Inferring the dynamics of oscillatory systems using recurrent neural networks

Cite as: Chaos **29**, 063128 (2019); doi: 10.1063/1.5096918
Submitted: 21 March 2019 · Accepted: 6 June 2019 ·
Published Online: 26 June 2019

Rok Cestnik[1,2,a)] and Markus Abel[1,3]

AFFILIATIONS

[1]Department of Physics and Astronomy, University of Potsdam, Karl-Liebknecht-Str. 24/25, 14476 Potsdam-Golm, Germany
[2]Institute for Brain and Behavior Amsterdam and Amsterdam Movement Sciences, Faculty of Behavioural and Movement Sciences, Vrije Universiteit Amsterdam, van der Boechorststraat 9, 1081BT Amsterdam, The Netherlands
[3]Ambrosys GmbH, David-Gilly-Str. 1, 14469 Potsdam, Germany

[a)]**Electronic addresses:** rokcestn@uni-potsdam.de and r.cestnik@vu.nl

ABSTRACT

We investigate the predictive power of recurrent neural networks for oscillatory systems not only on the attractor but in its vicinity as well. For this, we consider systems perturbed by an external force. This allows us to not merely predict the time evolution of the system but also study its dynamical properties, such as bifurcations, dynamical response curves, characteristic exponents, etc. It is shown that they can be effectively estimated even in some regions of the state space where no input data were given. We consider several different oscillatory examples, including self-sustained, excitatory, time-delay, and chaotic systems. Furthermore, with a statistical analysis, we assess the amount of training data required for effective inference for two common recurrent neural network cells, the long short-term memory and the gated recurrent unit.

Inference of most dynamical properties of any system is typically best done with an active experiment, meaning that one has the power to repeatedly manipulate the system state in controlled conditions in order to isolate the desired measure, e.g., car technical inspection, where the examiners manipulate the car into specific situations in order to asses its safety on the road. Often, however, one only has access to passive observations. This can be due to a number of reasons, for example, the system can be very big, like when studying the dynamics of our planet, or the system can be delicate, like when studying the physiology of the human body. The algorithms that attempt to distill dynamical measures from passive observations commonly make assumption on how the observations were collected and typically require very long observations. Imagine, for example, assessing a car's capabilities only from observing routine trips to work. Here, we propose a conceptually simple scheme, relying on the now well established artificial neural networks (ANNs). In particular, we use recurrent neural networks (RNNs) which have established themselves in timeseries forecasting, text generation, etc., and train the networks to mimic the system dynamics, allowing us to then perform an active experiment on the trained model. We test this on several oscillatory systems and measure their characteristic properties, such as bifurcations, dynamical response curves, and characteristic exponents, and compare them to the measures form the original system.

I. INTRODUCTION

Oscillatory systems can be found in all fields of natural science: in optics,[1] electronics,[2] chemistry,[3] biology,[4] climatology, life science, etc. Oscillations are present at all scales, both temporal and spatial, e.g., in biology, from cells like neurons[5] to organs like the heart[6] to oscillations spanning the entire organism such as the circadian rhythm[7] and the menstrual cycle.

Classical modeling of dynamical systems consists of reasoning of the terms involved followed by directly assessing the validity of the model. For low-dimensional systems, this works well; however, the aforementioned examples are all complex, high-dimensional, and coupled to their surrounding, like the brain which consists of many coupled neurons.[8] Other examples include climate models[9] and fluid dynamics, which have been a major driving force for the investigation of periodic motion, synchronization of oscillatory systems,[10] period doubling bifurcations, and chaotic oscillations. Accurate modeling of such systems is hard, but with increasing computer power, existing

methods to infer dynamical systems from measurements are easier to realize.[11–13]

For high-dimensional systems, one has to either measure with many channels or apply embedding methods,[14,15] or commonly both. If a system is truly periodic, then it lives on a one-dimensional manifold and may be in principle modeled by a two-dimensional system of equations. If weakly perturbed, under certain assumptions,[16] the system remains close to the unperturbed orbit. Such perturbations may originate from another oscillator, a network of oscillators, or elsewhere from the environment. However, if the system is close to a bifurcation, perturbations may cause it to undergo dramatic changes in its dynamics. Bifurcations, however, are hard to predict for heuristic models, whereas this is generally easier if equations are known. Under this point of view, previous approaches using symbolic regression methods[12,17,18] proved successful. Heuristic methods such as liquid state machines, echo state networks, or various types of artificial neural networks[19–23] perform very well in predicting dynamical systems. However, few studies are known for particular aspects of oscillatory systems inferred from time series. Here, we investigate several oscillatory models under perturbation, as they may occur in real measurements. Our focus is on the inference of dynamical properties, bifurcation behavior, and chaos, even if not all of the parameter variation is included in the measurement.

In all of the above methods one typically *a priori* assumes a model (or a class of models), sets an optimization criterion (e.g., least squares), and optimizes model parameters or its functional constituents for nonparametric methods. Mathematical aspects are most often left aside, e.g., basic assumptions on the existence of solutions and robustness under perturbation, in particular for heuristic methods. Here, we utilize the widely used artificial neural networks (ANNs). An ANN has several hyperparameters such as the actual topology of the network, the activation function, the learning rate, and is in general very pliable toward many different tasks. Since we consider timeseries, we investigate the capacities of recurrent neural networks[24] (RNNs). Due to loops in their connectivity, they retain past information, i.e., they inherently possess memory, similar to embedding. They tend to be particularly successful in speech recognition,[25] text generation,[26] and machine translation,[27] where a forward-oriented semantic is present. The aim of this study is to evaluate how suited RNNs are for modeling oscillatory systems under the aspect of parameter change and perturbations. In this way, the inferred model of the oscillator can be probed via changing the perturbation signal, effectively allowing the performance of an active experiment.

