Reservoir Computing with a Chaotic Circuit

Johannes H. Jensen[1], Gunnar Tufte[1]

[1]Department of Computer Science,
Norwegian University of Science and Technology, Trondheim, Norway
johannes.jensen@ntnu.no

Abstract

Reservoir Computing has been highlighted as a promising methodology to perform computation in dynamical systems. This makes Reservoir Computing particularly interesting for exploiting physical systems directly as computing substrates, where the computation happens "for free" in the rich physical domain. In this work we consider a simple chaotic circuit as a reservoir: the Driven Chua's circuit. Its rich variety of available dynamics makes it versatile as a reservoir. At the same time, its simplicity offers insight into what physical properties can be useful for computation. We demonstrate both through simulation and in-circuit experiments, that such a simple circuit can be readily exploited for computation. Our results show excellent performance on two non-temporal tasks. The fact that such a simple nonlinear circuit can be used, suggests that a wide variety of physical systems can be viewed as potential reservoirs.

Introduction

Most dynamical systems in nature are nonlinear (Strogatz, 2015). An abundance of these systems show complex dynamic behavior, giving rise to phenomena such as self-organization, robustness, adaptivity, learning and intelligence (Mainzer, 2007).

It is argued that many natural systems perform some form of *intrinsic computation*. Neural systems and self-organizing cellular structures are examples of large networks of simple electrochemical nodes that together process large amounts of information in support of the organism (Toffoli, 2004).

Transforming such intrinsic information processing to the artificial realm has been a key topic towards reproducing lifelike systems. Dynamics in the brain and neural systems has been targeted from the early days of cybernetics (Wiener, 1961; Ashby, 1960) to today's ongoing Human Brain Project (Markram, 2012). The processes of cellular communication also includes dynamic behavior that can be exploited toward mimicking natural information processing in artifacts, e.g. self-replication of structure (Langton, 1984).

An often neglected aspect about these phenomena is that they occur in *physical* systems. Through millions of years, computation has evolved bottom-up with electrical and chemical mechanisms as the basic building blocks. Evolution has discovered ways of doing computation by exploration and exploitation of the natural processes available in the physical substrate. Can we similarly exploit the underlying physics of matter to perform computation with such desirable properties as self-organization, robustness, vast parallelism and adaptivity?

To be successful, we need to consider computation both from a physical and a dynamical systems perspective. What types of dynamical/physical systems are suitable for computation? How do we "program" such systems (what are the inputs and outputs)? How should we define computation in terms of fixed points, attractors and trajectories? What's the role of bifurcations and chaos? (Stepney, 2012)

Finding natural models of computation would pave the way for exploiting physical systems directly for information processing. Such devices would be highly efficient, since computation happens "for free" directly in the substrate as the result of intrinsic physical properties. Complex materials with a vast number of interconnected elements could be exploited for immense parallel processing at the nanoscale. *Evolution in-materio* has indeed shown that computation can be evolved in physical matter (Miller et al., 2014).

Reservoir Computing (RC) has emerged as a promising technique for exploiting a dynamical system (the "reservoir") for computation. It is difficult to know how the output of a dynamical system should be interpreted to effectively perform useful computation. RC offers a flexible solution to this problem by utilizing a *readout* layer that is trained to produce some desired function as a linear combination of reservoir states. By virtue of its complex dynamics, the reservoir provides a rich repertoire of nonlinear transformations that can be utilized by the readout.

The study of nonlinear dynamics has shown that surprisingly simple systems can exhibit complex behavior. In this work, we present one of the simplest physical reservoirs: a chaotic circuit. We show both through simulations and in-circuit experiments that the rich dynamics of the circuit can be exploited for solving computational tasks.

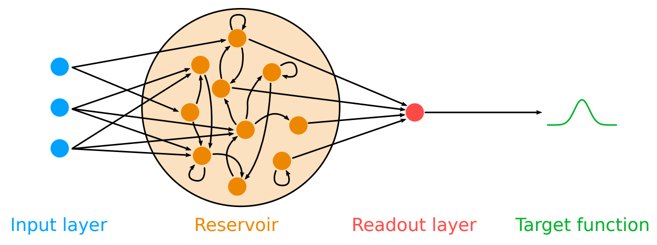

Figure 1: Reservoir Computing architecture.

Typically, reservoirs are complex systems with many state variables. However, even low-dimensional systems can be used effectively as reservoirs. Appeltant et al. (2011) show that by multiplexing output in time rather than space, a single nonlinear node can act as a virtually high dimensional system.

RC's primary success story has been processing temporal (time-dependent) signals, where the reservoir serves as a nonlinear memory. Appeltant et al. (2011) employ delayed feedback to provide the nonlinear node with memory of past states. Our reservoir is conceptually simpler in that it does not employ delayed feedback, and consequently has very limited memory.

It is often argued that a reservoir performs optimally when its dynamics lie on the "edge of chaos" (Bertschinger and Natschläger, 2004). However, a chaotic reservoir can still be used as long as the input is sufficiently large to drive its dynamics out of the chaotic regime (Ozturk and Principe, 2005).

