear node, created within Simulink 23.2.

The Simulink "Transfer Function" is able to simulate an arbitrary transfer function as defined by the *Filter_Numerator* and the *Filter_Denominator* variables. The *Filter_Numerator* and the *Filter_Denominator* variables represent the coefficients of the numerator and denominator polynomials respectively, which can be calculated by expanding the poles and zeros from the genetic representation.

7.3 Experimental Methodology

With the evolutionary algorithm and experimental models complete, the effect that a high-order filter non-linear node has on the performance (as a primary metric) and parameter sensitivity (as a secondary metric) of a delay-feedback reservoir can be investigated.

In order to compare with the previous experiments performed in chapter 5 and 6, both a 20- and 200-node reservoir system are to be explored with a τ of 80 s. Exploring two differently sized reservoir systems gives insight into which dynamics the evolutionary algorithm may take advantage of, and distinguish if they are virtual node dependent. As the NARMA-10 and Santa Fe require different computational dynamics to achieve the best performance, separate experiments will be run using the NARMA-10 and Santa

Fe benchmarks within the evolutionary algorithm. This is performed as the fitness score for different computational benchmark tasks will be different, as NARMA-10 favours high memory and Santa Fe favours non-linearity. Performing separate experiments for each benchmarks allows the evolutionary algorithm to better find solutions that best suit the benchmark. In theory, this should produce different high-order filters that have the best dynamics to solve a particular type of computational benchmark.

The population size within the evolutionary algorithm is set to 100 individuals and is kept constant. As each individual is randomly generated, a population of 100 individuals should produce an initial diverse population pool in terms of filter characteristics. The maximum number of generations to be run is set to 200. Although a fitness stagnation termination criteria was implemented within the evolutionary algorithm, it was decided to disable it for these experiment runs as the number of generations run is relatively small and preliminary experiments show that it may take a while for the fitness to change.

The maximum number of poles and zeros that an individual can represent is set to 5 and 4 respectively, and there will always be one more pole within an individual than zeros; this is done for two reasons. First, a maximum number of 5 poles allows for sufficiently strong dynamics to be generated within a high-order filter while still being computationally cheap to simulate. Second, as poles and zeros can cancel out the effects of each other, ensuring that there is always one more pole than zero ensures the high-order filter always has the ability to perform integration, which is an essential property for a delay-feedback reservoir system.

The mean value of the Gaussian distribution that initially generates the poles and zeros is set to -1.7. This value is chosen as the best performing timescale for a first-order filter was found to be 400ms within chapter 5. The value is then increased to 600ms to allow for a greater distribution in values, and converted into a pole location by calculating its reciprocal; giving approximately 1.7. The same mean value is chosen for both the pole and zero locations as it ensures a symmetrical distribution of values.

The standard deviation, σ, is chosen to be 0.2 for the pole distribution and 0.6 for the zero distribution. This ensures that there is a gradual change in pole values, as a small change can greatly affect the modes of the system, with a wider variation of zeros as they have a less prominent effect.

Given that there is no Mackey-Glass non-linear function to act as an attractor, it is very likely that for a coupling gain that is greater than one will cause unstable positive feedback within the delay-feedback reservoir resulting in saturation. Within chapters 5 and 6, it was found that there was a region of strongest performance at the bottom left of the parameter space for both the NARMA-10 and Santa Fe benchmark. Therefore, the range of the values for the coarse parameter sweep is between 0.2 to 0.8, with a step of 0.2. This produces 16 points within the bottom left of the parameter space, which should give a good indication of the performance of each individual.

The crossover rate is set to 0.75 for both the poles and zeros, as a high probability of inheriting the genes from the winning individual is desired. The mutation rate is set to 0.2 for both the poles and zeros, which is chosen as it is the reciprocal of the maximum number of poles that an individual can have. This means that statistically for an individual with a maximum number of poles, one gene should mutate.

The important hyper-parameters used within the evolutionary algorithm are listed in table 7.1 with their default value.

Name	Function	Value
Time Delay (τ)	Sets the time delay within the feedback loop.	80 s
Population Size	The maximum number of individuals within the population.	100
Maximum Generations	The maximum number of generational cycles to be run.	200
Maximum Pole Order	The maximum number of poles an individual may have.	5
Maximum Zero Order	The maximum number of zeros an individual may have.	4
Mean of Pole Distribution	The mean value of the pole within the Gaussian distribution, marking the centre point.	1.7
Mean of Zero Distribution	The mean value of the zero within the Gaussian distribution, marking the centre point.	1.7
σ of Pole Distribution	The standard deviation of the pole Gaussian distribution, controlling the spread of generation.	0.2
σ of Zero Distribution	The standard deviation of the zero Gaussian distribution, controlling the spread of generation.	0.6
Pole and Zero Crossover Rate	The probability that a gene crossover will occur.	0.75
Pole and Zero Mutation Rate	The probability that a mutation of a gene will occur.	0.2

Table 7.1 A list of hyper-parameters within an evolutionary algorithm to determine the optimum real-valued high-order filter to act as a non-linear node within a delay-feedback reservoir, with their default value.

7.4 Results: Baseline

Before the evolutionary algorithm is run, a baseline is established to evaluate the performance and parameter sensitivity of a delay-feedback reservoir system without a non-linear function. The experimental model shown in figure 7.6 is used to perform the parameter sweep. The transfer function is set to the same first-order transfer function used within chapter 6, with the timescale set to 400ms.

A parameter sweep of the input scaling and the coupling gains will be performed for a 20- and 200-node reservoir system running both the NARMA-10 and Santa Fe benchmarks and evaluated using NRMSE. The system metrics, as discussed in section 3.3, are also calculated to gain insight into the dynamics of a non-linear node

without a non-linear function. Table 7.2 shows the parameter values and ranges for the proposed parameter sweeps.

Name	Symbol	Value (20-Node)	Value (200-Node)
Coupling Gain	β	0.05-1.1	0.05-1.1
Input Scaling Factor	δ	0.05-2	0.05-2
System Timescale	T	400ms	400ms
Time Delay	τ	80 s	80 s
Masking Signal Period	θ	4 s	0.4 s

Table 7.2 A table of parameter values of the delay-feedback reservoir Simulink model during the input scaling and the coupling gain parameter sweep for a system without a non-linear function.

7.4.1 NARMA-10 Benchmark

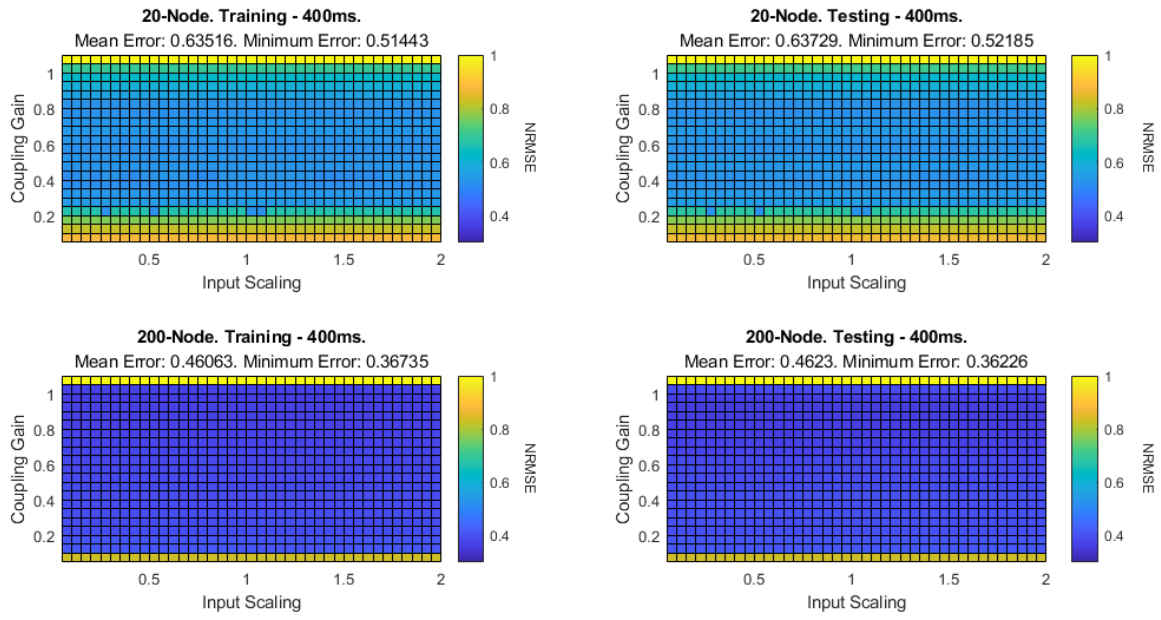

Figure 7.7 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20- and 200-node system running the NARMA-10 benchmark without a non-linear function, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1.

Figure 7.7 shows the effect of the input scaling and the coupling gain on a 20- and 200-node delay-feedback reservoir system, running the NARMA-10 benchmark, without a non-linear function within the non-linear node. It should be noted that although the "non-linear node" no longer contains a non-linear function, it is still referred to as a non-linear node as some non-linearity comes from the integration stage.

Within both the 20- and 200-node systems, the training and testing error is fairly consistent throughout most of the parameter space. There is a noticeable increase within the error values when the coupling gain is high and when the coupling gain is low. When the coupling gain is high, the system is prone to saturation as it becomes a positive feedback system. Whereas when the coupling gain is low, the output of the filter is significantly attenuated, creating a loss of memory within the system.

Table 7.3 shows that the training and testing errors within the parameter sweeps in figure 7.7 is slightly higher than in previous experiments within this work. A slightly lower performance is to be expected, as the NARMA-10 task requires a small amount of non-linearity within a system, which is now only provided by the first-order filter as the non-linear function has been removed.

An interesting observation, confirming the findings from chapter 6, is that without the non-linear function there is virtually no sensitivity to the input gain. It was suggested that the Mackey-Glass non-linear function caused an increase in sensitivity to the input scaling within the reservoir system. Hence, removing the non-linear function also removes the dependency on input scaling as there is no additional transform before the first-order transfer function; causing the relationship between the input and the output of the non-linear node to be only dependent on the coupling gain. This also explains why any coupling gain above one causes the system to become unstable, as it causes a positive feedback loop within the system.

The 20-node system shows a region of best performance when the coupling gain is approximately 0.8, while the 200-node system remains fairly constant between the top and bottom boundaries; this is likely due to the memory capacity of the system.

As the coupling gain increases, so does the memory capacity as more of the previous information is retained within the feedback loop. As the 20-node system has a theoretical maximum LMC of 20, the memory capacity will increase as the coupling gain increases, until an LMC of 10 is reached. At this point the system has enough LMC for the NARMA-10 benchmark to fully utilise, which occurs at a coupling gain of approximately 0.8. This effect is not visible within the 200-node system, as the minimum LMC is achieved much earlier than the 20-node system, as the theoretical maximum LMC of the 200-node system is 200.

	Best NRMSE from Chapter 5		Best NRMSE from Chapter 6		Current System Best NRMSE	
Nodes	20	200	20	200	20	200
Training Error	0.491	0.306	0.484	0.271	0.514	0.367
Testing Error	0.494	0.308	0.483	0.280	0.522	0.362

Table 7.3 A table showing the best performing NARMA-10 benchmark results from previous chapters and comparing them with the parameter sweeps performed in figure 7.7.

7.4.2 Santa Fe Benchmark

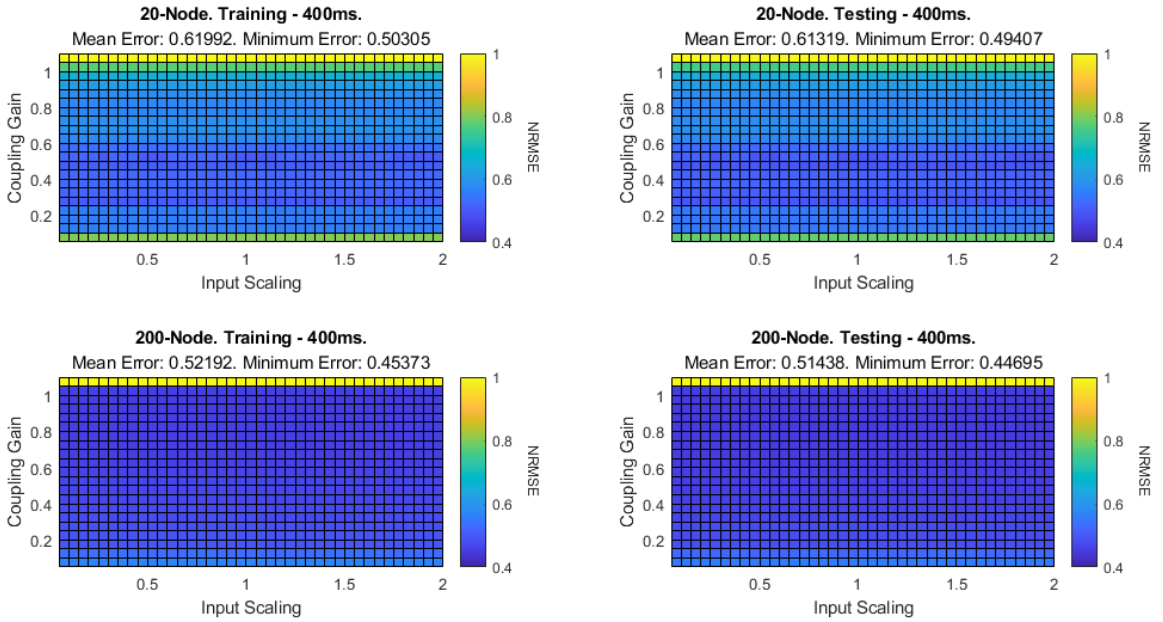

Figure 7.8 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20- and 200-node system running the Santa Fe benchmark without a non-linear function, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1.

Figure 7.8 shows the effect that the input scaling and the coupling gain has within a delay-feedback reservoir system, with the non-linear function removed from the non-linear node, with 20- and 200-node systems running the Santa Fe benchmark.

Both the 20- and 200-node Santa Fe parameter sweeps share many of the same features as their NARMA-10 counterpart; a region of poor performance at the top and at the bottom of the parameter sweeps, and an insensitivity to input scaling.

An interesting difference between the 20-node Santa Fe parameter sweep and its NARMA-10 counterpart is the insensitivity to coupling gain. This is likely due to the Santa Fe benchmark having a weaker reliance on the LMC of the system, enabling it to perform better with less memory within the system; this can be observed by the region of poor performance at the bottom of the parameter sweep, which was also observed within the NARMA-10 benchmark. This also implies that unlike the NARMA-10

benchmark parameter sweeps, the other system dynamics that Santa Fe is utilising to compute with are constant throughout the parameter space; this can be visually confirmed by the lack of gradient within the parameter sweeps.

Table 7.4 shows a minor change in performance within the parameter sweeps shown in figure 7.8, which is to be expected as the increase in virtual nodes amplifies the existing dynamics rather than producing new ones, but a large decrease in performance when compared to the previous experiments. This is to be expected as the main contributor to the non-linear dynamics within the reservoir system has been removed from the non-linear node; this greatly reduces the performance of the non-linear-intensive Santa Fe benchmark.

	Best NRMSE from Chapter 5		Best NRMSE from Chapter 6		Current System Best NRMSE	
Nodes	20	200	20	200	20	200
Training Error	0.320	0.112	0.194	0.085	0.503	0.454
Testing Error	0.300	0.098	0.175	0.087	0.494	0.447

Table 7.4 A table showing the best performing Santa Fe benchmark results from previous chapters and comparing them with the parameter sweeps performed in figure 7.8.

7.4.3 System Metrics

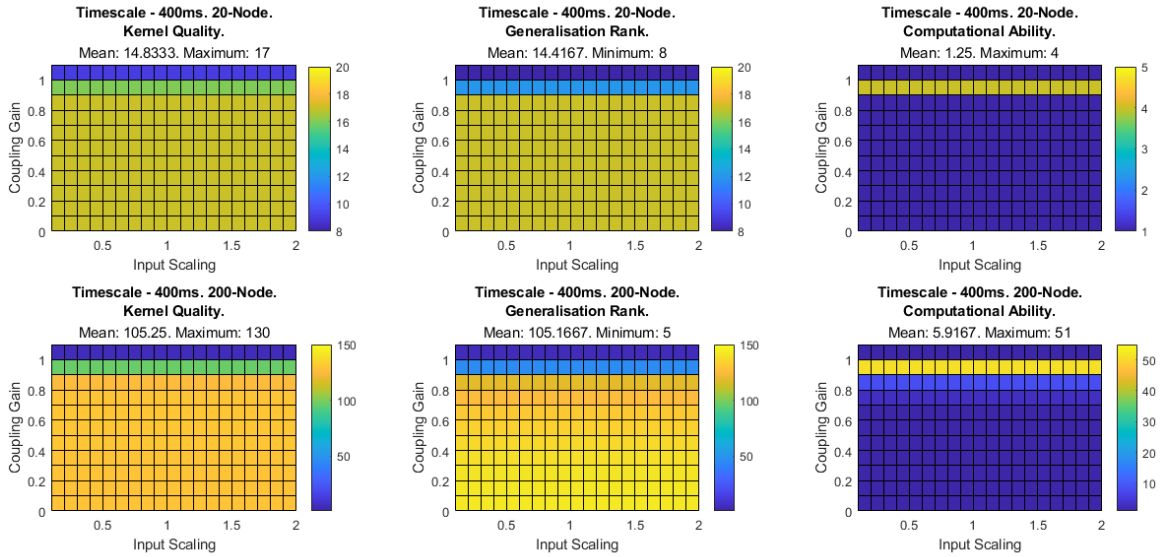

Figure 7.9 Graphs showing the system metrics of a parameter sweep for a 20- and 200-node system without a non-linear function, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1, for the following the system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

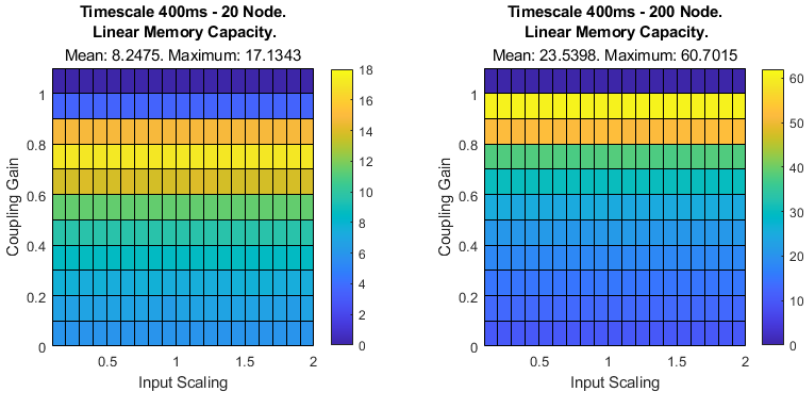

Figure 7.10 Graphs showing the Linear Memory Capacity of a parameter sweep for a 20- and 200-node system without a non-linear function, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1.

Figures 7.9 and 7.10 show the effect that the input scaling and the coupling gain has on the system metrics of a 20- and 200-node delay-feedback reservoir system, without a non-linear function inside the non-linear node.

As predicted, the KQ and GR is relatively constant throughout the entire parameter sweep and is not affected by the input scaling. This is consistent to what was observed within the Santa Fe parameter sweeps, where the performance remained constant throughout most of the parameter space. There is a region where the coupling gain becomes greater than one, where both the KQ and GR decreases, which is once again due to the system become unstable and saturating. The KQ decreases as inputs become harder to separate due to the system operating at the edge of stability, and it appears that the decrease in GR implies the inputs can be better generalised, but all inputs are converging towards saturation.

An interesting observation is that there is a slight decrease in GR as the coupling gain increases within the 200-node system. This is likely caused by the increase in the ratio between new inputs and the delayed output, as when this ratio increases, more information from previous inputs are passed on, allowing the system to train out noise.

Figure 7.10 shows a gradual increase in LMC as the coupling gain increases, until the coupling gain becomes greater than one where the system saturates and the LMC greatly decreases; which is the same behaviour observed within the NARMA-10 parameter plots.

7.5 Results: Complex Poles and Zeros

The genetic representation is modified to allow for complex-valued high-order filters to be generated and evaluated by the evolutionary algorithm to see if additional performance can be gained by adding oscillatory dynamics. The methodology for modifying this representation is described within appendix B.4, with a table of additional evolutionary algorithm parameters shown in table 7.5 for reference. An example of the new genetic encoding for a sixth-order three pole high-order filter, consisting of two complex poles and one complex zero for reference is shown within figure 7.11,

	p0	p1	p2	p3	p4	p5
Pole Active	1	1	0	1	0	1
Pole Value	-4.7	-1.9	-0.5	-1.1	-7.1	-2.7
Damping Ratio	0.21	0	0	0.67	0	0

	z0	z1	z2	z3
Zero Active	0	1	1	0
Zero Value	-2.2	-7.8	-4.5	-3.4
Damping Ratio	0	0	0.23	0

Figure 7.11 A new genetic encoding supporting complex values representing a sixth-order three pole high-order filter, consisting of two complex poles and one complex zero.

The results from this section are compared with the baseline parameter sweeps, shown in section 7.4, and the real-valued high-order filter results within appendix B.