The article is structured as follows: in Sec. I A, we refer to relevant related works and briefly recall the RNN functioning. In Sec. II A, we introduce the dynamical inference setup and training scheme. We then present numerical tests with example systems in Sec. II B where we compare signal reconstruction and other observables such as the phase response curve[28] (PRC) and the maximum Lyapunov exponent.[29] The different example systems are chosen as representatives of different mechanisms giving rise to oscillatory behavior; specifically, self-sustained and excitatory oscillations, time-delay induced oscillations, and chaos. We continue by presenting numerical tests on data requirement for successful inference in Sec. II C where we compare the inference quality for different lengths of timeseries used for training. We present the methods used in more detail in Sec. III and finally, discuss the novelties, limitations, and generalizations of our approach in Sec. IV.

A. Previous work

In this paragraph, we chronologically go through works related to this paper. In Ref. 30, the author uses a RNN for learning state space trajectories. In Ref. 31, the authors show that any trajectory generated by a finite-dimensional dynamical system can be effectively represented with a neural network. In Ref. 32, the authors model a dynamical system with a perturbation using a RNN. In Ref. 33, the authors use feed-forward neural networks[34] to model dynamical systems. They feed in delayed values of one variable as well as a control parameter as inputs and train the network for one step predictions. The approach works well and they reproduce bifurcation diagrams of several example dynamical systems. In our approach, we train RNNs for one step prediction where the input consists of several time-delayed values of one or more variables as well as an arbitrary number of perturbative signals (we will refer to the perturbative signal inputs as p-inputs). The past values of one or more variables contain information of the topology of the attractor of the complete system according to the Takens' delay embedding theorem.[15] The RNN topology prioritizes more recent values over older ones for the next prediction; therefore, we believe it is more suitable for timeseries prediction and demonstrate its efficiently throughout this paper.

B. Recurrent neural networks

Artificial neural networks is nowadays a relatively broad term as many different network topology classes are commonly used for dealing with different types of problems. The simplest class of ANNs are the feed-forward networks; they have directed connections between subsequent layers without any loops, effectively allowing the information to flow in only one direction—forward. The slightly more general class are the recurrent neural networks (RNNs); they can have loops in their connectivity, which can result in internal state memory. Different RNNs then differ in the fine architecture of the basic cells, the order, and the type of logical operations. In this work, we apply two commonly used cells, the long short-term memory cell[35] (LSTM) and the gated recurrent unit[36] (GRU). LSTM was constructed first in an attempt to deal with long term dependencies and GRU emerged as its faster simplification. Further details on the functioning of different RNN cells can be found in Ref. 37. The software implementation was accomplished with the help of TensorFlow[38] and Keras.[39]

II. RESULTS

A. Inference scheme

Consider a general dynamical system $\dot{\vec{x}}(t) = f(\vec{x}) \in \mathbb{R}^N$ perturbed by an external perturbation $\vec{p}(t) \in \mathbb{R}^N$. Suppose we have measured the timeseries of $n_x \geq 1$ state variables $\vec{x} = (x_1, x_2, \ldots, x_{n_x})$ as well as the n_p timeseries of the perturbation $\vec{p} = (p_1, p_2, p_3, \ldots)$ over a period of time. The question we investigate is if it is possible to recover both the autonomous dynamics of the system $\vec{x}$ and the system's response to the perturbation using RNNs. Without perturbation, we can only recover the dynamics on the attractor, but with a perturbation, the phase space around the attractor is explored and we have a means to infer the neighboring phase space too.

We train the RNN to receive historical values of $\vec{x}(t), \vec{p}(t)$ and return the time-evolved state $\vec{x}(t+\Delta t)$. In practice, this is accomplished by first "unrolling" the network. The RNN at each time step can be represented as a separate copy of the same network, where the recurrent connections have been replaced with regular connections linking every copy with its successor. Then, this chain of networks is truncated and a finite number of "rolls" (network copies) considered. The historical values effectively correspond to a time-delay embedding, allowing the RNN to infer the state of the system. The number of rolls, therefore, corresponds to the dimensionality of the time-delay embedding, although the time steps we use are typically much smaller then delays used in embeddings, making the succesive steps considerably correlated. Nevertheless, we can deduce from Takens' embedding theorem[15] that at least $R > 2M + 1$ rolls have to be considered, where M is the dimensionality of the attractor. The sampling Δt must be smaller than the smallest time scale which occurs in the system (or which we may want to include in our modeling). Heuristically, one can say that the time-resolution Δt should be chosen fine enough to see the details of interest.

Given an appropriate resolution and number of network copies we can begin to "train" our model, i.e., to start a loop for the statistical inference method. At each training step, the network "learns" the possible relation

$$\vec{x}(t), \vec{p}(t) \mapsto \vec{x}(t+\Delta t) \tag{1}$$

using the time instants t, $t-\Delta t$, $t-2\Delta t, \ldots, t-(R-1)\Delta t$. We use a least-squares optimization criterion $\frac{\|\vec{x}_e - \vec{x}\|}{\|\vec{x} - \langle\vec{x}\rangle\|}$ (where $\langle\cdot\rangle$ stands for the mean) to determine quantitatively how well the estimates $\vec{x}_e$ match the true values $\vec{x}$. Hereafter, we use estimated and modeled as synonymous.

B. Examples

We put our scheme to the test on several model systems, including time-delay, excitatory, and chaotic oscillators. The *validation test* consists of comparing the modeled signal with the original when presented with data never seen in training. As important measures of oscillatory systems, we estimate the phase response curve[28] (PRC) and the maximal Lyapunov exponent[29] for comparing the predictive power the model has in a dynamical systems context.

For all examples shown in this paper, we use a network with 1 hidden layer of 32 nodes and 36 rolls. We use tanh activation for all but the output layer, where we use linear activation so that a continuous signal can be produced. There are two common cells used in RNN: the long short-term memory cell[35] (LSTM) and the gated recurrent unit[36] (GRU). We tested both for the systems in this study; as a result, we found that GRU performed poorly, hence all results shown are for LSTM models, cf. Sec. II C for a comparison of the two cells. To generate the data we first simulate the perturbation signal using the stochastic Euler-Maruyama integration scheme and then integrate the dynamical equations with fourth order Runge Kutta. We use a sufficiently small time step and then resample the signals to an appropriately lower time resolution to create the network training data. The resolution is chosen such that 36 points (the number of considered historical values R) correspond to 1 natural period of the oscillator. In the case of chaotic oscillators, this was computed as the average period, in the case of excitatory systems the time needed to return from the excited state to the fixed point was used.

