The Effect of System Timescale on Virtual Node Connectivity within Delay-Feedback Reservoirs

Alexander C. McDonnell
School of Physics, Engineering and Technology
University of York
York, United Kingdom
alexander.mcdonnell@york.ac.uk

Martin A. Trefzer
School of Physics, Engineering and Technology
University of York
York, United Kingdom
martin.trefzer@york.ac.uk

***Abstract*—The delay-feedback reservoir is a branch of reservoir computing that has allowed for a more hardware friendly implementation by reducing the typically large number of inputs and outputs within neural networks to a single physical input and output. This is achieved by using a single non-linear node and time-multiplexing the input signal with a masking signal, to create many "virtual neurons" within a reservoir; emulating a larger spatial neural network. While the relationship of the number of virtual nodes, masking frequency, and overall delay length is well known, the effect that the timescale has on computational performance is not widely investigated or understood; usually because working with specific substrates implies the reservoir is confined to its inherent timescales. Hence, there is currently no general methodology for tuning delay-feedback reservoir systems to specific applications. Here, we create a parameterisable hardware-realistic computational model in order to simulate a delay-feedback reservoir operating at different timescales, and explore the effect this has on the connectivity between the virtual nodes and the computational performance of the system. Our results show that the timescale has indeed a significant effect on computational tasks with a long-term dependency on previous input stimuli. We then show that it is possible to emulate larger virtual node networks within smaller ones with little loss in performance and an increase in efficiency. This is an essential step towards understanding the function of the timescale within a delay-feedback reservoir, and a methodology to enable systematic tailoring of delay-feedback reservoirs to specific computational tasks across different timescales.**

***Index Terms*—Unconventional Computing, Reservoir Computing, Delay-Feedback Reservoir Computing, System Modelling, System Timescales.**

I. Introduction

In the search to meet the everlasting computational demand of modern systems, a biological approach has been applied to traditional computing paradigms. A particularly powerful brain-inspired computing paradigm, named Reservoir Computing, exploits the rich dynamical behaviours within a substrate to perform computation.

The Reservoir Computing framework is a machine learning paradigm inspired from the world of neurobiology. The foundation of the reservoir computing framework comes from complex neural networks, specifically Recurrent Neural Networks, that exhibit interesting dynamical behaviours [1], [2]. 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 FPGA and analogue circuitry [3], [4], to more unconventional systems, such as memristors and even a bucket of water [5], [6].

Many types of reservoir computing require a large amount of randomly connected recurrent neurons, which often requires significant space within hardware due to interconnect and IO requirements [7]. As an alternative, Appeltant et al. introduced a 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" to effectively emulate a much larger spatial recurrent neural network [9], allowing for a more compact and efficient use of hardware resources. Several interesting physical implementations of a delay-feedback reservoir have been realised with both commercial components, such as FPGAs and op-amps [4], [10], and more bespoke designs using photonic and optoelectronic techniques [11], [12]; both types of designs show promising results in performing time series prediction and other temporal tasks.

One of the most powerful features of using a delay-feedback reservoir is that they can be highly configurable. There are many parameters within the system that can be changed and will affect its computational ability, allowing for a robust solution to solving a variety of computational problems. However, its high configurability also means that designing a delay-feedback reservoir is often complex, as each parameter must be chosen carefully to maximise the potential computational performance of the system. Although previous research has investigated the effect that the node size [8], delay length [13], and masking procedure [14] have on a delay-feedback reservoir; the effects of the additional parameters are not well known. To better understand the function and configurability of a delay-feedback reservoir, we create a parameterisable hardware-realistic computational model that can be configured to operate at different timescales and configurations. We then determine how the connectivity between virtual nodes changes by altering the timescale of the system, this relationship is later defined ρ. The computational performance is then evaluated on two computational benchmark tasks as the connectivity between virtual nodes is changed. Here we chose two temporal