Name	Function	Value
Probability of a Complex Pole	Sets the probability of generating a complex pole during initialisation.	0.2
Probability of a Complex Zero	Sets the probability of generating a complex zero during initialisation.	0.1
σ of Damping Ratio Pole Distribution	The standard deviation of the pole damping ratio Gaussian distribution.	0.1
σ of Damping Ratio Zero Distribution	The standard deviation of the zero damping ratio Gaussian distribution.	0.1

Table 7.5 An additional list of hyper-parameters used within the modified evolutionary algorithm to determine the optimum complex-valued high-order filter to act as a non-linear node within a delay-feedback reservoir, with their default value.

7.5.1 Evolutionary Algorithm Results

NARMA-10

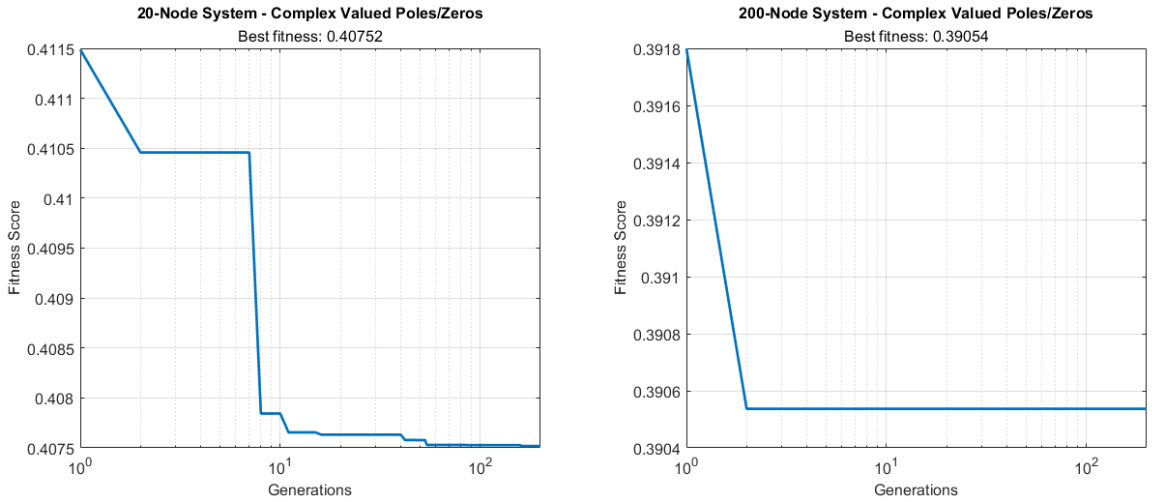

Figure 7.12 Logx graphs showing the best performing fitness of a complex-valued high-order filter acting as a non-linear node after each evolutionary generation for a 20- and 200-node delay-feedback reservoir system evaluating the NARMA-10 computational benchmark.

Figure 7.12 shows the fitness score of the best performing individual at each generation for a 20- and 200-node delay-feedback reservoir system, with a NARMA-10 optimised complex-valued filter as a non-linear node. The x-axis has had a log transform applied so that the fitness transitions can be better viewed.

The fitness graphs show some interesting observations, some of which are common to the real-valued fitness graphs. Both 20- and 200-node systems had a low initial fitness score, showing that a good initial solution was found early on due to the diverse filter dynamics within the initial population pool. In the beginning of the evolutionary process, relatively large changes occur within the fitness score. These changes become much smaller, with only small decremental changes made within the fitness score the longer the evolutionary algorithm is run; this once again suggests that there is a bottleneck within the dynamics of the system. The evolutionary algorithm within the 200-node system was only able to generate one better solution throughout the entire

evolutionary process. This is likely due to some individuals within the population pool having more advantageous genes initially, due to random generation, resulting in no better performing individual to emerge until a beneficial mutation has occurred.

Some insight into how the evolutionary process evolved the filters can be gained by examining the structure of the top three unique best performing high-order filters within the population pool; the top three 20-node filters are shown in table 7.6, and the top three 200-node filters are shown in table 7.7.

All three of the top performing filters within the 20-node system have evolved to be second-order complex-valued filters, with similar timescales and damping ratios. The close grouping suggests that these were likely inherited from the same individual at some point within the evolutionary cycle, suggesting that the evolutionary algorithm was able to narrow down the search for a good pole and zero placement. As the filters are complex, they exist as complex conjugates and therefore have two poles with the same real value, this will create a filter with a much steeper rise time to the dynamics of the time-response when compared to a single pole filter. The imaginary part of the complex filter also introduces oscillatory behaviour to the time-response, although as the damping ratio of the top performing filter is 0.544, the oscillatory behaviour will be weak. The time-domain response of the best performing 20-node filter can be seen within figure 7.13. The figure also shows that the oscillatory behaviour of the filter will persist through several virtual nodes, this will cause an additional sensitivity to the coupling gain as the additional gain will cause the system to saturate earlier.

The evolutionary process optimising the 200-node system appears to have favoured a wide range of different filter dynamics. The best performing filter is second-order with one zero, the second best performing filter is a second-order complex filter, and the third best performing filter is first-order. Examining the training and testing NRMSE values shows that there is a greater difference between the first, second, and third best performing filter errors when compared to the 20-node system. This indicates a dominating individual within the evolutionary process that evolution was unable to

exploit. This could be caused either by the dominating individual being on the edge of stability, so evolved individuals would result in instability, or by the dominating individual being generated with near-optimal dynamics, so it was rare to find a better scoring individual. This is apparent when looking at the fitness graph for the 200-node system, as only one better individual was created over 200 generations. However, as the experiments within this section focus on the use of complex-valued filters, the second best performing 200-node high-order filter will be chosen over the best performing for future experiments.

The second best performing 200-node filter has a damping ratio of 0.4505, which will add a stronger oscillatory behaviour than the 20-node complex high-order filter, with a very fast timescale relative to *theta*, of 75ms. The oscillatory behaviour persists through several virtual nodes, as seen within figure 7.13, spreading information over several virtual nodes, but will also increase the sensitivity to the coupling gain, as the additional gain will cause the system to saturate earlier.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values	Damping Ratio
0.3947	0.4206	-	-0.1906 ± 0.2938j	1554ms	0.39	0.544
0.3947	0.4206	-	-0.2065 ± 0.3033j	1534ms	0.38	0.563
0.3947	0.4211	-	-0.2341 ± 0.3237j	1467ms	0.37	0.586

Table 7.6 A table showing the top three unique best performing complex-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 20-node delay-feedback reservoir evaluating the NARMA-10 benchmark. The pole-zero plots are shown in figure C.1 within appendix C.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values	Damping Ratio
0.3929	0.3882	-0.6100	-1.4365 -1.1893	696ms 841ms	1.74 2.10	-
0.3956	0.3893	-	-3.4282 $\pm$ 5.8234j	75ms	0.19	0.507
0.3981	0.3894	-	-1.4365	696ms	1.74	-

Table 7.7 A table showing the top three unique best performing complex-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 200-node delay-feedback reservoir evaluating the NARMA-10 benchmark. The pole-zero plots are shown in figure C.2 within appendix C.

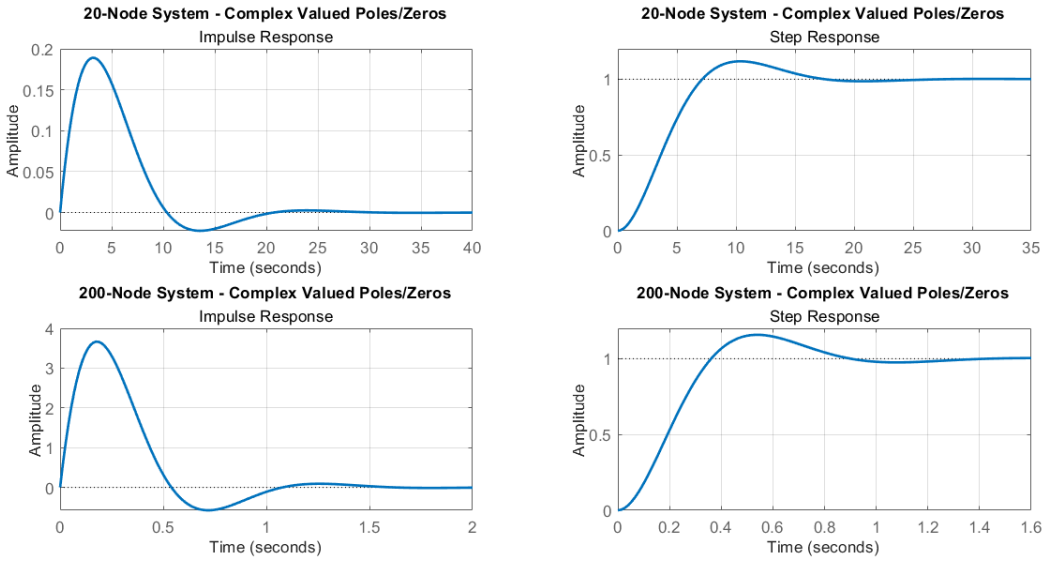

Figure 7.13 Graphs showing the impulse and step response of the best performing complex-valued high-order filters, generated within an evolutionary algorithm, evaluated on the NARMA-10 benchmark within a 20- and 200-node delay-feedback reservoir system.

Santa Fe

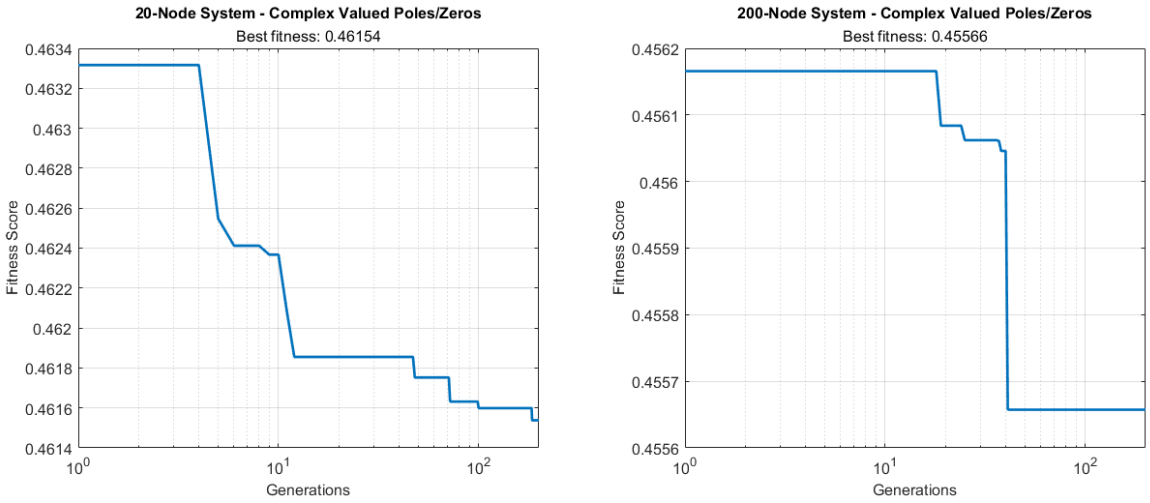

Figure 7.14 Logx graphs showing the best performing fitness of a complex-valued high-order filter acting as a non-linear node after each evolutionary generation for a 20- and 200-node delay-feedback reservoir system evaluating the Santa Fe computational benchmark.

Figure 7.14 shows the fitness score of the best performing individual after each generation for a 20- and 200-node delay-feedback reservoir system, modified with a Santa-Fe optimised complex value filter as a non-linear node; the generation axis has had a log transform applied so that the fitness transitions can be better observed.

The fitness graphs show many of the same observations from previous fitness graphs, such as good initial solutions and small decremental changes, but exhibit some unique differences. The evolutionary process appears to take longer on average to reach a new, better solution, when compared to previous evolutionary runs, with the 200-node taking almost 110 generations before a new better performing individual was found. Much like the 20-node NARMA-10 optimised filter, the initial drop in fitness score is relatively large, but the changes in fitness score become smaller as the evolutionary algorithm progresses. This suggests that the evolutionary algorithm found a pattern, and was able to optimise it over several generations.

In order to better determine how the evolutionary process evolved the filters, the structure of the top three unique best performing high-order filters within the population pool are shown in table 7.8 and table 7.9 for the 20- and 200-node systems respectively.

The top three best performing filters within the 20-node system are all second-order complex filters. The values of the timescale and damping ratio are very similar, indicating that these individuals are likely inherited from the same individual from a previous generation. As all three poles are complex, they are complex conjugates meaning they have a fast rise time and oscillatory behaviour. The best performing filter has a damping ratio of 0.481, which will give mild oscillatory dynamics to the time-response of the filter; which can be seen within the 20-node system within figure 7.15. The transient behaviour of the best performing filter is spread over approximately three virtual nodes, as the 20-node system has a θ of 4 s. This will both increase the memory capacity of the system, as the strong rise time allows for a strong connection between the adjacency node and others, and increase the sensitivity to coupling gain, as the additional gain introduced to the system causes the system to saturate.

Similarly to the 20-node system, the top three best performing filters within the 200-node system are all second-order complex filters. Given the complex nature of the filters, all three filters will have a fast rise time and exhibit oscillatory behaviour. The best performing filter has a damping ratio of 0.458, which will give slightly faster oscillatory behaviour than the 20-node best performing filter; this is shown in figure 7.15. Given that the transient behaviour is spread over approximately several virtual nodes, an increase to the coupling gain sensitivity should be expected.

An interesting observation is that the timescales of the top three 200-node filters are approximately one tenth of the 20-node filters. This causes the connectivity between virtual nodes to be similar, as shown by comparing the ρ values of the filters.

This choice seems to suggest that the evolutionary algorithm found that the system performance can not only be increased by changing the value of ρ, which affects the

memory of the system, but by changing the damping ratio. Given that the training and testing NRMSE values have not significantly changed, this implies that the damping ratio has more of an effect on the memory of the system than the non-linearity.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values	Damping Ratio
0.4666	0.4565	-	-0.2522 ± 0.4597j	917ms	0.23	0.481
0.4666	0.4566	-	-0.3216 ± 0.4722j	985ms	0.25	0.563
0.4676	0.4574	-	-0.3006 ± 0.4650j	981ms	0.25	0.543

Table 7.8 A table showing the top three unique best performing complex-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 20-node delay-feedback reservoir evaluating the Santa Fe benchmark. The pole-zero plots are shown in figure C.3 within appendix C.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values	Damping Ratio
0.4597	0.4517	-	-2.8227 ± 5.4830j	74ms	0.19	0.458
0.4630	0.4542	-	-4.0193 ± 5.5104j	86ms	0.22	0.589
0.4638	0.4551	-	-5.8950 ± 8.3477j	56ms	0.14	0.577

Table 7.9 A table showing the top three unique best performing complex-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 200-node delay-feedback reservoir evaluating the Santa Fe benchmark. The pole-zero plots are shown in figure C.4 within appendix C.

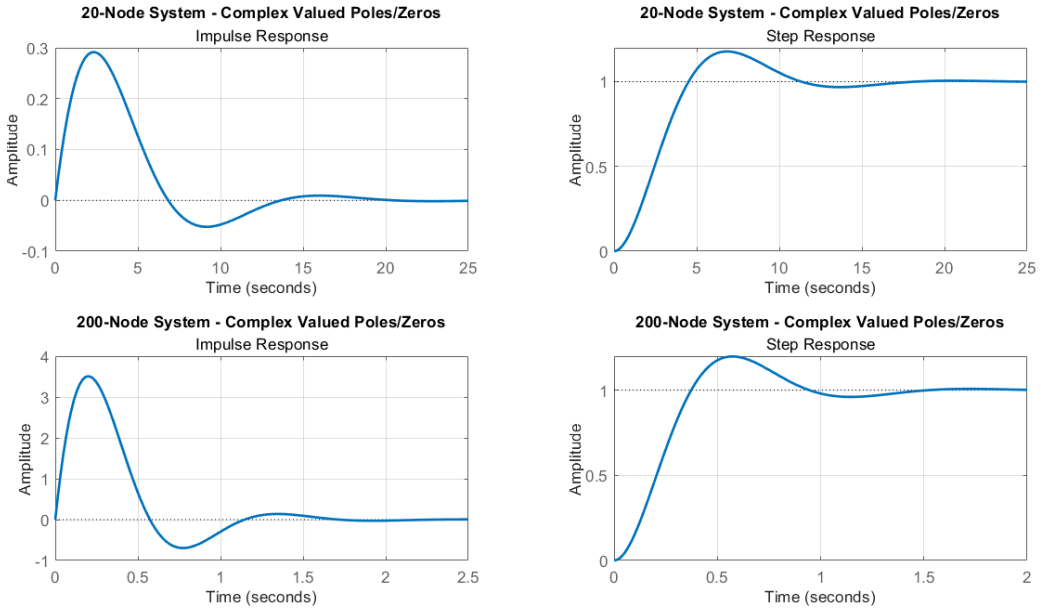

Figure 7.15 Graphs showing the impulse and step response of the best performing complex-valued high-order filters generated within an evolutionary algorithm on a 20- and 200-node delay-feedback reservoir evaluated the Santa Fe benchmark.

7.5.2 NARMA-10 Filter Analysis

Parameter Sweeps

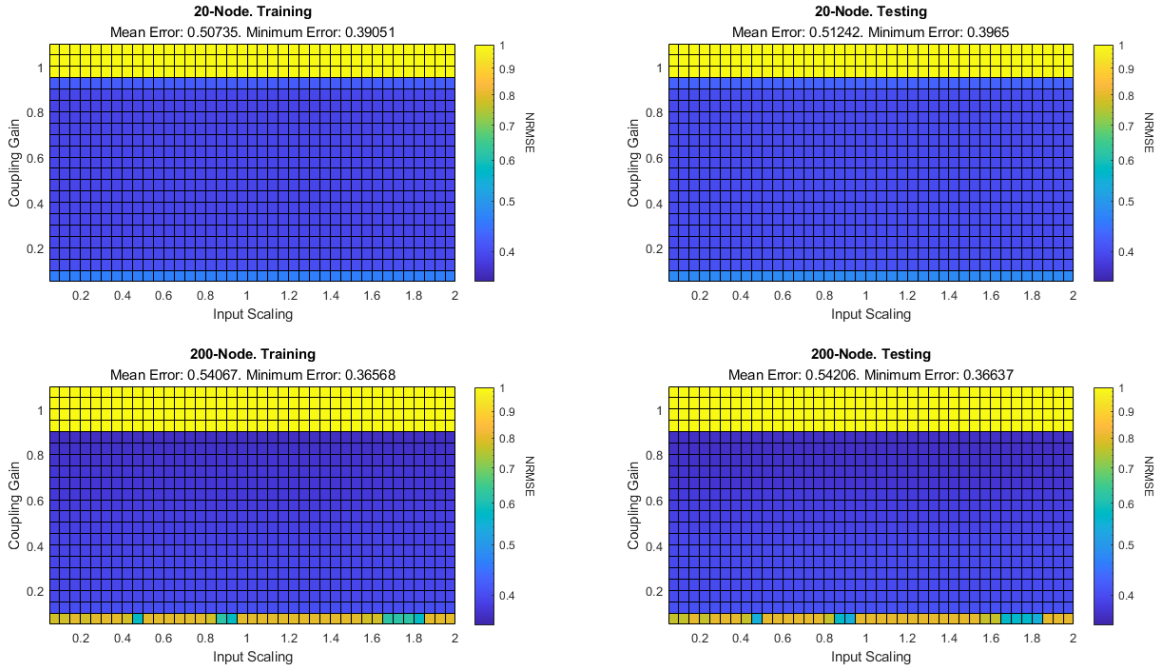

Figure 7.16 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20- and 200-node system containing the best performing complex-valued high-order filter, as shown in tables 7.6 and 7.7, used as a non-linear node within a delay-feedback reservoir evaluating the NARMA-10 benchmark; the input scaling range is between 0.05 and 2, and the coupling gain range is between 0.05 and 1.1.

Figure 7.16 shows the effect that the input scaling and the coupling gain have on a 20- and 200-node delay-feedback reservoir, utilising a complex-valued non-linear node, evaluating the NARMA-10 benchmark. To better view subtle changes within the NRMSE, a logarithmic scale is used for the colour map.

The parameter sweeps have very similar features to the real-valued parameter sweeps shown previously, indicating that there are minor differences within the system dynamics. A minor difference between the complex-valued 20-node system and the real-valued 20-node system shown previously, is that the complex-valued 20-node system has better performance when the coupling gain is low, but worse performance when the

coupling gain is high. Both of these effects are due to the oscillatory behaviour added to the system. The additional gain causes the system to saturate with smaller values of coupling gain, thus increasing the system's sensitivity to coupling gain. However when the coupling gain is small, the output of the filter is significantly attenuated, creating a loss of memory within the system.

Within the complex-valued 200-node system there appears to be a sensitivity to the input scaling when the coupling gain is approximately 0.05, with a sporadic change in NRMSE values. As observed within previous parameter sweeps, this area is known to produce significantly worse NRMSE values due to the very small signal values within this region. This suggests that the sporadic NRMSE values are due to training instability, rather than a sensitivity to input scaling.

Within table 7.10, the best performing NARMA-10 results from previous chapters and this chapter are shown. The table shows that the complex-valued 20-node system performed the same as the real-valued 20-node system, but much better than the 20-node benchmark system from previous experiments. While the complex-valued 200-node system performed better than the real-valued 200-node system, had equal performance to the 200-node baseline system, and performed much worse than other previous experiments.

Comparing these results initially indicates that the addition of complex-valued poles and zeros does not greatly change the non-linear dynamics within a system.

