Physical reservoir computing for classification of temporal data

Charlie Bushman

April 9, 2021

Abstract

Reservoir computing is a subclass of machine learning wherein a high dimensional, nonlinear neural network, the reservoir, transforms input data to a space in which it can be linearly classified by a single output layer. Over the last decade, physical systems which meet certain criteria — having complex nonlinearity, high dimensionality and the echo state property — have been implemented as reservoirs, harnessing the natural complexity of their dynamics to reduce computational complexity on computers. Two example reservoirs which we investigate are an ensemble of coupled mechanical Duffing oscillators and a recurrent quantum circuit that couples transmon qubits on one of IBM's cloud based quantum computers. While this is still a relatively new field, the potential for chaotic systems prediction, smart physical sensors and more is potentially revolutionary across a wide range of sciences and industries.

Contents

1 Introduction

Reservoir computing (RC) is a machine learning (ML) framework that has gained substantial interest over the last decade for its ability to harness the computational power of complex physical systems. The RC model makes use of a reservoir, which can be any suitably complex network meeting certain criteria, to mimic the dynamical topology, or the way in which a system evolves, of the system from which its inputs come [1]. This mimicry occurs within the reservoir and can be extracted by a single output layer that is trained with a computationally easy linear regression scheme. The trained RC system is then able to provide state of the art performance in short term predictive tasks or long term classification, the latter of which we will focus on in this paper. The reservoir can be a simulated graph (or neural network) on a computer, as is typical in most ML schemes, but it can also be a physical system, separate from the computer [2]. The latter offloads all the computational complexity of simulating those reservoir dynamics from the computer to the physical system. So physical RC provides a framework with lower upfront training cost and lower computational complexity but requires access to a physical reservoir.

In this paper, we will introduce the fundamental concepts of ML and the RC framework in Section 2 and then discuss the four essential features of a good reservoir in Section 3. We will then have all the necessary background to look at two examples of physical reservoir implementations in Section 4. In Section 4.1, we will see a proposed physical RC based on coupled nonlinear, mechanical oscillators. Then in Section 4.2 we will see the first published example of a quantum RC where the reservoir is a recurrent quantum circuit on a noisy intermediate-scale quantum (NISQ) computer. Additionally one of the first published examples of a physical RC in the form of an electronic Mackey-Glass circuit is examined in Appendix A.

2 Machine Learning and Reservoir Computing

Reservoir computing is a machine learning framework that is based on recurrent neural network (RNN) theory. Algorithms in RNN theory are produced via training of a neural network (NN) comprised of input and output layers connected to a reservoir. This structure is unique from the prototypical NN, in which there is no reservoir but instead a well-structured set of node layers [3]. While the traditional structure enables a very efficient algorithm to be trained, the training cost is also higher than in RC. And although RC algorithms are more computationally complex, when that complexity is outsourced to a physical system it leaves a framework with lower training overhead and lower complexity for the computer. The reduced complexity for the computer frees it up to perform other tasks for which physical systems are not well suited.

2.1 The Basics of Machine Learning

Broadly speaking, ML is any process by which a system gains some propensity for answering a certain question. In the most typical use case, a computer is used to simulate a neural network which is then trained to perform a task. Neural networks and training are both incredibly expansive topics but the core concepts are relatively simple. A neural network is a graph, a collection of nodes with associated values and edges connecting those nodes together [4]. This graph has weighted, directional edges — meaning each edge goes from one node to another (and not necessarily vice versa) — and each edge has an associated value that scales signals passing through it. These signals can be time-discrete or time-continuous evolutions of the graph, where nodes push their values along all outward-facing edges[1] and the edges scale that input's effect on the recipient nodes proportionally to the edge's weight. A schematic of a typical neural net is shown in Fig. 1.

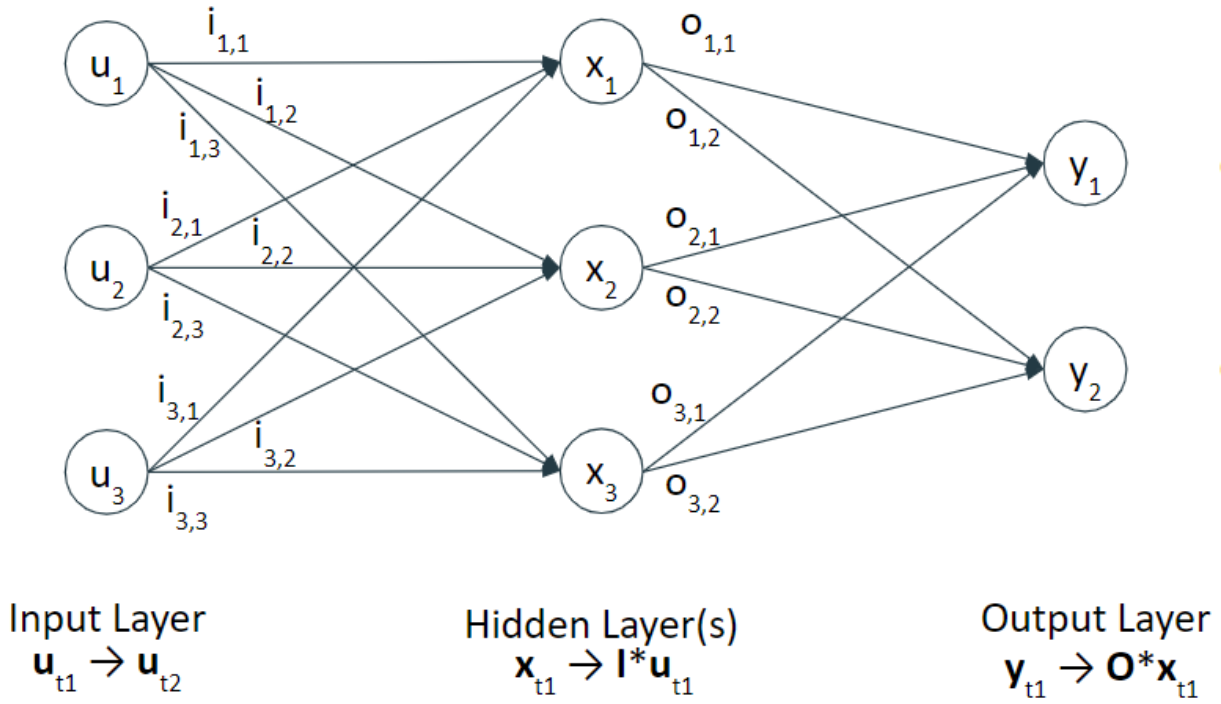

Figure 1: A graph representation of a typical feed forward neural network (FFNN) with three layers. The input layer, labelled by a vector u, maps input data onto the hidden layer nodes, labelled by x, which in turn map onto the output layer nodes, labelled by y. These mappings are accomplished by the weight matrices I and O, made up of the edge weights $i_{m,n}$ and $o_{m,n}$. The evolution of each layer of the network is given by the processes shown at the bottom of the figure. If this neural net is being trained for classification then we would expect that one output node is significantly higher-valued than the rest to indicate the correct classification of the input.

[1] An outward facing edge is one that is both directional and points from the node in question to another node.

Once a neural net is trained, a user can push inputs into a designated set of nodes, known as the input layer, and then will get outputs from another designated set, known as the output layer. So long as the neural net has been trained properly, the whole network can be treated as a blackbox function that just works. The training process is typically a minimization problem over some loss function that provides a measure for the difference between the expected output and the output given by the neural net.[2] To achieve this minimization, the weights of the neural net are adjusted. There are many methods for doing this but the details are beyond the scope of this paper.

2.2 Machine Learning Tasks

So far we have looked at how ML works but we haven't yet identified what kinds of problems it can solve. While there is a huge range of problems that ML is equipped to address, they can be generalized into many types including binary classification, multiclass classification and regression [3].[3] A binary classification task is defined by the process,

$$f(x \in \mathcal{X}) = y \in \{0, 1\}, \tag{1}$$

where x is the input chosen from the space of possible inputs $\mathcal{X}$, y is the binary output and f is our learned function, in essence the action of the neural net on x. A simple example of this sort of task is trying to tell apples from oranges: given some image of an apple or an orange, classify it as one or the other. In this example, the input x is some image from $\mathcal{X}$, the set of all images of apples and oranges, and y can be either 'apple' or 'orange'. See Section 3.2 for a visual example of these ideas.

Similarly there are multiclass classification tasks, which are defined by the process,

$$f(x \in \mathcal{X}) = y \in \{0, 1, ..., n\}, \tag{2}$$

where the only difference is that there is now a larger, non-binary output space $\mathcal{Y}$. The canonical example for this type of task is classifying handwritten digits zero through nine.

The third type of ML task we will see in this paper is known as regression and is defined by the process,

$$f(x \in \mathcal{X}) = y \in \mathbf{R}, \tag{3}$$

[2]For training data sets, you need to have both inputs and the outputs you expect those inputs to yield.

[3]This generalization is not well agreed upon nor is it comprehensive of all ML models but it is sufficient to encompass all the tasks we will encounter throughout this paper.

where the output y can now be any real number. A simple example of a regression task might be given a person's diet, sleep patterns and age, predict that person's body fat percentage. Of course there are many, many more variables we would want to incorporate into this model like exercise, job and gender but even without having every variable that affects a person's body fat percentage we can still (sometimes) get an effective algorithm. Traditionally classification and regression tasks are differentiated by the idea of labelling versus evaluating,[4] but when viewed in the form given by Eqs. 1-3 it is easier to see the mathematical connection between the two, specifically that regression is really just classification in the limit that classes are continuous rather than discrete [5].

2.3 Recurrent Neural Networks

The basis of all ML is in training and using neural networks but there is a wide variety of categories into which neural nets can fall, the broadest two being feed forward neural nets (FFNN) and feed backward neural nets (FBNN). The defining factor differentiating the two is whether or not there are closed loops in the network. A FFNN is comprised of layers of nodes where each node in a layer is unidirectionally connected to each node in the next layer and the first and last layers are the input and output layers respectively, as seen in Fig. 1. In FFNN training is performed on input/output pairs without any regard for the ordering of inputs with respect to each other. A picture of an orange should be identified as such no matter what picture is identified before it.

Meanwhile a FBNN has some backward facing connections denoted by the dotted edges in Fig. 2. While this might not seem like a big change from FFNN architecture, the implications are integral to RC working. This is because backward connections allow the neural net to have memory, meaning that the order in which inputs are fed in now impacts the classification of those inputs [6]. For example, if we vectorize images of oranges and apples and input them to a FBNN one pixel at a time we would expect that the order in which we input pixel data is important to properly identifying the image. This can only be accomplished with internal memory, which can only be accomplished with feedback. FBNN are also commonly referred to as RNN [4].

[4]For applying labels think of 'apple' versus 'orange' or zero through nine as labels the output applies to the input, whereas regression assigns a value to the input, for example saying that an image is 0.95 apple and 0.03 orange and 0.02 something else.

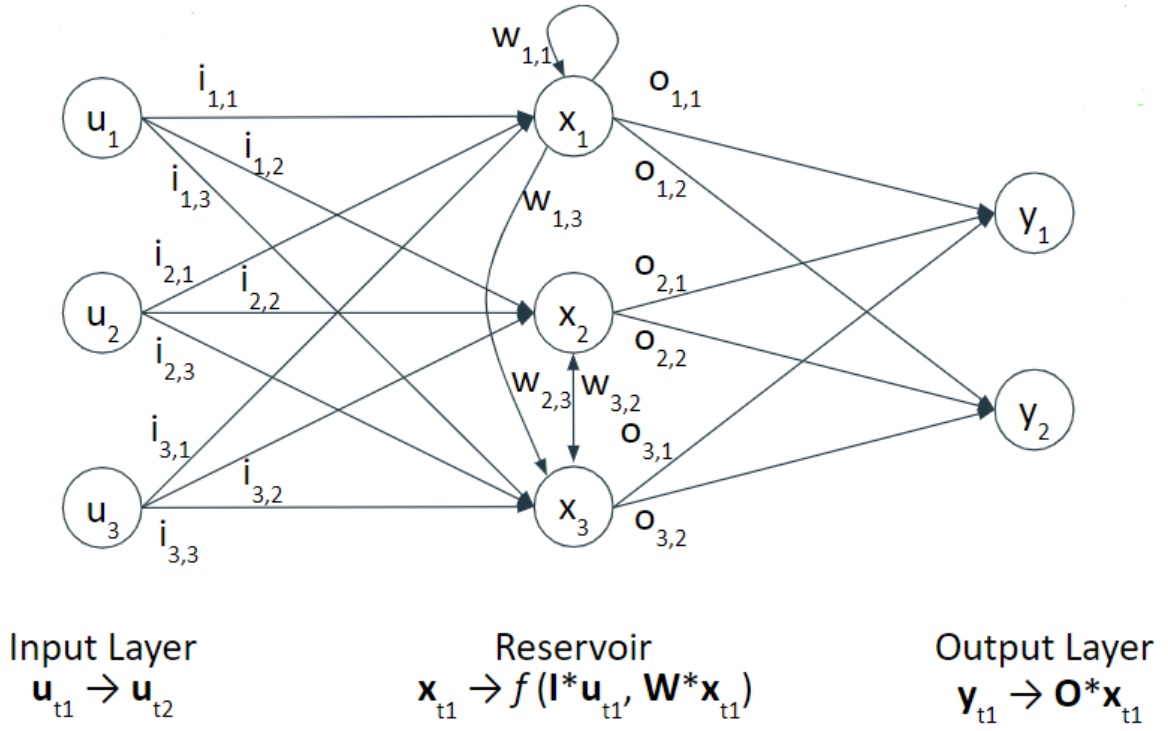

Figure 2: A graph representation of a typical recurrent neural network with three layers, an input layer, the reservoir (hidden layer in FFNN), and an output layer. As in Fig. 1, each layer maps onto the nodes of the next layer but additionally here there are recurrent relations, denoted by the $w_{m,n}$ edges. For simplicity here there are only recurrent connections from the reservoir onto itself but many others are possible, for example from the output layer back into the reservoir.