1. Roessler oscillator—Phase response curve, bifurcation diagram, and Lyapunov exponents

For our first test, we use the Roessler system,[40] because it exhibits many different regimes, i.e., simple periodic oscillations, higher period oscillations, and chaos, by varying just one parameter b, cf. Fig. 2(a) for the bifurcation diagram. The corresponding equations, including the perturbation p, read

$$\begin{aligned} \dot{x} &= -y - z, \\ \dot{y} &= x + ay, \\ \dot{z} &= b + z(x - c) + p(t), \end{aligned} \tag{2}$$

with parameters $a = 0.2$ and $c = 5.7$. To explore the phase space, we can vary b through a constant term in the perturbation $p(t)$. For the first test, we set $b = 2.0$ such that the system has a simple attractive periodic orbit. In the following, we use a stochastic perturbation, generated by an Ornstein-Uhlenbeck[41] process:

$$\dot{q} = -q/\tau + \epsilon\sqrt{2/\tau}\xi(t), \tag{3}$$

where ξ is Gaussian white noise $\langle\xi(t)\xi(t')\rangle = \delta(t-t')$ and $\epsilon = 0.5$ and $\tau = 5.0$ are the amplitude and correlation time of q. The stochastic differential equation is integrated with the Euler-Maruyama method to obtain noise with exponentially decaying correlation: $\langle q(t)q(t')\rangle = \epsilon^2 e^{-(t-t')/\tau}$.

Now, we set $p(t) = q(t)$ and feed both the signal $x(t)$ and the perturbation signal into the network as described in Sec. II A. The timeseries length corresponds to 1000 natural periods, which is presented to the network during 500 training epoch in batches of 100 time points with resolution $\Delta t = 0.17$ (the time step used for the integration is significantly smaller). The network is trained using stochastic gradient descent[42] with learning rate 0.005.

The network learns to reproduce the dynamics to a mean deviation of 2.5×10^{-2} (for the time window in Fig. 1) such that the reproduced signal is visually indistinguishable from the one generated with Eq. (2). This holds true for both the perturbed signal—where $p(t)$ is fed to both the network and the equations, as well as for the unperturbed signal—where $p(t) = 0$ is used.

Can we use the inferred network for more than just mimicking a signal, e.g., to study dynamical regimes? We want to study this scenario in probing the network for dynamical responses to stimuli. Since the system in question is a self-sustained oscillator, it is natural to estimate its PRC, cf. Sec. III B. The comparison of the estimate obtained from the RNN and the true PRC is displayed in Fig. 1. The coincidence is very good up to mean deviation of 0.1 in the entire phase range $[0, 2\pi)$. Indeed, this can be an effective method of inferring the PRC from data, cf. Refs. 43–45.

We perform another test with the Roessler oscillator, this time testing the power of the network to reproduce the system across several dynamical regimes. For this, we use $b = 0$ and a strong and varied p-input that considerably explores the state space,

$$p(t) = \frac{1}{2}\exp(q(t)), \tag{4}$$

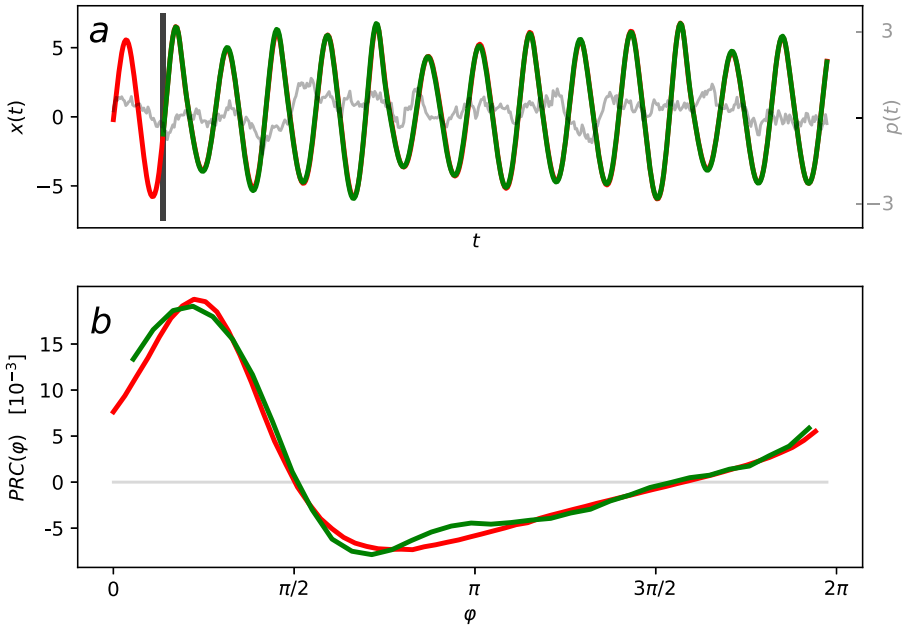

FIG. 1. (a) The training signal generated with Eq. (2) in red and the RNN reproduced signal in green. Both have the same p-input, depicted with gray (scaled for being visually comparable to the signal). The vertical black line marks the beginning of the forecast. (b) The true phase response curve of system (2) in red and the one inferred from the RNN in green.

where $q(t)$ is the p-input described in Eq. (3). It yields a process with a log-normal distribution: $P(p) \sim \frac{1}{p} \exp\left(-2(\log(p) + \log(2))^2\right)$. Such p-input spans a wide range of values, effectively introducing different regimes of our system to the network, see Fig. 2(a) for the p-input probability distribution with respect to b bifurcation (grey shaded region in the background). The idea is that the network then effectively learns to mimic the regimes corresponding to different values of b, which we can invoke via the offset of the p-input $p(t)$. For this study, we use a longer timeseries corresponding to 10 000 natural periods, which is presented to the network during 1000 training epochs.