For non-temporal tasks, a chaotic system can be more readily exploited as long as it can be reset between inputs. Goh and Crook (2007) demonstrate how the transients of the Lorenz attractor can be used for pattern recognition. All transients will eventually diverge in a chaotic attractor, effectively separating all inputs. However, transients will remain similar for an initial period of time. By selecting at which point(s) in time the transients are observed, the sensitivity of the classification can be adjusted. This selection, of course, can be done automatically by the readout.

This paper is organized as follows: we begin by discussing the relevant background theory and related work. Next, we describe the chaotic circuit and our approach for using it as a reservoir. We then explain our experimental setup, followed by results and discussion. Finally we conclude with a discussion of future work.

Background

Inspired by the wide array of intrinsic physical computation found in nature, *Evolution in-materio* takes a bottom-up approach to evolve computation in physical matter (Miller et al., 2014). These efforts have been rather successful and a variety of computational devices have been evolved, e.g. a tone discriminator and robot controller in liquid crystal

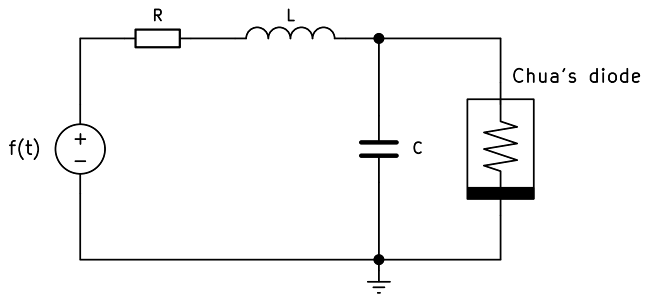

Figure 2: Driven Chua's circuit, adapted from Murali et al. (1994a)

(Harding and Miller, 2004, 2005) and logic gates evolved in carbon nanotubes (Lykkebø et al., 2014; Massey et al., 2015) and gold nanoparticles (Bose et al., 2015).

Stepney (2008) argues that the time is ripe to climb the "neglected pillar of material computation". To be successful we must find computational models that are natural for the physical substrate. Stepney further argues that model-free evolutionary search provides limited insight into what (physical) mechanisms are being exploited and for what reason.

Links between dynamical systems and information theory were established over half a century ago when Shannon entropy was used to describe uncertainty in nonlinear dynamics (see Crutchfield et al. (2010) for a historical discussion).

Shaw (1981) argues that attractors act as information sinks: in the joining of trajectories, information about the state history of the system is lost. Chaos conversely acts as an information source: divergence of trajectories brings into view new information not present in the initial conditions.

It has been suggested that chaos plays an important role in natural systems by providing a rich repertoire of dynamics that may be utilized for increased performance (Sinha and Ditto, 1998).

Reservoir Computing has its roots in neural network research, where it was discovered that the rich dynamics and memory capabilities of recurrent neural networks (RNNs) could be exploited without any training of the network. Good performance could be obtained with a random network coupled with a linear readout layer trained on the activations of the RNN nodes (Jaeger, 2001; Maass et al., 2002).

Figure 1 shows the typical RC architecture with three distinct parts: the input layer, the reservoir with its many recurrent nodes, and the readout layer. Note that the readout is the only trained part, both the input layer and the reservoir remain unchanged.

Formally, the reservoir transforms a low-dimensional time-dependent input $u(t)$ into a high-dimensional state vector $\boldsymbol{x}(t)$ which is more readily processed by the linear readout. The reservoir acts as a nonlinear kernel with memory, maintaining a rich, high dimensional, nonlinear transforma-

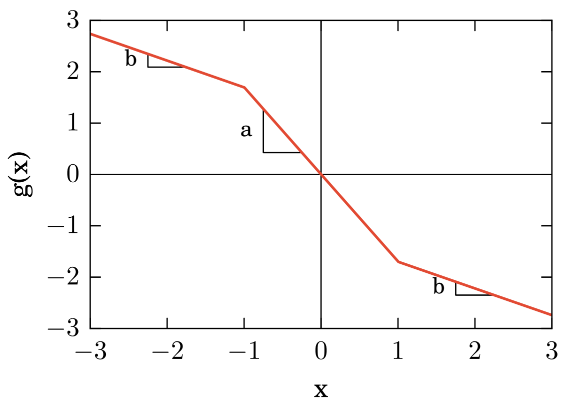

Figure 3: Current-voltage characteristic of Chua's diode. Given the (normalized) voltage x across the diode, the current through the diode is given by $g(x)$. The characteristic slopes a and b from equation (1) are also depicted.

tion of input history.

RNN reservoirs have outperformed state of the art methods in a wide range of challenging temporal tasks such as speech recognition and time series prediction. For an overview of RC methods, see Lukoševičius and Jaeger (2009).