2.4 The Reservoir Computing Framework

Reservoir computing is built on top of the FBNN structure introduced in this Section. The standard model of RC has three components to it: the input, the reservoir, and the output. In the simplest cases, the input and output layers are each a single layer of nodes with each node unidirectionally connected to each node within the reservoir. The input layer has a number of nodes corresponding with the size of input vectors at each time step and the output layer usually has a number of nodes corresponding to the number of labels. In the case of regression, a node's value can be read instead of picking the label associated with the output node that has the largest value. In almost any real-world application, there will also need to be pre- and post-processing of data and outputs.

The reservoir is the component that does the heavy lifting in terms of computational complexity. While Fig. 2 gives a simple depiction of the structure, the reservoir is significantly more complicated than what is

shown in the diagram. Additionally there can be recurrent connections with the input and output layers. The full dynamics of the system will be described by Eqs. 4 & 5 or by Fig. 3 (they contain the same information, one graphically and one algebraically) [7].

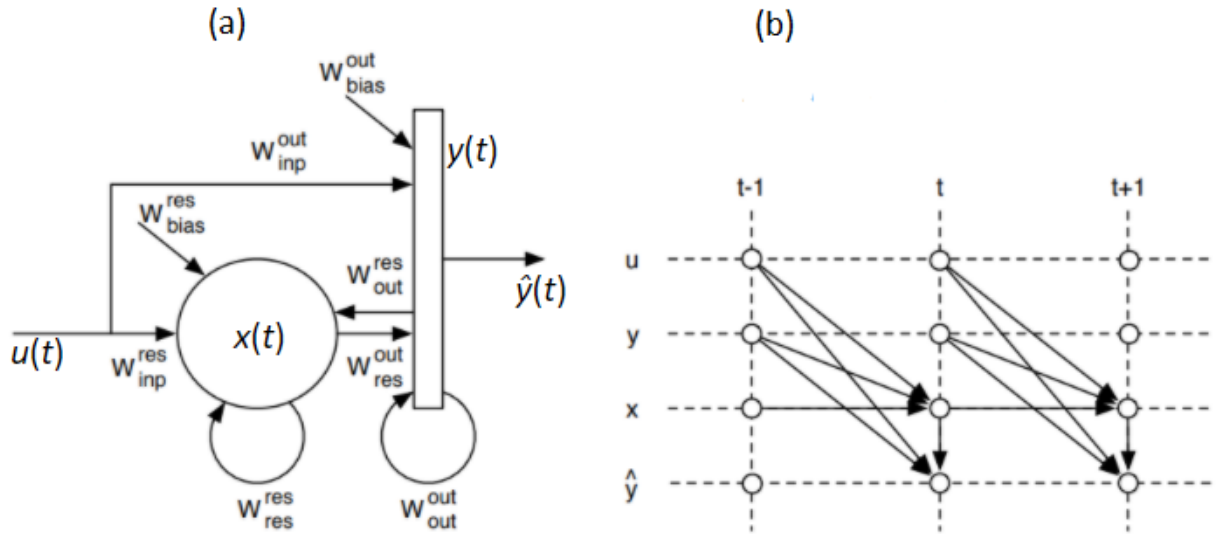

Figure 3: A high level overview of the processes in the RC framework. In (a) is a graph similar to Fig. 2 of the input layer (coming in from the left) mapping onto the reservoir (the large central circle) which then maps onto the output layer (the far right rectangle). This depiction also has many additional mappings between these layers which aren't seen in Fig. 2. In (b) is a temporal diagram describing the same mappings with well defined time steps [7].

For a reservoir of N nodes with input vectors, $u(t)$, of dimension Q and output vectors, $y(t)$, of dimension O the evolution of the reservoir's state vector[5] $x(t) \to x(t+1)$ can be described by,

$$x(t+1) = f\ [\ W^{res}_{res}x(t) + W^{res}_{inp}u(t) + W^{res}_{out}y(t) + W^{res}_{bias}\ ], \tag{4}$$

where $W^{res}_{res} \in \mathbf{R}^{N\times N}$ maps the current state vector $x(t)$ onto the reservoir, $W^{res}_{inp} \in \mathbf{R}^{N\times Q}$ maps the input $u(t)$ onto the reservoir and $W^{res}_{out} \in \mathbf{R}^{N\times O}$ maps the expected output $y(t)$ back into the reservoir. Additional bias can be provided to the reservoir in the form of a bias vector $W^{res}_{bias} \in \mathbf{R}^{N}$. Practically, bias can be thought of as some constant offset that is being applied. Typically $f(x)$ is chosen to be a sigmoid activation function such as $\tanh(x)$.[6] The computed output vectors can then be described by:

[5] A state vector is simply a vector that contains the state, or values of each node, of a layer of the network in vector form.
[6] A sigmoid activation function is one that can take in an arbitrarily sized input and output something in the range (0,1).

$$\hat{y}(t+1) = W_{res}^{out} x(t+1) + W_{inp}^{out} u(t) + W_{out}^{out} y(t) + W_{bias}^{out}, \tag{5}$$

where $W_{res}^{out} \in \mathbf{R}^{O \times N}$ maps the reservoir's current state vector onto the output, $W_{inp}^{out} \in \mathbf{R}^{O \times Q}$ maps the input directly onto the output, $W_{out}^{out} \in \mathbf{R}^{O \times O}$ maps the expected output onto the computed output and $W_{bias}^{out} \in \mathbf{R}^{O}$ provide fixed bias values. Bias matrices can often be ignored in theory and only need to be included for offsetting practical inconveniences in a physical reservoir. Also there are many examples where some of these matrices are zero and don't affect the reservoir's evolution. There are two values $y(t)$, the output vector, and $\hat{y}(t)$, the computed output vector, which are often confused. The recurrent relations onto the output layer, seen in Fig. 3, necessitate that both of these output vectors exist. $y(t)$ contains the mutable information that is still "within" the network while $\hat{y}(t)$ is the fixed readout.

One of the advantages that RC offers over other ML architectures is that only the matrices in Eq. 5 have to be trained and this can be achieved by a relatively simple, low-cost training scheme [2]. This is in contrast to FFNNs, where training algorithms must account for every hidden layer in the network. For the deepest NNs, this can range as high as multiple hundreds of layers and the number is increasing every year.

3 Reservoir Characteristics

In traditional computational approaches to RC, the reservoir itself is a network simulated by the computer, as in most other forms of ML. With this form, the entirety of an RC algorithm would take place on the computer's hardware. However, it is possible to outsource the reservoir to a physical system instead, leaving the computer with only the tasks of data processing and training on the output weights. This has the advantage of massively reducing the computational complexity of RC problems (for the computer) but requires access to a suitable physical reservoir to take on that complexity. An effective physical reservoir must have a number of properties including nonlinearity, high dimensionality and fading memory as detailed in this Section [7]. Further properties of significance to identifying effective reservoirs are discussed in Appendix B.

This is important in recurrent or multilayer NN to keep the node values from growing out of control, since every operation performed on them is additive.

3.1 Nonlinearity

"Nonlinear" is a descriptor that can be applied to a huge range of fields and phenomena because it encapsulates anything that does not have linear behavior. An oft-quoted parallel is that the study of nonlinearity is like the study of non-elephants; it is almost a ridiculously wide field which is why there are so many possibilities when it comes to creating physical reservoirs [8]. There are some qualifiers beyond just being nonlinear though, that narrow the field of suitable candidate systems considerably. Most importantly, we need a system that is chaotic,[7] but identifying chaotic systems can be difficult.

There are many tests for identifying chaotic systems but none of them are conclusive without knowing the governing equations for the dynamics. In most cases with real systems though, we will not know these governing equations and will have to make somewhat aesthetic arguments based on one or multiple time series data streams read out from the system. One of the most common methods is to visualize the chaotic attractor in the system's phase space[8] to look for the characteristic "strange" orbiting behaviour expected, an example of which is seen in Fig. 4 for the orbit of the mechanical reservoir introduced in Section 4.1. A problem with this method is that often with real systems it can be difficult to identify which measurements to make to yield each of the independent variables defining the system. In cases like this, instead of visualizing the attractor in phase space, we can create a close facsimile with a time delay embedding of one time series data stream. This entails taking one data stream and then plotting it against itself at multiples of some time delay, τ. If the variables of the system are tightly coupled,[9] as is the case in the systems we will be considering in this paper, then this embedding provides a very similar visualization of the attractor as a phase space plot. In fact, the plot in Fig. 4 is a time delay embedding plot but is almost indistinguishable from a phase space plot of the same system.

Another method for identifying chaotic systems is creating bifurcation plots of the system as it varies over a range of parameter values. A bifurcation plot takes the local maxima and minima of a single time series stream over some set amount of time and plots those values for the current parameter set of the system. Then some parameter in the system is incremented and the process is repeated, plotting the new maxima and minima with the new parameter value. In non-chaotic regions of the parameter space we expect to see fairly consistent behaviour as the parameter increments but in chaotic regions, where there is an exponential

[7] Chaotic systems form a proper subset of nonlinear systems, which is to say, all chaotic systems are nonlinear but not all nonlinear systems are chaotic. In theory it is possible to have an infinitely high-order linear system be chaotic but there is no practical reason to consider these systems. A chaotic system is one characterized by an infinitesimal dependence on initial conditions, with tiny differences causing exponentially diverging evolutions.

[8] A chaotic or strange attractor is a point in phase space which the trajectory of a system orbits as it evolves through time. Phase space is a space where every point corresponds to a unique state of the system.

[9] A system with tightly coupled variables are one's in which the components tend towards each other more than they are pulled apart.

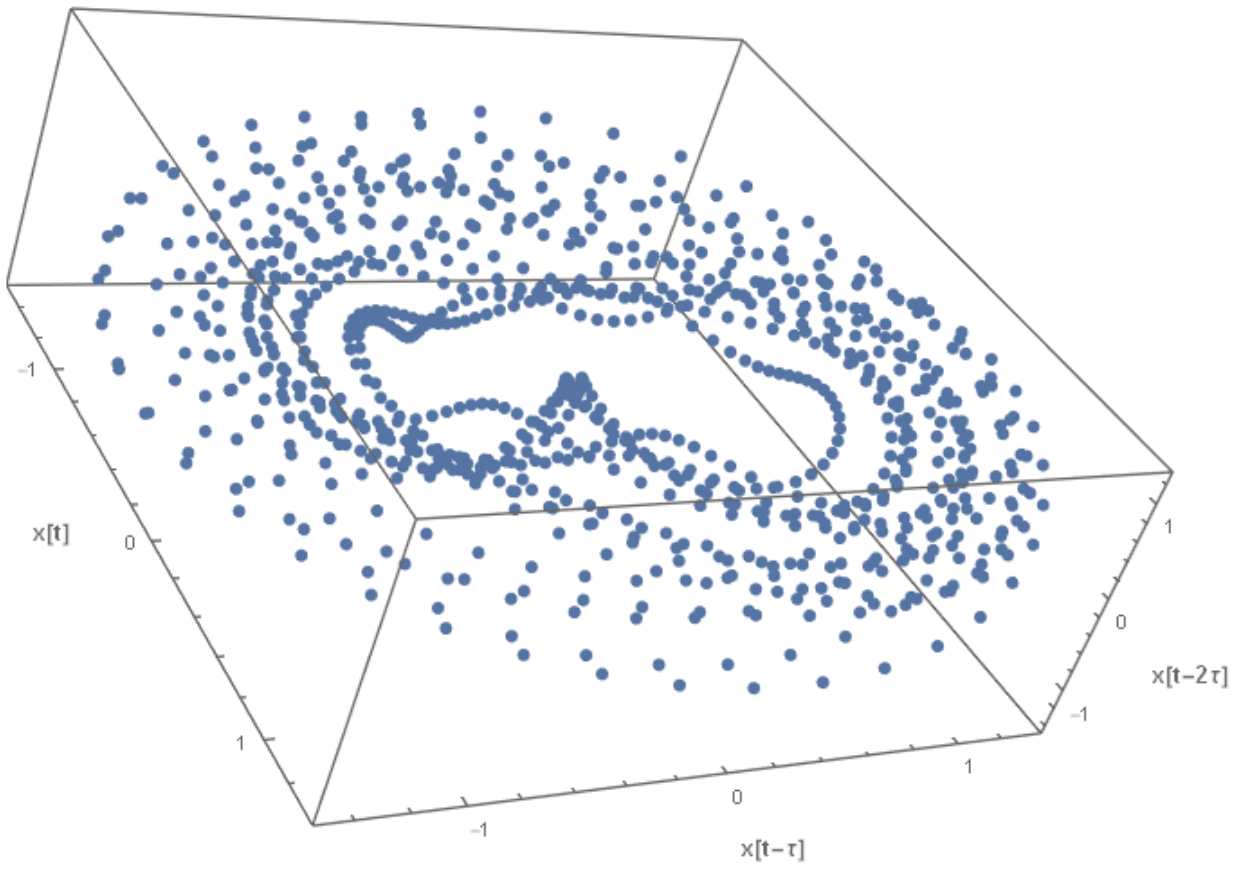

Figure 4: A time delay embedding visualization of the attractor for the mechanical reservoir introduced in Section 4.1. The time series data used is the position of the first oscillator in the chain, which is then plotted against itself at time delays of 6 and 12 time steps (where each point represents one time step). This attractor very clearly exhibits chaotic orbiting dynamics as it evolves.

dependence on changes to initial conditions, we expect to see the behaviour of the system varying wildly from one parameter set to the next. An example of a bifurcation plot for the physical reservoir examined in Section 4.1 in Fig. 5 shows a chaotic region on the left gradually stabilizing into a periodic region on the right as the resonant frequency of an oscillator is varied.

