Reservoir Computing: Evolution *in materio*'s Missing Link

Matthew Dale[1,3], Julian F. Miller[2,3], Susan Stepney[1,3], Martin A. Trefzer[2,3]

[1]Department of Computer Science, University of York, UK
[2]Department of Electronics, University of York, UK
[3]York Centre for Complex Systems Analysis
md596@york.ac.uk

Abstract. Reservoir Computing (RC) is a computational framework that has the potential to transfer onto any input-driven dynamical system, given two properties are present; a fading memory and input separability. A typical reservoir consists of a fixed network of recurrently connected processing units. When excited, the network maps the incoming temporal signal into its high-dimensional state space. The resulting spatial-temporal state space can then be trained to approximate a vast number of functions. To train the reservoir, a simple readout mechanism is adapted, selecting and combining different reservoir states to form the desired output. Our recent work has demonstrated how this framework can be applied to randomly-formed carbon nanotube composites to solve computational tasks. In this paper, we compare our proposed reservoir computing *in materio* technique to the original evolution *in materio* technique using the *Iris* classification task. The results show that despite the non-temporal nature of the task, reservoir computing *in materio* can outperform evolution *in materio* using the same composites.

Keywords: Material Computation, Evolution *in materio*, Reservoir Computing, Unconventional Computing, Machine Learning Classification

1 Introduction

Research in unconventional computing has shown that many diverse dynamical systems can be exploited to perform computational tasks [1]. Exploiting computation at the substrate-level offers potential advantages over classical computing architectures, such as exploiting physical and material constraints that could offer solutions"for free", or at least computationally cheaper [17].

Computational composites consisting of randomly dispersed carbon nanotubes have shown interesting electrical properties that can be trained to perform a variety of computational tasks [3]. To extract computation from these composites a technique known as evolution *in materio* is applied [14]. Evolution *in materio* attempts to intrinsically evolve physical solutions to computational problems, for example, by manipulating the electrical response of a material to input stimuli. This is achieved by evolving a number of physical parameters such as the value, type and placement of input signals on the material.

Evolution *in materio* is a growing field with new hardware developments, experimental materials and directions appearing from different research groups, e.g. [2, 13]. However, what is classed here as the "*vanilla*" technique has its limitations. For example, the technique typically used in the literature produces a discretised output, represented by voltage samples averaged in an output buffer. In that technique, "programming" is typically done using digital voltage signals, such as square waves, between a limited voltage range (due to hardware limitations). Using such a model one cannot fully exploit the temporal properties of the material, which is argued here to be a rich source of untapped information.

In recent work [6, 7], we have shown that by applying the Reservoir Computing framework to the same materials, we can exploit interesting temporal properties that were previously unused. The new addition of the reservoir framework has enabled temporal problems requiring both dynamic behaviours and memory to be tackled. Here, we investigate how well the new reservoir/material framework compares in performance to the *vanilla* evolution *in materio* technique on a non-temporal task. The task is to perform classification on the Fisher Iris data set across a range of carbon nanotube composites, including a control (a resistor array) and the same material used in [15]. The last section of this paper presents two simple analysis techniques that could provide further understanding as to what reservoir/material properties are being exploited by evolution.

2 Reservoir Computing

Reservoir Computing first emerged as efficient mechanism for training recurrent neural networks and later evolved into a general theoretical model for many dynamical systems [16]. By applying only a relatively simple training mechanism many physical systems have become exploitable unconventional computers (see summary [5]).

A reservoir functions much like a temporal *kernel* [10] and can represent any excitable non-linear medium, given the medium can: (i) create a high-dimensional projection of the input into observable *reservoir states*; and (ii) possess a fading memory of previous inputs and internal states. These prerequisite properties are described by the *separation*, *approximation* [11] and *echo state* properties [8]. With these properties a reservoir can realise any non-linear filter with bounded memory, and with the aid of a trained readout approximate any function.

To interpret and train the material as a reservoir we define the observed reservoir states $x(n)$ of the material as a combination of the *continuous* material and the *discrete* observation function:

$$x(n) = \Omega(\mathcal{E}(W_{in}u(t)) \tag{1}$$

where $\Omega(n)$ is the observation of the macroscopic material behaviour and $\mathcal{E}(t)$ the microscopic material function when driven by the input $u(t)$, multiplied by the input weight matrix W_{in}.