The reservoir can be any kind of dynamical system, as long as it can be perturbed by input and its output observed. This has inspired a diverse range of physical systems used as reservoirs. Examples include an optoelectronic system (Paquot et al., 2012), a photonics chip (Vandoorne et al., 2014), nanoscale switch networks (Sillin et al., 2013), the bacterium Escherichia coli (Jones et al., 2007) and even a bucket of water (Fernando and Sojakka, 2003).

System description

In this work we consider a simple nonlinear circuit as a reservoir. We have chosen the Driven Chua's circuit introduced by Murali et al. (1994a,b) since it is one of the simplest circuits with a rich variety of dynamics. The circuit (Figure 2) consists of only a handful of components: three linear elements (a resistor, an inductor and a capacitor) and a nonlinear resistor (a Chua's diode). An external periodic forcing signal $f(t)$ drives the dynamics of the circuit.

The circuit is described by the following set of normalized differential equations:

$$\begin{aligned} \dot{x} &= y - g(x) \\ \dot{y} &= -\beta y - \beta x + Ff(\omega t) \\ g(x) &= bx + 0.5(a-b)[|x+1| - |x-1|] \end{aligned} \quad (1)$$

where x corresponds to the voltage across the capacitor C and y corresponds to the current through the inductor L. The term $Ff(\omega t)$ is the external forcing signal with amplitude F and angular frequency ω. Note that in the original paper, the periodic driving force was sinusoidal, i.e. $f(t) = sin(t)$. Here we generalize the forcing term to include any type of periodic function.

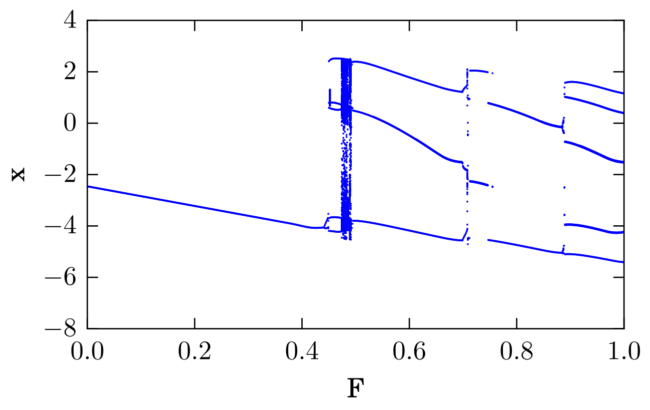

Figure 4: Bifurcation diagram as the forcing amplitude F is increased from 0.0 to 1.0. The system shown has $\beta = 1.0$, $a = -1.70$ $b = -0.52$ and $\omega = 0.7$.

$g(x)$ is the equation for the Chua's diode which has a piecewise-linear current-voltage characteristic as shown in Figure 3. The three linear regions have slopes a and b as shown with breakpoints at ± 1.

The dynamics of the circuit depends on the parameters β, a, b, ω and F. Figure 4 shows the bifurcation diagram as we increase the amplitude F of the forcing signal. Several interesting phenomena can be observed, such as period-doubling bifurcations, chaos and periodic windows.

For the current study we consider the case where the slopes of the Chua's diode are in the range $a < -1$ and $-1 < b < 0$. In the absence of the external forcing ($F = 0$), the system has three fixed points in this case: an unstable fixed point at the origin and two stable fixed points P^+ and P^-.

For the first part of this work we simulate the system numerically using the normalized version of the equations. When implementing the circuit physically we translate back to the corresponding circuit equations.

Input/output encoding

Before any dynamical system can be used as a reservoir, we need to decide how to perturb the system with input (the input encoding), and how to observe the corresponding response (the output decoding). Since we also wish to realize the circuit experimentally, we must keep in mind the physical constraints of the system, as well as the limitations of electrical components and test equipment.

First we tackle the question of input encoding: where in our system do we apply the input signal $u(t)$ and how should this signal be encoded? Note that the focus here is on non-temporal tasks, where the input u does not depend on time.

A common approach is to apply input as the initial condi-

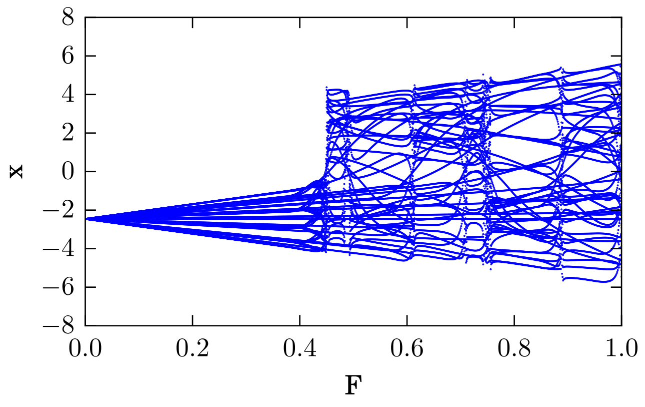

Figure 5: Bifurcation diagram of transients as the forcing amplitude F is increased from 0.0 to 1.0. The system shown has $\beta = 1.0$, $a = -1.70$, $b = -0.52$ and $\omega = 0.7$, sampled with $k = 10$ for $N = 5$ periods.

tions (x_0, y_0) of the system. This is possible in simulation where we are free to choose initial conditions, however in a physical system this may not be the case. For instance, forcing a physical system towards an unstable fixed point may be impossible.