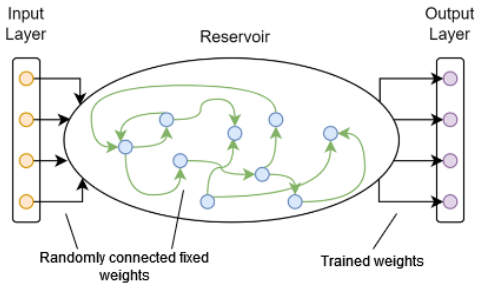

Fig. 1. The typical structure of a reservoir computer. It consists of three layers: an input layer which is randomly connected to the reservoir layer with random fixed connections; a reservoir layer which contains randomly connected recurrent neurons with randomly generated weighted connections; an output layer which is randomly connected to the reservoir layer but has weights which are trainable.

computational benchmarks, NARMA-10 and Santa Fe, as they require vastly different and opposite reservoir characteristics for optimal computational performance [15], [16]. This allows us to gain insight into which reservoir characteristics change as the timescale is altered within the reservoir system.

II. Reservoir Computing

A. The Reservoir Computing Framework

The reservoir computing framework has become a popular method of training input-driven dynamical systems, such as those derived from recurrent neural networks. The term reservoir comes from a randomly connected set of recurrent neurons, which are connected by fixed random weights, and observed as a dynamical black box; this is known as the reservoir layer. Inputs are then fed into the reservoir layer using an input layer, which is randomly connected to the reservoir layer with fixed random weighted connections. Outputs are observed from an output layer, which are connected randomly to the reservoir, but connections have trainable weights that can be calculated using training algorithms based on linear regression, e.g., Ridge regression or Moore-Penrose pseudo inverse, making the readout training simple; figure 1 depicts the typical structure of a reservoir computing system.

B. Delay-Feedback Reservoir Computing

The delay-feedback reservoir is fundamentally different to other types of reservoir computing. While other types of reservoir computing rely on a reservoir layer that consists of randomly connected recurrent neurons, the reservoir layer within the delay-feedback reservoir only consists of a single non-linear neuron and a time delay to emulate a much larger recurrent neural network [8]. Originally demonstrated by Appeltant in 2011, delay-feedback reservoirs have become a popular method of solving temporal computational tasks due to having a more hardware-friendly design, and being more compact than other traditional reservoir computing systems.

Much like other traditional reservoir computing systems, the delay-feedback reservoir contains three layers. The input layer, where the input signal is time-multiplexed with a much

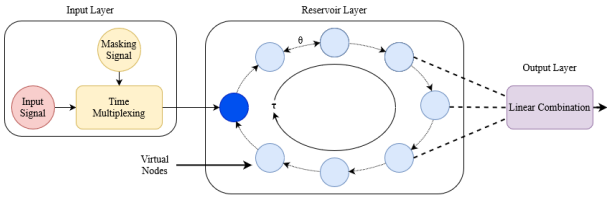

Fig. 2. The typical structure of a delay-feedback reservoir. A single non-linear neuron (dark blue) is used with a time delay to create a network of connected virtual nodes (light blue). The total time delay is notated as τ, which is often the same as the time period of the input signal. The time period of the masking signal, notated as θ, defines the spacing between the virtual nodes. The number of virtual nodes, N, can be calculated by $\frac{\tau}{\theta}$.

faster masking signal (often notated to have a time period of θ); this is known as the masking process. The reservoir layer, which contains a single non-linear neuron and a time delay (notated as τ). Finally, the output layer collates the states of the reservoir through linear combination, typically every θ, which is then used to train the network using weighted connections [8]. The number of virtual nodes within the system, N, is calculated by:

$$N = \frac{\tau}{\theta} \tag{1}$$

The greater the number of nodes within a reservoir, the greater its potential memory capacity will be. Figure 2 shows the typical structure of a delay-feedback reservoir network.

C. Virtual Node Connectivity

As with any neural computing paradigm, the connectivity between neurons can greatly affect the computational performance of the system. As a form of neural system, information within a reservoir computing network is processed by applying a stimulus to its input, thus creating transient activity patterns within the internal neurons [17]. The time in which these transients exist can be called the timescale of the system; where a system with a large timescale generates transients that are longer in time than a system with a small timescale. As previously discussed, the delay-feedback reservoir creates virtual neurons from a single non-linear neuron via the masking procedure. The topology of these virtual nodes is dependent upon the period of the masking signal and the timescale of the non-linear node, often denoted as T.