For classification tasks, there is some potentially nonlinear mapping between the input space $\mathcal{X}$ and the output space $\mathcal{N}$. In other ML models, the NN layers build up a series of mappings to eventually get from $\mathcal{X}$ to $\mathcal{N}$ but with RC, we can rely on the reservoir to provide *some* nonlinear mapping from $\mathcal{X}$ to another space $\mathcal{R}$ such that we can then train a linear mapping from $\mathcal{R}$ to $\mathcal{N}$. In this way, the only training we perform is linear regression,[10] and the complexities of nonlinear mapping are handled by the reservoir which we do not

[10]Linear regression should not be confused with regression tasks, although they have similar guiding principles they are not the same. There are many methods for performing linear regression but the most commonly used is a simple least-squares method.

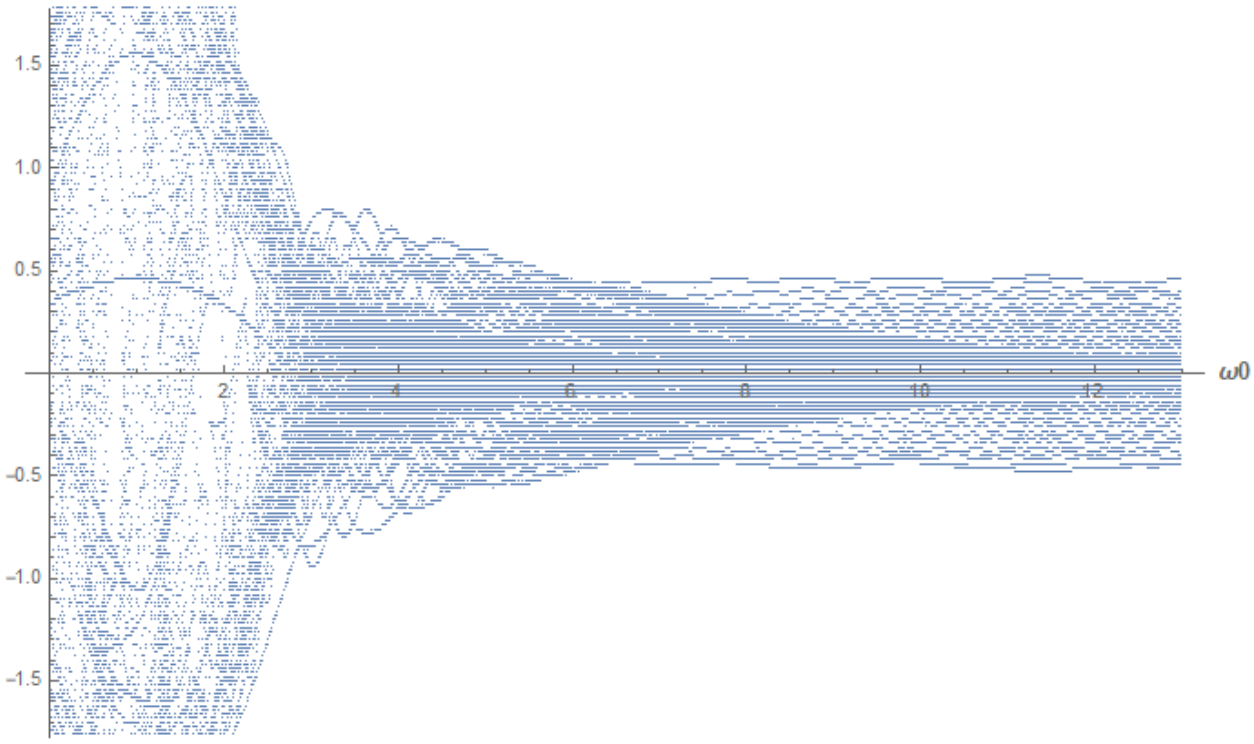

Figure 5: The bifurcation diagram for the mechanical reservoir examined in Section 4.1 as the resonant frequency, ω_0, is varied over the x-axis, ω_1 is 1.5, Q is 60, β_1, β_3 are 1 and β_2, β_4 are 0.05, A is 0.8, and Ω is 1.14. The y-axis shows the local maxima and minima of the oscillations of the first oscillator of the four that are coupled. On the left is a chaotic region of the parameter space and as the resonant frequency drifts away from the driving frequency we see the system stabilize into a periodic region.

have to train. This also clarifies the need for a chaotic system if we consider that many of the predictive tasks we would want a RC to handle would likely be chaotic in nature. Things like the stock market, chemicals, and speech are all chaotic data sources for which we would need a chaotic reservoir to create the necessary mapping from $\mathcal{X}$ to $\mathcal{R}$ for tasks like trading algorithms, molecular simulations, or speech prediction.

3.2 High Dimensionality

As mentioned in Section 3.1 the dimensionality of a reservoir is extremely important in an effective RC. But before jumping into why, we should take some time to try to understand what the "dimension" of a reservoir actually is. The dimension of a reservoir with n nodes is simply n but the meaningful dimension, also commonly called the "signal dimension", is actually that of the lower-dimensional surface on which signals in the reservoir propagate. There turn out to be a lot of ways to define this signal dimension, each with pros and cons and none having a compelling claim to be the absolute measure of a system's signal

dimension. In this paper, we will consider a measure which is well suited to the systems we investigate in Section 4 known as the information dimension [9].

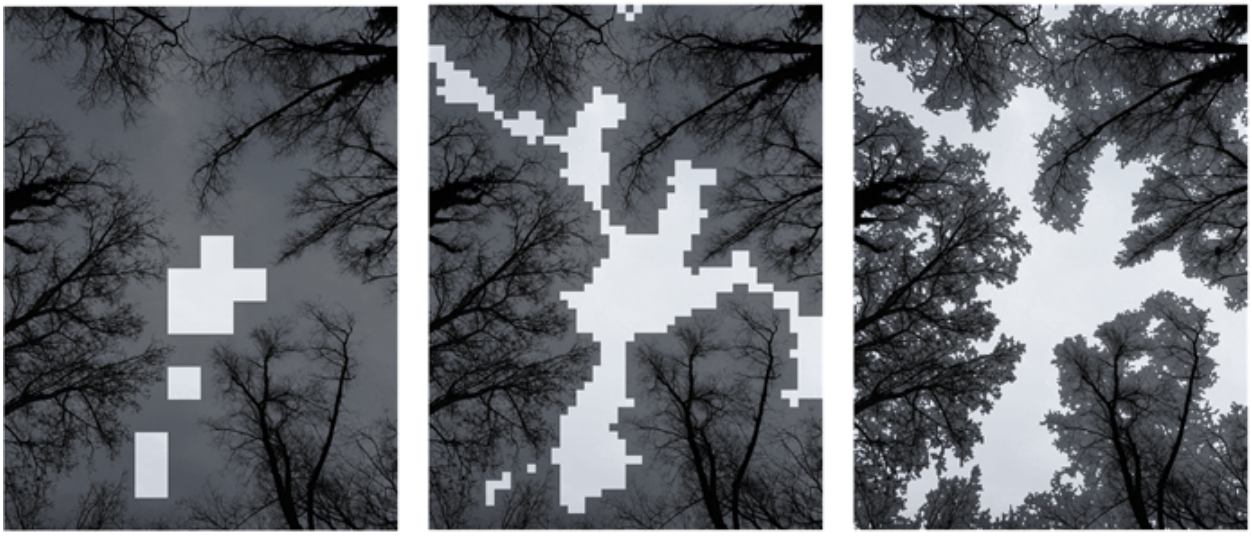

Figure 6: The fractal dimension of tree branches can be measured using a cellular automaton based method shown here. In each step, the algorithm counts how many boxes don't have any of the tree shown in them with unit boxes laid out on a grid. Then the box size is decreased and the process is repeated. The end result should be an exponential curve with a scaling factor equal to the fractal dimension of the tree branches [10].

The information dimension can be summarized as a measure of the increase in information as you zoom in on the system's phase space. This is typically applied to fractals with a very similar method of counting as in cellular automaton methods for finding the fractal dimension, seen in Fig. 6. With these methods, successively finer grids are imposed on top of the fractal, discretizing the phase space into smaller and smaller chunks. At each step, the number of grid squares without any of the fractal in them is counted. The resulting curve of box size versus boxes counted is then an exponential with a scaling factor equal to the fractal dimension of the system [10]. Although the reservoirs we consider are not prototypical fractals, all chaotic systems have enough self similarity that this method for determining the information dimension applies. While we will be relying mostly on aesthetic arguments in this paper, for the sake of completeness, here we introduce a closed form equation to represent the same process,

$$d(X) = \lim_{m \to \infty} \frac{\mathbf{H}_0(\langle X \rangle_m)}{\log_2 m}, \tag{6}$$

where $d(X)$ is the information dimension of our random variable X, $m \in \mathbf{Z}^{>0}$, and $\mathbf{H}_0$ gives the Shannon entropy of $\langle X \rangle_m = \frac{\lfloor mX \rfloor}{m}$ [11].

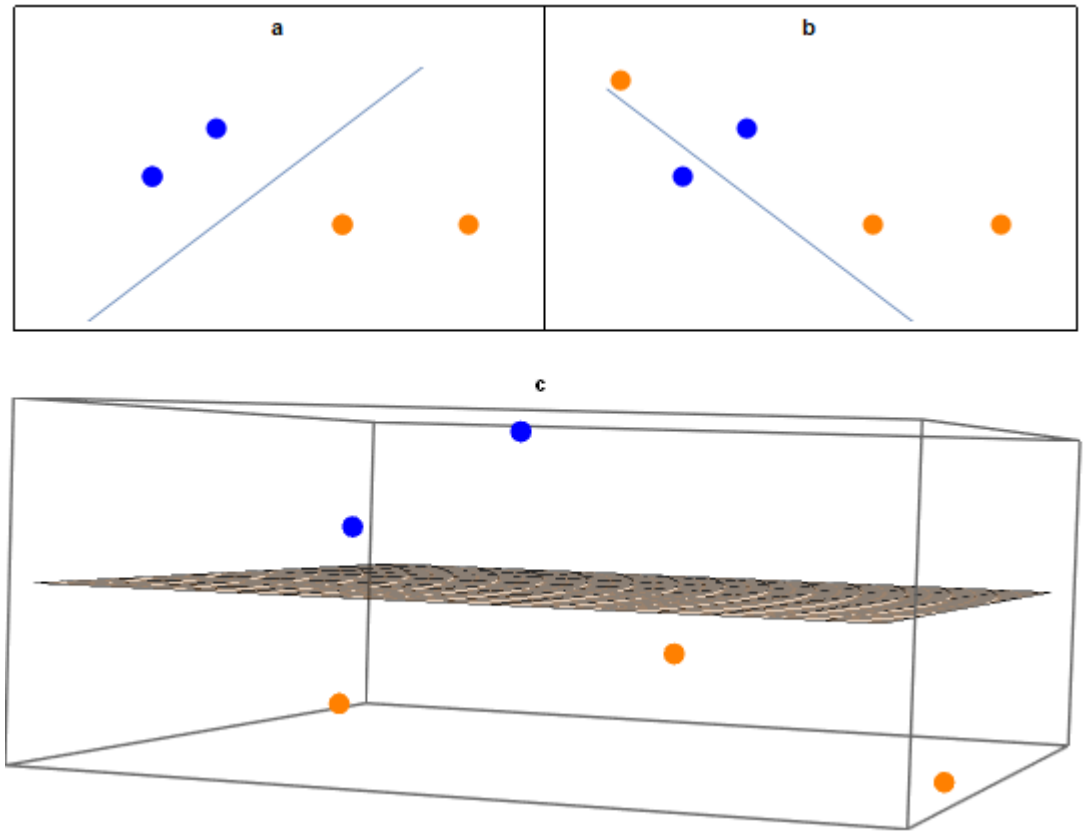

Figure 7: These three plots represent binary classifications on data sets. In (a) four points are being classified into blue and orange while in (b) a fifth data point is added such that the data set cannot be classified by linear regression. This can be remedied in (c) by promoting the data set to a higher dimensional space where a linear separator exists.

The importance of having a high dimensional reservoir is in the creation of the nonlinear mapping from the input space $\mathcal{X}$ to the reservoir's signal dimension $\mathcal{R}$. If $\mathcal{R}$ is high dimensional enough and exhibits sufficient separability for the size of the output space $\mathcal{N}$ (see Appendix B) then there will necessarily exist some linear map from $\mathcal{R}$ to $\mathcal{N}$. For a simpler, visual representation of this, see Fig. 7. In this example, there is some data set comprised of blue and orange points and we are performing a classification of it using a linear separator.[11] In (a), this is not difficult but in (b), with the addition of one more point, it becomes impossible. However, if the reservoir promotes the data to a higher dimension, as in (c), then the task becomes possible again still using a linear separator [9]. This is the true power of physical RC: being able to offload all the work of promoting data to a higher-dimensional, computational space such that all the computer has left to do is find a linear separator.

[11] We can only use a linear separator here because we only train the single, linear output layer in RC. These points are what are being spit out after the reservoir promotes the input data to its higher dimensional computational space.

3.3 Echo State Property

The fading memory or echo state property (ESP) is a fairly simple concept that is incredibly difficult to test for in a system. The principle is simply that the system's dependence on initial conditions vanishes with time or,

$$\lim_{t\to\infty} x(t, x_0) = x(t, \hat{0}), \tag{7}$$

where $x(t, x_0)$ is the state vector of the reservoir at time t and with initial conditions x_0 being any state vector. The idea is that we will often not be able to control the initial conditions of a reservoir precisely so having this property remedies the problem of running each input on identical systems to produce consistent outputs [12].