In this work, we instead apply input as part of the external forcing signal $Ff(\omega t)$. We map the input u to the amplitude F of the forcing signal in a linear range $[F_{min}, F_{max}]$, i.e. u is first normalized to the range $[0, 1]$ then $F = F_{min} + (F_{max} - F_{min})u$. Instead of sinusoidal forcing, we use square waves since they are more easily generated with our test equipment, i.e. $f(t) = sgn(sin(t))$. For each input u, the system is perturbed for N periods of the forcing signal. Between each input the system is reset to start in the same stable fixed point.

The output of our reservoir is the (discretized) transients of the system during the fixed perturbation period. We can visualize the output as a bifurcation diagram of transients, as shown in Figure 5. With such an output mapping, the reservoir produces many nonlinear transformations as a function of the input F. Compared to the long-term dynamics of the same system (Figure 4), the transients produce a richer repertoire of nonlinear functions.

Specifically, we observe the variable $x(t)$ while the system is perturbed by input. We record $x(t)$ at a fixed sampling rate $k\omega$, always an integer multiple of the forcing frequency, to obtain kN discrete output samples from the system. The reservoir state vector is thus $\boldsymbol{x} = [x(0), x(\tau), x(2\tau), ..., x(kN\tau)]$ where $\tau = 2\pi/k\omega$ is the sample interval. Note that with $k = 1$ we get the Poincaré map.

Experimental setup

The dynamics of our reservoir depends on several parameters which will affect performance. For the current study we

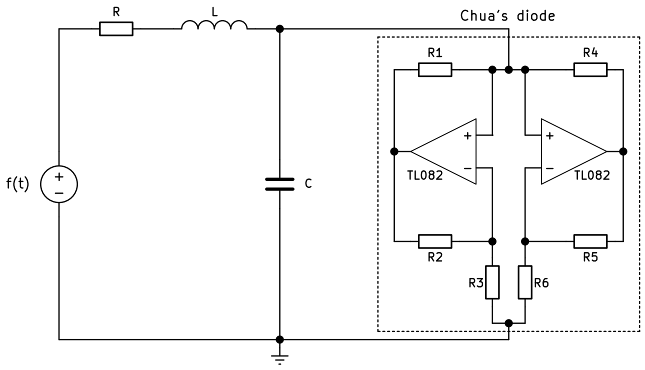

Figure 6: Driven Chua's circuit implemented using an active version of Chua's diode based on two op-amps.

fix $\beta = 1.0$, $\omega = 0.7$ and vary the slopes in the intervals $a = (-2, -1)$ and $b = (-1, 0)$.

From Figure 5 we observe a linear region from $F = 0$ up to the first bifurcation point (here at $F = 0.45$). As a reservoir, such a linear region is uninteresting (since any linear function can be constructed by the readout layer alone) so we should set F_{min} past the first bifurcation point. The width of the linear region depends on the parameters a, b, and ω. Thus for a given set of these parameters, there exists a (problem-specific) optimal value for F_{min} and F_{max}. However, to reduce the number of parameters we need to explore, we experimentally fix $F_{min} = 0.5$ and $F_{max} = 1.0$.

We perturb the reservoir with $N = 5$ periods of square waves and record $kN = 100$ output samples. Between each input, the system is reset to a stable fixed point, i.e. we set the initial condition $(x_0, y_0) = P^+$.

Tasks

We evaluate the performance of the reservoir on two non-temporal tasks: nonlinear regression and nonlinear classification.

The goal of the regression task is to approximate the 7th degree polynomial

$$y = (x - 3)(x - 2)(x - 1)(x)(x + 1)(x + 2)(x + 3)$$

in the range $(-3, 3)$. Such a smooth function was selected to test the reservoir's generalization capabilities and to investigate the effect of bifurcations and chaos on reservoir performance.

For classification we use the classical "circles" dataset with two classes organized in concentric rings (Ben-Hur et al., 2001). This is a simple dataset which requires a nonlinear decision surface. Being two-dimensional, it can be easily visualized which enables graphical analysis. We reset the reservoir between the application of each input feature, i.e. the x and y coordinates of each point are independently transformed by the reservoir. Furthermore, the number of