The purpose of the masking signal is to perturb the non-linear neuron in a way that it remains within its transient stage [18]. If the timescale of the non-linear neuron (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 its neighbouring virtual nodes; this creates a network topology where no virtual nodes are 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 depends on the response of its previous neighbours. In a

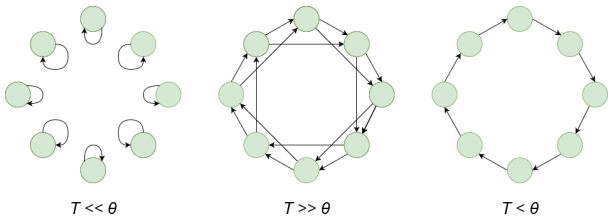

Fig. 3. The effect of the topology of virtual nodes within a delay-feedback reservoir when the timescale of the non-linear neuron (T) and the period of the masking signal (θ) changes.

typical application, the virtual nodes within a delay-feedback reservoir are in a ring topology, where information only passes between adjacent virtual nodes; in this case the timescale would be less than the period of the masking signal. This effect is shown in figure 3.

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. This effect is so prominent, we introduce a new parameter, ρ, which allows for a quick indication of how strong the connection is between each virtual node. It is defined by how long the transient response of the non-linear node exists within a single masking signal period. It is calculated by:

$$\rho = \frac{T}{\theta} \tag{2}$$

The effect of the timescale and masking period of a first-order non-linear node, with its corresponding ρ value, can be seen within figure 4.

III. System Evaluation

A. 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. There are many types of computational benchmarks, and here we select two temporal benchmarks: the NARMA-10 benchmark, which requires a large memory capacity and little non-linearity and dimensionality; and the Santa Fe Laser dataset which is the opposite, requiring a small memory capacity and a large non-linearity and dimensionality. Choosing two benchmarks that require opposite reservoir characteristics allows us to gain insight into how changing the timescale effects the reservoirs characteristics in the extreme cases, without having to perform extensive system metric sweeps.

NARMA-10. The non-linear autoregressive moving average task, or NARMA, is an imitation computational benchmark that utilises a weak non-linearity and a long-term dependency on previous input stimuli that can be used to evaluate a reservoirs dynamical capability. NARMA-10 in particular is of the 10th order, meaning it relies on inputs from a lag of 10 time steps. The equation for the discrete NARMA-10 sequence is:

$$y_{n+1} = 0.3y_n + 0.05y_n\left(\sum_{i=0}^{9} y_{n-i}\right) + 1.5u_n u_{n-9} + 0.1 \tag{3}$$

When applying the NARMA-10 benchmark to a reservoir computing system, the goal is to attempt to train the reservoir system to recreate the dynamics of the NARMA-10 signal when supplied with the same input stimuli; the trained output sequence and NARMA-10 equation output are then both compared so that the system performance can be evaluated.

Santa Fe Laser Dataset. The Santa Fe Laser dataset, also known as just Santa Fe or Laser, is a predictive highly non-linear computational benchmark derived from observing a far-infrared laser in a chaotic state [15]. The aim of the Santa Fe benchmark is to attempt to train the reservoir system 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 of previous input stimuli. This results in the Santa Fe benchmark requiring a reservoir system to have a stronger non-linear high-dimensionality rather than fading memory.

B. Training a Delay-Feedback Reservoir System

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. Here we calculate the error between the trained reservoir output and its target, then normalise it 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:

$$NRMSE = \sqrt{\frac{1}{m}\frac{\sum_{k=1}^{m}(\hat{y}_k - y_k)^2}{\sigma^2(y_k)}} \tag{4}$$

Where, m is the number of data samples within the experiment, $\hat{y}_k$ is the trained reservoir output, y_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.

IV. Experimental Methodology

A. Delay-Feedback Reservoir Evaluation

In order to create a hardware-realistic model, several existing implementations of delay-feedback reservoir systems were evaluated so that the fundamental building blocks could be established [4], [11], [19]. We found that there are two key components that a typical non-linear neuron has: a chaotic attractor, and an integrator.

