AI

MDPI

Article

Abstract Reservoir Computing

Christoph Walter Senn [1,2,*,†] and Itsuo Kumazawa [2,†]

[1] Institute of Applied Mathematics and Physics, School of Engineering, Zurich University of Applied Sciences, 8401 Winterthur, Switzerland
[2] Laboratory for Future Interdisciplinary Research of Science and Technology, Institute of Innovative Research, Tokyo Institute of Technology, Tokyo 152-8550, Japan; kumazawa.i.aa@m.titech.ac.jp
* Correspondence: christoph.senn2@zhaw.ch
† These authors contributed equally to this work.

Abstract: Noise of any kind can be an issue when translating results from simulations to the real world. We suddenly have to deal with building tolerances, faulty sensors, or just noisy sensor readings. This is especially evident in systems with many free parameters, such as the ones used in physical reservoir computing. By abstracting away these kinds of noise sources using intervals, we derive a regularized training regime for reservoir computing using sets of possible reservoir states. Numerical simulations are used to show the effectiveness of our approach against different sources of errors that can appear in real-world scenarios and compare them with standard approaches. Our results support the application of interval arithmetics to improve the robustness of mass-spring networks trained in simulations.

Keywords: reservoir computing; echo state networks; recurrent neural networks; simulation; robustness

Citation: Senn, C.W.; Kumazawa, I. Abstract Reservoir Computing. *AI* **2022**, *3*, 194–210. https://doi.org/10.3390/ai3010012

Academic Editor: Agostino Forestiero

Received: 31 January 2022
Accepted: 5 March 2022
Published: 10 March 2022

1. Introduction

In recent years, physical reservoir computing has enjoyed increased popularity in a wide variety of fields. Since the emergence of reservoir computing as a computational framework for RNNs in the form of echo state networks [1] and liquid state machines [2], it has since transcended the digital world, and has found various applications in physical systems. This is due to the nature of the fixed reservoir, which allows the usage of other dynamical systems, given that they exhibit certain properties, as reservoirs. Over the years, reservoir computing systems have used buckets of water [3], light [4,5], and soft robots [6,7]. This development has given rise to the field of physical reservoir computing. Recent advances in this field, like the Origami reservoir [8] or mass-spring networks [9], have extensively used numerical simulations as part of their research.

Depending on the application, numerical simulations are inevitable; think of systems that work in difficult-to-access environments, where a loss can be potentially hazardous or expensive, or where building is time-consuming and costly. Although such numerical simulations have improved in fidelity, it is not possible to accurately represent all facets of physical systems in simulated environments. This, in part, leads to a gap between simulations and reality, also called the sim2real gap.

In addition to errors caused by differences in fidelity, hardware issues are also a source of concern. Sensors might break, misbehave, or become susceptible to noise. Accounting for all possible sources of errors is difficult and time-consuming, but necessary to create reliable and safe systems. Fortunately, the field of formal verification for software systems gives us tools to handle the complexity of such situations.

One of these tools, abstract interpretation [10], helps in dealing with such uncertainties, by enveloping them in abstract objects. This idea has also recently found its way to the neural network community for verification [11–13], but also for training [13,14]. Abstracting single data-points with sets of points, e.g., by encompassing them in a hypercube, as

illustrated in Figure 1, it becomes possible to work with the complete neighbourhood of such data-points, without having to sample all points that lie around it.

By using simple interval arithmetic [15], Senn and Kumazawa [16] have shown how this idea can be leveraged to train robust echo state networks. We build on this approach and show how it could be applied to create robust physical reservoir computing systems in simulations. Our contributions are:

- An abstract regulariser leading to robust weights for reservoir computing systems
- A closed form solution for the regression problem, using the abstract regulariser
- Numerical study on the robustness of physical reservoir computing systems against different types of errors

Physical reservoir computing in the form of mass-spring networks and how we use abstraction to improve the robustness of such systems is introduced in Section 2; furthermore, the datasets and types of errors considered, along with the general experimental setup, are shown. Then, the achieved results are introduced and discussed in Section 3. Finally, in Section 4, we give a conclusion and an outlook for future research endeavours.

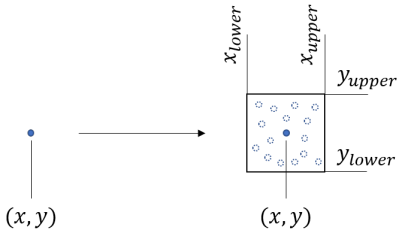


Figure 1. Visualisation of the abstraction of a single data-point to a set of points, representing a hypercube. The solid point represents the concrete point, and the dotted ones represent points that are now also considered using the abstraction.

2. Materials and Methods

In this section, we provide a brief introduction to reservoir computing (see Section 2.1), a physical implementation based on mass-spring networks (see Section 2.2), and how we can improve the robustness of such systems against noise introduced by the sim2real gap (see Section 2.3).

2.1. Reservoir Computing

Reservoir computing revolves around the exploitation of dynamical systems, called reservoirs, for computation. In terms of machine learning, we can divide the whole approach into two phases: the training (see Section 2.1.1) phase, in which teacher forcing is employed, and an exploitation phase (see Section 2.1.2), in which we use the dynamical system for our computations (e.g., predictions). Following this, we will provide a brief introduction to how the training and exploitation in reservoir computing works, by using a nonlinear map of the form:

$$x_{t+1} = \varphi(Ax_t + Bu_t), \tag{1}$$

with column vectors $x_t \in \mathbf{R}^d$ and $u_t \in \mathbf{R}^f$; matrices $A \in \mathbf{R}^{dxd}$ and $B \in \mathbf{R}^{dxf}$; a nonlinear function $\varphi : \mathbf{R}^d \to \mathbf{R}^d$ (e.g., the hyperbolic tangent); and time-steps $t = 1 \dots T$. In addition, we enforced the spectral radius of A to be $\rho(A) < 1$; this allowed the system to exhibit some kind of short-term memory. The rationale behind using a map as an example is that neural network or physics simulation-based approaches to reservoir computing can be reduced to discrete dynamical systems of this form.

2.1.1. Training

The training of reservoir computing systems was done in a supervised fashion; as such, we needed next the input signal u, a target signal y as ground truth. We further defined a washout time $0 \leq \tau < T$, which is the time that the system needs until its state

x is entirely dependent on the input u. Using Equation (1), we then drove the dynamical system using the input signal u, and collected the state x_t as row vectors for each time-step $t > \tau$ into a matrix. To conclude the training, the following equation was solved for the column vector $w \in R^d$:

$$y = \begin{pmatrix} x_{\tau+1} \\ x_{\tau+2} \\ \vdots \\ x_T \end{pmatrix} w, \tag{2}$$

this is usually done using linear regression techniques.

2.1.2. Exploitation

Once we have calculated our output weights w, we calculated an output $\hat{y}_t$ with:

$$\hat{y}_t = w^T x_t, \tag{3}$$

depending on the application, two ways of driving a reservoir computing system are possible. Either in an open-loop of the form:

$$\begin{aligned} x_{t+1} &= \varphi(Ax_t + Bu_t) \\ \hat{y}_{t+1} &= w^T x_{t+1}, \end{aligned} \tag{4}$$

or in a closed-loop:

$$\begin{aligned} x_{t+1} &= \varphi(Ax_t + B\hat{y}_t) \\ \hat{y}_{t+1} &= w^T x_{t+1}. \end{aligned} \tag{5}$$

As can be seen in Equations (4) and (5), the difference between a closed- and open-loop setup is how the input is generated. In the closed-loop setup, outputs were reused as inputs in the next time-step.

2.2. Mass-Spring Networks

Mass-spring networks are, in principle, coupled nonlinear oscillators that have been popularised by Hauser et al. [9], but also have been proposed by Couloumbe et al. [17]. Such systems can be used to approximate a variety of materials in simulations, like fabric [18]; compliant materials, as used in soft-robotics [19], or flesh-like setups [20].

We use a mass-spring system as shown in Figure 2, with nonlinear springs exhibiting a spring force of the following form:

$$f(\Delta l) = tanh(-k\Delta l), \tag{6}$$

Δl being the spring displacement and k the spring constant. This emulates a compliant, elastic material with a force displacement curve, as shown in Figure 3.

To exploit such a system for computation, an input signal u is translated to a force f and applied to predetermined input masses (green in Figure 2). The network then starts to oscillate accordingly, and we can record the mass accelerations as the state x (cf. Equation (2)). The recorded states can then be used for training and exploitation, as described in Sections 2.1.1 and 2.1.2.

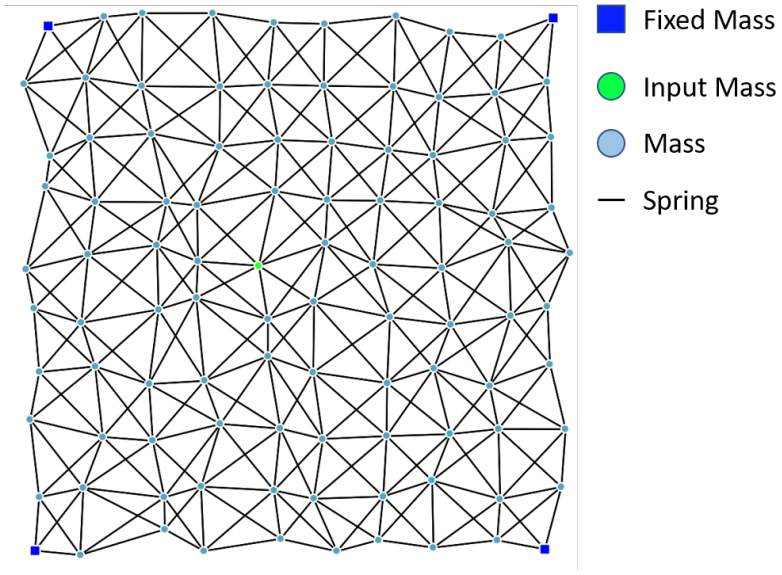


Figure 2. Visualisation of a mass-spring system as used in our experiment. The dark blue squares on the corners represent fixed masses (to fixate the network in space); the green circle in the middle is the input mass (where force is applied as input), the light blue circles are masses, and the black lines represent the nonlinear springs, connecting masses with each other.