	Best NRMSE from Chapter 5		Best NRMSE from Chapter 6		Best NRMSE from Baseline		Best NRMSE for Real Values		Current System Best NRMSE	
Nodes	20	200	20	200	20	200	20	200	20	200
Training Error	0.491	0.306	0.484	0.271	0.514	0.367	0.391	0.397	0.391	0.366
Testing Error	0.494	0.308	0.483	0.280	0.522	0.362	0.367	0.362	0.397	0.366

Table 7.10 A table showing the best performing NARMA-10 results from previous chapters, the baseline and real-valued pole and zero results for a high-order non-linear node, and parameter sweeps performed in figure 7.16.

System Metrics

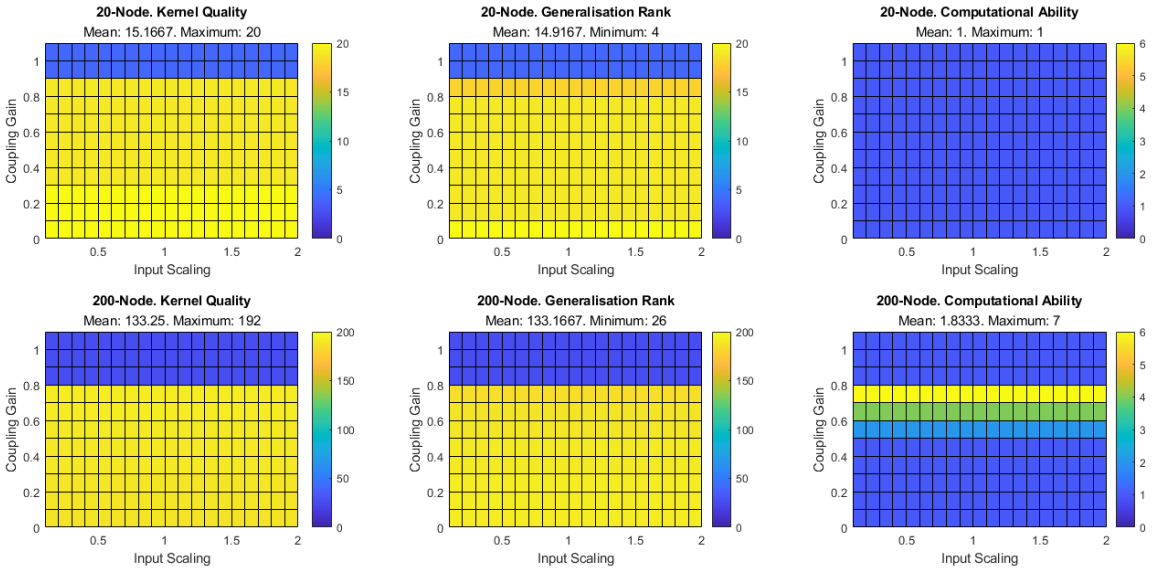

Figure 7.17 Graphs showing the system metrics of a parameter sweep for a 20- and 200-node system, with different optimised complex-valued high-order filters used as non-linear nodes within a delay-feedback reservoir evaluating the NARMA-10 benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1, for the following system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

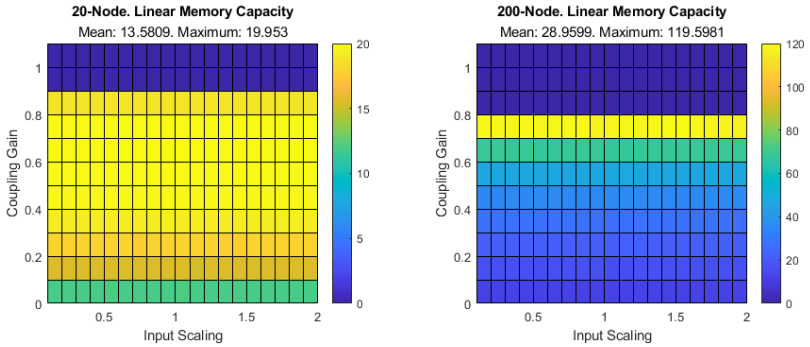

Figure 7.18 Graphs showing the Linear Memory Capacity of a parameter sweep for a 20- and 200-node system, with different optimised complex-valued high-order filters used as non-linear nodes within a delay-feedback reservoir evaluating the NARMA-10 benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1.

Figures 7.17 and 7.18 show the effect that the input scaling and the coupling gain have on the system metrics of a 20- and 200-node delay-feedback reservoir, modified with a NARMA-10 optimised complex-valued high-order filter as the non-linear node.

The complex-valued system metrics are almost identical to the real-valued system metrics shown previously; with both systems having a high KQ and GR throughout the parameter space. However, the complex-valued systems have some minor changes when compared to their real-valued counterpart.

Unlike the real-valued systems, the complex-valued systems have a slight decrease in KQ and GR as the coupling gain increases; which can be observed within the CA of the 200-node system. This is once again likely to how the information is non-linearly spread through the virtual nodes, creating complex virtual topologies that likely have sensitivity to coupling gain. The complex-valued systems also exhibit an increase in coupling gain sensitivity due to the oscillatory behaviour, causing the system to saturate with a smaller coupling gain value.

The LMC parameter sweeps of the complex-valued systems show that the 20-node system has a similar behaviour to the LMC parameter sweep of the real-valued 20-node system, but the LMC of the complex-valued 200-node system significantly outperforms the real-valued 200-node system.

Table 7.11 shows the best performing system metrics from this chapter and previous experiments, allowing a comparison of the two systems within figures 7.17 and 7.18 to previous experiments. The table shows that the complex-valued 20-node system is outperformed by the real-valued 20-node system in terms of having a slightly lower GR and higher CA, but the complex-valued 20-node system a marginally higher LMC. While the complex-valued 200-node system outperforms the real-valued 200-node system in terms of a higher KQ, CA, and LMC, but suffers by having a high GR. When comparing the high-order filter non-linear node systems to previous experiments, the complex-valued filters tend to have a higher KQ, GR, and LMC; but a much lower CA. The main advantage of the high-order filter non-linear node is that they have

the potential to reach a much higher LMC and a significantly decreased sensitivity to input scaling and coupling gain when compared the previous experiments.

This implies that although the non-linear dynamics are rather poor within the evaluated high-order filter non-linear node systems, the filters can be optimised for very high LMC rather than non-linearity.

	Best Metrics from Chapter 5		Best Metrics from Chapter 6		Best Metrics from Baseline		Best Metrics for Real Values		Current System Best Metrics	
Nodes	20	200	20	200	20	200	20	200	20	200
Kernel Quality	20	190	17	80	17	130	20	188	20	192
Generalisation Rank	1	1	1	1	8	5	3	10	4	26
Computational Ability	19	166	16	119	4	51	5	1	1	7
Linear Memory Capacity	12.3	37.5	15.7	50.7	17.1	60.7	19.8	76.8	20.0	119.6

Table 7.11 A table showing the best performing metrics from previous chapters, the baseline and real-valued pole and zero results for a high-order non-linear node, and parameter sweeps performed in figures 7.17 and 7.18.

7.5.3 Santa Fe Filter Analysis

Parameter Sweeps

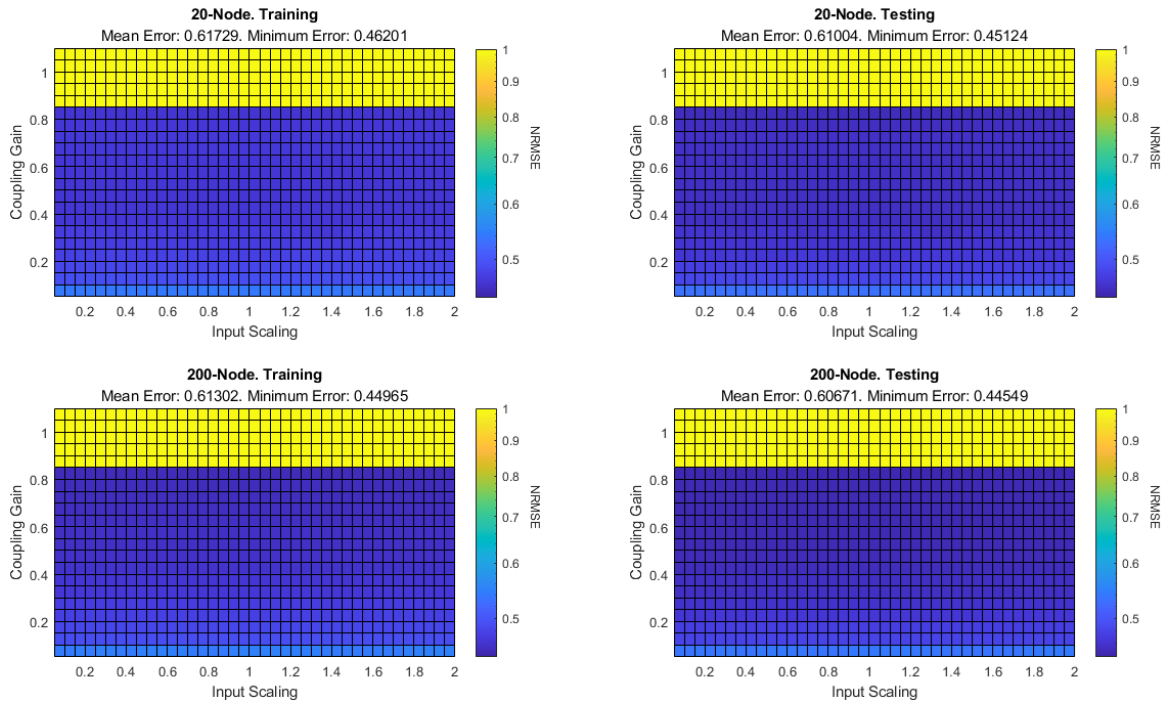

Figure 7.19 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20- and 200-node system containing the best performing complex-valued high-order filter, as shown in tables 7.8 and 7.9, used as a non-linear node within a delay-feedback reservoir evaluating the Santa Fe benchmark; the input scaling range is between 0.05 and 2, and the coupling gain range is between 0.05 and 1.1.

Figure 7.19 shows the effect that the input scaling and the coupling gain have on a 20- and 200-node delay-feedback reservoir evaluating the Santa Fe benchmark, which has been modified to have a complex-valued filter as the non-linear node. The colour map displaying the NRMSE error of the systems has had a log transform applied to better view any subtle changes.

The parameter sweeps are almost identical to the parameter sweeps within the Santa Fe optimised real-valued filters, with the only difference being that the region of saturation is greater within the complex-valued filters, which is due to the addition of the oscillatory behaviour within the complex-valued systems.

An interesting observation is that both the 20- and 200-node systems have similar performance, with the difference in training and testing error being approximately 0.012 and 0.006 respectively. This shows that the bottleneck is independent of the number of virtual nodes within the system, suggesting a lack of non-linearity within the dynamics of the system.

The table 7.12 shows the best performing Santa Fe results from previous experiments. This table shows that the performance of the real-valued filters is almost the same as the complex-valued filters, but slightly better than the baseline parameter sweeps, and in all other experiments the performance is significantly worse; this suggests three things. First, that both the Santa Fe optimised real and complex value filters exhibit the same bottleneck in system dynamics, which due to the requirements of the Santa Fe benchmark is non-linearity. Second, the use of a high-order filter does add some non-linearity to the system when compared to a first-order filter. Third, the use of complex-valued poles does not increase the amount of non-linearity within the system.

	Best NRMSE from Chapter 5		Best NRMSE from Chapter 6		Current System Best NRMSE		Best NRMSE for Real Values		Current System Best NRMSE	
Nodes	20	200	20	200	20	200	20	200	20	200
Training Error	0.320	0.112	0.194	0.085	0.503	0.454	0.461	0.452	0.462	0.450
Testing Error	0.300	0.098	0.175	0.087	0.494	0.447	0.451	0.445	0.451	0.445

Table 7.12 A table showing the best performing Santa Fe results from previous chapters, the baseline and real-valued pole and zero results for a high-order non-linear node, and parameter sweeps performed in figure 7.14.

System Metrics

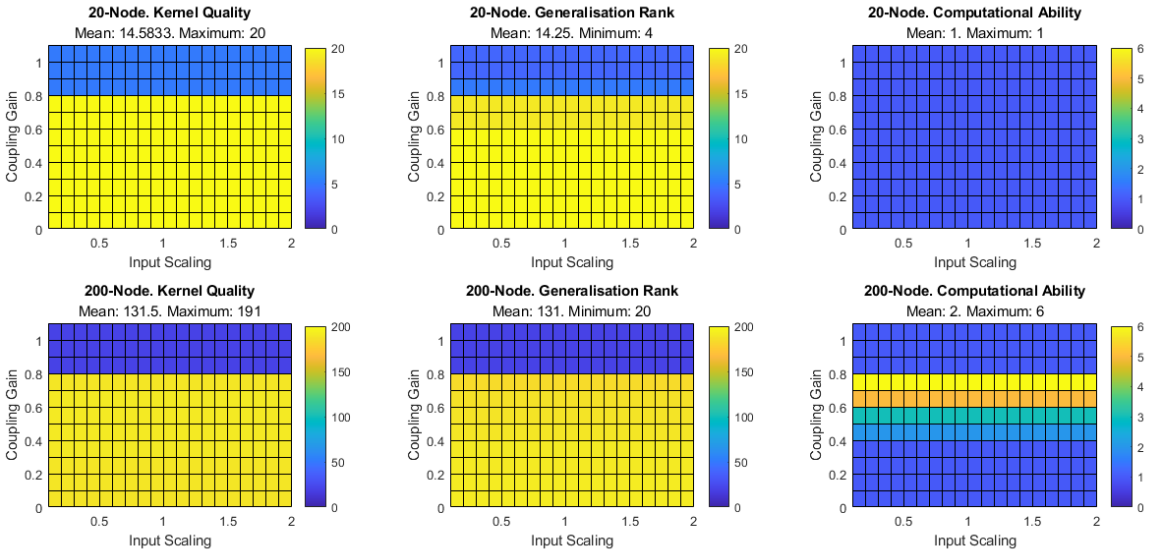

Figure 7.20 Graphs showing the system metrics of a parameter sweep for a 20- and 200-node system, with different optimised complex-valued high-order filters used as non-linear nodes within a delay-feedback reservoir evaluating the Santa Fe benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1, for the following system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

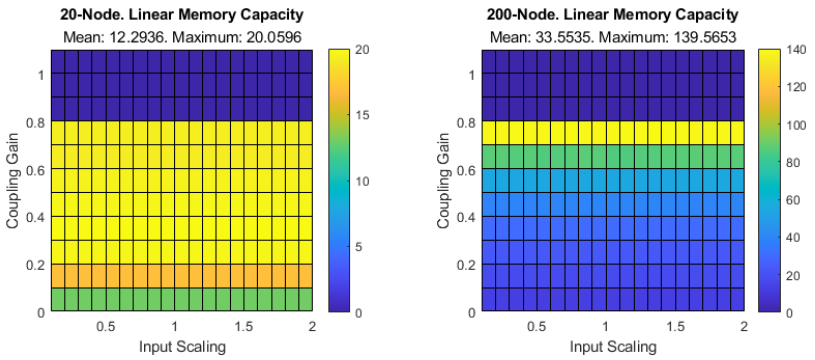

Figure 7.21 Graphs showing the Linear Memory Capacity of a parameter sweep for a 20- and 200-node system, with different optimised complex-valued high-order filters used as non-linear nodes within a delay-feedback reservoir evaluating the Santa Fe benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1.

Figures 7.20 and 7.21 show the effect that the input scaling and the coupling gain have on the system metrics of a 20- and 200-node delay-feedback reservoir, modified with a Santa Fe optimised complex-valued high-order filter as the non-linear node.

The system metric parameter sweeps are similar to the system metric parameter sweeps within the Santa Fe optimised real-valued filters, both systems have a high KQ and GR which is almost constant throughout the parameter space, but with a few exceptions.

Within the 200-node system, the GR has a minor dependency on the coupling gain, with the GR decreasing as the coupling gain increases; which can be observed within the CA of the system. As previously discussed, this dependency is most likely due to how information is being non-linearly mapped over several virtual nodes. If the information has been non-linearly mapped over several virtual nodes, then the system is able to better reject noise as similar information is reinforced and random information is lost over time; which could lead to the better generalisation of inputs.

Table 7.13 shows the best performing system metrics for previous experiments. Comparing the complex and real-valued filters shows the complex-valued filters generally outperforming the real-valued filters, with the exception of the 200-node complex-valued filter having a much higher GR than its real-valued counterpart. A further comparison with previous experiments show a strong KQ, but a poor GR and CA, in both 20- and 200-node systems. Despite the poor GR and CA, the complex-valued filters excel within LMC, where the LMC of the 200-node system is double that of the next best system. This further confirms that complex-valued filters do not add additional non-linear dynamics when compared to real-valued filters, but they can be optimised to greatly increase the amount of memory a system has.

	Best Metrics from Chapter 5		Best Metrics from Chapter 6		Best Metrics from Baseline		Best Metrics for Real Values		Current System Best Metrics	
Nodes	20	200	20	200	20	200	20	200	20	200
Kernel Quality	20	190	17	80	17	130	20	160	20	191
Generalisation Rank	1	1	1	1	8	5	7	8	4	20
Computational Ability	19	166	16	119	4	51	3	1	1	6
Linear Memory Capacity	12.3	37.5	15.1	50.7	17.1	60.7	20	72.6	20	139.6

Table 7.13 A table showing the best performing metrics from previous chapters, the baseline and real-valued pole and zero results for a high-order non-linear node, and parameter sweeps performed in figures 7.20 and 7.21.

7.6 Discussion

From the results shown in sections 7.4, 7.5, and appendix B, some useful conclusions can be made regarding: the effectiveness of the evolutionary algorithm; the effects of using a real or a complex-valued high-order filter; and if a high-order filter could be better used as an integration stage. This section aims to explore these conclusions further, so that sub-hypothesis 4 can be verified.

7.6.1 Effectiveness of the Evolutionary Algorithm

Within section 7.2 of this chapter, an evolutionary algorithm was created to help determine the optimal parameters of a high-order filter, so that the typical non-linear node within a delay-feedback reservoir could be replaced with a single high-order filter. In order to determine the fitness score of an individual, either the NARMA-10 or Santa Fe computational benchmark was run to evaluate the performance of an individual based upon its training and testing NRMSE values. The NARMA-10 and Santa Fe benchmarks were used as they require different computational dynamics to achieve the best performance. The aim was to let the benchmark guide the fitness, so that the evolutionary algorithm could find solutions that exploit the different computational

dynamics within the reservoir system. While this worked to some extent, the significant bottleneck of non-linearity within the filters made it very difficult for any other metrics, except memory, to be optimised, which is indicative of solving a multiple-objective problem with a single objective algorithm. A better solution would be to use a multi-objective evolutionary algorithm to generate fitness scores for the KQ, GR, and LMC.

This would give significantly more insight into how the evolutionary process affects the system metrics, allowing for the optimisation to create a more intelligent approach, as filters are evaluated on three metrics, rather than attempting to infer filter characteristics from evaluating a system-sensitive benchmark task. This allows the evolutionary algorithm to explore different trade-offs between system metrics, finding a range of solutions which could be general, creating a high-order filter that has good metric scores across all three metrics, or metric specific solutions where only one metric has a good performance score. It would also give further insight into which filter parameters provide which system dynamics, leading to some high-order filter design rules for solving computational tasks with specific dynamics. Several different multi-objective evolutionary algorithms have been used within reservoir computing networks, such as Non-dominated Sorting Genetic Algorithm II (NSGA-II) [133] and Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [134], to optimise a reservoir system to optimally solve a particular computational task. However, this is outside the scope of this thesis.

7.6.2 Real- and Complex-Valued High-Order Filters

Within sections 7.4, 7.5, and appendix B, the effect that a real- or complex-valued high-order filter non-linear node has on system performance, when compared to a single-order non-linear node and previous experiments, was investigated. It was shown that a high-order filter does provide a small amount of non-linearity, but not enough for computational benchmarks such as Santa Fe. The amount of non-linearity a high-order

filter can provide appears to be independent of the filter being real- or complex-valued, with real and complex-valued systems having very similar training and testing errors. However, one significant difference between a real- or complex-valued high-order filter, is the amount of LMC within the system; as it was shown that generally, a complex-valued high-order filter was able to reach a higher LMC than a real-valued high-order filter. This increase in LMC appears to be due to how information is passed between virtual nodes, as the high-order filters with high LMC showed time-response plots that had the non-linear transient persist through several virtual nodes.

Within chapter 5 it was found that the LMC within a delay-feedback reservoir system was dependent on the timescale of the system; the transient response of the system defines the topology of the virtual nodes. As the transient response of a filter can be designed, several different virtual node topologies can be created, each providing different dynamics and behaviours to a system. An example of an high-order filter that could provide more varied dynamics and topology is shown within figure 7.22, a seventh-order single complex pole three zero filter; the black dotted line indicates a θ of 4 s.