As a result, we find that the network reproduces the signal of the system perturbed by Eq. (4) well. Furthermore, we can estimate the bifurcation diagram from the network by feeding it different values of constant p-input, effectively setting the parameter b of the model (2) and observing the stationary signals, see Fig. 2(a). In the value range of the p-input (4), the diagram obtained from the RNN matches the true one closely. It reproduces simple oscillatory regimes, chaotic regimes, and the period doubling bifurcation. Throughout the range of b, the natural frequency (average frequency in the case of chaos) matches the true one closely, with mean deviation of 5×10^{-2}.

In chaotic regimes, the maximum Lyapunov exponent[29] is an important measure as it quantifies the divergence of nearby trajectories in time. We estimate it from the RNN and plot it against the true values, see Fig. 2(b). This is accomplished by long time observation of the evolution of two nearby states, while rescaling their difference to prevent them from diverging far from each other, see Sec. III C for further details. This can be an effective method for inferring the Lyapunov exponent from data, cf. Refs. 46 and 47.

Now we go even one step further and test the prediction of the RNN when presented with an input outside the range of trained values. We train two networks on slightly modified p-inputs,

$$p(t) = \frac{1}{2} \exp(\pm|q(t)|). \tag{5}$$

This effectively splits the probability distribution of the p-input (4) in two at the value $b = 0.5$. One network is trained only on values smaller than 0.5 and the other only on larger ones. Then, we perform the same prediction analysis as in the previous test, estimating the bifurcation diagram and the Lyapunov exponents across the full range of $b \in (0, 2)$. It stands to reason that the predictions in regimes far from those presented during training will have little to do with the original system, but nevertheless it is surprising just how much can be deciphered from them. For example, in Fig. 3(a), a period doubling bifurcation occurs outside of the trained regime, as it does in the original system (although the critical values are shifted), and throughout the entire test range, the system remains oscillatory (it does not settle to a fixed point). Not all features are reflected however; for instance, in Fig. 3(c) in the chaotic regime outside of the p-input range, the period-3 window is not observed.

2. FitzHugh-Nagumo oscillator–Example of an excitable system

In the following two examples, we want to study the power of RNN for two systems with different origin and dynamical behavior of oscillations. As a first important class, we investigate an excitatory system, namely, the FitzHugh-Nagumo oscillator,[48]

$$\begin{aligned} \dot{x} &= x - x^3/3 - y + I_0 + p(t), \\ \dot{y} &= \sigma(x + a - by), \end{aligned} \tag{6}$$

where parameters are $\sigma = 0.1$, $a = 0.7$, $b = 0.8$, and $I_0 = 0.25$. For the p-input we use, as once before, $p(t) = q(t)$, described by Eq. (3),

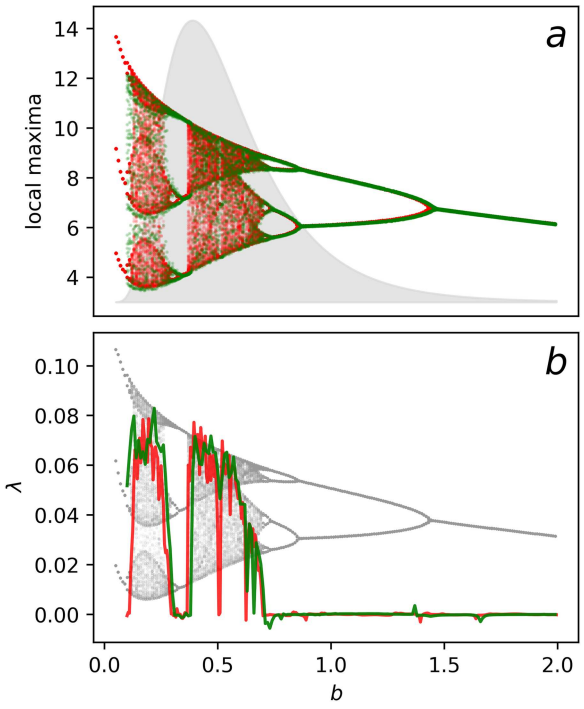

FIG. 2. (a) The bifurcation diagram of system (2) in red and the one inferred from the RNN in green. The probability density of the p-input is underlaid in gray (scaled for being visually comparable to the diagram). (b) The Lyapunov exponents of system (2) in red and the ones inferred from the RNN in green. The true bifurcation diagram is underlaid in gray (scaled for being visually comparable). Note that the bifurcation diagram as well as the Lyapunov exponent range were reproduced with a single RNN trained on correlated noise p-input, Eq. (4).

with $\epsilon = 0.05$ and $\tau = 25.0$. The RNN is trained on timeseries comprising of 1000 spikes, over 500 training epochs. The time resolution is $\Delta t = 1$. For the excitatory oscillations we compare how well does the model reproduce a spike train when presented with a novel p-input realization, see Fig. 4(a). Furthermore, we estimate the spiking frequency with respect to the input current I_0 (the p-input) and compare it to the true one, see Fig. 4(b). For low input currents I_0, system (6) is quiescent, i.e., it does not fire and remains close to its fixed point. When I_0 is increased, a bifurcation occurs, a limit cycle is born and the system begins to spike regularly. The corresponding first-order phase transition is clearly inferred from the RNN, with the critical value of the input accurately predicted up to the order 10^{-3}. In addition, the estimated frequency values match the true ones closely, with mean deviation 10^{-2}.

3. Mackey-Glass equation—Example of a delay system

For our final study, we briefly report on the RNN results for the Mackey-Glass equation,[49] as a representative of time-delay systems,

$$\dot{x} = a\frac{x_\theta}{1+x_\theta^n} - bx + p(t), \tag{7}$$

where x_θ represents the time-delayed variable $x(t-\theta)$, $a = 2$, $b = 1$, $n = 8$, and the time-delay $\theta = 2$. In this parameter regime, the equation yields a stable limit cycle with a period-2 orbit, see Fig. 5. We use p-input $p(t) = q(t)$, Eq. (3), with $\epsilon = 0.005$ and $\tau = 1.0$. The length of timeseries corresponds to 5000 natural periods, over 500 epochs in batches of 100 time points with resolution $\Delta t = 0.15$. The dynamics is well reproduced with a mean deviation of 5×10^{-2}. Intuitively, a RNN seems to be suited well for modeling delay equations, since it has inherent delay. We conclude that for this important model class RNNs work well.