However, determining whether or not a given reservoir has the ESP is a difficult problem that is somewhat open ended. There are tests for it but none have been discovered that are mathematically conclusive. Again, for completeness, we introduce here the most common method which is to calculate the spectral radius of W_{res}^{res},

$$\rho(W_{res}^{res}) = \max_{1\leq i\leq n} |\lambda_i|, \tag{8}$$

where λ_i is the ith eigenvalue of $W_{res}^{res} \in \mathbf{R}^{n\times n}$. If ρ is smaller than unitary then that is a good sign that the reservoir has the ESP, but is not conclusive [13]. As with the information dimension though, this paper will not delve too deeply into the math of the spectral radius in showing the ESP.

4 Physical Reservoirs

In a sense, our world is the strongest computer we know of. It can simulate physics better than any processor and it can render visuals better than any graphics card. That is what makes RC's ability to harness the complexity of natural systems for controlled computation so exciting. Anything from a bucket of water to arrays of memresistors [7] has the potential to work as a reservoir if it can take on the properties introduced in Section 3. In this section we will investigate two examples of physical systems from diverse branches of physics and see how each can be used to implement the RC framework.

4.1 Coulombe's Mechanical Reservoir

In 2017, Coulombe et al. introduced a "massively-parallel network of simple units with elementary non-linear processing capabilities" taking the form of masses coupled by springs and grounded by nonlinear springs [14]. While they do not actually implement this RC, they provide both computational models and suggestions for how such reservoirs might be produced through conventional micro-electromechanical system (MEMS) fabrication techniques. Because the model is left so open ended, any or all of the components in Eqs. 4 & 5 could be applied based on tweaks to the final implementation, so we will consider the full equations in this section.

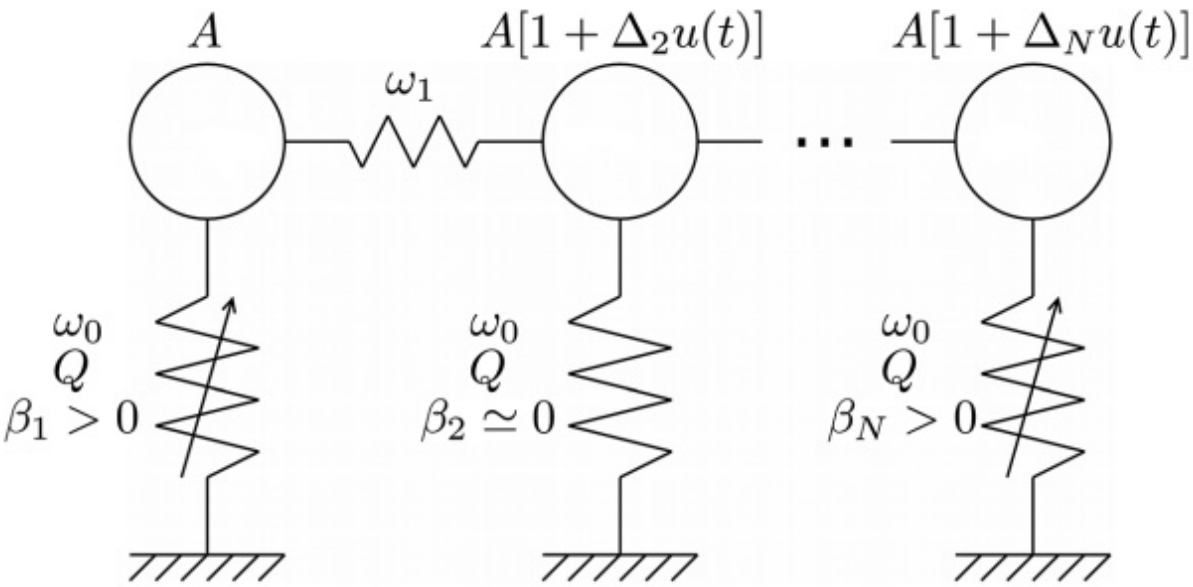

Figure 8: The mechanical reservoir takes the form of many linearly coupled damped, grounded Duffing oscillators. Each Duffing oscillator has the same resonant frequency ω_0 and quality factor Q but is assigned a different anharmonic term β_i. The linear coupling is achieved by springs of resonant frequency ω_1 and input driving is applied to each mass with an average amplitude A and a scaling factor of Δ_i [14].

4.1.1 Coupled Nonlinear Oscillators

The proposed system is modelled by the second-order differential equation,

$$\ddot{x}_i(t) = \underbrace{-\frac{\omega_0}{Q}\dot{x}_i(t) - \omega_0^2 x_i(t) - \beta_i x_i^3(t)}_{Anharmonic\ Oscillator} + \underbrace{A[1+\Delta_i u(t)]cos(\Omega t)}_{Input\ Driving} + \underbrace{\omega_1^2[x_{i-1}(t) - 2x_i(t) + x_{i+1}(t)]}_{Coupling}, \quad (9)$$

where $x_i(t)$ gives the position of the i^{th} mass at time t. A schematic is shown of the same system in Fig. 8. Equation 9 can be conveniently viewed as the sum of three parts, a term describing the anharmonic grounding oscillator, a term for the driving input signal and a term for the coupling between masses. An important thing to note is that the oscillators are each kept vertically aligned so that Eq. 9 refers to the vertical motion. This means that the coupling force is only the vertical component of whatever force the connecting springs apply. Because this is not physically implemented by Coulombe et al., the input and output mechanisms are left somewhat vague. The input force can be imagined as some force acting vertically on each mass, in conjunction with the springs, and the output can be thought of as any readout of the vertical positions of the masses. We will try to understand each of the three components that make up Eq. 9 individually to better understand the system as a whole.

The anharmonic oscillator component represents the acceleration of the masses due to the nonlinear springs grounding them, also known as a damped Duffing oscillator, and consists of three terms. The first term, $-\frac{\omega_0}{Q}\dot{x}_i(t)$, is a velocity dependent damping term with a damping ratio given by $\zeta = \frac{1}{2Qm}$ where Q is the quality factor of the spring and m is the mass. Depending on this ratio, the oscillators can then be under-damped ($\zeta < 1$), critically damped ($\zeta = 1$) or over-damped ($\zeta > 1$) which all result in qualitatively distinct behaviors but they are difficult to see in practice due to all the other terms in Eq. 9. The second term of the anharmonic oscillator component, $-\omega_0^2 x_i(t)$, provides the typical undamped, harmonic behavior of a spring with spring constant $k_0 = \omega_0^2$ and resonant frequency ω_0. Finally the third term, $-\beta_i x_i^3(t)$, introduces cubic nonlinearity which is necessary for producing the kinds of complex dynamics we want in our reservoir. The spring can be either hardening ($\beta_i > 0$) or softening ($\beta_i < 0$) meaning that there is more or less restoring force at large distances from equilibrium. In our case all the springs are either hardening or don't have the cubic nonlinearity term ($\beta \simeq 0$) [15].

The second component of Eq. 9 represents the driving force which is modulated by the input signal. The driving force is $A\cos(\Omega t)$ with mean amplitude A and angular frequency Ω. For large driving amplitudes relative to the length of the springs we can expect chaotic dynamics from the system whereas smaller driving amplitudes will yield complex but non-chaotic dynamics. The value of Ω is chosen such that at least 10 oscillations of $\cos(\Omega t)$ can occur between each input step, which brings us to the input modulation term, $1 + \Delta_i u(t)$. This term scales the driving force by the input at that time multiplied by a scaling factor Δ_i. This scaling factor is kept at 0 for half of the oscillators and set to a random value ~ 1 for all others. The authors refer to these Δ_i as the coupling strengths between the input signal, $u(t)$, and the oscillators. With weaker signals, a higher coupling strength is needed ($\Delta_i > 1$) and vice versa for stronger signals. While

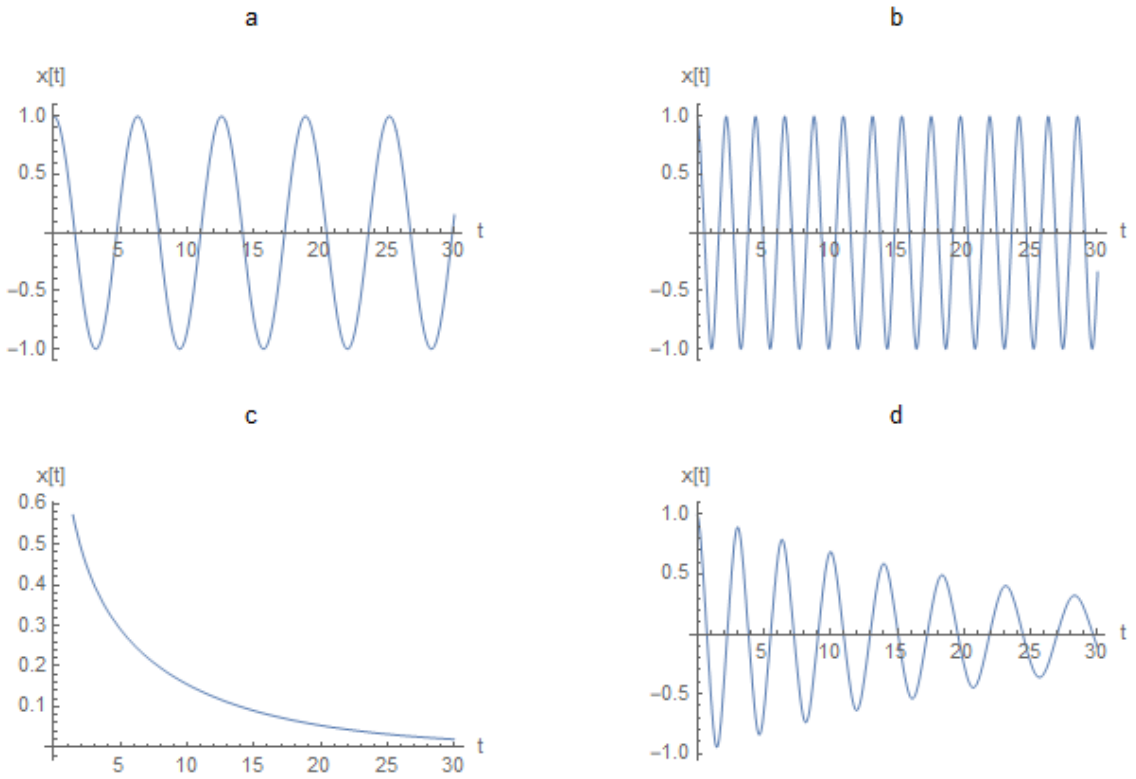

Figure 9: Plots of the vertical motion for different types of oscillators derived from Eq. 9. Plot (a) shows the motion of a harmonic oscillator given by the equation $\ddot{x}(t) = -\omega_0^2 x(t)$. Plot (b) shows a hardening anharmonic oscillator given by $\ddot{x}(t) = -\omega_0^2 x(t) - \beta x^3(t)$ with $\beta > 0$. Plots (c) & (d) both show a damped anharmonic oscillator, also called a Duffing oscillator, given by $\ddot{x}(t) = -\frac{\omega_0}{Q}\dot{x}(t) - \omega_0^2 x(t) - \beta x^3(t)$, where in plot (c) the oscillator is over-damped and in plot (d) it is under-damped.

it seems that keeping half the Δ_i values at zero is losing half the information of the input signal, it is actually necessary to ensure that later inputs do not override the signals propagating within the reservoir from previous inputs. So a balance is struck between the input and reservoir signals with the assumption that there is enough redundancy in both to function properly.

The third and final component in Eq. 9 represents the acceleration due to the linear springs coupling each mass to its nearest neighbors. Each coupling spring is identical with spring constant $k_1 = \omega_1^2$ and resonant frequency ω_1. It is easier to see why this component takes the form it does when expanded as $\omega_1[(x_{i+1}(t) - x_i(t)) - (x_i(t) - x_{i-1}(t))]$ where now it becomes clear that this is just the force of the right neighbor pulling from the right and the left neighbor pulling from the left. But because these oscillators are fixed moving in vertical paths, only the vertical offset between oscillators contributes to this linear coupling force.