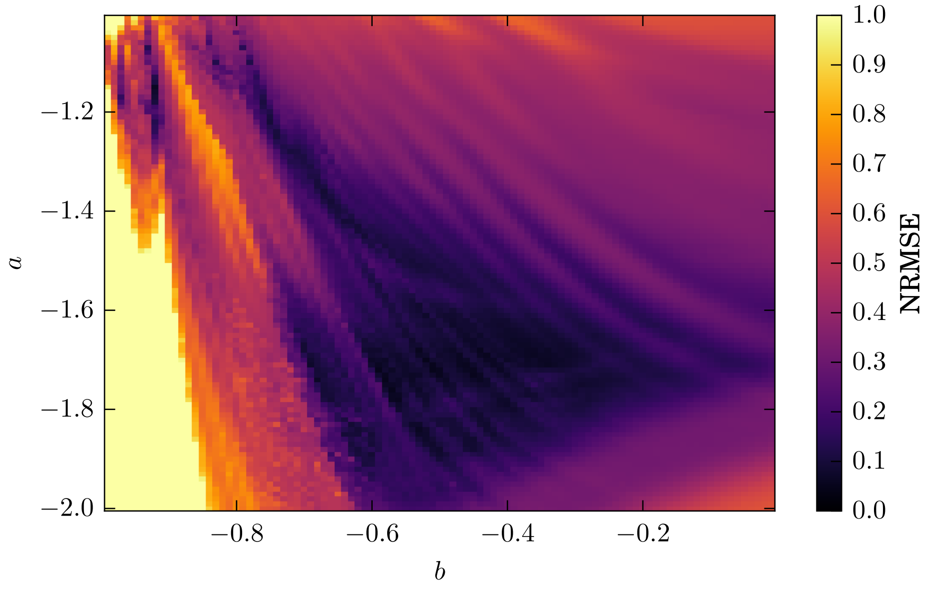

Figure 7: Score on the regression task as a function of the slopes a and b. Mean NRMSE from ten-fold CV is shown, where values have been capped to 1.0.

samples was set to $kN = 50$ for each feature resulting in a total of 100 output samples.

For each of the tasks, 10 000 examples were generated of which 75% were used for training and 25% for validation. As performance metric we have used the normalized root mean square error (NRMSE) for regression and accuracy for classification.

Ridge regression was used for the readout layer, which is widely adopted within the RC community as it reduces overfitting thanks to a regularization term (Hoerl and Kennard, 1970).

Simulation experiments

First we evaluate reservoir performance in simulation, i.e. the governing equations (1) were integrated numerically.

We sweep the slopes of the Chua's diode in the intervals $a = (-2, -1)$, $b = (-1, 0)$ and evaluate performance on the regression task. We use ten-fold cross validation on the training set to evaluate reservoir performance.

We then have a closer look at the best performing reservoir by analysing the approximated polynomial on the validation set.

Next we apply the same reservoir on classification, to demonstrate that the reservoir can be re-used for a different task.

Circuit experiments

Based on the sweeps from simulation, we select a good performing reservoir which we implement on a printed circuit board. We use high-quality components with low tolerances whenever possible. Some components are not readily available with low tolerances (such as the inductor L), so these must be measured.

Figure 6 shows our circuit implementation. A passive version of Chua's diode doesn't exist, but an active version can be implemented using two op-amps (Kennedy, 1992) which is what we have done here. This implementation allows the slopes to be set by the choice of resistors $R_1 - R_6$.

Given the normalized set of parameters $\beta = 1.0$, $a = -1.70$, $b = -0.52$ and $\omega = 0.7$ (the best performing reservoir from the sweeps), we calculate values for the circuit components as follows: select $L \approx 14mH$ (measured) and $C = 10nF$. Calculate $R = \sqrt{\beta L/C} = 1185\Omega$ and determine the frequency of the forcing signal $\Omega = \omega/RC \approx 9.4kHz$. The required slopes of the Chua's diode are then $G_a = Ga \approx -1.435mS$ and $G_b = Gb \approx -0.439mS$ where $G = 1/R$.

For Chua's diode we adjust the breakpoints $B_p = \pm 0.8V$ so that the dynamics of the circuit stays within the range of our ADC. Adjustable positive and negative power supplies of the op-amps allows fine-tuning these breakpoints.

The slopes of the Chua's diode are determined by resistors $R_1 - R_6$. Following the design procedure in Kennedy (1992), we set $R_1 = R_2 = 100\Omega$, $R_3 = E_{sat}/((B_p - E_{sat})G_b - B_p G_a) \approx 1915\Omega$, $R_4 = R_5 = E_{sat}/(B_p(G_b - G_a)) \approx 10052\Omega$ and $R_6 = E_{sat}/((E_{sat} - B_p)(G_b - G_a)) \approx 1116\Omega$ where $E_{sat} = 8.0V$ is the saturation level of the op-amps.