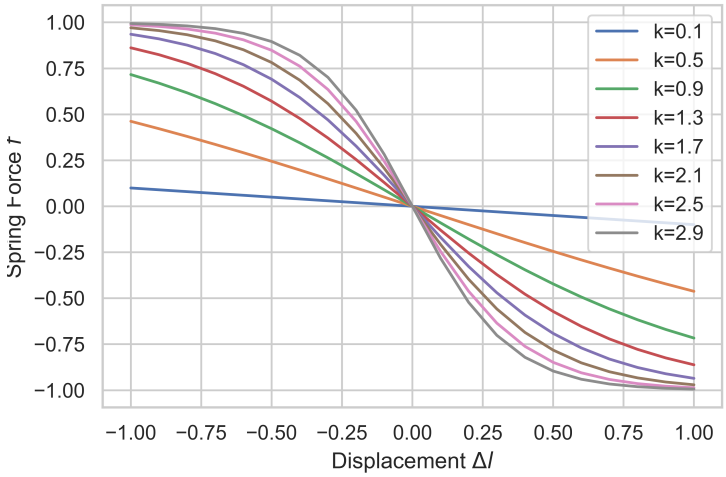


Figure 3. The force displacement curve of the springs used in the simulation at different spring constants k.

2.3. Abstract Reservoir Computing

When physically building mass-spring systems, as introduced in Section 2.2, we have to deal with tolerances due to imperfections in the creation process. Therefore, the initial position of masses diverted from the positions was assumed in the simulation; this is visualized in Figure 4. As a first step to deal with this problem, we can replace each component p_i of the location vector p of a mass, with an interval or ball of the form $(p_{i,centre}, p_{i,radius})$, representing the possible positions of the mass (red rectangle in Figure 4). We call this a hyperrectangle or, in abstract interpretation terminology, a box. This abstraction was then directly used in the simulation using ball arithmetic [21,22]. Instead of concrete numbers for our states x_t, we then got state tuples of the form $(x_{t,centre}, x_{t,radius})$, which we collected in the matrices X_c and X_r, respectively. Senn and Kumazawa [16] proposed to use the additional information as constraints for the linear regression and use a splitting conic solver [23] to solve:

$$\begin{aligned} &\underset{w}{argmin} \|X_c w - y_c\| \\ &s.t.\ |w| X_r \leq y_r. \end{aligned} \tag{7}$$

This approach has the advantage of having exact upper error bounds encoded in y_{radius}, which represent the maximum desired deviation from the concrete solution given as y_{centre}, but using a solver, slows down the training significantly. By relaxing the requirement for an upper error bound, we can reformulate Equation (7) as follows:

$$\underset{w}{argmin} \|X_c w - y_c\|^2 + \|X_r w\|^2, \tag{8}$$

Reformulating this as the cost function L allows us to derive a closed form solution, as shown in Equations (9) and (10):

$$\begin{aligned} L &= (X_c w - y_c)^2 + (X_r w)^2 \\ \frac{dL}{dw} &= 2X_c^T X_c w - X_c^2 y_c + 2X_r^T X_r w' \end{aligned} \tag{9}$$

then, setting the derivative $\frac{dL}{dw}$ equal to 0, we can solve for w:

$$\begin{aligned} 0 &= 2X_c^T X_c w - X_c^2 y_c + 2X_r^T X_r w \\ w &= (X_c^T X_c + X_r^T X_r)^{-1} X_c^T y \end{aligned} \tag{10}$$

Using Equation (10), instead of solving Equation (7), we trade assurance of error bounds for a significant speed-up.

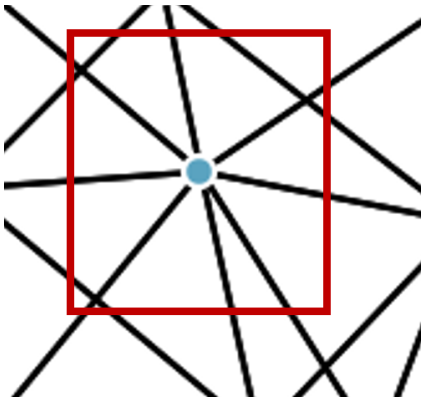


Figure 4. Visualisation of tolerances that can occur when physically building a mass-spring network. The light blue circle represents the exact location of the mass and the red rectangle represents the area of possible locations due to tolerances in horizontal and vertical directions. Such tolerances can change the dynamical behaviour of the network compared with simulated the counterpart, and poses a major obstacle when trying to transfer learnt parameters from simulation to real-world systems.

2.4. Experimental Setup

To evaluate our proposed approach, we implemented a numerical simulation of a mass-spring network using Julia 1.5 [24], and tested it with three datasets in open- and closed-loop setup for different types of error sources. The benchmark datasets are the same ones used by Goudarzi et al. [25], and were precomputed for 5000 time-steps, then each point in each time-series was repeated 5 times (giving time-series with 15,000 data points); finally, we split them into training and testing sets. The training sets consisted of the first 10,000 time-steps, whereas for the testing sets, the remaining 5000 time-steps were used.

2.4.1. Mass-Spring Network

We use a mass-spring network, as depicted in Figure 2. All masses are aligned to a regular grid first, and then slightly displaced with a value $\Delta p \in \mathcal{U}^2(-0.25, 0.25)$. Then, each mass is connected with its 8-neighbourhood through non-linear springs based on Equation (6). The four corner masses (blue) are fixed, and the input signal is applied as a force to a single input mass (green). As sensor readings, we use the acceleration of each mass in the network.

2.4.2. Hénon Time-Series

The Hénon time-series is based on the Hénon map introduced in 1976 [26]. Equation (11) was used to compute the time-series.

$$y_t = 1 - 1.4y_{t-1}^2 + 0.3y_{t-2} + \mathcal{N}(0, 0.001) \tag{11}$$

The Hénon time-series, as used in the experiments, is shown in Figure 5.

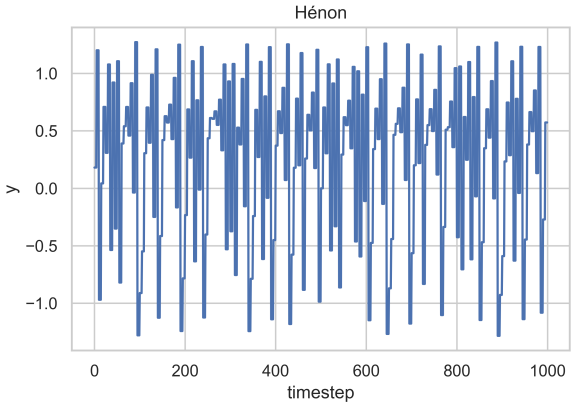


Figure 5. Hénon time-series as used in the experiments.

2.4.3. NARMA10 Time-Series

Non-linear autoregressive moving average (NARMA) tasks are widely used in the reservoir computing community as basic benchmarks. NARMA10 specifically is one of the most used benchmarks for reservoir computing and is defined as:

$$y_t = 0.3y_{t-1} + 0.05y_{t-1}\sum_{i=1}^{10} y_{t-i} + 1.5u_{t-10}u_{t-1} + 0.1 \tag{12}$$

with $u_t \in \mathcal{U}(0, 0.5)$ being drawn from a uniform distribution.

The NARMA10 time-series, as used in the experiments, is shown in Figure 6.

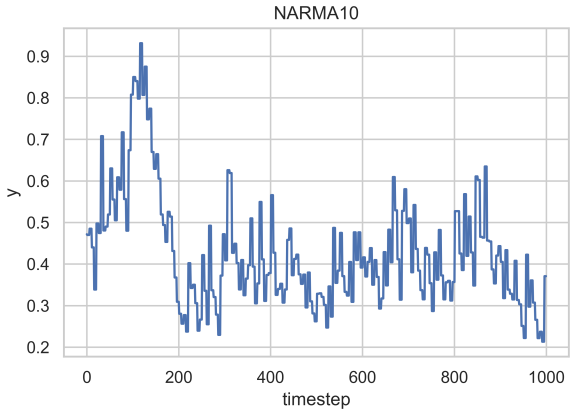


Figure 6. NARMA10 time-series as used in the experiments.

2.4.4. NARMA20 Time-Series

The NARMA20 task is the same as the NARMA10 one, except with longer time dependencies and an additional non-linearity:

$$y_t = tanh(0.3y_{t-1} + 0.05y_{t-1}\sum_{i=1}^{20} y_{t-i} + 1.5u_{t-20}u_{t-1} + 0.1) \tag{13}$$

The NARMA20 time-series, as used in the experiments, is shown in Figure 7.

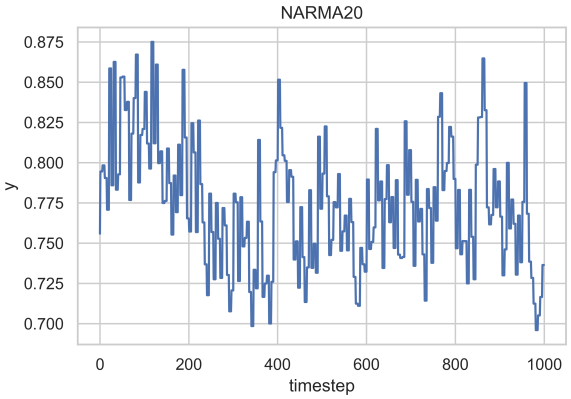


Figure 7. NARMA20 time-series as used in the experiments.