All these parameters can be fine tuned to a particular implementation or benchmark problem but values

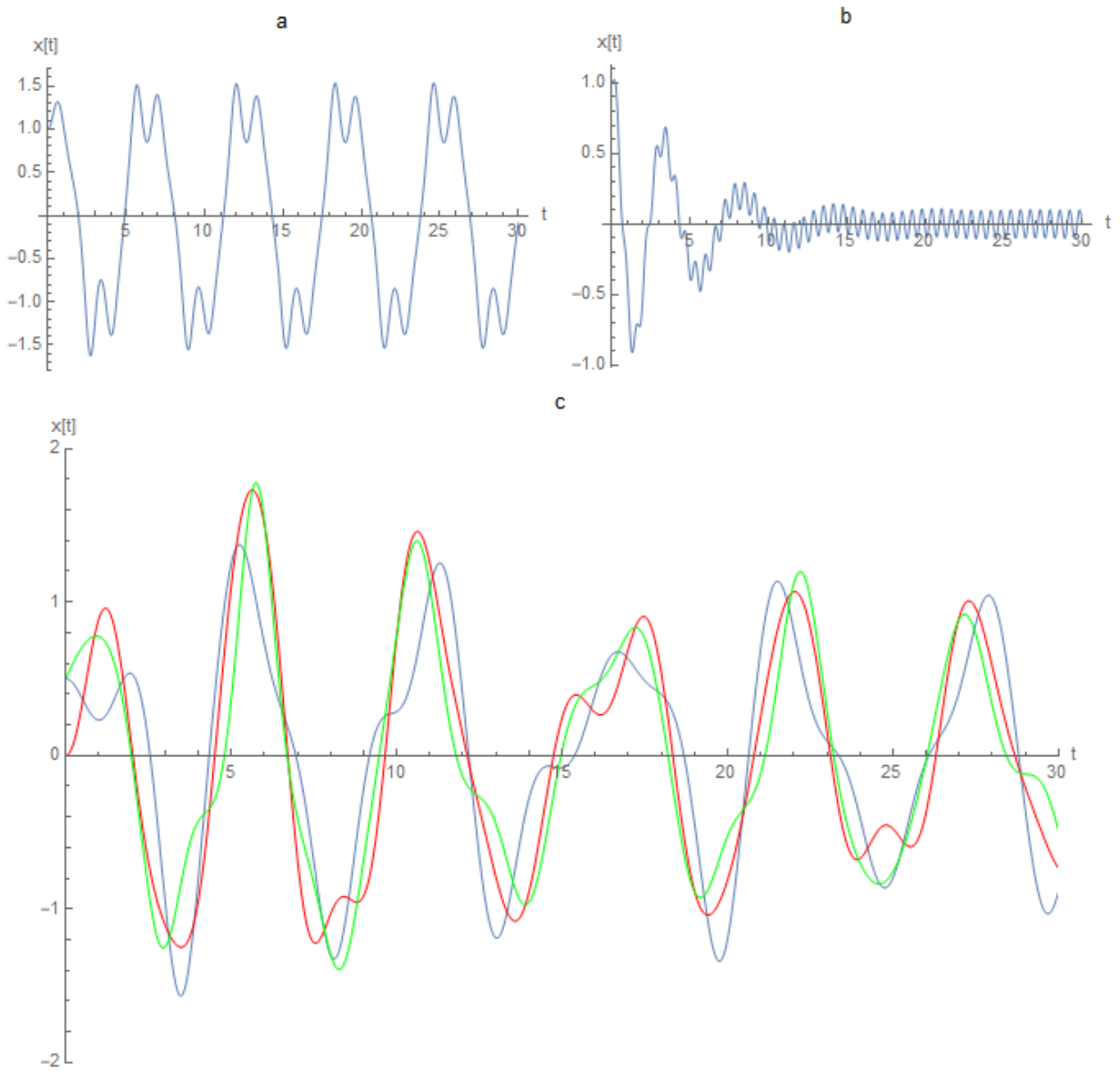

Figure 10: This is a continuation of Fig. 9 with plots of the vertical motion for different types of oscillators derived from Eq. 9. Plot (a) shows a Duffing oscillator driven by some force with a frequency close to ω_0, the resonant frequency of the Duffing oscillator. This is given by the first two components — Anharmonic Oscillator and Input Driving — of Eq. 9 Plot (b) shows the same Duffing oscillator but this time with a driving frequency much higher than ω_0, resulting in the driving amplitude being all that is left after enough time. Plot (c) shows the motions of three coupled Duffing oscillators given by the full Eq. 9. Here we can see what we will soon show is chaotic behaviour.

given here are a good starting point for a generalized model. More importantly though, we have to verify that this system has the four properties from Section 3 that will make it effective as a RC.

4.1.2 Coupled Oscillators as a Reservoir

Nonlinearity is the first property we have to show this mechanical system has in order for it to be a good reservoir. Using the methods introduced in Section 3.1, we can make a qualitative assessment of whether or not the system is chaotic. First, we look at the time delay embedding plot for a promising parameter set in Fig. 4. If this is a chaotic attractor, then the system's trajectory around it should be a non-periodic orbit, meaning it never returns to the same point. To determine this with certainty, we would have to visualize the trajectory through infinite time and with infinite granularity to ensure it never returns to the same point but obviously this is infeasible. For further evidence that the system is chaotic, we can look at the bifurcation diagram of the system, seen in Fig. 5. Here we see that there is a chaotic region on the left, where the dynamics of the system vary greatly from one parameter set to the next, while on the right it stabilizes into a periodic region as the resonant frequency diverges from the driving frequency. This means that as long as we use the system with parameters that put it within the chaotic region, it should be suitable as a reservoir.

High dimensionality is something that was assumed but not shown in Ref. [14] but we can lean on parallels with other works to get a sense for whether or not this assumption is reasonable. In particular we will look at Ref. [16] where Kiss et al. evaluate the information dimension of a nonlinear oscillator taking the form of nickel electrodes undergoing electrodissolution in sulfuric acid. What they found is that these oscillators in isolation produced dynamics with information dimensions of around three[12] which is on the order of what we should expect from a single Duffing oscillator. However, as they coupled more and more of these electrodes by means of external resistors they observed a fairly steady growth in the information dimension of the resultant system, as seen in Fig. 11. Given the mathematical similarities between the Duffing oscillator and the Ni electrode in solution, it seems like a reasonable assumption that many coupled Duffing oscillators will in fact yield a system with high information dimension and, importantly, that the information dimension of the reservoir can be scaled as necessary just by chaining more oscillators onto it.

The ESP is only briefly addressed by Coulombe et al. but the argument that this reservoir has the ESP is actually fairly straightforward. Because the system has a damping component on every node, if left to evolve from any initial state without an input driving force, it will eventually come to rest. This means that initial conditions of the system eventually disappear and all that can be left, if there is input, is the dynamics induced by the input signals. This is exactly the definition of the ESP so, while not rigorous, this provides a nice aesthetic explanation. However, there is one pitfall in particular which this logic could fall into that needs to be accounted for in the calibration of the reservoir's parameters that has to do with the

[12] Kiss et al. make use of Eq. 6 to calculate the information dimension.

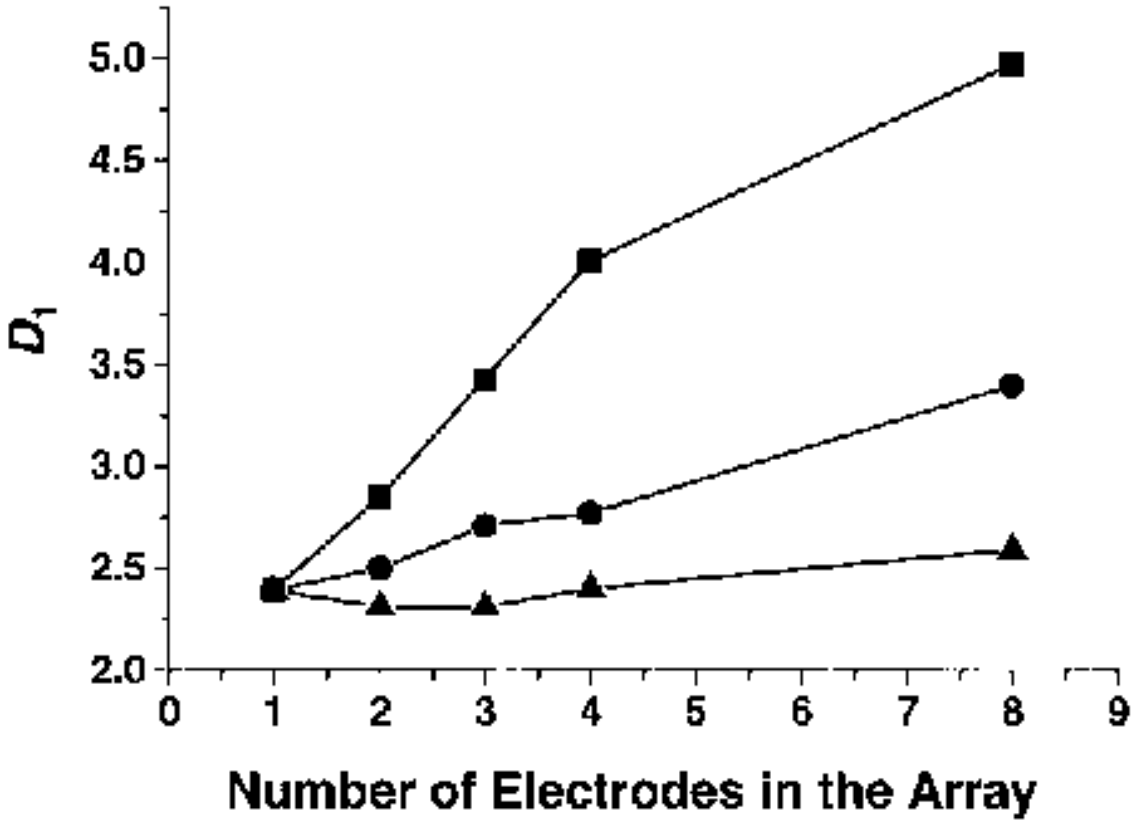

Figure 11: A plot of the information dimension, D_1, calculated with Eq. 6, of systems of n coupled nickel electrodes. The squares, circles, and triangles each represent a different value for a system parameter, ϵ. For the right choice of parameter set, such as with the squares, the information dimension increases roughly linearly with the number of nodes coupled, n [16].

system's chaotic attractors. Supposing a certain reservoir has dynamics that revolve around two attractors, then it is entirely possible that the input driving would not be enough to bridge the gap between these two attractors and would remain entirely in the basin of one attractor. In this case, the reservoir could produce entirely different dynamics for the same input stream depending on which attractor's basin it starts in. This breaks the starting state independence that the ESP grants a system which is so important for consistent computation. So in designing the reservoir, we need to be careful to test that there is only one attractor in the parameter space we are working in.

4.1.3 Proposed Implementation

While this reservoir could be implemented on any scale, it has the most potential for applications if it can be shrunk to microscopic size. This is proposed to be possible using MEMS fabrication techniques to create

a network of thin, doubly clamped silicon beams operated in their out-of-plane mode. In such a device, the silicon beams act as both the masses and nonlinear springs (grounded through the clamps on either end) and the coupling can be achieved by any sort of connection between the beams that is perpendicular to the out-of-plane oscillations. By performing some estimates of the minimal parameters required to achieve nonlinear oscillators and chaotic dynamics from the system as a whole, they found that this device could be one to two orders of magnitude smaller and more power efficient than other state of the art neuromorphic devices of similar computational abilities.

Another reason that this implementation is exciting is its mechanical nature because it can be set up to accept input in the form of physical stimuli, thus skipping the electronic computer middleman. This creates possibilities for these systems to be employed as low power sensors with complex computing capabilities fabricated into them mechanically. However, there is still a lot of further research and development to be done to verify that the estimations on size and power consumption are on the right order [14].

4.2 Chen's Quantum Reservoir

The second implementation of a physical RC is the quantum circuit reservoir developed by Chen et al. in Ref. [12] on IBM's cloud quantum computers. This RC scheme leverages the high dimensionality and builtin nonlinearity of transmon quantum bits (qubits) to achieve the necessary features for a good reservoir. This application promises to harness the powers of two rapidly growing and evolving models of computation: quantum computing (QC) and RC. Before digging into the dynamics of this quantum reservoir though, we will first explore some necessary background topics to build up an understanding of gate based quantum logic. This implementation uses only reduced versions of Eqs. 4 & 5 given by,

$$x(l+1) = f\ [\ W_{res}^{res} x(l) + W_{inp}^{res} u(l) + W_{bias}^{res}\ ], \tag{10}$$

where $x(l)$ gives the reservoir's internal state at time step l and $u(l)$ gives the input at that time step. The output is then given by,

$$\hat{y}(l+1) = f\ [\ W_{res}^{out} x(l+1) + W_{bias}^{out}\ ]. \tag{11}$$

4.2.1 Transmon Qubits

The bit is the fundamental unit of all information and computing theory and is capable of taking on the values of 0 or 1.[13] The parallel to the bit for the fields of quantum information theory and QC is the quantum bit, or qubit, which instead takes on some superposition of 0 and 1. We will dig deeper into what this means and its implications in Section 4.2.2 but here we will first try to understand how exactly we can find and control this sort of behavior in nature. Note that this section is not necessary for following the logic of the following sections, similarly to how understanding the architecture of a classical computer is not (absolutely) necessary to learn about programming them.

In modern computers bits can be implemented in a variety of ways — magnetic switches, two-level capacitors, the presence or absence of a conducting path — because there are so many ways to physically represent two discrete states. Similarly in QC, there are many ways to create qubits because so many quantum phenomena are, at their cores, two state quantum systems. Photons, trapped ions, and superconducting niobium rings are all viable qubits but the architecture leading the QC industry currently is the transmon qubit. A transmon (transmission-line shunted plasma oscillation) is a supercooled (and thus superconducting) LC circuit with nonlinear inductance, called a Josephson junction, that is relatively easy to control in its lowest two energy states by means of microwave resonators surrounding the circuit [17]. To get a complete picture of the behavior of a transmon, we can write its Hamiltonian as a sum of the energies of both the capacitor and the Josephson junction inductor,

$$\hat{H}_{tr} = \frac{\hat{Q}^2}{2C} - \frac{\Phi_0 I_0}{2\pi} cos(\frac{2\pi\hat{\Phi}}{\Phi_0}), \tag{12}$$

where $\hat{Q}$ is the charge on the capacitor of capacitance C, $\hat{\Phi}$ is the flux through the inductor of critical current I_0 and $\Phi_0 = h/2e$ with e being the charge of one electron. The hat symbol over variables denotes that this is the quantized or discrete Hamiltonian (and should not be confused with the hat over $\hat{y}$ in Eq. 5 which is the computed output). Equation 12 can be written in a more condensed form by introducing reduced charge $\hat{n} = \hat{Q}/2e$ and phase $\hat{\phi} = 2\pi\hat{\Phi}/\Phi_0$,[14]

$$\hat{H}_{tr} = 4E_c\hat{n}^2 - E_j cos(\hat{\phi}), \tag{13}$$

where $E_c = e^2/2C$ and $E_j = I_0\Phi_0/2\pi$ are the corresponding reduced energies. Thus far we have been

[13] "Bit" literally means "binary digit" and was coined in 1946 by John Tukey.