After application of input, the circuit may end up in either of the two stable fixed points P^+ and P^-. To make sure the circuit starts in the fixed point P^+ before each input, we use the following reset procedure: first a constant positive volt-

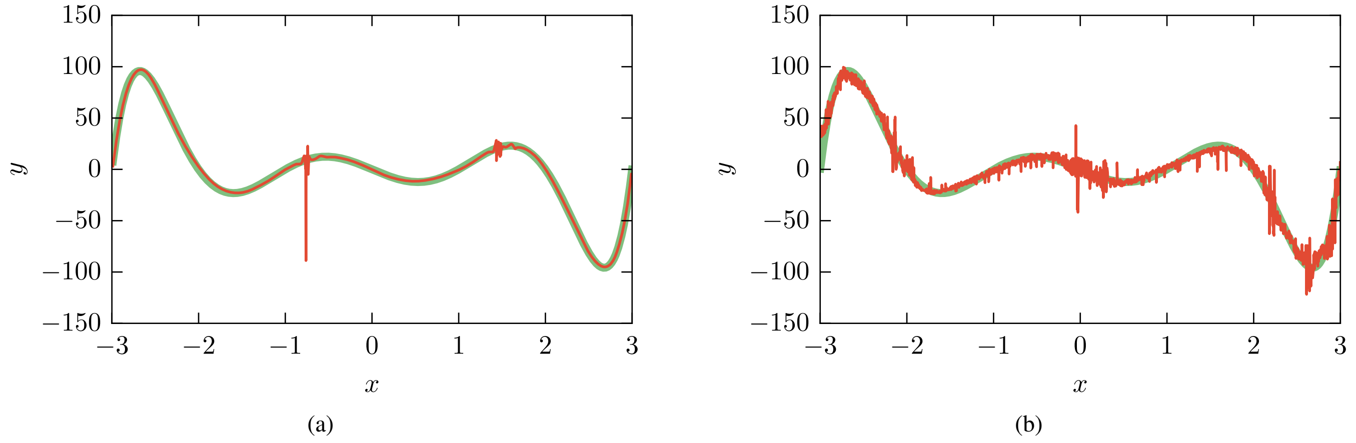

Figure 8: Approximated polynomial (red line) on validation set: (a) simulation NRMSE=0.07 (b) circuit NRMSE=0.16. The target polynomial is shown in green.

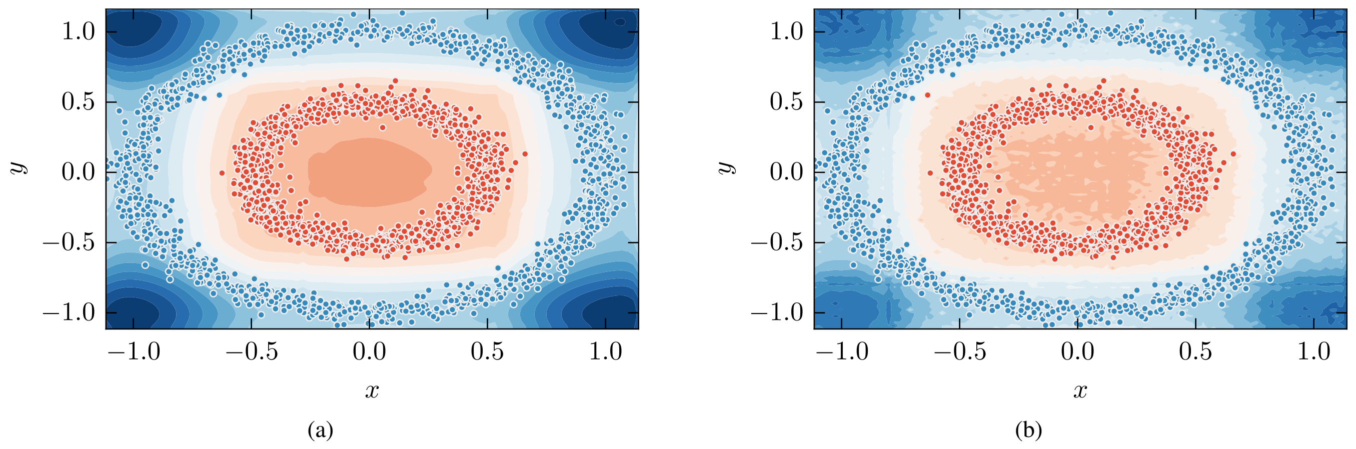

Figure 9: Decision surface and classification result on validation set: (a) simulation 100% accuracy (b) circuit 99.9% accuracy.

age $F > -a - 1$ is applied at the input $f(t)$. This has the effect of destroying the other two fixed points, leaving a single fixed point close to P^+ which the system will approach. Next, the constant voltage is removed ($F = 0$), causing the the system to return to the nearest fixed point P^+ as desired. The duration of the reset period must be sufficiently large ($\gg 1/\Omega$) to allow transients to settle.

To interface with the circuit we use the Mecobo platform (Lykkebø et al., 2014). The board can generate analog voltage signals using an onboard DAC and record analog voltages with an onboard ADC. For our experiments we use a 12-bit DAC (AD5308) with range set to $\pm 1.024V$ and a 13-bit ADC (AD7327) with a range of $\pm 5V$.