2.4.5. Baselines

We compare the results of our proposed approach to the following two baselines:

- Training with ridge regression (classical model)
- Training with linear regression and added noise (noise model)

Using the sensor readings $x_{\tau+1}, \ldots, x_T$ from the training simulation, the classical model is trained using Equation (14), and the noise model's training is based on Equation (15), with ϵ set to the amplitude of the current error parameter.

$$X = \begin{pmatrix} x_{\tau+1} \\ x_{\tau+2} \\ \vdots \\ x_T \end{pmatrix}$$
$$w = (X^T X + 0.001I)^{-1} X^T y \tag{14}$$

$$X = \begin{pmatrix} x_{\tau+1} \\ x_{\tau+2} \\ \vdots \\ x_T \end{pmatrix}$$
$$w = (X + \mathcal{U}(-\epsilon, \epsilon))^{\dagger} y \tag{15}$$

2.4.6. Sensor Augmentations

We tested each model under different types of errors that could potentially occur in a real-world scenario.

- Sensor Failure
  - Before testing, masses were randomly selected with a given probability p, and their readings were forced to 0 during testing.
- Sensor Noise
  - Gaussian noise with 0-mean and varying standard deviation σ is added during testing.
- Fixed Sensor Displacement
  - Sensor readings were displaced by a fixed value z.
- Mass Position Displacement
  - The mass positions were randomly displaced by a random vector $\Delta x \in [-k; k]^2$.

Each possible source of error was tested independently of other error sources. The parameter ranges are shown in Table 1.

Table 1. The parameter ranges used for each type of simulated error.

Augmentation	Parameter	Range
Sensor Failure	p	$[0, 0.1]$
Sensor Noise	σ	$[0, 0.1]$
Fixed Sensor Displacement	z	$[0, 0.1]$
Mass Position Displacement	k	$[0, 0.1]$

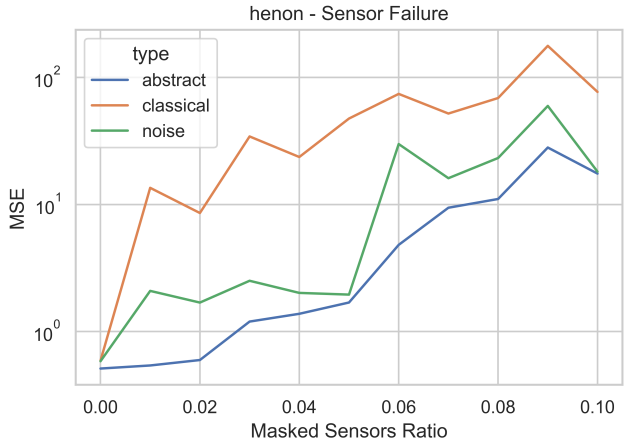
3. Results and Discussion

For each experiment, we measured the mean squared error (MSE), given by Equation (16), between the generated outputs $\hat{y}$ by the system and the values y in test sets. As can be seen in Figures 8–11, our proposed approach scores better regarding the MSE compared to the classical approach in all experiments. Compared to the training regime with noise, at high enough noise levels, it takes over or performs equally to our proposed training regime. Exact numbers can be found in the Appendix A in Tables A1–A4.

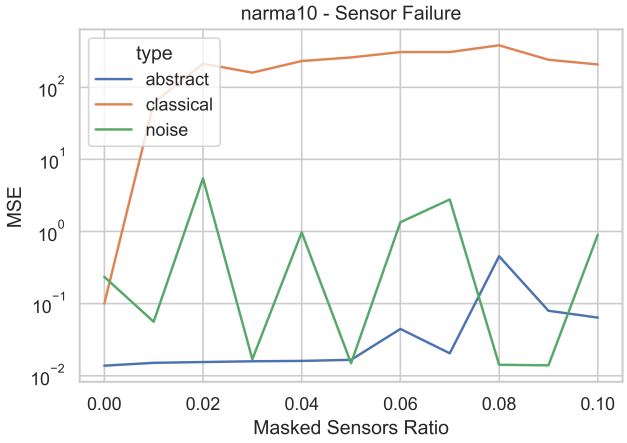
$$MSE = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2 \tag{16}$$

When looking at the results when no noise is present in the input, our proposal surpasses both baselines, in any case. This can indicate that the abstract regulariser is generally a better choice than $L2$-regularisation. Considering the results of the experiments with noise present, our proposed approach gives more consistent results over the whole spectrum of noise amplitudes, except for sensor failures, i.e., missing sensor readings. In this case, both baselines also perform poorly.

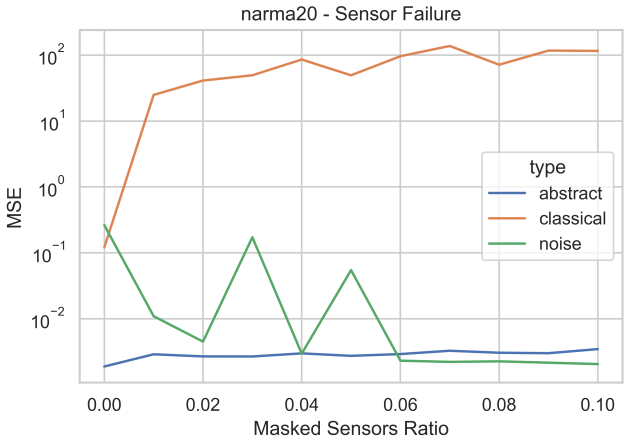
One reason for the better performance, even without noise present, is the fact that ball arithmetic [21], as used in the experiments, also captures numerical imprecisions. This can be compared to adding noise to the system, and thus help against overfitting. Looking at the standard deviations of the results (Tables A1–A4), we see that the abstract training regime leads to fewer variations of the output, indicating that our approach is more robust against randomness.



(**a**) MSE for simulated sensor failures with probability given by the x-axis for the Hénon dataset.

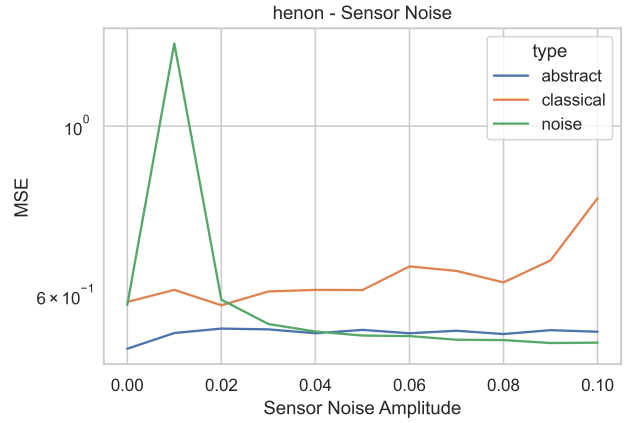


(**b**) MSE for simulated sensor failures with probability given by the x-axis for the NARMA10 dataset.

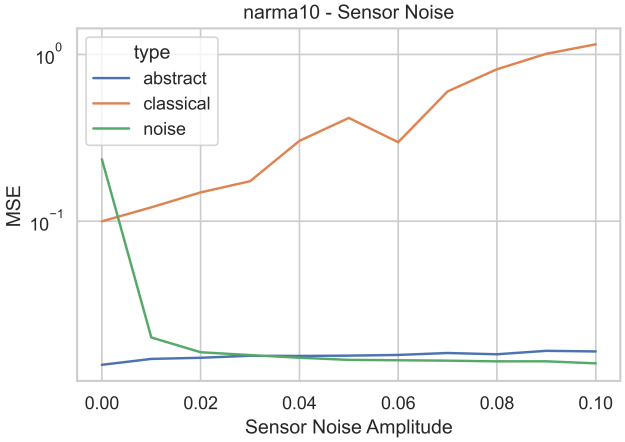


(**c**) MSE for simulated sensor failures with probability given by the x-axis for the NARMA20 dataset.

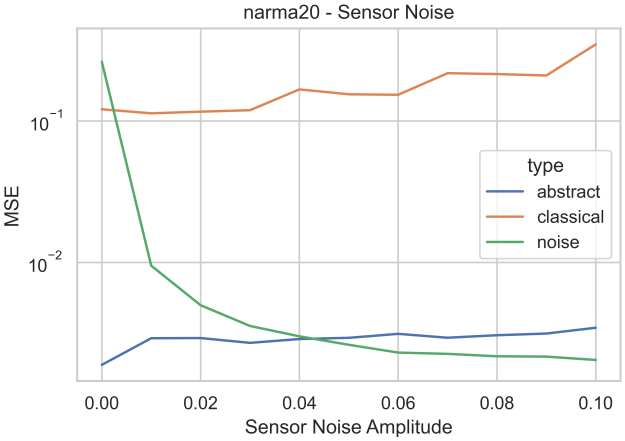
Figure 8. The average MSE with standard deviations for simulated sensor failures for the datasets (**a**) Hénon, (**b**) NARMA10 and (**c**) NARMA20 .



(**a**) MSE for simulated sensor noise with the amplitude given by the x-axis for the Hénon dataset.