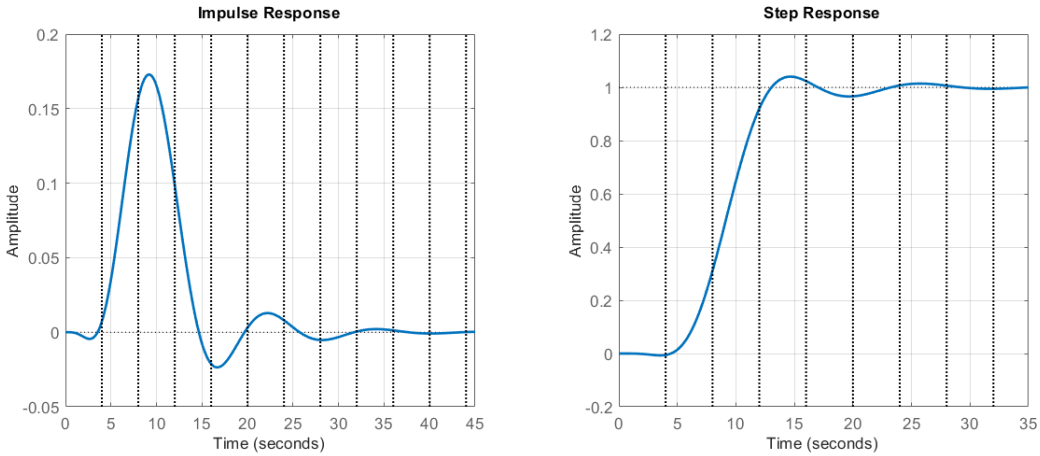

Figure 7.22 Graphs showing the impulse and step response of a seventh-order single complex pole three zero filter, designed to create a topology where the first virtual node is skipped; the black dotted line indicates every θ (4 s).

The filter in figure 7.22 may create a topology where the first virtual node is skipped, with the next "adjacent" node being the following virtual node. This would create a delay-feedback reservoir system where the information within the adjacent node contains the peak transient response of the previous node. This filter could be extended to then allow a separation of information, between odd and even virtual nodes for example, allowing either multiple delay-feedback reservoirs to be run within the same virtual nodes, or to store redundancy information which can be used for fault tolerance.

7.6.3 High-Order Filter as Integrator Function

During the analysis of the real and complex value high-order filters within sections 7.4, B, and 7.5, it became apparent that a high-order filter non-linear node either did not provide enough non-linear dynamics, or provided the wrong type of non-linear dynamics. The non-linearity within the high-order filter non-linear node is implemented exclusively through the time-response of the filter. Whereas it was observed that the Mackey-Glass non-linear function contains two parts; a non-linear transform and an attractor. This suggests that a mathematical non-linear transform is not enough, a non-linear attractor behaviour is also necessary to provide the type of non-linearity that is useful for computation. This poses the question: can a high-order filter replace the integration stage within the typical non-linear node, to supplement the dynamics of the non-linear function?

A cursory review of this question is investigated in order to determine if this is a worthwhile avenue to explore in future work. The experimental model within figure 7.6 was modified to once again include the Mackey-Glass non-linear function; the new experimental model is shown in figure 7.23.

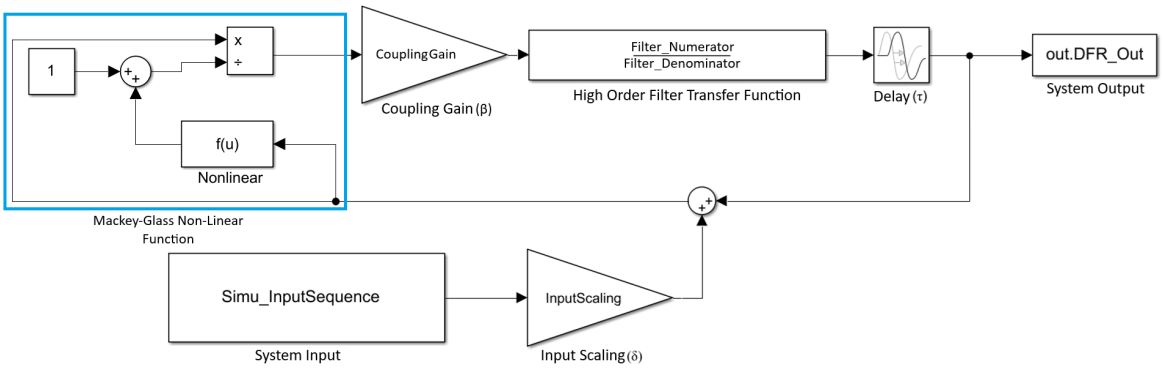

Figure 7.23 A schematic of a delay-feedback reservoir using the Mackey-Glass dynamical system as a non-linear function and a high-order transfer function as integration stage, created within Simulink 23.2.

Although the implemented evolutionary algorithm is not optimised for finding or optimising high-order filter parameters with the Mackey-Glass non-linear function within the non-linear node, the evolutionary algorithm is used to find some approximate filter parameters. The evolutionary algorithm was configured to find and optimise complex-valued high-order filters, using the methodology and parameters outlined within sections 7.3 and 7.5. The Mackey-Glass exponent is set to a value of 2, which was the best performing value within chapter 6.

The evolutionary algorithm is run for 180 generations, with the top performing complex-valued high-order filters within the population pool shown in table 7.14.

Computational Benchmark	Node Size	Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales
NARMA-10	20	0.6275	0.6327	-1.2436	-2.164 -3.556 -3.151	462ms 281ms 317ms
	200	0.3094	0.3307	0.5981	-0.708 ± 0.893j -2.67	545ms 374ms
Santa Fe	20	0.3271	0.3185	-	-1.825 -4.613	547ms 216ms
	200	0.1488	0.1458	-0.6109	-5.045 -1.988	198ms 503ms

Table 7.14 A table showing top performing complex-valued high-order filters, optimised through an evolutionary process, within the integration stage of a delay-feedback reservoir.

As the evolutionary algorithm has not been optimised, the general structure of the high-order filters are of interest rather than their specific dynamics.

The four generated high-order filters are all at least second-order, with all but the 20-node Santa Fe system having a fast pole. The timescales of each system are relatively spread out, a pattern that was also observed within previous evolutionary experiments which appears to aid in additional system memory. An interesting observation is the training and testing NRMSE values, with all but the 20-node NARMA-10 system performing significantly better than previous experiments where only a high-order filter was used as the non-linear node. This strongly suggests that a high-order filter can supplement the dynamics of the non-linear function.

In order to further explore the feasibility of a supplementary high-order filter, a parameter sweep of the input scaling and coupling gains will be performed for each high-order filter, evaluating their corresponding benchmark. This allows for a visualisation of the parameter sensitivity, and gives some insight into the maximum achievable performance of the systems. Table 7.15 shows the parameter values and ranges for the proposed parameter sweeps.

Name	Symbol	Value (20-Node)	Value (200-Node)
Coupling Gain	β	0.05-2	0.05-2
Input Scaling Factor	δ	0.05-2	0.05-2
Mackey-Glass Exponent	n	2	2
Time Delay	τ	80 s	80 s
Masking Signal Period	θ	4 s	0.4 s

Table 7.15 A table showing parameter values of a delay-feedback reservoir, utilising a high-order filter as the integration stage, during the input scaling and the coupling gain parameter sweep.

Figure 7.24 shows the effect that the input scaling and the coupling gain has on a 20- and 200-node delay-feedback reservoir system, with a modified non-linear node to include both the Mackey-Glass non-linear function and a high-order integration stage, evaluating the NARMA-10 and Santa Fe benchmarks.

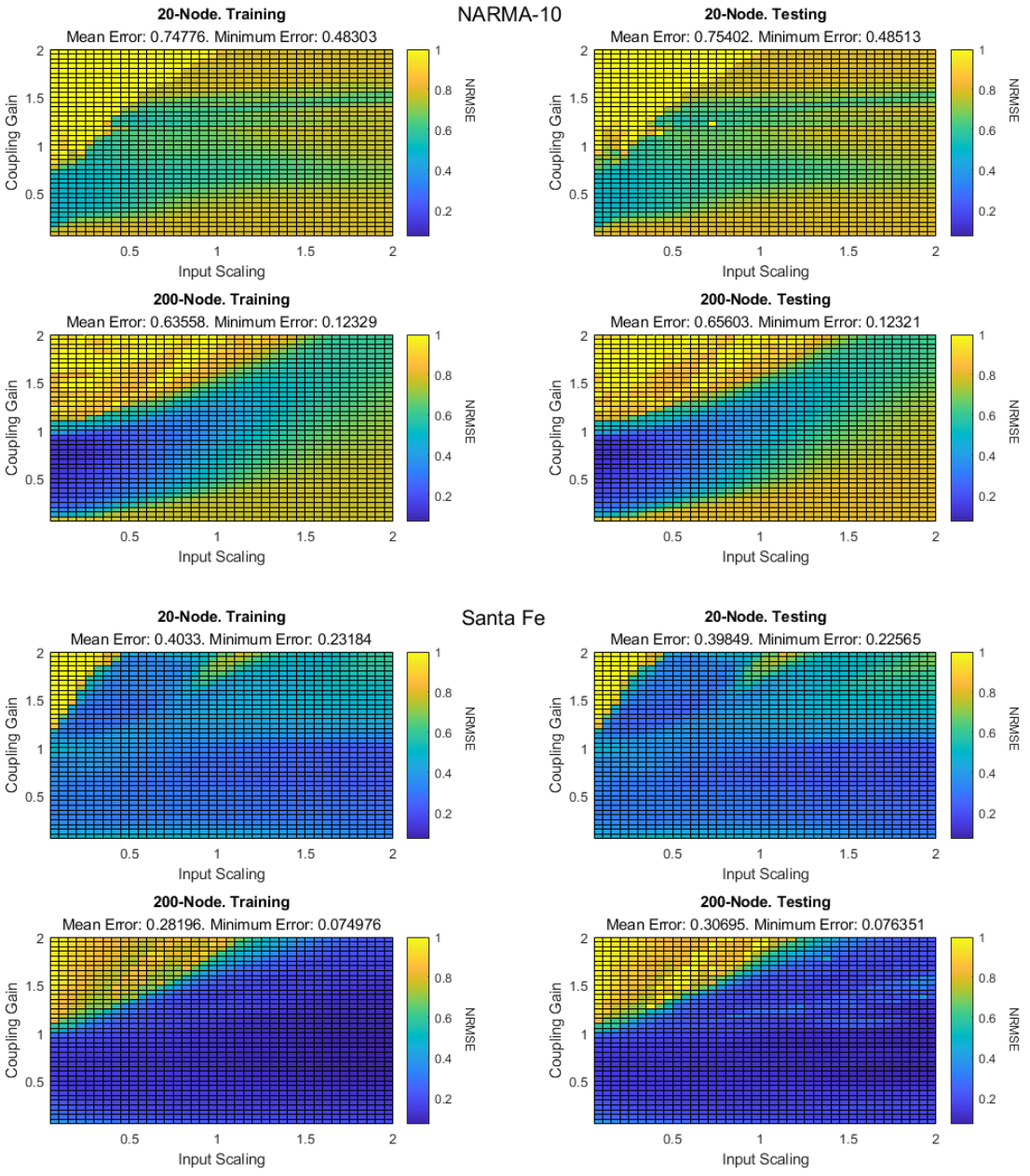

Figure 7.24 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20- and 200-node system utilising the best performing high-order filter as the integration stage of a delay-feedback reservoir for the NARMA-10 and Santa Fe benchmark; the input scaling and the coupling gain range is between 0.05 and 2.

The NARMA-10 parameter sweeps have some interesting observations. The region of best operation is once again visible within the parameter space, with the region of best operation in the bottom left of the parameter space for both the 20- and 200-node systems. The insensitivity to input scaling is no longer present, with the parameter sweeps looking similar to the Mackay-Glass exponent parameter sweeps within chapter 6. The most significant change is the training and testing errors of the system, with the minimum testing NRMSE of the 20-node system performing fairly average at 0.483, but the 200-node system has excellent performance with a minimum testing error of 0.123.

Similar observations can be made within the Santa Fe parameter sweeps. The sensitivity to input scaling is significantly better when compared to NARMA-10, with only the upper left corner of the parameter space having significantly poor performance. The two typical regions of best performance for the Santa Fe benchmark can be once again observed, with the first region being within the upper middle of the parameter space, and the second being within the middle right. The training and testing errors of the system are significantly better than with the high-order filter non-linear node, with the minimum testing NRMSE of the 20-node system being 0.232, and the minimum testing error of 0.075 for the 200-node system.

The best performing results from previous experiments and the results within figure 7.24 are shown in table 7.16.

Benchmark	Error	Best NRMSE from Chapter 5		Best NRMSE from Chapter 6		Best NRMSE from Chapter 7		Current System Best NRMSE	
		20	200	20	200	20	200	20	200
NARMA-10	Training Error	0.491	0.306	0.484	0.271	0.391	0.367	0.483	0.124
	Testing Error	0.494	0.308	0.483	0.280	0.397	0.362	0.485	0.123
Santa Fe	Training Error	0.320	0.112	0.194	0.085	0.461	0.450	0.232	0.075
	Testing Error	0.300	0.098	0.175	0.087	0.451	0.445	0.226	0.076

Table 7.16 A table showing the best performing results from this and previous chapters, and parameter sweeps performed in figure 7.24.

The table clearly shows that the addition of a supplementary high-order filter greatly increases the performance of the 200-node system, with the NARMA-10 200-node system showing a very large increase in performance compared to other systems, but a slight decrease in performance for the 20-node systems. This is most likely due to the lack of optimisation within the evolutionary algorithm when evaluating a delay-feedback reservoir with a non-linear function, producing sub-optimal 20-node filters.

This preliminary exploration into using a high-order filter to replace the integration stage within a typical non-linear node supports the idea that the high-order filter can supplement the dynamics of the non-linear function. This concept paired with a multi-objective evolutionary algorithm to find better high-order filters, could lead to more powerful delay-feedback reservoirs, and as high-order filters are easy to realise within analogue hardware, does not greatly increase the hardware complexity of the reservoir system.

7.7 Summary

At the beginning of this chapter, an alternative approach to the structure of a non-linear node within a delay-feedback reservoir was proposed in order to answer sub-hypothesis 4: *Replacing the non-linear function and integration stage of a delay-feedback reservoir with a higher order filter can lead to an increase in performance while simplifying the hardware architecture of a delay-feedback reservoir system model.*

In order to answer this hypothesis, an evolutionary algorithm was created in order to determine the optimal parameters for a high-order filter, so it can be used within an all-in-one non-linear node.

7.7.1 Effectiveness of a High-Order Filter Non-Linear Node

Before the evolutionary algorithm was run, a baseline was established to determine the performance of a delay-feedback reservoir system without the Mackey-Glass non-linear function.

This was done by removing the Mackey-Glass non-linear function from the experimental model within chapter 6, using only the best performing first-order filter determined within chapter 5, and performing parameter sweeps. The baseline results showed a significant decrease within system performance, particularly for the Santa Fe computational benchmark. The evolutionary algorithm was then run to generate optimal real-valued high-order filters, optimised using the NARMA-10 and Santa Fe benchmarks separately to guide the fitness of the evolutionary algorithm as NARMA-10 requires high memory, and Santa Fe requires high non-linearity. The results for the real-valued high-order filter showed only slightly better performance than the benchmark, in some cases only matching it. It became apparent that a real-valued high-order filter non-linear node only provided time-domain non-linearity and not state-space non-linearity (as in an attractor behaviour) to the reservoir system. Therefore the evolutionary algorithm was modified to handle complex-valued high-order filters, with the hope that the oscillatory behaviour within complex filters to further add non-linear dynamics of the reservoir system.

The results for the complex-valued high-order filter showed that complex-valued poles do not add any additional non-linear dynamics to the reservoir system, with the complex-valued high-order filters having a similar performance to the real-valued high-order filters. While the performance did not change, the system's LMC greatly increased. This suggests that complex-valued high-order filters are able to achieve a higher memory due to how information is distributed throughout the virtual nodes of the system.

Further experiments were then performed to see if a high-order filter can replace the integration stage to supplement the dynamics of the non-linear function. A cursory

review showed a large improvement within the 200-node systems and a slight decrease within the 20-node systems, which is likely due to the evolutionary algorithm producing non-optimal filters, supporting the idea of a supplemental high-order filter integration stage.

7.7.2 Conclusions

The experiments within this chapter show that removing the non-linear function and replacing the integration stage with a high-order filter, in order to simplify the non-linear node of a delay-feedback reservoir, results in a reduction of performance within the reservoir system due to the lack of state-space non-linearity; disproving the validity of sub-hypothesis 4. Therefore the following key conclusions are made within this chapter:

- A high-order filter non-linear node without a non-linear attractor reduces the performance within a delay-feedback reservoir by removing state-space non-linear dynamics.
- Both real-valued and complex-valued high-order filters provide the same amount of non-linearity within a system.
- The use of a complex-valued high-order filter over a real-valued high-order filter allows a reservoir system to achieve a significantly higher LMC.
- A high-order filter can replace the integration stage within a typical non-linear node to supplement the dynamics of a non-linear function to achieve a greater performance.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

At the beginning of this thesis, a hypothesis was postulated: *Tuning the non-linear node within a physical delay-feedback reservoir system provides optimal performance for a particular computational task while simplifying the hardware implementation of a typical delay-feedback reservoir system.*

In order to determine the validity of this hypothesis, the main hypothesis is split into four sub-hypotheses:

Sub-Hypothesis 1: Computational tasks with particular characteristics perform best within distinct areas of the parameter space of the delay-feedback reservoir system model.

Sub-Hypothesis 2: Adjusting the timescale of the integrator within the non-linear node can lead to an increase in performance without a significant change in the hardware architecture.

Sub-Hypothesis 3: Changing the behaviour of the Mackey-Glass non-linear function can lead to an increase in performance within different computational tasks, with minimal changes to the hardware architecture.

Sub-Hypothesis 4: Replacing the non-linear function and integration stage of a delay-feedback reservoir with a higher order filter can lead to an increase in performance while simplifying the hardware architecture of a delay-feedback reservoir system model.

In order to investigate the plausibility of these hypotheses, several existing physical delay-feedback reservoir systems were examined to create a hardware accurate simulation model, within Simulink, which was the basis for the simulation models used throughout the experimental chapters; see chapter 4.

Chapter 5 verifies sub-hypothesis 2 and partially verifies sub-hypothesis 1 by investigating the effect that the timescale of a first-order filter has on the performance and system metrics of a delay-feedback reservoir. The results of these experiments

showed four key findings. First, that if a computational task has a high dependency on memory rather than non-linearity, then additional performance of a system can be gained by increasing the timescale of the system. Second, if a computational task exhibits a strong requirement for non-linearity but a low dependency on memory, than additional performance may be gained by increasing the number of virtual nodes within a system. Third, if a computational task has a high dependency on memory and low dependency of non-linearity, it is possible to emulate the same performance within a smaller-node system providing that the new system has a value of ρ equal to that of the previous system, and new system exhibits the minimum non-linearity and dimensionality required to run the computational task. Fourth, evidence for an "area of best performance", where computational tasks perform best within distinct areas of the parameter space.

Chapter 6 verifies sub-hypothesis 3 and sub-hypothesis 1 by investigating the effect that the Mackey-Glass non-linear function has on the performance and system metrics of a delay-feedback reservoir. The results of these experiments showed five key findings. First, if a computational task has a strong dependency on non-linearity, than additional performance of a system can be gained by reducing the value of the Mackey-Glass exponent. Second, the Mackey-Glass non-linear function only has three operating modes where it provides different non-linear behaviours; at one, odd, or even. Third, the Mackey-Glass non-linear function is limited in terms of non-linear dynamics, only exhibiting a region of linearity and an attractor as its edges. Fourth, the Mackey-Glass non-linear function has a predictable region of best performance for both the NARMA-10 and Santa Fe benchmark tasks, with NARMA-10 primarily utilising the linear region while Santa Fe primarily utilising the non-linear regions. Fifth, confirms the concept for an "area of best performance" for each computational benchmark.

Chapter 7 refutes sub-hypothesis 4 by using an evolutionary algorithm to determine if a high-order filter can replace a typical non-linear node within a delay-feedback reservoir system. The results of these experiments showed four key findings. First, a

high-order filter non-linear node significantly reduces the performance within a system by removing the state-space non-linearity. Second, both real-valued and complex-valued high-order filters provide the same amount of non-linearity to a system. Third, using a complex-order filter significantly improves the memory of a system. Fourth, a high-order filter can be used with a non-linear function, in a non-linear node, to supplement the dynamics of a non-linear function to improve system performance.

8.1.1 Concluding Remarks

The goal of this thesis was to better understand the building blocks of the non-linear node within a delay-feedback reservoir so that a system could be tuned to solve a particular computational task within a physical substrate. This thesis has extensively evaluated the structure, functionality, and parameters of an analogue electronic non-linear node, with the results showing how a non-linear node can be modified to increase the performance when solving a particular type of computational task. Given the findings and outcomes of this work, the main hypothesis of this thesis can be verified to be true.

8.2 Future Work

While the fundamental aspects of an analogue non-linear node within a delay-feedback reservoir has been evaluated, there are several remaining avenues to explore to build upon a design methodology. This section lists some of the future work related to specific parts of this thesis.

8.2.1 Alternative Non-Linear Function

Within chapter 6, the effect that the Mackey-Glass non-linear function had on the system performance and dynamics was investigated. Although the Mackey-Glass non-

linear function is one of the most popular non-linear functions, there are several other non-linear functions that have been used or could be explored within the literature.

By far the second most popular non-linear function is sin^2, which is primarily used within photonic systems; but despite its popularity, it has had limited use within analogue electronic systems. From the conclusions drawn within chapter 6, the sin^2 non-linear function should perform well due to its non-linear and attractor behaviour, making it an ideal non-linear function for further investigation.