Results

Regression

Figure 7 shows the mean NRMSE of the reservoir plotted as a function of the slopes a and b. There is a fairly large region of parameter space with NRMSE below 0.1. The best performing reservoir has slopes $a = -1.70, b = -0.52$ and a NRMSE of $0.05(\pm 0.02)$. We can also see that performance is fairly smooth as a function of the two parameters.

Figure 8a shows the approximated polynomial on the validation set obtained with the best performing reservoir. For a majority of the input range, the approximation is almost perfect. There is however two noisy regions at around $x = -0.75$ and $x = 1.45$. The NRMSE score on the validation set is 0.07.

With the circuit reservoir we obtain an NRMSE score of 0.16 on the validation set (Figure 8b). Although the overall shape of the approximation resembles the desired polynomial, there is a fair amount of noise present. Note that no filtering has been performed on the sampled voltage data.

Classification

On the classification task, the simulated reservoir obtains perfect accuracy. Figure 9a shows the decision surface for the two classes, with the validation data superimposed. As can be seen, the decision surface is smooth and there is a sizeable margin separating the two classes.

Figure 9b shows the classification results with the circuit reservoir. Excellent performance is obtained with only one misclassified point (99.9% accuracy). However, the decision surface is markedly more noisy compared to simulation.

Discussion

Our results demonstrate that a chaotic circuit can be exploited for reliable computation within a reservoir computing framework.

The a and b parameter sweeps revealed a sizeable region of parameter space where good performance was obtained. Furthermore, the performance landscape has many smooth features, which looks promising for further exploration of the parameter space with stochastic search methods.

The sweep was performed with fixed values for F_{min} and F_{max} which, as highlighted earlier, may be a suboptimal choice depending on the slopes. It is likely that the overall performance could be improved quite substantially if F_{min} and F_{max} was tuned individually for each pair of slopes.

Although good results were obtained on the polynomial task, there were two noisy regions in the approximation. They were also present on results from the training set, although not as pronounced. These regions are likely caused by instability close to bifurcation points, where chaotic dynamics dominate even the initial transients. The locations of the noisy regions ($x = -0.75$ and $x = 1.45$) correspond to $F \approx 0.69$ and $F \approx 0.87$ with our input encoding. This is very close to two narrow chaotic bands which can be seen in the bifurcation diagram (Figure 4).

For the classification task, near-perfect performance was obtained with a very smooth decision surface. This result demonstrates that the same reservoir can be re-used for two quite different tasks, by re-training the readout layer only. This is particularly relevant for physical reservoirs whose properties cannot easily be changed to suit a particular task.

Our experimental results with the circuit implementation of the reservoir revealed comparable performance to that of simulation. However, there is clear performance degradation caused by noise, especially on the regression task. Classification seems more robust to this noise where we can get away with a rather rugged decision surface.

We can attribute much of the noise due to sampling errors which will be particularly large in regions of the signal with steep slopes. Although noise will always be a problem when dealing with a physical system, it can have an even more pronounced effect in sensitive chaotic systems.

Finally, the dynamics of our circuit implementation did deviate somewhat from simulation, likely due to nonidealities in the circuit components. However, good performance was still obtained, which illustrates the robustness and power of the Reservoir Computing methodology.

Conclusion

In this work we have shown that a simple chaotic circuit can be used effectively as reservoir. We demonstrate that its rich dynamics can be exploited directly for computation, both in simulation and with a circuit implementation. To the best of our knowledge, this is the first time a chaotic circuit has been used as a reservoir.

Here we have restricted the scope to non-temporal tasks. Future work will explore ways in which the chaotic circuit can be used to process temporal signals as well. Using the reservoir for more difficult tasks should also be attempted, e.g. with real-world datasets that contain more noise and/or require a more complex decision surface.

Different input/output encoding schemes should be investigated to obtain a richer repertoire of nonlinear transformations. Further exploration of parameter space through e.g. evolutionary search is likely to find better performing reservoirs.

To further investigate reservoirs within the same family, i.e. with a single nonlinear node, the Chua's diode could be replaced with other nonlinear elements. Memristive devices could potentially serve as nonlinear memory, making the circuit applicable to tasks such as speech recognition.

Our chaotic circuit can be viewed as an electrical analogy of any physical system with similar dynamic properties. The methods and results presented herein should therefore be transferable to any natural system that can be manipulated to behave within the desired dynamic regime. Given that such a simple physical system can be exploited, a wide variety of natural systems can be viewed as potential reservoirs. The grand goal of exploiting intrinsic properties of matter for computation seems within reach, paving the way towards highly efficient computation at the nanoscale.

References

Appeltant, L., Soriano, M., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C., and Fischer, I. (2011). Information processing using a single dynamical node as complex system. *Nature Communications*, 2:468.

Ashby, W. R. (1960). *Design for a Brain: The Origin of Adaptive Behavior*. Chapman & Hall, London, England.

Ben-Hur, A., Horn, D., Siegelmann, H. T., and Vapnik, V. (2001). Support vector clustering. *Journal of Machine Learning Research*, 2:125–137.