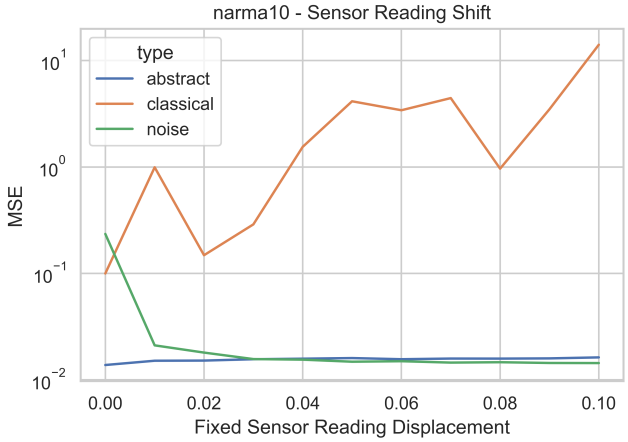
(**b**) MSE for simulated sensor noise with the amplitude given by the x-axis for the NARMA10 dataset.



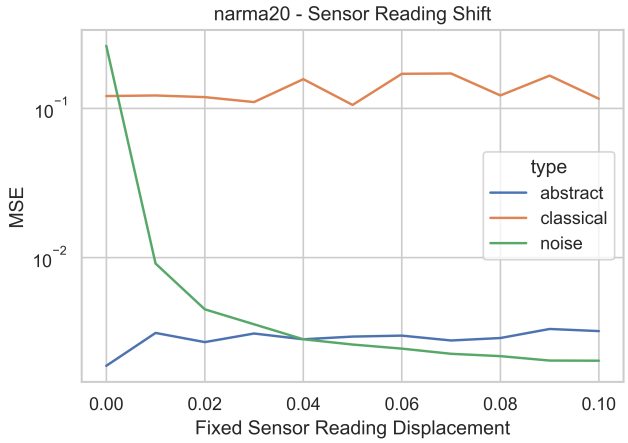
(**c**) MSE for simulated sensor noise with the amplitude given by the x-axis for the NARMA20 dataset.

Figure 9. The average MSE with standard deviations for simulated sensor noise for the datasets (**a**) Hénon, (**b**) NARMA10 and (**c**) NARMA20 .

(**a**) MSE for fixed simulated sensor perturbation with the amplitude given by the x-axis for the Hénon dataset.

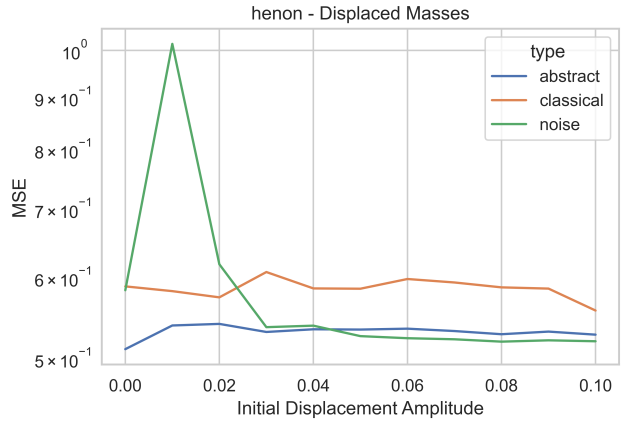


(**b**) MSE for fixed simulated sensor perturbation with the amplitude given by the x-axis for the NARMA10 dataset.

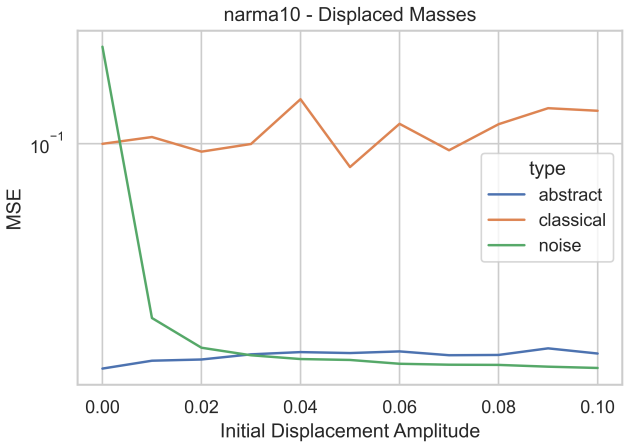


(**c**) MSE for fixed simulated sensor perturbation with the amplitude given by the x-axis for the NARMA20 dataset.

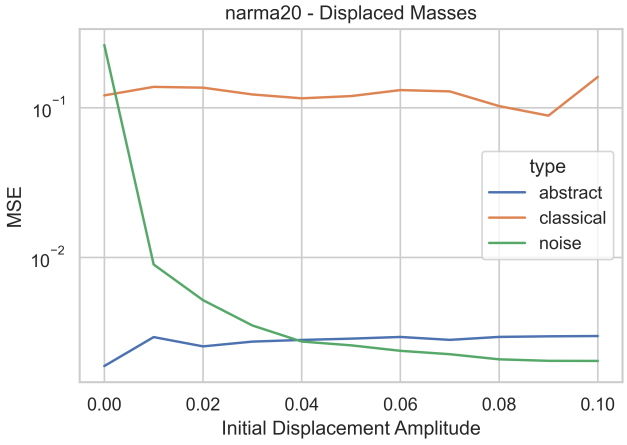
Figure 10. The average MSE with standard deviations for fixed simulated sensor reading displacements for the datasets (**a**) Hénon, (**b**) NARMA10 and (**c**) NARMA20 .



(**a**) MSE for simulated initial mass displacement with the amplitude given by the *x*-axis for the Hénon dataset.



(**b**) MSE for simulated initial mass displacement with the amplitude given by the *x*-axis for the NARMA10 dataset.



(**c**) MSE for simulated initial mass displacement with the amplitude given by the *x*-axis for the NARMA20 dataset.

Figure 11. The average MSE with standard deviations for simulated initial mass displacements for the datasets (**a**) Hénon, (**b**) NARMA10 and (**c**) NARMA20 .

4. Conclusions and Outlook

Although physical reservoir computing systems are becoming more and more important, and most experiments are based on numerical simulations, the direct transfer of trained systems from simulation to the real world has not been widely studied yet. We proposed a new training regime based on abstract interpretation to address some of the issues that are to be expected, such as building tolerances, sensor defects, noise, etc., and verified our approach in a series of simulated experiments. The results support the use of our abstract regulariser, not only in this setting, but also in general, as it achieved lower error rates. This is most likely due to the interval arithmetic used in our experiments. When adding noise, the difference to the classical approach with $L2$-regularisation becomes even more significant. In contrast, the training regime with added noise becomes better the more noise is added. Therefore, in settings where high noise amplitudes are to be expected, this approach seems to be the better choice.

For future works, we see a direct comparison between a physical reservoir computing system in simulations and real-life, e.g., based on fabrics, as this comes close to the mass-spring systems used in the experiments. Another direction could be the nature of noise captured by the abstraction. Currently, we only consider uniformly distributed abstractions—but most problems in nature are differently distributed, e.g., are from a normal distribution. Being able to use different distributions for the abstractions would give the ability to tailor the training regime specifically to the problems at hand.

In either case, both directions will allow for a more widespread use of physical reservoir computing, as it would allow for rapid iterations in simulations and a direct transfer of results to the real world.

Author Contributions: Conceptualization, C.W.S.; methodology, C.W.S.; software, C.W.S.; validation, C.W.S.; formal analysis, C.W.S.; investigation, C.W.S.; resources, C.W.S.; data curation, C.W.S.; writing—original draft preparation, C.W.S.; writing—review and editing, C.W.S. and I.K.; visualisation, C.W.S.; supervision, I.K.; funding acquisition, C.W.S. All authors have read and agreed to the published version of the manuscript.

Funding: Open access funding provided by ZHAW Zurich University of Applied Sciences.

Acknowledgments: The authors thank Carmen Mei Ling Frischknecht-Gruber of the Zurich University of Applied Sciences for her comments and proofreading while creating this manuscript.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. Results

Tables A1–A4 show additional statistics averaged over 10 runs for each experiment conducted.

Table A1. Average MSE for the experiments simulating failing sensors over 10 experiment runs. The parameter gives the fraction of sensors that were forced to 0.

Dataset	Model	Sensor Failure MSE ± Std Dev@Parameter Value			
		0.00	0.01	0.02	0.03
Hénon	abstract	$5.11 \times 10^{-1} \pm 3.10 \times 10^{-4}$	$5.40 \times 10^{-1} \pm 8.56 \times 10^{-3}$	$5.96 \times 10^{-1} \pm 1.24 \times 10^{-1}$	1.20 ± 1.35
Hénon	classical	$5.88 \times 10^{-1} \pm 3.17 \times 10^{-2}$	$1.35 \times 10^{1} \pm 2.69 \times 10^{1}$	$8.57 \pm 1.54 \times 10^{1}$	$3.43 \times 10^{1} \pm 4.09 \times 10^{1}$
Hénon	noise	$5.83 \times 10^{-1} \pm 2.38 \times 10^{-2}$	2.09 ± 2.74	1.69 ± 3.13	2.51 ± 3.56
NARMA10	abstract	$1.38 \times 10^{-2} \pm 3.88 \times 10^{-5}$	$1.51 \times 10^{-2} \pm 8.40 \times 10^{-4}$	$1.55 \times 10^{-2} \pm 4.69 \times 10^{-4}$	$1.59 \times 10^{-2} \pm 1.66 \times 10^{-3}$
NARMA10	classical	$9.99 \times 10^{-2} \pm 4.08 \times 10^{-2}$	$6.08 \times 10^{1} \pm 1.14 \times 10^{2}$	$2.13 \times 10^{2} \pm 2.20 \times 10^{2}$	$1.60 \times 10^{2} \pm 1.61 \times 10^{2}$
NARMA10	noise	$2.35 \times 10^{-1} \pm 1.19 \times 10^{-1}$	$5.60 \times 10^{-2} \pm 7.01 \times 10^{-2}$	$5.46 \pm 1.12 \times 10^{1}$	$1.69 \times 10^{-2} \pm 3.29 \times 10^{-3}$
NARMA20	abstract	$1.88 \times 10^{-3} \pm 3.67 \times 10^{-5}$	$2.88 \times 10^{-3} \pm 3.20 \times 10^{-4}$	$2.68 \times 10^{-3} \pm 1.95 \times 10^{-4}$	$2.67 \times 10^{-3} \pm 3.33 \times 10^{-4}$
NARMA20	classical	$1.21 \times 10^{-1} \pm 6.16 \times 10^{-2}$	$2.49 \times 10^{1} \pm 3.76 \times 10^{1}$	$4.11 \times 10^{1} \pm 4.70 \times 10^{1}$	$4.95 \times 10^{1} \pm 9.20 \times 10^{1}$
NARMA20	noise	$2.63 \times 10^{-1} \pm 1.43 \times 10^{-1}$	$1.09 \times 10^{-2} \pm 4.58 \times 10^{-3}$	$4.49 \times 10^{-3} \pm 7.36 \times 10^{-4}$	$1.72 \times 10^{-1} \pm 5.05 \times 10^{-1}$