Perhaps a more interesting, and hardware friendly, avenue to explore is to use transistors as a non-linear function. Depending on the configuration, many of the desired non-linear behaviours can be implemented relatively easy in hardware [135]. However this idea is not new, there are several analogue electronic delay-feedback reservoir systems that use a transistor as a non-linear stage, but these tend to be relatively simple transistor configurations, such as a BJT or complementary MOSFETs. It may be possible to create a much more dynamically rich non-linear node if more complex configurations were investigated. This could be achieved by using a field-programmable transistor array and evolution to investigate more complex transistor-based non-linear functions.

8.2.2 Input Masking Distributions

While this thesis has shown that the non-linear node is a fundamental part of a delay-feedback reservoir that can be optimised in a number of ways to increase performance and system dynamics, the input masking procedure is just as important. Within a delay-feedback reservoir, the masking procedure perturbs the non-linear node so it remains within its transient regime. This allows a topology of virtual nodes to be created, and facilitates the mixing of information within these virtual nodes so that computation can be performed.

Several different masking procedures have been investigated, for example the differences between binary and random masking by Gan, et al [95], but there is little in the literature about creating masks with the sole purpose of changing the dynamics of a system. It was observed within chapter 7 how more complex transient responses were able to change the dynamics of a system, such as topology and memory capacity, but what if this, and more, could be achieved by the input masking rather than the transient response of the non-linear node.

From the work within this thesis, three different input masking procedures could be explored. First, instead of generating the input weights uniformly, apply a particular distribution or function to the generation process. As the reservoir system will apply its own distribution transform to the input signals depending on the dynamics of the system, it may be possible to find a distribution or function that compliments the reservoirs own transform, with the hope of amplifying particularly useful dynamics. Second, a typical input mask is a series of weighted step responses with flat peaks. Instead of a flat peak within a weighted step response, overlay a function at the peak so that it can compliment the transient response of the non-linear node. One of the simplest examples would be to apply a slight ramp to the peak of the masking signal, which would change the time response of the non-linear node. Third, the period of the masking signal is typically constant and defined as θ. Instead of having a constant masking period, use a variable masking period which sums to τ, the total time delay. This would create some very interesting topologies, as information would be temporally mixed non-linearly between nodes.

8.2.3 Physical Hardware Implementation

One of the goals of this thesis was to develop enough understanding of the function of the non-linear node so that a formal design methodology can be created in the future. However before this can be done, a physical hardware implementation of a delay-feedback reservoir needs to be realised using the information to verify the

conclusions within this work. As all simulation models were performed within Simulink and were designed to be realised within hardware, a framework for the transition between simulation and analogue hardware has been put in place.

A promising substrate that this could be implemented on is the field-programmable analogue array (FPAA), which was discussed within chapter 2.1. Not only is the FPAA an analogue substrate, making it ideal to realise the findings within this work, it is reconfigurable in real-time. This means almost all the experiments covered within this thesis can be carried out with an FPAA and a microcontroller by simply reconfiguring the hardware. The FPAA also excels in realising many different types of filters, such as Butterworth, Chebyshev, and Bessel, which could enhance the experiments within chapter 7 by having a real-time analogue hardware evolution loop.

Closing the loop from simulation to hardware is an essential next step in creating a design methodology for delay-feedback reservoirs; bridging the gap between traditional analogue circuit design and delay-feedback reservoir design.

Appendix A

Evolutionary Algorithms

An evolutionary algorithm is a generic term to describe population-based stochastic optimisation algorithms that can provide optimal solutions to complex problems through evolutionary pressure, inspired from Darwin's theory of evolution [136]. This encompasses many different streams of stochastic optimisation algorithms based upon natural evolution [137, 138], such as evolution strategies, evolutionary programming, and genetic algorithms, however they operate under a similar principle; by subjecting a population to evolutionary pressure, such as selection and variation techniques, allows the population to evolve over several evolutionary cycles [139].

A.1 Structure of a Evolutionary Algorithm

At its core, the general structure of an evolutionary algorithm consists of six main operators [140]; representation, initialisation, evaluation, selection, variation, and termination.

First, an individual is encoded through a representation schema to represent a possible solution to the optimisation problem. A number of individuals are then initialised against a distribution to create a diverse set of individuals, called a population.

Once the initial population has been initialised, it is then subject to evolutionary pressure, typically using a fitness function which evaluates the performance of each individual within a given environment and task. Individuals are selected and then evolved through various variation operators, simulating a "survival of the fittest" environment through reproduction of individuals. This process is repeated iteratively, with each iteration known as a generation, until the optimisation criteria has been met; typically when a single or group of individuals reach a particular solution. A typical structure of an evolutionary algorithm is shown in figure A.1.

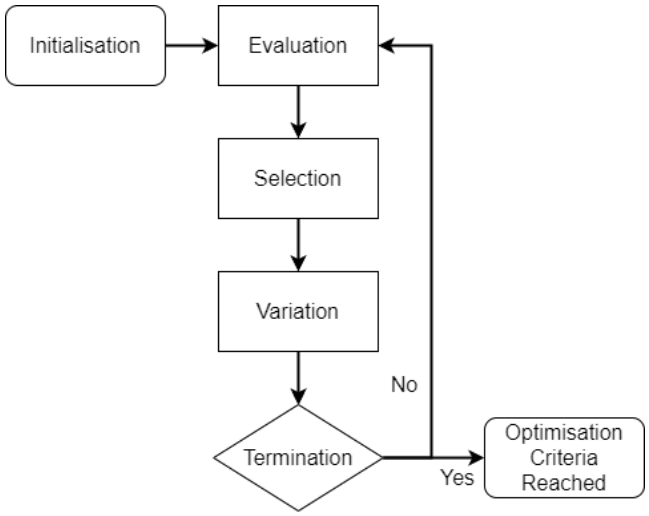

Figure A.1 A flow diagram showing the operations for a general evolutionary algorithm.

The operations for a general evolutionary algorithm are now explored in further detail.

Representation

Often one of the most challenging aspects when designing an evolutionary algorithm is how to represent a problem in a way that the evolutionary algorithm can provide optimisation. This is typically done by creating a data structure which holds the necessary parameters and variables required to represent an individual. The data structure that represents a individual is called a chromosome, while the parameters and variables within the chromosome are called genes; this is known as genetic encoding.

As evolutionary algorithms are usually employed to solve real-world applications, careful consideration must be given when choosing the allowed value ranges of genes or chromosomes, as real-world problems often have practical limitations that constrain solutions within a feasible range/region. As genetic encoding schemes are often tailored towards a particular problem, the representation schema is typically application specific, requiring a redesign for different problems.

Initialisation

With the generic encoding realised, a number of individuals are initialised to create the initial population pool; the number of individuals within the initialised population is called the population size. These individuals are generated either randomly, typically by a set distribution, or seeded, where the population is initialised with a specific configuration.

Evaluation

The evaluation operation aims to evaluate the performance of an individual by applying a fitness function to an individual, in order to determine how viable the suggested solution is. The fitness function chosen is dependant on the problem, but it usually generates a numerical value to represent the performance of an individual; this is often called a fitness score.

Selection

Once each individual within the population pool has been evaluated and assigned a fitness score, the selection operation determines which individuals should be passed on to the next generation. The aim of the selection operator is to determine the best performing individuals so they can be nurtured, while removing the poorly performing individuals. The selected individuals are then passed to the variation operator to allow for a search to take place and add genetic diversity to the next generation.

Variation

A genetic variation is an operation where variations are applied to the chromosome of an individual; this is also known as a genetic operator. The two most common genetic operators are crossover and mutation. A particular variant of crossover, called uniform crossover, is explored as it is better suited to the structure of a filter, as expressed in equation 7.1.

The uniform crossover genetic operator takes the chromosomes of the parent individuals and mixes them together to create a new combined chromosome, named the offspring individual. Typically one parent outperforms the other parents, where the best performing parent then passes its genes to the other parents to create new offspring in a stochastic process. The probability that the best performing parent passes on its genes is known as the crossover rate. An example of the uniform crossover genetic operator acting on a binary chromosome is shown in figure A.2, where parent 1 is the best performing individual.

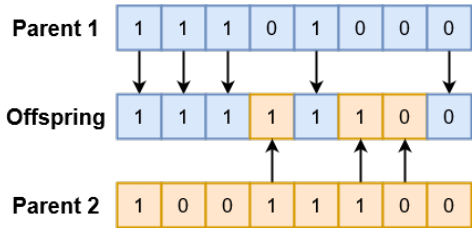

Figure A.2 An example of an uniform crossover genetic operator for a binary chromosome, where the probability that parent 1 passes a gene to the offspring is set by the crossover rate.

Another common genetic operator is mutation. The mutation operator typically occurs after crossover has taken place to add further genetic diversity to the chromosome of the offspring. The operator stochastically selects genes within a chromosome and modifies the gene based upon some mutation criteria; the probability that a gene will mutate is called the mutation rate. The mutation criteria is dependent on the problem to be optimised, but the gene modification is often restricted to keep the gene within

a feasible range. An example of the mutation genetic operator acting on a binary chromosome is shown in figure A.3.

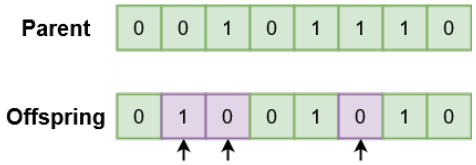

Figure A.3 An example of a mutation genetic operator for a binary chromosome, where the probability that a gene is flipped is set by the mutation rate.

A common issue with the Variation operator is that the genetic traits that make an individual perform well may be lost due to genetic variations [141]. To combat this issue, an elitism strategy can be used to ensure the best performing individuals are passed on to the next generation unmodified [142].

Termination

The termination operator allows the evolutionary algorithm to be stopped when the optimisation criteria has been reached. This optimisation criteria could be triggered when the evolutionary algorithm has reached a desired solution, reached the maximum number of generations, or become stagnant in the generation of solutions.

Appendix B

Results: Real Poles and Zeros

With a baseline established, the evolutionary algorithm is run to determine if a real-valued high-order filter can replace a typical non-linear node within a delay-feedback reservoir system. As the NARMA-10 and Santa Fe utilise different dynamics within a delay-feedback reservoir, separate experiments are performed using each benchmark and different virtual node sizes to evaluate the performance of each individual. This method will allow the evolutionary algorithm to favour different dynamics, such as memory or non-linearity, during the evolutionary process.

B.1 Evolutionary Algorithm Results

NARMA-10

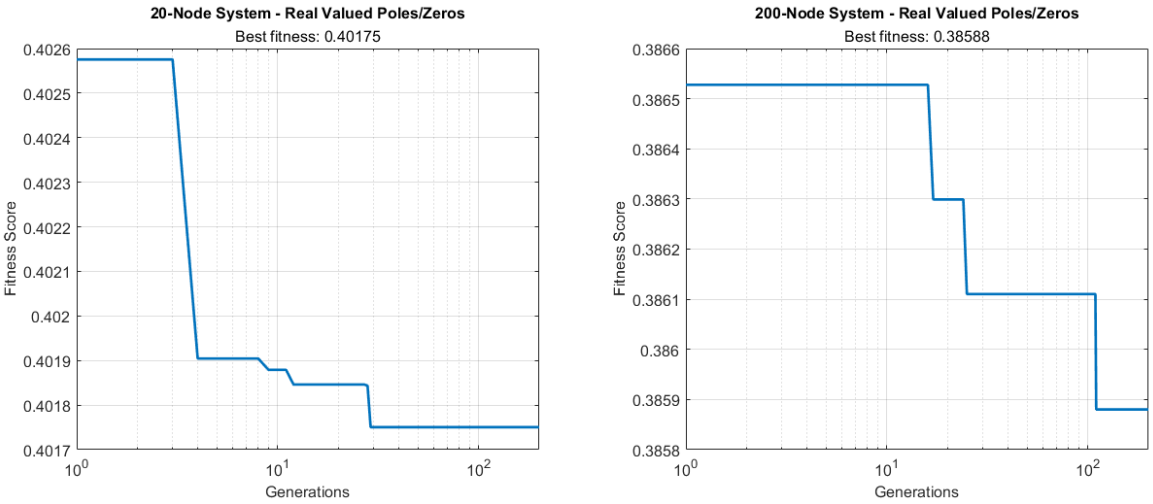

Figure B.1 Logx graphs showing the best performing fitness of a real-valued high-order filter acting as a non-linear node after each evolutionary generation for a 20- and 200-node delay-feedback reservoir system evaluating the NARMA-10 computational benchmark.

Figure B.1 shows the fitness of the best performing individual at each generation for a 20- and 200-node delay-feedback reservoir system, utilising a real-valued filter as the non-linear node, evaluating the NARMA-10 benchmark. The x-axis has had a log transform applied to the graph to better show the fitness transitions.

The main observations that can be made from the fitness graphs is that the initial fitness within both graphs is low for an initial solution, with relatively small changes being made throughout the evolutionary process. As the fitness score is the average of the testing and training NRMSE values, the initial fitness score indicates that the initial solution was fairly close to the optimised final. This implies that the initial population was sufficiently diverse in dynamics to find a near-solution relatively quickly. Only small changes were then made to the fitness score as evolution progresses, this strongly suggests that the evolutionary algorithm only had minimal dynamics to optimise as there is a limit to the dynamics that a real-valued filter can add. As the NARMA-10

benchmark was used to guide the fitness, it's very likely that there was insufficient non-linear dynamics within the system, so the evolutionary algorithm was optimising for memory capacity; this would explain the limit of the NRMSE been approximately 0.4.

Some insight can be gained into the behaviour of the evolutionary process by examining the top three unique best performing high-order filters within the population pool, this is shown in table B.1 for the 20-node system and table B.2 for the 200-node system.

Within the 20-node system, the top three filters have evolved to be fourth-order, with the top performing filter having a single zero. However as the single zero is almost a magnitude greater than the poles, it is likely to have minimal effect on the dynamics of the system. An interesting observation can be made within the timescale of each pole value, ρ value, and their distribution. Within the top two filters, there is a pattern of having a fast and slower timescales centred around two timescales which are close together. This essentially spreads out the transient response in time, as the sum of the exponential transient responses create a stronger overall transient that exists for longer; greatly increasing the system's ability to retain information. This behaviour can be observed within the 20-node system responses within figure B.2.

In contrast to the evolution within the 20-node system, the evolution within the 200-node system has favoured simpler filters; with the first and second best performing filters being second-order one zero, and the third being first order. The difference in order is likely due to the lack of non-linear dynamics of the system.

As the reservoir system has low non-linear dynamics due to the removal of the non-linear function, the theoretical minimum for the fitness score is similar between the 20- and 200-node systems as they share the same bottleneck. While both the 20- and 200-node systems share the same lack of non-linearity, the 200-node system is able to amplify the existing dynamics within the system due to the increased number of virtual nodes, allowing the 200-node system to achieve a greater memory capacity than

its 20-node counterpart. Therefore as the evolutionary algorithm attempts to optimise the individuals by increasing the memory capacity of the system, as it is unable to change the non-linearity of the system with real-valued poles and zeros. However as the bottleneck in achieving a higher fitness value is in non-linearity and not memory, the 200-node system is able to achieve the minimum amount of memory capacity required to compute the NARMA-10 benchmark with less complex high-order filters.

An interesting observation within the first two of the top performing 200-node filters is that they have two similar timescales as the 20-node system, a result that evolution has appeared to favour. The second-order filters also contain a single zero, which is close to the pole locations. The addition of the zero will increase the time-response of the system and will likely cause overshoot, which also increases the amount of information passed between virtual nodes; this can be observed within the 200-node system responses within figure B.2.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values
0.4046	0.399	10.2469	-1.5617 -1.1379 -1.0820 -0.7544	640ms 879ms 924ms 1326ms	0.16 0.22 0.23 0.33
0.4047	0.399	-	-1.5617 -1.1379 -1.1282 -0.7544	640ms 879ms 886ms 1326ms	0.16 0.22 0.22 0.33
0.4047	0.399	-	-1.1983 -1.1379 -1.1282 -0.9672	835ms 879ms 886ms 1034ms	0.21 0.22 0.22 0.26

Table B.1 A table showing the top three unique best performing real-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 20-node delay-feedback reservoir evaluating the NARMA-10 benchmark.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values
0.3997	0.3725	-0.7314	-1.5881 -1.4892	630ms 672ms	1.58 1.68
0.3989	0.3728	-0.7001	-1.5881 -1.4892	630ms 672ms	1.58 1.68
0.4053	0.3769	-	-1.7092	585ms	1.46

Table B.2 A table showing the top three unique best performing real-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 200-node delay-feedback reservoir evaluating the NARMA-10 benchmark.

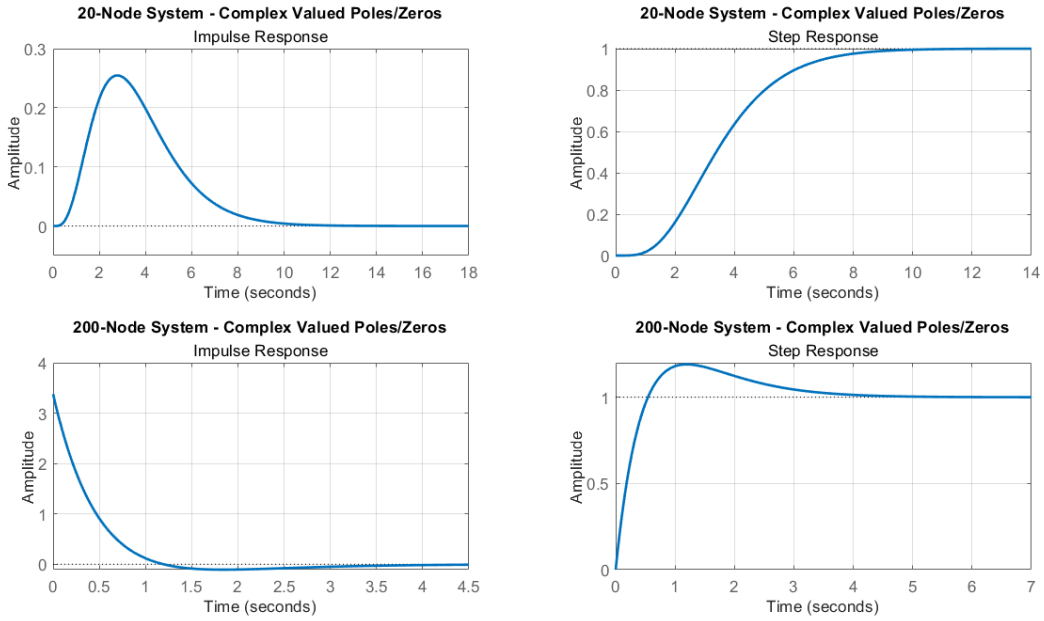

Figure B.2 Graphs showing the impulse and step response of the best performing real-valued high-order filters for a 20- and 200-node system generated within an evolutionary algorithm evaluated on the NARMA-10 benchmark.

Santa Fe

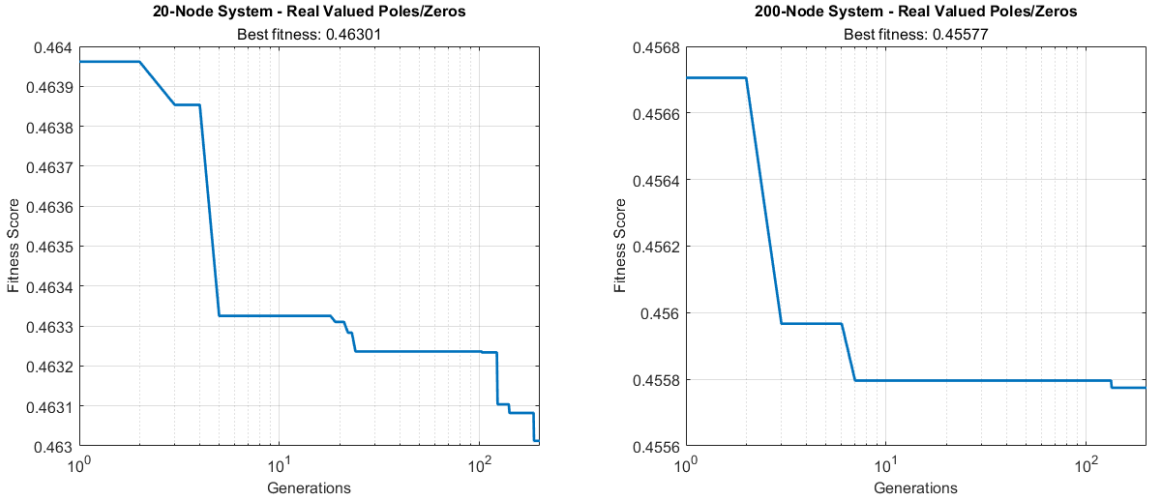

Figure B.3 Logx graphs showing the best performing fitness of a real-valued high-order filter acting as a non-linear node after each evolutionary generation for a 20- and 200-node delay-feedback reservoir system running the Santa Fe computational benchmark.

Figure B.3 shows the fitness of the best performing individual at each generation for a 20- and 200-node delay-feedback reservoir system evaluating the Santa Fe benchmark, utilising a real-valued filter as the non-linear node, on a x-axis log transformed graph to better show the fitness transitions.

Similar to the NARMA-10 trained filters, the initial fitness score is relatively low for an initial solution. This implies that the initial population pool was dynamically diverse, as there was enough diversity within the population pool to create a near optimum solution. Given that only small changes are made within the fitness score throughout the evolutionary process, this indicates that the evolutionary algorithm was unable to optimise the primary dynamics that the Santa Fe benchmark requires to reach peak performance. This can be observed within the fitness graphs, as the fitness score is almost the same for both the 20- and 200-node systems, suggesting that both systems exhibit the same bottleneck in terms of system dynamics; which is a lack of non-linearity given the Santa Fe benchmark requires a strong non-linearity for ideal performance.