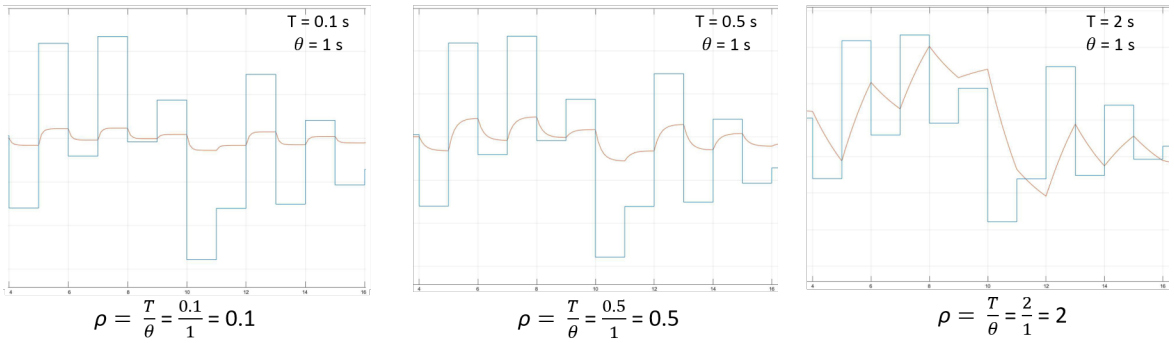

Fig. 4. The effect of the timescale and masking period of a first-order non-linear node within a delay-feedback reservoir for three timescales, 0.1 s, 0.5 s, and 2 s, with their calculated ρ values. The blue line represents the masked input signal, while the orange line represents the response of a first-order non-linear node.

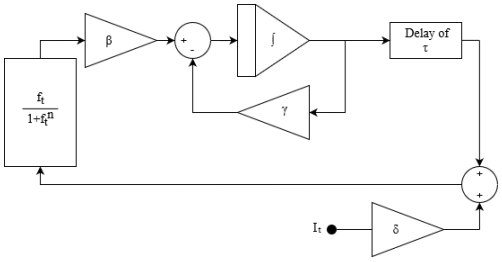

Fig. 5. The block diagram of the Mackey-Glass Non-Linear Time Delay Differential Equation.

The chaotic attractor serves two purposes within a delay-feedback reservoir system. First, it constrains values within the feedback loop within a specific range, and second, it adds a non-linear transform to the system. In theory, any type of chaotic attractor will work, but as we are focusing on a hardware-realistic model, we consider a chaotic attractor that has been shown to have a realistic hardware implementation [4].

Here we use the Mackey-Glass non-linear time delay differential equation, initially used to model the variation of mature blood cells [20], as it is a common chaotic attractor within the literature and is used in Appeltant's delay-feedback reservoir [8], [14]. The Mackey-Glass equation is expressed as:

$$\dot{x} = \frac{\beta\Big(x(t-\tau) + \delta I_t\Big)}{1 + \Big(x(t-\tau) + \delta I_t\Big)^n} - \gamma x(t-\tau) \tag{5}$$

This can also be expressed as a block diagram, as shown in figure 5:

Where, β is the coupling gain between the chaotic attractor and integration stage, τ is the length of the delay within the delay feedback loop, $x(t-\tau)$ is the output of the feedback loop after the delay τ, I_t is the external input to the feedback loop, δ is the input scaling factor, γ is the feedback gain within the leaky integrator, and n is the value of the Mackey-Glass exponent.

The integrator within a non-linear neuron is typically realised by a leaky integrator. However, as we are interested in creating a model which is highly parameterisable, we chose to set the leaky integrator feedback gain (γ) to unity and close the loop around the leaky integrator. This gives a first-order system that can be realised by an op-amp or a simple resistor-capacitor circuit; this is expressed within the Laplace domain as the following transfer function:

$$T(S) = \frac{\beta}{Timescale.S + 1} \tag{6}$$

Where $Timescale$ is the time constant of the system in seconds, and β is the coupling gain in between the chaotic attractor and transfer function.

Not only is the first-order system approach easier to implement within hardware, but it also allows the system to have a tuneable time constant. This allows us to configure the time constant of the reservoir to match any physical substrate, which not only produces a more realistic hardware model (giving us greater insight into the dynamics of the behaviour of the system), but is the first steps in having a configurable timescale that can be adjusted to optimise the performance of a particular computational task. An adjustable coupling gain is also included between the chaotic attractor and leaky integrator to model any losses between these stages.