Table A1. *Cont.*

Dataset	Model	Sensor Failure MSE ± Std Dev@Parameter Value			
		0.04	0.05	0.06	0.07
Hénon	abstract	1.38 ± 1.77	1.69 ± 2.36	4.81 ± 7.96	$9.43 \pm 1.33 \times 10^{1}$
Hénon	classical	$2.36 \times 10^{1} \pm 2.17 \times 10^{1}$	$4.75 \times 10^{1} \pm 5.20 \times 10^{1}$	$7.42 \times 10^{1} \pm 5.73 \times 10^{1}$	$5.20 \times 10^{1} \pm 6.11 \times 10^{1}$
Hénon	noise	2.01 ± 4.35	1.95 ± 3.86	$2.99 \times 10^{1} \pm 8.65 \times 10^{1}$	$1.61 \times 10^{1} \pm 4.46 \times 10^{1}$
NARMA10	abstract	$1.61 \times 10^{-2} \pm 1.20 \times 10^{-3}$	$1.66 \times 10^{-2} \pm 1.82 \times 10^{-3}$	$4.46 \times 10^{-2} \pm 7.64 \times 10^{-2}$	$2.05 \times 10^{-2} \pm 4.45 \times 10^{-3}$
NARMA10	classical	$2.31 \times 10^{2} \pm 1.64 \times 10^{2}$	$2.59 \times 10^{2} \pm 1.51 \times 10^{2}$	$3.07 \times 10^{2} \pm 2.39 \times 10^{2}$	$3.08 \times 10^{2} \pm 2.19 \times 10^{2}$
NARMA10	noise	$9.72 \times 10^{-1} \pm 2.87$	$1.49 \times 10^{-2} \pm 8.54 \times 10^{-4}$	1.34 ± 3.99	2.78 ± 4.31
NARMA20	abstract	$2.98 \times 10^{-3} \pm 3.64 \times 10^{-4}$	$2.73 \times 10^{-3} \pm 3.37 \times 10^{-4}$	$2.91 \times 10^{-3} \pm 4.38 \times 10^{-4}$	$3.26 \times 10^{-3} \pm 5.69 \times 10^{-4}$
NARMA20	classical	$8.58 \times 10^{1} \pm 7.37 \times 10^{1}$	$4.94 \times 10^{1} \pm 4.46 \times 10^{1}$	$9.63 \times 10^{1} \pm 5.72 \times 10^{1}$	$1.38 \times 10^{2} \pm 1.25 \times 10^{2}$
NARMA20	noise	$2.95 \times 10^{-3} \pm 4.11 \times 10^{-4}$	$5.46 \times 10^{-2} \pm 1.55 \times 10^{-1}$	$2.31 \times 10^{-3} \pm 1.37 \times 10^{-4}$	$2.21 \times 10^{-3} \pm 1.08 \times 10^{-4}$
Dataset	**Model**	**Sensor Failure MSE ± Std Dev@Parameter Value**			
		0.08	0.09	0.10	
Hénon	abstract	$1.10 \times 10^{1} \pm 1.91 \times 10^{1}$	$2.81 \times 10^{1} \pm 3.49 \times 10^{1}$	$1.75 \times 10^{1} \pm 3.93 \times 10^{1}$	
Hénon	classical	$6.89 \times 10^{1} \pm 1.30 \times 10^{2}$	$1.77 \times 10^{2} \pm 2.03 \times 10^{2}$	$7.70 \times 10^{1} \pm 1.06 \times 10^{2}$	
Hénon	noise	$2.32 \times 10^{1} \pm 6.62 \times 10^{1}$	$5.97 \times 10^{1} \pm 7.35 \times 10^{1}$	$1.81 \times 10^{1} \pm 5.20 \times 10^{1}$	
NARMA10	abstract	$4.53 \times 10^{-1} \pm 1.26$	$7.96 \times 10^{-2} \pm 1.77 \times 10^{-1}$	$6.39 \times 10^{-2} \pm 5.86 \times 10^{-2}$	
NARMA10	classical	$3.82 \times 10^{2} \pm 1.75 \times 10^{2}$	$2.41 \times 10^{2} \pm 1.04 \times 10^{2}$	$2.07 \times 10^{2} \pm 1.66 \times 10^{2}$	
NARMA10	noise	$1.42 \times 10^{-2} \pm 3.38 \times 10^{-4}$	$1.39 \times 10^{-2} \pm 5.19 \times 10^{-4}$	$9.00 \times 10^{-1} \pm 2.66$	
NARMA20	abstract	$3.04 \times 10^{-3} \pm 4.95 \times 10^{-4}$	$3.00 \times 10^{-3} \pm 3.61 \times 10^{-4}$	$3.46 \times 10^{-3} \pm 4.50 \times 10^{-4}$	
NARMA20	classical	$7.15 \times 10^{1} \pm 6.04 \times 10^{1}$	$1.17 \times 10^{2} \pm 9.58 \times 10^{1}$	$1.16 \times 10^{2} \pm 9.45 \times 10^{1}$	
NARMA20	noise	$2.25 \times 10^{-3} \pm 1.34 \times 10^{-4}$	$2.15 \times 10^{-3} \pm 2.44 \times 10^{-4}$	$2.05 \times 10^{-3} \pm 1.46 \times 10^{-4}$	

Table A2. Average MSE for the experiments simulating sensor noise over 10 experiment runs. The parameter gives the amplitude of the noise added to the sensor readings.