The training of the reservoir readout is typically done using Ridge Regression [9], manipulating the weights W_{out} to reduce the error (Normailised Root Mean Squared Error *NRMSE*) between the training signal $y_{target}(n)$ and the reservoir output $y(n)$:

$$W_{out} = Y_{target} X^T (XX^T + \beta I)^{-1} \tag{2}$$

where I is the identity matrix and β the Tikhonov regularisation parameter. The trained reservoir system is given in (3).

$$y(n) = W_{out} x(n) \tag{3}$$

3 Experimental Platform

The materials used in this experiment consist of various concentrations (w.r.t weight) of single-walled carbon nanotubes, randomly dispersed in an insulating polymer: poly-butyl-methacrylate (PBMA) or poly-methyl-methacrylate (PMMA). Each material is deposited onto a gold/chromium microelectrode array of 12 electrodes to form input-outputs to the material. The electronic properties of the dispersed nanotubes are approximately $\frac{1}{3}$ metallic nanotubes and $\frac{2}{3}$ semiconducting nanotubes. The relative size of the nanotubes (100nm to 1000nm length and diameter between 0.8nm and 1.2nm) is significantly less than the gap between the electrodes (between 100 to 150 µm) suggesting the nanotubes form conductive pathways between the electrodes. The polymer mixed with the nanotubes acts both as a dielectric and as a suspension for the nanotube network. In previous experiments, it has been observed that both a conductive network is formed and a high computational performance is reached around a percolation threshold of 1% carbon nanotubes [6, 7, 12].

To interface the computer with the material, two input/output data acquisition cards are used. Each card is controlled by a MATLAB interface and is set to either input, or read, analogue voltages from the micorelectrode array. To allow each card to communicate with any electrode, a cross-point switch is added allowing evolution to reconfigure and route different input-output combinations.

4 Machine Learning Classification

The Iris data set[1](also known as *Fisher's* Iris data set) is a well-known multivariate data type classification problem that was previously used to benchmark the evolution *in materio* technique [4, 15]. The task is to classify three species of the Iris plant given the four attributes petal and sepal length and width. The Iris data set contains 150 instances, with 50 instances of each class/species. The data set was evenly divided into training and testing sets of 75 randomised instances,

[1]The Iris data set can be found on the UCI Machine learning repository at: https://archive.ics.uci.edu/ml/

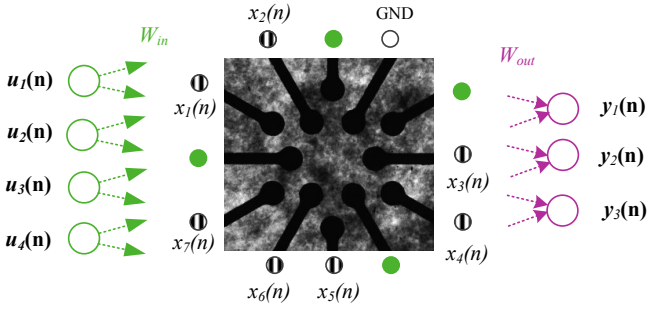


Fig. 1: Input-Output mapping of the task inputs $u(n)$ and observed reservoir states $x(n)$. For this task 4 inputs are required ($u_{1:4}(n)$), each input is multiplied by the input weight matrix W_{in} and applied to an electrode (green). To form the 3 output classes $y_{1:3}(n)$, the reservoir states ($x_{1:7}(n)$) are multiplied by the output matrix W_{out} (the trained readout layer).

containing 25 instances of each class. Each class in the reservoir framework is represented as a separate reservoir output (Fig.1) with a binary value, and each attribute is represented by a floating-point input voltage.