Additional insight into how the behaviour of the evolutionary process can be gained by examining the structure of the top three unique best preforming high-order filters within the population pool; these filters are shown in table B.3 for the 20-node system and table B.4 for the 200-node system.

The top three filters within the 20-node system have evolved to be second-order, with a single zero. All three filters have very similar pole and zero values, suggesting that these pole and zero locations are at the optimal location to compute the Santa Fe benchmarks with the dynamics available to the system. The addition of the fast positive zero within the filters creates an initial negative undershoot to the step response within the time-domain of the filter as seen in the 20-node system responses within figure B.4. As this time-response occurs within θ (4 s), as seen by ρ being less than 1, this likely adds additional non-linearity to the system without greatly affecting the saturation or the connectivity between virtual nodes of the delay-feedback reservoir.

The evolutionary process within the 200-node system appears to have favoured larger order filters, with the top two performing filters being fourth-order with one zero and the third best performing filter being third-order. Given that the timescales within these filters are spread out, this indicates there is an evolutionary advantage to spreading the information out over a longer time period. This filter structure is similar to the NARMA-10 optimised 20-node filter, where it was found that having multiple timescales can increase the time-response of the filter, increasing the memory capacity of a system. The addition of a zero within the top two filters will increase the non-linear dynamics of the system, but will also add an overshoot to the step response as seen within the 200-node system responses shown in figure B.4. As the θ value of the 200-node system is 0.4 s, the transient response will last throughout 10 virtual nodes before the system reaches its settling point; therefore the overshoot will add additional gain to the system, causing the system to saturate early.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values
0.4682	0.4578	1.2303	-1.5866 -1.1526	630ms 868ms	0.16 0.22
0.4683	0.4578	1.2164	-1.5866 -1.1526	630ms 868ms	0.16 0.22
0.4684	0.4579	1.2164	-1.5314 -1.1683	653ms 856ms	0.16 0.21

Table B.3 A table showing the top three unique best performing real-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 20-node delay-feedback reservoir evaluating the Santa Fe benchmark.

Training NRMSE	Testing NRMSE	Zero Values	Pole Values	Filter Timescales	ρ Values
0.4601	0.4514	-0.5726	-7.8648 -2.1179 -1.5389 -1.3924	127ms 472ms 650ms 718ms	0.32 1.18 1.63 1.80
0.4601	0.4515	-0.5623	-7.5667 -2.1179 -1.5389 -1.3924	132ms 472ms 650ms 718ms	0.33 1.18 1.63 1.80
0.4646	0.4556	-	-8.4415 -2.1179 -1.5389	118ms 472ms 650ms	0.30 1.18 1.63

Table B.4 A table showing the top three unique best performing real-valued high-order filters, optimised through an evolutionary process, when used as a non-linear node within a 200-node delay-feedback reservoir evaluating the Santa Fe benchmark.

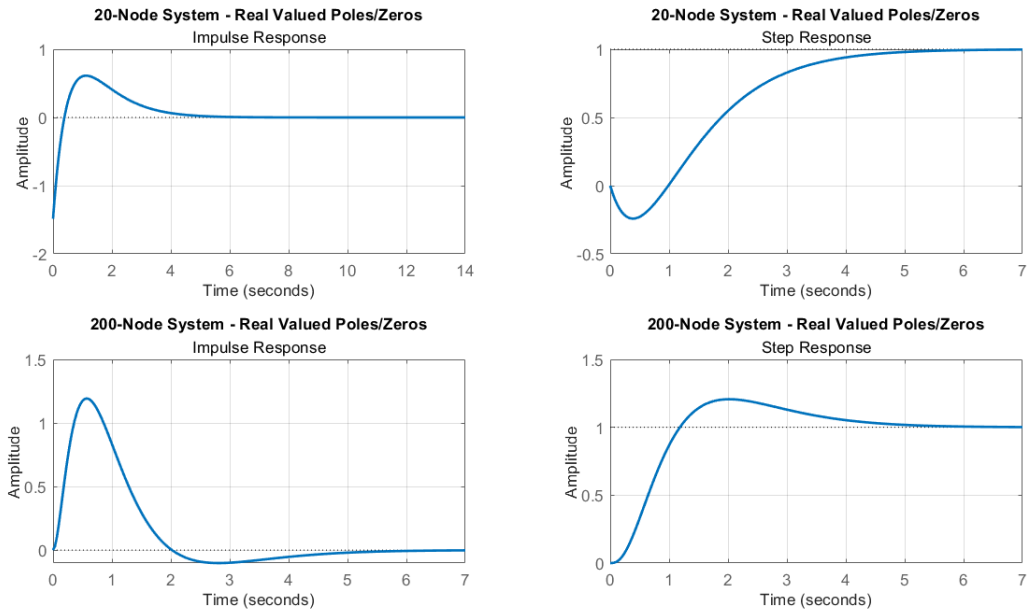

Figure B.4 Graphs showing the impulse and step response of the best performing real-valued high-order filters for a 20- and 200-node system generated within an evolutionary algorithm evaluated on the Santa Fe benchmark.

B.2 NARMA-10 Filter Analysis

Parameter Sweeps

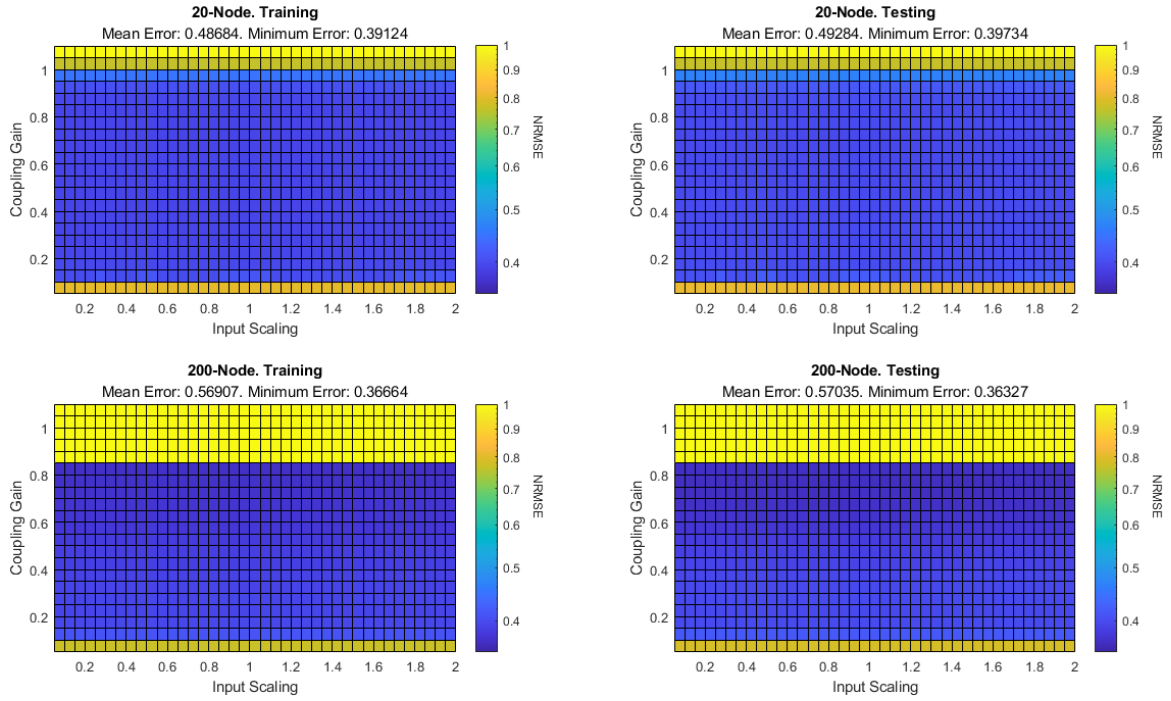

Figure B.5 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20- and 200-node system containing the best performing real-valued high-order filter, as shown in tables B.1 and B.2, used as a non-linear node within a delay-feedback reservoir evaluating the NARMA-10 benchmark; the input scaling range is between 0.05 and 2, and the coupling gain range is between 0.05 and 1.1.

Figure B.5 shows the effect that the input scaling and the coupling gain has on a 20- and 200-node delay-feedback reservoir system, running the NARMA-10 benchmark, with the best performing real-valued filter used as a non-linear node. As there are subtle changes within the NRMSE values within the parameter sweep, the colour map is set to logarithmic to better display the changes within error.

The parameter sweeps share many of the same features as the NARMA-10 baseline results shown in section 7.4.1; with an insensitivity to input scaling present, and regions of poor operation at the bottom and top of the parameter sweeps. The insensitivity to input scaling is likely to be present within all parameter sweeps without the non-linear

node, as the input/output relationship of the delay-feedback reservoir is determined by only the high-order filter. When the Mackey-Glass non-linear node was also included within the non-linear node, it was possible that the addition of the scaled input and output of the reservoir could cause the input to the non-linear node to be pushed outside the linear region, causing information to be lost. However as there is no attractor behaviour within the current system, information is always retained regardless of its magnitude, with information being purely transformed by the high-order filter.

While the parameter sweeps show the same regions of poor operation at the top and bottom of the parameter space as the baseline results, the 200-node system shows an increased sensitivity within the coupling gain, with the system becoming saturated at a coupling gain of approximately 0.825 rather than 1; which is most likely due to the zero within the high-order filter. As there is a zero present fairly close to the right-hand side of the pole positions, this zero will speed up the time-domain response and cause a slight overshoot to the step response of the filter. This overshoot causes the system to saturate earlier, as during the overshoot an additional gain to the system is applied, pushing it into positive feedback. The testing results within the 20-node system appear to be much more uniform within the 200-node system, which is the opposite that was observed within the baseline results; this is most likely due to the memory capacity of the system. The gradient of the decrease in NRMSE as the coupling gain increased implies that the memory capacity within the 200-node system rises much slower than in the 20-node system.

As previously discussed, the bottleneck when computing the NARMA-10 benchmark is the lack of non-linearity within the system, therefore the gradient within the NRMSE values implies that the 200-node system requires more coupling gain to reach the same memory capacity as the 20-node system. This is initially surprising given that a 200-node system can reach higher maximum system metrics, but when observing the timescales of the filters within tables B.1 and B.2, the best performing 20-node high-order filter appears to be much more optimised for memory given how it spreads the transient response.

The best performing NARMA-10 results from previous chapters and the results within this chapter, show two interesting observations; as shown in table B.5. First, despite the lack of a non-linear function, the 20-node system outperformed all previous experiments. Second, both the 20- and 200-node systems have almost the same testing and training error. This strongly suggests that the high-order filter is providing some non-linear dynamics, just not very strong dynamics.

	Best NRMSE from Chapter 5		Best NRMSE from Chapter 6		Best NRMSE from Baseline		Current System Best NRMSE	
Nodes	20	200	20	200	20	200	20	200
Training Error	0.491	0.306	0.484	0.271	0.514	0.367	0.391	0.397
Testing Error	0.494	0.308	0.483	0.280	0.522	0.362	0.367	0.362

Table B.5 A table showing the best performing NARMA-10 results from the previous chapters, the baseline results for a high-order non-linear node, and parameter sweeps performed in figure B.5.

System Metrics

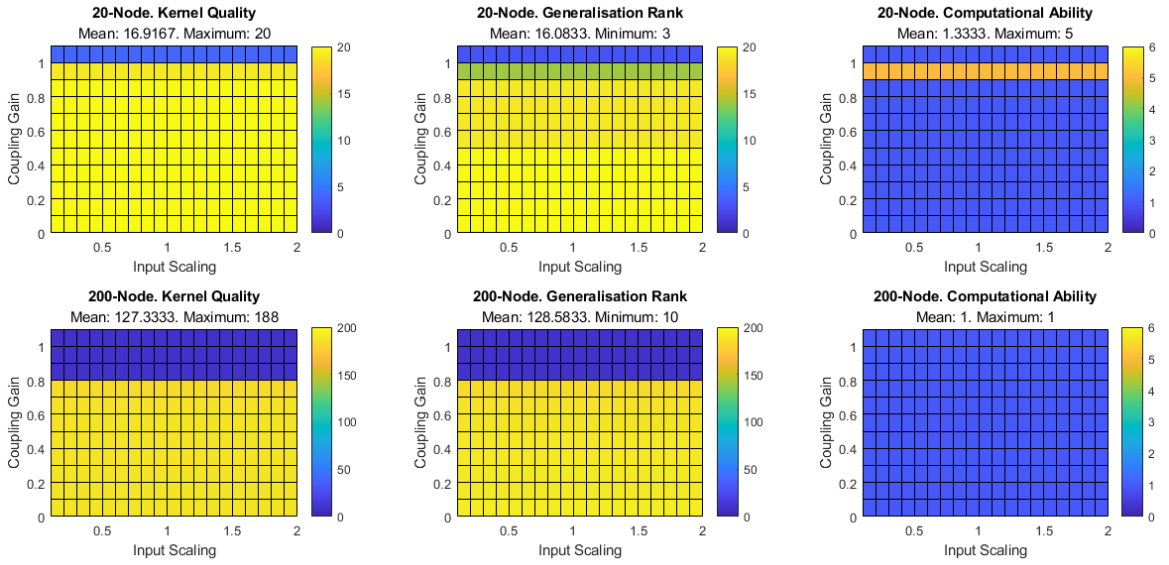

Figure B.6 Graphs showing the system metrics of a parameter sweep for a 20- and 200-node system, with different optimised real-valued high-order filters used as non-linear nodes within a delay-feedback reservoir evaluating the NARMA-10 benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1, for the following the system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

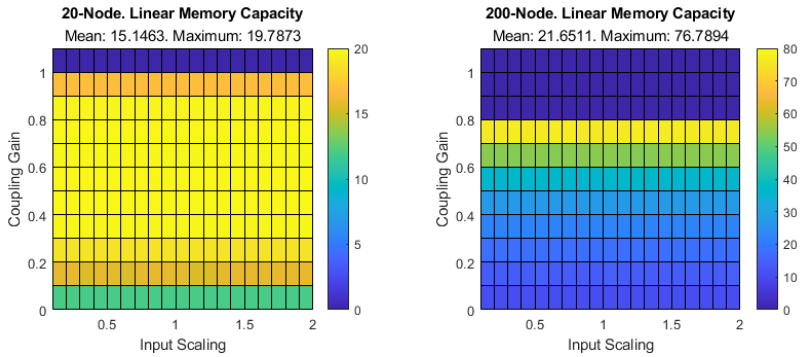

Figure B.7 Graphs showing the Linear Memory Capacity of a parameter sweep for a 20- and 200-node system, with different optimised real-valued high-order filters used as non-linear nodes within a delay-feedback reservoir evaluating the NARMA-10 benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1.

Figures B.6 and B.7 show the effect that the input scaling and the coupling gain has on the system metrics of a 20- and 200-node delay-feedback reservoir system, with a NARMA-10 optimised high-order filter as the non-linear node.

The system metrics confirm many of the observations made within the training and testing NRMSE parameter sweeps. With the exception of the saturation region at the top of the parameter space, the KQ and GR remain very high and constant throughout the entire parameter sweeps of both the 20- and 200-node systems, showing almost no sensitivity to both input scaling and coupling gain. The high values of GR indicate the system behaves poorly when generalising inputs, which is indicative of poor non-linearity within the system. There is a slight decrease in GR at the edge of stability, which is caused by the system saturating for some inputs, and thus appearing to generalise inputs. This gives the system a poor CA overall. The increased sensitivity in the coupling gain within the 200-node system, caused by the close zero within the filter, can be observed within the system metrics, causing the saturation region to occur at approximately 0.85 rather than 1.

As predicted, figure B.7 shows that the 20-node system is able to reach its maximum LMC with a much lower coupling gain when compared to the 200-node system, with the 20-node system reaching an LMC of approximately 20 when the coupling gain is in the region of 0.2. While the 200-node system also reaches an LMC of around 20 when the coupling gain is 0.2, the 200-node system is able to reach much higher LMC values as the coupling gain increases. This implies that the LMC of the system is restricted by the number of virtual nodes, rather than the filter design, as the filters used within the 20-node system reach their maximum value with minimal coupling gain.

Table B.6 lists the best performing system metrics from both the previous chapters along with this chapter, so a comparison between systems can be made. Comparing the system metrics from previous experiments show that the NARMA-10 optimised real-valued high-order filters exhibit very good KQ metrics but very poor GR and

CA, which is due to the weak non-linearity within the system, and exceptional LMC performance. The LMC is much higher than in previous experiments, which once again indicates that the evolutionary process has heavily optimised the filters in terms of memory, rather than non-linearity.

	Best Metrics from Chapter 5		Best Metrics from Chapter 6		Best Metrics from Baseline		Current System Best Metrics	
Nodes	20	200	20	200	20	200	20	200
Kernel Quality	20	190	17	80	17	130	20	188
Generalisation Rank	1	1	1	1	8	5	3	10
Computational Ability	19	166	16	119	4	51	5	1
Linear Memory Capacity	12.3	37.5	15.7	50.7	17.1	60.7	19.8	76.8

Table B.6 A table showing the best performing metrics from previous chapters, the baseline results for a high-order non-linear node, and parameter sweeps performed in figures B.6 and B.7.

B.3 Santa Fe Filter Analysis

Parameter Sweeps

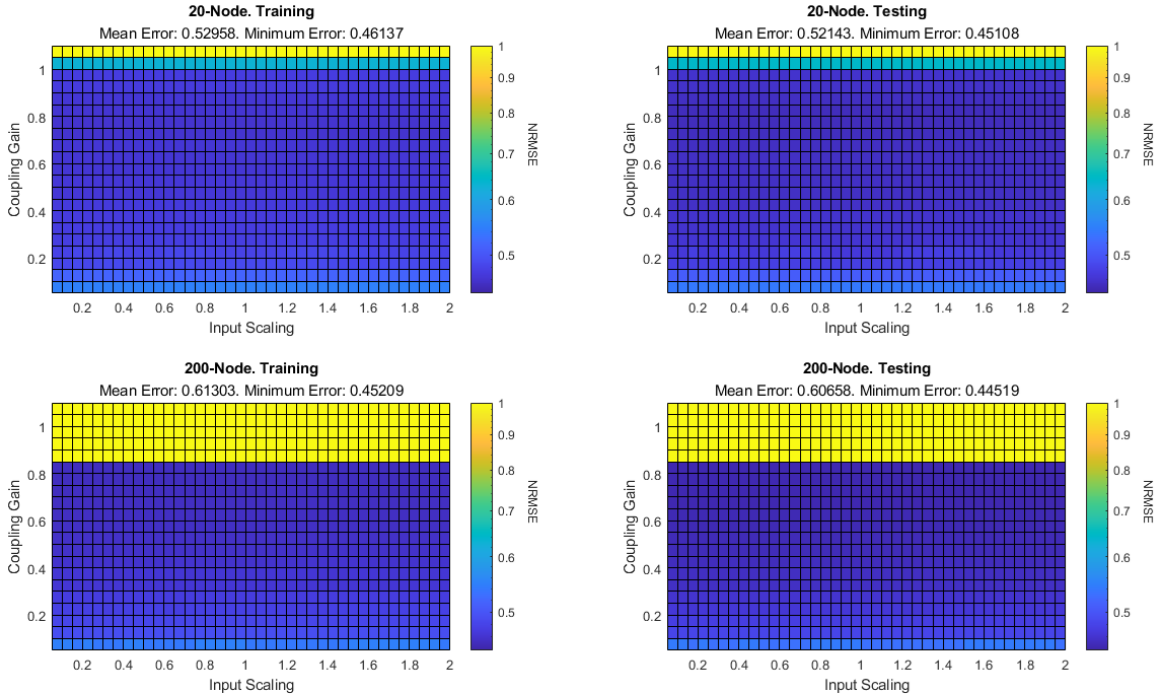

Figure B.8 Graphs showing the training and testing NRMSE values of a parameter sweep for a 20- and 200-node system containing the best performing real-valued high-order filter, as shown in tables B.3 and B.4, used as a non-linear node within a delay-feedback reservoir evaluating the Santa Fe benchmark; the input scaling range is between 0.05 and 2 and the coupling gain range is between 0.05 and 1.1.

Figure B.8 shows the effect the input scaling and the coupling gain has on a 20- and 200-node delay-feedback reservoir system, with the best performing real-valued filter used as a non-linear node, running the Santa Fe benchmark. Once again as there are subtle changes within the NRMSE values within the parameter sweep, the colour map is set to logarithmic to better display the changes within NRMSE.

The parameter sweeps show many of the same features as the baseline and NARMA-10 benchmark, which is not surprising given the insensitivity to the input scaling is a feature of the architecture of the non-linear node rather than the structure of the high-order filters. Both systems show a minimal sensitivity to the coupling

gain, with the exception of the region of saturation at the top of the parameter sweep and the region of information loss at the bottom of the parameter sweep. This is to be expected given the real-valued high-order filters providing a weak amount of non-linearity, where the Santa Fe benchmark requires strong non-linearity, which is set by the dynamics of the time-response, making it independent to the coupling gain.

A larger region of saturation can be observed within the 200-node system, which was predicted by the zero position within the filter. This suggests that adding a zero may increase the time-domain dynamics of the filter, but this creates an overshoot within the step response that makes it more sensitive to coupling gain. Given that the testing and training NRMSE values are almost uniform throughout the majority of the parameter sweep, the increased sensitivity to coupling gain is a good trade-off if it adds additional non-linear dynamics.