[14] The charge quanta here is $2e$ because in superconducting circuits we deal with Cooper pairs of electrons. The associated flux quanta is then given by Φ_0.

assuming that this system is operating in the quantum regime because of the hats over operators but nothing else separates this Hamiltonian from the classical equivalent. To get a more useful quantum Hamiltonian, we have to write Eq. 13 in terms of creation and annihilation operators $\hat{c}$ and $\hat{c}^\dagger$ which correspond to adding or removing one Cooper pair to or from the transmon. We can express the reduced charge and phase variables in terms of these operators,

$$\begin{aligned} \hat{n} &= i n_{zpf}(\hat{c} + \hat{c}^\dagger) \qquad (1) \\ \hat{\phi} &= \phi_{zpf}(\hat{c} - \hat{c}^\dagger), \qquad (2) \end{aligned} \tag{14}$$

where $n_{zpf} = (E_j/32E_c)^{1/4}$ and $\phi_{zpf} = (2E_c/E_j)^{1/4}$ are the zero point fluctuations of charge and phase. Using Eqs. 14.1 & 14.2 and a Taylor expansion of $cos(\hat{\phi})$ we can transform Eq. 13 into Eq. 15.1,

$$\begin{aligned} \hat{H}_{tr} &= -4E_c n_{zpf}^2(\hat{c} + \hat{c}^\dagger)^2 - E_j\left(1 - \frac{1}{2}\phi_{zpf}^2(\hat{c} - \hat{c}^\dagger)^2 + \frac{1}{24}\phi_{zpf}^4(\hat{c} - \hat{c}^\dagger)^4 + \dots\right) \qquad (1) \\ &\approx \sqrt{8E_cE_j}\left(\hat{c}^\dagger\hat{c} + \frac{1}{2}\right) - E_j - \frac{E_c}{12}(\hat{c}^\dagger - \hat{c})^4, \qquad (2) \end{aligned} \tag{15}$$

where in Eq. 15.2 the terms beyond fourth order have been dropped. This is almost where it needs to be but there is still a $(\hat{c}^\dagger - \hat{c})^4$ that contains terms which do not have a noticeable impact on the dynamics of the transmon (terms where there is an uneven number of $\hat{c}$ and $\hat{c}^\dagger$) and the constant E_j which should not affect dynamics either. Making these simplifications and substituting $w_0 = \sqrt{8E_cE_j}$ and $\delta = -E_c$ we get to the form we want,

$$\hat{H}_{tr} = \left(\omega_0 + \frac{\delta}{2}\right)\hat{c}^\dagger\hat{c} + \frac{\delta}{2}(\hat{c}^\dagger\hat{c})^2. \tag{16}$$

Although it is not at all obvious, especially without familiarity with quantized Hamiltonians, the transmon qubit's Hamiltonian is actually that of a quantum Duffing oscillator, the quantum analog to the individual (uncoupled) oscillators from Section 4.1.[15] This parallelism between the two reservoirs will allow us to share arguments for the reservoirs being well suited for RC and provide reassuring evidence that RC is a broadly applicable paradigm.

[15]One way to try to gain an intuition for why this Hamiltonian is representative of a Duffing oscillator is to start by looking at the quantum harmonic oscillator (QHO). This is given by $\hat{H}_{QHO} = \hbar\omega_0(\hat{c}^\dagger\hat{c} + \frac{1}{2})$, which is very similar in form to the first half of Eq. 16. So it then follows that by adding higher order terms to the QHO we can get the quantum Duffing oscillator just as with the classical variants.

4.2.2 Gate-Based Quantum Computation

The transmon qubit which we delved into in Section 4.2.1 is an effective qubit because it can be precisely controlled as a two-state system. Looking at Fig. 12 we see that the energy levels of a quantum harmonic oscillator (QHO) are evenly spaced whereas transmon energy levels are defined by $\omega_j = (\omega - \delta/2)j + \delta/2j^2$ with $\omega = \omega_0 + \delta$ [17]. If a QHO is used as a qubit, it can be difficult to maintain a strictly two-state system because the energy input or output to jump from any state to its neighbors is identical. With a transmon qubit though, only one input or output energy is allowed, $\omega_2 - \omega_1$, which is unique from all other transitions and so it can be kept in only the first two energy states.

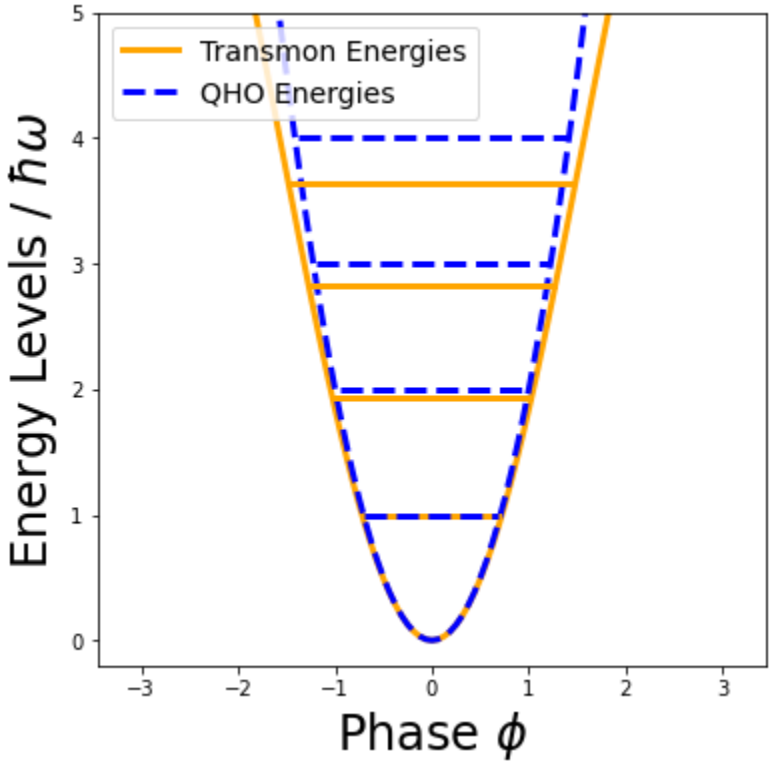

Figure 12: A graph of the potential well of the quantum harmonic oscillator (in dotted blue) and transmon (in yellow) with their quantized energy levels marked. The important feature to note here is that there is an even spacing between QHO energy levels, making it harder to control the state of the system, whereas the energy gaps of the transmon well are non-constant [17].

But because this is a qubit and not just a bit, it is not as simple as just being in the first or second energy level. Instead the qubit occupies some superposition of the first energy level $|0\rangle$ and the second $|1\rangle$, usually written as $|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$ with $a, b \in \mathbf{C}$ and $|a|^2 + |b|^2 = 1$. Another useful interpretation of qubits is a geometric mapping onto the Bloch sphere. The Bloch sphere has three axes $|\pm X\rangle$, $|\pm Y\rangle$, and $|\pm Z\rangle$

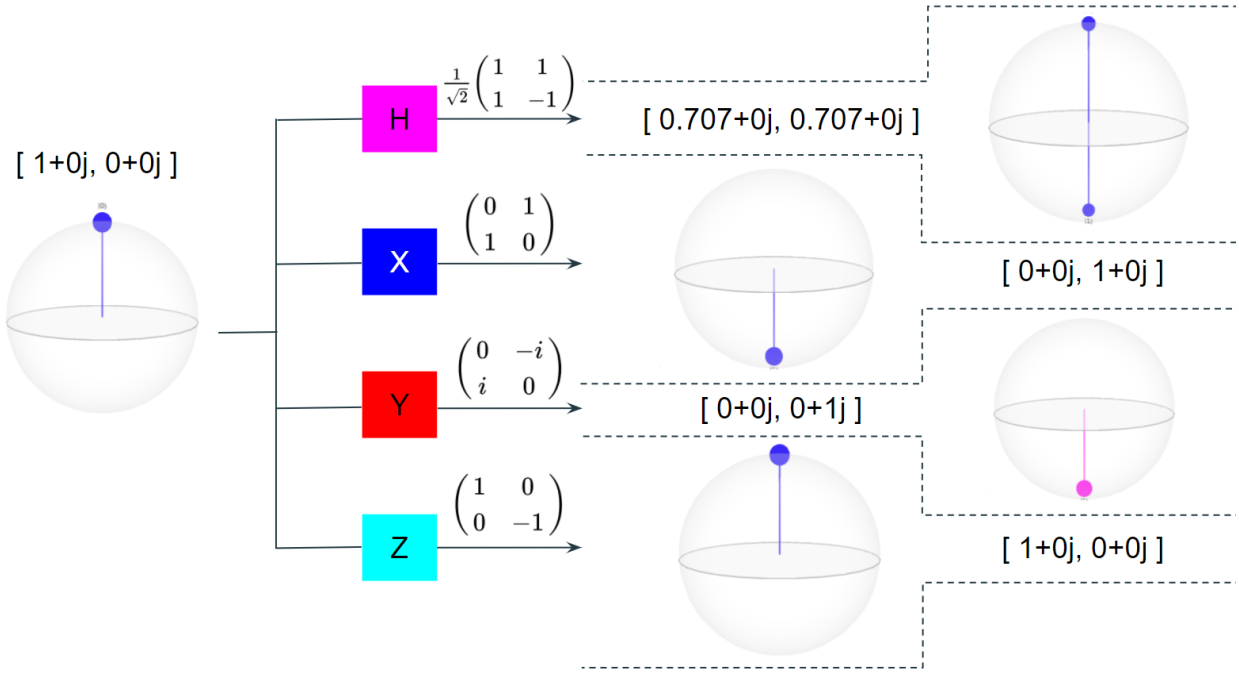

Figure 13: Some of the most common single-qubit quantum logic gates — Hadamard, X, Y, and Z — and their effects on a $|0\rangle$-state qubit. The Hadamard gate acts to flip the qubit between bases while each of the three X, Y, and Z gates act as logical nots in their respective bases. Both Bloch sphere and statevector respresentations are given for the transformed qubit and the matrix form of each gate is shown.

representing the three bases of a two-state quantum system, also known as the Pauli spin matrices. Under this interpretation, any operations on single qubits can be thought of as rotations about the Bloch sphere. This inspired one of the most prevalent models of QC, wherein quantum logic gates are used to evolve qubits that pass through them.

All single qubit gates can be thought of as rotations around the Bloch sphere and eight of the most common single qubit gates are shown in Figs. 13 & 14. $R_x(\theta)$, $R_y(\theta)$ and $R_z(\theta)$ are three commonly used gates to rotate around one of the three axes on the Bloch sphere by an angle θ and in fact, they form a universal set for single qubit operations.[16] Another set of gates worth introducing is the collection of X, Y and Z operators which flip a qubit in the basis of the gate. So if q_0 is a qubit in state $|1\rangle$, then $X(q_0)$, where the gate is written as acting on the qubit as a matrix, will flip q_0 to be in state $|0\rangle$.[17] However, if instead q_0 is sent through a Z gate, it will end up in the $-|1\rangle$ state and if it is sent through a Y gate, it will end up in the $-i\,|0\rangle$ state. The takeaway here is that a Pauli gate will act as a logical not on qubits in its own basis

[16] A universal gate set is one in which the component gates can be used to create any other gate. The rotation gates are only universal over the set of single qubit gates because they can't make some multi-qubit gates.

[17] The states of a quantum system are often labelled $|1\rangle$ and $|0\rangle$ in the X basis, $|+\rangle$ and $|-\rangle$ in the Z basis, and $|\circlearrowleft\rangle$ and $|\circlearrowright\rangle$ in the Y basis. This strange notation for the Y basis comes from describing photons' polarization which can be up-down, left-right, or spinning left or right.

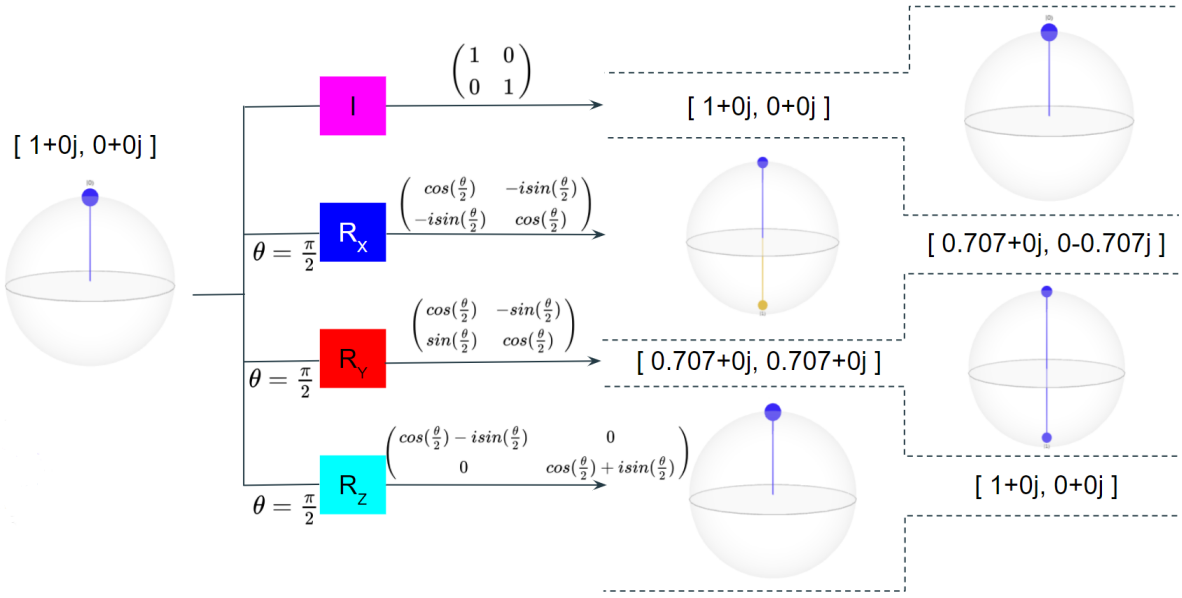

Figure 14: A continuation of Fig. 13 that introduces the identity and three rotation gates. The identity has no effect on qubits passing through it and each rotation gate, R_X, R_Y, R_Z, rotates the qubit about its respective basis. The three rotation gates are shown with parameter $\theta = \frac{\pi}{2}$.

and will have more complicated behavior otherwise.