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

Bose, S. K., Lawrence, C. P., Liu, Z., Makarenko, K. S., van Damme, R. M. J., Broersma, H. J., and van der Wiel, W. G. (2015). Evolution of a designless nanoparticle network into reconfigurable Boolean logic. *Nature Nanotechnology*, 10(12):1048–1052.

Crutchfield, J. P., Ditto, W. L., and Sinha, S. (2010). Introduction to focus issue: Intrinsic and designed computation: Information processing in dynamical systems-beyond the digital hegemony. *Chaos*, 20(3).

Fernando, C. and Sojakka, S. (2003). Pattern Recognition in a Bucket. *Advances in Artificial Life*, 2801(12):588–597.

Goh, W. J. and Crook, N. (2007). Pattern Recognition using Chaotic Transients. In *ESANN*, number April, pages 25–27.

Harding, S. and Miller, J. (2005). Evolution In Materio : A Real-Time Robot Controller in Liquid Crystal. In *2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*, number January, pages 229–238. IEEE.

Harding, S. and Miller, J. F. (2004). Evolution in materio: A tone discriminator in liquid crystal. *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, 2:1800–1807.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge Regression: Bias Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks - with an Erratum note. Technical Report 148, German National Research Center for Information Technology.

Jones, B., Stekel, D., Rowe, J., and Fernando, C. (2007). Is there a Liquid State Machine in the Bacterium Escherichia Coli? *2007 IEEE Symposium on Artificial Life*, pages 187–191.

Kennedy, M. P. (1992). Robust OP Amp Realization of Chua's Circuit. *Frequenz*, 46(3-4):66–80.

Langton, C. G. (1984). Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):135–144.

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

Lykkebø, O. R., Harding, S., Tufte, G., and Miller, J. F. (2014). Mecobo: A Hardware and Software Platform for In Materio Evolution. In Ibarra, O. H., Kari, L., and Kopecki, S., editors, *Lecture Notes in Computer Science 2014*, volume 8553 of *Lecture Notes in Computer Science*, pages 267–279. Springer International Publishing, Cham.

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

Mainzer, K. (2007). *Thinking in Complexity*. Springer Berlin Heidelberg, Berlin, Heidelberg, 5 edition.

Markram, H. (2012). The Human Brain Project. *Scientific American*, 306(6):50–55.

Massey, M. K., Kotsialos, A., Qaiser, F., Zeze, D. A., Pearson, C., Volpati, D., Bowen, L., and Petty, M. C. (2015). Computing with carbon nanotubes: Optimization of threshold logic gates using disordered nanotube/polymer composites. *Journal of Applied Physics*, 117(13):134903.

Miller, J. F., Harding, S. L., and Tufte, G. (2014). Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67.

Murali, K., Lakshmanan, M., and Chua, L. (1994a). The simplest dissipative nonautonomous chaotic circuit. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 41(6):462–463.

Murali, K., Lakshmanan, M., and Chua, L. (1994b). BIFURCATION AND CHAOS IN THE SIMPLEST DISSIPATIVE NON-AUTONOMOUS CIRCUIT. *International Journal of Bifurcation and Chaos*, 04(06):1511–1524.

Ozturk, M. C. and Principe, J. C. (2005). Computing with transiently stable states. *Proceedings of the International Joint Conference on Neural Networks*, 3:1467–1472.

Paquot, Y., Duport, F., Smerieri, A., Dambre, J., Schrauwen, B., Haelterman, M., and Massar, S. (2012). Optoelectronic Reservoir Computing. *Scientific Reports*, 2:1–6.

Shaw, R. (1981). Strange Attractors, Chaotic Behavior, and Information Flow. *Zeitschrift für Naturforschung A*, 36(1):81–87.

Sillin, H. O., Aguilera, R., Shieh, H.-H., Avizienis, A. V., Aono, M., Stieg, A. Z., and Gimzewski, J. K. (2013). A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology*, 24(38):384004.

Sinha, S. and Ditto, W. (1998). Dynamics Based Computation. *Physical Review Letters*, 81(1):2156–2159.

Stepney, S. (2008). The neglected pillar of material computation. *Physica D: Nonlinear Phenomena*, 237(9):1157–1164.

Stepney, S. (2012). *Nonclassical Computation — A Dynamical Systems Perspective*. Springer Berlin Heidelberg.

Strogatz, S. H. (2015). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering (Steven H. Strogatz)*. Westview Press, 2 edition.

Toffoli, T. (2004). Nothing makes sense except in light of evolution. *International Journal of Unconventional Computing*, 1:3–29.

Vandoorne, K., Mechet, P., Van Vaerenbergh, T., Fiers, M., Morthier, G., Verstraeten, D., Schrauwen, B., Dambre, J., and Bienstman, P. (2014). Experimental demonstration of reservoir computing on a silicon photonics chip. *Nature communications*, 5:3541.

Wiener, N. (1961). *Cybernetics: or Control and Communication in the Animal and the Machine, Second Edition*. MIT Press, Cambridge.