C. Amount of data and noise study

Any good statistics-based study includes a section on the dependence of the result on the amount of data provided and the sensitivity to noise—we do so in the following paragraphs. We present only results for the Roessler oscillator, Eq. (2). We train independent RNN models with different lengths of timeseries. We vary the amount of data supplied to the RNN in the following way: we keep the product of the timeseries length and number of epochs constant, thereby always introducing the same number of data points to the network (500 000), i.e., we change the number of occurrences of the same points. The sampling rate is kept constant. We test the range from 15 to 1000 periods and measure the error of the PRC and signal, see Fig. 6. For each set of parameters, 100 models are trained and evaluated. The PRC error is evaluated as the L_2 norm of the difference between the true and the reconstructed curve,

$$\int_0^{2\pi} \left(PRC_{RNN}(\varphi) - PRC_{TRUE}(\varphi)\right)^2 d\varphi,$$

and similarly for the signal error,

$$\int_0^{\Delta} \left(x_{RNN}(t) - x_{TRUE}(t)\right)^2 dt,$$

where we further have to determine over what interval we evaluate it, Δ.

The results of this evaluation are shown in Fig. 6. Here, we also demonstrate the difference between LSTM and GRU cell types, underlining our previous remark on the poor results for GRU. For the LSTM cell, the PRC error is on average rather good and it clearly shows a dependence on the amount of data provided, approximately at 100 periods worth of data, the error saturates around the value 0.1, see Fig. 6(a). In the case of GRU, the PRC error is large and does not seem to improve with greater amounts of data, Fig. 6(b). That is not to say that GRU intrinsically cannot perform this task, it might just require a larger network to achieve the same effect—recall that we use the same number of nodes throughout this work. GRU was designed as a clever faster simplification of the LSTM cell. It merges the hidden cell state into the regular cell state as well as merging several logical operations into fewer ones.[37] These simplifications are reasoned by its developers[36] but apparently noticeably impair the cell in performing our particular task.

The error of the signal undoubtedly should depend on the interval Δ over which it is evaluated. Even with a near perfect model, the small errors build up and after a long time, the true and reconstructed

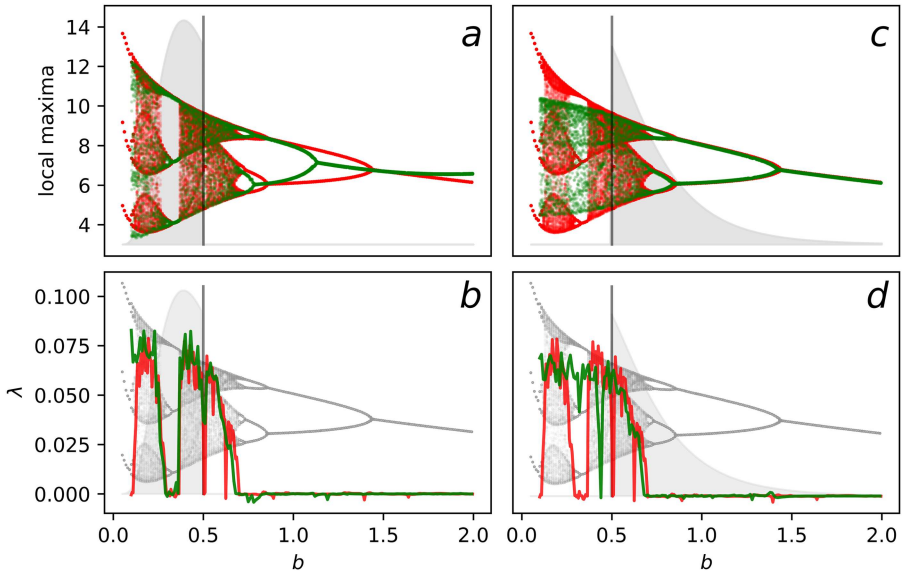

FIG. 3. The bifurcation diagram and maximal Lyapunov exponent for system (2) perturbed by Eq. (5). On the left [(a) and (b)], the negative exponent $(-)$ is used, therefore limiting the p-input values below 0.5, and on the right, [(c) and (d)] the positive exponent $(+)$ is used, limiting the p-input values to above 0.5, see the underlaid probability distributions in gray. Therefore, each side has a range of parameter values b that the RNN has not been presented with during training. In all subplots, the true values of system (2) are depicted with red and the RNN inferred ones with green.

signals become incoherent, which means that with increasing Δ the error should grow. We see that for both cells in Figs. 6(c) and 6(d), although on average, the GRU signal errors are significantly larger. As with the PRC error, the signal error decreases with the amount of data.

Now for the robustness of the inference against measurement noise. We only present a basic study where we consider the Roessler system, Eq. (2), with p-input $p(t) = q(t)$, Eq. (3). We fix $b = 0.6$ which corresponds to a chaotic regime. Then, we add to each time point a random uncorrelated Gaussian number with mean 0 and standard deviation 1 to represent strong measurement noise and train the network on the noisy signals. We introduce 10 000 average periods worth of training data over 500 epochs. The network effectively extracts the relevant dynamics and reproduces the attractor well, see Fig. 7.

III. METHODS

In this section, we specify the methods we used to evaluate the properties of oscillatory systems. For each property, we write how we computed it from the equations as well as how we computed it from the RNN.