B. Experimental Model

With the chaotic attractor and integration stages chosen, the delay-feedback reservoir model is constructed within Simulink, a MathWorks simulation tool (version 10.5 within MATLAB 2022a), allowing for a near physical system model to be created. The model is created using the block diagram in figure 5 as reference, the completed model is shown in figure 6:

As previously discussed in section IV-A, the Mackey-Glass chaotic attractor is used, which is built from the Simulink model blocks, and the integration stage is realised by the proposed first-order system; the delay is implemented using the Simulink "Transport Delay" block. Everything within the blue dotted line is part of the Mackey-Glass chaotic

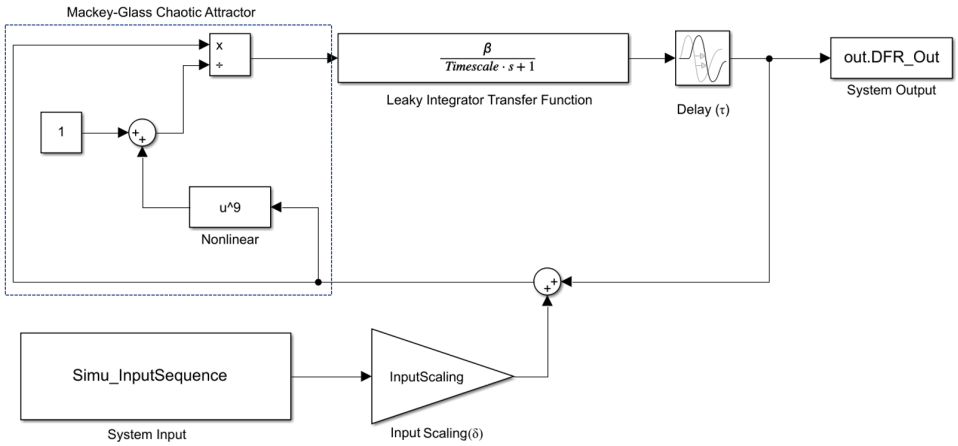

Fig. 6. A schematic of a delay-feedback reservoir, using the Mackey-Glass chaotic attractor and a first-order system as the integration stage, created within Simulink 10.5.

Name	Symbol	Default Value
Timescale	T	Variable
Coupling Gain	β	Variable
Input Scaling Factor	δ	Variable
Time Delay	τ	Variable
Mackey-Glass Exponent	n	9

TABLE I
A LIST OF MODEL PARAMETERS WHICH ARE SET WITHIN THE DELAY-FEEDBACK RESERVOIR SIMULINK MODEL.

attractor and will remain constant throughout experimentation. A commonly-used average non-linearity is chosen, as the Mackey-Glass exponent (n) is set to 9, because some non-linearity is required in the benchmarks chosen.

The input signal is generated in MATLAB and is injected into the reservoir using a "From Workspace" block, named "`Simu_InputSequence`" within figure 6. The states of the reservoir are taken using a "To Workspace" block, named "`out.DFR_Out`" within figure 6, at a sample rate of θ; allowing the states of each virtual node to be sampled at the correct time. These states are then sent back into the MATLAB workspace, so that the training and system evaluation can take place.

Table I contains a list of the parameters, and their values, within the Simulink model.

C. Experimental Procedure

With the experimental model completed, the effect of the connectivity between virtual nodes is to be investigated. 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 (as defined by the transfer function 6) and the period of the masking signal will be modified. The ρ values of the two systems and their timescales are shown in table II.

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 II
THE VALUES OF ρ FOR TWO DELAY-FEEDBACK RESERVOIR SIMULINK MODELS WITH DIFFERENT VIRTUAL NODES, WHEN TESTED AT FOUR DIFFERENT TIMESCALES.

Name	Symbol	Value (20-Node)	Value (200-Node)
Coupling Gain	β	0-2	0-2
Input Scaling Factor	δ	0-2	0-2
Time Delay	τ	80	80
Masking Signal Period	θ	4	0.4

TABLE III
PARAMETER VALUES OF THE DELAY-FEEDBACK RESERVOIR SIMULINK MODEL DURING THE INPUT SCALING AND COUPLING GAIN PARAMETER SWEEP.