5 Training Reservoir Computers

Each material is deposited onto a microelectrode array, providing electrical inputs and outputs to the system. Using computer-based evolution, the role of each electrode is decided as well as some other associated parameters. For example, evolution decides which electrodes are inputs $u(n)$, or outputs $x(n)$, and evolves a *weight* value for each input (see Fig.1). In addition to the weights, another parameter is evolved called the *leak rate*. The leak rate parameter α is used to match the dynamics of the reservoir to the task input and/or target output. The leak rate parameter is applied post-collection of states, see eq.(4). This turns eq.(3) into eq.(5) – this parameter, however, may have little effect on reservoir performance for non-temporal tasks.

$$\tilde{x}(n) = (1-\alpha)x(n-1) + \alpha\Omega(\mathcal{E}(W_{in}u(t)) \tag{4}$$

$$y(n) = W_{out}\tilde{x}(n) \tag{5}$$

In this experiment we apply an evolutionary strategy of (1+4) of 150 generations, across 20 runs. The implementation of this strategy when combined with Gaussian mutation operators functions similar to a hill-climber algorithm. This is set to reduce the retention effect of degenerative fitness jumps experienced by rapidly changing the configuration parameters.

Training and testing is split into two phases. Using the *training set*, evolution selects appropriate input-output mappings and ridge regression trains the

readout layer. Then in testing, a final evaluation of the material configuration (and trained readout) is carried out using the *test set*.

5.1 Fitness Evaluation

To train the reservoir and evolve a solution, the *NRMSE* between the trained output and the target output is used to define fitness. However, as the task is to classify binary classes – not a time-series output – a threshold mechanism is used to translate the trained outputs into binary classes. To evaluate the accuracy of the classifier, and to conduct a fair comparison to the evolution *in materio* technique, the fitness calculation in [15] is applied. This is provided in eq.(6), evaluating the number of true positives *TP*, true negatives *TF*, false positives *FP* and false negatives *FN* that occur.

$$\text{Fitness} = \frac{TP.TN}{(TP + FP)(TN + FN)} \tag{6}$$

To select a threshold, a simple optimisation loop was used (post-state collection). The best threshold was then attached and stored to the given configuration and reimplemented at the testing phase.

6 Experimental Results

Fig.2 and Fig.3 show the results of both simulated and physical reservoirs on the Iris task. Fig.2 shows that performance varies with respect to carbon nanotube concentration, with concentrations of 0.71% and above outperforming the others, and well above the control material (resistor array). In Fig.2, two 0.71% PMMA materials are shown, one of which is the material used in [15]; which one is unknown. Both outperform the reported *average* test accuracy of 77.1%, with both average *training* and *test* accuracies above 90%.

Fig.3 compares the performance of a physical reservoir (the 1% PBMA material) to three evolved simulated (*in silico*) reservoirs. The similarity in accuracies suggests the material is forming an optimised reservoir and is possibly exploiting the benefits of both a smaller readout and a larger network; much like the sampled version which is only using 7 states out of a possible 50. The performances of these reservoirs also compare favourably to the Cartesian Genetic Programming method stated in [15] (*train*: 97.7%, *test*: 93.6%), with training accuracies typically above 95% and test accuracies averaging between 90-95%. These results therefore suggest our *in materio* method can compete with two optimised *in silico* methods on this task.

The results of this experiment are surprising, showing that applying the reservoir computing framework on a non-temporal task, performance can be increased over *vanilla* evolution *in materio*. The results appear counter intuitive on the surface, as one would assume the direct programming nature of evolution *in materio* would outperform our technique. However, this is not the case. Possible explanations for this increase in performance might be that: (i) stimulating the

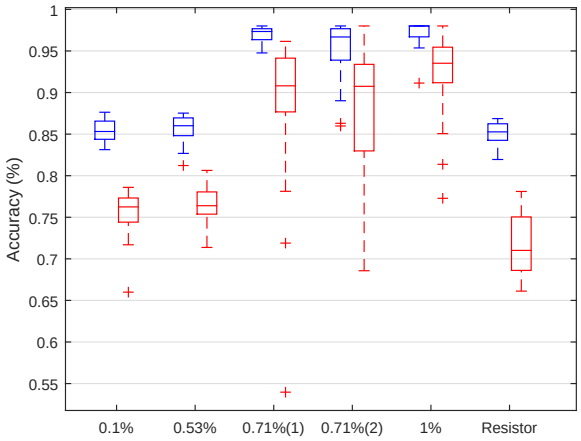


Fig. 2: Experimental results of each material using eq.(6). Each material shows the training accuracy (blue) and test accuracy (red) across 20 evolutionary runs. Two 0.71% PMMA materials are given, with one being the same used in [15]

material with variations of the input causes interesting interactions not present in the *vanilla* technique; (ii) a conductive network might not be present across all electrodes and instead several networks may exist across the array, therefore additional inputs-outputs grant access to each of these networks; (iii) combining the outputs and weighting their importance allows training to exploit the whole material rather than exploiting a single area around a particular electrode (which relates closely to (ii)). These possibilities are suggested here as starting points for further exploration.