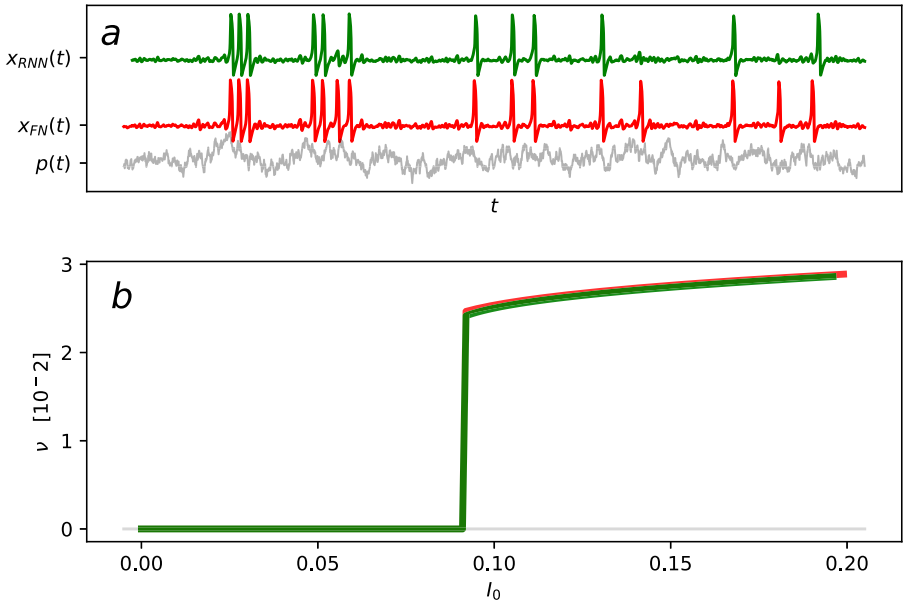

FIG. 4. (a) The training signal generated with Eq. (6) in red and the RNN reproduced signal in green (shifted up for distinction). Both have the same p-input realization, depicted with gray (scaled and shifted for being visually comparable to the signal). (b) The true spiking rate of system (6) in red and the one inferred from the RNN in green.

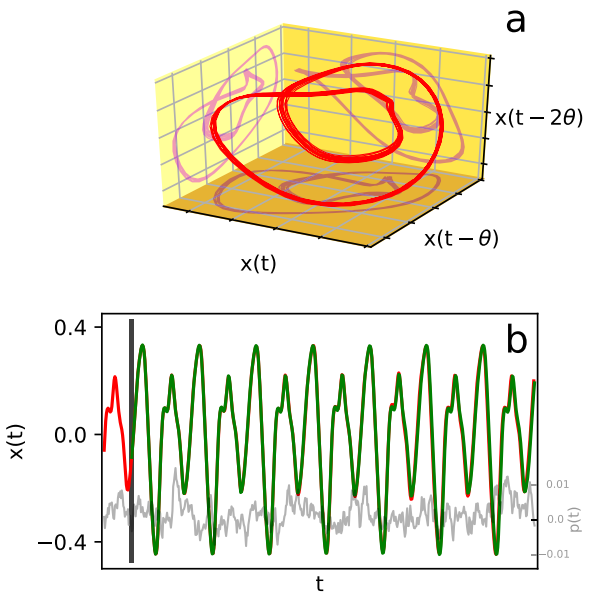

FIG. 5. (a) Delay embedded trajectory of system (7) in red, its two-dimensional projections in purple. For the chosen set of parameters, the system has a stable limit cycle, the variation is due to the p-input. (b) The signal generated with Eq. (2) in red and the RNN reproduced signal in green. Both have the same p-input realization, depicted with gray (scaled and shifted for being visually comparable to the signal). The vertical black line marks the beginning of the forecast.

A. Natural period estimation

The period is measured as the time between two successive signal-threshold crossings from bellow when the system is unperturbed. From equations, the time of crossing is accurately estimated using the Hénon trick.[50] When estimating from a network, a linear interpolation from a point before and after the threshold crossing is used.

B. Phase response curve estimation

Firstly, the natural period T_0 has to be accurately estimated, see Sec. III A. Then, the system in question is weakly and instantaneously perturbed at particular phases φ^*, i.e., at times $t^* = \frac{\varphi^*}{2\pi} T_0$ after the beginning of a period, $p(t) = \epsilon\delta(t - t^*)$. Then, the evoked phase shift is evaluated as

$$Z(\varphi^*) = 2\pi \frac{nT_0 - \sum_{i=1}^{n} T_i}{\epsilon T_0}, \tag{8}$$

where T_1 is the period in which the perturbation arrives and $T_2, T_3, T_4, \ldots$ the periods that follow. n counts how many periods we wait to evaluate the shift, and since we are looking for the asymptotic shift, n should be big enough that the PRC does not depend on it; in this paper, we used $n = 5$.

In the case of the network, the time for inputting perturbations is discrete and the best we can do is input perturbation $\epsilon/\Delta t$, where Δt is the time increment between two consecutive points in the unrolled RNN.

C. Maximal Lyapunov exponent estimation

For computing the exponents from the true system we use the standard technique, since we have the dynamical equations.

To estimate the exponent from the RNN, a different approach is needed. Suppose we have access to all the variables of the original system $\vec{x} = (x_1, x_1, \ldots, x_{n_x})$. In such a case, the intuitive method can be used as follows:

(1) simulate a trajectory $\vec{x}$ for a long time so it settles to the attractor,