As the performance of the reservoir is highly dependant upon the input scaling and coupling gain, a coarse parameter sweep is performed on both the 20- and 200-node reservoir systems and timescale setting, for both the NARMA-10 and Santa Fe benchmarks, in order to find the best performing parameters. Table III shows the parameter values and ranges for the proposed parameter sweep.

The procedure for generating the input sequences of both the NARMA-10 and Santa Fe benchmarks within MATLAB are the same. A sequence of 6000 data points, either generated from the NARMA-10 equation or copied from the Santa Fe dataset, is time-multiplexed by a random-weighted no-offset mask. A washout period (consisting of 100 data points) is used, with the training and test datasets being split 80/20 respectively.

V. EXPERIMENTAL RESULTS

Following the experimental procedure outlined in section IV-C, the parameter sweeps are performed. Each parameter sweep is evaluated by calculating the training and testing

NRMSE at each parameter interval, and is ranked in order of best testing NRMSE.

A. NARMA-10 Benchmark

Table IV shows the best performing sweep parameters, in terms of testing NRMSE, for the NARMA-10 benchmark. Within both the 20- and 200-node reservoirs, it is clear that as ρ increases, the test NRMSE decreases. This is to be expected as when ρ increases, more information is being passed on to the next virtual node, increasing the computational performance of the reservoir. However, the performance of the reservoirs are particularly poor, with the highest test NRMSE being 0.494; this is likely due to the lack of memory capacity within the system.

As discussed in section III-A, the NARMA-10 benchmark relies heavily on the long-term dependency of previous input stimuli in order to fully recreate the dynamics of the NARMA-10 signal; specifically, a lag of 10 time steps for the NARMA-10 benchmark. The maximum long-term memory capacity of a reservoir system is equal to the maximum number of nodes within the reservoir [21], although typically the actual memory capacity is much less than this. This would imply that there is insufficient memory capacity within the 20-node reservoir to provide the required long-term dependency, thus greatly reducing the computational performance of the reservoir. This is evident when comparing the 20- and 200-node reservoir systems together, as there is an improvement within the computational performance, with the highest test NRMSE being 0.308.

Despite the 200-node reservoir having a greater potential for maximum memory capacity, the performance appears to be dominated by the connectivity between the virtual nodes, implying that there is a connection between the connectivity and the memory capacity. This would make sense as ρ is defined as the time the transient response of the non-linear node exists within a single masking signal period, and the greater ρ is, the more of the previous input passes on to the next virtual node.

A further observation can be made when the value of ρ equals 0.025 within both the 20- and 200-node reservoirs. The 20-node reservoir has a minimum testing NRMSE of 0.608, while the 200-node reservoir has a minimum testing NRMSE of 0.586. This would indicate that the full dynamics of the 200-node system is not being utilised, as the 20-node system performs almost as well, suggesting a 200-node reservoir could be emulated within a 20-node reservoir providing they have the same value of ρ. This can not be said with confidence as each parameter sweep consists of only one run. Therefore, further statistical analysis must be performed to confirm this hypothesis.

B. Santa-Fe Benchmark

Table V shows the best performing sweep parameters, in terms of testing NRMSE, for the Santa Fe benchmark. One of the first things to note is that the computational performance of the reservoir running the Santa Fe benchmark is much better than the NARMA-10 alternative, with the lowest test NRMSE being 0.098. This suggests that the current model configuration is better suited to low-memory, high-dimensional benchmark tasks.

Although there is a relationship between the computational performance of the reservoir and the value of ρ, it is not as prominent as it is within the NARMA-10 benchmark. Within the 20-node reservoir, a gradual decrease occurs within the testing NRMSE, implying the dominant factor within the reservoir performance is the connectivity between the virtual nodes. However, within the 200-node reservoir, there is a large decrease in testing NRMSE from the 10ms and 100ms reservoir, but as ρ increases, there is only a minor effect on the computational performance when ρ is greater than 0.25. This would imply that other dynamics, such as the non-linearity of the reservoir, dominate the value of ρ within the 200-node reservoir. This would make sense as the Santa Fe benchmark requires only minimal memory capacity, as it only has 1 lag dependency on previous input stimuli, but requires strong non-linear high-dimensionality.