Although it does not show up in the circuit we will be using as a reservoir, a proper introduction to QC requires at least seeing the Hadamard gate as well. This gate flips a qubit between the X and Z bases and is a component in almost all universal sets of quantum gates (yes, there are many). There is nothing innately special about it, as it equates to a rotation about the Y, but it comes up often enough in QC that it is worth mentioning.

Things get trickier when we start introducing multi-qubit gates because now qubits can have representations within the Bloch sphere (they are no longer restricted to the surface) as mixed states. Luckily, for the purposes of this RC we only need to introduce one multi-qubit gate: the CX gate. The controlled not (in the X basis) gate is very similar to the X gate, except that CX has a control qubit. This means that if the control qubit is a $|1\rangle$ then the gate is 'on' and will flip the target qubit, while if the control qubit is a $|0\rangle$ then the gate is 'off' and will have no effect. More complicated behavior arises when the control qubit is neither and can lead to phenomena such as phase kickback by which the control qubit's state can end up being changed even though it is just controlling whether or not something else gets changed. The important thing is that with this multi-qubit gate, we now have the ability to couple qubits, commonly referred to as entanglement, and achieve similar dynamics as what we saw in the mechanical reservoir from Section 4.1.

4.2.3 Quantum Circuit as a Reservoir

The reservoir developed by Chen et al. is a quantum circuit run on a 5-qubit processor [12]. Therefore the state of the reservoir, given by $x(l)$ in Eq. 10, actually takes the form of a density operator ρ_l which is a tensor product[18] over N subsystems,

$$\rho_l = \bigotimes_{k=1}^{N} T^{(k)}(u_l)\rho_{l-1}^{(k)}, \tag{17}$$

where the superscripted k in parentheses indexes subsystems and T is an operator evolving ρ_{l-1} according to the input $u(l)$. In the actual implementation of this circuitry, QCs are still small enough that it doesn't make sense to include multiple subsystems so from here on out we can drop the k index. If we look at the form of the T operator, we find that it matches the form of Eq. 10,

$$T(u_l)\rho_{l-1} = (1-\epsilon)[\ u_l U_0 + (1-u_l)U_1\ ]\rho_{l-1} + \epsilon\sigma, \tag{18}$$

for $0 \leq u_l, \epsilon \leq 1$ and where σ is an arbitrary fixed density operator representative of the input bias. Here, u_l is some randomly generated weight assigned to U_0 and U_1 and ϵ is another such weight assigned to that sum of U_0 and U_1 and the bias. U_0 and U_1 are fixed matrices that we define below in Eq. 19. Remembering that the mechanical oscillator from Section 4.1 is a chain of linearly coupled, anharmonic Duffing oscillators and that the transmon qubit introduced in Section 4.2.1 exhibits the same dynamics as the Duffing oscillator, it follows that to create an effective quantum reservoir we will want a quantum circuit that linearly couples qubits together, so we rely on U_0 and U_1 to do that. There are many ways to create linear couplings between qubits but the variant chosen by Chen et al. is described by,

$$\begin{aligned} U_0(\theta) &= \prod_{j=1}^{N_0}[\ R_Y^{(j_t)}(\theta_{j_t})CX_{j_c j_t}R_Y^{(j_t)}(\theta_{j_t})^\dagger\ ] \qquad (1) \\ U_1(\phi) &= \bigotimes_{i=1}^{n} R_X^{(i)}(\phi_i), \qquad (2) \end{aligned} \tag{19}$$

where N_0 is the number of iterations of the subcircuit in Eq. 19.1, n is the number of qubits in the subsystem and the i, j subscripts index qubits. Some are then further separated into control and target qubits by c, t subscripts. The parameter N_0 is chosen to be close to n which in this case is on the order of five. A schematic

[18] A tensor product is not an easy concept to explain technically but abstractly it equates to just putting two qubits next to each other such that they can interact in the same space. For the purposes of this circuit, a number of k-qubit circuits are being "put next to each other" in the sense that they can be run in parallel but they don't ever interact.

is given in Fig. 15 that also includes a product over N_1 subcircuits. This is not included in Eq. 19.2 because it is not necessary for the smallest possible implementations of this RC scheme.

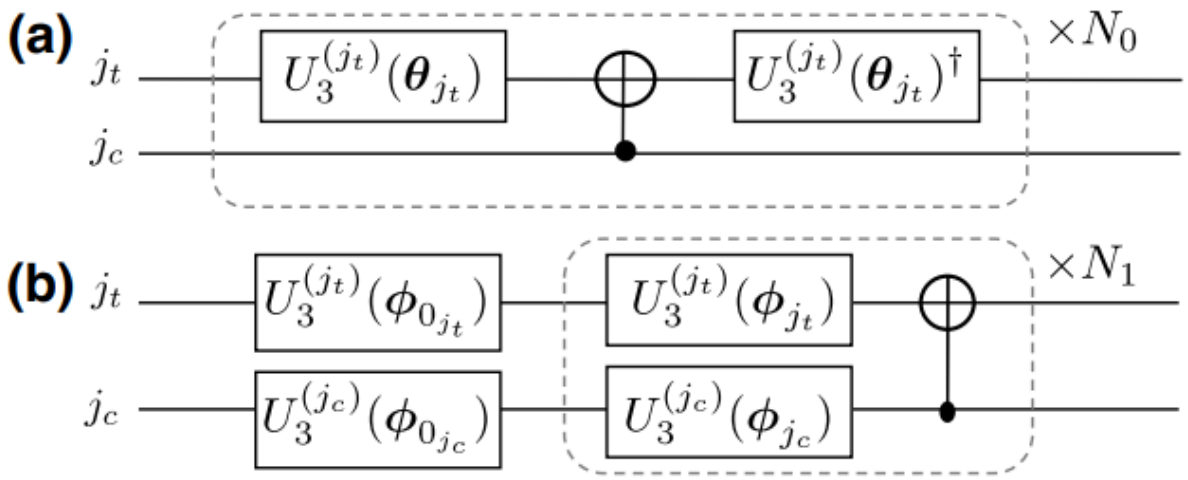

Figure 15: Quantum circuit schematics of U_0 (a) and U_1 (b) from Eq. 18. Quantum circuits are commonly depicted this way with qubits being stacked vertically and then travelling along the horizontal lines and being evolved by gates they hit. The U_3 gates are rotations about a given axis by their angle parameter and the dagger (†) indicates a conjugate transpose. The cross and dot gate connecting two qubits is a controlled not with target j_t and control j_c and the sets of gates contained in the dotted line areas are repeated in series N_0 and N_1 times respectively [12].

The full reservoir is then given by the quantum circuit shown in simplified form in Fig. 16. In this circuit, each $T(u_l)$ gate is the T from Eq. 18 with the proper input being fed into it. After all the inputs have been fed in, each qubit is measured to get the reservoir's state[19] as output. The most intuitive way to see the linear coupling that this reservoir creates among its qubits is looking at the U_0 circuit. If the two rotation gates here are given 180° as their arguments, then this circuit will create a maximally entangled Bell pair — a very common circuit in beginning QC. Thus the circuit with randomized angles should achieve some amount of entanglement between the qubits being acted on.

So now that we've established the governing equations for this quantum reservoir, we need to check that it has all the properties necessary to perform well. But because of the similarities it holds with that of Coulombe et al., we can borrow a lot of the arguments from Section 4.1.2 in some form or another. The arguments that this reservoir will exhibit high dimensional nonlinear behavior is exactly the same because the nonlinearity results from the same dynamics of anharmonic Duffing oscillators. The coupling, which lends the system its high information/embedding dimension, is linear in both cases so will be equivalent up

[19] Actually this process has to be repeated a number of times, called the number of "shots", because quantum computers give probabilistic results in most cases.

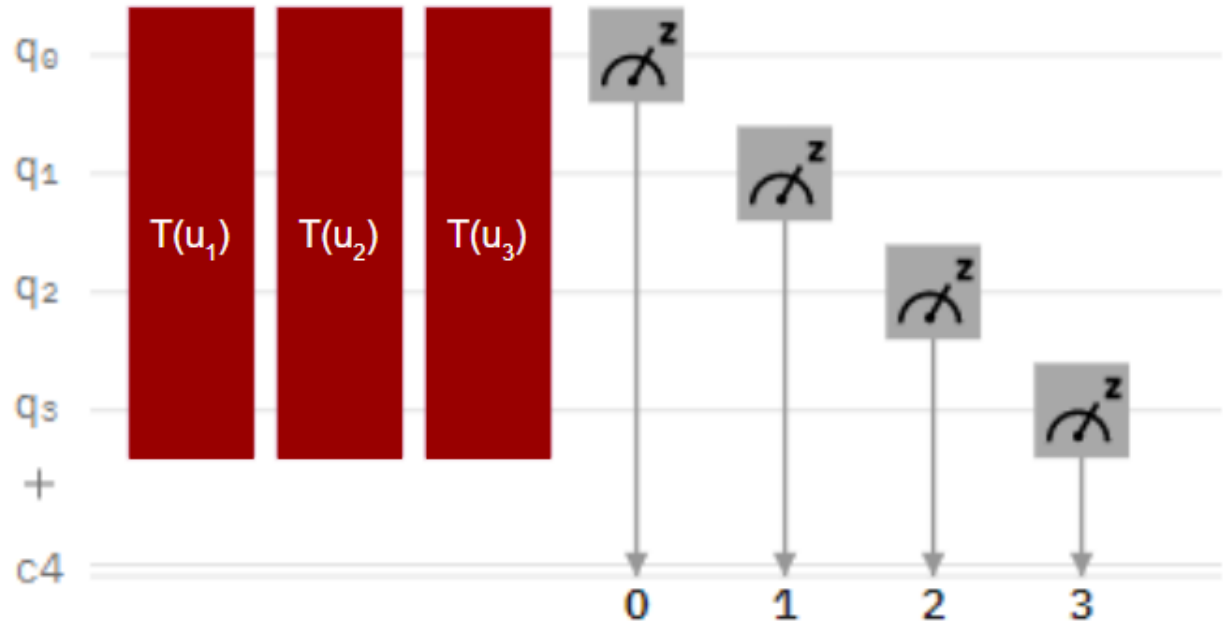

Figure 16: A quantum circuit representing the full reservoir presented by Chen et al. Each T gate is given by Eq. 18 and takes the input u_l at the l^{th} time step. In practice, there would be many more T gates. The qubits are then measured over many runs to get a probabilistic measure of the state of the reservoir at the end of the inputs.

to a scalar. The argument for the ESP would require only a few tweaks to fit the quantum reservoir but there is actually an even easier one in this case. Because it is built into the hardware of the qubits used in this implementation to initialize in the $|0\rangle$ state, every run of the circuit will start in the state,

$$|\Psi\rangle = \bigotimes_{i}^{n\times N} |0\rangle \,, \tag{20}$$

where n is the number of qubits in each of the N subsystems and so the ESP is implicit in the consistent initial conditions. If we really press this argument, it turns out to have a few holes but to provide a really thorough information theoretic proof of convergence would take significantly more set up than we can do here (see Section A.1 & A.2 of Ref. [12] for the details).

5 Conclusion

RC is an exciting new model for computation because of it enables us to harness the natural complexities of physical systems to reduce the computational complexity of problems. We have seen that such systems

must exhibit certain properties, namely nonlinearity, high dimensionality, and the ESP, but in the realm of dynamical systems this encompasses a huge subset. Possibilities for physical implementations stretch over many fields including classical and quantum mechanics, chemistry, biology, fluid dynamics, electronics, neurology and more. And they also have a wide range of possible applications that extend beyond the scope of this paper including long and short term chaotic systems prediction [18], fabricating low-power yet computationally powerful electro-mechanical sensors [14], and better modeling for parts of the brain [7]. With such a broad range of possibilities the coming decades will likely see numerous theoretical, physical, and industrial applications of RC.

A Appeltant's Electronic Reservoir

In 2010, Lennert Appeltant introduced one of the first implementations of a "physical" RC using an electronic version of the Mackey-Glass (MG) system as a reservoir. The MG circuit is a single node delayed dynamical system with variable dimensionality and as we will see this makes it a perfect candidate for RC [19][20].

This implementation uses reduced versions of Eqs. 4 & 5 without input bias or either output recurrent or loop connections. These reduced versions are given in Eqs. 21 & 22[20],

$$x(k+1) = f\ [\ W_{res}^{res} x(k) + W_{inp}^{res} u(k)\ ], \tag{21}$$

where W_{res}^{res} maps the current state vector onto the reservoir and W_{inp}^{res} maps the input onto the reservoir. The W_{out}^{res} and W_{bias}^{res} terms have been dropped because Appeltant's model includes neither recurrent connections from the output layer nor bias at the input layer. The reduced output equation is given by:

$$\hat{y}(k+1) = W_{res}^{out} x(k+1) + W_{in}^{out} u(k) + W_{bias}^{out}, \tag{22}$$

where W_{res}^{out} maps the current reservoir state onto the output, W_{in}^{out} maps the input directly onto the output and W_{bias}^{out} adds a fixed bias while the W_{out}^{out} output loop connection is left out.

A.1 Mackey-Glass System

The Mackey-Glass system is a delay differential equation (DDE) that was originally designed to model complex dynamics in physiological control systems. Mackey and Glass believed that bifurcations between

[20]These reduced equations are given as functions of k instead of t to indicate that they operate on discrete time intervals.

periodic and chaotic dynamics could explain physiological disorders, in essence arguing that if you are on one side of the bifurcation point you will have this disorder while if you are on the other side you will not. Since then the model itself, given in Eq. 23,

$$\dot{x} = \frac{\beta x(t-\tau)}{1+x(t-\tau)^n} - \gamma x(t), \tag{23}$$

where $\gamma, \beta, n, \tau > 0$ are all variable parameters, has seen a wide array of uses including approximation by transistor circuits. One such circuit is shown in Fig. 17 with two transistors and a PyBoard for programmatic control of the delayed feedback, which will be examined in detail in Section A.2 [21].