7 Analysing Reservoir and Material Characteristics

As part of the investigation three evolved simulated reservoirs (*Echo State Networks* [8]) of different sizes were added for comparison (Fig.3). Each reservoir has three key parameters that were selected through evolution; the *spectral radius*, *input scaling* and *leak rate*. Each parameter is adjusted to tune the dynamical behaviour and memory of the reservoir. In Echo State Networks, the spectral radius determines how fast the influence of the input degrades and the stability of the reservoirs activations. The Input scaling parameter is used to tune the non-linearity of the reservoir and tune the proportional effect the current input has compared to previous activations. The leak rate parameter is used to match the speed of the reservoirs dynamics to the task input and/or output; essentially applying additional filtering to the activation of each node [9].

Fig.4 shows the relationship between parameters and task performance for each evolved network. By visualising the evolved parameters and their relative performance, we can determine the desired dynamics for a given task and suggest what dynamics need to be exploited in the material. From the plots we can see

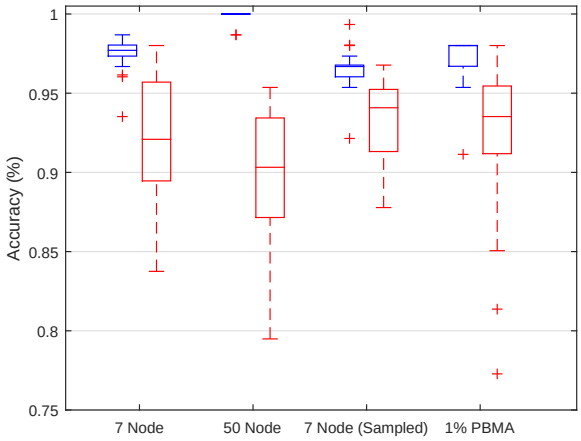


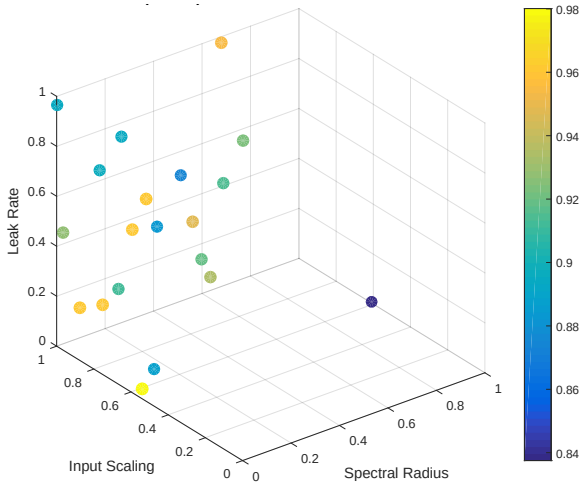
Fig. 3: Experimental results of three simulated reservoirs and the 1% PBMA material using eq.(6). Training accuracy (blue) and test accuracy (red) are given across 20 evolutionary runs.

that each network (independent of size) typically possesses the following for the Iris task:

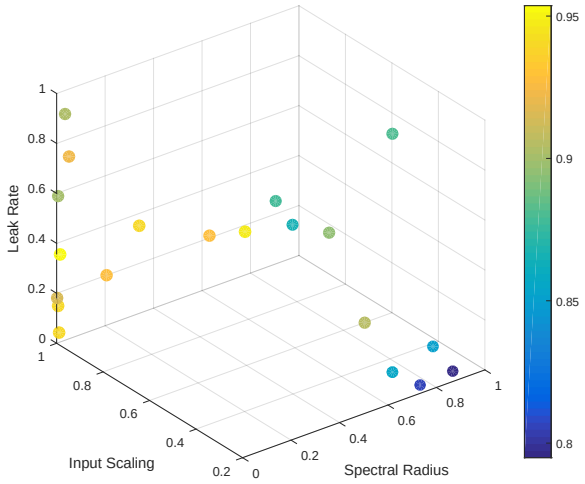
- Low spectral radius; i.e. does not require longer memory – output depends upon current input – and the reservoir is stable.
- Large variation in leak rates; suggesting the parameter does not significantly effect performance – as expected for a non-temporal task.
- Input scaling ≤ 1; neurons are not saturated and sway towards linear dynamics.