During the evaluation of the NARMA-10 benchmark, it was noticed that the 100ms 20-node reservoir and the 10ms 200-node reservoir performed almost the same. 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 reservoir, indicating that a 200-node reservoir may be able to be emulated within a 20-node reservoir. For the Santa Fe benchmark, the testing NRMSE value for the 100ms 20-node reservoir is 0.416 and for the 10ms 200-node reservoir it is 0.374. It is not clear whether the dynamics of the reservoir are the dominating factor, or whether it is the value of ρ. To determine if a reservoir running the Santa Fe benchmark can be emulated, a full statistical evaluation will need to be performed in future work.

VI. System Emulation

Within section V, it was 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 was found that when running the NARMA-10 benchmark, the 100ms 20-node system and the 10ms 200-node system had similar performance, having a testing NRMSE difference of only 0.022; implying that the NARMA-10 benchmark could be emulated. Conversely, during the Santa Fe benchmark run, the 100ms 20-node system and the 10ms 200-node system had a testing NRMSE value of 0.416 and 0.374 respectively, making it unclear if the Santa Fe benchmark could be emulated.

To check the validity of this hypothesis, the best performing input scaling and coupling gain parameters that were determined, from the parameter sweeps within section V, were simulated 100 times so that a statistical analysis could be performed. Each set of 100 simulations has the same input scaling, coupling gain, and value of ρ, but has a different randomly generated random-weighted no-offset mask; the NARMA-10 benchmark also has different randomly generated

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
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 IV
TABLE OF THE BEST PERFORMING SWEEP PARAMETERS FOR THE NARMA-10 BENCHMARK, FOLLOWING THE METHODOLOGY OUTLINED IN SECTION IV-C.

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
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 V
TABLE OF THE BEST PERFORMING SWEEP PARAMETERS FOR THE SANTA-FE BENCHMARK, FOLLOWING THE METHODOLOGY OUTLINED IN SECTION IV-C.

input. The testing NRMSE values were calculated from the resulting simulations and plotted as a box-plot; this is shown in figure 7.

The NARMA-10 results shown in figure 7 show that the value of ρ is the dominating factor when it comes to the performance of the systems, and is independent of the number of virtual nodes within the reservoir. The 20- and 200-node systems (which share a ρ value of 0.025) have almost the same performance, with the 200-node reservoir having a slight advantage. This is what we would expect, 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.

The Santa Fe results, once again shown in figure 7, shows that the number of virtual nodes within the reservoir is the dominating factor in terms of the performance of the systems. 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 were already on the edge of stability, meaning the random deviations within the random mask could have forced the reservoir to perform worse. The effect is much more prominent within the 20-node reservoir as the 200-node reservoir is more robust.

These results confirm the hypothesis that 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 that of the previous system. The limiting factor of how small the new system can be, is that the new system must exhibit the minimum non-linearity and dimensionality required to run the computational task.

VII. CONCLUSION AND FURTHER WORK

The delay-feedback reservoir is a highly configurable and versatile method of performing temporal computational tasks. It is highly configurable in the sense that there are many parameters that can be changed with relative ease, both with the reservoir and pre-processing stage, that have a large effect on the performance of the system.

Within this paper, we created a parameterisable realistic hardware model of a delay-feedback reservoir that is able to emulate both simulated and physical systems in order to further the understanding of the parameters and functionality of the system. We have shown the importance of the virtual node connectivity and how it can affect the computational performance of two computational benchmarks with opposite operating requirements; a high-memory, low-dimensionality benchmark task (NARMA-10) and a low-memory, high-dimensionality benchmark task (Santa Fe). To focus the scope of this paper, we have not significantly varied or explored parameters within the non-linear neuron, although this is an avenue for future research. We found that it was possible to emulate larger virtual node networks within smaller ones, providing that the computational task has a large dependency on previous input stimuli and low-dimensionality, with little performance loss. This allows the period of the masking signal to be larger, which reduces the frequency of the masked input stimuli, allowing for the input data to be applied at a higher frequency.