Dataset	Model	Sensor Noise MSE ± Std Dev@Parameter Value			
		0.00	0.01	0.02	0.03
Hénon	abstract	$5.11 \times 10^{-1} \pm 3.10 \times 10^{-4}$	$5.36 \times 10^{-1} \pm 6.54 \times 10^{-3}$	$5.43 \times 10^{-1} \pm 1.42 \times 10^{-2}$	$5.42 \times 10^{-1} \pm 1.42 \times 10^{-2}$
Hénon	classical	$5.88 \times 10^{-1} \pm 3.17 \times 10^{-2}$	$6.10 \times 10^{-1} \pm 5.59 \times 10^{-2}$	$5.82 \times 10^{-1} \pm 3.28 \times 10^{-2}$	$6.07 \times 10^{-1} \pm 3.15 \times 10^{-2}$
Hénon	noise	$5.83 \times 10^{-1} \pm 2.38 \times 10^{-2}$	1.28 ± 1.35	$5.92 \times 10^{-1} \pm 5.49 \times 10^{-2}$	$5.50 \times 10^{-1} \pm 3.63 \times 10^{-2}$
NARMA10	abstract	$1.38 \times 10^{-2} \pm 3.88 \times 10^{-5}$	$1.50 \times 10^{-2} \pm 5.35 \times 10^{-4}$	$1.52 \times 10^{-2} \pm 4.91 \times 10^{-4}$	$1.56 \times 10^{-2} \pm 6.17 \times 10^{-4}$
NARMA10	classical	$9.99 \times 10^{-2} \pm 4.08 \times 10^{-2}$	$1.21 \times 10^{-1} \pm 2.87 \times 10^{-2}$	$1.49 \times 10^{-1} \pm 8.47 \times 10^{-2}$	$1.74 \times 10^{-1} \pm 9.33 \times 10^{-2}$
NARMA10	noise	$2.35 \times 10^{-1} \pm 1.19 \times 10^{-1}$	$2.01 \times 10^{-2} \pm 1.09 \times 10^{-3}$	$1.64 \times 10^{-2} \pm 7.40 \times 10^{-4}$	$1.58 \times 10^{-2} \pm 5.47 \times 10^{-4}$
NARMA20	abstract	$1.88 \times 10^{-3} \pm 3.67 \times 10^{-5}$	$2.91 \times 10^{-3} \pm 3.86 \times 10^{-4}$	$2.91 \times 10^{-3} \pm 4.49 \times 10^{-4}$	$2.70 \times 10^{-3} \pm 2.75 \times 10^{-4}$
NARMA20	classical	$1.21 \times 10^{-1} \pm 6.16 \times 10^{-2}$	$1.13 \times 10^{-1} \pm 6.94 \times 10^{-2}$	$1.17 \times 10^{-1} \pm 4.06 \times 10^{-2}$	$1.19 \times 10^{-1} \pm 4.61 \times 10^{-2}$
NARMA20	noise	$2.63 \times 10^{-1} \pm 1.43 \times 10^{-1}$	$9.48 \times 10^{-3} \pm 2.21 \times 10^{-3}$	$4.97 \times 10^{-3} \pm 7.51 \times 10^{-4}$	$3.55 \times 10^{-3} \pm 6.72 \times 10^{-4}$
Dataset	**Model**	**Sensor Noise MSE ± Std Dev@Parameter Value**			
		0.04	0.05	0.06	0.07
Hénon	abstract	$5.35 \times 10^{-1} \pm 2.98 \times 10^{-3}$	$5.41 \times 10^{-1} \pm 1.04 \times 10^{-2}$	$5.35 \times 10^{-1} \pm 6.49 \times 10^{-3}$	$5.39 \times 10^{-1} \pm 6.83 \times 10^{-3}$
Hénon	classical	$6.10 \times 10^{-1} \pm 3.41 \times 10^{-2}$	$6.10 \times 10^{-1} \pm 3.51 \times 10^{-2}$	$6.55 \times 10^{-1} \pm 6.32 \times 10^{-2}$	$6.46 \times 10^{-1} \pm 8.54 \times 10^{-2}$
Hénon	noise	$5.38 \times 10^{-1} \pm 1.55 \times 10^{-2}$	$5.32 \times 10^{-1} \pm 1.34 \times 10^{-2}$	$5.31 \times 10^{-1} \pm 9.41 \times 10^{-3}$	$5.25 \times 10^{-1} \pm 1.00 \times 10^{-2}$
NARMA10	abstract	$1.56 \times 10^{-2} \pm 7.24 \times 10^{-4}$	$1.56 \times 10^{-2} \pm 7.65 \times 10^{-4}$	$1.58 \times 10^{-2} \pm 5.73 \times 10^{-4}$	$1.62 \times 10^{-2} \pm 9.39 \times 10^{-4}$
NARMA10	classical	$3.03 \times 10^{-1} \pm 2.94 \times 10^{-1}$	$4.16 \times 10^{-1} \pm 3.04 \times 10^{-1}$	$2.98 \times 10^{-1} \pm 2.25 \times 10^{-1}$	$6.00 \times 10^{-1} \pm 3.36 \times 10^{-1}$
NARMA10	noise	$1.52 \times 10^{-2} \pm 6.63 \times 10^{-4}$	$1.48 \times 10^{-2} \pm 6.42 \times 10^{-4}$	$1.47 \times 10^{-2} \pm 2.84 \times 10^{-4}$	$1.46 \times 10^{-2} \pm 4.15 \times 10^{-4}$
NARMA20	abstract	$2.87 \times 10^{-3} \pm 3.53 \times 10^{-4}$	$2.93 \times 10^{-3} \pm 3.31 \times 10^{-4}$	$3.12 \times 10^{-3} \pm 5.87 \times 10^{-4}$	$2.93 \times 10^{-3} \pm 3.55 \times 10^{-4}$
NARMA20	classical	$1.67 \times 10^{-1} \pm 1.07 \times 10^{-1}$	$1.55 \times 10^{-1} \pm 7.66 \times 10^{-2}$	$1.53 \times 10^{-1} \pm 6.87 \times 10^{-2}$	$2.18 \times 10^{-1} \pm 1.51 \times 10^{-1}$
NARMA20	noise	$3.00 \times 10^{-3} \pm 3.13 \times 10^{-4}$	$2.61 \times 10^{-3} \pm 2.11 \times 10^{-4}$	$2.30 \times 10^{-3} \pm 9.01 \times 10^{-5}$	$2.25 \times 10^{-3} \pm 1.21 \times 10^{-4}$
Dataset	**Model**	**Sensor Noise MSE ± Std Dev@Parameter Value**			
		0.08	0.09	0.10	
Hénon	abstract	$5.34 \times 10^{-1} \pm 8.08 \times 10^{-3}$	$5.40 \times 10^{-1} \pm 8.01 \times 10^{-3}$	$5.38 \times 10^{-1} \pm 7.52 \times 10^{-3}$	
Hénon	classical	$6.24 \times 10^{-1} \pm 4.45 \times 10^{-2}$	$6.67 \times 10^{-1} \pm 1.12 \times 10^{-1}$	$8.04 \times 10^{-1} \pm 2.75 \times 10^{-1}$	
Hénon	noise	$5.24 \times 10^{-1} \pm 3.89 \times 10^{-3}$	$5.20 \times 10^{-1} \pm 2.76 \times 10^{-3}$	$5.20 \times 10^{-1} \pm 3.61 \times 10^{-3}$	
NARMA10	abstract	$1.59 \times 10^{-2} \pm 7.09 \times 10^{-4}$	$1.67 \times 10^{-2} \pm 1.62 \times 10^{-3}$	$1.66 \times 10^{-2} \pm 7.81 \times 10^{-4}$	
NARMA10	classical	$8.15 \times 10^{-1} \pm 4.14 \times 10^{-1}$	$1.01 \pm 6.59 \times 10^{-1}$	$1.15 \pm 8.90 \times 10^{-1}$	
NARMA10	noise	$1.45 \times 10^{-2} \pm 4.38 \times 10^{-4}$	$1.45 \times 10^{-2} \pm 2.60 \times 10^{-4}$	$1.41 \times 10^{-2} \pm 2.49 \times 10^{-4}$	
NARMA20	abstract	$3.05 \times 10^{-3} \pm 7.32 \times 10^{-4}$	$3.14 \times 10^{-3} \pm 3.92 \times 10^{-4}$	$3.45 \times 10^{-3} \pm 2.25 \times 10^{-4}$	
NARMA20	classical	$2.15 \times 10^{-1} \pm 8.25 \times 10^{-2}$	$2.10 \times 10^{-1} \pm 1.12 \times 10^{-1}$	$3.49 \times 10^{-1} \pm 1.67 \times 10^{-1}$	
NARMA20	noise	$2.17 \times 10^{-3} \pm 7.56 \times 10^{-5}$	$2.15 \times 10^{-3} \pm 1.45 \times 10^{-4}$	$2.04 \times 10^{-3} \pm 1.05 \times 10^{-4}$	

Table A3. Average MSE for the experiments simulating a sensor reading shift over 10 experiment runs. The parameter gives the shift that was added to the sensor readings.