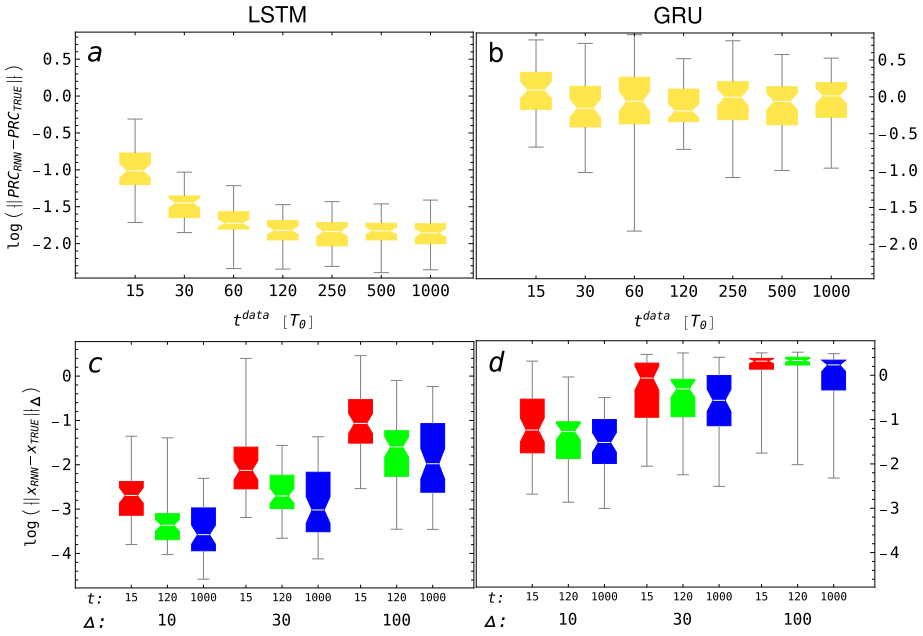

FIG. 6. Comparison of data requirement for two different cells, LSTM left [(a) and (c)] and GRU right [(b) and (d)]. In the top plots [(a) and (b)], the error of the inferred PRC with respect to the length of data provided, t^{data} (in units of the natural period T_0). In the bottom plots [(c) and (d)], the error of the reproduced signal with respect to the length of data provided t^{data} (15, 120, and 1000) for three different forecast lengths, Δ (10, 30, and 100).

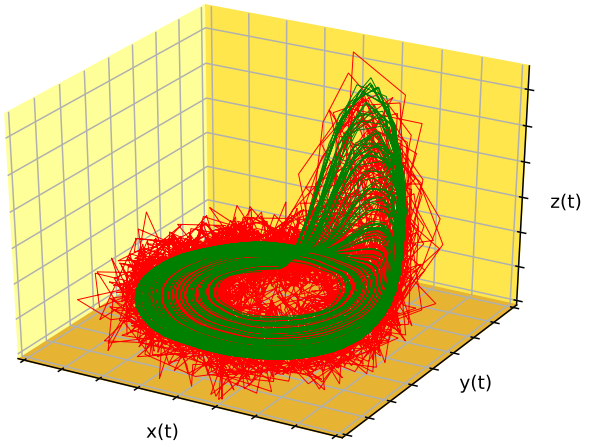

FIG. 7. The training data in red and the RNN reproduced attractor in green.

(2) start a new trajectory $\vec{x^\dagger} = \vec{x} + \vec{p}$ with a small arbitrary perturbation $\|\vec{p}\| = \delta x$ and evolve both for a short time δt,
(3) evaluate the deviation $\Delta = \|\vec{x^\dagger} - \vec{x}\|$,
(4) renormalize the second trajectory for the deviation to have the same amplitude as the one we started with $\vec{x^\dagger} = \vec{x} + \delta x * (\vec{x^\dagger} - \vec{x})/\|\vec{x^\dagger} - \vec{x}\|$, but keep the direction of the perturbation the same so that the maximal exponents takes over in the course of several repetitions,
(5) loop to step 3 and average the quantity $\frac{1}{\delta t}\log(\Delta/\delta x)$ which tends toward the maximal Lyapunov exponent.

Here, $\|\cdot\|$ stands for the L_2 norm: $\|\vec{v}\| = \left(\sum_i v_i^2\right)^{1/2}$.

The more general approach concerns cases where we do not have access to all the variables but only a few, in the extreme case only one x_1—common when dealing with real data. In such case, the state of the system has to be characterized with several historical values, $\vec{w} = (x_1(t), x_1(t-\Delta t), x_1(t-2\Delta t), \ldots)$, and then the algorithm above can be used as before. This is the case in Sec. II B 1.

IV. DISCUSSION

The aim of this study was to test the predictive capacity of recurrent neural network applied to different oscillatory systems. One problem common to all oscillators is that the state space collapses to a low-dimensional manifold, and therefore any reconstruction only allows the prediction on that inertial manifold. However, if perturbed we can achieve a much better understanding of the system around its attractor. We even can follow and predict a bifurcation outside the range of values which were provided by the data. This is a notable fact and it may as well work for other methods, like symbolic regression.

We have applied the method to a range of oscillatory systems, from a time-delay oscillator with a period-2 orbit (Sec. II B 3, Fig. 5), to an excitatory system (Sec. II B 2, Fig. 4), and finally, a chaotic attractor (Sec. II B 1, Fig. 2). We demonstrate that the trained neural networks can be probed for dynamical responses. As typical characteristics of oscillatory systems, we estimated the phase response curve[28] (PRC), the spiking rate, and the maximal Lyapunov exponent.[29] Other quantities, such as the Floquet exponent,[51] the amplitude response, the isochronal structure,[52] synchronization properties,[53] etc., could be estimated in a similar way. We can say that RNNs provide an effective way of estimating oscillatory properties from timeseries, cf. Refs. 43–47. Our way of applying them to data is novel and should be explored further not only in the context of oscillations. It is, for example, not clear how well RNNs perform for scaling systems like turbulence.

Since the success of each machine learning method depends on data, we performed a statistical analysis on how the size of the training data set influences the inference. The training data required for an effective inference proved to be reasonably small, with only a few 10 periods sufficing for reliably estimating the mentioned dynamical systems' quantities. We used two popular recurrent network cells in our study: the long short-term memory cell[35] and the gated recurrent unit.[36] The latter proved to be inferior in performing these tasks (at least for the same network size). We also tested the inference with the addition of measurement noise and it proved to be robust, see Sec. II C.

Along with this publication, we (RC) published a Python software package, OscillatorSnap,[54] available on the Python Package Index (PyPI) as oscillator_snap. It contains most of the examples shown here as well as an array of high level functions for analyzing oscillatory systems, such as a function that computes the phase response curve or the maximal Lyapunov exponent from dynamical equations as well as from a trained RNN model.

ACKNOWLEDGMENTS

We thank Nicolas Deschle, Bastian Pietras, Thomas Kreuz, as well as the employees of Ambrosys, Markus, Franz, Tino, Maxim, Thomas, and Greta for useful discussions. This work was funded by the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie Grant Agreement No. 642563 (COSMOS).