Within table B.7, the best performing Santa Fe results from previous chapters and this chapter are shown. The table shows that the 20-node system outperforms the baseline 20-node system and performs equally to its 200-node counterpart. This indicates that the evolutionary algorithm does work, as it is able to beat the performance of the 20-node system and match the 200-node system. However when comparing the current systems to the previous experiments, the performance is significantly worse. This result once again confirms that the real-valued poles and zeros only provide minimal non-linear dynamics to a system.

	Best NRMSE from Chapter 5		Best NRMSE from Chapter 6		Best NRMSE from Baseline		Current System Best NRMSE	
Nodes	20	200	20	200	20	200	20	200
Training Error	0.320	0.112	0.194	0.085	0.503	0.454	0.461	0.452
Testing Error	0.300	0.098	0.175	0.087	0.494	0.447	0.451	0.445

Table B.7 A table showing the best performing Santa Fe results from chapter 5, chapter 6, the baseline for a high-order non-linear node, and the parameter sweeps performed in figure B.8.

System Metrics

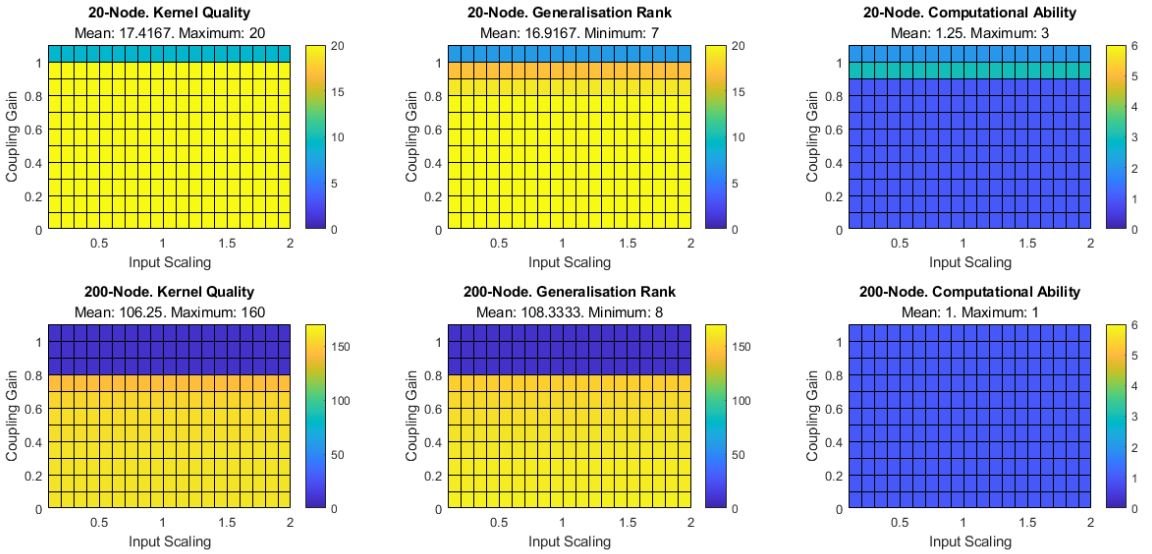

Figure B.9 Graphs showing the system metrics of a parameter sweep for a 20- and 200-node system, with different optimised real-valued high-order filters used as a non-linear node within a delay-feedback reservoir evaluating the Santa Fe benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1, for the following system metrics: Kernel Quality, Generalisation Rank, and Computational Ability.

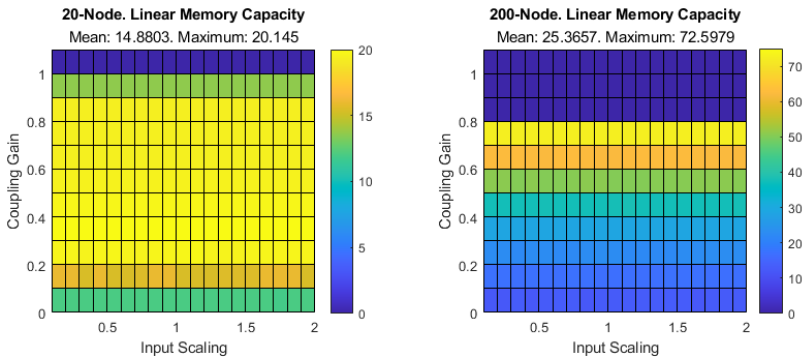

Figure B.10 Graphs showing the Linear Memory Capacity of a parameter sweep for a 20- and 200-node system, with different optimised real-valued high-order filters used as a non-linear node within a delay-feedback reservoir evaluating the Santa Fe benchmark, with the input scaling ranging between 0.05 and 2 and the coupling gain ranging between 0.05 and 1.1.

Figures B.9 and B.10 show the effect that the input scaling and the coupling gain has on the system metrics of a 20- and 200-node delay-feedback reservoir modified with a Santa Fe optimised high-order filter as the non-linear node.

The system metrics parameter sweeps are almost identical to the real-valued NARMA-10 filters, as observed within section B.2, with some minor differences. The KQ and GR within the 200-node system has a slight dependency on the coupling gain of the system, where both KQ and GR decrease as the coupling gain increases. Previously, it was thought that this was caused by the increase in the ratio between new inputs and the delayed output, however this does not explain why this occurs within some systems and not others. The reason for this is not obvious, but given the shape of the transient response and how the information is spread through approximately 10 virtual nodes, this suggests that there may no longer be a simple adjacency connectivity between virtual nodes. As the coupling gain increases, the connections between the virtual nodes become stronger, scaling linearly. However as the information is spread across virtual nodes non-linearly, the linear scaling factor may cause information to either be lost by having a stronger signal eclipse it, or be mixed with information in other virtual nodes.

An interesting observation when comparing the LMC of the 200-node Santa Fa optimised system to the LMC of the 200-node NARMA-10 optimised system is that the LMC within the current system increases faster, as the coupling gain increases, but the NARMA-10 optimised system reaches a higher maximum LMC value. This is likely due to the 200-node Santa Fe optimised system having a slower rise time than the 200-node NARMA-10 optimised system, as it allows information to be spread out over several virtual nodes.

Generally, the Santa Fe optimised system metrics are overall worse than those within the NARMA-10 optimised filters, but better than the baseline system metrics. This confirms that although different computational benchmarks were used to guide the fitness of the filters, due to the bottleneck in non-linear dynamics, a very similar

solution emerged due to the limited dynamics inherent within a real-valued filter non-linear node, but an improvement was still made over the baseline system.

Table B.8 shows the best performing system metrics from this chapter and previous chapters so a comparison can be made between systems. The table shows that the real-valued Santa Fe optimised filters exhibit a strong KQ and LMC but a very poor GR and CA, with the LMC outperforming previous experiments significantly. This once again shows that real-valued filters exhibit little tunable non-linear dynamics, but strong tunable memory dynamics.

	Best Metrics from Chapter 5		Best Metrics from Chapter 6		Best Metrics from Baseline		Current System Best Metrics	
Nodes	20	200	20	200	20	200	20	200
Kernel Quality	20	190	17	80	17	130	20	160
Generalisation Rank	1	1	1	1	8	5	7	8
Computational Ability	19	166	16	119	4	51	3	1
Linear Memory Capacity	12.3	37.5	15.7	50.7	17.1	60.7	20	72.6

Table B.8 A table showing the best performing system metrics from chapter 5, chapter 6, the baseline for a high-order non-linear node, and the parameter sweeps performed in figures B.9 and B.10.

B.4 Complex-Valued Genetic Representation

The evolutionary algorithm is now modified to handle complex poles and zeros in able to determine if additional performance can be gained by adding oscillatory dynamics to the high-order filters. The genetic encoding is modified by adding an additional row within the existing encoding to store the damping ratio, ζ, of a pole or zero, which is then used to compute the imaginary part of the complex number, ω_d, by the following equations:

$$\begin{aligned} \theta &= \arccos(\zeta) \\ \omega_d &= \alpha \tan(\theta) \end{aligned} \tag{B.1}$$

The damping ratio is chosen to generate a complex value as it has a physical meaning in terms of the oscillatory dynamics. When the damping ratio is 0, the pole or zero will only have an imaginary part and therefore exhibits undamped oscillatory behaviour, while a damping ratio of between 0 and 1 gives a complex pole or zero and a critically damped oscillatory behaviour. Finally, when the damping ratio is set to 1, the pole or zero will be purely real, exhibiting no oscillatory dynamics as used within section B. As a complex pole or zero exist as complex conjugates, a pole or zero which has a damping ratio is represented as two poles or zeros, doubling the maximum number of poles and zeros to 10 and 8 respectively.

The probability that a pole or zero will be initialised as complex is determined from a random uniform distribution, this is set to 0.2 and 0.1 by default. As the desired values for the damping ratio are between 0 and 1, another uniform distribution, set with a range between 0 and 1, is used to generate the initial value of the damping ratio. A uniform distribution is chosen over a Gaussian distribution as it is desirable in this case to have a wide range of damping ratios within individuals, which the evolutionary algorithm can then use to optimise. A table of the additional parameters are listed within table B.9.

The genetic operators used will also be applied to the damping ratio of an individual. Within uniform crossover, the damping ratio will also be inherited for a selected pole or zero, and within mutation, a selected pole or zero will also have its damping ratio mutated by a Gaussian distribution.

Figure B.11 shows an example of the new genetic encoding for a sixth-order three pole high-order filter, consisting of two complex poles and one complex zero.

	p0	p1	p2	p3	p4	p5
Pole Active	1	1	0	1	0	1
Pole Value	-4.7	-1.9	-0.5	-1.1	-7.1	-2.7
Damping Ratio	0.21	0	0	0.67	0	0

	z0	z1	z2	z3
Zero Active	0	1	1	0
Zero Value	-2.2	-7.8	-4.5	-3.4
Damping Ratio	0	0	0.23	0

Figure B.11 A new genetic encoding supporting complex values representing a sixth-order three pole high-order filter, consisting of two complex poles and one complex zero.

Name	Function	Value
Probability of a Complex Pole	Sets the probability of generating a complex pole during initialisation.	0.2
Probability of a Complex Zero	Sets the probability of generating a complex zero during initialisation.	0.1
σ of Damping Ratio Pole Distribution	The standard deviation of the pole damping ratio Gaussian distribution.	0.1
σ of Damping Ratio Zero Distribution	The standard deviation of the zero damping ratio Gaussian distribution.	0.1

Table B.9 An additional list of hyper-parameters used within the modified evolutionary algorithm to determine the optimum complex-valued high-order filter to act as a non-linear node within a delay-feedback reservoir, with their default value.

Appendix C

Pole and Zero Graphs

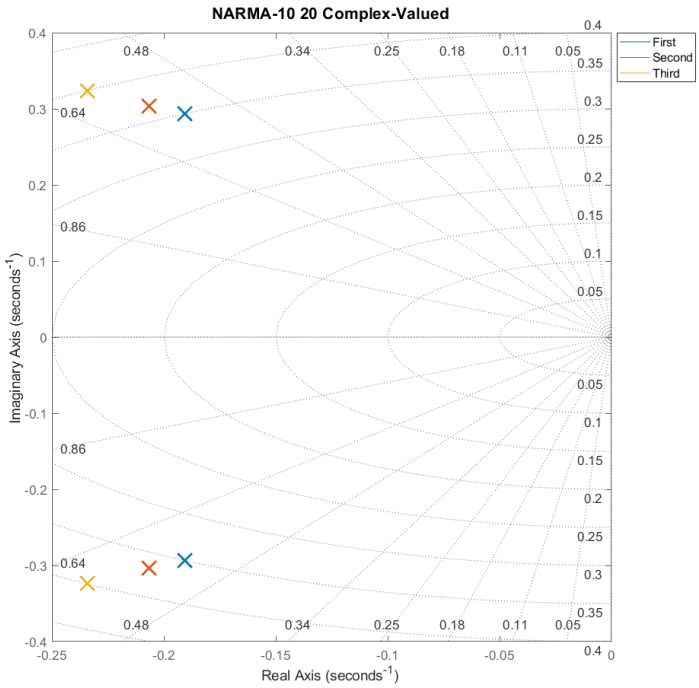

Figure C.1 Pole-Zero map of the top three performing complex-valued filters optimised by a evolutionary algorithm within a NARMA-10, 20-node system.

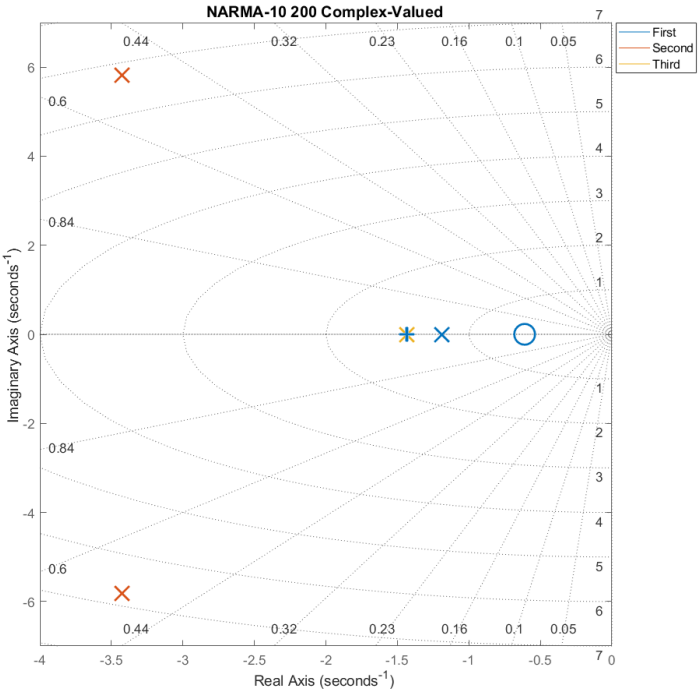

Figure C.2 Pole-Zero map of the top three performing complex-valued filters optimised by a evolutionary algorithm within a NARMA-10, 200-node system.

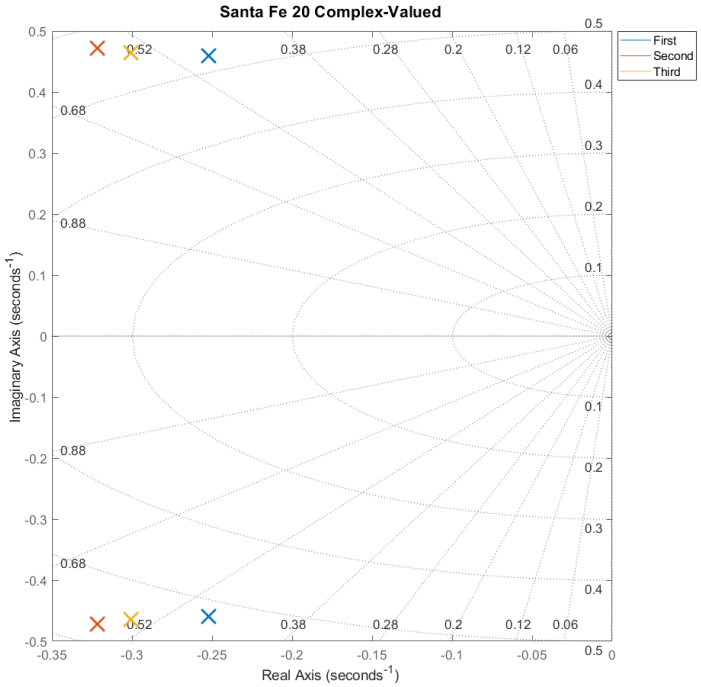

Figure C.3 Pole-Zero map of the top three performing complex-valued filters optimised by a evolutionary algorithm within a Santa Fe, 20-node system.

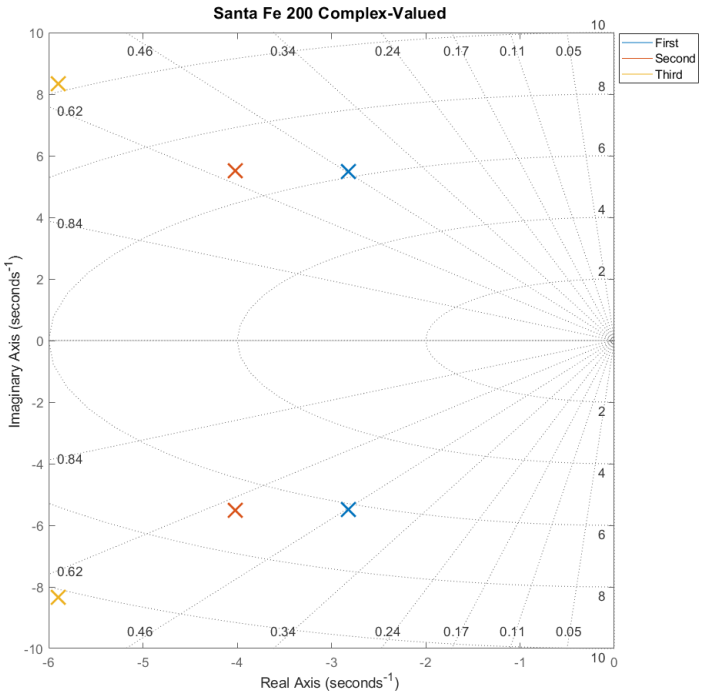

Figure C.4 Pole-Zero map of the top three performing complex-valued filters optimised by a evolutionary algorithm within a Santa Fe, 200-node system.

Appendix D

Thesis Summary

A summary of the experimental results and technical details for each experimental chapter is now listed. This aims to provide a reference to the outcomes and results within this thesis.

Within chapter 5, the effect of the system timescale on the performance and dynamics of a delay-feedback reservoir is investigated. A summary of the chapter is as follows:

- Concept of the ρ parameter introduced: A measure of how strong the connection is between virtual nodes.
- Simulation Environments: Simulink model of a delay-feedback reservoir with the Mackey-Glass non-linear function as the non-linear node and a first-order transfer function as the integration stage.
- Simulation Parameters: τ of 80 s, Mackey-Glass Exponent of 9.
- Four Integrator Timescales: 10ms, 100ms, 200ms, 400ms.
- Two Delay-Feedback Reservoir Systems: 20-node and 200-node.
- Two Benchmarks Tasks: NARMA-10 and Santa Fe.
- Experiment One: Performance Parameter Sweeps
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-2).
  - Performance evaluated by NRMSE.
  - Results: Timescale has effect on performance of NARMA-10.
  - Results: Timescale has little effect on performance of Santa Fe.
  - Results: Suggests idea of a "Region of Best Performance" to be true.
- Experiment Two: System Metrics Parameter Sweeps
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-2).
  - System Metrics: Kernel Quality, Generalisation Rank, Computational Ability, and Linear Memory Capacity.
  - Results: Timescale has no effect on KQ, GR, or CA.
  - Results: Timescale has large effect on LMC.
- Experiment Three: System Emulation
  - Randomly generated random-weighted mask for statistical simulation.

  - Benchmarks with large memory requirement can be emulated within smaller node systems, providing smaller node system exhibits the minimum non-linearity and dimensionality.

Within chapter 6, the effect of the Mackey-Glass non-linear function has on the performance and dynamics of a delay-feedback reservoir is investigated. A summary of the chapter is as follows:

- Simulation Parameters: τ of 80 s, Timescale of 400ms.
- Eight Mackey-Glass exponents: $1, 2, 5, 6, 9, 10, 13$, and 14.
- Two Delay-Feedback Reservoir Systems: 20-node and 200-node.
- Two Benchmarks Tasks: NARMA-10 and Santa Fe.
- Experiment One: Performance Parameter Sweeps
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-2).
  - Performance evaluated by NRMSE.
  - Results: Non-linearity can be increased by increasing the Mackey-Glass exponent value.
  - Results: Mackey-Glass non-linear function has three operating modes.
  - Results: Mackey-Glass non-linear function has limited dynamics.
  - Results: Confirms idea of a "Region of Best Performance" to be true.
- Experiment Two: System Metrics Parameter Sweeps
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-2).
  - System Metrics: Kernel Quality, Generalisation Rank, Computational Ability, and Linear Memory Capacity.
  - Results: Mackey-Glass non-linear exponent has no effect on KQ GR, and CA.
  - Results: Mackey-Glass non-linear exponent has moderate effect on LMC.
  - Results: Odd Mackey-Glass exponent values have better KQ, GR, and CA.
- Experiment Three: Utilisation
  - Simulation Environments: Modified existing Simulink model to record the input and output of the Mackey-Glass non-linear equation.

  - Heat-map to record the point density of input and output of the non-linear equation.
  - Results: 20-node system uses less of the function than 200-node system.
  - Results: NARMA-10 utilises linear region, whereas Santa Fe utilises non-linear region.

Within chapter 7, the effect of replacing the non-linear node with a high-order filter has on the performance and dynamics of a delay-feedback reservoir is investigated. A summary of the chapter is as follows:

- Created evolutionary algorithm to determine the optimal filter parameters.
- Represented a complex valued high-order filter as a 30-gene chromosome.
- Simulation Environments: Modified Simulink model to remove Mackey-Glass non-linear function.
- Simulation Parameters: τ of 80 s.
- Two Delay-Feedback Reservoir Systems: 20-node and 200-node.
- Two Benchmarks Tasks: NARMA-10 and Santa Fe.
- Experiment One: First-Order Filter Baseline
  - Experiment: Performance Parameter Sweeps (NRMSE)
  - Experiment: System Metrics Parameter Sweeps.
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-1.1).
  - Results: Insensitivity to input scaling.
  - Results: Very poor non-linear system dynamics.
- Experiment Two: Real Valued High-Order Filters
  - Fitness Graphs
  - Experiment: Performance Parameter Sweeps (NRMSE)
  - Experiment: System Metrics Parameter Sweeps.
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-1.1).
  - Results: Adds slightly more non-linearity when compared to baseline.
  - Results: KQ increases, CA decreases, and GR remains similar.
  - Results: Filters add time-domain non-linearity, not state-space non-linearity.