These evolved dynamics appear to match the known electrical properties of our materials (both stable and slightly non-linear I-V characteristics). This therefore suggests why the reservoirs perform similarly both *in materio* and *in silico*.

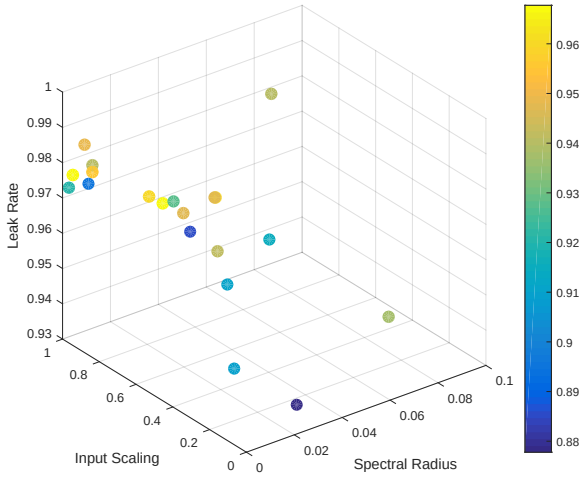
Mapping the dynamical and macroscopic relationships between different architectural reservoir systems can be enlightening, not only to benchmark the systems but to help identify what electrical properties of the material respond well to reservoir-style training. However, identifying exactly what properties in the material are being exploited by evolution is challenging. As discussed in [5], analysing and modelling the nanotube structures and electrical pathways being used is difficult. In Fig.5 we introduce a simple visualisation that highlights areas of interesting activity typically exploited by evolution. The figure displays the electrode arrangement for each material, showing what frequency an electrode is selected as an input (circle size) and what average voltage value is supplied (circle colour) across different evolutionary runs.



(a) 7 node network

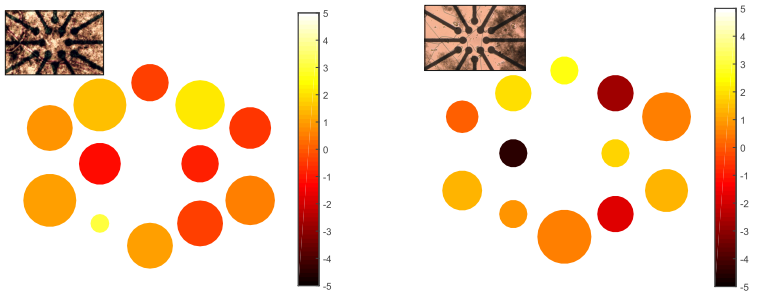


(b) 50 node network

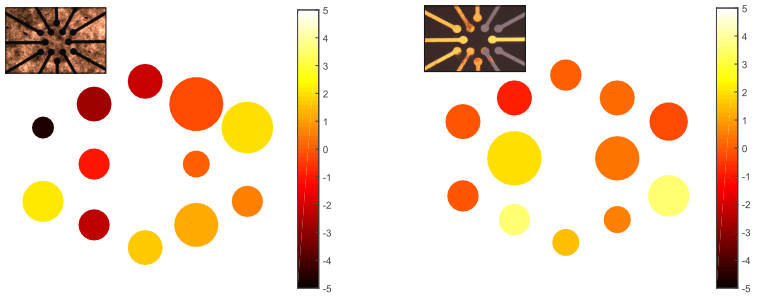


(c) 50 node (sampled) network

Fig. 4: Each plot displays 20 evolved Echo State Networks with network sizes of; (a) 7 nodes, (b) 50 nodes, and (c) 50 nodes with only 7 nodes being used to train/form the output. The parameters under evolution are input scaling (x axis), leak rate (y axis) and spectral radius (z axis). The colour map shows test accuracy of each evolved parameter set.