Dataset	Model	Sense or Shift MSE ± Std Dev@Parameter Value			
		0.00	0.01	0.02	0.03
Hénon	abstract	$5.11 \times 10^{-1} \pm 3.10 \times 10^{-4}$	$5.38 \times 10^{-1} \pm 1.40 \times 10^{-2}$	$5.41 \times 10^{-1} \pm 9.89 \times 10^{-3}$	$5.40 \times 10^{-1} \pm 1.22 \times 10^{-2}$
Hénon	classical	$5.88 \times 10^{-1} \pm 3.17 \times 10^{-2}$	$6.14 \times 10^{-1} \pm 8.34 \times 10^{-2}$	$5.93 \times 10^{-1} \pm 6.51 \times 10^{-2}$	2.87 ± 4.44
Hénon	noise	$5.83 \times 10^{-1} \pm 2.38 \times 10^{-2}$	$8.87 \times 10^{-1} \pm 8.11 \times 10^{-1}$	$5.71 \times 10^{-1} \pm 5.64 \times 10^{-2}$	$5.47 \times 10^{-1} \pm 3.61 \times 10^{-2}$
NARMA10	abstract	$1.38 \times 10^{-2} \pm 3.88 \times 10^{-5}$	$1.51 \times 10^{-2} \pm 5.03 \times 10^{-4}$	$1.52 \times 10^{-2} \pm 4.06 \times 10^{-4}$	$1.56 \times 10^{-2} \pm 9.01 \times 10^{-4}$
NARMA10	classical	$9.99 \times 10^{-2} \pm 4.08 \times 10^{-2}$	$9.94 \times 10^{-1} \pm 2.71$	$1.48 \times 10^{-1} \pm 1.51 \times 10^{-1}$	$2.89 \times 10^{-1} \pm 2.88 \times 10^{-1}$
NARMA10	noise	$2.35 \times 10^{-1} \pm 1.19 \times 10^{-1}$	$2.11 \times 10^{-2} \pm 5.82 \times 10^{-3}$	$1.80 \times 10^{-2} \pm 2.44 \times 10^{-3}$	$1.57 \times 10^{-2} \pm 6.76 \times 10^{-4}$
NARMA20	abstract	$1.88 \times 10^{-3} \pm 3.67 \times 10^{-5}$	$3.13 \times 10^{-3} \pm 3.54 \times 10^{-4}$	$2.72 \times 10^{-3} \pm 2.57 \times 10^{-4}$	$3.10 \times 10^{-3} \pm 6.85 \times 10^{-4}$
NARMA20	classical	$1.21 \times 10^{-1} \pm 6.16 \times 10^{-2}$	$1.22 \times 10^{-1} \pm 6.48 \times 10^{-2}$	$1.19 \times 10^{-1} \pm 5.79 \times 10^{-2}$	$1.10 \times 10^{-1} \pm 3.31 \times 10^{-2}$
NARMA20	noise	$2.63 \times 10^{-1} \pm 1.43 \times 10^{-1}$	$9.13 \times 10^{-3} \pm 1.64 \times 10^{-3}$	$4.50 \times 10^{-3} \pm 4.86 \times 10^{-4}$	$3.58 \times 10^{-3} \pm 5.00 \times 10^{-4}$
Dataset	**Model**	**Sensor Shift MSE ± Std Dev@Parameter Value**			
		0.04	0.05	0.06	0.07
Hénon	abstract	$5.35 \times 10^{-1} \pm 1.02 \times 10^{-2}$	$5.37 \times 10^{-1} \pm 8.50 \times 10^{-3}$	$5.39 \times 10^{-1} \pm 1.24 \times 10^{-2}$	$5.35 \times 10^{-1} \pm 5.82 \times 10^{-3}$
Hénon	classical	2.46 ± 5.26	$7.37 \times 10^{-1} \pm 2.69 \times 10^{-1}$	3.91 ± 6.66	1.93 ± 3.51
Hénon	noise	$5.35 \times 10^{-1} \pm 1.81 \times 10^{-2}$	$5.33 \times 10^{-1} \pm 1.14 \times 10^{-2}$	$5.28 \times 10^{-1} \pm 1.41 \times 10^{-2}$	$5.31 \times 10^{-1} \pm 1.88 \times 10^{-2}$
NARMA10	abstract	$1.58 \times 10^{-2} \pm 7.38 \times 10^{-4}$	$1.60 \times 10^{-2} \pm 7.04 \times 10^{-4}$	$1.56 \times 10^{-2} \pm 6.62 \times 10^{-4}$	$1.58 \times 10^{-2} \pm 6.64 \times 10^{-4}$
NARMA10	classical	1.55 ± 3.42	$4.15 \pm 1.05 \times 10^{1}$	3.42 ± 9.81	4.44 ± 8.82
NARMA10	noise	$1.55 \times 10^{-2} \pm 7.33 \times 10^{-4}$	$1.48 \times 10^{-2} \pm 4.89 \times 10^{-4}$	$1.50 \times 10^{-2} \pm 5.83 \times 10^{-4}$	$1.45 \times 10^{-2} \pm 3.55 \times 10^{-4}$
NARMA20	abstract	$2.84 \times 10^{-3} \pm 2.18 \times 10^{-4}$	$2.96 \times 10^{-3} \pm 2.84 \times 10^{-4}$	$3.00 \times 10^{-3} \pm 2.51 \times 10^{-4}$	$2.78 \times 10^{-3} \pm 4.59 \times 10^{-4}$
NARMA20	classical	$1.57 \times 10^{-1} \pm 8.50 \times 10^{-2}$	$1.06 \times 10^{-1} \pm 5.45 \times 10^{-2}$	$1.71 \times 10^{-1} \pm 1.05 \times 10^{-1}$	$1.72 \times 10^{-1} \pm 1.20 \times 10^{-1}$
NARMA20	noise	$2.84 \times 10^{-3} \pm 2.25 \times 10^{-4}$	$2.62 \times 10^{-3} \pm 1.80 \times 10^{-4}$	$2.46 \times 10^{-3} \pm 1.32 \times 10^{-4}$	$2.27 \times 10^{-3} \pm 1.47 \times 10^{-4}$
Dataset	**Model**	**Sensor Shift MSE ± Std Dev@Parameter Value**			
		0.08	0.09	0.10	
Hénon	abstract	$5.34 \times 10^{-1} \pm 7.57 \times 10^{-3}$	$5.32 \times 10^{-1} \pm 5.62 \times 10^{-3}$	$5.34 \times 10^{-1} \pm 1.06 \times 10^{-2}$	
Hénon	classical	2.44 ± 3.27	3.76 ± 3.88	2.78 ± 3.70	
Hénon	noise	$5.30 \times 10^{-1} \pm 1.14 \times 10^{-2}$	$5.27 \times 10^{-1} \pm 9.84 \times 10^{-3}$	$5.23 \times 10^{-1} \pm 7.76 \times 10^{-3}$	
NARMA10	abstract	$1.58 \times 10^{-2} \pm 7.20 \times 10^{-4}$	$1.59 \times 10^{-2} \pm 7.41 \times 10^{-4}$	$1.63 \times 10^{-2} \pm 5.71 \times 10^{-4}$	
NARMA10	classical	$9.65 \times 10^{-1} \pm 1.37$	3.49 ± 9.69	$1.41 \times 10^{1} \pm 2.72 \times 10^{1}$	
NARMA10	noise	$1.46 \times 10^{-2} \pm 3.87 \times 10^{-4}$	$1.44 \times 10^{-2} \pm 2.95 \times 10^{-4}$	$1.44 \times 10^{-2} \pm 3.67 \times 10^{-4}$	
NARMA20	abstract	$2.89 \times 10^{-3} \pm 2.88 \times 10^{-4}$	$3.33 \times 10^{-3} \pm 4.70 \times 10^{-4}$	$3.22 \times 10^{-3} \pm 3.43 \times 10^{-4}$	
NARMA20	classical	$1.22 \times 10^{-1} \pm 5.11 \times 10^{-2}$	$1.66 \times 10^{-1} \pm 8.10 \times 10^{-2}$	$1.16 \times 10^{-1} \pm 5.58 \times 10^{-2}$	
NARMA20	noise	$2.19 \times 10^{-3} \pm 9.98 \times 10^{-5}$	$2.04 \times 10^{-3} \pm 5.94 \times 10^{-5}$	$2.04 \times 10^{-3} \pm 8.45 \times 10^{-5}$	

Table A4. Average MSE for the experiments simulating tolerances in mass placements over 10 experiment runs. The parameter gives the amplitude of the noise added to the initial mass positions.

Dataset	Model	Mass Displacement MSE ± Std Dev@Parameter Value			
		0.00	0.01	0.02	0.03
Hénon	abstract	$5.11 \times 10^{-1} \pm 3.10 \times 10^{-4}$	$5.39 \times 10^{-1} \pm 1.26 \times 10^{-2}$	$5.41 \times 10^{-1} \pm 9.56 \times 10^{-3}$	$5.31 \times 10^{-1} \pm 6.81 \times 10^{-3}$
Hénon	classical	$5.88 \times 10^{-1} \pm 3.17 \times 10^{-2}$	$5.82 \times 10^{-1} \pm 3.08 \times 10^{-2}$	$5.74 \times 10^{-1} \pm 2.52 \times 10^{-2}$	$6.07 \times 10^{-1} \pm 4.02 \times 10^{-2}$
Hénon	noise	$5.83 \times 10^{-1} \pm 2.38 \times 10^{-2}$	1.01 ± 1.10	$6.18 \times 10^{-1} \pm 7.47 \times 10^{-2}$	$5.37 \times 10^{-1} \pm 1.81 \times 10^{-2}$
NARMA10	abstract	$1.38 \times 10^{-2} \pm 3.88 \times 10^{-5}$	$1.48 \times 10^{-2} \pm 4.17 \times 10^{-4}$	$1.49 \times 10^{-2} \pm 2.87 \times 10^{-4}$	$1.56 \times 10^{-2} \pm 6.36 \times 10^{-4}$
NARMA10	classical	$9.99 \times 10^{-2} \pm 4.08 \times 10^{-2}$	$1.06 \times 10^{-1} \pm 4.09 \times 10^{-2}$	$9.32 \times 10^{-2} \pm 2.04 \times 10^{-2}$	$9.97 \times 10^{-2} \pm 3.41 \times 10^{-2}$
NARMA10	noise	$2.35 \times 10^{-1} \pm 1.19 \times 10^{-1}$	$2.15 \times 10^{-2} \pm 2.83 \times 10^{-3}$	$1.66 \times 10^{-2} \pm 1.03 \times 10^{-3}$	$1.55 \times 10^{-2} \pm 6.84 \times 10^{-4}$
NARMA20	abstract	$1.88 \times 10^{-3} \pm 3.67 \times 10^{-5}$	$2.94 \times 10^{-3} \pm 3.89 \times 10^{-4}$	$2.55 \times 10^{-3} \pm 1.75 \times 10^{-4}$	$2.74 \times 10^{-3} \pm 5.86 \times 10^{-4}$
NARMA20	classical	$1.21 \times 10^{-1} \pm 6.16 \times 10^{-2}$	$1.38 \times 10^{-1} \pm 6.91 \times 10^{-2}$	$1.36 \times 10^{-1} \pm 5.19 \times 10^{-2}$	$1.23 \times 10^{-1} \pm 5.56 \times 10^{-2}$
NARMA20	noise	$2.63 \times 10^{-1} \pm 1.43 \times 10^{-1}$	$8.98 \times 10^{-3} \pm 1.97 \times 10^{-3}$	$5.19 \times 10^{-3} \pm 5.42 \times 10^{-4}$	$3.52 \times 10^{-3} \pm 4.68 \times 10^{-4}$