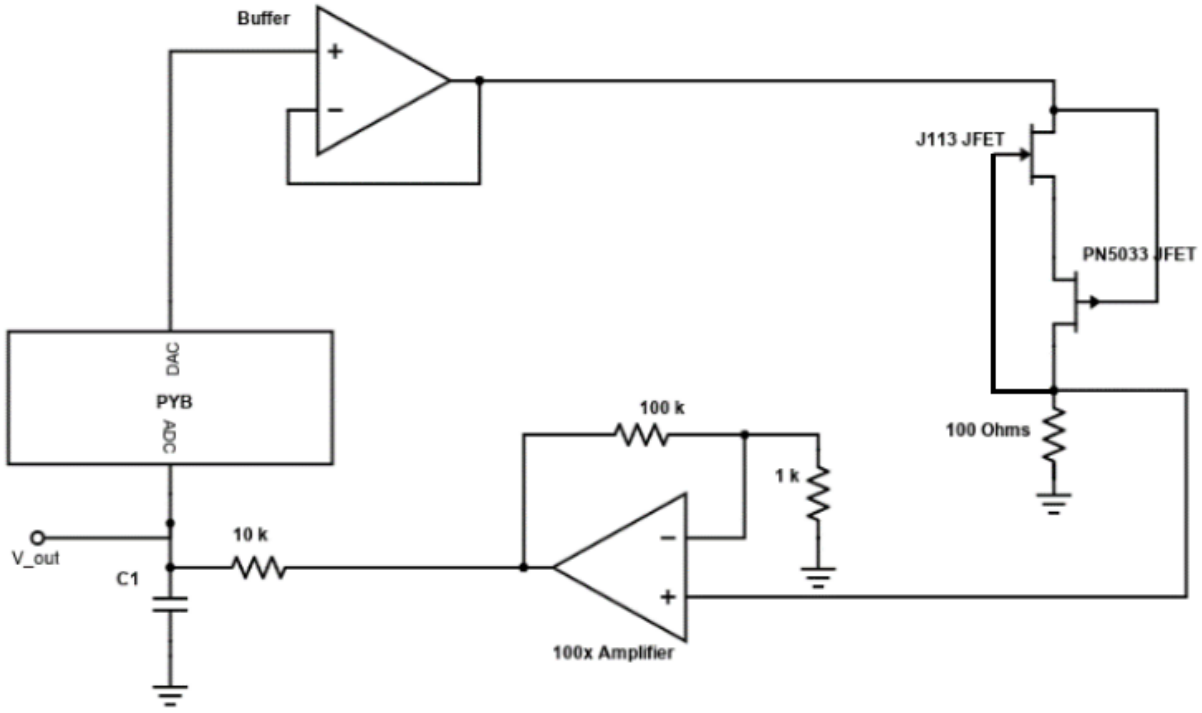

Figure 17: There are five parts to this diagram of the MG circuit (going clockwise from PYB): the PyBoard, a buffer, the two-transistor element, a 100 times amplifier, and a low pass filter. The buffer, amplifier, and low pass filter components are all very common designs in electronics. The PyBoard recombines the feedback from the circuit and a sawtooth signal on the PyBoard's internal clock. The two-transistor setup provides a nonlinear response curve. [21].

Achieving high dimensional, nonlinear behaviour with the MG system is actually not very difficult. The nonlinearity is inherent in Eq. 23, which is clearly nonlinear in $x(t-\tau)$, and the high dimensionality arises from time delay. Because this is a time delayed system, the embedding dimension[21] can be increased

[21]The embedding dimension of certain systems like this one can be good facsimiles of the information dimension while being

arbitrarily high. While this is not a perfect measure of the dimensionality of the surface that signals in the reservoir live on, in the case of the MG system the fractal dimension[22] increases fairly steadily with the embedding dimension which means that the dynamics actually expand to fill the space they are given.

Showing that the MG system has the ESP is a more difficult task and is given a thorough treatment in Section 5 of Appeltant et al. [19]. Here we will only present an aesthetic argument for both properties starting with the ESP. In order for a system to have the ESP, it must have a dependence on past inputs that falls off with time. This can be taken to mean that it must have a memory that has some maximum capacity and as the memory fills up the oldest stuff gets pushed out. So if we can define a memory function and show that it falls off with time for our system that will be sufficient to illustrate that the MG system has the ESP. The memory function of a system is typically defined to be,

$$m_i = corr\ [y(k), u(k-i)], \tag{24}$$

which calculates the cross correlation between outputs and inputs i time steps prior. As can be seen in Fig. 18 the memory function for the MG system is perfect for the first ≈ 10 time steps and then as the memory begins to overfill, its retention of old inputs drops off to zero. This only shows two parameter sets being tested but this trend does hold for all of the useful parameter space as well.

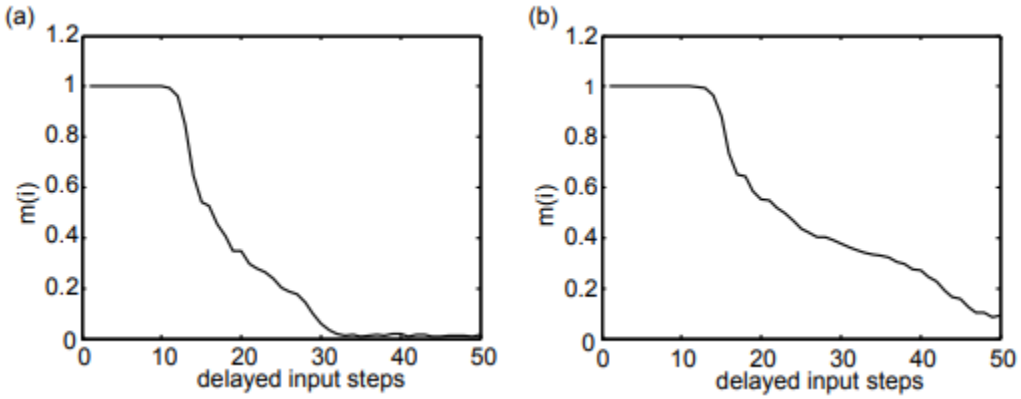

Figure 18: The memory function over increasing delays for two parameter sets of the MG circuit. The rate of falloff in memory capacity changes between the two parameter sets ((a) and (b)) but both exhibit the necessary property of falling to zero with increasing delay for the ESP [20].

easier to work with.

[22]The fractal dimension of a chaotic attractor is a measure of the space-fillingness of the surface. For example, a line (dimension 1) has a fractal dimension of 1 because it fills all the space in one dimension. A sheet of paper (dimension 2) has a fractal dimension of 2. If you were to crumple up the sheet of paper though, now it exists in a three dimensional space but there is lots of "space" that it does not fill because it was originally a two dimensional object. Thus it now has some fractal dimension between 2 and 3.

A.2 Electronic Mackey-Glass

The electronic MG circuit, diagrammed in Fig. 17, consists of five main components in a feedback loop. The PyBoard controls the feedback by combining the feedback signal, scaled by ϵ, with a sawtooth signal which the PyBoard generates, scaled by $1 - \epsilon$. Both analog-to-digital (ADC) and digital-to-analog (DAC) conversions are performed with 12-bit precision. After this recombination, the signal passes through a buffer which shields the rest of the circuitry against voltage spikes. From there it goes into the two-transistor component which provides the nonlinear response of the circuit. Each transistor is shown in Fig. 17 with three pins — drain, gate and source from top to bottom — where the gate voltage controls what proportion of the drain current comes out at the source pin. The grounded resistor following the two transistors acts as a current sink so that voltages in the two-transistor component can change more freely. From there the signal goes through a 100 times amplifier because the signal coming out of the two-transistor component is fairly weak. The scaling provided by this amplifier should be adjusted accordingly to make the unamplified value be as close to the peak-to-peak maximum that the PyBoard can handle, which is $3.6V$ for this one. Finally the signal passes through a low pass filter to remove noise and higher harmonics.

B Separation and Generalization Properties

The separation property (SP) is one that a reservoir needs to be effective in RC and, similar to the ESP, is simple to understand but very difficult to test for and define completely. In simplest terms, a reservoir with the SP should map distinct inputs into distinct output regions such that distinct regions of the output space can be clearly associated with labels. Given the governing equations for a certain reservoir, the output space will be the phase space of these equations and the stable and chaotic regions of this space will be the regions associated with labels [7][1]. Along with having this separability of the output space though, we must also have generalizability so that similar outputs are mapped into the same regions as each other. Again, the generalization property (GP) is easy to define but hard to test for. Appeltant uses an interesting method of measuring the kernel quality of the reservoir which shows his electronic reservoir to possess both the SP and GP but the details of how that technique works are beyond the scope of this paper.

In the interest of simplicity, in Section 4's case studies we assume that each exhibits sufficient separability of their phase spaces for at least some classification tasks. Quantizing the separability would be important in determining how many classes could be managed before the phase space is saturated.

References

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

The first theoretic introduction to the ideas of physical reservoir computing and the properties required of reservoirs.

[2] Schrauwen, B., Verstraeten, D., Campenhout, J.V., "An overview of reservoir computing: theory, applications and implementations", European Symposium on Artificial Neural Networks, 2007.

Gives a thorough overview and definition for RC and many important terms and concepts related to the field.

[3] Smola, Alex and Viswanathan, S.V.N., "Introduction to Machine Learning", Cambridge University Press, 17-19, 2008.

An introduction to ML concepts such as classification and regression tasks.

[4] Abiodun, O.I., Jantan, A., Omolara, A.E., et al., "State-of-the-art in artificial neural network applications: A survey", Helion, 4, 2018.

Provides useful definitions and distinctions for types of ANNs.

[5] Dettling, Marcel, "Applied Time Series Analysis," Swiss Federal Institute of Technology Zurich, 15, 2014.

An introduction to time series data and analysis. While this section was cut in the final version of the paper, the insights from it were applied throughout.

[6] Miljanovic, M. "Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction", Indian Journal of Computer Science and Engineering, 2012.

Provides comparison between FIR and RNN architectures under the umbrella of FBNN.

[7] Tanaka, G., Yamane, T., Heroux, J.B., et al., "Recent advances in physical reservoir computing: A review", 115, 100-123, 2019.

Introduces two of the three RC examples I look at in depth and the four properties of good reservoirs.

[8] Hardesty, Larry, "Explained: Linear and nonlinear systems", MIT News, Feb 26, 2010.

Provides a very clear description of nonlinearity and its distinction from linearity.

[9] Carroll, T.L., "Dimension of reservoir computers", Chaos, 30, 2020.
A very in depth look at dimensionality of RC.

[10] Horvat, Szabolcs, "Measuring the fractal dimension of a tree photo", Wolfram Community, 2017.
An awesome post in the Wolfram Community forums on a method for using Mathematica to find the fractal dimension of a picture taken of tree branches.

[11] Weisstein, Eric W. "Information Dimension." From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/InformationDimension.html
This article provides a quick overview of the information dimension from a mathematical perspective, introducing a lot of information theoretic terms beyond the scope of this paper, but worth including in short.

[12] Chen, J., Nurdin, H., Yamamoto, N., "Temporal Information Processing on Noisy Quantum Computers", Phys. Rev. Applied, 14, 2020.
The first example of a quantum circuit being used as the reservoir in an RC model. Also builds up a very technical and useful background to RC and the necessary characteristics of reservoirs.

[13] Yildiz, I., Jaeger, H., and Kiebel S., "Re-visiting the echo state property", Neural Networks, 35, 1-9, 2012.
Finds counterexamples to the often used method for determining whether a network has the echo state property and proposes new methods.

[14] Coulombe, J.C., York, M.C., Sylvestre, J., "Computing with networks of nonlinear mechanical oscillators", PLoS ONE, 12, e0178663.
Original publication on using linearly-coupled masses grounded through nonlinear springs as a reservoir.

[15] Srinivasan, Manoj, "Basic phenomenology of simple nonlinear vibration (free and forced)", 2016.
Slides providing a very thorough overview of spring-based oscillator variants and dynamics.

[16] Kiss, I.Z., Wang, W., & Hudson, J.L., "Complexity of globally coupled chaotic electrochemical oscillators", Phys. Chem. Chem. Phys., 2000, 2, 3847-3854.
A tangential paper evaluating the information dimension of a system of coupled nonlinear oscillators. This is a useful parallel for evaluating the efficacy of the Coulombe RC in terms of its signal dimension.

[17] IBM, "Qiskit: An Open-source Framework for Quantum Computing: Introduction to Transmon Physics", 2019.

A comprehensive introduction to quantum computing, quantum software (on the qiskit framework), and the physics of qubits.

[18] Rafayelyan, M., et al., "Large-Scale Optical Reservoir Computing for Spatiotemporal Chaotic Systems Prediction", ArXiV, abs/2001.09131, 2020.
Provides a very recent example of RC being applied to long and short term chaotic systems prediction tasks with great success.

[19] Appeltant, L., "Reservoir computing based on delay-dynamical systems", Vrije Universiteit Brussel and Universitat de les Illes Balears, 2012.
Thesis paper providing one of the first experimental examples of physical reservoir computing.

[20] Appeltant, L., et al., "Information processing using a single dynamical node as a complex system", Nat. Commun. 2, 468, 2011.
Another thorough treatment of the electronic Mackey-Glass reservoir.

[21] Cho, A., "Lab 4: Low Pass Filter, Feedback Code and Mackey-Glass", Carleton College, 2020.
Lab handout that includes a circuit diagram of the MG circuit used in Appeltant's implementation.