(a) 0.53% PMMA (b) 0.71% PMMA(1)



(c) 0.71% PMMA(2) (d) 1% PBMA

Fig. 5: Evolved circular electrode plot: Each plot shows how frequent an electrode is selected as an input (circle size) for each material, and the average input weight (circle colour), i.e. average voltage intensity, converged upon through evolution.

8 Conclusion

The results of this investigation are intriguing as they show that adding the reservoir framework – which enables the full exploitation of temporal information and some level of general-purpose functionality – there is no degradation in performance relative to the *vanilla* technique. We demonstrated this across several materials, including a control (a resistor array) and the same material used in the comparison work. This initial experiment shows that reservoir computing *in materio* can outperform the evolution *in materio* technique and match the performance of *in silico* techniques.

The final section of this work presented visualisation tools to aid understanding of the required task dynamics. These tools can be used to identify suitable tasks for the material, and identify regions of interesting activity/conductivity in the material often exploited by evolution.

Acknowledgements. Matthew Dale is funded by a Defence Science and Technology Laboratory (DSTL) PhD studentship. The authors thank the EU NASCENCE Project for providing the SWCNT materials used in this work.

References

1. A. Adamatzky, L. Bull, and B. D. L. Costello. *Unconventional computing 2007*. Luniver Press, 2007.
2. S. Bose, C. Lawrence, Z. Liu, K. Makarenko, R. van Damme, H. Broersma, and W. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nature nanotechnology*, doi:10.1038/nnano.2015.207, 2015.
3. H. Broersma, J. F. Miller, and S. Nichele. Computational matter: Evolving computational functions in nanoscale materials. In A. Adamatzky, editor, *Advances in Unconventional Computing*, volume 2, pages 397–428. Springer, 2017.
4. K. D. Clegg, J. F. Miller, M. K. Massey, and M. C. Petty. Practical issues for configuring carbon nanotube composite materials for computation. In *ICES 2014, IEEE International Conference on Evolvable Systems*, pages 61–68. IEEE, 2014.
5. M. Dale, J. F. Miller, and S. Stepney. Reservoir computing as a model for in-materio computing. In A. Adamatzky, editor, *Advances in Unconventional Computing*, volume 1, pages 533–571. Springer, 2017.
6. M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. Evolving carbon nanotube reservoir computers. In *Unconventional Computation and Natural Computation: 15th International Conference, UCNC 2016, Manchester, UK, July 11-15, 2016*, pages 49–61. Springer International Publishing, 2016.
7. M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. Reservoir computing in materio: An evaluation of configuration through evolution. In *IEEE International Conference on Evolvable Systems (ICES), (submitted)*. IEEE, 2016.
8. H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.
9. M. Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.
10. M. Lukoševičius, H. Jaeger, and B. Schrauwen. Reservoir computing trends. *KI-Künstliche Intelligenz*, 26(4):365–371, 2012.
11. W. Maass. Liquid state machines: motivation, theory, and applications. *Computability in context: computation and logic in the real world*, pages 275–296, 2010.
12. M. Massey, A. Kotsialos, F. Qaiser, D. Zeze, C. Pearson, D. Volpati, L. Bowen, and M. Petty. Computing with carbon nanotubes: Optimization of threshold logic gates using disordered nanotube/polymer composites. *Journal of Applied Physics*, 117(13):134903, 2015.
13. M. Massey, A. Kotsialos, D. Volpati, E. Vissol-Gaudin, C. Pearson, L. Bowen, B. Obara, D. Zeze, C. Groves, and M. Petty. Evolution of electronic circuits using carbon nanotube composites. *Scientific Reports*, 6, 2016.
14. J. F. Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In *NASA/DoD Conference on Evolvable Hardware 2002*, pages 167–176. IEEE, 2002.
15. M. Mohid, J. F. Miller, S. Harding, G. Tufte, O. R. Lykkebø, M. K. Massey, and M. C. Petty. Evolution-in-materio: Solving machine learning classification problems using materials. In *PPSN XIII, Parallel Problem Solving from Nature*, pages 721–730. Springer, 2014.
16. B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European symposium on artificial neural networks*. Citeseer, 2007.
17. S. Stepney. The neglected pillar of material computation. *Physica D: Nonlinear Phenomena*, 237(9):1157–1164, 2008.