REFERENCES

[1] M. E. Marhic, *Fiber Optical Parametric Amplifiers, Oscillators and Related Devices* (Cambridge University Press, Cambridge, 2008), Chap. 8.
[2] M. Tooley, *Electronic Circuits: Fundamentals and Applications* (Newnes, Oxford, 2002), Chap. 9.
[3] I. R. Epstein and J. A. Pojman, *An Introduction to Nonlinear Chemical Dynamics, Oscillations, Waves, Patterns, and Chaos* (Oxford University Press, Oxford, 1998), Chap. 8.
[4] A. T. Winfree, *The Geometry of Biological Time* (Springer, Berlin, 1980).
[5] K. M. Stiefel and G. B. Ermentrout, J. Neurophysiol. **116**, 2950–2960 (2016).
[6] A. D. A. Babloyantz, Biol. Cybern. **58**, 203 (1988).
[7] R. Foster and L. Kreitzman, *Circadian Rhythms: A Very Short Introduction* (Oxford University Press, Oxford, 2017).
[8] G. Buzsáki, *Rhythms of the Brain* (Oxford University Press, Oxford, 2006).
[9] H. A. Dijkstra, *Nonlinear Climate Dynamics* (Cambridge University Press, 2013).
[10] A. Pikovsky, M. Rosenblum, and J. Kurths, *Synchronization: A Universal Concept in Nonlinear Sciences* (Cambridge University Press, 2001).
[11] H. U. Voss, P. Kolodner, M. Abel, and J. Kurths, Phys. Rev. Lett. **83**, 3422 (1999).
[12] H. Voss, M. J. Bünner, and M. Abel, Phys. Rev. E **57**, 2820 (1998).
[13] M. Abel, Int. J. Bifurcat. Chaos **14**, 2027 (2004).

[14]H. Whitney, Ann. Math. **37**, 645 (1936).
[15]F. Takens, in *Dynamical Systems and Turbulence*, Lecture Notes in Mathematics Vol. 898, edited by D. Rand and L. S. Young (Springer, Berlin, 1981).
[16]J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Applied Mathematical Sciences (Springer, New York, 2002).
[17]M. Quade, M. Abel, K. Shafi, R. K. Niven, and B. R. Noack, Phys. Rev. E **94**, 012214 (2016).
[18]M. Schmidt and H. Lipson, Science **324**, 81 (2009).
[19]J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Phys. Rev. Lett. **120**, 024102 (2018).
[20]J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, Chaos **27**, 121102 (2017).
[21]Z. Lu, B. R. Hunt, and E. Ott, Chaos **29**, 061104 (2018).
[22]R. Z. U. Parlitz, Chaos **28**, 043118 (2018).
[23]I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016), see http://www.deeplearningbook.org.
[24]Z. C. Lipton, CoRR abs/1506.00019 (2015).
[25]H. Sak, A. Senior, and F. Beaufays, in *INTERSPEECH-2014* (International Speech Communication Association, 2014), Vol. 338.
[26]I. Sutskever, J. Martens, and G. E. Hinton, in *ICML'11 Proceedings of the 28th International Conference on International Conference on Machine Learning* (Omnipress, 2011), pp. 1017–1024.
[27]K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phase representations using RNN encoder-decoder for statistical machine translation," preprint arXiv:1406.1078 (2014).
[28]C. C. Canavier, Scholarpedia **1**, 1332 (2006).
[29]A. Politi, Scholarpedia **8**, 2722 (2013).
[30]B. A. Pearlmutter, Neural Comput. **1**, 263 (1989).
[31]K. Funahashi and Y. Nakamura, Neural Netw. **6**, 801 (1993).
[32]C. A. Bailer-Jones, D. J. MacKay, and P. J. Withers, Network Comput. Neural Syst. **9**, 531 (1998).
[33]R. Falahian, M. M. Dastjerdi, M. Molaie, S. Jafari, and S. Gharibyadeh, Nonlinear Dyn. **81**, 1951 (2015).
[34]D. Svozil, V. Kvasnička, and J. Pospichal, Chemometr. Intell. Lab. Syst. **39**, 43 (1997).
[35]S. Hochreiter and J. Schmidhuber, Neural Comput. **9**, 1735 (1997).
[36]K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, e-print arXiv:1409.1259 (2014).
[37]C. Olah, see https://colah.github.io/posts/2015-08-Understanding-LSTMs/ for "Understanding LSTM Networks" (2015).
[38]M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale Machine Learning on Heterogeneous Systems" (2015), software available from https://tensorflow.org.
[39]See https://github.com/keras-team/keras for "K. team: Keras" (2019).
[40]O. E. Roessler, Phys. Lett. **57A**, 397–398 (1976).
[41]G. E. Uhlenbeck and L. S. Ornstein, Phys. Rev. **36**, 823 (1930).
[42]J. Peters, Scholarpedia **5**, 3698 (2010).
[43]K. Ota, M. Nomura, and T. Aoyagi, Phys. Rev. Lett. **103**, 024101 (2009).
[44]T. Imai, K. Ota, and T. Aoyagi, J. Phys. Soc. Jpn. **86**, 024009 (2017).
[45]R. Cestnik and M. Rosenblum, Sci. Rep. **8**, 13606 (2018).
[46]A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, Physica D **16**, 285 (1985).
[47]M. T. Rosenstein, J. J. Collins, and C. J. D. Luca, Physica D **65**, 117 (1993).
[48]E. M. Izhikevich and R. FitzHugh, Scholarpedia **1**, 1349 (2006).
[49]L. Glass and M. C. Mackey, Ann. N.Y. Acad. Sci. **316**, 214 (1979).
[50]M. Henon, Phys. D Nonlinear Phenom. **5**, 412 (1982).
[51]P. Kuchment, Floquet Theory for Partial Differential Equations, Operator Theory: Advances and Applications (Birkhäuser, 1993), Vol. 60.
[52]K. Josic, E. T. Shea-Brown, and J. Moehlis, Scholarpedia **1**, 1361 (2006).
[53]A. Pikovsky, M. Rosenblum, and J. Kurths, *Synchronization: A Universal Concept in Nonlinear Sciences.* (Cambridge University Press, Cambridge, 2001).
[54]R. Cestnik, see https://github.com/rokcestnik/oscillator_snap for "Oscillator Snap" (2019).