- Experiment Three: Complex Valued High-Order Filters
  - Fitness Graphs.
  - Experiment: Performance Parameter Sweeps (NRMSE)
  - Experiment: System Metrics Parameter Sweeps.
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-1.1).
  - Results: No additional non-linearity added to a system.
  - Results: LMC has increased significantly.
  - Results: Complex filters may be able to create complex virtual node topologies.
- Simulation Environments: Modified Simulink model to reintroduce Mackey-Glass non-linear function.
- Simulation Parameters: τ of 80 s, Mackey-Glass Exponent of 2.
- Experiment Four: Supplementary High-Order Filter
  - Experiment: Performance Parameter Sweeps (NRMSE).
  - Sweep Parameters: Input Scaling (0.05-2) and Coupling Gain (0.05-2).
  - Results: Complex filters can be used in conjunction with non-linear functions to enrich system dynamics and improve system performance.

References

[1] I. Tabian, H. Fu, and Z. Sharif Khodaei, "A convolutional neural network for impact detection and characterization of complex composite structures," *Sensors*, vol. 19, no. 22, p. 4933, 2019.

[2] N. Purkait, *Hands-On Neural Networks with Keras: Design and create neural networks using deep learning and artificial intelligence principles.* Packt Publishing Ltd, 2019.

[3] T. N. Impulse. (2016) Threshold potential.

[4] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural networks: Tricks of the trade.* Springer, 2012, pp. 659–686.

[5] Y. Liu, S. S. Yenamachintala, and P. Li, "Energy-efficient fgpa spiking neural accelerators with supervised and unsupervised spike-timing-dependent-plasticity," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 3, pp. 1–19, 2019.

[6] I. Vidamour, C. Swindells, G. Venkat, L. Manneschi, P. Fry, A. Welbourne, R. Rowan-Robinson, D. Backes, F. Maccherozzi, S. Dhesi *et al.*, "Reconfigurable reservoir computing in a magnetic metamaterial," *Communications Physics*, vol. 6, no. 1, p. 230, 2023.

[7] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Opt. Express*, vol. 20, no. 3, pp. 3241–3249, Jan 2012.

[8] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing

using a single dynamical node as complex system," *Nature communications*, vol. 2, no. 1, pp. 1–6, 2011.

[9] P. Kumar, M. Jin, T. Bu, S. Kumar, and Y.-P. Huang, "Efficient reservoir computing using field programmable gate array and electro-optic modulation," *OSA Continuum*, vol. 4, no. 3, pp. 1086–1098, 2021.

[10] M. C. Soriano, S. Ortín, L. Keuninckx, L. Appeltant, J. Danckaert, L. Pesquera, and G. Van der Sande, "Delay-based reservoir computing: noise effects in a combined analog and digital implementation," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 2, pp. 388–393, 2014.

[11] A. C. McDonnell and M. A. Trefzer, "The effect of system timescale on virtual node connectivity within delay-feedback reservoirs," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8.

[12] S. Stepney, "Guide to unconventional computing for music," in *Guide to Unconventional Computing for Music*, E. R. Miranda, Ed. Springer, 2017, pp. 1–21.

[13] M. Horowitz and E. Grumbling, *Quantum computing: progress and prospects*. National Academies Press, 2019.

[14] A. Adamatzky, "Slime mould processors, logic gates and sensors," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 373, no. 2046, p. 20140216, 2015.

[15] C. S. Calude, *Unconventional computing: A brief subjective history*. Springer, 2017.

[16] T. Freeth, Y. Bitsakis, X. Moussas, J. H. Seiradakis, A. Tselikas, H. Mangou, M. Zafeiropoulou, R. Hadland, D. Bate, A. Ramsey *et al.*, "Decoding the ancient greek astronomical calculator known as the antikythera mechanism," *Nature*, vol. 444, no. 7119, pp. 587–591, 2006.

[17] W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M.-F. Chang, H.-J. Yoo, H. Qian, and H. Wu, "Neuro-inspired computing chips," *Nature electronics*, vol. 3, no. 7, pp. 371–382, 2020.

[18] J.-Q. Yang, R. Wang, Y. Ren, J.-Y. Mao, Z.-P. Wang, Y. Zhou, and S.-T. Han, "Neuromorphic engineering: from biological to spike-based hardware nervous systems," *Advanced Materials*, vol. 32, no. 52, p. 2003610, 2020.

[19] J. Zou, Y. Han, and S.-S. So, "Overview of artificial neural networks," *Artificial neural networks: methods and applications*, pp. 14–22, 2009.

[20] M. Brin and G. Stuck, *Introduction to dynamical systems*. Cambridge university press, 2002.

[21] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[22] P. Antonik, A. Smerieri, F. Duport, M. Haelterman, and S. Massar, "Fpga implementation of reservoir computing with online learning," in *24th Belgian-Dutch Conference on Machine Learning*, 2015.

[23] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, "Reservoir computing using dynamic memristors for temporal information processing," *Nature communications*, vol. 8, no. 1, p. 2204, 2017.

[24] C. Fernando and S. Sojakka, "Pattern recognition in a bucket," in *European conference on artificial life*. Springer, 2003, pp. 588–597.

[25] F. M. Atay, *Complex time-delay systems: theory and applications*. Springer, 2010.

[26] M. L. Alomar, E. S. Skibinsky-Gitlin, C. F. Frasser, V. Canals, E. Isern, M. Roca, and J. L. Rossello, "Efficient parallel implementation of reservoir computing systems," *Neural Computing and Applications*, vol. 32, no. 7, pp. 2299–2313, 2020.

[27] D. Brunner, B. Penkovsky, B. A. Marquez, M. Jacquot, I. Fischer, and L. Larger, "Tutorial: Photonic neural networks in delay systems," *Journal of Applied Physics*, vol. 124, no. 15, p. 152004, 2018.

[28] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Scientific reports*, vol. 2, no. 1, pp. 1–6, 2012.

[29] R. R. Schaller, "Moore's law: past, present and future," *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[30] M. Miscuglio, Y. Gui, X. Ma, Z. Ma, S. Sun, T. El Ghazawi, T. Itoh, A. Alù, and V. J. Sorger, "Approximate analog computing with metatronic circuits," *Communications Physics*, vol. 4, no. 1, pp. 1–11, 2021.

[31] G. E. R. Cowan, R. C. Melville, and Y. P. Tsividis, "A vlsi analog computer/math co-processor for a digital computer," in *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.* IEEE, 2005, pp. 82–586.

[32] N. Guo, Y. Huang, T. Mai, S. Patil, C. Cao, M. Seok, S. Sethumadhavan, and Y. Tsividis, "Continuous-time hybrid computation with programmable nonlinearities," in *ESSCIRC Conference 2015-41st European Solid-State Circuits Conference (ESSCIRC).* IEEE, 2015, pp. 279–282.

[33] E. F. Camacho and C. B. Alba, *Model predictive control.* Springer science & business media, 2013.

[34] E. Lee and P. Gulak, "A cmos field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1860–1867, 1991.

[35] H. Kutuk and S.-M. Kang, "A field-programmable analog array (fpaa) using switched-capacitor techniques," in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, vol. 4. IEEE, 1996, pp. 41–44.

[36] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R. Wunderlich, S. Nease, and S. Ramakrishnan, "A programmable and configurable mixed-mode fpaa soc," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2253–2261, 2016.

[37] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[38] D. O. Hebb, *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

[39] B. Farley and W. Clark, "Simulation of self-organizing systems by digital computer," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 76–84, 1954.

[40] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[41] C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

[42] B. Widrow, "Adaptive adaline neuron using chemical memistors." Stanford University, Tech. Rep., 1960.

[43] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[45] H. J. Kelly, "Gradient theory of optimal flight paths," *ARS Journal*, vol. 30, no. 10, pp. 947–954, 1960.

[46] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[47] M. Roodschild, J. G. Sardiñas, and A. Will, "A new approach for the vanishing gradient problem on sigmoid activation," *Progress in Artificial Intelligence*, vol. 9, no. 4, pp. 351–360, 2020.

[48] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[49] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of chemical information and computer sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[50] M. V. Valueva, N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020.

[51] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*. Ieee, 2017, pp. 1–6.

[52] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of computer and system sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[53] M. Miljanovic, "Comparative analysis of recurrent and finite impulse response neural networks in time series prediction," *Indian Journal of Computer Science and Engineering*, vol. 3, no. 1, pp. 180–191, 2012.

[54] K.-i. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural networks*, vol. 6, no. 6, pp. 801–806, 1993.

[55] J. Kilian and H. T. Siegelmann, "The dynamic universality of sigmoidal neural networks," *Information and computation*, vol. 128, no. 1, pp. 48–56, 1996.

[56] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988.

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[59] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery that learns to tie knots using recurrent neural networks," *Advanced Robotics*, vol. 22, no. 13-14, pp. 1521–1537, 2008.

[60] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[61] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[62] K. Boahen, "Neuromorphic microchips," *Scientific American*, vol. 292, no. 5, pp. 56–63, 2005.

[63] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[64] G. H. Bishop, "Natural history of the nerve impulse," *Physiological reviews*, vol. 36, no. 3, pp. 376–399, 1956.

[65] M. Kiselev, "Rate coding vs. temporal coding-is optimum between?" in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 1355–1359.

[66] H. S. Seung, "Learning in spiking neural networks by reinforcement of stochastic synaptic transmission," *Neuron*, vol. 40, no. 6, pp. 1063–1073, 2003.

[67] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[68] U. D. Schiller and J. J. Steil, "Analyzing the weight dynamics of recurrent learning algorithms," *Neurocomputing*, vol. 63, pp. 5–23, 2005.

[69] M. Dale, S. Stepney, J. F. Miller, and M. Trefzer, "Reservoir computing in materio: A computational framework for in materio computing," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2178–2185.

[70] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, 2019.

[71] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik Bonn, 2002, vol. 5, no. 01.

[72] ——, "Adaptive nonlinear system identification with echo state networks," *Advances in neural information processing systems*, vol. 15, pp. 609–616, 2002.

[73] Q. Ma, L. Shen, W. Chen, J. Wang, J. Wei, and Z. Yu, "Functional echo state network for time series classification," *Information Sciences*, vol. 373, pp. 1–20, 2016.

[74] D. Li, M. Han, and J. Wang, "Chaotic time series prediction based on a novel robust echo state network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 5, pp. 787–799, 2012.

[75] G. K. Venayagamoorthy and B. Shishir, "Effects of spectral radius and settling time in the performance of echo state networks," *Neural Networks*, vol. 22, no. 7, pp. 861–863, 2009.

[76] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[77] M. Hu, Y. Chen, J. J. Yang, Y. Wang, and H. H. Li, "A compact memristor-based dynamic synapse for spiking neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 8, pp. 1353–1366, 2016.

[78] Z. Guo, J. Wang, and Z. Yan, "Global exponential synchronization of two memristor-based recurrent neural networks with time delays via static or dynamic coupling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, pp. 235–249, 2014.

[79] S. Wen, R. Hu, Y. Yang, T. Huang, Z. Zeng, and Y.-D. Song, "Memristor-based echo state network with online least mean square," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 9, pp. 1787–1796, 2018.

[80] W. Maass, "Liquid state machines: motivation, theory, and applications," *Computability in context: computation and logic in the real world*, pp. 275–296, 2011.

[81] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *Journal of computer and system sciences*, vol. 69, no. 4, pp. 593–616, 2004.

[82] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. Van Campenhout, "Compact hardware liquid state machines on fpga for real-time speech recognition," *Neural networks*, vol. 21, no. 2-3, pp. 511–523, 2008.

[83] Q. Wang, Y. Li, and P. Li, "Liquid state machine based pattern recognition on fpga with firing-activity dependent power gating and approximate computing," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 361–364.

[84] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer, "Reservoir computing in material substrates," *Reservoir Computing: Theory, Physical Implementations, and Applications*, pp. 141–166, 2021.

[85] K. Nakajima, "Physical reservoir computing—an introductory perspective," *Japanese Journal of Applied Physics*, vol. 59, no. 6, p. 060501, 2020.

[86] J. H. Jensen and G. Tufte, "Reservoir computing in artificial spin ice," in *Artificial Life Conference Proceedings 32*. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info, 2020, pp. 376–383.

[87] D. A. Allwood, M. O. A. Ellis, D. Griffin, T. J. Hayward, L. Manneschi, M. F. K. Musameh, S. O'Keefe, S. Stepney, C. Swindells, M. A. Trefzer, E. Vasilaki, G. Venkat, I. Vidamour, and C. Wringe, "A perspective on physical reservoir computing with nanomagnetic devices," *Applied Physics Letters*, vol. 122, no. 4, p. 040501, 01 2023.

[88] M. Dale, D. Griffin, R. F. L. Evans, S. Jenkins, S. O'Keefe, A. Sebald, S. Stepney, F. Torre, and M. Trefzer, "Reservoir computing with magnetic thin films," 2023.

[89] E. Y. Tsymbal and D. G. Pettifor, "Perspectives of giant magnetoresistance," in *Solid state physics*. Elsevier, 2001, vol. 56, pp. 113–237.

[90] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Optics express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[91] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature communications*, vol. 4, no. 1, p. 1364, 2013.

[92] M. C. Soriano, D. Brunner, M. Escalona-Morán, C. R. Mirasso, and I. Fischer, "Minimal approach to neuro-inspired information processing," *Frontiers in computational neuroscience*, vol. 9, p. 68, 2015.

[93] Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu, "Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing," *Nature communications*, vol. 12, no. 1, p. 408, 2021.

[94] T. Zheng, W. Yang, J. Sun, X. Xiong, Z. Wang, Z. Li, and X. Zou, "Enhancing performance of reservoir computing system based on coupled mems resonators," *Sensors*, vol. 21, no. 9, p. 2961, 2021.

[95] T. Gan, S. Stepney, and M. A. Trefzer, "Tradeoffs with physical delay feedback reservoir computing," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 1–8.

[96] F. Köster, D. Ehlert, and K. Lüdge, "Limitations of the recall capabilities in delay-based reservoir computing systems," *Cognitive Computation*, pp. 1–8, 2023.

[97] T. Gan, S. Stepney, and M. A. Trefzer, "Combining multiple inputs to a delay-line reservoir computer: Control of a forced van der pol oscillator system," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–7.

[98] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[99] S. Dasgupta, F. Wörgötter, and P. Manoonpong, "Information theoretic self-organised adaptation in reservoirs for temporal memory tasks," in *Engineering Applications of Neural Networks: 13th International Conference, EANN 2012, London, UK, September 20-23, 2012. Proceedings 13*. Springer, 2012, pp. 31–40.

[100] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2002.

[101] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[102] A. F. Atiya and A. G. Parlos, "New results on recurrent network training: unifying the algorithms and accelerating convergence," *IEEE transactions on neural networks*, vol. 11, no. 3, pp. 697–709, 2000.

[103] C. Wringe, M. Trefzer, and S. Stepney, "Reservoir computing benchmarks: a review, a taxonomy, some best practices," *arXiv preprint arXiv:2405.06561*, 2024.

[104] A. S. Weigend, *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

[105] H. Jaeger, "Short term memory in echo state networks. gmd-report 152," in *GMD-German National Research Institute for Computer Science (2002), http://www. faculty. jacobs-university. de/hjaeger/pubs/STMEchoStatesTechRep. pdf*. Citeseer, 2002.

[106] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," *Scientific reports*, vol. 2, no. 1, p. 514, 2012.

[107] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar, "All-optical reservoir computing," *Opt. Express*, vol. 20, no. 20, pp. 22 783–22 795, Sep 2012.

[108] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural networks*, vol. 20, no. 3, pp. 323–334, 2007.

[109] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, "Hands-on reservoir computing: a tutorial for practical implementation," *Neuromorphic Computing and Engineering*, vol. 2, no. 3, p. 032002, 2022.

[110] J. Li, C. Zhao, K. Hamedani, and Y. Yi, "Analog hardware implementation of spike-based delayed feedback reservoir computing system," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3439–3446.

[111] G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, no. 3, pp. 561–576, 2017.

[112] K. Bai and Y. Yi, "Dfr: An energy-efficient analog delay feedback reservoir computing system for brain-inspired computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 4, pp. 1–22, 2018.

[113] G. Donati, C. R. Mirasso, M. Mancinelli, L. Pavesi, and A. Argyris, "Microring resonators with external optical feedback for time delay reservoir computing," *Optics Express*, vol. 30, no. 1, pp. 522–537, 2021.

[114] K. Zetie, S. Adams, and R. Tocknell, "How does a mach-zehnder interferometer work?" *Physics Education*, vol. 35, no. 1, p. 46, 2000.

[115] S. Sano, A. Uchida, S. Yoshimori, and R. Roy, "Dual synchronization of chaos in mackey-glass electronic circuits with time-delayed feedback," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, vol. 75, no. 1, p. 016207, 2007.

[116] L. Appeltant *et al.*, "Reservoir computing based on delay-dynamical systems," *These de Doctorat, Vrije Universiteit Brussel/Universitat de les Illes Balears*, 2012.

[117] L. Berezansky and E. Braverman, "Mackey-glass equation with variable coefficients," *Computers & Mathematics with Applications*, vol. 51, no. 1, pp. 1–16, 2006.

[118] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer, "A substrate-independent framework to characterize reservoir computers," *Proceedings of the Royal Society A*, vol. 475, no. 2226, p. 20180723, 2019.

[119] L. Junges and J. A. Gallas, "Intricate routes to chaos in the mackey–glass delayed feedback system," *Physics letters A*, vol. 376, no. 30-31, pp. 2109–2116, 2012.

[120] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[121] MathWorks, "Simulation and model-based design," https://uk.mathworks.com/products/simulink.html, 2024.

[122] L. Jaurigue and K. Lüdge, "Reducing reservoir computer hyperparameter dependence by external timescale tailoring," *Neuromorphic Computing and Engineering*, vol. 4, no. 1, p. 014001, 2024.

[123] M. Dale, S. O'Keefe, A. Sebald, S. Stepney, and M. A. Trefzer, "Computing with magnetic thin films: Using film geometry to improve dynamics," in *International Conference on Unconventional Computation and Natural Computation*. Springer, 2021, pp. 19–34.

[124] L. Appeltant, G. Van der Sande, J. Danckaert, and I. Fischer, "Constructing optimized binary masks for reservoir computing with delay systems," *Scientific reports*, vol. 4, no. 1, pp. 1–5, 2014.

[125] Y.-c. Wu and J.-w. Feng, "Development and application of artificial neural network," *Wireless Personal Communications*, vol. 102, pp. 1645–1656, 2018.

[126] K. Vandoorne, J. Dambre, D. Verstraeten, B. Schrauwen, and P. Bienstman, "Parallel reservoir computing using optical amplifiers," *IEEE transactions on neural networks*, vol. 22, no. 9, pp. 1469–1481, 2011.

[127] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima *et al.*, "Neuromorphic computing with nanoscale spintronic oscillators," *Nature*, vol. 547, no. 7664, pp. 428–431, 2017.

[128] A. Namajūnas, K. Pyragas, and A. Tamaševičius, "An electronic analog of the mackey-glass system," *Physics Letters A*, vol. 201, no. 1, pp. 42–46, 1995.

[129] P. Amil, C. Cabeza, and A. C. Martí, "Electronic implementation of the mackey-glass delayed model," *arXiv preprint arXiv:1408.5083*, 2014.

[130] S. E. Lyshevski, *Control systems theory with engineering applications*. Springer Science & Business Media, 2012.

[131] I. Harvey, "The microbial genetic algorithm," in *Advances in Artificial Life. Darwin Meets von Neumann: 10th European Conference, ECAL 2009, Budapest, Hungary, September 13-16, 2009, Revised Selected Papers, Part II 10*. Springer, 2011, pp. 126–133.

[132] G. Syswerda, "A study of reproduction in generational and steady-state genetic algorithms," in *Foundations of genetic algorithms*. Elsevier, 1991, vol. 1, pp. 94–101.

[133] A. F. Krause, V. Dürr, B. Bläsing, and T. Schack, "Multiobjective optimization of echo state networks for multiple motor pattern learning," in *18th IEEE Workshop on Nonlinear Dynamics of Electronic Systems: NDES 2010. Dresden University of Technology, Dresden, Germany, 26-28 May 2010; proceedings*, 2010.

[134] C. Yang and Z. Wu, "Multi-objective sparse echo state network," *Neural Computing and Applications*, vol. 35, no. 3, pp. 2867–2882, 2023.

[135] A. Buscarino, L. Fortuna, M. Frasca, G. Sciuto *et al.*, *A concise guide to chaotic electronic circuits*. Springer, 2014.

[136] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, "Evolutionary algorithms," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 178–195, 2014.

[137] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[138] D. Simon, *Evolutionary optimization algorithms*. John Wiley Sons, 2013.

[139] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)*. IEEE, 2016, pp. 261–265.

[140] X. Yu and M. Gen, *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[141] M. A. Trefzer and A. M. Tyrrell, *Evolvable hardware*. Springer, 2015.

[142] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2015.