In future work, we will further explore the effect of the timescale within the delay-feedback reservoir to create a

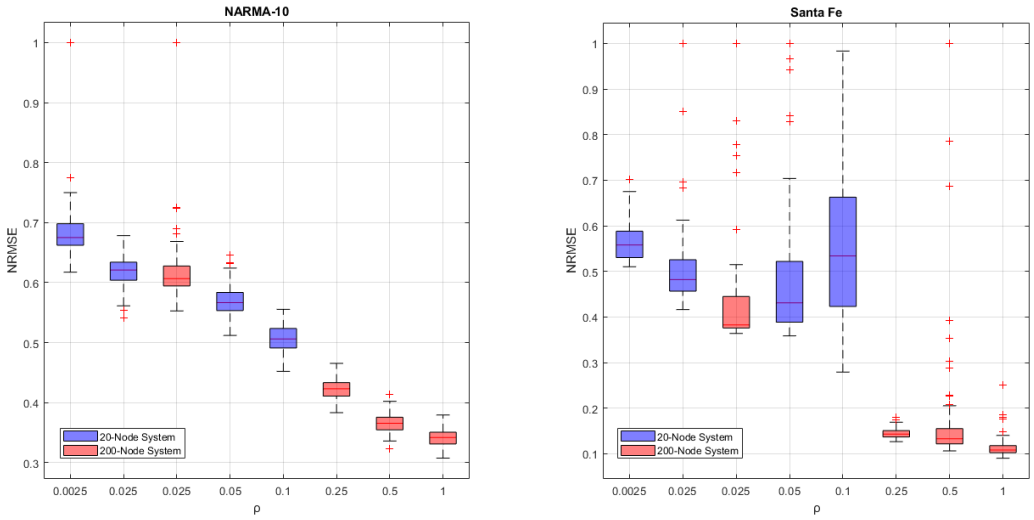

Fig. 7. A graph showing the training NRMSE results of 100 simulation runs for several values of ρ. Each box-plot is a reservoir configured with the best performing input scaling and coupling gain parameters found in section V.

feature space of the system in terms of the system parameters, allowing a view of their sensitivity. To better understand the difference in performance between the NARMA-10 and Santa Fe benchmark tasks we will also analyse the system in terms of metrics (such as memory capacity, kernel quality, and generalisation rank) and perform further computational benchmarks that require less extreme reservoir characteristics on the system. We will also investigate the performance of the system with additional benchmarks that have a mixture of memory capacity and dimensionality, not just the opposite extremes. This will allow us to gain insight into why the benchmark tasks perform differently, and hopefully allow us to identify the key properties that are needed for good performance.

Acknowledgment

Supported by the EPSRC studentship. Experiments were carried out using the Viking Cluster, a compute cluster provided by the University of York.

References

[1] M. Brin and G. Stuck, *Introduction to dynamical systems*. Cambridge university press, 2002.

[2] 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.

[3] 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.

[4] 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.

[5] 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, pp. 1–10, 2017.

[6] C. Fernando and S. Sojakka, "Pattern recognition in a bucket," in *European conference on artificial life*. Springer, 2003, pp. 588–597.

[7] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 659–686.

[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 comms.*, vol. 2, no. 1, pp. 1–6, 2011.

[9] F. M. Atay, *Complex time-delay systems: theory and applications*. Springer, 2010.

[10] 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.

[11] 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.

[12] 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.

[13] T. Hülser, F. Köster, L. Jaurigue, and K. Lüdge, "Role of delay-times in delay-based photonic reservoir computing," *Optical Materials Express*, vol. 12, no. 3, pp. 1214–1231, 2022.

[14] 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.

[15] A. S. Weigend, *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

[16] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, "Information processing via physical soft body," *Scientific reports*, vol. 5, no. 1, p. 10487, 2015.

[17] 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.

[18] 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.

[19] 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.

[20] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[21] H. Jaeger, "Short term memory in echo state networks. gmd-report 152," in *GMD-German National Research Inst. for Comp. Science (2002), http://www. faculty. jacobs-university. de/hjaeger/pubs/STMEchoStatesTechRep. pdf*. Citeseer, 2002.