Table A4. *Cont.*

Dataset	Model	Mass Displacement MSE ± Std Dev@Parameter Value			
		0.04	0.05	0.06	0.07
Hénon	abstract	$5.34 \times 10^{-1} \pm 5.45 \times 10^{-3}$	$5.34 \times 10^{-1} \pm 1.00 \times 10^{-2}$	$5.35 \times 10^{-1} \pm 5.70 \times 10^{-3}$	$5.32 \times 10^{-1} \pm 5.11 \times 10^{-3}$
Hénon	classical	$5.86 \times 10^{-1} \pm 2.65 \times 10^{-2}$	$5.85 \times 10^{-1} \pm 3.60 \times 10^{-2}$	$5.98 \times 10^{-1} \pm 3.42 \times 10^{-2}$	$5.93 \times 10^{-1} \pm 5.82 \times 10^{-2}$
Hénon	noise	$5.38 \times 10^{-1} \pm 2.55 \times 10^{-2}$	$5.26 \times 10^{-1} \pm 1.09 \times 10^{-2}$	$5.24 \times 10^{-1} \pm 5.41 \times 10^{-3}$	$5.22 \times 10^{-1} \pm 8.28 \times 10^{-3}$
NARMA10	abstract	$1.59 \times 10^{-2} \pm 8.17 \times 10^{-4}$	$1.58 \times 10^{-2} \pm 7.70 \times 10^{-4}$	$1.60 \times 10^{-2} \pm 7.59 \times 10^{-4}$	$1.55 \times 10^{-2} \pm 2.96 \times 10^{-4}$
NARMA10	classical	$1.48 \times 10^{-1} \pm 4.72 \times 10^{-2}$	$8.14 \times 10^{-2} \pm 2.71 \times 10^{-2}$	$1.19 \times 10^{-1} \pm 5.17 \times 10^{-2}$	$9.44 \times 10^{-2} \pm 2.53 \times 10^{-2}$
NARMA10	noise	$1.50 \times 10^{-2} \pm 5.30 \times 10^{-4}$	$1.49 \times 10^{-2} \pm 7.80 \times 10^{-4}$	$1.44 \times 10^{-2} \pm 3.22 \times 10^{-4}$	$1.43 \times 10^{-2} \pm 3.03 \times 10^{-4}$
NARMA20	abstract	$2.81 \times 10^{-3} \pm 3.76 \times 10^{-4}$	$2.87 \times 10^{-3} \pm 3.02 \times 10^{-4}$	$2.95 \times 10^{-3} \pm 4.10 \times 10^{-4}$	$2.82 \times 10^{-3} \pm 3.37 \times 10^{-4}$
NARMA20	classical	$1.16 \times 10^{-1} \pm 4.46 \times 10^{-2}$	$1.20 \times 10^{-1} \pm 6.07 \times 10^{-2}$	$1.31 \times 10^{-1} \pm 6.77 \times 10^{-2}$	$1.29 \times 10^{-1} \pm 4.15 \times 10^{-2}$
NARMA20	noise	$2.75 \times 10^{-3} \pm 2.23 \times 10^{-4}$	$2.59 \times 10^{-3} \pm 1.90 \times 10^{-4}$	$2.38 \times 10^{-3} \pm 1.45 \times 10^{-4}$	$2.26 \times 10^{-3} \pm 1.82 \times 10^{-4}$
Dataset	**Model**	**Mass Displacement MSE ± Std Dev@Parameter Value**			
		0.08	0.09	0.10	
Hénon	abstract	$5.28 \times 10^{-1} \pm 4.41 \times 10^{-3}$	$5.31 \times 10^{-1} \pm 7.63 \times 10^{-3}$	$5.28 \times 10^{-1} \pm 8.50 \times 10^{-3}$	
Hénon	classical	$5.87 \times 10^{-1} \pm 2.03 \times 10^{-2}$	$5.85 \times 10^{-1} \pm 3.31 \times 10^{-2}$	$5.57 \times 10^{-1} \pm 1.14 \times 10^{-2}$	
Hénon	noise	$5.19 \times 10^{-1} \pm 2.33 \times 10^{-3}$	$5.21 \times 10^{-1} \pm 7.42 \times 10^{-3}$	$5.20 \times 10^{-1} \pm 3.08 \times 10^{-3}$	
NARMA10	abstract	$1.55 \times 10^{-2} \pm 5.72 \times 10^{-4}$	$1.65 \times 10^{-2} \pm 9.19 \times 10^{-4}$	$1.57 \times 10^{-2} \pm 9.65 \times 10^{-4}$	
NARMA10	classical	$1.19 \times 10^{-1} \pm 3.58 \times 10^{-2}$	$1.37 \times 10^{-1} \pm 8.37 \times 10^{-2}$	$1.34 \times 10^{-1} \pm 7.29 \times 10^{-2}$	
NARMA10	noise	$1.42 \times 10^{-2} \pm 2.14 \times 10^{-4}$	$1.40 \times 10^{-2} \pm 4.53 \times 10^{-4}$	$1.39 \times 10^{-2} \pm 2.72 \times 10^{-4}$	
NARMA20	abstract	$2.95 \times 10^{-3} \pm 5.17 \times 10^{-4}$	$2.98 \times 10^{-3} \pm 3.89 \times 10^{-4}$	$2.99 \times 10^{-3} \pm 3.90 \times 10^{-4}$	
NARMA20	classical	$1.03 \times 10^{-1} \pm 3.27 \times 10^{-2}$	$8.87 \times 10^{-2} \pm 3.96 \times 10^{-2}$	$1.61 \times 10^{-1} \pm 7.64 \times 10^{-2}$	
NARMA20	noise	$2.09 \times 10^{-3} \pm 1.06 \times 10^{-4}$	$2.04 \times 10^{-3} \pm 1.03 \times 10^{-4}$	$2.04 \times 10^{-3} \pm 7.46 \times 10^{-5}$	

References

1. Jaeger, H. *The "Echo State" Approach to Analysing and Training Recurrent Neural Networks-with an Erratum Note*; Technical Report; German National Research Center for Information Technology GMD: Bonn, Germany, 2001; Volume 148.
2. Maass, W.; Markram, H. On the computational power of circuits of spiking neurons. *J. Comput. Syst. Sci.* **2004**, *69*, 593–616. [CrossRef]
3. Fernando, C.; Sojakka, S. Pattern Recognition in a Bucket. In Proceedings of the European Conference on Artificial Life, Dortmund, Germany, 14–17 September 2003; Volume 2801, pp. 588–597. [CrossRef]
4. Vandoorne, K.; Dierckx, W.; Schrauwen, B.; Verstraeten, D.; Baets, R.; Bienstman, P.; Campenhout, J. Toward optical signal processing using Photonic Reservoir Computing. *Opt. Express* **2008**, *16*, 11182–92. [CrossRef] [PubMed]
5. Vandoorne, K.; Dambre, J.; Verstraeten, D.; Schrauwen, B.; Bienstman, P. Parallel Reservoir Computing Using Optical Amplifiers. *IEEE Trans. Neural Netw.* **2011**, *22*, 1469–1481. [CrossRef] [PubMed]
6. Nakajima, K.; Hauser, H.; Li, T.; Pfeifer, R. Information processing via physical soft body. *Sci. Rep.* **2015**, *5*, 1–11. [CrossRef] [PubMed]
7. Hauser, H.; Füchslin, R.; Nakajima, K. Morphological Computation: The Body as a Computational Resource. In *Opinions and Outlooks on Morphological Computation*; Hauser, H., Füchslin, R., Pfeifer, R., Eds.; Self-Published: Zürich, Switzerland, 2014; pp. 226–244.
8. Bhovad, P.; Li, S. Physical reservoir computing with origami and its application to robotic crawling. *Sci. Rep.* **2021**, *11*, 1–18. [CrossRef] [PubMed]
9. Hauser, H.; Ijspeert, A.J.; Füchslin, R.M.; Pfeifer, R.; Maass, W. Towards a theoretical foundation for morphological computation with compliant bodies. *Biol. Cybern.* **2011**, *105*, 355–370. [CrossRef] [PubMed]
10. Cousot, P.; Cousot, R. Abstract interpretation. In Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages—POPL '77, Paris, France, 15–17 January 1977; ACM Press: New York, NY, USA, 1977. [CrossRef]
11. Singh, G.; Gehr, T.; Püschel, M.; Vechev, M. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.* **2019**, *3*, 1–30. [CrossRef]
12. Singh, G.; Gehr, T.; Püschel, M.; Vechev, M. Boosting Robustness Certification of Neural Networks. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
13. Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; Vechev, M. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–23 May 2018. [CrossRef]
14. Mirman, M.; Gehr, T.; Vechev, M. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; PMLR: Stockholm Sweden, 2018; Volume 80, pp. 3578–3586.
15. Van Der Hoeven, J. Ball arithmetic. In Proceedings of the Conference Logical Approaches to Barriers in Computing and Complexity, Greifswald, Germany, 17–20 February 2010.

16. Senn, C.W.; Kumazawa, I. Abstract Echo State Networks. In *Artificial Neural Networks in Pattern Recognition*; Schilling, F.P., Stadelmann, T., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 77–88.
17. Coulombe, J.C.; York, M.C.; Sylvestre, J. Computing with networks of nonlinear mechanical oscillators. *PLoS ONE* **2017**, *12*, e0178663. [CrossRef] [PubMed]
18. Provot, X. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. In Proceedings of the Graphics Interface '95, Québec, QC, Canada, 17–19 May 1995; Canadian Human-Computer Communications Society: Toronto, ON, Canada, 1995; pp. 147–154.
19. Urbain, G.; Degrave, J.; Carette, B.; Dambre, J.; Wyffels, F. Morphological Properties of Mass–Spring Networks for Optimal Locomotion Learning. *Front. Neurorobot.* **2017**, *11*, 16. [CrossRef] [PubMed]
20. Murai, A.; Hong, Q.Y.; Yamane, K.; Hodgins, J.K. Dynamic skin deformation simulation using musculoskeletal model and soft tissue dynamics. *Comput. Vis. Media* **2017**, *3*, 49–60. [CrossRef]
21. Johansson, F. Ball Arithmetic as a Tool in Computer Algebra. In *Maple in Mathematics Education and Research;* Gerhard, J., Kotsireas, I., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 334–336.
22. Fieker, C.; Hart, W.; Hofmann, T.; Johansson, F. Nemo/Hecke: Computer Algebra and Number Theory Packages for the Julia Programming Language. In Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation (ISSAC '17), Kaiserslautern, Germany, 25–28 July 2017; ACM: New York, NY, USA, 2017; pp. 157–164. [CrossRef]
23. O'Donoghue, B.; Chu, E.; Parikh, N.; Boyd, S. Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding. *J. Optim. Theory Appl.* **2016**, *169*, 1042–1068. [CrossRef]
24. Bezanson, J.; Edelman, A.; Karpinski, S.; Shah, V.B. Julia: A fresh approach to numerical computing. *SIAM Rev.* **2017**, *59*, 65–98. [CrossRef]
25. Goudarzi, A.; Banda, P.; Lakin, M.R.; Teuscher, C.; Stefanovic, D. A Comparative Study of Reservoir Computing for Temporal Signal Processing. *arXiv* **2014**, arXiv:1401.2224.
26. Hénon, M. *A Two-Dimensional Mapping with a Strange Attractor*; Springer: New York, NY, USA, 1976; Volume 50, pp. 69–77. [CrossRef]